Efficient 6D Pose Estimation for Augmented Reality through Hybrid Methods and Domain-Invariant Matching

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

doctor rerum naturalium
(Dr. rer. nat.)
im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
Humboldt-Universität zu Berlin

von
M. Sc. Niklas Gard

Gutachter/innen:
1. Prof. Dr. Peter Eisert
2. Prof. Dr. Sebastian Knorr
3. Prof. Dr. Didier Stricker

Tag der mündlichen Prüfung: 3. November 2025

Abstract

In *augmented reality* (AR) applications, knowledge about the camera pose and the positions of known objects relative to the camera is crucial for seamlessly aligning virtual content with the real world. Recently, deep neural networks have significantly improved the accuracy of pose estimation, but matching computer graphics data with 2D camera images remains a challenge. This challenge, known as the *reality gap* or *domain gap*, arises from differences in representation and simplified assumptions in synthetic image generation, making direct comparison difficult.

This work proposes advanced methods for six-dimensional (6D) pose estimation and refinement of cameras and objects, optimizing the integration of digital 3D models into the real world. It investigates how *domain-invariant features*—specifically object silhouettes and their 3D shape edges, persistent semantic layouts, and actively projected known textures—enhance the matching between synthetic and real-world data, thereby supporting accurate pose prediction.

The thesis contributes to bridging the reality gap by integrating techniques from both learning-based and non-learning-based approaches to improve overall performance and robustness in AR systems. It also facilitates effective generalization to new target geometries by leveraging domain-invariant features, aiming to reduce dependence on costly training data generation.

First, a framework for model-based 6D pose refinement is developed, with a particular focus on projector-based tracking and *spatial augmented reality* (SAR). It combines *analysis-by-synthesis* with object and projection contours for precise 6D tracking. By integrating projection distortions on real objects and utilizing edge images, it bridges the reality gap in SAR scenarios where existing methods fail.

A novel method for 6D pose estimation enhances the capacity of a convolutional neural network (CNN) for multiple object classes. It utilizes the class and shape information from semantic mask predictions to increase the accuracy, while employing a single-stage architecture to estimate 2D-3D correspondences for multiple objects in a single pass.

Building on that, a hybrid approach enhances robustness through 3D geometry-based mismatch detection by integrating CNN-based pose estimation with local pose refinement. This results in improved reliability and simplified synthetic training for real-time AR, particularly for geometrically similar objects.

Finally, SPVLoc is introduced for 6D camera pose estimation in indoor scenes, ensuring pose determination relative to *unseen* semantic indoor models. A novel matching technique represents the environment as rendered semantic panoramas, aligns the image content within these references, and determines the global pose based on ranked references. The method improves localization accuracy over existing scene-independent methods, estimating more degrees of freedom of the pose.

Extensive qualitative and quantitative experiments evaluate these methodologies, highlighting their advantages for AR. In addition, the thesis presents practical application examples, demonstrating how the developed methods and insights can be implemented in real-world scenarios.

Zusammenfassung

In *Augmented Reality* (AR)-Anwendungen sind Kenntnisse über die Kamerapose und die Positionen bekannter Objekte relativ zur Kamera entscheidend, um virtuelle Inhalte nahtlos in die reale Umgebung zu integrieren. In den letzten Jahren haben tiefe neuronale Netze die Genauigkeit der Posenschätzung erheblich verbessert, dennoch bleibt der Abgleich computergenerierter Referenzdaten mit Kamerabildern eine Herausforderung. Diese Herausforderung, bekannt als *Reality Gap* oder *Domain Gap*, entsteht durch Unterschiede in der Repräsentation und vereinfachte Annahmen bei der synthetischen Bilderzeugung, die einen direkten Vergleich erschweren.

Diese Arbeit entwickelt Methoden zur sechs-dimensionalen (6D) Posenschätzung von Kameras und Objekten, um die Integration digitaler 3D-Modelle in die reale Welt zu optimieren. Sie untersucht, wie *domäneninvariante* Merkmale – insbesondere Objektsilhouetten, 3D-Geometriekanten, persistente semantische Layouts und aktiv projizierte Texturen – das Abgleichen von synthetischen und realen Daten sowie die Posenschätzung unterstützen. Die Dissertation trägt zur Überbrückung des Reality Gaps bei, indem sie lernbasierte und nicht-lernbasierte Ansätze integriert, um die Effektivität und Robustheit von AR-Systemen zu steigern. Zudem wird gezeigt, dass domäneninvariante Merkmale eine Generalisierung auf neue Zielgeometrien erleichtern und die Abhängigkeit von kostspieliger Trainingsdatengenerierung reduzieren.

Zunächst wird ein Framework zur modellbasierten 6D-Posenverfeinerung mit Fokus auf projektorbasiertes Tracking und *Spatial Augmented Reality* (SAR) entwickelt. Es kombiniert *Analyse durch Synthese* mit Objekt- und Projektionskonturen für ein präzises 6D-Tracking. Durch die Berücksichtigung von Projektionsverzerrungen auf realen Objekten und der Nutzung von Kantenbildern überbrückt es den Reality Gap in SAR-Szenarien, in denen bestehende Verfahren scheitern.

Ein neues Verfahren zur 6D-Posenschätzung erhöht die Kapazität eines *Convolutional Neural Networks* (CNN) für mehrere Objektklassen. Es nutzt Klassen- und Forminformationen aus semantischen Maskenvorhersagen zur Erhöhung der Genauigkeit und setzt eine einstufige Architektur ein, um 2D-3D-Korrespondenzen für mehrere Objekte in einem einzigen Durchgang zu schätzen.

Ein hybrider Ansatz verbessert die Robustheit durch 3D-Geometrie-basierte Fehlererkennung, indem die CNN-basierte Posenschätzung mit lokaler Verfeinerung kombiniert wird. Dies vereinfacht das synthetische Training und steigert die Zuverlässigkeit in Echtzeit-AR-Anwendungen, besonders bei geometrisch ähnlichen Objekten.

Schließlich wird das SPVLoc-Verfahren zur 6D-Kameraposenschätzung in Innenräumen eingeführt, das die Pose relativ zu *ungesehenen* semantischen Umgebungsmodellen bestimmen kann. Ein neuartiges Matching stellt die Umgebung als gerenderte semantische Panoramabilder dar, verortet den Bildinhalt innerhalb dieser Referenzen und schätzt die globale Pose basierend auf den geschätzten Zuordnungen. Das Verfahren verbessert die Lokalisierungsgenauigkeit im Vergleich zu bestehenden szenenunabhängigen Methoden und schätzt mehr Freiheitsgrade der Pose.

Umfassende qualitative und quantitative Experimente bewerten die Methoden und heben ihre Vorteile für AR hervor. Darüber hinaus werden praktische Beispiele präsentiert, die zeigen, wie die entwickelten Methoden und Erkenntnisse in realen Szenarien angewendet werden können.

Danksagung

Die Fertigstellung dieser Arbeit wäre ohne die Unterstützung zahlreicher Personen nicht möglich gewesen.

Mein besonderer Dank gilt meinem Doktorvater Prof. Dr. Peter Eisert, der mich während des gesamten Forschungsprozesses nicht nur fachlich exzellent begleitet, sondern mich auch stets ermutigt und durch wertvolle Hinweise unterstützt hat.

Weiterhin danke ich den externen Gutachtern Prof. Dr. Sebastian Knorr und Prof. Dr. Didier Stricker für die sorgfältige Begutachtung dieser Arbeit und die investierte Zeit.

Ebenso danke ich meiner Gruppenleiterin Dr. Anna Hilsmann für ihre Unterstützung und Beratung bei der Fertigstellung der Publikationen, die inhaltlich in diese Arbeit eingeflossen sind. Ihr Feedback und ihre Anregungen haben die Publikationen bereichert und weiterentwickelt.

Mein Dank gilt zudem allen aktuellen und ehemaligen Mitgliedern der Computer Vision & Graphics Gruppe am Fraunhofer HHI. Die zahlreichen inspirierenden Diskussionen, der offene Austausch von Ideen und die freundliche, unterstützende Atmosphäre haben meine Arbeit entscheidend geprägt und meine Zeit in der Gruppe besonders wertvoll gemacht.

Darüber hinaus danke ich den Kolleginnen und Kollegen der Abteilung Vision & Imaging Technologies sowie allen Beteiligten der Forschungsprojekte EASY COHMO, DigitalTWIN und BIMKIT. Ihre Mitwirkung ermöglichte es, die in dieser Arbeit entwickelten Verfahren in Augmented-Reality-Demonstratoren zu integrieren und interaktiv erlebbar zu machen. Besonderer Dank gilt hierbei Aleixo Cambeiro Barreiro, Angelina Aziz, Paul Chojecki, David Przewozny, Detlef Runde, Dominykas Strazdas, Sylvain Renault, Fabian Schmid, Gergey Matl, Philipp Kopriwa und Jean-Claude Rosenthal.

Ein weiterer Dank gilt meiner Familie, insbesondere meinen Eltern, für ihren beständigen Rückhalt, ihr Interesse an meiner Arbeit und ihr Vertrauen in mich während meiner Promotionszeit. Zuletzt danke ich Stefanie für ihren unermüdlichen Beistand, ihr Verständnis in schwierigen Phasen und dafür, dass sie stets an mich geglaubt hat.

Contents

List of Figures

List of Tables

Nomenclature

Basic Notations

$\mathbb{I}_n$	$n \times n$ Identity matrix
x	Scalar
X	Set of scalars or vectors
$\mathbf{x}$	Vector
$\mathbf{X}$	Matrix
$\mathbf{x}^T$, $\mathbf{X}^T$	Transpose of a vector or matrix
$\mathbf{X}^{-1}$	Inverse of a matrix
$\lvert.\rvert$	L1 norm (sum of absolute values of elements)
$\lVert.\rVert$	L2 norm (Euclidean norm, square root of sum of squares)
$\pi(.)$	Perspective division: divides by the last coordinate.
$\mathrm{tr}(.)$	Trace of a matrix: The sum of its diagonal elements.

Coordinates of a 2D point or vector are denoted as $\mathbf{p} = (p_x, p_y)$, and the coordinates of a 3D point or vector are denoted as $\mathbf{p} = (p_x, p_y, p_z)$. Similarly, 2D coordinates can be written as $\mathbf{x} = (x, y)$.

Elements of higher dimensional vectors, matrices or tensors are denoted by subscripts:

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix} a_{1,1} & \dots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \dots & a_{n,m} \end{bmatrix} \qquad \mathbf{T} = \begin{bmatrix} t_{1,1,1} & \dots & t_{1,m,p} \\ \vdots & \ddots & \vdots \\ t_{n,1,p} & \dots & t_{n,m,p} \end{bmatrix} \qquad \dots$$

A vector $\mathbf{x} \in \mathbb{R}^n$ has elements x_i, a matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ has elements $a_{i,j}$, and a tensor $\mathbf{T} \in \mathbb{R}^{n \times m \times p}$ has elements $t_{i,j,k}$.

Common Quantities

$\mathbf{I}$	Image. A 2D image with width w and height h.
$\mathbf{K}$	Intrinsic camera matrix, 3×3
$\mathcal{L}$	Loss function
$\mathbf{P}$	Pose of a camera or an object (translation and rotation)

$\Delta\mathbf{P}$	Offset in pose, including translation ($\Delta\mathbf{t}$) and rotation ($\Delta\mathbf{R}$)
$\mathbf{R}$	Rotation matrix, 3×3
$\mathbf{t}$	Translation vector

Acronyms

2DP	2D Projection Metric
ADD/S	Average Distance of the Closest Points (*symmetric/asymmetric objects*)
AI	Artificial Intelligence
AR	Augmented Reality
AR	Average Recall (*used in the context of BOP Challenge, page 73, ff.)*
BN	Batch Normalization
BOP	Benchmark for 6D Object Pose Estimation
CNN	Convolutional Neural Network
CIN	Conditional Instance Normalization
CLADE	Class Adaptive (De)normalization
DKR	Differentiable Keypoint Regression
FoV	Field of View
fps	frames per second
gt	ground truth
HB	HomebrewedDB
IRLS	Iteratively Reweighted Least Squares
LM	LINEMOD
LM-O	Occluded LINEMOD
LS	Least Squares
MLP	Multi-layer Perceptron
MSPD	Maximum Symmetry-Aware Projection Distance
MSSD	Maximum Symmetry-Aware Surface Distance
RMSE	Root Mean Square Error
NN	Nearest Neighbor
PnP	Perspective-n-Points
pbr	physically-based rendering
RANSAC	Random Sample Consensus
RGB	Red Green Blue
RGB-D	Red Green Blue and Depth
S3D	Structured3D
SAR	Spatial Augmented Reality
VSD	Visible Surface Discrepancy
ZInD	Zillow Indoor

1 Introduction

Augmented reality (AR) has the potential to significantly enhance our everyday lives by enriching our visual perception with information beyond what is naturally visible. While robotic automation is often about automating or replacing specific human tasks, AR aims to enhance human capabilities by enriching their interaction with the real world. Camera sensors capture the environment, enabling dynamic and adaptive extraction of knowledge about the surroundings. This allows AR to overlay additional information onto the real world, with the level of detail increasing as more information is extracted. Recent advances in AR technologies are driving adoption across many domains [MRTMFH+23]. AR can help us navigate our surroundings [BHK19, KCU+22], simplify complex work tasks [BTM15, EM20, EGIB23], and make daily activities more intuitive and efficient [MGDRGLC+20, JN21, MYY22].

New information can be provided in various ways. For example, overlays in *spatial augmented reality* (SAR) applications are projected precisely onto objects in the environment with a projector. With glasses-based systems, the information is placed directly in the field of vision, while with display-based systems, the overlay is shown on the screen of a smartphone or similar device [EGIB23]. Regardless of the visualization, a deep understanding of the semantic and spatial structure of a scene is essential to spatially align real and virtual elements. However, achieving stable alignment, reliable localization, and fast visual feedback remains a challenge for broad acceptance [MRTMFH+23].

This work aims to enhance the connection between digital 3D models of the environment or specific objects, and the real world using 2D camera images. The link between a digital model and an external coordinate system is established by a six-dimensional (6D) pose, defined by position and orientation in 3D space. Both the camera pose relative to an indoor scene and the poses of 3D objects relative to the camera are considered, focusing on finding robust representations for reliable matching while ensuring efficiency for AR applications.

Pose estimation in AR benefits from the tremendous progress in artificial intelligence (AI) in recent years. Historically, camera localization with respect to 2D images relied on printed fiducial markers. Nowadays, AI-based approaches that use large quantities of annotated training data to learn the appearance of an object or scene from different viewpoints have gained popularity [FZH+22]. The capturing and manual annotation of data are very costly and time-consuming. Furthermore, state-of-the-art algorithms rely on retraining or fine-tuning a *convolutional neural network* (CNN) for each new object or scene, adding to the complexity [BHW+24, CCPB24]. A primary goal of this work is to eliminate the need to acquire real-world data before integrating 6D pose estimation for new objects or environments into AR systems.

A key challenge in achieving this is ensuring that digital images created from 3D models are comparable to real-world photographs—an issue that also affects non-learning-based pose estimation. By introducing comparable representations that bridge the gap between synthetic and real domains, pose estimation performance and robustness are enhanced. This thesis contributes to this in two main ways: First, it introduces new methods for object pose estimation and tracking that efficiently utilize synthetic data. These methods emphasize domain invariances to address challenges in learning-based and non-learning-based approaches, enhancing effectiveness by combining them in hybrid methods. Second, it introduces a scene-agnostic neural network that uses a 3D model of an environment as input. This network uses semantic information to generalize camera pose estimation to new environments, without needing scene-specific training. The next section details the research problems, the domain invariances utilized, and their interrelationships.

1.1 Domain Invariant Features and Synthetic Training

Establishing the extrinsic pose between a camera and an object requires comparing a 3D model to a 2D image. Correspondences between the model and the image allow the estimation of the camera's position and orientation This can be achieved using either hand-crafted features or a CNN [HFZL20].

A significant challenge arises from the domain differences between real and synthetic data, particularly when aligning these sources for accurate analysis. For example, *analysis-by-synthesis*-based object tracking requires an image-to-image comparison and real-time adaptation of a rendered image to the camera view to recover the object's pose. However, parameters such as lighting, shadows, reflections, physical effects, and occlusions are typically unknown. Even when known, real-time image renderers struggle to replicate the complexity and noise captured by real image sensors. While point-feature detectors are invariant to illumination changes, untextured objects often lack meaningful features [HFZL20]. Moreover, 3D models from digital planning, including those of industrial objects and building structures, often lack textures, making it difficult to create synthetic images for direct comparison with camera images [AKW19, GRP22]. Domain-invariant features aim to bridge the gap between rendered representations and camera images.

A primary image feature defined by an object's shape is its silhouette, a binary mask indicating which pixels in an image belong to the 2D projection of the object with a given pose. The silhouette's outline, or contour, is essential for recovering the object's pose. The visible part of the silhouette, consisting of all pixels belonging to the object, is referred to as its instance mask. Additionally, geometry edges, representing intensity changes due to surface discontinuities, are crucial for deriving the object's pose. Transforming the camera image into these representations while rendering corresponding projections from the 3D model facilitates accurate localization of the model within the scene.

Furthermore, in SAR scenarios, an external signal provides information about the object's pose or movement. When a new texture is projected onto an object using a rigid projector-camera system, movements of the object cause characteristic distortions in the projection on its surface, which become visible in the camera image. The edges of the projected texture serve as a spatial reference and remain domain-invariant—appearing consistently in both camera and computer-generated images—making them useful for tracking movement. Unfortunately, assumptions about the movement of rigid objects do not apply to the projections, causing traditional contour-based trackers to fail. However, these distortions can still be used for tracking if their relationship to the object's movement in 3D space is modeled directly [ZSW13, KH19], though this has only been applied to 2D targets. This thesis develops a tracking system that integrates projection distortions into analysis-by-synthesis frameworks. It introduces the theoretical foundation for estimating 3D object motion from these distortions, overcoming prior limitations.

Training a CNN to estimate the pose of a given object requires large amounts of labeled image data. While synthetic data generation offers an unlimited supply of annotated examples, it introduces the *reality gap* or *domain gap*. This refers to the performance drop when a network trained on synthetic images is tested on real images, primarily due to the differences between simulated and real environments. Domain randomization [TFR+17] narrows this gap by maximizing variability in the simulated training data—randomizing numerous parameters of the simulated environment. Models trained on this data perceive real-world data as just another variation of the training data, improving generalization to real-world scenarios without requiring additional training [TFR+17, SMD+18].

With advancements in *physically based rendering*, near-photorealistic synthetic images are now frequently employed [TTB18, ZZL+20, HSL+24]. However, generating a believable simulation requires detailed knowledge of the scene parameters of the target environment. Additionally, randomizing factors such as illumination and object arrangements remain essential to prevent the model from overfitting to the simulated data. Tools for generating randomized annotated training data [TTM+18, DSW+19] are commonly used in pose estimation tasks. While scene randomization and physically based rendering mitigate the reality gap, it remains a significant challenge. The best results are achieved when real images are incorporated during training [SHL+23].

Synthetic training data is essential when training a CNN to estimate the poses of multiple objects in an image simultaneously. Occluded objects present a major challenge for the network and also complicate manual annotation. A common workaround, training object-specific networks and using a bounding box detector, improves accuracy but also increases runtime, which is problematic for real-time AR performance [BHW+24]. Object masks offer a consistent feature across both synthetic and real images, provide pose information, and facilitate clear object separation. This thesis introduces a CNN architecture that estimates and integrates semantic object masks to enhance domain invariance and improve multi-object handling in a single real-time pass.

Building on this, the edges of 3D shapes further contribute to domain invariance, particularly for human-made objects. While geometrically similar objects may be difficult for a CNN to distinguish, the 2D projection of the 3D geometry is likely to match the image edges for only one candidate exactly. Iterative pose refinement using the 3D model provides an opportunity for verification of the detection. This process refines the initial pose estimate, thereby relaxing the realism requirements for synthetic data and narrowing the domain gap. This thesis presents a real-time capable pipeline where detection and refinement complement each other, pursuing a reliable AR backbone with temporal consistency.

AI-based methods for object recognition or pose estimation of *unseen* objects remain relatively rare compared to object-specific approaches. However, they have recently attracted research interest for their greater flexibility [LSY+24]. An object-independent CNN incorporates the object's appearance—typically as annotated example views for RGB-based approaches—and learns *cross-domain matching*, such as aligning deep features from synthetic 3D renderings with those from an input photo [MGGL21].

Cross-domain matching is also relevant for model-based camera localization, where the goal is to localize an image within a 3D environment model rather than an object in an image. This work focuses on indoor scenes, treating the entire scene model as the "object" to be localized. If the model is derived from captured images or point clouds, its creation requires considerable effort, substantial data storage capacity, and an on-site visit. If an abstract model, such as a BIM model from digital planning, is available, it offers a data-efficient alternative for localization. Synthetic views of the model can be rendered with known poses, enabling the training of a pose regression model [AKW19]. However, creating detailed 3D representations of a scene that minimize domain differences during rendering is costly and often impractical. In both cases, models trained on scene-specific data may struggle to generalize to new environments and can be overly sensitive to changes within the scene [HJRSP21].

Focusing on invariant semantic information during matching helps to narrow the domain gap [MKB+22]. In indoor environments, elements like walls, doors, and windows are fixed, and the scene's structure is a spatial reconfiguration of these components. A model trained to match semantic renderings that highlight only essential layout information with image content naturally focuses on invariances. Training such a model on a large number of scenes enables it to generalize well to new environments, as all appearance variations of semantic elements must match the same class.

So far, scene-agnostic camera localization in indoor environments has not been proposed for 6D camera pose estimation and 3D reference models, and this thesis addresses that gap. The proposed method introduces a highly efficient matching, leveraging the advantages of synthetic data to simulate various camera parameters. By learning to match synthetic omnidirectional semantic panorama images with perspective camera images, only a few references are required to capture a scene.

To summarize, the thesis explores three aspects of pose estimation: local pose estimation (optimizing the pose relative to a starting pose), global object pose estimation (localizing an object relative to the camera without prior pose knowledge), and global camera pose estimation (localizing a camera relative to a scene with limited or no prior pose knowledge). The common thread among all contributions is the effort to bridge the domain gap between synthetic and real data by utilizing domain invariances and model knowledge. Section 1.2 provides a detailed description of each contribution.

Applications The technologies developed in this thesis have broad applications in AR tasks. The projector-distortion-based pose estimation is especially useful when precise projection onto a moving component is required. For instance, information projected onto the surface of workpieces at stationary workbenches can reduce mental load and decrease error rates [UGM⁺18]. It also helps establish a visual link between digital models and real-world objects, enabling intuitive interfaces [CSP⁺23]. The proposed method uses a simple, markerless hardware setup, making SAR workstation configuration and operation more efficient and accessible compared to other 6D tracking approaches.

In dynamic environments such as construction sites, head-mounted displays can assist with assembly and construction tasks, potentially reducing task completion time compared to traditional manuals [HF10]. However, user acceptance remains a challenge [ME20], as reliable tracking is crucial for creating immersive and user-friendly AR systems. The proposed pipeline enhances AR stability by incorporating multi-object 6D pose estimation and tracking. This combination ensures the robust discrimination of similar objects amid dynamic movements, occlusions, and reappearances. Potential applications include construction [KHKA22], assembly state detection [SRM⁺19, LSR⁺20], and industrial processes [EGIB23].

While stable (re)localization is already integrated into widely used hardware [VRCP17], image-based camera localization in previously unseen 3D scenes has traditionally been limited to scenarios with restricted degrees of freedom [MKB⁺22], or specialized camera types, such as omnidirectional cameras [HJRSP21]. The developed algorithm links photos with a digital 3D building model in a scene-independent manner. This benefits self-localization in unfamiliar environments, for example, with a mobile app. Eliminating the need for scene-specific training data enhances the overall usability of such localization systems. Furthermore, 6D localization enables the transfer of visible objects or details from the images into the model, thereby facilitating building digitalization [AGK⁺25].

1.2 Contributions

The contributions of this thesis focus on effectively overlaying and aligning 3D models with 2D images, while utilizing domain-crossing information. They are divided into four distinct parts, corresponding to Chapters 3, 4, 5, and 6, respectively:

- A method for **model-based 6D object tracking** with special focus on **projector-based tracking**. The real-time capable method integrates optical flow with object and texture contours to achieve precise 6D object tracking. A robust edge detection process is employed to determine contours that match those of the rendered object, effectively bridging the domain gap between the rendering and the camera image. While model-based 6D tracking is a well-covered topic, the core contribution to this field lies in the incorporation of spatial augmented reality projection in the tracking process. Typically, model-based trackers encounter challenges when a new texture is projected onto a real-world object, as the projection *floats* over the object rather than moving with it. A motion model is derived to reconstruct object movement from local projection distortions. The analysis-by-synthesis tracking adapts the pose using computer graphic projection simulation to recreate visible distortions iteratively. The method can be applied in several variants, including motion models specifically for planar objects, arbitrary meshes, and extrinsic pose refinement between projector and camera.

- A novel approach for **multi-object 6D pose estimation with semantic guidance**. CASAPose (**C**lass-**A**daptive and **S**emantic-**A**ware **Pose** estimation) is a single-stage architecture that determines 2D-3D correspondences for pose estimation of multiple different objects in RGB images in a single pass, unlike other methods that require separate networks per object and multiple inferences per image. The approach leverages the output of a semantic segmentation decoder as class-adaptive local guidance for a keypoint recognition decoder via class-adaptive denormalization (CLADE). Both decoders share the same backbone, making the method fast and memory-efficient. Experiments show that incorporating a small set of object-specific parameters through CLADE increases the multi-object capacity of a pose estimation CNN. Additionally, the estimated segmentation serves as shape-based guidance through segmentation-aware convolutions and upsampling operations, improving focus within the object mask and reducing interference from occluding objects. A novel differentiable 2D keypoint estimation method further enhances the accuracy of 2D-3D correspondences, benefiting from the segmentation-aware operations. The network is trained only on synthetic images and outperforms several recent methods on real data. Furthermore, it is shown that the introduced operations help narrow the domain gap between real and synthetic data.

- A robust linkage of **tracking and detection** focused on **synthetic training and object similarity**. The semantically guided feature-point decoder for object classification and rough pose estimation is integrated with local pose refinement in a combined pipeline that effectively addresses the weaknesses of both local tracking and global pose estimation. Drift is mitigated by automatic mismatch detection and CNN-based reinitialization. The domain gap, when training pose estimation with synthetic data, is narrowed by the local edge-based pose refinement that explicitly exploits known object geometry. Detection mismatches caused by geometric ambiguities are filtered through a geometric edge-based pose verification. For continuous movements, the sole use of local refinement reduces pose mismatches due to geometric ambiguities or occlusions. The system addresses real-time AR scenarios, where the robustness of detection, accurate pose estimation, and temporal stability are of high importance. An evaluation of the entire tracking pipeline demonstrates the benefits of the combined approach in terms of robustness and efficiency. Experiments on a challenging set of non-textured, similar objects show improved pose estimation accuracy compared to the baseline method.

- A new method for **6D camera pose estimation within semantic indoor models** focused on **scene independence**. SPVLoc (**S**emantic-**P**anoramic **V**iewport **Loc**alization) is a global indoor localization method that determines the 6D camera pose of an RGB image with respect to an untextured 3D reference model, which includes approximate structural information about room shapes, along with door and window annotations. The core contribution is scene-independent pose estimation, whereas previous methods that regress the 6D pose require scene-specific retraining and data acquisition. Previous scene-independent methods, however, were confined to 3D localization (2D position and 1D orientation). The experiments illustrate superior localization accuracy compared to the state-of-the-art scene-independent method, concurrently estimating more degrees of freedom. The algorithmic innovation is a matching method that identifies which region within a rendered semantic panoramic image is visible in a camera image. Matches are scored and ranked, facilitating the identification of the most probable match within large sets of reference panoramas encompassing extensive spaces. In contrast to the methods for estimating the object pose, the relative pose between the reference panorama and the target image is regressed directly. In order to achieve scene independence and to close the gap between camera images and semantic renderings, large datasets with thousands of environments are used during training.

1.3 Overview

The dissertation is structured as follows:

Related Work Chapter 2 includes a literature review on model-based real-time pose estimation (Sec. 2.1), 6D deep learning-based pose estimation of known objects (Sec. 2.2), and global camera localization within 3D models (Sec. 2.3).

Enhancing Model-Based 6D Tracking for Projector-Camera Systems Chapter 3 develops a framework for analysis-by-synthesis-based pose estimation for model-based tracking in videos [GHE22b] and with projector-camera systems [GE18, GHE19]. The mathematical foundation is established in Sec. 3.1, then used in a model for contour-based pose estimation, including both the derivation of motion as well as implementation details (Sec. 3.2). The method is then extended to account for projection distortions in its motion model, deriving 6D motion from the optical flow of the projection in Sec. 3.3. This innovation is also the main focus of the experimental evaluation in Sec. 3.4.

Class-Adaptive and Semantic-Aware Deep Pose Estimation Chapter 4 proposes a CNN for multi-object pose estimation incorporating semantic masks [GHE22a]. The approach introduces three key innovations: object-adaptive local weights (Sec. 4.1.1), a semantically guided decoder (Sec. 4.1.2), and a differentiable keypoint regression (Sec. 4.1.3). The resulting CASAPose architecture (Sec. 4.2) is then evaluated in various ablation studies and compared with synthetically trained state-of-the-art methods (Sec. 4.3).

Linking Model-based Refinement to CNN Architectures Chapter 5 links the results from the previous two chapters by combining pose estimation and model-based tracking to form a robust and efficient system [GHE22b]. It shows how to spot false detections and identify pose drift with a set of rules for geometric pose validation (Sec. 5.1). The experiment section (Sec. 5.2) demonstrates the improved ability to distinguish similar objects and the benefits of the combined pipeline. An outlook (Sec. 5.3) elaborates on making the local estimation a deep learning component.

Semantically Guided Model-based Camera Localization Chapter 6 introduces a method for global pose estimation against unknown models at inference time [GHE24], with a focus on indoor camera localization (Sec. 6.1). The SPVLoc method combines matching photos with rendered semantic panorama images (Sec. 6.2.2) and relative pose regression (Sec. 6.2.3) for this task. Numerous experiments evaluate component efficiency, robustness, and limitations, along with comparisons to state-of-the-art methods (Sec. 6.3).

Selected Applications Chapter 7 presents real-world AR applications using the proposed algorithms. These applications include weld inspection with SAR support (Sec. 7.1), construction assistance applying real-time multi-object tracking (Sec. 7.2), and building digitalization using global camera localization (Sec. 7.3).

Conclusion Finally, Chapter 8 closes this thesis and offers starting points for future work and ideas for possible extensions based on the solutions presented.

1.4 Research Publications

In accordance with Paragraph 7, Article 5 of the doctorate regulations of the Faculty of Mathematics and Natural Sciences at Humboldt University of Berlin, parts of this thesis have been presented at the following international conferences, workshops, and journals:

- **SPVLoc: Semantic Panoramic Viewport Matching for 6D Camera Localization in Unseen Environments**
  Niklas Gard, Anna Hilsmann, Peter Eisert
  European Conference on Computer Vision (ECCV), 2024 *(Oral Presentation)* [GHE24]
- **CASAPose: Class-Adaptive and Semantic-Aware Multi-Object Pose Estimation**
  Niklas Gard, Anna Hilsmann, Peter Eisert
  British Machine Vision Conference (BMVC), 2022 *(Spotlight Paper)* [GHE22a]
- **Combining Local and Global Pose Estimation for Precise Tracking of Similar Objects**
  Niklas Gard, Anna Hilsmann, Peter Eisert
  International Conference on Computer Vision Theory and Applications (VISAPP), 2022 [GHE22b]
- **Projection Distortion-based Object Tracking in Shader Lamp Scenarios**
  Niklas Gard, Anna Hilsmann, Peter Eisert
  IEEE Transactions on Visualization and Computer Graphics (TVCG), 2019 [GHE19]
- **Markerless Closed-Loop Projection Plane Tracking for Mobile Projector-Camera Systems**
  Niklas Gard, Peter Eisert
  IEEE International Conference on Image Processing (ICIP), 2018 [GE18]

These publications form the foundation of this thesis, with the author of this dissertation being the first author of all listed papers. This thesis incorporates these works, providing advanced analyses of the methods along with enhancements, updated results, and extended applications where necessary.

Additionally, the methods developed in this thesis were applied in collaborative systems, as detailed in articles in *MDPI Sensors* [CSP+23] and *Journal of Computing in Civil Engineering* [AGK+25], as well as in a conference paper at *Forum Bauinformatik* [GCB23], highlighting their broader applicability.

The author also contributed to publications on related topics, including 2D-3D similarity learning via domain randomization (ICIP 2021) [JGHE21], structured light using a system consisting of a projector and an event camera (CVPR Workshop 2023) [MGB+23], and medical instrument tracking (SPIE Medical Imaging 2019) [GRJ+19], though these are not directly related to 6D pose estimation.

2 Related Work

This chapter provides an overview of the key research areas and approaches relevant to this thesis, focusing on pose estimation techniques that align monocular RGB or grayscale images with rigid 3D models. It begins with a discussion of model-based tracking approaches in Sec. 2.1, which are object-independent due to their reliance on classical optimization techniques, such as geometric alignment and contour matching.

Section 2.2 then delves into 6D pose estimation, where state-of-the-art methods often integrate object-specific information into convolutional neural networks (CNNs), introducing model dependence and limitations in multi-object scenarios.

Lastly, Sec. 2.3 addresses global pose estimation of a camera in indoor environments. It explores advancements made by CNN-based approaches while also highlighting the ongoing challenge of achieving full scene or model independence. The section discusses both the progress and limitations of current methods.

2.1 Model-based Real-time 6D Tracking

Model-based 6D tracking refers to determining the pose (translation and rotation) of a moving rigid object in real-time, starting from an initial estimate and using its 3D model. Tracking incorporates temporal information to ensure the pose stability required for dynamic vision applications, such as AR, robotics, and industrial automation, where accurate pose estimation enables seamless interaction between virtual and physical environments. Over the years, various approaches have been developed to improve the efficiency, accuracy, and robustness of real-time 6D tracking systems. This section reviews state-of-the-art methods for model-based real-time 6D tracking, beginning with model-based frame-to-frame techniques (Sec. 2.1.1). It then expands to tracking in projector-camera systems, where advanced strategies are required to manage the interference of projector light during object tracking (Sec. 2.1.2).

2.1.1 Model-based Frame-To-Frame Tracking

Despite significant progress in using CNNs for pose estimation, state-of-the-art results in model-based frame-to-frame tracking continued to rely on non-learning-based methods until recently [SPS+22, TLZQ22]. It was only in 2023 that the first algorithm incorporated a learning component into the classical approach to enhance robustness [WYZ+23]. In the case of potentially non-textured objects and image-based tracking, available approaches can be roughly categorized into two types: edge-based and region-based methods.

Region-based methods utilize image statistics to separate an object from the background and optimize the pose to maximize discrimination. They explicitly or implicitly involve semantic segmentation of the object using shape prior knowledge. Early region-based methods such as Rosenhahn et al. [RBW07] were computationally very complex. They first develop a contour using a zero-level set function to segment the object and then back-project each point on the contour to rays, optimizing the pose to satisfy tangency constraints between the 3D object and these rays. PWP3D [PR12] improves efficiency by using a parallelizable pixel-wise posterior formulation [BR08], which incorporates color histogram appearance models for foreground and background. This approach allows for a direct comparison between the 3D object's silhouette and the 2D image. Instead of evolving the contour over time in an unconstrained space, PWP3D refines the contour by differentiating with respect to pose parameters. Later extensions, such as those by Tjaden et al. [TSSC18] and Zhong et al. [ZZ19], use temporary local color histograms to enhance robustness. However, region-based methods are most effective for objects that are distinct from the background and often struggle with objects that have similar colors to the background [SZZ+21].

Edges or contours are suitable visual cues for tracking non-textured objects. Based on the RAPID algorithm [HS90], 2D-3D correspondences are searched on scanlines perpendicular to the object contour. Extensions filter those correspondences based on the contour orientation [HZSQ20], global or local color histograms, [SPP+13, WWZ+15, HZSQ20], or consider multiple hypotheses per scanline [TLZQ22]. Tian et al. [TLZQ22] observe that the commonly used coarse-to-fine search to handle large displacements is less effective for 3D rotations. They introduce a non-local search focusing on out-of-plane rotations and an efficient local pose optimization method with long precomputed search lines for translation and in-plane rotation, enabling the handling of very large pose offsets. Stoiber et al. [SPS+20, SPS+22] adopt the idea of scanlines, which are referred to as *correspondence lines* in their otherwise region-based approach. By sparsely sampling the image around the projected contour, they significantly improve efficiency compared to other region-based methods. Their Gaussian-conforming sparse probabilistic model ensures quick convergence, and the smoothed step function models both local and global uncertainty [SPS+22]. Other edge-based algorithms do not explicitly use point-to-point correspondences but instead minimize a pixel-based distance metric directly on the intensity image [WZQ19, DJW+20].

Similarly, analysis-by-synthesis-based methods [SHE17, ZLZ17] aim to synthetically recreate the camera image using a rendered representation and minimize the image distance through motion compensation with respect to the optical-flow constraint [HS81, LK81]. A comparable representation of real and synthetic images must be found, for example, through illumination estimation using the textured mesh [ZLZ17] or by explicitly modeling scene or image parameters, such as motion blur [SHE17].

A recent learning-based approach to contour tracking is DeepAC [WYZ$^+$23]. This method uses a network to predict the true boundary locations on scanlines perpendicular to the projected object's contour. The network predicts gradients that facilitate quick convergence of an optimization-based pose estimation. A lightweight backbone encoder-decoder network followed by just a few convolutional layers increases the robustness of the *traditional* algorithm while still maintaining real-time performance.

This thesis presents a novel approach to pose estimation by combining edge-based tracking with analysis-by-synthesis. The target object is rendered, and edges are extracted from the camera image with guidance from the rendered image. This helps to achieve the goal to adapt model-based tracking for projector camera systems. The rendered simulation can incorporate a simulation of the texture projection onto the target object, and use it to find corresponding edges in the camera-image for the pose offset computations.

Additionally, while the presented methods demonstrate significant progress in pose tracking, they treat the pose refinement in isolation from the detection process. This thesis explores how model-based tracking can enhance real-time deep learning-based pose estimation (Sec. 2.2) by incorporating geometric refinement for pose improvement, validation, and temporal consistency. The tracker alternates between RAPID-based silhouette alignment [HS90, HZSQ20] to bridge large pose offsets and analysis-by-synthesis to fine-tune the pose using the object's inner edges.

2.1.2 Registration for Projector-Camera Systems

A projector-camera system is a useful tool for performing projection mapping and presenting information precisely aligned with the local environment. This is beneficial in different contexts, for example, to allow robots to project their intentions spatially correct into working environments [AMMA16], to enhance medical procedures by projecting additional information [BTM15, CGHSMR18], or to improve interaction with objects through spatial user interfaces [MSWT14], e.g., in industrial visual inspection scenarios. In contrast to glasses-based systems, multiple people can see the projection at once without the need for wearable hardware. The spread of mobile projector systems has increased over the last few years, and currently, pico projectors are even integrated into mobile phones. Laser projectors, or LCoS projectors with a laser light source, overcome the limitation of the depth of field and generate a *focus-free* image [GH12]. Additionally, 3D printers and low-cost 3D scanners have gained popularity, enabling the digitalization of objects and the creation of mockups, which can be used for spatial AR applications in the creative industry, e.g., by designers or architects [PLS$^+$15].

This thesis follows the shader lamps paradigm by Raskar et al. [RWLB01]. The appearance of a real object, with known 3D geometry, is manipulated by rendering a synthetic image of the object from the point of view of the projector. Using shader lamps in dynamic environments and the related sensing and tracking has been addressed in different ways in literature over the last few years but still faces several challenges.

Raskar et al. [RvBB$^+$03] use fiducial markers to detect the 3D pose of objects with a projector-camera system and to adapt the projection with respect to the object geometry dynamically. Also, more recent approaches [NWI17, AIS18, FMI20, IMG22] make use of fiducials printed or placed directly on the object surface. The markers are optimized for high-speed tracking [NWI17], integrated into a system that also uses a depth sensor [IMG22], or made less visible to the human eye with the projector [AIS18]. However, each object must be carefully designed and prepared to be suitable for tracking.

Hashimoto et al. [HK16] make the projection invisible to the camera sensor by using an infrared camera, allowing them to use a classical tracking approach unaffected by the projection. Similarly, the dynamic face projection system of Bermano et al. [BBIG17] uses an infrared image to track facial movements. Besides the need for special hardware, sensors working with short-wave illumination often establish gesture and person recognition in spatial augmented reality scenarios. This leads to an interference in the image of the infrared camera, which can disturb the tracking.

In contrast to this, other approaches [KH14, KKH15] use depth sensors, which are not distracted by a projection in the visible spectrum and estimate the object pose by aligning the object to the captured point cloud with the iterative closest point (ICP) algorithm and a particle filter. Due to the limited resolution of those sensors, the accuracy of the alignment between the real object and the projection is reduced compared to an image-based approach [HK16].

Closed-loop approaches capture the projection with a camera and correct the projection in order to improve the matching between projected content and the real world. Audet et al. [AOT10] introduced the use of projection for image alignment by modeling the light emitted by the projector and reflected into the camera. Zheng et al. [ZSW13] elaborate on the closed-loop concept and extend it to different AR paradigms, but as Audet et al., they do the alignment only for 2D targets. Kagami et al. [KH19] hide a projected fiducial in single binary frames of a high-speed DMD (Digital Micromirror Device) projection system and achieve real-time tracking of projections on a planar target following the closed-loop principle. Aside from the limitation of planar targets, the approach requires high-speed projector-camera pairs and advanced synchronization.

Resch et al. [RKK16] present a closed-loop tracking approach for 3D objects. They use discrete feature point correspondences between the projector image and the captured camera image. The 2D-2D correspondences are triangulated to reconstruct 3D points, which are then registered to the point cloud of the reference model via the ICP algorithm. This allows stable convergence of the optimization as long as a large number of feature point correspondences are found. However, simple textures and geometric patterns, as frequently used in shader lamp scenarios, do not provide enough distinct features [RKK16].

This work also focuses on the task of estimating the pose of a 3D object in a closed-loop fashion but combines it with the alignment methods as given by Audet et al. [AOT10]. An analysis-by-synthesis approach is employed, utilizing synthetically generated images of the projection on the reference surface. The object pose is iteratively corrected to simulate the appearance of the mismatched projection with the renderer, which simultaneously leads to the observed pose offset. Compared to the approach of Resch et al. [RKK16], this approach does not rely on point feature correspondence but instead minimizes the image difference between the simulation and the camera image. A radiometric calibration is omitted by using edge images. Also, Resch et al. convert the mesh into an intermediate point cloud, as needed for ICP, where the mesh density depends on the working distance, camera/projector resolution, and calibration quality [RKK16]. This conversion and the associated limitations are eliminated in the introduced method, as a direct image comparison is used for pose estimation.

2.2 6D Pose Estimation of Known Objects

Following the discussion on 6D rigid body pose tracking, this section focuses on estimating 6D poses for known objects without temporal knowledge. Recent advancements in CNNs have significantly enhanced 6D pose estimation capabilities. These advancements are explored alongside two key challenges this thesis aims to address. First, pose estimation becomes more complex when multiple objects are involved. Approaches often include training separate networks for each object or adapting existing models in overly simplistic ways to handle multiple objects (Sec. 2.2.1). Second, domain discrepancies between synthetic training data and real-world scenarios present a challenge, requiring strategies to bridge the reality gap (Sec. 2.2.2). Mask-guided operations are introduced as a key advancement to address the limitations of current methodologies in both challenges and improve pose estimation accuracy (Sec. 2.2.3).

2.2.1 Multi-Object 6D Pose Estimation

Approaches for 6D pose estimation with CNNs usually either regress the object pose directly [XSNF18, BJ19, HFWS20, TPV23], describe the object's appearance with a latent space code to compare it to pre-generated codes [SMD$^+$18, PMXF20, WPYW20], or, most often, regress the position of 2D projections of 3D points and calculate the pose with a Perspective-n-Points (PnP) algorithm. Approaches from the last category either predict object specific keypoints [RL17, TTS$^+$18, HHFS19, PLH$^+$19, SSH20] or dense correspondence/coordinate maps [PPV19, LWJ19, ZSI19, HAZ$^+$20, HBM20, TLPV21, HB22].

As an example, PVNet [PLH$^+$19] predicts 2D vectors at each pixel within the object silhouette, pointing towards predefined keypoints on the object's surface. These vectors are used to vote for the keypoint locations, and the 6D pose is then estimated using a PnP algorithm based on the aggregated votes. Yu et al. [YZKL20] enhance this with a differentiable proxy voting loss, which incorporates distances between pixels and keypoints, ensuring that distant pixels are less tolerant of inaccurate direction vector estimates to reduce prediction errors.

In contrast, EPOS [HBM20] establishes dense correspondences between pixels and object fragments. For each pixel, the network predicts the presence of an object, the likelihood of fragment presence given that object, and the precise 3D location on each surface fragment. This approach effectively handles symmetries by allowing 2D-3D correspondences to form a many-to-many relationship.

To deal with multiple objects, most approaches train a separate network per object and need multiple inferences per image [SMD+18, TTS+18, PLH+19, ZSI19, PPV19, LWJ19, SSH20]. Alternatively, increasing the number of output maps is proposed as a multi-object extension [RL17, PLH+19, ZSI19, HBM20], which risks serious accuracy drops [SCA+20] or complex and slow processing [HBM20]. Each added object contributes multiple extra output channels of the same size as the input image, which also requires significant GPU memory and complicates training. Sock et al. [SCA+20] add additional weights to optimize a CNN for multiple objects and reduce the multi-object performance gap. Still, they require the object class as input from a separate bounding box detector, such as RetinaNet [LGG+17], to perform their re-parametrization and one inference for every visible object. The same principle—detecting objects with a separate network in the first stage, followed by using object-specific networks for each detection—is employed by many multi-stage methods [SMD+18, LWJ19, BJ19, PPV19, ZZG+21, YYY21, HB22]. In this work, a *single-stage* method refers to an approach that utilizes a single trainable network for the entire process. In contrast to Hu et al. [HFWS20], who focus on direct pose regression from intermediate correspondences by combining multiple CNNs, applying PnP to the output 2D-3D correspondences is not treated as a separate stage.

Single-stage strategies for multi-object scenarios have been discussed less frequently in the literature than multi-stage strategies. The category-level approach by Hou et al. [HAZ+20] unifies features from different instances of one class. It requires similar geometric structures per category and aligned models during training. A generic corner detector [PIL19] can be used for new objects without retraining, but since generic 3D corners are detected, object pose is found by brute force. Recently, methods have emerged that unify object-independent pose estimation in a single network, leveraging foundation models [ÖLT+24] or training on object-independent matching using large-scale datasets that depict thousands of objects [NGSL24]. However, for these methods, template-based object-independent detection remains a separate stage [NGP+23], and they still exhibit a significant performance and efficiency gap compared to object-specific networks.

Similar to the method in this thesis, other studies [ALCL21, TLPV21, TPV23] focus on single-stage multi-object pose estimation for known objects. PyraPose [TLPV21] extends the network's capacity by improving occlusion handling through the aggregation of multi-resolution features in a feature pyramid network. InstancePose [ALCL21] predicts the 3D object coordinates of multiple objects in combined output maps and generates an error mask to filter out faulty pixels near object silhouettes. COPE [TPV23] integrates object detection with direct 6D pose regression from intermediate correspondences. A small network head transforms 2D-3D correspondences, which are then directly regressed for each feature map location to a 6D pose. Multiple hypotheses per object are clustered and averaged using class probabilities and bounding-box predictions.

The keypoint-predicting decoder in this thesis utilizes object masks, predicted by a separate network head, to enhance occlusion handling. It not only minimizes errors near the object silhouette but also uses this region specifically for the predictions, all while maintaining a fixed number of output maps.

2.2.2 Synthetic Training Data for 6D Pose Estimation

Manually labeling data for 6D pose estimation is a time-consuming process, as not only do the objects in the image need to be labeled, but also their exact poses need to be specified. Since it is challenging to generate comprehensive data manually, early methods for pose estimation mainly used training data similar to the later test data [RL17, XSNF18]. This similarity, e.g., in terms of the used camera, the scene configuration, or the illumination, severely limited their generalizability to new situations.

Applications involving rigid objects with known 3D geometry allow the use of computer graphics to generate synthetic data for freely configurable scenes. The previously described *reality gap* can be addressed through random scene reconfiguration (domain randomization) [TFR+17] or by using near-photorealistic training data based on physically-based rendering (*pbr*) [HVG+19]. Publicly available tools [TTM+18, DSW+19] offer easy solutions for data generation, and the BOP (Benchmark for 6D Object Pose Estimation) Challenge [Bop20a] emphasizes the use of synthetic training data combined with real test data to evaluate the robustness and generalization of 6D object pose estimation algorithms.

Tremblay et al. showed that by combining photorealistic and domain-randomized synthetic training data, sufficient variation is generated for their method DOPE to operate on real test data [TTS+18]. They demonstrate this capability by applying their method to real-world robotic tasks without any fine-tuning. However, later work has recognized that even the data considered photorealistic still leaves a performance gap compared to real data if not treated appropriately [WMS+20, LHSJ21, TLPV21, YYY21, ZZG+21]. As an example, SD-Pose [LHSJ21] bridges the reality gap by decomposing the input image into multi-level semantic representations, which are assumed to be more domain-invariant than the raw RGB image. These representations include edge detection, separate grayscale and color images, and a sketch representation. Other methods [LCAS20, HB22] use the render-and-compare strategy, a learned variant of iterative model-based pose refinement, such as DeepIM [LWJ+18], to improve coarse pose estimates via 3D model alignment. While efficient for closing the domain gap, it takes seconds depending on the number of objects, making it unsuitable for real-time applications.

Other methods extend the training data with unlabeled real data to close the reality gap. DAKDN [ZZG+21] aims to enforce deep feature similarity between real and synthetic images. To achieve this, it incorporates unlabeled real data during training and constrains the detected 3D keypoints in these images according to their relative positions in a 3D keypoint graph. DSC-PoseNet [YYY21] initializes pose estimation by training on synthetic data and refines it with real data, using only 2D bounding box annotations. The method employs a differentiable renderer to align the estimated mask with a predicted mask. By estimating poses at two scales and enforcing consistency between predicted points, the training process is stabilized. Compared to the earlier Self6D [WMS$^+$20] method, which likewise initializes with synthetic data and uses a differentiable renderer, DSC-PoseNet does not require depth information for evaluating self-supervised training with real data.

Especially for multi-object approaches [HBM20, TLPV21, TPV23], the use of synthetic training data is crucial due to the difficulty of creating real datasets with multiple annotated objects per image. This work narrows the reality gap by providing the network access to silhouettes, a nearly domain-invariant feature [BJ19, WPYW20]. This enables precise results without refinement and makes it fast enough for real-time applications.

2.2.3 Mask-guided operations

This thesis introduces a new network structure that decouples semantic object identification from the keypoint regression. Consequently, keypoint regression can utilize pixel-wise segmentation results to guide the prediction of 2D locations for 3D keypoints. To maximize the benefits of the mask, various techniques originating from Generative Adversarial Network (GAN)-based conditional image synthesis and style transfer are introduced to the field of object pose estimation.

Conditional normalization Normalization layers in CNNs speed up the training and improve the accuracy [IS15]. A learnable affine transformation recenters and rescales the normalized features. In the unconditional case [IS15, UVL16], the normalization does not depend on external data. Conditional instance normalization (CIN) [DSK17] increases the capacity of a CNN by learning multiple sets of normalization parameters for different classes, e.g., for neural style transfer [GEB15]. Adaptive instance normalization [HB17] uses statistics of a style image as (de)normalization parameters, which avoids extra training per style. Sock et al. apply CIN to multi-object 6D pose estimation [SCA$^+$20], but require object identity as input and handle only one identity at a time.

Spatially-adaptive instance (de)normalization (SPADE) [PLWZ19] uses per-pixel normalization parameters depending on semantic segmentation and a pixel's position. Zhu et al. extend SPADE with semantic region-adaptive normalization (SEAN) by allowing the selection of a specific style for each class instead of selecting one style per image [ZAQW20]. Tan et al. reduce computational and parameter overhead of SPADE by prioritizing semantic over spatial awareness [TCC$^+$21]. In their class-adaptive instance (de)normalization (CLADE), a guided sampling operation selects the set of de-normalization parameters based on the semantic class of a pixel.

The proposed method builds on this work by first estimating semantic masks for multiple objects and then locally applying learnable, object-specific parameters through CLADE operations in a mask-guided decoder. These adaptive modulation parameters help the network adjust to different object characteristics without significantly increasing its overall size, contributing to its ability to handle multiple object classes.

Content-aware transformations While the spatial invariance of convolutions is beneficial for most computer vision tasks, for some tasks, local awareness of a filter can be helpful. CoordConv [LLM$^+$18] demonstrates the benefit of giving a filter access to its position, enabling networks to learn varying degrees of translation dependence as required. This approach has been successfully used in panoptic segmentation [SBK19, WKS$^+$20] and semantic image synthesis [TCC$^+$21].

Similarly, spatial awareness can be introduced by using binary maps to mask out regions of the feature map that should not contribute to the network's output, e.g., for inpainting, depth upsampling, or padding [USS$^+$17, LRS$^+$18, LSW$^+$18]. Mask-guided convolutions have been extended for non-binary annotations [YLY$^+$19] and adapted for multi-class image synthesis by Dundar et al. [DSL$^+$20]. Their operation helps the network capture the relationship between panoptic masks and the generated image, facilitating the synthesis of clearly separated, distinct instances with consistent spatial alignment.

Mazzini et al. [Maz18] point out that spatially invariant operations for feature-map upsampling fail to capture the semantic information needed for dense prediction tasks. However, guided operations can make upsampling learnable [Maz18, WCX$^+$19]. Content-aware upscaling, applied in multi-class image synthesis [DSL$^+$20], takes advantage of a higher-resolution mask to keep features aligned with instance segmentation, further improving the visual quality of the generated output.

This work applies guided convolution and upsampling [DSL$^+$20] in a segmentation-aware decoder to guide the network to focus on the object region when inferring keypoint locations. This is expected to enhance the effectiveness of CLADE within the respective segmented region, improve the separation of predictions for occluding instances, and increase awareness of the object's shape.

2.3 Global Model-based Camera Localization

Understanding how a camera image is located in an environment is crucial for advancing spatial awareness in robotic systems and AR applications. This section explores key developments in indoor camera localization, focusing on methods applicable to new, unseen scenes (Sec. 2.3.1). This matches the goal of Chapter 6: developing a 6D pose estimation approach that aligns RGB images with a 3D scene model not encountered during training. The following review of related work on relative pose regression (Sec. 2.3.2) and perspective-to-panorama matching (Sec. 2.3.3) highlights techniques that contribute to this objective.

2.3.1 Indoor Camera Localization

Indoor camera localization is a longstanding challenge in the computer vision community. Early systems relied on hand-crafted, invariant visual features to assess the similarity between a captured image and images in a database, enabling the retrieval of the most similar localized image [WBB05]. Recent advancements in neural networks have led to the proposal of many deep learning-based approaches. Typically, these methods rely on prior scene knowledge to build a scene representation and estimate a relation to a query image. Some methods use a 3D model generated via Structure-from-Motion [AFS+11] from registered camera images of the indoor environment and calculate the pose from 2D-3D correspondences, which might be either sparse [LLD17, XG22] or dense [TOS+18].

Alternatively, encoding the entire scene within a neural network and performing regression to determine the absolute pose is a commonly used strategy [KGC15, NB17, RVB18]. However, training such networks is often very time-intensive [CCPB24]. To address this, Map-Relative Pose Regression [CCPB24] introduces a partially scene-agnostic approach that offers greater general applicability. In this method, the pose regressor is trained on hundreds of scenes, and a smaller, scene-specific network is trained for each new environment, requiring only a few minutes of adaptation. Despite these advancements, the reliance on pre-built scene representations or scene-specific models still limits the adaptability of these methods, making them unsuitable for localization in entirely new environments.

By leveraging synthetic renderings from a detailed 3D BIM model, Acharya et al. reduced the need for real-world reference data. Their approaches achieve real-time indoor localization [AKW19] and an accuracy of approximately 1 meter [ATM+22] but are limited to the building of their origin, lacking generalizability to other structures. Ha et al. [HKPK18] utilize features from a pre-trained neural network to match reference images in a database of encoded renderings. This technique, applied in a facility management field study [BHK19], is restricted in its ability to handle new poses and scene alterations. It relies on a highly detailed BIM model to minimize the domain gap, ensuring that the deep features of the rendering and camera images are sufficiently similar. Liu et al. [LSK+15] pioneered floor plan-based image localization, predicting the facing wall within a cuboidal layout without estimating a full 3D pose. Howard et al. [HJRSP21, HJP22] demonstrated 2D localization, specifically the xy coordinates relative to a floor plan, for panoramic query images from unvisited scenes. They achieve this by aligning inferred room layouts with panoramic reference renderings [HJRSP21] or encoded floor plans [HJP22].

Several methods estimate the 2D position and 1D rotation of a perspective camera relative to a floor plan. PF-net [KHL18] uses a recurrent neural network optimized for a differentiable particle filter, designed for sequential updating but suboptimal for single images [MKB$^+$22]. It extends Monte Carlo Localization (MCL) [DFBT99], a probabilistic method typically used with LiDAR data, such as in robot localization, by attaching a learnable image-based observation model. Since it is trained end-to-end with the particle filter, it learns to approximate a complex observation model optimized for the specific task, thereby reducing the complexity of learning. F^{3}Loc [CWVP24] improves sequential 2D localization against a floor plan by integrating multiview geometry cues. During training, it uses augmented roll and pitch angles to handle non-upright camera poses. However, the method requires data from an inertial measurement unit during inference and cannot estimate angles itself. Additionally, because the method focuses on sequential updating and omits semantic information, its single-view localization accuracy is limited. LASER [MKB$^+$22] localizes the camera with respect to a semantic floor plan and considers both perspective and panoramic queries. The rendering codebook scheme efficiently encodes floor plan information but is limited to 2D data, which constrains the network's ability to estimate the remaining degrees of freedom.

To sum up, existing methods either depend on accurate scene-specific data for a detailed 3D representation or do not estimate the full 6D pose. In contrast, the method presented in this thesis neither depends on real-world reference images, a detailed 3D model, or scene-specific training but is still able to estimate the 6D perspective camera pose by introducing a new cross-domain panoramic-to-perspective image registration approach between RGB images and semantic panoramas.

2.3.2 Relative Pose Regression

Relative pose regression (RPR) aims to estimate the relative motion between a query image and reference images with known poses. These frameworks, relying on image comparisons, often enhance generalizability. Typically, they directly utilize localized examples at inference time [DWS$^+$19]. Retrieval-based methods like NetVLAD [AGT$^+$16] and DenseVLAD [TAS$^+$15] start by identifying the most similar images in a database, providing a starting point for relative pose regression. PixLoc [SUL$^+$21] predicts multiscale deep features for a direct alignment from the reference and the query image with a CNN and uses geometric optimization for calculating the pose. Turkoglu et al. [TBS$^+$21] employ a GNN to re-localize cameras with respect to multiple frames. A graph links the query image to training counterparts, and a GNN refines representations for a consistent camera pose estimate.

The proposed method also estimates the absolute pose by matching against multiple localized references, but goes further by using simplified synthetic reference data and incorporating multimodal matching between panoramic references and perspective queries.

2.3.3 Perspective-to-Panorama Matching

Matching between perspective query and panoramic reference images is a crucial aspect of localization, yet it has received limited attention in the literature. Most systems avoid multimodal matching and either match perspective crops from panoramas to perspective queries [TAS$^+$15, AGT$^+$16], or match panoramic images to other panoramic images [ZBLC21]. An exception is the non-learning-based localization approach by Yazawa et al. [YUS$^+$09]. They match SURF features between a database of cylindrical panoramas and perspective images captured by a user and retrieve the localized panorama with the largest number of feature matches, which they assume to represent the user's position. However, due to the differing projections between panoramic and perspective images, feature matches may be unreliable or sparse, leading to potential inaccuracies in localization.

While previous work has discussed geometric fundamentals for integrating catadioptric and perspective camera systems into hybrid stereo setups, including concepts such as epipolar constraints [Stu02] and calibration [HCL12], these ideas have not yet been incorporated into AI-based localization systems. In the domain of CNNs and panoramic image processing, distortion-aware convolutional filters were proposed by Tateno et al. [TNT18] and further refined by Fernandez et al. [FLFPY$^+$20], particularly for indoor equirectangular images. Unlike traditional convolutions, these distortion-aware filters target the differences between perspective and panoramic images, making them particularly valuable for cross-modal matching.

Recently, perspective-to-panoramic matching advanced due to sliding window approaches in feature space [OB21, SSY$^+$23]. It improved visual recognition compared to methods that simply sampled perspective reference images from panoramas [ZS10]. Orhan et al. [OGB22] explored leveraging semantic segmentation similarity to address appearance variations between reference and query images.

In this work, an approach that includes perspective-to-panorama matching is presented, using virtual scene renderings as panoramic references. This approach is inspired by multimodal zero-shot retrieval [JGHE21] and object detection, particularly Mercier et al.'s work [MGGL21]. Their method, DTOID, employs a multi-branch network to learn deep template matching between a set of object templates and a camera image without requiring that a specific object has been seen during training.

The approach for 6D camera localization proposed in this thesis utilizes the semantic panorama as global context and uses the perspective image as a template, learning a multimodal matching with a similar architecture.

3 Enhancing Model-Based 6D Tracking for Projector-Camera Systems

This chapter presents a framework for 6D tracking of 3D objects in monocular camera images and extends it to incorporate projected textures, enabling tracking in projector-camera systems.

The framework follows the principle of model-based analysis-by-synthesis [DG88], iteratively refining an object's pose by minimizing discrepancies between camera images and synthetic renderings. Starting from an initial approximate pose, the object's motion parameters—relative translation and rotation—are estimated from the visual differences between the real and synthetic images. To bridge the domain gap between real-world and synthetic images, image processing methods address texture and illumination disparities to create a comparable representation. This allows the use of simple rendering techniques, which are less computationally expensive than photorealistic rendering.

The first part of this chapter introduces a real-time tracking system based on contours and edges as domain-invariant features. It begins with an overview of the principles of projective geometry and the mathematical tools used (Sec. 3.1). A motion model for contour-based tracking is then derived and integrated into a robust tracking algorithm (Sec. 3.2). This monocular tracking algorithm forms the foundation for the hybrid AR system developed in Chapter 5.

The main contribution of this chapter is the enhancement of the tracking for *spatial augmented reality* (SAR) (Sec. 3.3). In SAR, the *shader lamps* technique augments the real environment by projecting new textures onto known target geometries [RWLB01]. The projected image shows the texture-mapped object rendered from the projector's view. It aligns properly with the real object, assuming its 6D pose is known. In dynamic scenes, accurate pose tracking is crucial because the projected image needs to be constantly updated to maintain the illusion that the projected texture is part of the real object.

However, in a camera image, the projected texture on an object's surface behaves differently from that of a naturally textured object [AOT10]. When a naturally textured object undergoes a transformation, all texture points are affected uniformly. In contrast, the projected texture lags behind as it slides over the object's surface during movement. This delay leads to perspective distortion, which conflicts with the assumptions of feature- or contour-based trackers and hinders accurate pose estimation. Previous approaches mitigate this interference by either preparing the object with markers (e.g., [AIS18, IMG22]) or using special hardware to render the projection invisible to the camera (e.g.,

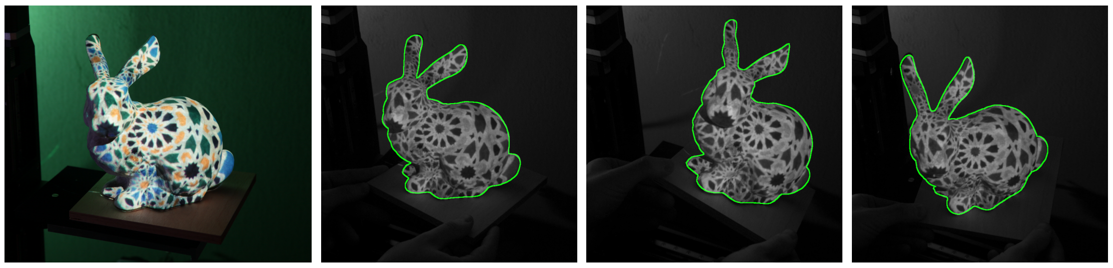


Figure 3.1: Projection-distortion-based object tracking. The projector-camera system captures the moving object and estimates the object pose (highlighted by green contour) from projection distortions. The projected image is corrected in a feedback loop to generate a matching projection. *(Reprinted from [GHE19], © 2019 IEEE)*

[HK16, BBIG17]). Alternatively, *closed-loop* systems utilize visual feedback from the projection onto the object for registration [AOT10]. So far, closed-loop approaches are limited to planar targets [ZSW13, KH19] or require the projected texture to be detailed enough for dense point feature correspondences to be found [RKK16].

The proposed method integrates a closed-loop into model-based pose tracking via analysis-by-synthesis. It requires only conventional hardware (a projector and a machine vision camera), works with 3D targets, and uses texture edges for registration, omitting radiometric calibration and feature point matching. The method simulates the projection onto the object, including its appearance during movement under the projection. Two new motion models describe the relationship between the movement of a projected point on the image sensor and the movement of planar (Sec. 3.3.1) and non-planar objects (Sec. 3.3.2) in 3D space. The pose is iteratively optimized to minimize the difference between the real observation and the rendered expectation, while compensating for the projection distortion. Figure 3.1 shows an object that is augmented with a shader lamp projection and tracked solely by analyzing the projection captured by a camera.

To optimize the object pose, precise projector-camera calibration is essential. An adapted motion model enables optimizing the projector's pose relative to the camera, given a known object pose (Sec. 3.3.4). This requires only minor modifications to the previously described formulation, simplifying the calibration refinement of projector-camera systems and maximizing the accuracy of projections on real-world objects.

The experiments focus on validating the projection-distortion-based tracking using synthetic and real data (Sec. 3.4), while contour-based tracking is evaluated in conjunction with deep pose estimation in Chapter 5.

3.1 Mathematical Background

A grayscale image of size $w \times h$ is represented as a matrix $\mathbf{I} = (i_{x,y}) \in \mathbb{R}^{w \times h}$, where each element $i_{x,y}$ denotes the intensity value at pixel position (x, y). Similarly, a rendered grayscale image of a virtual 3D scene is represented by the matrix $\hat{\mathbf{I}} \in \mathbb{R}^{w \times h}$. A depth map is represented as $\hat{\mathbf{D}} = (\hat{d}_{x,y}) \in \mathbb{R}^{w \times h}$, where each $\hat{d}_{x,y}$ encodes the z-distance of pixel (x, y) from the camera and is generated during rendering. A normal map, also generated during rendering, encodes surface normals at each pixel and is represented as a matrix $\hat{\mathbf{N}} = (\hat{\mathbf{n}}_{x,y}) \in \mathbb{R}^{w \times h \times 3}$, where $\hat{\mathbf{n}}_{x,y}$ is a 3D normal vector.

Three-dimensional objects in the scene are modeled as meshes $\mathcal{M} : \{\mathcal{V}, \mathcal{F}\}$ consisting of a set of K vertices $\mathcal{V} = [v_1, \ldots, v_k]$ and a set $\mathcal{F}$ of faces, with each face comprising three indices of vertices to form a triangle. The coordinates of $\mathcal{V}$ are given in the local coordinate system of the object, whose origin is usually in the center of the object.

Cameras are simplified using the pinhole camera model with intrinsic parameters represented by the 3×3 camera matrix $\mathbf{K}$. Here, f_x, f_y are the scaled horizontal and vertical focal length, and u_x, u_y are pixel coordinates of the principal point:

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & u_x \\ 0 & f_y & u_y \\ 0 & 0 & 1 \end{pmatrix}. \tag{3.1}$$

A 3D point $\mathbf{p} = [p_x, p_y, p_z]^T$ defined relative to the camera coordinate system is projected to the 2D image point $\mathbf{x} = [x, y]^T$ with the intrinsic matrix $\mathbf{K}$ and a dehomogenization $\pi(\mathbf{q}) = [q_x/q_z, q_y/q_z]^T$.

$$\mathbf{x} = \pi(\mathbf{K}\mathbf{p}) \tag{3.2}$$

Given a depth value $\hat{d}_{x,y}$, the corresponding 3D point $\hat{\mathbf{p}}_{x,y} = [\hat{p}_x, \hat{p}_y, \hat{p}_z]^T$ in the camera coordinate system is computed as:

$$\hat{p}_x = \frac{(x - u_x) \cdot \hat{d}_{x,y}}{f_x}, \quad \hat{p}_y = \frac{(y - u_y) \cdot \hat{d}_{x,y}}{f_y}, \quad \hat{p}_z = \hat{d}_{x,y}. \tag{3.3}$$

A motion of a point in 3D space is described by a 3×3 rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t} = [t_x, t_y, t_z]^T$, transforming $\mathbf{p}$ to $\mathbf{p}' = \mathbf{R}\mathbf{p} + \mathbf{t}$.

Local tracking models usually assume an initial pose of an object to be known up to an unknown offset transformation $\Delta\mathbf{R}$, $\Delta\mathbf{t}$. To apply the rotation offset in the coordinate frame of a 3D model, the model is translated to the coordinate origin before being rotated.

$$\mathbf{p} = \Delta\mathbf{R}(\hat{\mathbf{p}} - \mathbf{t}) + \mathbf{t} + \Delta\mathbf{t} \tag{3.4}$$

Under the assumption of small motion, $\mathbf{R}$ can be approximated by a linearized rotation matrix $\Delta\mathbf{R}$, where the unknown parameters $\Delta\mathbf{r} = [\Delta r_x, \Delta r_y, \Delta r_z]^T$ are the rotation angles around the x-, y- and z-axis.

$$\Delta\mathbf{R} = \begin{pmatrix} 1 & -\Delta r_z & \Delta r_y \\ \Delta r_z & 1 & -\Delta r_x \\ -\Delta r_y & \Delta r_x & 1 \end{pmatrix} \tag{3.5}$$

The linearized representation can later be used better during motion estimation since it condenses the three degrees of freedom of a rotation matrix to three scalar values. In total, the transformation has six degrees of freedom to be estimated.

To measure the distance between an estimated rotation matrix $\mathbf{R}_{\text{est}}$ and a ground truth rotation matrix $\mathbf{R}_{\text{gt}}$, this work utilizes the geodesic distance on the 3D manifold of rotation matrices (in degrees), where $\text{tr}(\cdot)$ denotes the trace of a matrix:

$$r_{\text{err}} = \cos^{-1}\left(\frac{\text{tr}(\mathbf{R}_{\text{est}}\mathbf{R}_{\text{gt}}^{\top}) - 1}{2}\right) \cdot \frac{180}{\pi}. \tag{3.6}$$

3.1.1 Flow-based Optimization

The optical flow constraint equation [HS81] is a fundamental equation in motion estimation. It relates the temporal difference and partial derivatives of the image intensity in horizontal and vertical directions. Therefore, let $\hat{\mathbf{I}}$ denote an image at time t, and $\mathbf{I}$ denote an image at time $t+1$. The terms $\partial\hat{\mathbf{I}}/\partial x$ and $\partial\hat{\mathbf{I}}/\partial y$ represent the partial derivatives of $\hat{\mathbf{I}}$ with respect to the horizontal and vertical directions, respectively.

Assuming a point at coordinates (x, y) in image $\mathbf{I}$ corresponds to a point at coordinates $(\hat{x}, \hat{y})$ in image $\hat{\mathbf{I}}$, the displacements $\mathbf{U} = (u_{x,y}) = (x - \hat{x}) \in \mathbb{R}^{w\times h}$ and $\mathbf{V} = (v_{x,y}) = (y - \hat{y}) \in \mathbb{R}^{w\times h}$ describe the flow field. The frame-to-frame difference $\mathbf{I} - \hat{\mathbf{I}}$ serves as a proxy for the temporal change in image intensity $\partial\hat{\mathbf{I}}/\partial t$. Equation (3.7) is used to estimate the relative motion between the two images.

$$\frac{\partial\hat{\mathbf{I}}}{\partial x} \cdot \mathbf{U} + \frac{\partial\hat{\mathbf{I}}}{\partial y} \cdot \mathbf{V} \approx \hat{\mathbf{I}} - \mathbf{I} \tag{3.7}$$

In analysis-by-synthesis for relative motion estimation, the goal is to minimize displacements between the real image $\mathbf{I}$ and the synthetic image $\hat{\mathbf{I}}$ and relate these displacements to the motion parameters of a rigid body. A way to establish this relation is to isolate the motion parameters as unknowns on a per-pixel level. This allows for the formulation of a system of equations, with one equation per pixel in the object's silhouette, which can then be solved to estimate the unknown motion parameters. Since the object is assumed to be rigid, the same transformation applies to all pixels in the silhouette, allowing the estimation of the global transformation by solving the system of equations.

Solving for motion parameters in a least-squares sense An overdetermined system of equations of the form $\mathbf{Ax} = \mathbf{b}$ consists of more equations than unknowns, with m equations and n unknowns, where $m > n$. Typically, no unique solution satisfies all equations; instead, a best-fit solution is sought. In this context, $\mathbf{A}$ is the $m \times n$ coefficient matrix, $\mathbf{x}$ is the $n \times 1$ vector of unknowns, and $\mathbf{b}$ is the $m \times 1$ vector of constants on the right-hand side of the equations. The least-squares solution of $\mathbf{Ax} = \mathbf{b}$ is obtained by solving the normal equation:

$$\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b}. \tag{3.8}$$

Assuming that $\mathbf{A}^T\mathbf{A}$ is invertible and that the columns of $\mathbf{A}$ are linearly independent, the least-squares solution of equation $\mathbf{Ax} = \mathbf{b}$ is given by:

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}. \tag{3.9}$$

Here, the matrix $\mathbf{A}^T\mathbf{A}$ is a square matrix of size $n \times n$ and $\mathbf{A}^T\mathbf{b}$ is an $n \times 1$ vector. A mathematical tool for efficiently solving this system is LU decomposition. It factors $\mathbf{A}^T\mathbf{A}$ into a lower triangular matrix $\mathbf{L}$ and an upper triangular matrix $\mathbf{U}$, such that:

$$\mathbf{A}^T\mathbf{A} = \mathbf{LU}. \tag{3.10}$$

The decomposition can be obtained using Gaussian elimination. Solving the system is then performed in two steps:

1. $\mathbf{Ly} = \mathbf{A}^T\mathbf{b}$ is solved for $\mathbf{y}$ via forward substitution, with $\mathbf{y}$ being a $n \times 1$ vector of unknowns. Since $\mathbf{L}$ is lower triangular, this can be done efficiently.
2. $\mathbf{Ux} = \mathbf{y}$ is solved via backward substitution, which can also be done efficiently since $\mathbf{U}$ is upper triangular.

LU decomposition cannot be performed if $\mathbf{A}$ has linearly dependent columns, $\mathbf{A}^T\mathbf{A}$ is not invertible, or the determinant of $\mathbf{A}^T\mathbf{A}$ is zero. In contrast, a generalized inverse, also known as the Moore-Penrose pseudoinverse, is defined for any matrix, including singular matrices. It can still be calculated using singular value decomposition (SVD) [BIG06].

3.1.2 Iteratively Reweighted Least Squares with M-estimators

When comparing real camera images with synthetically generated ones, it is common that not all visual phenomena and scene configurations can be accurately replicated by computer graphics. For example, during motion tracking, when an object is held by a human hand, parts of the object are occluded. As long as the hand-object interaction is not explicitly modeled, the assumptions of a motion model are not met in these areas. Therefore, in a least squares system in the form $\mathbf{Ax} = \mathbf{b}$, many equations will not contribute to finding a correct solution. Since the previously presented method is sensitive to noise, a suitable solution may not be found.

One method to improve this is the Iteratively Reweighted Least Squares (IRLS) [HW77]. IRLS iteratively reweights the observations to fit a linear model to them while reducing the influence of outliers by assigning small weights. This is related to the robust technique called *M-estimators* [Zha97]. A *residual* r_i is the difference between the i^{th} observation and its fitted value. Least squares minimizes $\sum_i r_i^2$, which leads to strong deviations in case of outliers due to the quadratic weighting. An M-estimator replaces r_i^2 with another penalty function p, allowing for the use of norms other than the L_2-norm.

$$\min \sum_i p(r_i) \tag{3.11}$$

The function p can be any function that is positive-definite, symmetric, and has a unique minimum at zero. It should also increase slower than the square function as the residual increases [Zha97]. The *influence function* $\psi(r_i) = \mathrm{d}p(r_i)/\mathrm{d}r_i$ measures the effect of a residual on the parameter estimation. The *weight function* $w(r_i) = \psi(r_i)/r_i$, in contrast, assigns a weight to a residual.

The parameter vector $\mathbf{x} = [x_1, \ldots, x_m]$ to be estimated is the solution to the following m equations

$$\sum_i w(r_i) r_i \frac{\partial r_i}{\partial x_j} = 0, \quad \text{for } j = 1, \ldots, m \tag{3.12}$$

and the same as the following IRLS estimation

$$\min \sum_i w(r_i^{(k-1)}) r_i^2. \tag{3.13}$$

The estimation is repeated K times, with k being the iteration number and the weights $w(r_i^{(k-1)})$ being recomputed in each iteration. The calculation is initialized with $w(r^0) = 1$, meaning that the initial estimates correspond to solving a linear system with equally weighted components, e.g., using LU decomposition.

M-estimation allows for the use of different penalty functions to handle outliers. The L_2 penalty (i.e., $p(x) = x^2/2$) has an unbounded influence function and a constant weight function, and is therefore not robust. In contrast, the L_1 penalty (i.e., $p(x) = |x|$) is not strictly convex, has an unbounded weight function that is undefined at $x = 0$, and is therefore unstable.

In the context of optical flow, the robust Charbonnier penalty [CBFAB94, BWS05]

$$p(x) = \sqrt{x^2 + \epsilon^2}, \tag{3.14}$$

where ϵ is a small constant (e.g., $\epsilon = 0.001$), has been shown to be a robust convex function [SRB10]. It is a differentiable variant of the $L1$ norm. Convexity is ensured as it converges to the L_2 norm for small values of x, while robustness is achieved by approximating the L_1 norm for larger values, assigning less weight to large residuals.

Sun et al. [SRB10] show that the Charbonnier penalty performs better than the likewise frequently used non-convex Lorentzian penalty in the context of optical flow estimation. They reason that non-convex functions are more difficult to optimize, which leads to a non-optimal local minimum being found more often. Additionally, the Lorentzian penalty has a descending first derivative, which can cause it to yield erroneous solutions [Zha97].

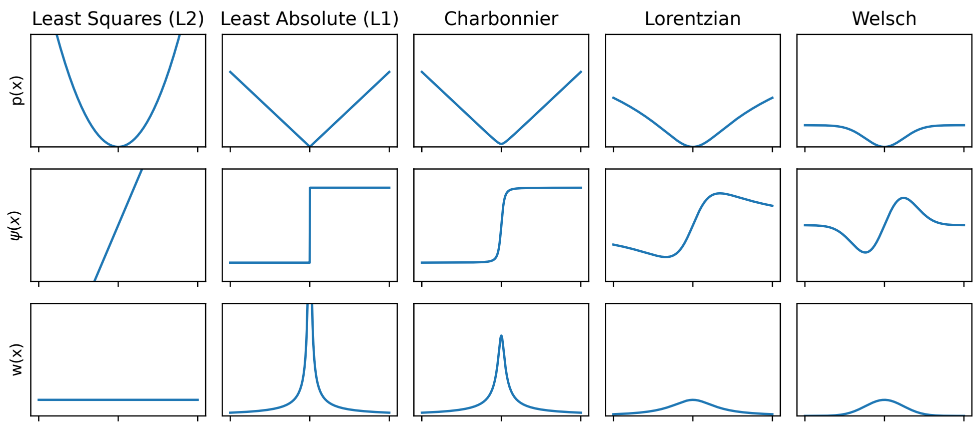


Figure 3.2: Comparison of different M-estimators. The plots show the behavior of the penalty $p(x)$, influence $\psi(x)$, and the weight function $w(x)$ around zero.

Also, functions such as the Welsch or Tukey penalty functions, which further reduce or even suppress the influence of outliers, could be suboptimal for optical flow-related optimization [Zha97]. The noise level and the separation of outliers and inliers are generally unknown, which complicates the choice of the threshold parameter. An overview of the characteristics of different M-estimators is given in Fig. 3.2.

3.2 Contour-based Registration

In analysis-by-synthesis applications, estimating the pose offset involves comparing a captured grayscale image $\mathbf{I}$ with a rendered image $\hat{\mathbf{I}}$. Both images depict a target object or its 3D model $\mathcal{M}$ from similar viewpoints. A similarity measure is defined to evaluate how well $\hat{\mathbf{I}}$, generated with the current scene parameters (e.g., object pose), matches $\mathbf{I}$. This measure should allow for optimization, enabling scene parameters to be adjusted to improve the alignment between the two images.

An important requirement for the comparison is that the features used are domain-invariant, meaning they can be extracted in a consistent manner from both the captured image and the rendered image. Furthermore, perspective invariances are required, so the features should still be extractable if the pose changes.

The contours of an object are such stable features. In images, they are noticeable through local changes in brightness, regardless of whether the scene is real or virtual. One way of classifying contours is by dividing them into three types of edges: silhouette, geometry, and texture edges. Silhouette edges define the boundary between the object and its background, geometry edges are formed by the internal structure of the object, and texture edges are created by changes in the surface color of the object. In this thesis, focusing on the alignment of image edges with object contours provides an effective method for local pose refinement, particularly for potentially untextured objects.

3.2.1 Dense Contour-based Optimization

One advantage of using contours over local point features (e.g., [Low04]) is that edges can be extracted with simple local operators [SF+68, Can86]. This work explores the use of potentially untextured objects with simple geometric shapes, where the search for stable, invariant point features is particularly challenging [WZQ19]. Furthermore, the contour-based approach enables pose estimation to be framed as a pixel-wise comparison of edge images, aligning with the scheme described in Sec. 3.1.1. First, the relationship between image formulation and the optical flow equation is introduced, building on the work of Eisert et al. [EG97]. The model is then integrated into edge-based tracking.

A *motion model* for motion compensation defines the brightness value in the motion compensated virtual image $\hat{\mathbf{I}}$ as a function of the pose offset $\Delta\mathbf{R}$, $\Delta\mathbf{t}$. The steps for estimating 6D rigid body motion are outlined below and applied similarly in subsequent sections. First, perspective division projects $\mathbf{p}$ in Eq. (3.4) onto the 2D image plane:

$$\begin{aligned} \frac{p_x}{p_z} &= \frac{\hat{p}_x - (\hat{p}_y - t_y)\Delta r_z + (\hat{p}_z - t_z)\Delta r_y + \Delta t_x}{\hat{p}_z - (\hat{p}_x - t_x)\Delta r_y + (\hat{p}_y - t_y)\Delta r_x + \Delta t_z} \\ \frac{p_y}{p_z} &= \frac{\hat{p}_y + (\hat{p}_x - t_x)\Delta r_z - (\hat{p}_z - t_z)\Delta r_x + \Delta t_y}{\hat{p}_z - (\hat{p}_x - t_x)\Delta r_y + (\hat{p}_y - t_y)\Delta r_x + \Delta t_z}. \end{aligned} \tag{3.15}$$

The equation is simplified using the first-order Taylor approximation of the form $1/(1+d) \approx 1 - d$. This form is achieved by dividing the numerators and denominators in Eq. (3.15) by $\hat{p}_z$. The approximation holds if d is small, which is the case for continuous motion offset estimation, where both $\Delta\mathbf{R}$ and $\Delta\mathbf{t}$ are assumed to be small.

$$\begin{aligned} \frac{p_x}{p_z} &\approx \frac{\hat{p}_x}{\hat{p}_z} - \frac{\hat{p}_y - t_y}{\hat{p}_z}\Delta r_z + \left(1 - \frac{t_z}{\hat{p}_z}\right)\Delta r_y + \frac{\Delta t_x}{\hat{p}_z} + \frac{\hat{p}_x}{\hat{p}_z}\left(\frac{\hat{p}_x - t_x}{\hat{p}_z}\Delta r_y - \frac{\hat{p}_y - t_y}{\hat{p}_z}\Delta r_x - \frac{\Delta t_z}{\hat{p}_z}\right) \\ \frac{p_y}{p_z} &\approx \frac{\hat{p}_y}{\hat{p}_z} - \frac{\hat{p}_x - t_x}{\hat{p}_z}\Delta r_z + \left(1 - \frac{t_z}{\hat{p}_z}\right)\Delta r_x + \frac{\Delta t_y}{\hat{p}_z} + \frac{\hat{p}_y}{\hat{p}_z}\left(\frac{\hat{p}_x - t_x}{\hat{p}_z}\Delta r_y - \frac{\hat{p}_y - t_y}{\hat{p}_z}\Delta r_x - \frac{\Delta t_z}{\hat{p}_z}\right) \end{aligned} \tag{3.16}$$

This projection is given in normalized image space and does not take into account the intrinsic camera parameters. It can not directly be used with the displacement error from Eq. (3.7). Looking back at Eq. (3.2), the following relation between normalized image coordinates (x_n, y_n) and centered pixel coordinates (x_c, y_c) is deduced:

$$x_c = -f_x \cdot x_n \qquad y_c = f_y \cdot y_n \qquad \text{with} \quad x_n = \frac{p_x}{p_z},\ y_n = \frac{p_y}{p_z}. \tag{3.17}$$

The displacement error in pixel coordinates can be expressed as:

$$\begin{aligned} u_{x,y} &= x_c - \hat{x}_c = -f_x(x_n - \frac{\hat{p}_x}{\hat{p}_z}), \\ v_{x,y} &= -(y_c - \hat{y}_c) = -f_y(y_n - \frac{\hat{p}_y}{\hat{p}_z}). \end{aligned} \tag{3.18}$$

After replacing x_n and y_n in Eq. (3.16) and inserting the result into Eq. (3.7), the left side of the equation is reordered to isolate the unknown motion parameters. This results in a linear equation with $\mathbf{a} = (a_0, a_2, \ldots a_5)$ and six unknown motion parameters $\Delta\mathbf{t}$ and $\Delta\mathbf{r}$:

$$\mathbf{a}\begin{pmatrix}\Delta\mathbf{r}\\ \Delta\mathbf{t}\end{pmatrix} = \hat{\mathbf{I}} - \mathbf{I}. \tag{3.19}$$

At each position (x, y), the values in $\mathbf{a}$ are computed using $\hat{p}_z = \hat{d}_{x,y}$ from the rendered depth map $\hat{\mathbf{D}}$, along with the intrinsic camera parameters [EG97]:

$$\begin{aligned} a_0 &= g_x(\hat{x}_n(\hat{y}_n - b_y)) + g_y(\hat{y}_n(\hat{y}_n - b_y) - b_z) & a_3 &= g_x(-1/\hat{p}_z) \\ a_1 &= g_y(\hat{y}_n(b_y + \hat{y}_n)) + g_x(\hat{x}_n(b_x - \hat{x}_n) - b_z) & a_4 &= g_y(-1/\hat{p}_z) \\ a_2 &= g_x(\hat{y}_n - b_y) + g_y(b_x - \hat{x}_n) & a_5 &= g_x(\hat{x}_n/\hat{p}_z) + g_y(\hat{y}_n/\hat{p}_z). \end{aligned} \tag{3.20}$$

The following parameter substitutions are utilized in these calculations:

$$g_x = -f_x\frac{\partial\hat{\mathbf{I}}}{\partial x}, \quad g_y = -f_y\frac{\partial\hat{\mathbf{I}}}{\partial y}, \quad b_x = \frac{t_x}{\hat{p}_z}, \quad b_y = \frac{t_y}{\hat{p}_z}, \quad b_z = 1 - \frac{t_z}{\hat{p}_z}.$$

Here, g_x, g_y, $\hat{x}_n$, $\hat{y}_n$ and $\hat{p}_z$ all depend on the pixel coordinates (x, y), but the additional indexing is omitted for readability. An equation is added to the equation system for every pixel inside the object silhouette in $\hat{\mathbf{I}}$ and then solved using IRLS. If the movement between $\mathbf{I}$ and $\hat{\mathbf{I}}$ is small and the renderer can accurately replicate the camera image with all scene parameters known, this model alone could be sufficient for tracking the object.

3.2.2 Guided Thresholds for Edge Detection

To accommodate the varying characteristics of the images $\mathbf{I}$ and $\hat{\mathbf{I}}$, binary edge maps are extracted from both images. Subsequently, a 7×7 box filter is applied to spatially extend the edges, which helps obtain more stable gradients [EFR08]. Gradients are computed in the x and y directions, as well as temporally in the t direction between the images.

A two-step method is employed to extract approximately the same edges in the images $\mathbf{I}$ and $\hat{\mathbf{I}}$. First, 3×3 Sobel filters are applied to $\hat{\mathbf{I}}$, and the magnitude of the resulting gradient is binarized using a fixed threshold. The padded bounding box around the rendered object is subdivided into a $d_n \times d_n$ grid of equal-sized rectangular cells. Here,

$d_n = 16$ at the lowest image pyramid level, halving at each subsequent level. The number of edge pixels is counted within each tile.

When detecting edges in the camera image $\mathbf{I}$, the same grid is used; however, an individual binarization threshold is determined for each cell to achieve approximately the same number of edge pixels as those detected in the reference grid. Thereby $\hat{\mathbf{I}}$ guides the edge extraction from $\mathbf{I}$. For example in $\hat{\mathbf{I}}$ the object may appear either textured or untextured, shaded or unshaded. In the case of projector-based tracking, it might display a simulation of the projection. Figure 3.3 illustrates guided thresholding for the latter case. The adaptive thresholds ensure that both images contain a similar amount of information, even in the presence of contrast and illumination variations. This binarization process results in a comparable value range for the optical-based pose estimation.

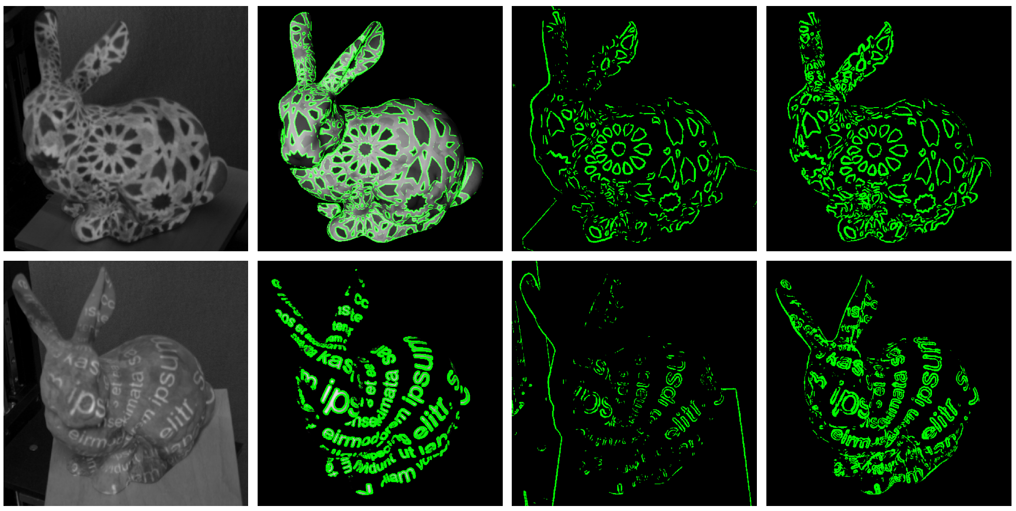


Figure 3.3: Visualization of the effect of adaptive thresholding. The figure shows the camera image, the simulated projection with detected edges using a global threshold, the edges from the camera image using the same global threshold, and the edges from the camera image using adaptive thresholds (f.l.t.r.). The bottom row shows another example. *(Reprinted from [GHE19], © 2019 IEEE)*

3.2.3 Discrete Contour-based Optimization

Inspired by the edge-based RAPID tracker [HS90] and recent extensions [HZSQ20], a fast tracking method based on the object silhouette complements the dense approach. The rendered depth map $\hat{\mathbf{D}}$ allows generating a silhouette mask from which m 2D boundary points $\mathbf{e}_i$ and the corresponding 3D points $\hat{\mathbf{p}}_i^e$ are extracted at regular intervals of t_d in 2D space.

Next, correspondence hypotheses are searched along m scanlines $l_i(t) = \mathbf{e}_i + t\mathbf{s}_i$, where $t \in \{-t_l, \cdots, t_l\}$. The scanline follows the 2D unit vector $\mathbf{s}_i$, the locally approximated normal vector to the projected contour at a given point. Intensity values are sampled from $\mathbf{I}$ along l_i and convolved with rotated 5×5 extended Sobel kernels [ZH87] to extract edges with similar orientation to the projected edge. All locations where the convolved value is a local maximum along the scanline and larger than a small threshold t_e are stored as edge hypothesis points $\mathbf{h}_{i,j}$. The error function E to be minimized is given by:

$$E(\Delta\mathbf{R}, \Delta\mathbf{t}) = \sum_{i=0}^{m} \omega(r_i)((\mathbf{h}_i^* - \pi(\mathbf{K}\mathbf{p}_i^e)) \cdot \mathbf{s}_i), \tag{3.21}$$

where the pose delta transforms $\hat{\mathbf{p}}_i^e$ to $\mathbf{p}_i^e$ (Eq. (3.4)) and $\mathbf{h}_i^*$ is the hypothesis with the smallest line-to-point distance to the projected contour. Both $\mathbf{s}_i$ and the selection of $\mathbf{h}_i^*$ are influenced by the contour's projection, which in turn depends on the pose. During optimization, the normal distance between the edge hypotheses from the image and the projected contour is minimized, facilitating rapid convergence of the pose estimation.

As in Sec. 3.2.1, a first-order Taylor approximation linearly relates the pose offset to the displacement error, computed along the normal direction:

$$\mathbf{a}_i \begin{pmatrix} \Delta\mathbf{r} \\ \Delta\mathbf{t} \end{pmatrix} = (\tilde{\mathbf{h}}_i^* - \tilde{\mathbf{p}}_i) \cdot \mathbf{s}_i. \tag{3.22}$$

The coefficients, $\mathbf{a}_i$, are similar to those in Eq. (3.20), with (g_x, g_y) replaced by $\mathbf{s}_i = (s_{i,x}, s_{i,y})$. Here, $\tilde{\mathbf{p}}_i = \pi(\hat{\mathbf{p}}_i^e)$ and $\tilde{\mathbf{h}}_i^* = \pi(\mathbf{K}^{-1}[h_{i,x}^*, h_{i,y}^*, 1]^T)$ are represented in normalized image coordinates. An overdetermined linear equation system is solved for the pose parameters using IRLS to stabilize the estimation against outliers. The weighting function $\omega(x)$ incorporates the robust Charbonnier penalty [SRB10], which adjusts based on the hypothesis residual r_i from the previous IRLS iteration:

$$\omega(x) = 1/\sqrt{x^2 + \epsilon^2}, \tag{3.23}$$

with $\epsilon = 0.001$. Three weight updates generally stabilize the estimate against outliers.

3.2.4 Implementation Details

The 6D object tracking involves straightforward concepts to ensure stability.

First, an **image pyramid** enhances tracking robustness and reduces computational complexity. The input image is downsampled D_P times by a factor of two, creating a pyramid with progressively lower resolutions. At each pyramid level, the renderer generates the image at the corresponding resolution, with lower-resolution estimates initializing higher resolutions. Larger pose offsets are corrected at lower resolutions, while finer adjustments are made at higher resolutions. Each pyramid stage consists of four to six repetitions of the discrete method (Sec. 3.2.3), followed by refinement using the dense method (Sec. 3.2.1) to compensate for mismatches along the scanlines. IRLS is applied during each optimization with three weight updates. The off-screen renderer and hypothesis selection for discrete optimization are applied at the beginning of each stage, after which only the extracted 3D contour is transformed.

Second, to accelerate convergence, certain **degrees of freedom are fixed** during the optimization process. At the highest pyramid level, only the translation in the image plane (Δt_x, Δt_y) is estimated, similar to 2D template matching. Subsequently, scale and rotation in the image plane (Δt_z, Δr_z) are corrected before estimating all six degrees of freedom at the lower pyramid stages.

Third, a **smoothness regularization** with a damping factor is incorporated into the system to smooth rotations over time and reduce frame-to-frame jitter. An equation is added to the linear system for each rotation component of the relative rotation matrix, linearized using the small-angle assumption, to penalize large deviations between the current and the previous estimate. As a result, the system ensures more stable and continuous pose estimation. The damping factor balances responsiveness to new data and smooth motion trajectories.

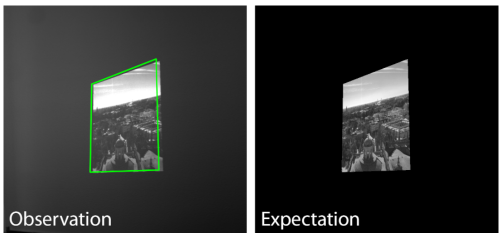
Shader-based performance optimization The dense optimization, including the technique discussed in Sec. 3.2.1, along with the projector-based approaches detailed in the following section, creates a pixel-based linear equation system. All operations are executed on a per-pixel basis, allowing for efficient processing using custom **compute shaders** on the GPU. This encompasses radial distortion compensation, guided edge detection, box filtering, gradient calculation, and linear equation formulation.

On the GPU, the coefficients of these equations are combined using matrix reduction. Each observation contributes to a coefficient vector $\mathbf{a}$ and a residual b. For each equation, the upper triangular part of the symmetric 6×6 matrix $\mathbf{a}^T\mathbf{a}$ and the 6×1 vector $\mathbf{a}^T b$ are computed. This results in 27 values stored in an 8×1 RGBA texture patch.

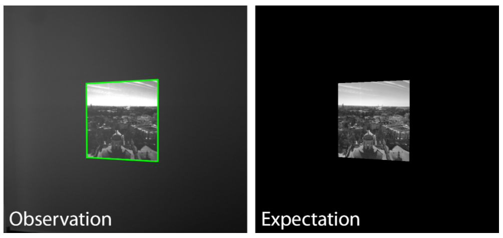
The construction of $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}^T\mathbf{b}$ (see Eq. (3.9)) is achieved by accumulating the observations through parallelized matrix reduction in multiple passes via a custom compute shader. Once the reduction is complete, only a single 32-byte texture patch is transferred from the GPU to the CPU for solving the system $\mathbf{Ax} = \mathbf{b}$, significantly reducing data transfer overhead.

3.3 Projection-Distortion-based Registration

The previously presented approach enables real-time object tracking, but encounters significant challenges in SAR scenarios. Projected textures generate strong gradients that deviate from the motion model's expected behavior, as they slide over the object surface. This undermines the robustness of traditional tracking methods and leads to pose drift [AOT10]. To overcome this limitation, this section extends the approach for projector-camera systems by including simulated projections in the rendering and extending the



(a) No alignment



(b) Corrected result

Figure 3.4: Projection-distortion-based plane tracking. An image is projected onto a plane and captured (observation). Mismatched plane parameters (a) cause misalignment between the rendered simulation (expectation) and observation (green frame). After alignment (b), the images match, resulting in an undistorted projection. *(Reprinted from [GE18], © 2018 IEEE)*

motion model to incorporate the projection. Unlike a conventional camera, the projector emits light onto the object, acting as an inverse camera [MT12]. In the following, scene information is derived by analyzing the geometric relationship between the simulated appearance of the projection on the known object and the captured image.

3.3.1 Projection Plane Tracking

Knowing the distance and orientation of the projection plane relative to the projector allows the projection to be displayed undistorted and in the desired size. Pose estimation for planes forms the basis of the full 6D model presented in the next section. It can be used to eliminate keystone effects and is essential for AR applications, such as integrating localized projective interfaces on flat surfaces [AMMA16, PRH$^+$22]. For example, when using a projective interface with hand gestures, the hand's position must be determined relative to the surface. If the interface in the virtual scene aligns with the real-world surface, the hand's position or distance to the plane can be computed [PRH$^+$22].

The presented approach distinguishes itself by not relying on the four corner points of the image [SSM01] or discrete feature points to match the projected image and its reflection for homography estimation. This is advantageous because the outer corner points may not be visible in the case of a dark background, such as when projecting only text. In contrast, relying on feature points, as done in previous studies [FJ07, RKK16], requires a sufficient number of points to achieve a good match, and lighting conditions can affect the quality of corresponding points. Moreover, a radiometric calibration of the projector and camera response is necessary. Unlike previous work [AOT10], the proposed method does not require alignment of the projection to a printed element on the projection plane; instead, it aligns the projection to its own reflection, resulting in high flexibility without the need for additional markers. Figure 3.4 illustrates the plane registration.

It is assumed that the projector is directed toward a flat surface, with known intrinsic

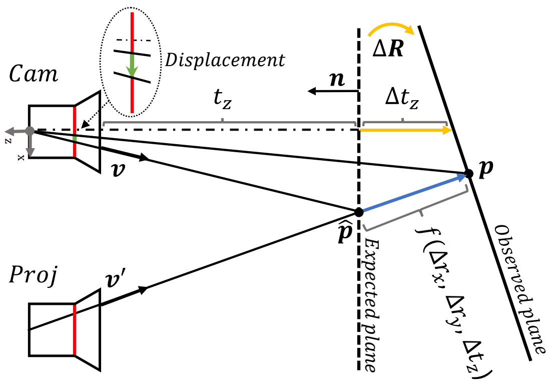


Figure 3.5: Geometry of a projector-camera system during projection plane tracking. The expected plane differs from the observed plane by a certain transformation ($\mathbf{\Delta R}$, $\mathbf{\Delta t_z}$). The displacement in image space is related to $\mathbf{v}'$ and the result of f. *(Reprinted from [GE18], © 2018 IEEE)*

parameters for both the projector and the camera, along with the extrinsic parameters between the two devices. The camera and the projector are modeled using the pinhole camera model, with the projector treated as an inverted camera. The camera image is denoted by $\mathbf{I}$, while $\hat{\mathbf{I}}$ represents a simulated camera image, which includes a virtual plane and a simulated projection carried out via projective texture mapping.

The projection plane is defined by a normal vector $\mathbf{n}$ and a vector $\mathbf{t} = [0, 0, t_z]^T$, which represents the point where the camera's optical axis intersects the plane. Three degrees of freedom must be estimated: the distance t_z and two rotation angles, r_x and r_y, which determine the orientation of the plane's normal vector.

The solution is approached iteratively, by estimating a motion offset Δt_z, Δr_x, Δr_y starting from an initial estimate. It is assumed that the movement of the projection plane is small, allowing for the change in orientation to be approximated using a linearized rotation matrix with two degrees of freedom (as shown in Eq. (3.5) with $\Delta r_z = 0$). The translation offset Δt_z represents the difference added to t_z to hit the new plane.

Motion model To define a viewing ray for a pixel $(\hat{x}, \hat{y})$ in the simulated image $\hat{\mathbf{I}}$, the equation

$$\mathbf{v} = \left[-\frac{\hat{x} - u_x}{f_x}, \frac{\hat{y} - u_y}{f_y}, -1 \right]^T \tag{3.24}$$

is used, where f_x and f_y are the scaled horizontal and vertical focal length, and u_x and u_y are the pixel positions of the principal point. The intersection point $\hat{\mathbf{p}}$ between a viewing ray and a plane can be determined using the equation

$$\hat{\mathbf{p}} = \mathbf{v}\frac{\mathbf{t}^T\mathbf{n}}{\mathbf{v}^T\mathbf{n}}, \tag{3.25}$$

where $\mathbf{n}$ is the normal vector and $\mathbf{t}$ is a vector from the camera center to the intersection point of the camera's optical axis with the plane. By using the extrinsic parameters of the projector-camera system, the intersection point can be transformed to the projector coordinate system and back-projected onto the image plane of the projector, resulting in the viewing ray $\mathbf{v}'$ of the projector.

Moving a point $\hat{\mathbf{p}}$ by a motion offset causes a translation by $f(\Delta t_z, \Delta r_x, \Delta r_y)$ in the direction of the projector's light beam. The resulting new plane is intersected at point $\mathbf{p}$, which can be computed using the equation

$$\mathbf{p} = \hat{\mathbf{p}} + \mathbf{v}' f(\Delta t_z, \Delta r_x, \Delta r_y), \tag{3.26}$$

where $\mathbf{v}'$ is the view vector of the projector in the camera coordinate system. The function f can be calculated using the formula for a line-plane intersection and is expressed as

$$f(\Delta t_z, \Delta r_x, \Delta r_y) = \frac{(\mathbf{\Delta t} - (\mathbf{p} - \mathbf{t}))^T \mathbf{\Delta R n}}{\mathbf{v}'^T \mathbf{\Delta R n}}. \tag{3.27}$$

After transforming the projection plane, $\mathbf{p}$ projects back to the camera image at (x, y) following Eq. (3.17). The relationship between the resulting displacement, $\mathbf{v}'$, and the outcome of f is shown in Fig. 3.5.

Optical flow extension Similar to 2D-3D tracking, the motion model is integrated into an optical flow scheme to minimize displacement errors $(u_{x,y}, v_{x,y})$ between the captured image $\mathbf{I}$ and the simulated image $\hat{\mathbf{I}}$. A first-order Taylor expansion is used to express the projection of the point onto the sensor:

$$\begin{aligned} \frac{p_x}{p_z} &\approx \frac{\hat{p}_x}{\hat{p}_z} + \frac{b}{\hat{p}_z(\mathbf{n}^T\mathbf{v}')}(\Delta r_x o_y - \Delta r_y o_x + \Delta t_z n_z) \\ \frac{p_y}{p_z} &\approx \frac{\hat{p}_y}{\hat{p}_z} + \frac{c}{\hat{p}_z(\mathbf{n}^T\mathbf{v}')}(\Delta r_x o_y - \Delta r_y o_x + \Delta t_z n_z), \end{aligned} \tag{3.28}$$

with

$$b = v'_x - v'_z \frac{\hat{p}_x}{\hat{p}_z}, \quad c = v'_y - v'_z \frac{\hat{p}_y}{\hat{p}_z}, \quad o_x = \hat{p}_x n_z - (\hat{p}_z - t_z) n_x, \quad o_y = \hat{p}_y n_z - (\hat{p}_z - t_z) n_y.$$

When Eq. (3.7) and Eq. (3.28) are combined, a linear equation with three unknown motion parameters is obtained. This equation is given by:

$$-o_y d \Delta r_x + o_x d \Delta r_y + -n_z d \Delta t_z = \hat{\mathbf{I}} - \mathbf{I}, \tag{3.29}$$

with

$$d = \frac{1}{\hat{p}_z(\mathbf{n}^T\mathbf{v}')} \left(\frac{\partial \hat{\mathbf{I}}}{\partial x} f_x b + \frac{\partial \hat{\mathbf{I}}}{\partial y} f_y c \right).$$

The equation is formulated for each pixel of the projection, forming an overdetermined system, which can be solved using a least squares method or a robust estimator. The variables on the left side of Eq. (3.29) depend on the image location (x, y), with the indexation omitted for better readability.

3.3.2 Projection-Distortion-based Object Tracking

The previous section's model has several limitations. Firstly, it only allowed registration with three degrees of freedom, and secondly, it constrained the registered object's shape to a plane. While it ensured that the projected image appeared undistorted and maintained a constant specified size, it lacked the visual cues needed to determine how the content was rotated or shifted within the plane.

These limitations can be overcome by considering the more general case of 6D shader lamps [RWLB01]. A texture $\mathcal{T}$ is mapped to a mesh $\mathcal{M}$, the image is rendered from the projector's point of view, and then projected onto the object. This section focuses on updating the 6D pose (and thus the projection) as the object moves. This ensures that the projection remains consistent with the geometry and that the object in the camera image appears as if it has been textured with computer graphics.

The general idea of projection-distortion-based 6D tracking is that a local plane approximates the local environment of a point, and that a tracking model is defined that is valid on locally smooth surface patches. Special treatment is needed at steep edges, where this assumption does not hold. Each equation in the system applies to a local point in the image, providing information about the three degrees of freedom of the local plane. By combining these equations, all six degrees of freedom can be solved. This is similar to an earlier multi-camera shape refinement method, which, however, does not solve for the 6D pose or consider projector-camera systems [SEG01].

Motion model Each point $\hat{\mathbf{p}} = [\hat{p}_x, \hat{p}_y, \hat{p}_z]^T$ on the object surface, along with its corresponding surface normal $\mathbf{n}$, defines a tangent plane. The extrinsic parameters of the projector-camera system transform $\hat{\mathbf{p}}$ into the projector's coordinate system. Projecting $\hat{\mathbf{p}}$ back onto the projector's image plane yields the direction $\mathbf{v}'$ of the projected ray.

The offset $\Delta\mathbf{t}$ represents the change in the object's translation relative to its center $\mathbf{t}$. The rotation is described by the linearized rotation matrix $\Delta\mathbf{R}$ (Eq. (3.5)), with three degrees of freedom around the object center.

Moving the object results in a movement of the projection on a point $\hat{\mathbf{p}}$ by the distance d in direction of the projector's light beam $\mathbf{v}'$, yielding $\mathbf{p} = \hat{\mathbf{p}} + \mathbf{v}'d$ similar to Eq. (3.26).

A plane is generally defined by all 3D points $\mathbf{x}$ that fulfil the equation $(\mathbf{x} - \mathbf{p_0}) \cdot \mathbf{n} = 0$, where $\mathbf{p_0}$ is an arbitrary point on the plane. When a 3D transformation is applied to the object and consequently to a local plane, the resulting equation is:

$$(\mathbf{x} - (\Delta\mathbf{R}(\hat{\mathbf{p}} - \mathbf{t}) + \mathbf{t} + \Delta\mathbf{t}))^T \Delta\mathbf{R}\mathbf{n} = 0. \tag{3.30}$$

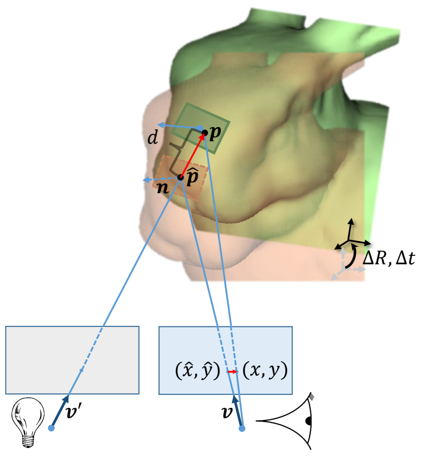


Figure 3.6: Geometry of a projector-camera system during 6D object tracking. The expected object pose (orange) differs from the observed object pose (green). The transformation ($\Delta\mathbf{R}$, $\Delta\mathbf{t}$) is estimated from the displacement in image space, which is related to $\mathbf{v}'$ and d. *(Reprinted from [GHE19], © 2019 IEEE)*

The intersection with the transformed plane and the viewing ray can be found by inserting $\mathbf{x} = \hat{\mathbf{p}} + \mathbf{v}'d$ in Eq. (3.30) and solve the equation for d.

$$d = \frac{(\Delta\mathbf{R}(\hat{\mathbf{p}} - \mathbf{t}) + \Delta\mathbf{t} - (\hat{\mathbf{p}} - \mathbf{t}))^T \Delta\mathbf{R}\mathbf{n}}{\mathbf{v}'^T \Delta\mathbf{R}\mathbf{n}} \tag{3.31}$$

Figure 3.6 provides a visualization of this principle. As the object moves, the projector ray aimed at hitting the object at point $\hat{\mathbf{p}}$ instead intersects it at an approximate location $\mathbf{p}$, which is directly related to the movement of the point in the image space.

Optical flow extension In contrast to the plane tracking case, where three degrees of freedom are estimated, linearizing the projection of a point now requires considering all six unknowns. This is because the coordinate systems of the object and the local plane are not aligned, so the plane's movement affects both $\Delta\mathbf{r}$ and $\Delta\mathbf{t}$ in all components. In the linearized projection, $\mathbf{h} = \mathbf{p} \times \mathbf{n}$ denotes the cross product of $\mathbf{p}$ and $\mathbf{n}$:

$$\begin{aligned} \frac{p_x}{p_z} &\approx \frac{\hat{p}_x}{\hat{p}_z} + \frac{b}{\hat{p}_z(\mathbf{n}^T\mathbf{v}')}(\Delta\mathbf{r}^T\mathbf{h} + \Delta\mathbf{t}^T\mathbf{n}) \\ \frac{p_y}{p_z} &\approx \frac{\hat{p}_y}{\hat{p}_z} + \frac{c}{\hat{p}_z(\mathbf{n}^T\mathbf{v}')}(\Delta\mathbf{r}^T\mathbf{h} + \Delta\mathbf{t}^T\mathbf{n}). \end{aligned} \tag{3.32}$$

Motion compensation is expressed as a system of linear equations, as in Eq. (3.19), but with different coefficients, $\mathbf{a}$ (Eq. (3.33)). These equations describe the relationship between projection distortion and changes in the object's position and orientation.

$$\mathbf{a} = \left(\frac{1}{\hat{p}_z(\mathbf{n}^T\mathbf{v}')} \left(-f_x b \frac{\partial \hat{\mathbf{I}}}{\partial x} + f_y c \frac{\partial \hat{\mathbf{I}}}{\partial \hat{y}} \right) \right) \begin{pmatrix} \mathbf{h} \\ \mathbf{n} \end{pmatrix}^T \tag{3.33}$$

Limitation due to aperture problem Ambiguities in pose estimation can arise for certain object shapes because of the aperture problem in optical flow [BB95]. For instance, a perfectly spherical object illuminated by a projector will not exhibit any visible change in the camera image during rotation about its center, thereby providing no rotational information. Similarly, as previously discussed, a flat surface can only provide information about three degrees of freedom; in-plane translation and rotation cannot be distinguished from projection distortion. A cylindrical object, on the other hand, can undergo translation along its axis without altering the local appearance of the projection. In contrast, more complex object shapes may offer varied cues about motion in different regions, enabling the unique determination of all motion parameters. Incorporating projection-independent terms (see Sec. 3.3.3), can help mitigate such ambiguities.

3.3.3 Implementation Details

The projection is displayed in fullscreen mode on the projector. The projected image displays the object from the point of view of the projector. Assuming the object's pose is known, the projection perfectly maps onto its surface. The projection is rendered in an offscreen buffer, and a depth-buffer-based mask removes artifacts along the object's contour by applying a morphological erosion to the object silhouette [KWW17]. Before projecting the image, the radial lens distortion of the projector is compensated by pre-distorting the image.

The projected image, along with the depth buffer information, is stored as a texture and used to simulate the projection from the camera's perspective. The projector is simulated using projective texture mapping [Eve01], incorporating shadow maps to simulate self-occlusions. Additionally, the z-buffer and surface normals are required for pose estimation. The rendered camera image is decomposed into two components—one showing the object surface with ambient illumination $\hat{\mathbf{I}}_\mathbf{A}$ and the other showing the reflected projector light $\hat{\mathbf{I}}_\mathbf{P}$—whose sum simulates the appearance of the projection on the object.

The framework for object tracking generally follows the approach described in Sec. 3.2.4. The primary difference is the use of adapted equations to minimize the image difference between $\hat{\mathbf{I}}_\mathbf{P}$ and $\mathbf{I}$, pre-processed with guided thresholding (Fig. 3.3) to extract the relevant edges. For projection-based 6D tracking, three filters are applied to decide whether an equation is added to the system or not:

1. Equations are created only where depth values exist in $\hat{\mathbf{I}}_\mathbf{P}$.

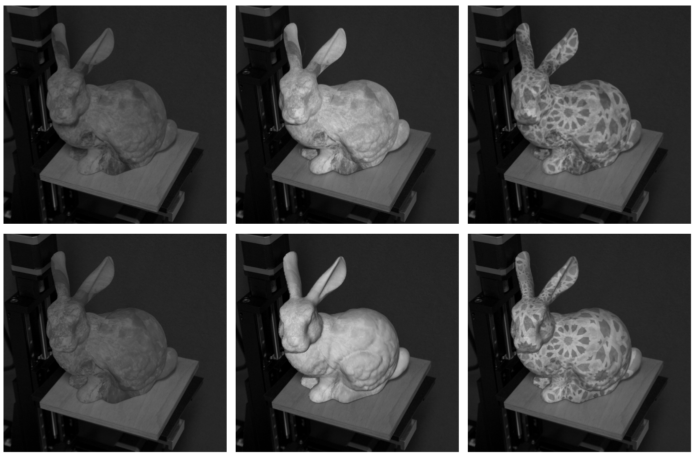


Figure 3.7: Initial illumination estimation. Top: Camera images. Bottom: Images with the textured bunny model rendered as an overlay. The estimated ambient illumination (left) and projector brightness (middle) enable the simulation of a texture projection (right). Images are displayed in grayscale, as the optimization operates in grayscale. *(Reprinted from [GHE19], © 2019 IEEE)*

2. Pixels near the object's geometric borders are ignored due to the invalidity of the locally planar assumption. Depth discontinuities, including the silhouette, are detected in $\hat{\mathbf{D}}$ using a 5 cm threshold between pixels (as used in the experiments). The edges are extended using a 7×7 filter, and values within this mask are excluded.
3. Pixels at positions where the angle between the projector's viewing ray and the surface normal exceeds 70 degrees (as used in the experiments) are ignored, as the intensity of the projection diminishes in these areas.

Illumination and projector brightness initialization In order to enhance the accuracy of pose estimation, an initialization step can be performed to estimate the approximate scene illumination and the brightness of the projection. This helps make the simulated images more similar to the camera image. Since tracking compares adaptive edge images, an approximate illumination model based on grayscale images is sufficient.

The object's initial position is determined using contour-based methods (see Sec. 3.2), with the projector turned off. The Lambert illumination model approximates the brightness of points on the object. The goal is to minimize the absolute image difference between the simulation $\hat{\mathbf{I}}_{\text{ProjOff}} = (\tilde{i}_{x,y}) \in \mathbb{R}^{w \times h}$ and the camera image $\mathbf{I}$.

$$\tilde{i}_{x,y} = (c_D \cdot \max(\mathbf{n}_{x,y}^T \mathbf{l}, 0) + c_A) \cdot (c_{\text{Off}} + c_{\text{Obj}}) \tag{3.34}$$

Once the object is registered, the surface normal $\mathbf{n}$ is known for each point. Additionally, the object's approximate color c_{Obj} (or texture intensity) is assumed to be predetermined.

The unknown parameters—ambient intensity c_A, diffuse reflection coefficient c_D, light direction $\mathbf{l}$, and brightness offset c_{Off}—are estimated iteratively. In each step, a system of linear equations (one per pixel on the object's surface) is solved, and the simulation is updated until convergence.

To estimate the brightness of the projector, a white texture is projected onto the object. The resulting intensity in the simulated image, denoted by $\hat{\mathbf{I}}_{\text{ProjOn}} = (\bar{i}_{x,y}) \in \mathbb{R}^{w \times h}$, is affected by the cosine of the angle between the surface normal and the direction of the projector light $\mathbf{v}'$. It is also linearly attenuated by the distance $\|\hat{\mathbf{p}}\|$ between the point and the projector.

$$\bar{i}_{x,y} = \tilde{i}_{x,y} + \frac{c_P \cdot \max(\mathbf{n}_{x,y}^T \mathbf{v}'_{x,y}, 0)}{\|\hat{\mathbf{p}}\|_{x,y}} \tag{3.35}$$

The brightness of the projector c_P is again estimated by iteratively minimizing the absolute image difference between $\hat{\mathbf{I}}_{\text{ProjOn}}$ and $\mathbf{I}$ using linear equation systems. Sample outcomes of the illumination estimation are shown in Fig. 3.7.

Combining projection and contour-based optimization The direct pose offset computation using a linear system can be easily extended by incorporating equations from other motion models with the same unknown parameters, thereby introducing additional constraints to the optimization. One such approach integrates the projection-based method with contour-based tracking to ensure that object contours are considered during tracking. A mask that contains the spatially extended object outline and geometric edges is available for filtering the projection-related equations (see Step 2 above). Inside this mask, the equations of the dense contour-based model (Sec. 3.2.1) are used to minimize the image difference between $\hat{\mathbf{I}}_{\mathbf{A}}$ and $\mathbf{I}$. The guided thresholding operation is repeated, using only the region in the mask to extract appropriate edges from the image.

The usefulness of the combined approach depends on the visibility of the object contours in the image. It is influenced by ambient light and the character of the projected content. Especially if the projector illuminates only a part of the object, the contour provides useful information. Also, the object contour helps prevent ambiguous configurations where a projection on the object appears similar in the camera frame under different object poses.

3.3.4 Extension for Extrinsic Camera Pose Refinement

Accurate calibration of the intrinsic and extrinsic parameters of the stereo system is crucial for projector-camera-based tracking. When a known 3D object is presented to the camera with a known pose $\mathbf{R}, \mathbf{t}$, several conditions must be met to ensure accurate tracking. First, the rendered object overlay in the camera image must align with the image of the real-world object (i.e., intrinsic camera calibration). Second, the rendered texture projected onto the object must align perfectly with its shape. Third, the simulated projection must align with the observed projection in the camera image.

A well-known method for calibrating a projector-camera system is the approach by Moreno et al. [MT12]. This technique involves projecting a sequence of gray-code patterns onto a static checkerboard and repeating the procedure across various checkerboard orientations. The process generates hundreds of images for a single calibration. The method provides a good estimation of the parameters but is also very time-consuming. If the extrinsic parameters of the projector-camera system change, such as by minimal displacements of the two sensors in a dynamic system, the process must be repeated.

Resch et al. [RNK+15] introduced a fast, model-based re-calibration technique that builds on an initial calibration estimate. Their method reconstructs a point cloud of a known object using gray-code patterns and iteratively refines the calibration to align the 3D reconstruction with the shape of the object's 3D mesh. In contrast, the analysis-by-synthesis approach in this thesis refines calibration parameters in image space. It iteratively aligns the captured projection with the simulated target appearance, seamlessly integrating the shader lamp projection into the process.

The starting point is a known 3D object accurately localized relative to the camera (e.g., using the method from Sec. 3.2), along with a precise initial calibration of the projector and camera, and an approximate extrinsic calibration. The equations from Sec. 3.3.2 form the basis of the updated motion model. The updated model includes a translation offset of $\Delta\mathbf{t}_P$ and a rotation offset of $\Delta\mathbf{R}_P$ for the projector relative to the camera. Unlike Fig. 3.6, the movement between $\hat{\mathbf{p}}$ and $\mathbf{p}$ now results from a rotation and translation of the projector's viewing ray, $\mathbf{v}'$.

$$\mathbf{p} = \hat{\mathbf{p}} + \Delta\mathbf{t}_P + d\Delta\mathbf{R}_P\mathbf{v}' \tag{3.36}$$

As per Eq. (3.31), the distance d can now be expressed as:

$$d = \frac{(\hat{\mathbf{p}} - \mathbf{t}_p - \Delta\mathbf{t}_P)^T\mathbf{n}}{(\Delta\mathbf{R}_p\mathbf{v}')^T\mathbf{n}}. \tag{3.37}$$

The relation of the unknown motion parameters to the optical flow constraint is described by a linear system, as described in Eq. (3.19), with the coefficients $\mathbf{a} = (a_0, a_2, \dots a_5)$.

$$\mathbf{a} = \frac{f_x}{\hat{p}_z}\frac{\partial\hat{\mathbf{I}}}{\partial x}\left(\mathbf{u_1} + \frac{b}{\mathbf{n}^T\mathbf{v}'}\begin{pmatrix}\mathbf{h}\\ \mathbf{n}\end{pmatrix}\right) - \frac{f_y}{\hat{p}_z}\frac{\partial\hat{\mathbf{I}}}{\partial y}\left(\mathbf{u_2} + \frac{c}{\mathbf{n}^T\mathbf{v}'}\begin{pmatrix}\mathbf{h}\\ \mathbf{n}\end{pmatrix}\right), \tag{3.38}$$

where:

$$\mathbf{u_1} = \left(v'_y\tfrac{\hat{p}_x}{\hat{p}_z}, \quad -v'_x\tfrac{\hat{p}_x}{\hat{p}_z} - v'_z, \quad v'_y, \quad -1, \quad 0, \quad \tfrac{\hat{p}_x}{\hat{p}_z}\right)^T$$
$$\mathbf{u_2} = \left(v'_y\tfrac{\hat{p}_y}{\hat{p}_z} + v'_z, \quad -v'_x\tfrac{\hat{p}_y}{\hat{p}_z}, \quad -v'_x, \quad 0, \quad -1, \quad \tfrac{\hat{p}_y}{\hat{p}_z}\right)^T.$$

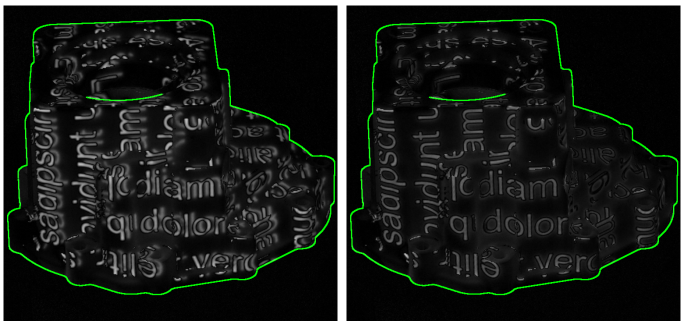


Figure 3.8: Extrinsic parameter refinement. Comparison of the alignment between the captured and simulated projections (shown as a rendered overlay), before (left) and after (right) refining the extrinsic projector-to-camera pose. The refinement improves alignment while keeping the object pose (indicated by green contour) fixed. The target object and its expected appearance are shown in Fig. 3.11. *(Reprinted from [GHE19], © 2019 IEEE)*

Above, b and c are computed as in Eq. (3.32) and $\mathbf{h} = -(\mathbf{v} \times \mathbf{n})$. Aside from the substitution of these equations, the registration pipeline (see Sec. 3.3.3) remains unchanged.

Objects for which the pose can be refined from the projection distortion without suffering from the aperture problem (e.g., spheres or flat surfaces are unsuitable) can be used as calibration targets. To support convergence, it is advisable that the object occupies a significant portion of the image and extends in the z-direction. Stable results can be obtained by capturing high-resolution images in an environment with reduced ambient light. The procedure is especially useful for *focus-free* projectors, such as laser scanning or laser + LCoS projectors, where the intrinsic parameters remain fixed independent of the setup, as these projectors do not require an adjustable physical focus [GH12].

Figure 3.8 illustrates the effect of extrinsic pose refinement. The initial calibration of the projector-camera system is performed using the method proposed by Moreno et al. [MT12]. Subsequently, the object is precisely registered using contour-based 6D pose refinement without active projection. With ambient light deactivated, a new texture is projected. The left side of Figure 3.8 shows the camera view, where the object contour is marked in green, and a rendered overlay displays the expected appearance. A clear deviation between expectation and observation is visible. After applying the refinement (right), the expected appearance aligns with the observation. This improved alignment enhances the projection's fit to the real object and stabilizes tracking, as the simulation more accurately predicts the actual image.

3.4 Experiments

In dynamic closed-loop shader lamp scenes, the projected image is influenced by the previously estimated pose, which affects subsequent camera frames. This hinders the recording of test sequences with moving objects and the evaluation of these sequences under varying parameters. Synthetic sequences enable repeatable tests with easily adjustable parameters. However, since these synthetic tests operate in an idealized environment and cannot model all physical parameters of light and materials, tests with real data are also essential. A motorized linear stage allows reproducible object movements, facilitating comparisons of various parameter settings against ground truth data.

3.4.1 Synthetic Evaluation of Projection-Distortion-based Registration

In this section, projection-distortion-based tracking is evaluated in a synthetic test environment that simulates projection onto the object and assesses tracking performance.

Setup and method A target object is placed in a 3D scene and a texture projection on the object is simulated. The object moves on a specific path and the initial pose and the corresponding projection are known. For the following frames, the pose to create the projection is estimated from a synthesized camera image showing the previous projection on the moved object. Thus, effects like drift over time can be simulated. The virtual projector-camera system uses calibration and illumination parameters from the real data test. The system is placed inside a textured, cube-shaped box, serving as a background. If the object moves, a part of the projected texture misses the target object and hits the background object, simulating the delay in the closed-loop approach (see Fig. 3.9). The rest of the projection is perspectively distorted by the object's surface.

The synthetic tests utilize the well-known Stanford Bunny model. Two different projective textures augment the model during the tests, simulating various spatial AR scenarios. Both textures evenly cover the object's surface with a detailed pattern (see Fig. 3.11 where the same textures are applied to different objects).

Texture 1 consists of white dummy text on a black background. Text projection plays a crucial role in AR assistance scenarios, where additional information must be placed on the object's surface.

Texture 2 features a colored geometric pattern, as it could be found in design mockups or art installations. It does not contain black areas, and the projector illuminates the entire object. This texture, applied to the bunny model, is shown in Fig. 3.1.

Additionally, the model itself is textured, as shown on the left side of Fig. 3.7. This introduces another difficulty since contours not related to the projection can appear in the adaptive edge image.

Two experiments are conducted with synthetic data to test the projector-based tracking. In each experiment, 100 different sequences of random movements are simulated in one iteration. The initial distance between the object and the camera is 0.7 meters. The initial orientation is chosen randomly for each sequence. The target pose differs randomly from

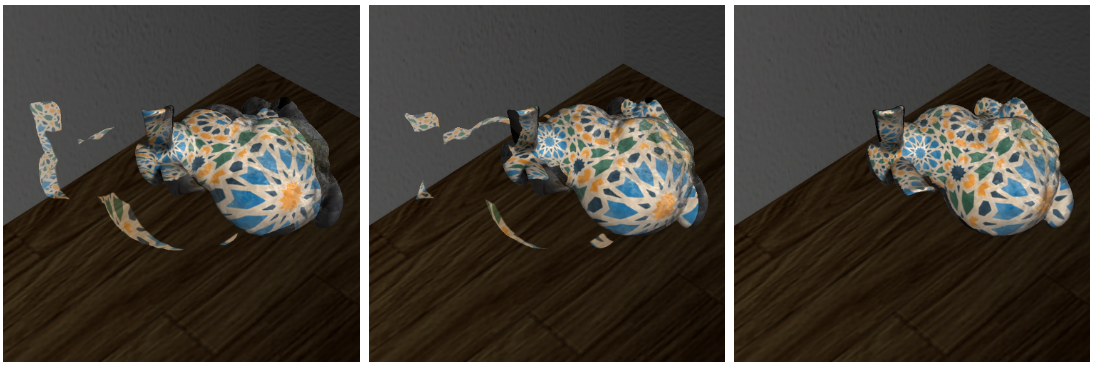


Figure 3.9: Example result of Experiment 2 and Texture 2. Initially (left), much of the projection misses the object. By the fifth frame (middle), the match improves. After 12 frames (right), the projection perfectly aligns with the object's shape. *(Reprinted from [GHE19], © 2019 IEEE)*

the initial pose, with the absolute sum of the translation offsets along the axes constrained to s_t cm and the absolute sum of the rotation offsets around the axes constrained to s_r degrees.

Experiment 1: Handling small pose offsets This experiment tests the tracking accuracy for small pose offsets between frames and a continuous motion. To move the object to the target pose, the movement is linearly interpolated over s_d frames. The root mean square error (RMSE) of the motion parameters is computed independently for each sequence to evaluate tracking quality. The translation error t_{err} (in millimeters) refers to the Euclidian distance, and the rotation error r_{err} (in degrees) is calculated as defined in Eq. (3.6).[1] Furthermore, a sequence is considered *valid* if the RMSEs of r_{err} and t_{err} are below 10 degrees and 10 mm, respectively. Sequences with higher error values indicate a drifting pose over time. The average error values across all *valid* sequences, those in which the algorithm successfully tracks the object to the end, are presented in Tab. 3.1. For each test, the number of *valid* sequences out of 100 is reported as well.

In the first iteration of the test, a movement of 60 degrees and 6 cm is distributed over 60 frames. In the second and third iterations, the movement speed is doubled by reducing the number of frames for the movement to 30 or 15 frames, respectively.

For Texture 1, the poses are highly precise, with the average RMSE staying below 1 mm and 1 degree across all movement speeds. Additionally, for slow and medium speeds, 100% of the sequences are *valid*. Errors only begin to accumulate in 2% of the sequences at the fastest movement speed. For Texture 2, tracking remains accurate, especially at slow and medium speeds. The increased texture detail results in slightly higher average

[1]For consistency, the metrics differ from [GHE19], leading to slightly higher error values.

Test			**Texture 1**			**Texture 2**		
s_t	s_r	s_d	t_{err}	r_{err}	$valid$	t_{err}	r_{err}	$valid$
6	60	60	0.15	0.38	100	0.47	0.95	96
6	60	30	0.20	0.47	100	0.60	1.26	94
6	60	15	0.26	0.71	98	1.10	2.11	75

Table 3.1: Tracking continuous movement with small pose offsets. The values for r_{err} (in degrees) and t_{err} (in millimeters) represent the mean of the RMSE calculated over all valid sequences.

RMSE values. At the highest movement speeds, errors become more pronounced, leading to a reduction of valid sequences. The bright background of the texture increases the likelihood that the algorithm, when using only the projection terms, will converge to a local minimum. As the object moves continuously, inaccuracies can accumulate over time. Larger relative movements can be mitigated by giving the algorithm more time to converge, as shown in the next experiment.

Experiment 2: Handling large pose offsets Experiment 2 tests the ability to handle larger pose offsets. As in the first experiment, a random target pose is calculated, but this time, the object abruptly jumps to the target pose. In the following, s_d frames are allowed for the tracker to converge to the visible pose. For each frame, an updated projector image is generated and virtually projected. Figure 3.9 shows input images during one successfully converging sequence. In this test, only the error of the last frame of each sequence is relevant. Therefore, r_{err} and t_{err} are averaged over those final errors and not over the RMSEs of the sequences.

Test			**Texture 1**			**Texture 2**		
s_t	s_r	s_d	t_{err}	r_{err}	$valid$	t_{err}	r_{err}	$valid$
1	10	20	0.09	0.25	100	0.42	0.62	95
2	20	20	0.18	0.35	94	0.42	0.57	83

Table 3.2: Convergence performance with large pose offsets. The values for r_{err} (in degrees) and t_{err} (in millimeters) represent the mean of the final errors calculated individually for each valid sequence.

As shown in Tab. 3.2, an abrupt movement larger than the largest movement between two frames in Experiment 1 can be compensated if the algorithm is allowed to iterate longer and has more time to converge. The relative pose offset in the first test is 2.5 times larger than the largest frame-to-frame movement in Experiment 1 and can be bridged for all trials for Texture 1 and 95% of the trials for Texture 2. The second row shows that the number of invalid tests increases when the relative offset exceeds a certain threshold. However, if convergence is achieved, the final error remains below 1 mm and 1 degree. Offsets of this magnitude can be avoided, for example, by initializing the pose with the projection deactivated.

3.4.2 Real-Data Evaluation of Projection-Distortion-based Registration

Tests with real data are indispensable, as synthetic tests operate in idealized environments and cannot fully model all physical parameters of light and materials. To enable precise comparisons against ground truth data, a motorized linear stage is used to produce reproducible object movements. Three approaches are evaluated under a shader lamp projection: (1) a projection-based algorithm, (2) a contour-based tracker that disregards the projection (Sec. 3.2), and (3) the combined optimization approach described in Sec. 3.3.3. A CPU implementation is used for evaluation, and the impact of running parts of the algorithm on the GPU, specifically on movement speed, is discussed.

Setup and method The experimental setup comprises a *Basler ace acA2440-75uc* camera equipped with a *Kowa LM12SC* objective and a *Cremotech Laser Beam Pro C200* projector. These devices are rigidly mounted on a tray with a baseline of approximately 15 cm. The system is calibrated using a classical method [MT12], achieving a stereo reprojection error of 0.489 pixels. This is followed by an iterative refinement of the extrinsic parameters, as described in Sec. 3.3.4. The distance between the projector and the target object is set to 0.7 meters.

The linear stage enables movement along all three coordinate axes. Although only translations are applied to the objects during the tests, all six degrees of freedom are estimated. Figure 3.10 illustrates the trajectory of the object's movement. A top-down view of the test setup, as well as images of the two target objects, each with two texture projections, are provided in Fig. 3.11. The test objects are 3D-printed, uniformly white spare parts of industrial workpieces.

Initially, the object is placed on the platform, and the scene illumination and projector brightness are estimated (Sec. 3.3.2). An accurate starting pose is then determined (Sec. 3.2) with the projector turned off. The shifting unit moves 5 cm along the x-axis, and pose estimation is performed again at this new position. The object is placed on the platform with its y-axis pointing upwards. The direction of the x-axis, combined with the up-vector, determines the orientation of the global coordinate system relative to the object's orientation.

In addition to interchanging the two textures from the synthetic test, tracking is evaluated both with and without ambient illumination. Ambient illumination enhances the visibility of the object contour, requiring greater robustness during pose estimation. It also enables the use of the combined tracking approach. The contour-based pose

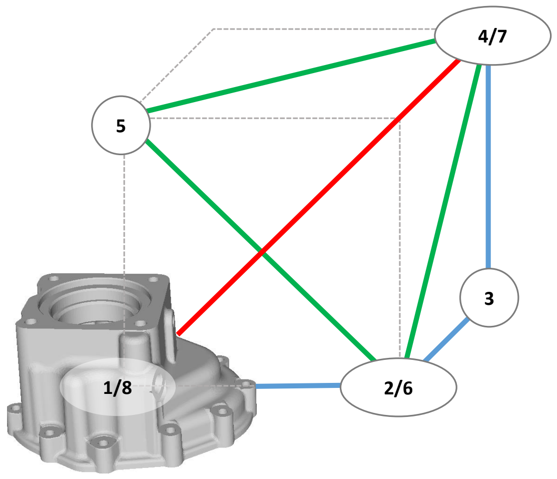


Figure 3.10: Trajectory of the object on the motorized linear translation stage. The object moves following increasing numbers. The side length of the cube is 5 cm. *(Reprinted from [GHE19], © 2019 IEEE)*

estimation is tested with and without shader lamp illumination. The speed of each axis is set to 5 mm per second. When two or three axes move simultaneously (as indicated by the green and red lines in Fig. 3.10), the object's speed increases accordingly. In this test, translation error is defined as the Euclidian distance between the estimated position and the trajectory, while the rotation error is the angular difference between the rotation matrices, as in the previous section.

Results The obtained error values are summarized in Tab. 3.3. Object 1 is successfully tracked throughout its entire trajectory using Texture 1, both with (**Proj.** + Amb.) and without (**Proj.**) ambient illumination. The RMSE for translation remains below 1 mm, even smaller than the 1.29 mm obtained using the contour-based reference approach (**Ref.**) without an active projection. Furthermore, without ambient illumination, the rotation error is less than 1 degree. It differs from the reference method by approximately half a degree. In the presence of ambient light, the rotation error fluctuates around 2 degrees but is significantly stabilized when the combined approach (**Comb.** + Amb.) is used.

Also, for Texture 2, tracking is successfully maintained throughout the complete sequence, both with and without ambient illumination. In the dark setting, tracking stability is comparable to that of Texture 1. With ambient illumination, diagonal movements introduce a slight rotational deviation, which is automatically corrected after reaching a turning point. The recovery is more effective with the approach using only the projection compared to the combined approach, which fails during the last movement. Even if the projection is slightly misaligned, a large portion of the object remains covered, providing

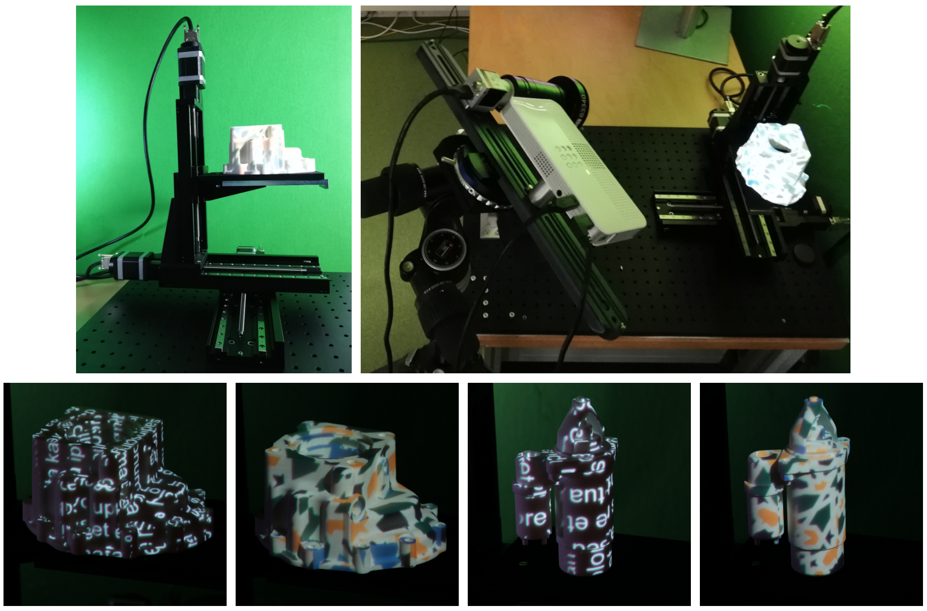


Figure 3.11: First row: Visualization of the test setup. The target object is placed on a motorized linear translation stage (left) and is illuminated by the projector-camera system (right). **Second row: Object 1 and Object 2 with Textures 1 and 2** (from left to right). *(Reprinted from [GHE19], © 2019 IEEE)*

enough information to estimate the correct movement direction. However, the bright border of the projection introduces an error in the contour-based terms, leading to drift.

As expected, the pure contour-based tracker only functions effectively without shader lamp illumination and begins to drift after reaching position 4 on the path (**Cont.** + Amb.) for both projected textures.

Object 2 has a cylindrical shape and is narrower than the first object. As a result, a larger portion of the projection misses the object during a movement of the same size. Using Texture 1, the object can be tracked with the projector-based model with and without ambient illumination. A consistent translational deviation is observed during the movement from position 4 to position 5. However, this offset is mitigated when the combined tracker is used, as the additional contour equations counteract the aperture problem (as detailed in Sec. 3.3.2). The offset is attributed to the object's cylindrical shape, where movement along the main axis does not induce significant changes in the projection's image within the camera view, resulting in a likely error accumulation in that direction.

For Texture 2, the same phenomenon is observed. Tracking remains robust up to the end of the sequence when the combined approach is employed. Without the contour terms, tracking can still be maintained, but only if the movement speed is reduced by

			Object 1				**Object 2**			
			Texture 1		**Texture 2**		**Texture 1**		**Texture 2**	
Method	SL	Amb.	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}
Proj.	✓	-	0.78	0.87	0.99	1.12	1.40	1.15	$3.32^{\dagger}$	$4.21^{\dagger}$
Proj.	✓	✓	0.84	2.00	1.16	2.07	1.63	1.00	$1.86^{\dagger}$	$3.51^{\dagger}$
Comb.	✓	✓	0.85	1.53	Fail (Pos. 7)		1.05	1.33	1.46	1.48
Cont.	✓	✓	Fail (Pos. 4)		Fail (Pos. 4)		4.44	0.99	Fail (Pos. 1)	
Ref.: Cont.	-	✓	(t_{err}: 1.29, r_{err}: 0.21)				(t_{err}: 1.67, r_{err}: 0.38)			

Table 3.3: Real-data evaluation. RMSE of rotation and translation (r_{err}, t_{err}) for the two objects in the presence of shader lamp illumination (SL) with two different textures. Measurements annotated with (†) are obtained with 20% reduced movement speed.

20%. This highlights the potential of utilizing the faster GPU implementation, which was not used in this experiment but is readily available for future work (see Sec. 3.4.3). Figure 3.12 illustrates the corrective effect of the combined approach.

Also for this object, the tracking fails or shows large translation deviations if the projector-based terms are ignored.

All experiments were conducted using two pyramid levels and a maximum resolution of 1224×1024, but additional tests were also performed with one or three pyramid levels. For three levels, the large amount of texture detail in relation to the size of the object in the image (especially if the object is far away from the camera) causes pose instabilities in the coarsest resolution. On the other hand, the highest resolution alone only allows the compensation of small offsets. If the input image is subsampled to a resolution of 612×512, larger offsets can be compensated; however, the pose is not as accurate as with two levels. Thus, the chosen settings represent a good compromise between speed and accuracy.

Discussion In conclusion, the projection-based approach is suitable for precisely tracking objects with different geometric shapes, illuminated with various textures, whereas a purely contour-based tracker fails in the presence of projection. Optimal results are achieved under reduced ambient light, although tracking remains feasible in the presence of ambient light as well. The use of a linear stage for repeatable movements facilitated the identification of scenarios where the combined tracker, utilizing both projection and contour information, offers significant advantages, e.g., in mitigating the aperture problem. Future work should extend the current simplistic separation of projection and object edges with a more advanced method to further reduce the risk of pose drift.

Figure 3.13 demonstrates successful tracking of a manually moved object, with minor

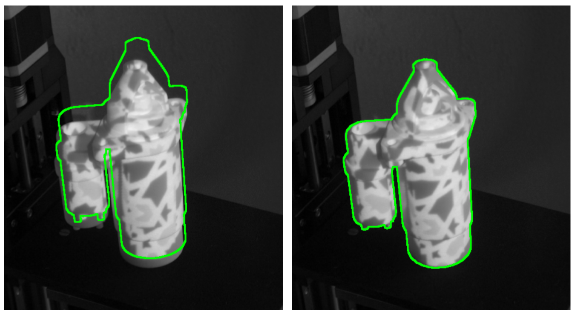


Figure 3.12: Corrective effect of the combined tracking approach. The combined approach (right) resolves ambiguities caused by the aperture problem, preventing the estimated pose from drifting along the axis of a cylindrical object in projection-based tracking (left). *(Reprinted from [GHE19], © 2019 IEEE)*

occlusions from the user's hand and free movement involving rotation and translation. A video showing tracking for both target objects, as well as the Stanford Bunny (as shown in Fig. 3.1), is available as supplementary material to the original publication [GHE19].[2] For slow movements, the CPU implementation, with an approximate frame rate of 10 frames per second, is sufficient. Higher frame rates reduce the risk of drift, as the relative movement between frames becomes smaller, as demonstrated in both the synthetic tests and the linear-stage experiment with reduced movement speed. A strategy to achieve higher frame rates is discussed in the next section.

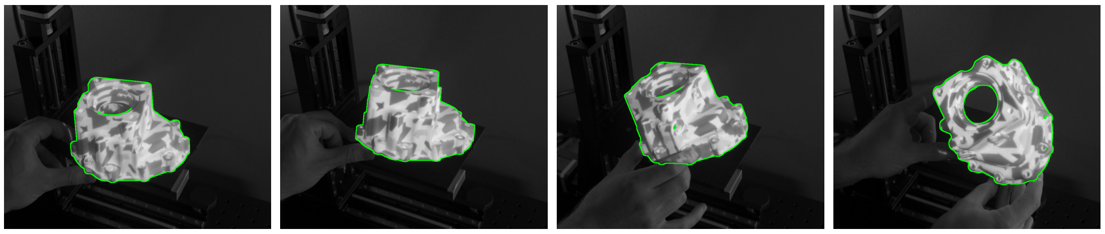


Figure 3.13: Tracking a manually moved object. Object 1 with Texture 2 is tracked using the combined projection-distortion and contour-based approach (images cropped to 70% of their original size for better visibility). The green contour shows the estimated pose.

3.4.3 Synchronization and Timing

The projector-camera system uses off-the-shelf hardware and simple software synchronization, deliberately avoiding costly hardware-based synchronization. The GPU im-

[2]Video available on IEEE Xplore: https://ieeexplore.ieee.org/ielx7/2945/8855102/8794641/video.mp4

plementation leveraging compute shaders, as detailed in Sec. 3.2.4, processes a single 1224×1024 pixel frame in approximately 25 ms on an *Nvidia GeForce GTX TITAN* graphics card. This implementation employs two pyramid levels and three iterations of IRLS. In contrast, the CPU implementation requires about 100 ms for the same task.

The projector operates at 60 Hz, with the OpenGL processing loop synchronized to the vertical blank interrupt. A frame update rate of 30 Hz is achievable if the total processing time, including rendering, stays under 33.33 ms. The GPU implementation meets this requirement. To capture the projection, the camera's exposure time matches the color wheel's rotation, which lasts approximately 16 ms for a 60 Hz projection.

For real-time tracking at 30 frames per second, the combined delays from the projector's input lag, the camera's exposure, and frame readout must remain under 33.33 ms, as outlined by Petković et al. [PPDD16]. However, the *Laser Beam Pro C200* projector's input lag exceeds 16 ms, introducing a processing delay of at least one frame in the current setup. As a result, the system would need to accurately track which projected frame corresponds to the captured frame to ensure correct pose estimation.

Petković et al. [PPDD16] demonstrate a suitable hardware configuration to achieve 30 fps tracking without additional delays. Modern gaming projectors, with input lag as low as 4 ms with 240Hz and 1080p resolution [Tai24], could potentially enable 60 Hz tracking. However, shorter camera exposure times required for such setups necessitate larger apertures, increasing the risk of blurry or underexposed images.

3.5 Chapter Summary and Outlook

This chapter has introduced a framework for model-based 6D object tracking focusing on projector-based methods. The presented real-time algorithm integrates optical flow with object contours to accurately track the 6D pose of moving objects. The main novelty is the addressing of challenges posed by projection distortions in SAR applications. The closed-loop tracking method refines the object pose by analyzing and synthesizing the misaligned projection on the object surface.

This approach enables projection mapping on both planar and non-planar moving objects using projector-camera systems. It overcomes previous limitations associated with reliance on fiducial markers [AIS18] and extended sensors, such as infrared cameras [HK16]. Radiometric calibration is omitted by utilizing binary edge images with adaptive thresholds and a very fast initial illumination estimation. The linear optimization algorithm has been extended to optimize the projector's pose instead of the object's pose. This allows for fast refinement of the extrinsic parameters of a projector-camera system using a reference model with a known 3D shape. Refining the projector's intrinsic parameters with additional poses is an interesting direction for future research.

The experiments have shown, through tests with synthetic and real data, that the algorithm is capable of tracking moving objects with or without ambient illumination and with different projected textures. The findings highlight significant potential for enhancing user interfaces and interactivity in SAR applications (see Sec. 7.1).

Looking ahead, the integration of AI-driven projector compensation methods [HSL21, PJM22, WLH23] promises to advance projection mapping. These methods adjust the projector's input image to counteract geometric and photometric distortions caused by the environment and hardware. In particular, learning-based approaches may be well-suited to accurately localize distorted projections in camera images, overcoming the limitations of the current guided thresholding method. This will facilitate the combination of different motion models, making the tracking process more robust. On the other hand, dynamic projection mapping remains a key challenge in recent projector compensation techniques [HSL21, PJM22, WLH23], which are limited to static scenes. Integrating the model-based frame-to-frame optimization discussed in this chapter and extending it to incorporate additional scene parameters could help to overcome this challenge.

4 Class-Adaptive and Semantic-Aware Deep Pose Estimation

Retrieving the pose of objects in front of a camera in real-time is essential for AR and robotic object manipulation. The local pose refinement methods presented in the previous chapter rely on an initial pose estimate and perform a direct image comparison between rendered and camera images, allowing the domain gap to be closed based on an expected approximate pose.

Finding the initial pose of objects in monocular images is nowadays almost exclusively solved with neural networks, as these significantly boost accuracy and robustness against occlusion or varying illumination [BHW$^+$24]. Training on large pose-annotated datasets encodes the target objects' appearance in the network's weights. As manual annotation and capturing of training images is very laborious, training on synthetic data with automatically available ground truth annotations is becoming more common [BHW$^+$24]. This is especially useful for applications involving many objects and occlusions, as synthetic scenes can include heaps of objects that are hard to annotate manually [GRP22].

Despite significant progress, existing methods are limited when real-time pose estimation for multiple objects is required, as is the case, e.g., in providing AR guidance in complex assembly tasks. Most methods perform best when object-specific CNNs are utilized [PLH$^+$19, PPV19, LWJ19, SSH20, SSF$^+$22, HB22]. As a result, either the entire image must be scanned for each object, or a separate bounding box detector must be used to apply the object-specific network to cropped and scaled instance detections. Both approaches require multiple inferences per image, scaling computation time with the number of objects, which complicates real-time execution. Furthermore, training multiple networks prolongs training, and storing these networks increases memory usage, problematic in resource-constrained scenarios.

Extending single-object networks to handle multiple objects presents challenges. Bounding box-based pose estimation relies on uniformly scaled image patches and a class prior input. Even with a multi-object network, each patch still requires separate encoding and decoding. Alternatively, pose estimation can extend semantic segmentation with additional output maps to derive the pose, such as 3D object coordinates or keypoint locations encoded in vector fields [PLH$^+$19, HBM20]. Adding more objects increases the number of output map channels, which has been shown to degrade performance [SCA$^+$20] and also complicates training (see Sec. 5.2.3).

For example, the well-established PVNet [PLH$^+$19] is lightweight, potentially suitable for AR, and robust against occlusions, but faces the challenges mentioned above when extended to a multi-object model. PVNet predicts 2D vector fields pointing to predefined surface keypoints within the object silhouette and infers 2D-3D correspondences using voting based on RANSAC [FB81]. It then calculates the 6D pose using a PnP solver.

This chapter introduces a method optimized for inferring the poses of multiple different objects from a single image in real-time in a single stage. The approach extends PVNet to better handle multiple objects, but the new ideas can also be applied to other encoder-decoder-based pose estimators.

The proposed method enhances multi-object capability by incorporating a small set of trainable, object-specific weights applied via Conditional Instance Normalization (CIN) [DSK17, SCA+20]. A key innovation is the integration of a class-prior mechanism, inspired by bounding box methods, within a single end-to-end trainable network. This design decouples semantic object identification and keypoint regression into two network heads sharing the same encoder, where pixel-wise segmentation guides the keypoint prediction head. Object-specific parameters are applied locally to the respective mask regions using class-adaptive de-(normalization) (CLADE) [TCC+21].

Unlike previous approaches, the number of output maps in the keypoint decoder remains constant, regardless of the number of objects. Joint vector fields are predicted, and the predicted masks are used to distinguish between objects. To further enhance the network's focus, the masks serve as a shape-based guidance in two semantic-aware operations [DSL+20]. First, guided convolution re-weights the keypoint branch's convolutions to focus on the mask region and its boundaries. Second, segmentation-aware upsampling aligns low-resolution feature maps with high-resolution segmentation masks during feature upscaling, reducing misalignments. Both operations reduce the interference between mutually occluding objects, resulting in clearly separated outputs and improved pose estimates.

Lastly, a new loss function based on weighted least squares is proposed, enabling the supervision of vector field learning using the common ground truth intersection point. The introduction of differentiable keypoint regression (DKR) eliminates the non-differentiable RANSAC voting of PVNet, thereby improving the baseline's performance and robustness.

Directly providing the network with access to the predicted object mask, a domain-invariant feature, helps reduce the domain gap between synthetic training and real-world testing. All experiments use synthetic data for training, while testing involves both real and synthetic images to evaluate how different operations contribute to closing this domain gap.

4.1 Multi-Object Pose Estimation with CASAPose

The proposed CASAPose (**C**lass-**A**daptive and **S**emantic-**A**ware **Pose** Estimator) predicts the 6D poses of n_c distinct objects, each defined by n_p 3D keypoints. It localizes

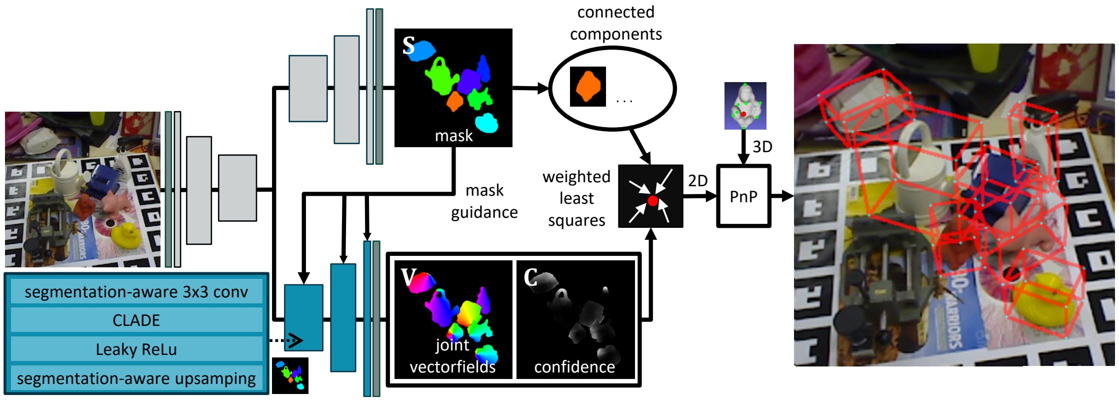


Figure 4.1: Overview of the CASAPose approach. A segmentation decoder (top) estimates object masks that guide the keypoint decoder (bottom) to predict 2D-3D correspondences for single-stage multi-object pose estimation. In the keypoint decoder, CLADE leverages class information, while segmentation-aware operations utilize shape information. Differentiable keypoint estimation is performed using weighted least squares with the estimated confidence and vector maps.

the 2D projections of these keypoints within the input image of size $w \times h$ and estimates the pose from 2D-3D correspondences with a PnP solver [LMNF09].

CASAPose uses an encoder-decoder CNN with two successive decoders to estimate all correspondences in a single pass, as shown in Fig. 4.1. The first decoder (*segmentation decoder*) predicts a segmentation map, $\mathbf{S} \in \mathbb{R}^{w \times h \times (n_c+1)}$, providing pixel-wise object localization. Guided by $\mathbf{S}$, the second decoder (*keypoint decoder*) estimates joint 2D vector fields of unit vectors, $\mathbf{V} \in \mathbb{R}^{w \times h \times n_p \times 2}$, where the vectors point toward the keypoints of the object specified by each pixel's semantic label in $\mathbf{S}$.

The object-specific weights in the CLADE layers of the keypoint decoder help to decipher the encoded features in a characteristic manner (Sec. 4.1.1). The local processing by the object-aware operations minimizes performance degradation and crosstalk between objects (Sec. 4.1.2). The keypoint decoder also outputs a confidence map, $\mathbf{C} \in \mathbb{R}^{w \times h \times n_p}$, from which common intersection points are calculated using weighted least squares (LS). This differentiable operation enables direct optimization of the intersection point in the loss function (Sec. 4.1.3).

The fused vector field output is used directly for loss computation. The keypoint decoder maintains a constant number of channels, while the segmentation decoder adds only one additional channel per object. In total, CASAPose outputs $3n_p + n_c + 1$ channels: $n_c + 1$ for $\mathbf{S}$ and $3n_p$ for $\mathbf{V}$ and $\mathbf{C}$. For instance, CASAPose produces only 41 output channels for 13 objects and 9 keypoints, compared to 248 for PVNet [PLH+19] or 3342 for EPOS [HBM20]. This nearly constant output channel count reduces the GPU memory footprint and accelerates both training and inference.

4.1.1 Object-adaptive Local Weights

The injection of object-specific weights aims to improve a pose estimation network's ability to handle multiple objects. This approach increases the network's capacity using a small set of parameters per object, accessed specifically at the object's spatial location in the image. The process is divided into two subtasks, each managed by a dedicated decoder connected to a shared backbone. The first decoder estimates a semantic segmentation map $\mathbf{S}$, which the second decoder uses to select the corresponding set of object-specific parameters at the correct location. The procedure described below is applied in the second decoder after each convolution, replacing the commonly used unconditional normalization (e.g., Batch Normalization [IS15]).

Let $\mathbf{X} = (x_{x,y,k}) \in \mathbb{R}^{w \times h \times n_k}$ denote a feature tensor in the second decoder, where n_k is the number of channels. A conditional normalization layer [DSK17], applied after a convolution layer, normalizes the features $x^{\text{in}}_{x,y,k}$ of each channel k using its mean μ_k and standard deviation σ_k. The result is modulated with a learned scale γ^l_k and shift β^l_k, which depend on both the channel k and a condition $l = 1, \ldots, n_c$.

$$x^{\text{out}}_{x,y,k} = \gamma^l_k \frac{x^{\text{in}}_{x,y,k} - \mu_k}{\sigma_k} + \beta^l_k \tag{4.1}$$

To achieve semantic-awareness, the approach follows the idea of class-adaptive (de)normalization (CLADE) [TCC+21] and makes the modulation parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ functions of an input segmentation. Thereby, a set $\Gamma = (\boldsymbol{\gamma}^1, \ldots \boldsymbol{\gamma}^{n_c})$ of scale and a set $B = (\boldsymbol{\beta}^1, \ldots \boldsymbol{\beta}^{n_c})$ of shift parameters is learned. Guided Sampling [TCC+21] converts the sets to dense modulation tensors by filling each semantic region in the segmentation map with the corresponding parameters. The segmentation is assumed to be available as an intermediate result.

Guided Sampling uses discrete indices to select specific entries from the sets of learned de-normalization parameters. The required *argmax* operation over the estimated class probabilities sacrifices differentiability, which is necessary for end-to-end training. To address this, the parameter sets are interpreted as matrices $\mathbf{\Gamma} = (\gamma_{l,k}) \in \mathbb{R}^{n_c \times n_k}$ and $\mathbf{B} = (\beta_{l,k}) \in \mathbb{R}^{n_c \times n_k}$. The segmentation input is a tensor $\mathbf{S} = (s_{x,y,l}) \in \mathbb{R}^{w \times h \times n_c}, s \in (0, 1)$. The dense modulation tensors are the scalar product over the last dimension of $\mathbf{S}$ and the parameter matrices $\mathbf{\Gamma}$ and $\mathbf{B}$.

$$\bar{\gamma}_{x,y,k} = \mathbf{S} \cdot \mathbf{\Gamma} = \sum_{l=1}^{n_c} s_{x,y,l} \gamma_{l,k}, \quad \bar{\beta}_{x,y,k} = \mathbf{S} \cdot \mathbf{B} = \sum_{l=1}^{n_c} s_{x,y,l} \beta_{l,k} \tag{4.2}$$

Since this operation should simulate a discrete selection, the predicted label $s_{x,y,l}$ must be either close to 1 or 0. The raw intermediate segmentation $\hat{\mathbf{S}} = (\hat{s}_{x,y,l})$ is normalized with a softmax function scaled with a temperature parameter τ to push all but one value

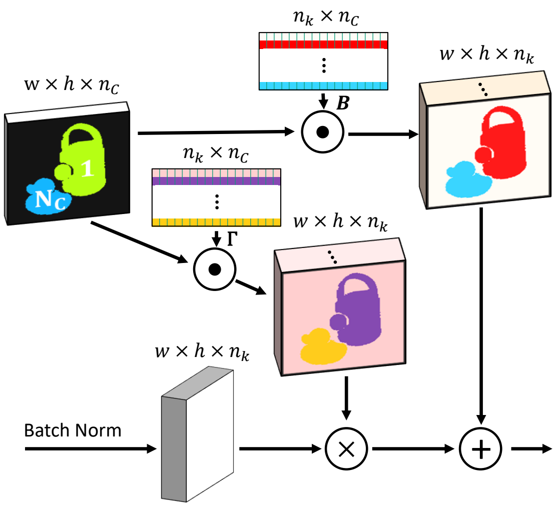


Figure 4.2: Object-adaptive local weights. The segmentation is used to select object-specific modulation parameters from the weight matrices $\mathbf{\Gamma}$ and $\mathbf{B}$ after batch normalization.

close to 0. In this case, τ is set to 1×10^6.

$$s_{x,y,l} = softmax(\tau \hat{s}_{x,y,l}) \tag{4.3}$$

To get the final formulation in Eq. (4.1), the modulation parameters are replaced with $\bar{\beta}_{x,y,k}$ and $\bar{\gamma}_{x,y,k}$, thereby making them spatially dependent on the logits in $\hat{\mathbf{S}}$.

Figure 4.2 shows the class-adaptive weight selection. Using the differentiable CLADE operations allows $\mathbf{\Gamma}$ and $\mathbf{B}$ to be learnable network weights and gradient flow to both inputs.

4.1.2 Semantically Guided Decoder

The keypoint decoder aims to produce a joint vector field $\mathbf{V}$ that is coherent and locally constrained for each object. Based on the object a pixel belongs to, an estimated vector should point to a specific location on that object. However, occluding objects should not influence each other, decreasing the accuracy in overlapping regions (see Fig. 4.6).

A simple convolutional decoder typically decodes backbone features using multiple blocks with convolutions, normalization, activation, and upsampling. If CLADE is used for normalization with pixel-wise, object-specific weights, $\mathbf{S}$ is downsampled using nearest-neighbor interpolation to match the spatial resolution of the feature map. Upon closer examination, two key challenges arise for the simple decoder implementation:

1. Due to the spatial invariance of convolution, the filter is independent of the object boundaries, although this information would be available in $\mathbf{S}$. At boundary points,

the filter lacks information on which (de)normalization parameters were applied to neighboring pixels in the previous block.

2. Nearest neighbor (NN) upsampling after CLADE misaligns upsampled features with the higher-resolution semantic map in the next block, causing potential inaccuracies in predicted outputs (e.g., vector fields) near occlusion boundaries between objects.

To address these issues, segmentation awareness is integrated into convolution and upsampling, as shown in Fig. 4.3. The proposed object-aware operations resemble the panoptic-aware operations by Dundar et al. [DSL+20] but serve a different task. Dundar et al. used these operations for generative color image synthesis based on given panoptic masks, whereas the proposed operations guide the prediction of locally coherent correspondence fields (e.g., in the proposed vector field representation) using segmentation output from a second CNN branch.

Object-aware convolution ensures that the convolution output for a specific point on an object is influenced only by feature values from locations on that same object. A mask $\mathbf{M}$ filters convolution weights $\mathbf{W}$ for every image patch. The mask defines which locations contribute to the feature values. It depends on the semantic segmentation $\mathbf{S}$, normalized with Eq. (4.3). To preserve differentiability, *argmax* is avoided, and a soft mask is used.

$$m_{x,y}(i,j) = \begin{cases} \bar{s}_{x+i,y+j} & \text{if} \quad s_{x+i,y+j,l} = \bar{s}_{x+i,y+j} \\ & \text{with} \quad \{l | s_{x,y,l} = \bar{s}_{x,y}\} \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

In Eq. (4.4), $\bar{s}$ is the maximum value along the class dimension of $\mathbf{S}$. The indices x, y correspond to the filter location in the tensor, while i, j correspond to the position in the filter. The object-aware convolution applies a 3×3 filter at every location through element-wise multiplication of the input features $\mathbf{X}$ with mask $\mathbf{M}$ before filtering. It includes normalization to compensate for the varying number of active elements in $\mathbf{M}$.

$$x' = \mathbf{W}^T(\mathbf{X} \odot \mathbf{M}) \frac{9}{sum(\mathbf{M})} \tag{4.5}$$

Object-aware upsampling enlarges the feature map without losing the alignment with $\mathbf{S}$. Initially, the resolution of $\mathbf{S}$ is halved n times (where $n = 3$) using NN downsampling, enabling it to match features at different resolutions. The segmentation in the next higher resolution $\mathbf{S}^u$, and current resolution $\mathbf{S}^d$, guide the upsampling from low to high-resolution features, $\mathbf{F}^d$ and $\mathbf{F}^u$ [DSL+20]. This preserves the spatial layout in the feature map after the object-aware convolution and CLADE.

To align with the higher resolution mask, the object-aware layer scans a 2×2 region in $\mathbf{S}^d$ and only copies a feature from $\mathbf{F}^d$ to the new location in $\mathbf{F}^u$ if the semantic class

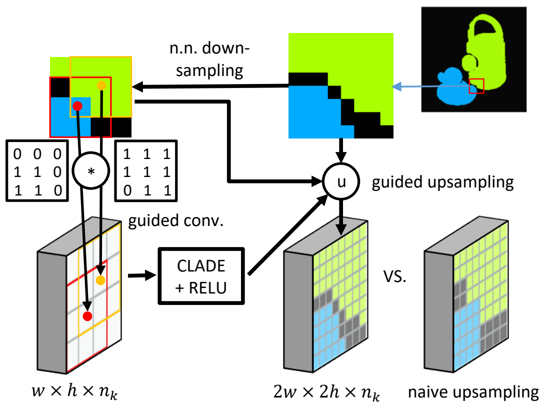


Figure 4.3: Improving alignment with guided operations. Guided convolution and upsampling enhance the alignment between the high-resolution mask and CNN features.

in $\mathbf{S}^d$ and $\mathbf{S}^u$ matches. In contrast, blind NN upsampling simply replicates each feature's value to fill a 2×2 block in the upsampled feature map.

A label may appear in $\mathbf{S}^u$ that cannot be matched with $\mathbf{S}^d$ as details in the masks are lost during downsampling. In that case, the upper-left feature in the 2×2 windows is copied. The hole-filling with an additional convolution, as proposed by Dundar et al. [DSL+20], is omitted, as they show that this method is particularly effective for very low-resolution feature maps. CASAPose retains a feature resolution of $w/8 \times h/8$ in the bottleneck, and the operation is left out to prioritize efficiency over the minimal expected performance gain.

Overall, the modified decoder allows backbone features to flow through multiple resolutions within the semantic masks. This ensures that class-specific denormalizations are applied within the masks and that the receptive fields of the convolutions remain within the mask boundaries.

4.1.3 Differentiable Keypoint Regression

The local processing of the latent map enhances semantic awareness, enabling independent processing of different regions. This helps to eliminate the need for the non-differentiable RANSAC-based voting commonly used to extract 2D intersection points from vector fields [PLH+19, SSH20, YZKL20]. Instead, the common intersection point for an entire image region is directly optimized. While exact intersection points can only be determined for line pairs, weighted least squares provides an effective approximation by considering multiple vectors [Tra13].

In 2D space, a collection of k lines is represented by a set of points $\mathbf{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_k\}$ and a set of direction vectors $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$, with each line given by the parametric equation $p_j(t) = \mathbf{e}_j + t\mathbf{v}_j$, for $j = 1, \ldots, k$.

The error function $E(\mathbf{p}; \mathbf{E}, \mathbf{V}, \mathbf{C})$ computes the weighted sum of squared distances between a specific point $\mathbf{p}$ and k lines:

$$E(\mathbf{p}; \mathbf{E}, \mathbf{V}, \mathbf{C}) = \sum_{j=1}^{k} c_j (\mathbf{e}_j - \mathbf{p})^T (\mathbb{I}_2 - \mathbf{v}_j \mathbf{v}_j^T)(\mathbf{e}_j - \mathbf{p}), \tag{4.6}$$

where $\mathbf{C} = \{c_1, \ldots, c_k\}$ are the weights for each line, and $\mathbb{I}_2$ is the identity matrix. The term $(\mathbb{I}_2 - \mathbf{v}_j \mathbf{v}_j^T)$ projects the distance between $\mathbf{p}$ and the line point $\mathbf{e}_j$ onto the subspace perpendicular to the direction vector $\mathbf{v}_j$, ensuring only the shortest distance is considered. The weights $\mathbf{C}$ help reduce the influence of noise in the line-point distance calculations.

Minimizing E yields a point $\mathbf{p}$ that best represents the common intersection of the lines, weighted by their significance. This is done by differentiating E with respect to $\mathbf{p}$, setting the derivative to zero, and solving the resulting linear system $\mathbf{A}\mathbf{p} = \mathbf{b}$, where:

$$\mathbf{A} = \sum_{j=1}^{k} c_j (\mathbb{I}_2 - \mathbf{v}_j \mathbf{v}_j^T), \quad \mathbf{b} = \sum_{j=1}^{K} c_j (\mathbb{I}_2 - \mathbf{v}_j \mathbf{v}_j^T)\mathbf{e}_j. \tag{4.7}$$

Based on this general mathematical concept, Differentiable Keypoint Regression (DKR) is proposed. Therefore, let $\mathbf{K} = (\mathbf{k}_{l,m}) \in \mathbb{R}^{n_c \times n_p \times 2}$ be the set of n_p projected 3D keypoints for n_c objects, and let $\mathbf{V} = (\mathbf{v}_{x,y,m}) \in \mathbb{R}^{w \times h \times n_p \times 2}$ be the estimated combined vector field. The core idea of DKR is to estimate pixel-wise weights $\mathbf{C} = (c_{x,y,m}) \in \mathbb{R}^{w \times h \times n_p}$ and calculate the common intersection points via weighted least squares over a 2D pixel grid.

For each pixel, $\mathbf{e}_{x,y} = (x, y)$ specifies its 2D coordinates, where $x \in [0, w-1]$ and $y \in [0, h-1]$. A line from (x, y) to a keypoint $\mathbf{k}_{l,m}$ is described by $\mathbf{e}_{x,y}$ and the estimated direction vector $\mathbf{v}_{x,y,m}$. Within each semantic region in $\mathbf{S}$, a weighted equation is added for each pixel to the corresponding linear system for each keypoint on the respective object. The object index l is determined by a pixel's membership in a semantic region from $\mathbf{S}$. The estimated weight $\mathbf{c}_{x,y,m}$ specifies the contribution of the pixel (x, y) for keypoint m to the corresponding equation system. Using the Moore-Penrose pseudoinverse to solve these systems ensures a unique solution for the intersection points, with per-pixel confidence indicating the likelihood of correct vector direction.

An additional loss function minimizes the Euclidean distance between the calculated 2D coordinates and the ground truth. The differentiability of the pseudoinverse solution enables the network to learn which regions produce the most accurate vectors, thus increasing their weight in calculations. This results in a focus on nearby regions and the object contour, as illustrated in Fig. 4.4 and Fig. 4.7.

During inference, semantic maps are clustered into connected components, and the system is solved for the largest component per class, filtering out potential misdetections. This approach works well when only one instance of each object is visible. To handle multiple instances, the semantic segmentation encoder would need to be replaced with an instance segmentation decoder (e.g., [CCZ+20]). In future work, CASAPose can be

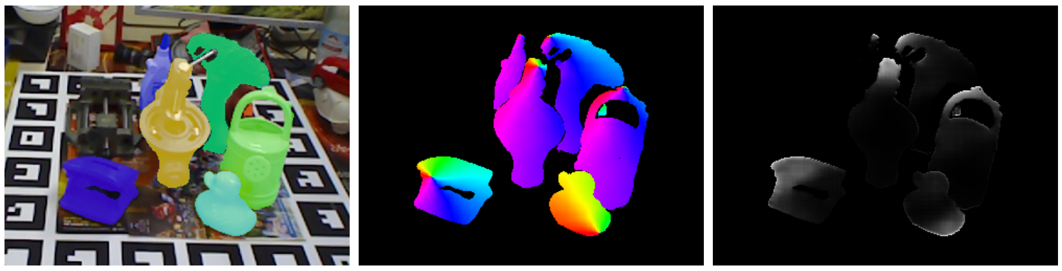


Figure 4.4: Example outputs of CASAPose. Estimated semantic masks, color-coded vector fields, and confidence maps (from left to right), shown for one keypoint per object.

adapted for instance masks. If the object-aware operations (Sec. 4.1.2) separate instances and CLADE (Sec. 4.1.1) distinguishes by class, both operations retain their benefits.

4.2 Implementation Details

Architecture In CASAPose, a pre-trained ResNet-18 [HZRS16] provides features for two decoders. Once the feature map size inside the backbone reaches $w/8 \times h/8$, downsampling is stopped by removing the pooling layers. The subsequent convolutions are replaced with dilated convolutions to preserve the pre-trained receptive field [PLH$^+$19].

The segmentation decoder predicts semantic masks by five blocks of consecutive skip connections, convolutions, batch normalization (BN), leaky ReLU, and upsampling (in blocks 2, 3, and 4). A final 1×1 convolution converts the feature map to the correct number of output maps.

The keypoint decoder is similar but replaces BN with CLADE, the regular convolution with an object-aware convolution, and the blind upsampling with object-aware upsampling. Each block takes a scaled semantic mask as additional input to guide the replaced layers. Improved training convergence is observed when ground truth segmentation is used as guidance instead of the segmentation decoder's output. This allows both decoders to compute their results in parallel, speeding up the training process.

Due to its lightweight backbone, the network is compact, with only approximately 14.8 million weights, and the CLADE layers add just 1024 object-specific weights per object.

The semantic masks, the vector fields, and the confidence maps are passed to a DKR layer (Sec. 4.1.3) that outputs the 2D keypoints for each object. The pose is estimated with OpenCV's EPnP [LMNF09] in a RANSAC scheme. The custom DKR layer computes intersection points in parallel on the GPU, enabling the network to output 2D locations directly. All systems are solved in parallel using `tf.linalg.pinv` to compute the Moore-Penrose pseudoinverse. DKR uses the *softplus* function to convert network outputs into weights for least squares calculations. This smooth approximation of the *ReLU* function constrains the output to be non-negative and allows weights greater than 1. A regularization term during training prevents weights from approaching zero or infinity. Specifically, the mean value in the foreground regions of each output map is L_1 regularized to stay near 0.7, the approximate value of *softplus* at zero.

Training strategy During training, 49 of the 50 scenes from the synthetic LINEMOD dataset of the BOP Challenge 2020 [HSD+20] are used. Each scene consists of 1000 images, which are rendered nearly photorealistically using physically-based rendering (*pbr*) with BlenderProc [DSW+19]. Object and scene parameters, such as the objects' positions, camera and illumination positions, background texture, and material, are randomized. The objects are randomly placed on a flat surface with mutual occlusions. The domain gap is narrowed through strong augmentation techniques, including contrast, color, blur, and noise. The augmentation follows Thalhammer et al. [TLPV21] but modifies the gain for sigmoid contrast [Jun20] to be sampled from a uniform distribution $\mathcal{U}(5, 10)$ to avoid excessive image corruption caused by the originally proposed values.

Each object is described by $n_p = 9$ keypoints, with the first keypoint representing the object center. The 3D keypoints locations are initially calculated using the Farthest Point Sampling algorithm [PLH+19] (see Fig. A.1 on page 139 for the keypoint locations).

Each network is trained for 100 epochs using the Adam optimizer [KB15] and a batch size of 18 on two NVIDIA A100 GPUs (4 is the maximum for a single Nvidia RTX 2080Ti). The initial learning rate of 0.001 is halved after 50, 75, and 90 epochs.

The smooth L_1 loss [Gir15], $\mathcal{L}_{\text{Vec}}$, and differentiable proxy voting loss [YZKL20], $\mathcal{L}_{\text{PV}}$, are used to learn the unit vectors (see Appendix A.1 for the exact calculations). Cross-entropy loss with softmax, $\mathcal{L}_{\text{Seg}}$, is employed to learn the segmentation. Additionally, the keypoint loss, $\mathcal{L}_{\text{Key}}$, is defined as the smooth L_1 of the average Euclidean distance between the estimated keypoints and the ground truth keypoints. The overall loss is:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{Seg}} + \lambda_2 \mathcal{L}_{\text{Vec}} + \lambda_3 \mathcal{L}_{\text{PV}} + \lambda_4 \mathcal{L}_{\text{Key}}, \tag{4.8}$$

with $\lambda_1 = 1.0$, $\lambda_2 = 0.5$, $\lambda_3 = 0.015$ and $\lambda_4 = 0.007$. The λ values were empirically determined and validated using the unseen *pbr* scene.

4.3 Experiments

The experiments utilize test images and objects from the widely used datasets LINEMOD (LM) [HLI+12], Occluded LINEMOD (LM-O) [BKM+14], and HomebrewedDB (HB) [KZSI19]. LM features 13 different objects, while LM-O and the evaluated HB sequences focus on subsets of 8 and 3 objects, respectively. In the following, a 13-object model recognizes all objects, while an 8-object model is trained on the LM-O subset.

Training uses only synthetic *pbr* images [HSD+20], excluding real camera images. For multi-object networks synthetic training is the only practical choice as annotating large multi-object datasets is prohibitively costly. The network estimates the poses of all detected objects in a single pass. See Fig. 4.5 and 4.8 for example images, and Appendix A.3 for dataset details, including a visualization of all target objects.

The standard metrics **ADD** [HLI+12], **ADD-S** [XSNF18], and **2DP** (2D projection) [BMK+16] are reported in most experiments. Recall, defined as the fraction of annotated object instances for which the correct pose is estimated according to the metrics, is provided for each object. If the objects are not specified, the average recall across all objects is reported.

ADD/ADD-S To compute the **ADD** metric, the vertices of the object are transformed with the ground truth and the estimated pose. The average 3D distance between corresponding points must be less than 10% of the object's diameter for the pose to be considered correct. For symmetric objects (such as *glue* and *eggbox*), the **ADD-S** metric is used instead, measuring the average minimum distance between each ground truth vertex and the nearest transformed vertex to avoid ambiguity due to symmetry. The combined **ADD/S** metric applies **ADD** for asymmetric objects and **ADD-S** for symmetric ones, allowing for consistent evaluation across both types.

2DP (2D projection) The vertices of the models are projected into the image using both the estimated and ground truth poses. The average 2D distance between corresponding projected vertices must be less than 5 pixels for the pose to be considered correct.

4.3.1 Ablation Studies

In this experiment, the effects of various components of the proposed architecture are evaluated. Multi-object models are trained on 13 objects from the LM dataset in different configurations. The trained models are then evaluated on LM, LM-O, and the unseen *pbr* scene [HSD+20] to assess improvements both with and without the domain gap. The simplest model (*Base*) uses the merged vector field but otherwise resembles PVNet [PLH+19]. The model is gradually expanded to include the second decoder with CLADE

		LM-O		LM		pbr		Avg.	
Arch.		2DP	ADD/S	2DP	ADD/S	2DP	ADD/S	2DP	ADD/S
Base	+ RV	45.3	22.2	90.0	49.8	74.6	42.2	70.0	38.1
C	+ RV	47.3	26.6	91.8	55.7	76.5	47.5	71.9	43.3
C/GCU	+ RV	49.2	26.7	91.4	59.0	77.3	49.0	72.6	44.9
Base	+ DKR	49.4	28.9	91.7	59.2	77.9	52.5	73.0	46.9
C	+ DKR	51.4	29.9	93.6	64.7	79.3	56.1	74.8	50.2
C/GCU	+ DKR	**51.5**	**32.7**	**93.8**	**68.1**	**79.6**	**57.6**	**75.0**	**52.8**

Table 4.1: Ablation study. Comparison of different network architectures on different datasets. The *Avg.* column shows the average results across the three datasets.

	LM-O		LM		Avg.	
Arch.	2DP	ADD/S	2DP	ADD/S	2DP	ADD/S
C	51.4	29.9	93.6	64.7	72.5	47.3
C/GU	50.7	30.0	93.9	62.9	72.3	46.5
C/GC	51.5	31.3	**94.2**	65.6	**72.9**	48.5
$\text{C/GCU}_{3/5}$	**52.0**	32.1	93.7	65.5	**72.9**	48.8
C/GCU	51.5	**32.7**	93.8	**68.1**	72.7	**50.4**

Table 4.2: Ablation study of the semantically guided decoder. The results are based on the 13-object model with *DKR*. The *Avg.* column shows the average results across both datasets.

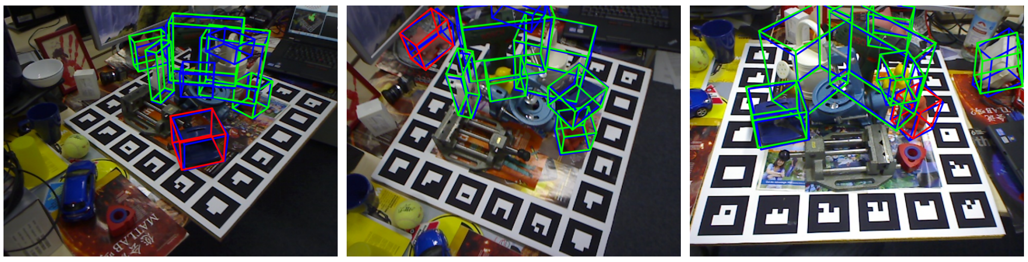
(*C*) and the semantically guided operations (*C/GCU*). For all configurations, a network is trained with and without confidence output, i.e., with Differentiable Keypoint Regression (*DKR*) or RANSAC-based voting (*RV*), so that six configurations are compared in total. Table 4.1 summarizes the results.

Averaged over the three datasets (*Avg.*), adding *C* and *C/GCU* improves **ADD/S** by 13.6% and 17.9% compared to *Base* for the *RV* networks. This demonstrates that extended network capacity with *C* and the enforced local processing with *GCU* improve the quality of the estimated vector fields. Adding *DKR* further improves **ADD/S** by 16.1% for *C* and 17.6% for *C/GCU*, compared to the respective network with *RV*. The total improvement from *Base* with *RV* to the final network is 38.7% for **ADD/S** and 7.14% for **2DP**. Adding *DKR* to *Base* also improves **ADD/S** by 24.5%, and **2DP** by 4.3%, showing that its ability to assign low confidences, e.g., at overlapping regions, improves even the simplest architecture.

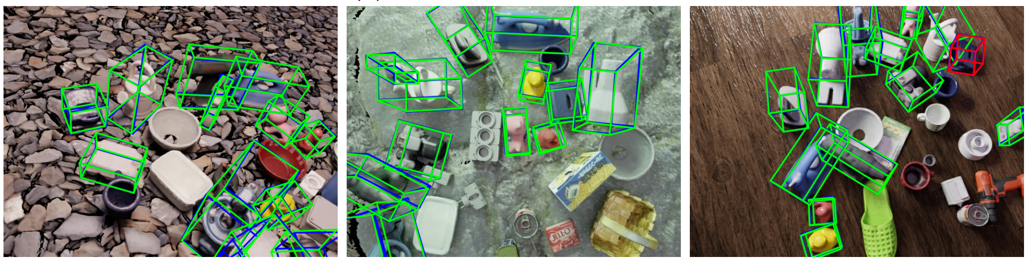
It is observed that adding *GCU* to a network using *DKR* is especially effective for the datasets with domain gap and even more if occlusion is present. **ADD/S** improves upon *C* + *DKR* by 9.3% for LM-O and 7.3% for LM, but only by 2.7% for *pbr*. The proposed reason for this is that the contour acts as a cross-domain feature. Access to the silhouette and, consequently, the higher weighting of nearby vectors helps bridge the domain gap. A detailed visual analysis that supports the quantitative insights is provided in Sec. 4.3.2.

Ablation of Guided Decoder Architectures To further investigate the architecture of the semantically guided decoder, various configurations for the 13-object model trained with *DKR* are tested. The results are summarized in Tab. 4.2.

The first configuration, *C/GU*, uses guided upsampling without guided convolution. This does not improve results over CLADE alone (*C*) since the benefits of upsampling are canceled out by regular convolution, which ignores the masks. In contrast, adding guided convolutions without upsampling (*C/GC*) improves the results across both datasets and metrics compared to *C*. Combining guided upsampling and guided convolutions further



(a) Results for LM-O images.



(b) Results for unseen *pbr* images.

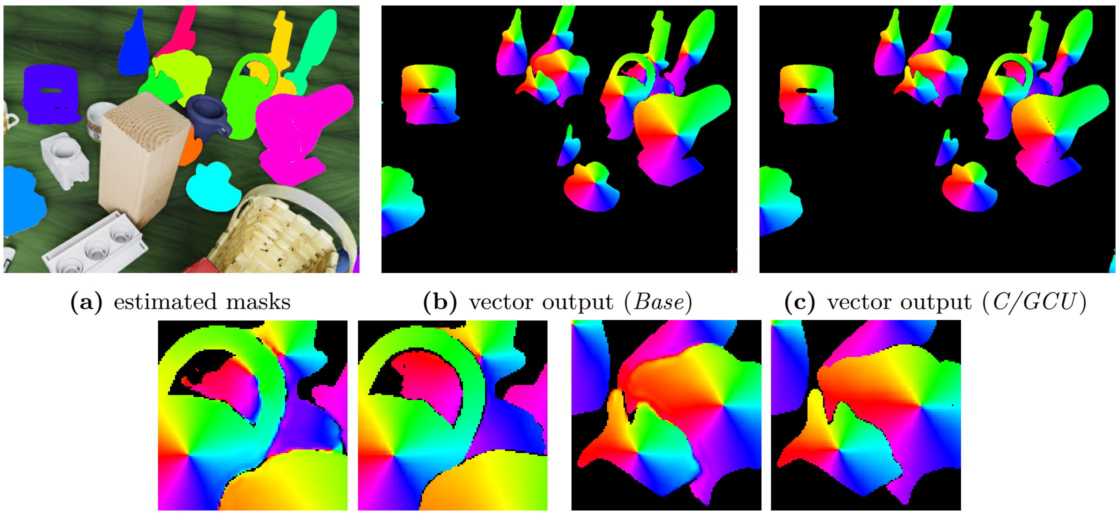
Figure 4.5: Example results of CASAPose for LM-O and unseen *pbr* images. Green bounding boxes indicate correctly estimated poses, red bounding boxes indicate incorrect poses, and ground truth poses are shown in blue (ADD/S metric).

enhances performance. Although the final model C/GCU shows a slight decrease by 0.3% in average **2DP** compared to C/GC, the increase by 3.9% in average **ADD/S** outweighs this. Limiting guided convolutions to the first three of five decoder blocks ($C/GCU_{3/5}$) maintains **2DP** but significantly reduces **ADD/S**, highlighting the importance of applying guided convolutions in the decoder's final layers. Overall, C/GCU is the most effective configuration among those tested and will be used in the following experiments.

4.3.2 Qualitative Analysis

Figure 4.5 shows estimated poses for LM-O and unseen *pbr* images, using an 8-object and 13-object model, respectively. Looking at the poses marked as incorrect with respect to the **ADD/S** metric shows that the criterion is very strict, particularly for small objects distant from the camera. In general, the images illustrate accurate pose estimation; however, the strengths of the developed method become particularly clear when examining the network output in detail:

Effect of guided operations The ablation study revealed that semantic guidance improves the estimated vector fields and the accuracy of pose estimation. The effect is best seen in direct visual comparison. Figure 4.6 shows the enhancement exemplified for an image from the *pbr* dataset using color-coded vector fields for visualization. It



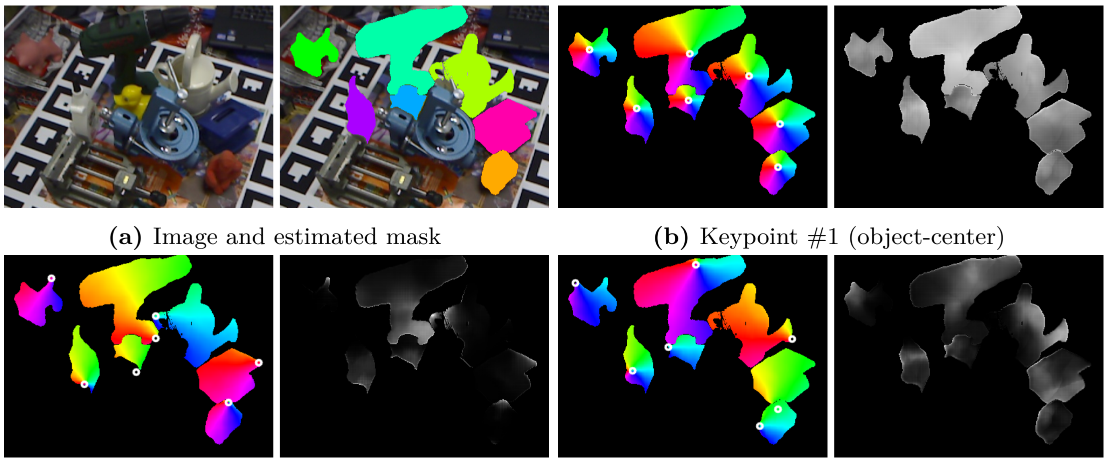
(a) estimated masks **(b)** vector output (*Base*) **(c)** vector output (*C/GCU*)

(d) detail comparison 1 **(e)** detail comparison 2

Figure 4.6: Improved vector fields through semantically guided operations. Visual comparison of the estimated vector fields for a network with (*C/GCU*) and without (*Base*) the semantically guided operations. In detail comparisons, *Base* is on the left and *C/GCU* on the right.

shows the vector fields for the first keypoint, which is always located in the center of each object. Comparing the output of the *Base* model in Fig. 4.6b with the output of a model including object-aware convolutions and object-aware upsampling (*C/GCU*) in Fig. 4.6c, there is a clear improvement in vector fields, especially in regions where objects overlap. Notably, networks without semantic guidance struggled to produce well-separated vector fields, even when heavily overfitting to only a few images. Using CLADE alone without semantic guidance already improves the quality of the vector fields per object due to the object-specific parameters (see Tab. 4.1), but a clear separation as in Fig. 4.6c can only be achieved in combination.

Characteristics of the learned confidence maps Figure 4.7 shows the estimated vector fields, confidence maps, and keypoint locations for an image from LM-O using the 8-object model. For the first keypoint (Fig. 4.7b), which is always in the center of the object, the confidence is relatively constant in each mask, indicating that it is easy for the network to predict this point with high accuracy. In Fig. 4.7c and Fig. 4.7d, it can be seen that the regions where the network predicts high confidence are often spatially close to the actual keypoint location. For example, for the tip of the tail of the *cat* object (green mask) in Fig. 4.7c, it is logical that the most accurate prediction of the location is made near its actual location. Similarly, the *ape* object (orange mask) in Fig. 4.7d demonstrates that the model predicts high confidence and thus computes the 2D position of a keypoint



(a) Image and estimated mask **(b)** Keypoint #1 (object-center)

(c) Keypoint #4 **(d)** Keypoint #5

Figure 4.7: Keypoint-specific focus in the estimated confidence maps. The central keypoint shows uniform confidence, while others emphasize contours or nearby regions. Estimated vector fields and confidence maps for three of nine keypoints, with estimated 2D locations marked by white circles. Confidence values are normalized within each semantic mask for clearer presentation.

primarily from pixels near the object silhouette. Especially for non-textured objects, the silhouette provides information information about orientation, making it reasonable for vectors near the contour to be estimated with higher precision.

4.3.3 Influence of Keypoint Regression

This experiment evaluates the effectiveness of *DKR* by comparing different variants of 2D keypoint position estimation. The comparison is conducted using the 13-object model on LM-O. The results are shown in Tab. 4.3.

$LS_{1st\ Comp.}$ is the variant used in the final model. It applies *DKR* to the largest connected component of each semantic mask and clearly outperforms the network trained without the introduced loss, applying RANSAC voting (PV_{RANSAC}) [PLH+19]. Interestingly, when a network learns to estimate confidence maps with *DKR*, also the results of RANSAC voting (PV_{RANSAC^*}) improve. This suggests that the region-wise least squares optimization during training also improves the global accuracy of the estimated vectors. However, applying *DKR* during both training and testing yields the best performance.

Applying *DKR* on complete masks without connected component filtering (LS_{All}) degrades performance, indicating that potential clutter in the estimated semantic masks should be removed before calculating the 2D positions. Testing *DKR* on the second-largest connected component ($LS_{2nd\ Comp.}$) reveals that it results in a correct pose in

	2DP	ADD/S
PV_{RANSAC}	49.2	26.7
PV_{RANSAC^*}	50.4	30.8
LS_{All}	45.3	29.7
$LS_{2nd\ Comp.}$	7e-3	2e-3
$LS_{1st\ Comp.}$	**51.5**	**32.7**

Table 4.3: Comparison of different variants of 2D keypoint calculation using the 13-object model on LM-O.

	LM-O		LM	
	2DP	ADD/S	2DP	ADD/S
13 Obj.	51.5	32.7	96.0	60.4
8 Obj.	54.7	35.9	96.9	59.2
4 Obj. (2x)	**58.4**	**38.7**	**97.3**	**64.5**

Table 4.4: Influence of the number of objects per network on ADD/S and 2DP recall.

only 0.2% of cases. Therefore, when only one object instance per class is visible, relying solely on the largest connected component proves to be highly effective.

4.3.4 Influence of the Object Number

The multi-object capacity of CASAPose with respect to the **2DP** and **ADD/S** metric is evaluated in Tab. 4.4. Poses are estimated for the eight LM-O objects on LM and LM-O. The 8-object network on LM performs comparably to the 13-object network, indicating that the multi-object gap is minimized for objects without occlusion. However, for LM-O, the average **ADD/S** for the 8-object network is 8% higher than that of the 13-object network, with nearly identical **2DP** values. This indicates that for complex object arrangements, a more specified network still performs better. It is likely that a more powerful segmentation backbone can further narrow down the observed performance gap.

Splitting the objects into two groups of four (*ape-drill*, *duck-holepuncher*) leads to a further increase in both metrics. The two-network solution increases **ADD/S** on LM-O to 38.7%, further extending the distance to the other methods in Tab. 4.5. In particular, the objects in the first group exhibited significant improvement, which can be attributed to the exclusion of symmetrical objects from the group.

Symmetries introduce ambiguities in keypoint estimation that are not explicitly addressed in CASAPose. Future work could explore the addition of a differentiable renderer and a projection-based loss, such as evaluating edge alignment, to better account for symmetry. Beyond the scope of this work, strategies to address symmetries have been outlined, for example, by Song et al. [SSH20].

4.3.5 Comparison with the State of the Art

The presented method is compared against state-of-the-art methods at the time of the research [GHE22a] and beyond (Tab. 4.8). As the field is evolving rapidly, the outlook sections in this and the next chapter discuss ongoing advancements. The methods used for comparison in Tab. 4.5, Tab. 4.6, and Tab. 4.7, all introduced in Sec. 2.2.1, share a

Method	Training	single-st.	Ape	Can	Cat	Drill	Duck	Eggb.	Glue	Hol.	**Avg.**
DPOD [ZSI19]	syn.	✓	2.3	4.0	1.2	10.5	7.2	4.4	12.9	7.5	6.3
CDPN [LWJ19]	syn.	-	20.0	15.1	16.4	5.0	22.2	36.1	27.9	24.0	20.8
DSC-PoseNet [YYY21]	pbr+RGB	-	9.1	21.1	**26.0**	33.5	12.2	39.4	37.0	20.4	24.8
PyraPose [TLPV21]	pbr	✓	18.5	46.4	11.7	48.2	19.4	16.7	30.7	33.0	28.1
Self6D [WMS+20]	pbr+RGB-D	-	13.7	43.2	18.7	32.5	14.4	**57.8**	52.3	22.0	32.1
DAKDN [ZZG+21]	pbr+RGB	-	-	-	-	-	-	-	-	-	33.7
SD-Pose [LHSJ21]	pbr	-	21.5	56.7	17.0	44.4	**27.6**	42.8	45.2	21.6	34.6
CASAPose	pbr	✓	**24.3**	**59.5**	15.2	**57.5**	26.0	14.7	**55.4**	**34.3**	**35.9**

Table 4.5: Comparison with state-of-the-art approaches on LM-O. The table lists to ADD/S recall. Comparison is conducted against methods that use only synthetic training or weakly supervised training using unlabeled real data.

common focus on addressing the challenge of bridging the domain gap. More methods were trained and tested on synthetic data in the BOP Challenge [HSD+20]. These will be compared to CASAPose in a separate paragraph, discussing also additional variations of the network, including variants of the network with different backbones.

Occluded LINEMOD (LM-O) [BKM+14]

Table 4.5 compares the **ADD/S** metric on LM-O against other methods that only train on synthetic data [LWJ19, ZSI19, LHSJ21, TLPV21], or use additional unlabelled images from the test domain [WMS+20, LHSJ21, YYY21].

CASAPose outperforms the single-stage multi-object approach PyraPose [TLPV21] by 27.8%, while also being significantly smaller (14.8 million vs. 43 million parameters). Contrary to the findings of PyraPose [TLPV21], which suggest that patch-based methods have an advantage over encoder-decoder networks in the presence of a domain gap, the results show that a small encoder-decoder network can achieve superior performance.

CASAPose also outperforms SD-Pose [LHSJ21], a multi-stage method that uses object-specific networks and a separate bounding box detector, by 3.8%. This improvement underscores the effectiveness of the single-network approach, resulting in computational savings. Additionally, CASAPose surpasses DAKDN [ZZG+21], a weakly-supervised approach, by 6.5%. This demonstrates that the network architecture, designed to address domain invariance, can achieve competitive performance without relying on additional training data from the test domain.

Similar to CASAPose, the Top-2-6 approaches [WMS+20, LHSJ21, TLPV21, YYY21, ZZG+21] all use physically-based rendered (*pbr*) images in training. These methods consistently outperform approaches using simpler synthetic data, such as CDPN [LWJ19] and DPOD [ZSI19], as *pbr* images better capture real-world variations in object appearance, lighting, and pose, leading to more robust model generalization.

Method	Ape	Bv.	Cam	Can	Cat	Drill	Duck	Eggb.	Glue	Hol.	Iron	Lamp.	Ph.	**Avg.**
AAE [SMD+18]	4.2	22.9	32.9	37.0	18.7	24.8	5.9	81.0	46.2	18.2	35.1	61.2	36.3	32.6
MHP [MAR+19]	11.9	66.2	22.4	59.8	26.9	44.6	8.3	55.7	54.6	15.5	60.8	-	34.4	38.8
Self6D-LB [WMS+20]	37.2	66.9	17.9	50.4	33.7	47.4	18.3	64.8	59.9	5.2	68.0	35.3	36.5	40.1
DPOD [ZSI19]	35.1	59.4	15.5	48.8	28.1	59.3	25.6	51.2	34.6	17.7	84.7	45.0	20.9	40.5
PyraPose [TLPV21]	22.8	78.6	56.2	81.9	56.2	70.2	40.4	84.4	82.4	**42.6**	**86.4**	62.0	59.5	63.4
SD-Pose [LHSJ21]	**54.0**	76.4	50.2	81.2	**71.0**	64.2	**54.0**	**93.9**	**92.6**	24.0	77.0	82.6	53.7	67.3
CASAPose	30.3	**94.8**	**60.0**	**83.9**	60.5	**89.2**	37.6	71.0	80.7	30.7	84.5	**89.9**	**71.7**	**68.1**

Table 4.6: Comparison with state-of-the-art approaches on LM. The table lists the ADD/S recall. The comparison is conducted against methods that use synthetic training.

The *eggbox* object, however, poses a challenge due to its symmetry, which leads to ambiguities in 2D keypoint projections during training. Additionally, heavy occlusion of the object in many images causes the segmentation branch to fail to detect it. Future work could involve augmenting the training with random occlusions or adding a semi-supervised component to the network's detection part to improve performance.

LINEMOD (LM) [HLI+12]

Table 4.6 compares the performance of the network trained on the LM dataset with other methods that utilize only synthetic training data. A single network is used to detect all 13 objects in the dataset, which, aside from our method, is also the case for PyraPose [TLPV21]. The achieved mean **ADD/S** of 68.1% is the highest among the listed methods, exceeding SD-Pose [LHSJ21] by 1.2% and PyraPose by 7.4%.

HomebrewedDB (HB) [KZSI19]

The second sequence of HB is a benchmark to check whether a method generalizes well to a novel domain [KZSI19]. It contains three objects from LM, captured with a different camera in a new environment. The network, trained for LM, is used without retraining, and the result far surpasses the next-best method DAKDN [ZZG+21] by 35%, as shown in Tab. 4.7. Interestingly, the margin of improvement has increased, suggesting that other methods are more prone to overfitting on the characteristics of the LM dataset. Additionally, including a small set of unlabelled RGB-D images from the dataset [WMS+20] or using synthetic training with meshes reconstructed from the dataset [TLPV21] can not narrow the gap as much as the proposed solution. By focusing on the object mask and because 2D-3D correspondences are predicted instead of a full 6D pose, the network shows high invariance to new environments and different capture devices. To verify this, another version of the same sequence captured with a Kinect, also part of HB, is evaluated (see Fig. 4.8). Despite greater differences in camera parameters

	DPOD [ZSI19]	PyraP.† [TLPV21]	DSC-P. [YYY21]	Self6D† [WMS+20]	DAKDN [ZZG+21]	**Ours**	Ours‡
Benchvise	52.9	62.9	72.9	72.1	-	**85.8**	83.2
Drill	37.8	22.6	40.6	65.1	-	**94.1**	89.1
Phone	7.3	38.5	18.5	41.8	-	**78.8**	80.8
Avg.	32.7	41.3	44.0	59.7	63.8	**86.3**	84.4

Table 4.7: Comparison with state-of-the-art approaches on HB sequence with LM objects. The table lists the ADD/S recall. Methods indicated with (†) retrain with data from the target domain. Results are listed for the Primesense and Kinect (‡) sequence. No per-object results were provided for DAKDN.

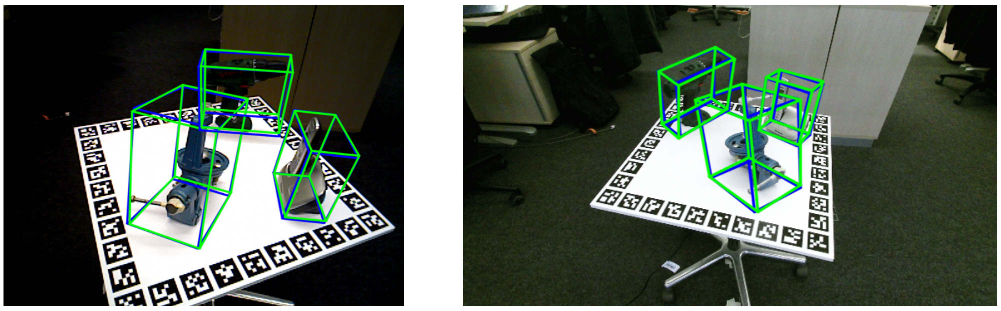

(a) PrimeSense Carmine (b) Microsoft Kinect 2

Figure 4.8: Example results of CASAPose for HB. Green bounding boxes indicate correctly estimated poses, and blue bounding boxes indicate ground truth poses (ADD/S metric). When highly accurate, the green contour overlaps the blue one.

(e.g., opening angle and aspect ratio), the result remains nearly as robust. Figure 4.8 shows results from both sequences, demonstrating precise pose estimates for both sensors.

BOP Challenge Evaluation

In the annual BOP Challenges [HSD+20, SHL+23, HSL+24], multiple approaches submit results for several pose estimation datasets, including LM-O, which uses synthetic *pbr* training. In many cases, the submitted results are improved through subsequent changes compared to those in the original publications. In the following, CASAPose is evaluated and compared using the BOP benchmark (see Appendix A.2 for details). It computes an accuracy score, denoted Average Recall (**AR**), which is the average of the results for three pose error functions: Maximum Symmetry-Aware Projection Distance (**MSPD**) measures the 2D projection error between the estimated and ground truth poses in the image space, considering object symmetries; Maximum Symmetry-Aware Surface Distance (**MSSD**) evaluates the alignment of 3D object surfaces in the estimated and ground truth

Method	CNN Config	CNNs	ref.	*AR*	AR_{MSPD}	AR_{MSSD}	AR_{VSD}
COPE [TPV23]	1/set	1	-	54.3	-	-	-
EPOS [HBM20, HSD+20]	1/set	1	-	54.7	75.0	50.1	38.9
CASAPose$_{\mathrm{R18*}}$	1/set	1	-	55.4	75.3	50.8	40.2
CASAPose$_{\mathrm{R18}}$	2/set	2	-	57.4	77.1	52.9	42.1
PVNet [PLH+19, Bop20b]	det. + 1/obj.	9	-	57.5	75.4	54.2	42.8
CASAPose$_{\mathrm{R34}}$	1/set	1	-	57.8	76.2	54.2	42.8
CASAPose$_{\mathrm{R50}}$	1/set	1	-	57.9	77.8	53.5	42.4
CASAPose$_{\mathrm{R50}}$	2/set	2	-	60.0	80.3	55.4	44.2
CDPNv2 [LWJ19, HSD+20]	det. + 1/obj.	9	-	62.4	81.5	61.2	44.5
CASAPose$_{\mathrm{R50+gt}}$	1/set	1	-	63.2	80.1	58.0	50.6
CosyPose [LCAS20, HSD+20]	det.+ 1/set + ref.	3	✓	63.3	81.2	60.6	48.0
CASAPose$_{\mathrm{R50+gt}}$	2/set	2	-	65.3	83.0	60.0	52.7
SurfEmb [HB22, SHL+23]	det. + 1/obj.*	9*	✓	66.3	85.1	64.0	49.7

Table 4.8: BOP benchmark evaluation. Comparison of different CASAPose variants on LM-O with different state-of-the-art approaches. Besides performance metrics, the number of *CNNs* required is listed. *CNN Config* denotes the CNN distribution, specifying whether a detector (*det.*), or object-specific (*1/obj.*, * indicates a shared encoder) vs. shared networks (*1/set*) are used, *ref.* indicates pose refinement.

poses, accounting for symmetry; and Visible Surface Discrepancy (**VSD**) quantifies the differences in visible regions of the object under occlusion or partial visibility.

Table 4.8 presents results for the BOP subset of LM-O (200 frames from the original 1,214-frame dataset). Various configurations of the proposed model are compared against state-of-the-art approaches that train exclusively on *pbr* RGB data. CASAPose$_{\mathrm{R18*}}$[1] slightly outperforms EPOS [HBM20, HSD+20], which remains the only submitted single-network multi-object method up to the 2023 challenge [HSL+24]. Notably, CASAPose is 4.4 times faster (107 ms vs. 468 ms) using the same GPU as in the EPOS experiment (Nvidia P100) and achieves real-time capability on a recent GPU (see Sec. 4.3.6). Compared to the newer single-stage multi-object method COPE [TPV23], which is also real-time capable, CASAPose$_{\mathrm{R18*}}$ achieves a 2.0% improvement in **AR**.

Increasing the backbone capacity to ResNet-34 in CASAPose$_{\mathrm{R34}}$ raises the **AR** from 55.4 to 57.8, increasing the improvement over COPE and EPOS. It also surpasses an extended version of PVNet [Bop20b], which uses a separate 2D detector and trains a dedicated network for each object. Further increasing the backbone capacity to ResNet-50 in CASAPose$_{\mathrm{R50}}$ results in a marginal **AR** improvement of 0.1, indicating diminishing performance gains with deeper networks.

As shown in Sec. 4.3.4, splitting the set of objects into two groups boosts perfor-

[1] R18 indicates the ResNet-18 backbone, and $*$ denotes minimally adjusted hyperparameters, specifically increasing λ_4 from 0.007 to 0.01.

mance. This improvement is evident in the *2/set* configurations of CASAPose$_{\mathrm{R18}}$ and CASAPose$_{\mathrm{R50}}$, which increase the **AR** by 2.0 and 2.1 points, respectively. The latter also further increases the distance to PVNet, supporting the claim that the domain gap is effectively reduced through the introduced *DKR*.

Regarding other non-real-time multi-stage methods, there are approaches that outperform CASAPose. For example, CDPNv2 [LWJ19, HSD+20] achieves an **AR** of 62.4, but it trains a single ResNet-34-based network for every object. The BOP version extends CDPN with additional enhancements, such as more complex domain randomization, which could, beyond the scope of this experiment, also benefit CASAPose. The presented innovations, aimed at converting a multi-stage approach (involving one network per object and a bounding box detector) into a single-stage approach (utilizing one network for all objects without the need for a bounding box detector), could also enhance CDPNv2.

Finally, the keypoint head's accuracy can be evaluated in isolation using ground truth masks during inference (CASAPose$_{\mathrm{R50+gt}}$). Even with a separate instance detector, all masks could be processed in a single pass through the multi-object network. Using ground truth masks increases the **AR** by 5.3 points for both the *1/set* and *2/set* configurations. Improving the detection head or outsourcing this task to a separately trained instance detector could narrow the gap compared to methods that incorporate advanced symmetry handling and pose refinement with a neural network (e.g., CosyPose [LCAS20]) or iterative optimization (e.g., SurfEmb [HB22]). However, both of these methods are unsuitable for real-time applications.

4.3.6 Runtime Evaluation

The average runtime from image input to final poses for all visible objects in LM-O is 30.0 ms with the 8-object ResNet-18-based model. It splits in 16.4 ms for network inference, 2.9 ms for *DKR*, 5.7 ms for PnP, and 5.0 ms for finding the largest connected component for each class. This results in approximately 33 frames per second on the test system (Nvidia A100 GPU, AMD EPYC 73F3 CPU). Using ResNet-34 as the backbone increases processing time by 2.9 ms, while switching to ResNet-50 adds 8.1 ms. The ResNet-34 configuration with a runtime of 32.9 ms is significantly faster than the *2/set* ResNet-18 configuration, which requires 46.6 ms, while still offering more precise results.

The CNN runtime, which constitutes the largest portion of the total runtime, remains independent of the number of objects in the image, an advantage of the single-stage method. In contrast, approaches that train separate networks for each object experience a linear increase in CNN runtime as the number of objects increases.

The connected component analysis and PnP solving are inefficiently parallelizable on the GPU. Future work can remove the connected component analysis by incorporating instance awareness. Instance center regression [CCZ+20] provides a method for mask separation while retaining the benefits of combined training and the single-stage approach.

Alternatively, masks from a state-of-the-art instance segmentation network could be used to guide the keypoint decoder, requiring minimal changes to the proposed architecture.

4.4 Chapter Summary and Outlook

This chapter demonstrates that class-adaptiveness and semantic awareness improve the performance and multi-object capacity of a 6D pose estimator CNN by incorporating a small set of object-specific parameters via CLADE in a secondary decoder. Local feature processing minimizes interference between overlapping regions through segmentation-aware transformations and allows for a reduction in the number of outputs without sacrificing accuracy.

The introduced Differentiable Keypoint Regression—a new method for accurate 2D keypoint estimation—improves the 2D-3D correspondences and reduces the domain gap. The focus on shape allows the method to outperform several semi-supervised approaches that use both real and synthetic data, while it was trained solely on synthetic data. The pixel-wise regression of least-squares weights and the direct attachment of a least-squares solver to a CNN is an innovation that could be useful for training networks for other tasks, such as end-to-end camera calibration. The presented layers are flexible enough to integrate into other pose estimation or different-purpose architectures.

A compelling direction for future research is to enhance the method by incorporating instance awareness while maintaining the benefits of a single-stage approach. For instance, the work of Chen et al. [CWM22] could be a starting point as they explore the layering of touching and overlapping objects into distinct image layers in their paper. This would allow guided operations in the second branch to be managed at the instance level, representing a simple yet effective extension for complex scenes.

An emerging trend, observed after the publication of CASAPose, is the increasing use of foundation models, such as the Segment Anything Model (SAM) [KMR$^+$23] and DinoV2 [ODM$^+$23], to address limitations in recognizing previously unseen objects. For example, SAM produces object-agnostic masks, and the method CNOS [NGP$^+$23] builds on this by leveraging DinoV2 features to match pre-generated template views and identify corresponding masks in the image. In this context, a modified mask-conditioned decoder, similar to the one used in CASAPose, could also help in identifying object-specific keypoints.

5 Linking Model-based Refinement to CNN Architectures

Detecting and stably registering 3D rigid objects in videos is crucial for AR systems. Individually, these processes offer limited utility; combined, they form the backbone of robust dynamic systems [DJW$^+$20, FZH$^+$22]. This chapter integrates the proposed model-based refinement and CNN architectures to enhance AR performance, particularly in industrial and construction scenarios, where precision and real-time feedback are essential. In such applications—ranging from object assembly [EGIB23] to object sorting [PRFG23]—reliable detection and tracking become particularly challenging when dealing with textureless objects, small production runs, or subtle geometric differences [BHW$^+$24]. Computerized assistance is particularly useful in these cases, as it is highly demanding for a person to distinguish similar objects by eye and to remember their shapes [HBJ$^+$21].

While monocular object pose tracking provides temporally stable poses, one of its primary issues is drift. Tracking errors may accumulate over time since the pose is typically estimated between two frames. Once accumulated, these errors cannot be eliminated because the initial pose differs too much from the actual pose [FZH$^+$22]. Automatic drift detection followed by pose reinitialization are essential.

On the other hand, CNN-based pose estimation provides robust pose initialization but typically processes images at a limited resolution and, in most cases, only one frame at a time, which restricts the accuracy of the pose estimates. The detections are expected to be jittery in a video sequence if solely the neural network is used [FZH$^+$22].

Although pose tracking and pose estimation can benefit from each other, these topics are usually addressed separately in the literature. Few frameworks cover both [ZAA19, YCL$^+$23], but they focus on single-object applications and do not explicitly address scenarios involving multiple geometrically similar objects. To address this gap, the proposed pipeline simultaneously tracks and refines the poses of multiple objects, evaluating how the components enhance one another in a dynamic system.

Contrary to the trend of making training data more realistic [HVG$^+$19], this chapter shows that pose refinement narrows the domain gap, allowing the use of a more straightforward and less realistic domain-randomized training data [TTM$^+$18]. The local, edge-based refinement process aligns the 3D model with image edges in high-resolution images. To further simplify data generation and facilitate the compilation of datasets with specific object sets, the training of the multi-object network is conducted using only

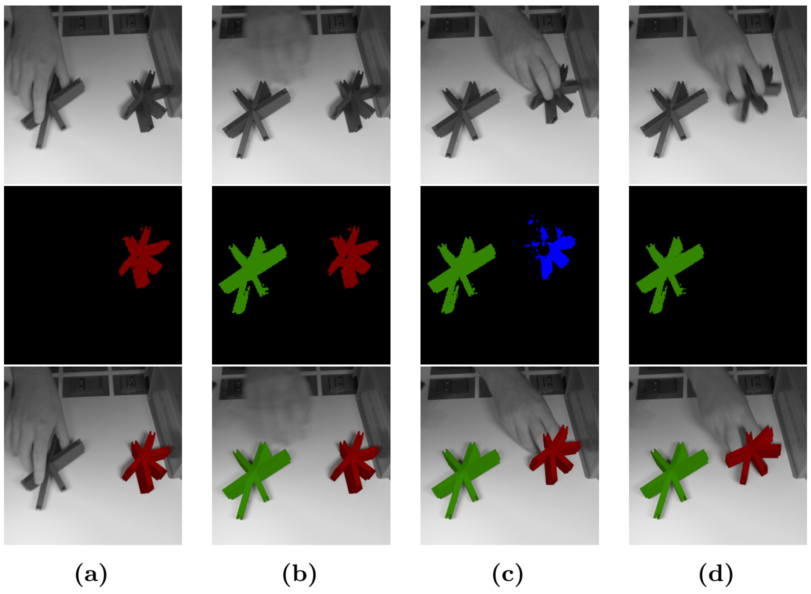


Figure 5.1: Combined pose detection and tracking. Tracking starts immediately once the left object is uncovered (a, b). Local tracking remains active for the right object, even if the CNN detects a wrong (c) or no object (d). Rows from top to bottom: camera image, estimated semantic segmentation, camera image with rendered overlay.

single-object images. Again, the additional refinement is intended to close the enlarged domain gap.

For the detection, false-positive detections may arise from the CNN due to shape ambiguities, especially for similar objects. Object edges provide a cue to suppress them. Poses are accepted only if it is possible to accurately match the projected contour of the 3D model with the image edges, using an edge deviation error from local pose refinement.

Although CNN-based solutions are real-time capable, prioritizing local refinement in video sequences often proves beneficial (see Fig. 5.1c and Fig. 5.1d), particularly in situations that are not explicitly represented in the training data. In cases where local refinement is not successful due to occlusion or rapid motion, the edge-based pose validation actively initiates reinitialization (as illustrated in Fig. 5.1a and Fig. 5.1b), eliminating the need for CNN evaluation.

The previous chapter demonstrated the extended capacity to detect multiple object classes with a single network. This chapter presents a case study evaluating how this capability applies to a set of 13 geometrically similar, textureless objects, where an even more significant benefit is expected. Additionally, the experiments qualitatively and quantitatively show how the approach addresses the aforementioned challenges under an extended domain gap and in video, resulting in a robust and reliable system.

5.1 Direct Linking of Local Refinement with Deep Registration

This section introduces a real-time pipeline for multi-object detection and tracking, designed to operate without relying on pre-existing photographs of real objects for recognition. It builds upon the local refinement techniques discussed in Chapter 3 and integrates them with the deep pose estimation models presented in Chapter 4.

The process begins with an offline phase where synthetic renderings are generated as training data for a pose estimation CNN (Sec. 5.1.1). During real-time operation, the video stream is continuously scanned, and the CNN provides initial pose estimates for each visible, known object. Once an object is detected, a local refinement step aligns the edges of its 3D model with the corresponding image edges (Sec. 5.1.2). A subsequent validation step (Sec. 5.1.3) evaluates the quality of these alignments, confirming whether the pose is valid or invalid. For objects with validated poses, local tracking is sufficient in the next image, eliminating the need to evaluate the CNN for that frame.

5.1.1 Data Generation

The input of the data generation step is a set of n_c 3D models to be detected and tracked. They may have similar shapes and may be untextured or from the same material. The models are centered at their 3D bounding box and n_p keypoints, the object center and $n_p - 1$ points on the object surface, are determined using the Farthest Point Sampling algorithm [PLH+19].

A dataset consisting of synthetic renderings is created using NDDS [TTM+18], an Unreal Engine plugin for generating annotated training images. Domain randomization is used to mitigate performance degradation caused by the domain gap when training exclusively on synthetic data and testing on real data [TFR+17]. By introducing extensive random variations in the synthetic scene, the model ideally learns to interpret real-world data as another variation of the training set.

In each image, the target object is rendered against a random background: a solid color, a procedurally generated multicolored grid, or a randomly selected photo from the *Flickr-8K* dataset [HYH13], with each occurring with equal probability. The position and number of light sources, the object's position, orientation, and texture, as well as the camera's position and orientation are all randomized. Half of the images feature the object placed above a flat surface with a random texture to cast shadows. The random textures are sourced from a publicly licensed texture database [Dem]. To introduce partial occlusion, distractor objects (such as spheres, cylinders, cones, and pyramids) are randomly placed and scaled. Each training image contains only one visible target object to simplify dataset reconfiguration for different object sets.

The data generation prioritizes simple synthetic data rather than near-photorealistic images. Later, local pose refinement further mitigates inaccuracies introduced by the domain gap, while incorrect estimates are filtered through a geometric validation step.

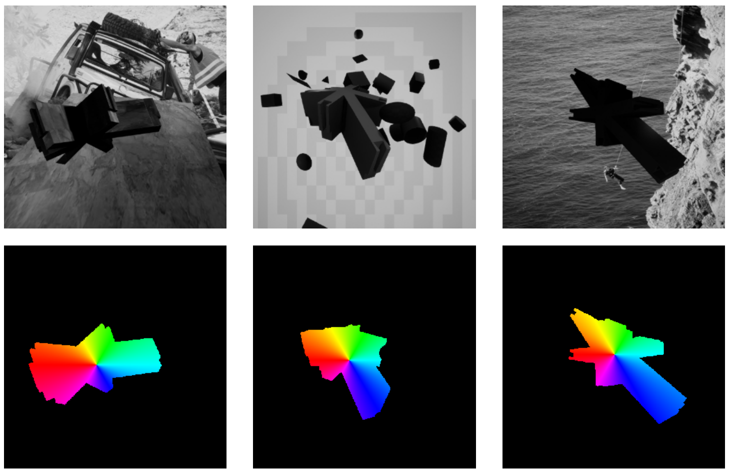


Figure 5.2: Synthetic training data. Top: Examples of synthetically rendered training images. Bottom: Color-coded vector fields. Within the object mask, the vectors point to the 2D projections of corresponding 3D keypoints, specifically the center keypoint.

For each training image, it stores a mask of the visible object region, its pose, and the positions of keypoints in both 3D and 2D space. Grayscale images are employed for training and testing to emphasize shape differences, as the target objects in later experiments are uniformly colored. Figure 5.2 illustrates examples of the training images. Compared to the use of highly realistic synthetic imagery (e.g., [HVG+19] or the BOP data from the previous chapter), this approach offers greater versatility due to its simplified scene configuration.

5.1.2 Detection and Refinement

The 6D object detection network utilized in this chapter is a slightly simplified version of CASAPose from Chapter 4. It performs 2D-3D correspondence estimation for n_c objects, each defined by n_p keypoints in a single pass. Using a single CNN for detection and pose estimation of multiple objects reduces runtime by requiring only one inference per image. By learning to differentiate between geometrically similar objects, the network's descriptive capability is improved, which helps reduce false positives. Building on PVNet [PLH+19], the following enhancements for multi-class handling are added:

1. The semantic segmentation and joint 2D vector fields of unit vectors pointing toward object keypoints are predicted using two decoders connected to the same encoder. The pixel's semantic label guides the prediction of the vector fields.
2. The keypoint decoder applies class-adaptive instance (de)normalization (CLADE) [TCC+21] instead of batch normalization. The estimated semantic segmentation guides the CLADE layers in selecting pixel-wise object-specific weights.

Compared to CASAPose, excluding guided convolution, upsampling (*GCU*), and Differentiable Keypoint Regression (DKR) simplifies the model and enhances computational efficiency. This enables a combined system in which 6D detection and pose refinement operate at real-time frame rates.

Refining each detected pose compensates for the accuracy loss introduced by the simplified model. Subsequent experiments will focus on the advantages of the aforementioned modifications and the combined system. The full CASAPose architecture could enhance performance by providing more accurate initial pose estimates.

Pose refinement is either performed after a valid pose initialization or begins from the valid pose of the previous frame when initialization is not required. The refinement process begins with discrete contour-based optimization (Sec. 3.2.3), which matches hypotheses along scanlines perpendicular to the projected contour. This method minimizes the perpendicular distance between the contour points and their corresponding match hypotheses in the image. Subsequently, dense optimization (Sec. 3.2.1) further refines the object pose. It employs adaptive thresholding (Sec. 3.2.2) to align the rendered and camera images, mitigating the effects of illumination. By combining dense and discrete refinement, the tracking process leverages the complementary contributions of both the silhouette and inner edges. The discrete and dense methods are stabilized using IRLS with Charbonnier weighting (Eq. (3.23)), where $\epsilon = 0.001$. The implementation alternates the methods within the image pyramid, following Sec. 3.2.4.

5.1.3 Pose Validation

In a single-camera system, the presence of objects with subtle geometric variations may necessitate reliable filtering to resolve ambiguities. Similar 2D projections can arise for different 3D objects in certain camera perspectives, especially when the objects are partially occluded. To address this, pose validation helps filter out incorrect pose estimates that arise from drift or false-positive detections. A metric is required to quantify the alignment between an object's 3D geometry and the image, facilitating the continuous evaluation of both CNN predictions and geometric refinement.

The pose detection, refinement, and validation pipeline is shown in Fig. 5.3. Pose validation is divided into two parts. First, detection validation rejects the most inconsistent predictions of the CNN and suppresses noise in its output. After passing this stage, geometric refinement is performed, and the refined pose is either retained for tracking or rejected. For subsequent frames, a temporal heuristic evaluates the geometric fit and, if necessary, triggers reinitialization.

Detection validation In RANSAC-based voting for 2D keypoint detection [PLH+19], a hypothesis is validated if at least 25% of voting pixels are inliers. Inliers are determined using a cosine similarity threshold of 0.99 between the vector from the pixel to the hypothesis and the estimated direction vector. A cosine value close to 1 indicates strong

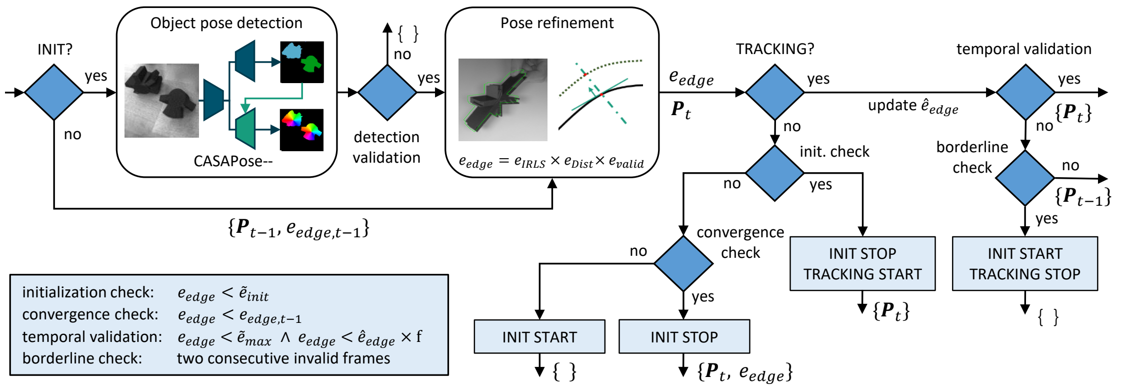

INIT?
yes
no
Object pose detection
CASAPose--
{ }
no
yes
detection validation
Pose refinement
$e_{edge} = e_{IRLS} \times e_{Dist} \times e_{valid}$
$\{\boldsymbol{P}_{t-1}, e_{edge,t-1}\}$
e_{edge}
$\boldsymbol{P}_t$
TRACKING?
yes
no
update $\hat{e}_{edge}$
temporal validation
yes
$\{\boldsymbol{P}_t\}$
no
borderline check
no
$\{\boldsymbol{P}_{t-1}\}$
yes
init. check
no
yes
convergence check
no
yes
INIT STOP TRACKING START
$\{\boldsymbol{P}_t\}$
INIT START TRACKING STOP
{ }
INIT START
{ }
INIT STOP
$\{\boldsymbol{P}_t, e_{edge}\}$
initialization check: $e_{edge} < \tilde{e}_{init}$
convergence check: $e_{edge} < e_{edge,t-1}$
temporal validation: $e_{edge} < \tilde{e}_{max} \wedge e_{edge} < \hat{e}_{edge} \times f$
borderline check: two consecutive invalid frames

Figure 5.3: Illustration of the pose estimation, tracking, and validation pipeline. Refinement and validation at time step t are shown for one object. In practice, individual errors are tracked for each object. P_t represents the refined pose at time t. CASAPose-- indicates a simplified variant of the CASAPose architecture.

alignment, ensuring that only relevant pixels support the keypoint hypothesis. This same test could also be applied to the results of DKR by evaluating the residuals of randomly selected pixels. After pose initialization, the reprojection error must be smaller than a threshold (12 pixels in the conducted experiments) for at least 4 out of 9 keypoints.

Refinement validation The initial pose can still be coarse, so pose validation, including acceptance or rejection, is performed on the refined poses. This process requires the projected contour of the object to match the image content. The residual error of the discrete contour-based refinement is suitable for measuring this.

Here, e_{IRLS} is the mean residual value from the last iteration of IRLS at the lowest pyramid level, reflecting the average of weighted discrepancies between the projected and observed image contours. A lower e_{IRLS} indicates a closer fit. The metric e_{Dist} denotes the mean distance between the projected contour points and their closest correspondence hypothesis, with smaller distances indicating a more accurate match. Finally, e_{valid} is the ratio of the number of scanlines to the number of scanlines on which a correspondence hypothesis was found, penalizing excessive occlusions. The edge matching score is defined as:

$$e_{edge} = e_{IRLS} \times e_{Dist} \times e_{valid}. \tag{5.1}$$

This combined score provides a holistic measure of pose validity. As a product of three metrics, any poor fit—such as high residuals, large distances, or a high ratio of unmatched scanlines—lowers the overall score. It balances each metric's contribution, ensuring no single aspect dominates and enabling a nuanced pose accuracy assessment.

The evaluation of e_{edge} in video sequences involves tracking its progression over time (see Fig. 5.3). First, the initial pose of an object is refined. If e_{edge} is smaller than a threshold $\tilde{e}_{init}$, an initial pose is accepted (*initialization check*). Otherwise, the object is kept as a candidate, and further pose refinement is performed for subsequent frames as long as e_{edge} decreases. If the error increases compared to the last frame, the detection is discarded (*convergence check*). This allows for a high frame rate to be maintained while still enabling ongoing refinement as long as the error continues to decrease.

Next, e_{edge} is monitored continuously, and the moving average $\hat{e}_{edge}$ is updated as long as the object remains valid (*temporal validation*). As long as e_{edge} is smaller than a maximum value $\tilde{e}_{max}$ and $\hat{e}_{edge} \times f$, where f is a predefined factor, an object keeps its validity, ensuring that both absolute error and relative consistency over time are within acceptable bounds. If the error exceeds either threshold, the object enters a borderline state. The pose is not updated, and a mismatch flag is set. If two consecutive invalid frames are detected (*borderline check*), reinitialization is performed with the CNN from the next frame, ensuring robustness against transient errors.

5.2 Experimental Evaluation of the Integrated Pipeline with Custom Objects

The registration pipeline is first evaluated using near-photorealistic synthetic data, followed by a discussion of general observations with real data in the context of AR-guided construction. In the conducted case study, the evaluation focuses on a miniaturized model of a gridshell facade, consisting of 13 node elements (see Fig. 5.4) and 42 connector sticks. The node elements are of particular interest due to their similar, yet not identical shapes, which can lead to geometric ambiguities. More details on the application scenario are discussed in Sec. 7.2.

The evaluation uses the **2D projection** [BMK$^+$16] (**2DP**) metric (see Sec. 4.3) to assess pose estimation. It is particularly relevant for AR applications, where precise alignment of virtual objects with the real world is crucial for visual quality, as it evaluates the pose error in image space. Higher-resolution images allow for a more refined assessment and thus enable more precise pose estimation. To emphasize this, experiments on initial pose estimation (Sec. 5.2.3) use the CNN input resolution of to access **2DP**, while experiments on refinement (Sec. 5.2.4) use full image resolution (see Sec. 5.2.2).

5.2.1 CNN Training and Architecture

The training dataset consists of approximately 15,000 synthetic images per object. The CNN is trained for 125 epochs with a batch size of 42 using the Adam optimizer. The initial learning rate of 0.002 is halved every 25 epochs. The network architecture follows CASAPose, with the aforementioned simplifications and a ResNet-18 [HZRS16] backbone.

Training images have a resolution of 320×320. Image augmentations include random adjustments to contrast and brightness, normally distributed noise, and random rotations

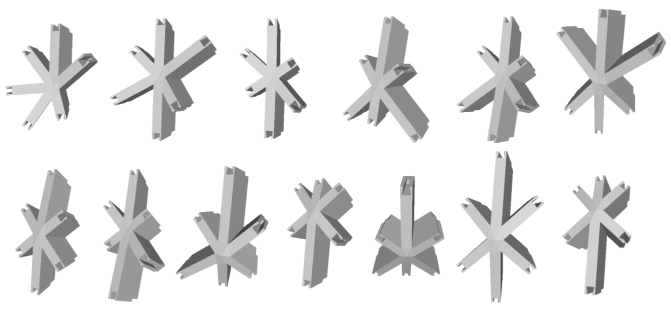


Figure 5.4: Renderings of the geometrically similar reference objects. The 13 objects (*I01–I13*) are numbered from top left to bottom right. The sizes range from 11.7 cm to 15.6 cm, based on the largest distance between any pair of vertices.

and translations. Input images are grayscale; however, to utilize pre-trained ImageNet weights, the intensities are stacked to create a three-channel image. Differentiable proxy voting loss (DVPL) [YZKL20] and smooth L1 loss are applied to learn the vector field, while softmax cross-entropy loss is used for semantic segmentation. During training, the ground truth segmentation is input to the CLADE layers in the vector field decoder. All results for PVNet are computed using a custom TensorFlow implementation of the original code.

5.2.2 CNN Inference and Live Tracking

The developed tracking tool integrates the CNN into a real-time software, leveraging OpenCV with CUDA extensions for inference. Initial poses are estimated from 2D-3D correspondences using EPNP [LMNF09].

All test images have a resolution of 1024×1024, used for pose refinement, while the CNN input size is 400×400. Refinement is performed with three pyramid levels and a scanline length of $t_l = 15$ pixels, and a sampling interval of $t_d = 2$. The process utilizes an off-screen OpenGL renderer to generate intermediate images. While the contour-based component runs efficiently on the CPU, the dense refinement uses custom GLSL compute shaders (see Sec. 3.2.4) for image processing and equation system formation. Only the equation system solving is done on the CPU, minimizing data transfer overhead.

By default, the full pipeline expects images where the object-to-camera distance approximately matches the training data (*default mode*). The subsampled camera image is passed to the detection network, where multiple objects are searched simultaneously and tracked independently. As an extension, *far-range mode* addresses the challenges of an increased object-to-camera distance during testing. In this mode, image patches are used as input to the CNN, ensuring that the object's size relative to the patch matches the median size of objects in the training images (see Sec. 5.2.5, page 91).

5.2.3 Multi-Object Detection of Similar Objects

The following two experiments emphasize the multi-object model's ability to distinguish between similar objects, a challenge that is less pronounced in commonly used pose estimation datasets like Linemod [HLI+12] and YCB Video [XSNF18]. These datasets feature objects that are easily distinguishable due to clear differences in color and shape.

The evaluation dataset consists of 200 images per object, generated synthetically (see Fig. 5.5). A different rendering engine is employed to introduce a domain gap between training and testing data. BlenderProc [DSW+19], built on the 3D creation suite Blender, enables the rendering of nearly photorealistic images. Each image shows an object on a textured surface, surrounded by randomly placed distractor objects from T-Less [HHO+17].

The poses of the camera, object, and light source vary, with the camera positioned in the upper hemisphere of a spherical shell (radius: 17–35 cm, elevation: 30°–89°). It points at the *target mesh*, with random in-plane rotations ($\pm 45°$), out-plane rotations ($\pm 10°$ along the x and y axes), and a horizontal field of view of 46.28°.

Table 5.1 compares the results for three multi-object models across the 13 objects:

1. **PV-M**: This model employs the straightforward multi-object extension of PVNet. With the addition of each object, 19 channels are appended to the network output.
2. **PV-M-C**: This model represents a modification of the baseline architecture that generates joint vector fields as output.
3. **CLADE-PV**: This model adopts the proposed network structure (see Sec. 5.1.2).

The proposed model significantly enhances pose accuracy relative to both baselines. The average recall, using the **2DP** metric, for CLADE-PV is 85.6%, compared to 75.9% for PV-M. Notably, CLADE-PV requires substantially less GPU memory for training than PV-M (which was trained with half the batch size due to higher memory demands), thus accelerating the training process. In multi-object configurations, CLADE-PV supports

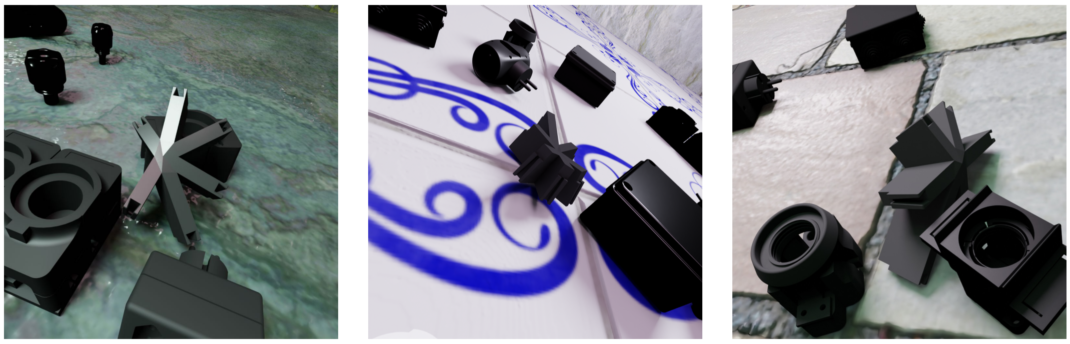


Figure 5.5: Three images from the synthetic evaluation set. Rendering nearly photorealistically introduces a domain gap to the domain randomized training data.

Method	*I*01	*I*02	*I*03	*I*04	*I*05	*I*06	*I*07	*I*08	*I*09	*I*10	*I*11	*I*12	*I*13	**Avg**
PV-M	75.0	78.0	71.0	82.5	93.5	66.5	80.0	73.3	53.5	75.0	85.0	69.0	84.0	75.9
PV-M-C	60.0	54.5	54.0	68.0	62.5	41.5	70.0	54.2	49.5	80.0	68.5	53.5	72.5	60.6
CLADE-PV	**79.0**	**79.0**	**76.5**	**86.5**	**95.5**	**85.0**	**90.0**	**82.4**	**87.0**	**94.0**	**91.5**	**77.0**	**90.0**	**85.6**

Table 5.1: Comparison of the proposed method with the multi-object baselines. Pose accuracy is assessed using the 2DP recall on the synthetic evaluation dataset.

Type	Method	*I*01	*I*02	*I*03	*I*04	*I*05	*I*06	**Avg**
Single	PV-S	76.5	71.0	63.5	84.5	93.5	**86.5**	79.3
Object	PV-S+	**80.5**	70.0	61.0	77.0	87.0	78.0	75.6
Multi	PV-M	75.0	78.0	71.0	82.5	93.5	66.5	77.8
Object	**CLADE-PV**	79.0	**79.0**	**76.5**	**86.5**	**95.5**	85.0	**83.6**

Table 5.2: Comparison of the single-object and multi-object models. Pose accuracy is assessed using the 2DP recall on the synthetic evaluation dataset. For CLADE-PV and PV-M, the same trained network as in Tab. 5.1 are used.

larger batch sizes on the same hardware. This does not apply to PV-M-C, which, however, performs substantially worse, suggesting a poor prediction of joint vector fields without the proposed architectural modifications. The superior performance of CLADE-PV over the baselines underscores the importance of the object-specific parameters, which enhance the model's ability to distinguish between similar objects.

Table 5.2 compares the performance of multi-object and single-object models. Two separate single-object configurations of PVNet are trained for six different objects, with variations in the training data used. The **PV-S** configuration only trains on images from the sub-dataset of a single object. In contrast, **PV-S+** also incorporates randomly selected images of other objects to learn to distinguish its own object from others. In practical applications, the PV-S network is likely to produce more false-positive detections than PV-S+ for similar objects. Both networks either require a separately trained 2D detector or need to process the input image with multiple networks if the identity of visible objects is unknown, resulting in computational overhead.

On average, the single-object model PV-S performs better than its multi-object counterpart, PV-M, confirming the existence of a multi-object gap. Additionally, PV-S is more accurate than PV-S+ on average, indicating that negative examples degrade performance when predicting the poses of target objects. However, PV-S falls short compared to the adapted multi-object network (CLADE-PV). This suggests that, for similar objects, CLADE layers and the more diverse training data enhance the network's descriptive capability, allowing it to outperform even a set of separately trained single-object instances.

Method	Iter.	$2DP_{<5}$	$2DP_{<1}$	$\varnothing r_{\text{err}}$	$P{+}{+}$
init.	–	26.4	0.0	3.50°	–
S1	1	76.5	45.7	1.44°	92.3
S2	1	73.8	27.3	1.54°	**97.8**
S1+2	1	84.3	64.7	0.74°	97.1
S1	3	78.7	52.9	1.20°	90.0
S2	3	83.7	57.8	0.85°	97.7
S1+2	3	**86.5**	**73.9**	**0.59°**	96.5

Table 5.3: Improvement of refinement on initial poses. Influence of two refinement variants (S1 and S2) and their combination (S1+S2), each with 1 or 3 iterations on the initial poses (init.). Results are reported for the synthetic evaluation dataset. The 2DP metric is reported for high-resolution images used in refinement.

5.2.4 Benefits of the Combined Pipeline and Geometric Validation

Local refinement This experiment demonstrates how pose refinement improves the stability of CNN predictions. While the accuracy of the CNN is limited by its input resolution, the refinement process leverages the full image resolution and directly integrates the 3D model. The comparison includes discrete contour-based refinement (**S1**) and dense refinement (**S2**) considered separately, as well as combined (**S1+S2**), alternating the two methods within the image pyramid. The results for **init.** correspond to the initial poses (the results of the previous section) but are recalculated for the full resolution.

Table 5.3 lists the percentage of valid frames regarding the **2DP** metric with respect to high resolution, using thresholds of 5 ($\mathbf{2DP}_{<5}$) and 1 ($\mathbf{2DP}_{<1}$) pixels. The average rotational error ($\varnothing\mathbf{r}_{\text{err}}$), as defined in Eq. (3.6), is computed over all valid frames using a 5-pixel threshold. The percentage of estimates where refinement improves the projection accuracy (**P++**) is also reported. **Iter.** represents the number of iterations through the image pyramid during refinement.

The results show that pose refinement significantly enhances CNN predictions. Both contour-based (**S1**) and dense (**S2**) refinements push many predictions within the defined thresholds, with the combination (**S1+S2**) achieving the best results. A single iteration through the image pyramid of **S1+S2** improves accuracy more than three iterations of **S1** or **S2** alone. **S1** converges quickly, especially in the first iteration, and can bridge larger gaps, while **S2** is more likely to converge in the right direction (as indicated by **P++**). **S1+S2** joins both advantages, which is beneficial for real-time systems. Additional iterations further improve accuracy.

Pose validation During the pose refinement, the error value e_{edge} is the criteria for acceptance or rejection of poses. Ideally, it is larger than $\tilde{e}_{init}$ if the estimated pose is

$\tilde{\mathbf{e}}_{\mathbf{init}}$	$2DP_{<5}$ ↑	C_{OK} ↑	F_{OK} ↑	$F_{OK_{ADD}}$ ↑	FP ↓
0.08	80.4	94.1	95.2	99.5	0.12
0.10	83.2	96.7	92.6	98.6	0.27
0.12	85.5	98.7	89.3	98.6	0.46
0.14	85.6	99.1	85.8	97.7	0.81
0.16	85.7	99.6	77.5	94.7	0.86

Table 5.4: Pose validation results with different initialization thresholds. Results are reported for the synthetic evaluation dataset, with the initialization threshold $\tilde{e}_{init}$ varied. The evaluation metrics are defined in the text. (↓/↑: lower/higher is better)

wrong and smaller than $\tilde{e}_{init}$ if the estimated pose is correct. This experiment evaluates the influence of $\tilde{e}_{init}$. The correctness of a pose is generally determined by **2DP** with a 5-pixel threshold. The entire dataset is processed through the registration pipeline (Sec. 5.1). The multi-object network is applied to each image and approximately 46% of incorrect target estimates are already excluded during the detection validation stage. Three iterations of refinement are allowed. Table 5.4 lists the following percentages:

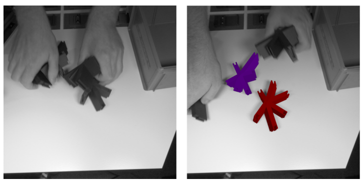
1) $\mathbf{2DP}_{<5}$: Correct target estimates accepted (relative to the number of images).
2) $\mathbf{C}_{OK}$: Correct target estimates accepted (relative to correct target estimates).
3) $\mathbf{F}_{OK}$: Incorrect target estimates declined (relative to incorrect target estimates).
4) $\mathbf{F}_{OK_{ADD}}$: Incorrect target estimates (using the **ADD** [HLI+12] metric with 10% threshold) declined (relative to incorrect target estimates).
5) **FP**: False-positive estimates incorrectly accepted (relative to the number of images).

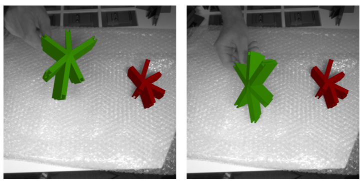
The choice of $\tilde{e}_{init}$ is a trade-off between potentially declining correct estimates if set too low and possibly accepting false detections if set too high. For the test set, $\tilde{e}_{init} = 0.12$ is a good choice. $\mathbf{2DP}_{<5}$ decreases quickly for smaller values and $\mathbf{F}_{OK}$ decreases quickly for larger values. Furthermore, $\mathbf{F}_{OK_{ADD}}$ confirms that more than 98% of the wrong poses that have a substantial offset in 3D space are filtered correctly. For less than 0.5% of the images, false positives are incorrectly accepted.

False-positively accepted objects provide substantial perspective shape ambiguities since not only the wrong object is detected but also its projected contour overlaps with the edges of the image. An interesting observation is that the CNN partitions the segmentation output between multiple candidates when unclear about the object identity (similar as in Fig. 5.8). This results in multiple identities and poses estimated for a single object. If a correct pose cannot be verified, it is likely that a large part of the contour is occluded or not visible. False pose detections of present objects can result from false initial estimates converging to local minima with large edge overlap. In both cases, a small object or camera movement is often sufficient in AR applications with continuous video capture to find a starting position that converges correctly.

Seq.	Method	valid	$I01$	$I04$	$I06$	$I08$	$I10$	$I11$	FP
1	Init.+Ref.	52.7%	6/34	24/154	17/64	447/646	156/376	72/96	1/1215
1	Init.+Ref.+Valid.	87.5%	16/34	138/154	29/64	614/646	313/376	89/96	3/1215
2	Init.+Ref.	78.0%	371/501	–	311/373	–	–	–	0/727
2	Init.+Ref.+Valid.	90.4%	460/501	–	330/373	–	–	–	0/727

Table 5.5: Evaluation of real-data sequences. The number of frames with edge error smaller than $\tilde{e}_{init} = 0.12$ is listed for each object. Only frames where interaction with the object (movement or occlusion) occurs are considered. FP denotes false positives.


(a) appearing objects


(b) textured background

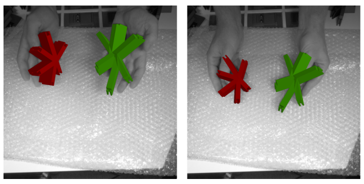

(c) movement on hand palm

Figure 5.6: Tracking and detection in diverse scenarios. (a) is from **Seq. 1**, while (b) and (c) are from **Seq. 2**, both showing objects in varied backgrounds and conditions.

5.2.5 Qualitative Evaluation of the Combined Approach

While previous experiments validated the individual components of the system in the context of the domain gap, this experiment demonstrates the applicability of the approach to real-world data. The evaluated recordings are from a live AR demonstrator guiding a construction scenario in two phases: first, objects are sorted into the correct order (*default mode*), then they are mounted in that order (*far-range mode*). A supplementary video showing these sequences with overlays is available online, complementing the experimental data and images presented in this text[1].

The pipeline (*default mode*) is demonstrated exemplarily in two sequences, captured at 25 fps and 1024×1024 pixels with an industrial camera. While **Seq. 1** includes a white table background, **Seq. 2** features a textured bubble wrap background with stronger gradients and light reflections. Multiple randomly picked objects are moved in front of the camera by hand. The sequences depict fast movements, indirect movements, simultaneous movement of multiple objects, occlusion, and disappearing and reappearing objects.

To evaluate tracking quality, multi-object tracking is applied to the sequences, determining whether to refine or reinitialize individual objects for the next frame based on the validation criteria outlined in Sec. 5.1.3. The result confirms the choice of $\tilde{e}_{init} = 0.12$:

[1]For a dynamic demonstration of the sequences, see the video titled *"Combining Local and Global Pose Estimation for Precise Tracking of Similar Objects / VISAPP 2022"*, uploaded by Computer Vision & Graphics Group Fraunhofer HHI on Feb 10, 2022: https://www.youtube.com/watch?v=siUCev5608g

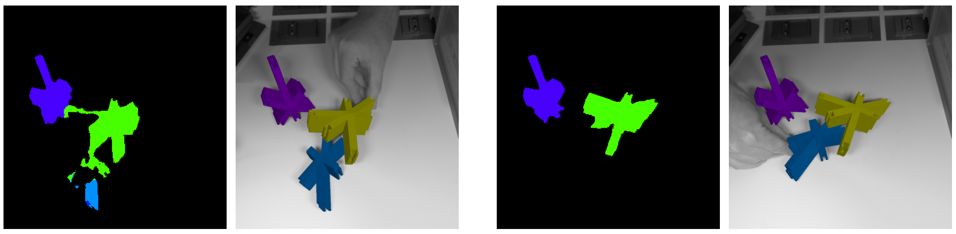


Figure 5.7: Corrective effect of local tracking. Local tracking remains valid (right side of image pair), while the CNN segmentation (left side) would be unreliable.

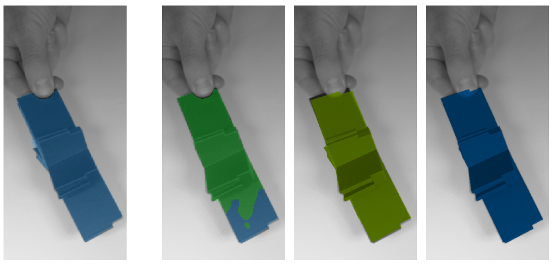


Figure 5.8: Perspective detection ambiguity. In one frame (left), the object is identifiable (semitransparent mask overlay). A small movement in the next frame (second) causes the mask to split, resulting in similar projections for two objects (third and fourth). The false positive will be withdrawn from tracking via temporal verification.

Objects with $e_{edge} < \tilde{e}_{init}$ provide a nearly pixel-accurate overlay without visible pose drift. The local and global registration benefit from each other in multiple ways:

1 During training, the network has never seen multiple objects in an image; during testing, the network is able to estimate the poses of multiple objects simultaneously. Accuracy degradations are compensated by local refinement (Fig. 5.6a).

2 Occlusions between detectable objects were not seen during training, but even if the network is unable to find the occluded object, tracking will continue as long as local tracking remains valid (Fig. 5.7).

3 Fast movements or heavy occlusions possibly stop the local tracking, but it continues as soon as the object is clearly visible again (Fig. 5.1b).

4 False-positive detections of the network are suppressed since the projected contour does not match image content well enough.

The textured background in **Seq. 2** has no impact on the tracking accuracy (Fig. 5.6b). In the course of the sequence, the objects are also moved on the palm of a hand, whereby tracking as well as initialization also succeed in that situation (Fig. 5.6c).

Table 5.5 shows quantitative results. For each object, the number of frames in which it is either in motion or partially occluded—but no more than half (three out of six connectors are visible)—by a moving hand/object is counted. Two configurations are compared: one using the CNN and refinement independent of the previous pose for every frame (**Init.+Ref.**), and another dynamically selecting whether the CNN is needed based on pose validation (**Init.+Ref.+Valid.**). The number of correctly validated poses increases for every object with the latter configuration, as knowledge of the previous pose often provides more meaningful context than the CNN output. Nevertheless, on average, 65% of the frames can be used for reinitializing local tracking, allowing for quick reinitialization if local tracking fails. Fairly low values for $I01$ and $I06$ in **Seq. 1** result from fast movements with motion blur, possibly under occlusion.

In **Seq. 1**, the ambiguous appearance of an object during rotation triggers a false-positive detection (Fig. 5.8). While the network is sure of the correct classification in the next frame, local tracking continues for two more frames but then correctly rejects the object once the projected contour deviates too much from the image content.

Far-range mode Direct application of the trained pose estimation network is challenging when the object-to-camera distance significantly exceeds that in the training data, resulting in smaller objects in the images. For example, in the demonstrator discussed in Sec. 7.2 and shown in Fig. 5.9, objects are tracked from a bird's-eye static camera as they are assembled into a target structure. While expanding the training dataset to include a wider range of object-camera distances is one option, it could complicate data generation and potentially degrade network performance. Alternatively, training a separate 2D object detector and applying the pose estimator to bounding box crops could solve this issue but would increase complexity and slow down processing time.

The *far-range mode* combines tracking and detection without requiring new training data or an additional global detection network. The CNN is applied to a square bounding box centered around the last detection or a predefined start position, ensuring that the object size within the bounding box is consistent with the training data. The bounding box's side length is set to half of the image height, though it could also be scaled dynamically based on the object's size in the previous frame. Since only one object is montaged at a time, the use of CNN can be avoided as long as local tracking is successful, thereby contributing to higher frame rates (approximately 60 fps using an Nvidia GTX 2080 Ti).

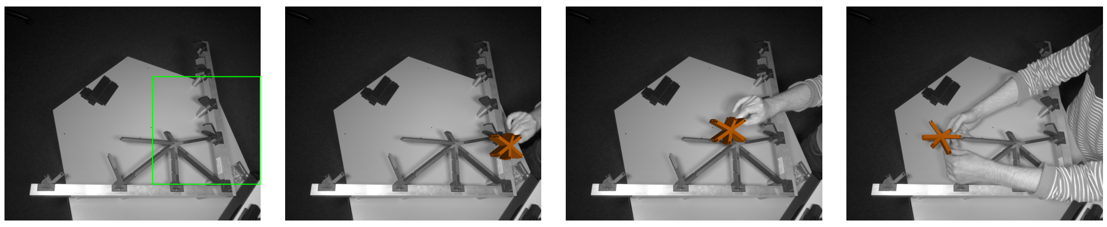


Figure 5.9: Tracking an object in *far-range mode.* A green box highlights the initialization area. The registered object is shown as an orange rendered overlay.

For distant objects, empirical tests show that using two pyramid levels instead of three, along with a lower e_{init}, improves performance.

5.3 Learning Pose Refinement – Future Directions

In the presented detection and tracking pipeline, the two components operate as separate entities, connected in a robust handcrafted system for pose validation and rejection. The state of the art in pose refinement did not employ neural networks (e.g., [SPS+22, TLZQ22]) until the recent introduction of DeepAC by Wang et al. [WYZ+23]. This section outlines ideas for future development, aiming to enhance the interconnection between pose detection and tracking through deep learning and incorporating neural networks into the refinement process.

Differentiable PnP To integrate pose refinement in an end-to-end fashion, all non-differentiable components have to be eliminated from the pipeline. In CASAPose, applying the EPnP operation [LMNF09] within a RANSAC scheme is non-differentiable due to the random sampling and the undefined backward gradient of the PnP operation. Chen et al. argue that the PnP phase encodes the basic geometric properties but cannot influence learning if it is non-differentiable [CPC+20]. They developed a back-propagatable PnP solver (BPnP) using implicit differentiation via the implicit function theorem [KP02]. Differentiable PnP algorithms have been used for iterative pose optimization in the past [LTGD22] and can extend CASAPose to achieve differentiability up to the pose output.

Integrating refinement via differentiable rendering When an initial pose is known, refinement can be performed through iterative rendering and comparison. However, since the renderer's rasterization process is non-differentiable, gradients with respect to the pose cannot be computed using the renderer alone. Differentiable renderers [LB14, RRN+20, CGS+21, WMR24] are specifically designed to produce gradients with respect to their input, including the object pose. Also, the contour-based pose estimation, as presented in Chapter 3, can be interpreted as an extension of a standard renderer designed to efficiently compute gradients with respect to the pose, making it differentiable specifically for relative pose estimation. If the pose estimation network is adjusted to estimate the unoccluded amodal object mask [WML+21] instead of just the visible portion, a silhouette rendered by the differentiable renderer can be directly compared to it. This allows the use of a contour-based loss that is unaffected by object symmetries.

Learning refinement by optimizing least squares Section 4.1.3 demonstrated how to integrate a linear equation system calculation into a CNN layer and adjust the layer's input based on the calculation's outcome. Since the proposed relative pose estimation also relies on solving a linear equation system, this provides a natural attachment point for integrating a learning-based component. In that respect, an extended edge-based optimization could be similar to DeepAC [WYZ+23]. They employ a boundary prediction network that takes the features in a local region around the projected 3D contours as input and outputs a probability distribution of the true boundary locations. Subsequently, the object pose is optimized using Newton's method on the boundary probabilities, thereby generating a module differentiable with respect to the pose. They utilize a MobileNetV2 feature pyramid network (FPN) [SHZ+18] to efficiently extract features with varying receptive fields, similar to how the classical approaches extract different contour images from the image pyramid. In that respect, the guided thresholding (Sec. 3.2.2) can be replaced with an FPN to extract edges that lead the subsequent differentiable pose optimization to a quicker and more robust convergence. Concatenating the camera image with an initial rendered image as network input enables the edge extraction to access both modalities, facilitating the extraction of suitable correspondences.

Object-independent refinement training Pose refinement benefits from access to the known geometry of the object, and for learned refinement methods, it is crucial that the approach remains object-independent. To ensure this, the dataset of objects must be divided into distinct training and validation subsets to effectively measure performance. Simultaneous training on images from multiple datasets is essential to mitigate overfitting to specific dataset characteristics. For example, DeepAC [WYZ+23] trains concurrently on six datasets from the BOP challenge [HSD+20], using separate training and validation object sets, and evaluates performance on three distinct datasets: RBOT [TSSC18], BCOT [LWZ+22], and OPT [WLT+17]. Notably, the training datasets do not require video data, as initial poses for training can be synthesized by introducing random noise to the ground-truth poses of input images.

Learning temporal consistency In the current system, detection operates on individual frames, while tracking relies on frame-to-frame associations. In dynamic, cluttered scenes, severe occlusions can disrupt tracking. Leveraging temporal information from video sequences can replace simplistic assumptions with learned priors, improving robustness. MOTPose [PB24] integrates temporal prior knowledge across multiple frames into multi-object 6D pose detection. Building on DETR [CMS+20], which treats object detection as a set prediction problem, the method generates 1D embeddings, which are transformed into 2D bounding boxes, object classes, and 2D locations of 3D bounding box keypoints using an MLP. These embeddings enable temporal aggregation through linear projection and cross-attention to incorporate sequence knowledge. Importantly, training for temporal stability requires datasets with video data, such as YCBVideo [XSNF18].

While significant architectural differences make integration into encoder-decoder-based architectures challenging, the bottleneck could serve as a promising point of integration for incorporating aggregatable 1D embeddings. These embeddings could generate scaling and shifting parameters for conditional normalization (similar to CLADE), enabling the network to dynamically adjust its feature representations based on temporal context.

5.4 Chapter Summary and Outlook

This chapter has presented a pipeline that links model-based pose refinement with CNN-based pose estimation to enhance AR applications. The findings demonstrate that refinement techniques effectively bridge the gap between synthetic training data and real-world performance. Instead of relying on highly realistic datasets, the approach utilizes simple domain-randomized training data and employs local edge-based refinement to align 3D models with high-resolution image edges. A pose validation step based on edge deviation errors mitigates false-positive detections from CNNs, ensuring precise matching between the projected contours of 3D models and image edges. This is particularly beneficial for industrial applications where high reliability is crucial, and similar untextured objects are common [EM20, HBJ^{+}21].

Moreover, the experiments have shown that the object-adaptive local weights, introduced in Chapter 4, improve CNN pose estimation accuracy for geometrically similar objects. These enabled the multi-object model to outperform a set of single-object models in this challenging scenario.

Prioritizing pose refinement in video sequences has proven advantageous, especially in scenarios not explicitly modeled during training. Knowledge from previous images maintains tracking, and the CNN is evaluated only if local tracking fails, e.g., due to rapid motion or occlusion. In such cases, edge-based pose validation triggers reinitialization.

The evaluation of the integrated pipeline with a set of custom objects has demonstrated its effectiveness in detecting and tracking similar objects in both synthetic and real-world scenarios. The results highlight the advantages of combining local refinement with CNNs, showing robustness against challenges like drift, occlusion, and fast motion. While previous approaches, such as CosyPose [LCAS20], addressed the detection of similar, untextured objects using the T-Less dataset [HHO^{+}17] in static images [LCAS20], there has been limited research on similar objects in motion.

Future research could explore integrating temporal information into the detection component and investigate end-to-end learning for pose refinement (see Sec. 5.3). By the time the presented research was conducted, datasets including real-world videos of moving objects with pose annotations were very limited. The recent large-scale dataset HOT3D [BSM^{+}24], which focuses on hand-object interaction, will allow for the extension and further validation of the presented findings.

6 Semantically Guided Model-based Camera Localization

The 6D pose estimation approach proposed in Chapter 4 is model-dependent, requiring prior knowledge of the target geometry, which is explicitly utilized during AI training. This chapter transitions from object-centric methods to a model-independent approach by developing a pose estimation method capable of handling unseen models. The scope of observation also changes: instead of localizing objects in an image, the focus is on localizing the camera image within a geometric model of the environment.

The geometric 3D models used are derived from structured representations, such as architectural building models or floorplans, offering an alternative to the point clouds or image databases typically used for localization (e.g., [TOS$^+$18, XG22, CCPB24]). As a result, the deployment of localization in new environments is based on an objective representation, eliminating the need for manual on-site scans. Additionally, the model defines the coordinate system, rather than relying on an arbitrary scan origin, benefiting AR and navigation applications. New virtual objects can be placed straight into the model, that creates a direct link to the real environment.

Focusing on indoor scenes facilitates model-independent localization due to the structural similarities of human-made spaces: the environments share common architectural features. While apartments or office buildings may have varying room shapes and layouts, they typically consist of reconfigurations of walls, floors, ceilings, windows, and doors.

The challenge lies in establishing a robust match between the environment model and the structural information within the camera image. This involves maximizing the structural overlap between the image and the model, considering an estimated camera perspective. CNNs are not particularly adept at directly comparing images and 3D models, especially when the model is represented as an untextured or semantically annotated mesh. Compared to pixel grids, meshes present a more complex, irregular, and significantly different representation [GRL18].

This chapter contributes by reformulating the image-to-model matching problem as an image-to-image matching task, where a perspective RGB image is matched to a set of semantically rendered reference panoramas, along with the estimation of the relative pose. The matching process correlates the deep embedding of the RGB image with the embeddings of the omnidirectional semantic panorama images, which label fundamental structural features, as shown in Fig. 6.1.

Rendered panoramas capture the environment in all directions, offering comprehensive spatial information and reducing the number of views needed to represent a space. In contrast, perspective reference images would significantly expand the search space and the risk of missing critical information due to their limited fields of view [OB21]. To address the differing fields of view between panorama references and perspective queries, the unique approach is to use a neural network to predict the perspective image's *viewport* within the panorama—the exact region visible to the camera.

A RetinaNet-like architecture [LGG$^+$17] predicts and classifies the 2D bounding box around the viewport. The resulting classification score is used to retrieve the closest panorama, and the exact 6D pose is determined through relative pose estimation with an MLP starting from the panorama's position. The approach learns wide-baseline relative pose estimation and accurately predicts poses with few reference renderings.

By leveraging panoramic references and a cross-domain matching, the proposed method, SPVLoc, enhances the robustness and generalization capability of camera localization in new environments. It relies on a minimalist 3D semantic building model [ZZL$^+$20], which could be automatically extracted from simple 2D plans [LZYZ21, PK21, CBTHE23].

In the experiments, publicly available large-scale datasets, including Structured3D [ZZL$^+$20] and Zillow Indoor [CHL$^+$21], are used for training and testing. By sampling perspective query images from equirectangular panoramas, the dataset is efficiently augmented, generating an almost infinite amount of training data for learning matching and pose estimation. Compared to the state of the art in localization for unseen indoor scenes (i.e., [MKB$^+$22, CWVP24]), the presented method is the only one that estimates all degrees of freedom of the camera pose. Additionally, the estimates are more precise in terms of both the number of localized images and the accuracy of localization.

6.1 Challenges of Model-Based Indoor Localization

Indoor camera localization is a fundamental challenge of computer vision, aiming to precisely determine a camera's position and orientation within an indoor environment. Applications include navigation [WFU15, TOS$^+$18], AR [ARKW19], building management [AGK$^+$25], and building digitization [LSK$^+$15]. Current methods often depend on extensive scene-specific data acquired through the manual creation of databases with localized images, depth information, or point clouds [KGC15, TOS$^+$18, XG22]. However, this limits their applicability, especially in situations where exploring the building is impractical or data acquisition is costly and time-consuming.

Recent methods incorporate synthetic data during training but require retraining or fine-tuning for new scenes using detailed BIM (Building Information Modeling) models [AKW19, ATK23]. The adoption of the Structured3D format [ZZL$^+$20] streamlines and standardizes 3D structure annotation by simplifying complex models into their most fundamental geometric representations. This simplification aligns with how humans perceive

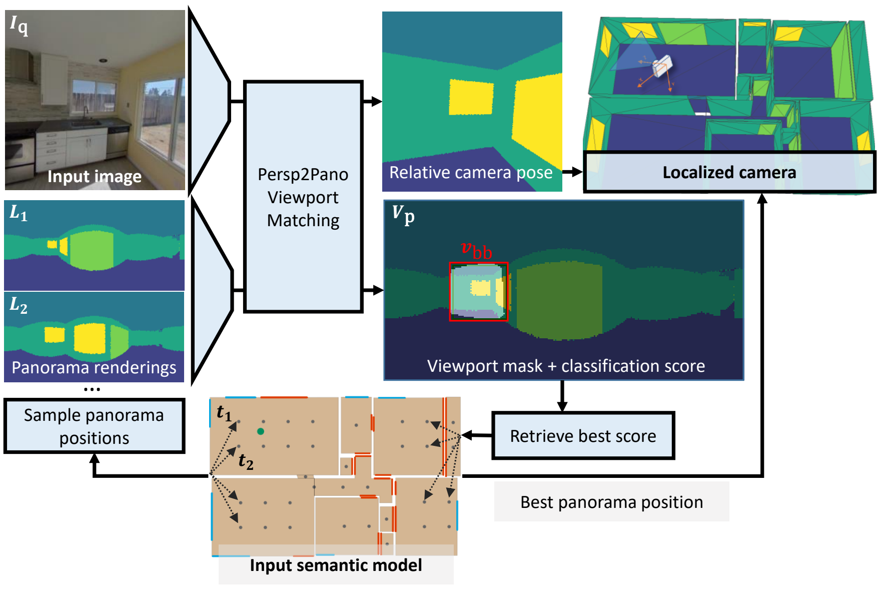


Figure 6.1: SPVLoc for 6D indoor localization. The method estimates the 6D camera pose in a semantically annotated 3D reference model by determining its relative pose to rendered semantic panoramas. The best panoramic match is found through semantic viewport matching. The binary viewport mask $\mathbf{V}_\mathrm{p}$ (shown as an overlay) represents the visible region of the image $\mathbf{I}_\mathrm{q}$ in a panorama $\mathbf{L}_i$, with $\mathbf{v}_\mathrm{bb}$ as the bounding box around it.

3D scenes, relying on salient global structures such as lines, contours, planes, smooth surfaces, symmetries, and repetitive patterns to navigate and interpret their environment [ZZL+20].

Methods that match a query image to simplified structural representations and utilize AI to learn the matching across a large set of scenes exhibit robustness in unexplored, dynamic environments [HJRSP21, HJP22, MKB+22, CWVP24]. However, these existing methods mainly focus on localizing a camera image with respect to the 2D floor plan, missing at least two degrees of freedom for rotation and one degree of freedom for the position. Additionally, some methods are specifically tailored to the localization of equirectangular panoramic images and are therefore unsuitable for localizing perspective camera images [HJRSP21, HJP22]. The panoramic queries are captured under controlled conditions, involving a tripod-mounted camera with a precise horizontal alignment and a fixed camera height [HJRSP21]. However, in contexts such as AR applications or scenarios where archived photos must be accurately registered, e.g., to annotate visible building details into a digital building model, ideal capture conditions cannot always be guaranteed. In such cases, the demand for precise 6D localization, such as that offered by the proposed SPVLoc method, becomes imperative.

6.2 Global Localization in Semantic Building Models

The proposed SPVLoc (**S**emantic-**P**anoramic **V**iewport 6D **Loc**alization) performs 6D indoor localization of a 2D RGB query image within a simple semantic textureless 3D scene model. The model adheres to the Structured3D annotation format [ZZL$^+$20], containing walls, doors, windows, ceiling, and floor as planes (see Fig. A.7, page 145 for examples). It can be derived directly from building models [ZZL$^+$20] or from floor plans using contemporary AI tools [LZYZ21, PK21, CBTHE23].

SPVLoc, as illustrated in Fig. 6.1, first estimates the image's viewport (Sec. 6.2.1) within a set of rendered semantic panorama images from the semantic scene model using cross-domain image-to-panorama matching (Sec. 6.2.2). The predictions are ranked based on classification accuracy, and relative 6D pose regression is performed with respect to the best-matching reference panorama (Sec. 6.2.3). The reference panorama positions are sampled at random nearby locations during training (Sec. 6.2.4), or are placed on a grid superimposed on top of the floorplan during testing (Sec. 6.2.5).

6.2.1 Problem Definition

The objective is to accurately estimate the 6D camera pose $\mathbf{P}$ of a query RGB image $\mathbf{I}_\mathrm{q}$, relative to a semantically annotated 3D model $\mathcal{S}$. To achieve this, the localization of $\mathbf{I}_\mathrm{q}$ within $\mathcal{S}$ is reformulated as a cross-domain image-to-panorama matching problem. The model is represented by a set of n reference layouts $L = \{\mathbf{L}_i \mid \mathbf{L}_i \in \{0, 1, \ldots, m\}^{w_p \times h_p},\ i = 1, \ldots, n\}$, where each $\mathbf{L}_i$ is a synthetic equirectangular panorama consisting of $m+1$ semantic labels. A layout captures the entire environment around a specific vantage point $\mathbf{t}_i \in \mathbb{R}^3$. It is generated from a pose $\mathbf{P_i} = (\mathbf{t}_i, \mathbb{I}_3)$, where $\mathbb{I}_3$ is the identity rotation matrix, aligning the layout to a horizontal and northward orientation.

The viewport $\mathbf{V}$ is a distinct *cutout* section from a panorama, defining the exact region a camera positioned nearby can see. The perspective distortion of $\mathbf{V}_i$ contains information about the relative 6D pose offset $\Delta\mathbf{P}_i = (\Delta\mathbf{t}_i, \Delta\mathbf{R}_i)$ between $\mathbf{L}_i$ and $\mathbf{I}_\mathrm{q}$.

Each viewport $\mathbf{V}$ is represented by a viewport mask $\mathbf{V}_\mathrm{P} \in \{0,\ 1\}^{w_p \times h_p}$, a binary image that indicates which points in the panorama $\mathbf{L}$ are visible in the perspective camera image $\mathbf{I}_\mathrm{q}$, given the camera's pose offset $\Delta\mathbf{P}$. A 2D bounding box $\mathbf{v}_\mathrm{bb}$ represents the minimal axis-aligned bounding box surrounding the visible points in $\mathbf{V}_\mathrm{p}$. It is circularly defined within the 360° panorama, allowing the overlap on the right image side to re-enter on the left side of the panorama.

For each pair of $\mathbf{I}_\mathrm{q}$ and $\mathbf{L}$, the proposed network predicts $\Delta\mathbf{P}$, but also $\mathbf{V}_\mathrm{P}$ and $\mathbf{v}_\mathrm{bb}$. A confidence score c is predicted for the outputted bounding box $\mathbf{v}_\mathrm{bb}$, quantifying the probability of it being a valid match. Thereby, during estimation, the reference panoramas are assigned a set of scores $C = \{c_i \mid c_i \in \mathbb{R},\ i = 1, \ldots n\}$, reflecting the quality of the estimated relative poses. For example, if there is no visual overlap between the image and the panorama, c is expected to be zero, whereas for nearby positions, it is expected

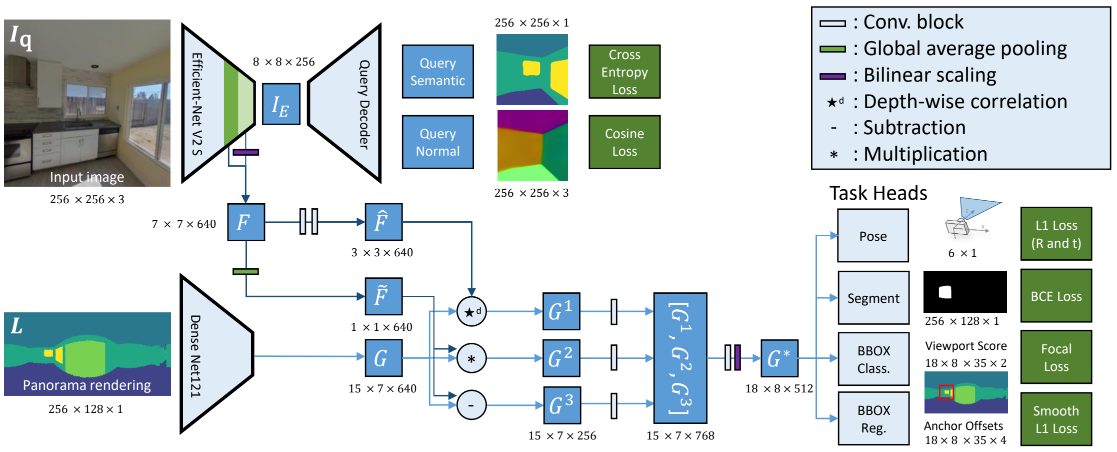


Figure 6.2: Network architecture overview. The information from the query branch is fed into the panorama branch via depth-wise correlation. Three task heads predict the corresponding viewport, and one task head predicts the relative pose offset.

to be high. Scoring and ranking the viewports aids in selecting the optimal panorama $\mathbf{L}_{i^*}$, where $i^* = \text{argmax}(C)$ is the index of the layout with the highest score.

The absolute pose of the camera in the world coordinate system is obtained by concatenating $\mathbf{P}_{i^*}$ with $\Delta\mathbf{P}_i$, yielding $\mathbf{P} = (\Delta\mathbf{t}_i + \mathbf{t}_i^*, \Delta\mathbf{R}_i^*)$. Identifying the viewport and the corresponding pose offset poses several challenges. First, as the distance between the two cameras increases, the perspective distortion between the image and the panorama viewport also increases. Second, the matching involves a severe domain gap as an RGB image is matched against a rendered semantic panorama. Third, the network must identify subtle differences in the image to resolve ambiguities from similar-looking rooms during matching.

6.2.2 Semantic Panoramic Viewport Matching

The matching network extracts deep features from the query image $\mathbf{I}_\text{q}$ and correlates them with features of each semantic reference panorama $\mathbf{L}$, which serves as the global context. The correlated features are then decoded using several prediction heads to estimate the pose offset $\Delta\mathbf{P}$ (Sec. 6.2.3), the viewport bounding box $\mathbf{v}_\text{bb}$ together with the classification score c, and the pixel-wise viewport mask $\mathbf{V}_\text{p}$. Figure 6.2 shows the proposed network, with references to the component names used throughout the text.

Input encoding Both $\mathbf{I}_\text{q}$ and $\mathbf{L}$ are initially encoded using vision backbones. The choice of backbones enhances the encoding capacity for $\mathbf{I}_\text{q}$, as extracting relevant information from the photo is more complex than from the semantic panoramas. A lightweight network is used for encoding panoramas, speeding up the process for large reference sets.

For $\mathbf{I}_\mathrm{q}$, an *EfficientNet-S* backbone [TL19] is used, preserving the features after the fourth and final downsampling operations. Subsequently, both outputs are rescaled to a tensor of spatial size of 7×7 and concatenated to a tensor $\mathbf{F} \in \mathbb{R}^{7\times7\times640}$.

For $\mathbf{L}$, a *DenseNet-121* backbone [HLVDMW17] is employed, extending up to the fifth downsampling operation to extract features $\mathbf{G} \in \mathbb{R}^{w'\times h'\times640}$, where $h' = (h_p/16) - 1$ and $w' = (w_p/16) - 1$, accounting for padding effects from DenseNet's downsampling.

Feature correlation The extracted information from the input image guides the panorama branch in estimating the viewport mask and bounding box. This is achieved through the application of a *depth-wise correlation* technique, denoted as $\star_\mathbf{d}$ [AFS+18, MGGL21]. $\mathbf{F}$ is processed through two consecutive *convolution blocks*, each incorporating convolution, batch normalization, and ELU activation, resulting in a set of 3×3 filters $\hat{\mathbf{F}}$. Depth-wise correlation computes the similarity between the panorama features $\mathbf{G}$ and the filters $\hat{\mathbf{F}}$, producing $\mathbf{G_1}$. The correlation is performed separately for each channel, enabling the model to capture distinct spatial patterns. Furthermore, $\mathbf{F}$ undergoes compression via *global average pooling*, resulting in $\tilde{\mathbf{F}}$. It is then element-wise multiplied and subtracted from $\mathbf{G}$, producing two additional feature tensors, $\mathbf{G}_2$ and $\mathbf{G}_3$.

$$\mathbf{G}_1 = \mathbf{G}\star_\mathbf{d}\hat{\mathbf{F}}, \quad \mathbf{G}_2 = \tilde{\mathbf{F}} \odot \mathbf{G}, \quad \mathbf{G}_3 = \tilde{\mathbf{F}} \ominus \mathbf{G} \tag{6.1}$$

The three feature tensors with injected image information are individually processed with an additional convolution block and then concatenated. The concatenated features undergo a final convolution block (along with minor upscaling to facilitate the later linking of the output features with the *Pose* head), resulting in the correlated features $\mathbf{G}^\star$, that are linked to the task heads.

The matching employs feature correlation, where several operations fuse backbone features $\mathbf{F}$ and $\mathbf{G}$. Previous studies applied feature correlation for template-based unseen object detection and their ablation studies support the necessity of the presented module design. Ammirato et al. [AFS+18] showed that combining element-wise multiplication and subtraction with the $1 \times 1 \times N$ tensor $\tilde{\mathbf{F}}$ improves matching, justifying $\mathbf{G}^2$ and $\mathbf{G}^3$. Mercier et al. [MGGL21] showed that correlation with the $3 \times 3 \times N$ tensor $\hat{\mathbf{F}}$, along with concatenation of the three tensors, enhances matching accuracy, justifying $\mathbf{G}^1$.

Viewport decoding The correlated features $\mathbf{G}^\star$ are linked to four task heads: one estimates the relative pose offset (Sec. 6.2.3), while the other three predict the viewport of the perspective camera within the layout of the omnidirectional panorama.

Two network heads (*BBox Reg.* and *BBox Class.*) predict the viewport bounding box $\mathbf{v}_\mathrm{bb}$ and its corresponding classification score c using *anchor-based* detection, similar to RetinaNet [LGG+17]. These anchors are predefined bounding boxes of different aspect ratios and scales, centered around potential viewport locations in the panorama. The network learns to predict the most likely viewport by matching these anchors to the ground truth, enabling accurate localization of the viewport in the panorama.

At each feature map location, 35 anchors are evaluated using 7 scales (20, 40, 60, 80, 100, 120, 140 pixels) and 5 aspect ratios (0.4, 0.6, 0.8, 1.0, 1.2), with the square of the scale representing the box area. These anchors cover the ranges for describing viewports in a 256×128 equirectangular panorama, including full-image and sufficiently small boxes. Bounding boxes crossing the image edge wrap around to the opposite side.

The *BBox Reg.* head refines the position and size of the bounding box by predicting the adjustments (e.g., offsets) to the anchor box coordinates. Its output is optimized using a smooth L_1 loss $\mathcal{L}_1^{\mathrm{VP}}$, encouraging accurate predictions of the bounding box coordinates.

The *BBox Class.* head assigns a classification score to each anchor, indicating the probability of a correct match with the ground truth. The box with the highest score is considered the viewport estimate $\mathbf{v}_{\mathrm{bb}}$. The classifications are optimized using focal loss $\mathcal{L}_2^{\mathrm{VP}}$ to address class imbalance and focus on hard-to-classify examples [LGG+17].

The *Segment* head predicts the exact viewport mask $\mathbf{V}_{\mathrm{p}}$ using a convolutional decoder optimized with binary cross-entropy loss $\mathcal{L}_3^{\mathrm{VP}}$ for accurate pixel-wise segmentation.

Perspective supervision The encoded image features, denoted as $\mathbf{I_E}$ (see Fig. 6.2), are fed into a convolutional decoder, denoted *Query Decoder*, that outputs the corresponding semantic and normal maps. This is intended to extract semantically meaningful features that closely correspond to the image content. The inclusion of normal estimation helps capture geometric information, such as the angles between walls, which may not be fully represented in the semantic map alone. The semantics are learned with cross-entropy loss $\mathcal{L}_4^{\mathrm{VP}}$ and the normals are learned with cosine loss $\mathcal{L}_5^{\mathrm{VP}}$.

6.2.3 Feature-Correlation-based Pose Regression

The correlated features, denoted as $\mathbf{G}^\star$, encode information about the viewport of $\mathbf{I}_{\mathrm{q}}$ within a $\mathbf{L}$, but should also lead to relative pose offset $\Delta\mathbf{P}$ between them. Since L is synthetically generated, the precise position of $\mathbf{L}$ is known. Therefore, determining the relative position of $\mathbf{I}_{\mathrm{q}}$ in relation to $\mathbf{L}$ is equivalent to determining its absolute position. All rendered panoramas exhibit identity orientation, due to horizontal alignment and a northward orientation. Consequently, the rotation delta directly corresponds to the absolute rotation of the camera.

The *Pose* head processes $\mathbf{G}^\star$ to estimate the relative pose offset of the camera with respect to a panorama rendering. The architecture is shown in Fig. 6.3. This process starts with three convolution blocks, each followed by a max-pooling layer to compress the features. The flattened features form a 1D embedding $\mathbf{g}^\star \in \mathbb{R}^C$. Here, $C = 640$ for a panorama of dimensions 256×128.

Field of view (FoV) awareness To handle varying camera FoVs during training and testing, $\mathbf{g}^\star$ is normalized and rescaled with the horizontal FoV angle in radians, extracted from the camera matrix $\mathbf{K}$. This enables the model to become aware of variations in camera properties and generalize across different setups. An analogy can be drawn

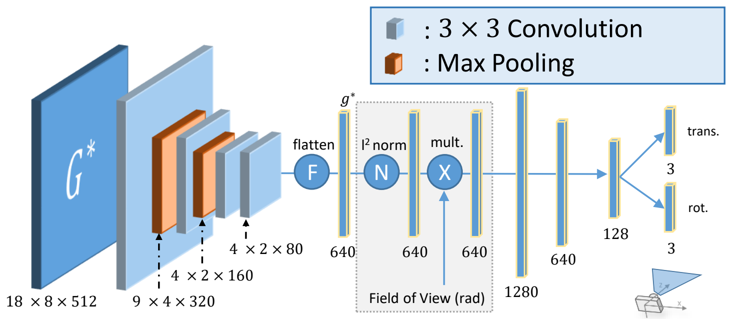


Figure 6.3: Architecture of the *Pose* head. Convolutions compress the input features, an MLP regresses the pose, and a side input is used to adapt to varying FoVs.

to the *vertigo effect* in cinematography, where the perceived size of the image center remains constant while the observer's position and FoV change simultaneously. Without knowledge of the current FoV, estimating the distance between the camera and objects would be nearly impossible in many cases. Unlike viewport mask estimation, which focuses on detecting visual correspondences regardless of the camera's FoV, pose estimation requires adaptation to account for how the FoV affects the position of the camera.

Pose regression The scaled features vector is fed into a four-layer MLP. It outputs the 3D offset to the query image location $\Delta\mathbf{t}_i \in \mathbb{R}^3$ and a rotation given as the logarithm of the unit quaternion $\Delta\hat{\mathbf{r}}_i = \log(\mathbf{q}_i) \in \mathbb{R}^3$ [TBS+21].

The logarithm of the unit quaternion is chosen because it provides a 3D representation of $\Delta\mathbf{R}$ that is not over-parametrized, represents a valid rotation without additional constraints, and can be directly optimized using L_1 or L_2 loss [BGK+18].

A unit quaternion $\mathbf{q} = [u, \mathbf{v}]$ is represented by a scalar u and a 3D vector $\mathbf{v}$ with $\|\mathbf{q}\| = 1$. Its logarithmic mapping is given by

$$\log(\mathbf{q}) = \begin{cases} \frac{\mathbf{v}}{\|\mathbf{v}\|}\cos^{-1}(u) & \text{if } \|\mathbf{v}\| \neq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{6.2}$$

Exponential mapping converts the logarithmic form $\mathbf{w} = \log(\mathbf{q})$ back to a unit quaternion: $e^{\mathbf{w}} = [\cos(\|\mathbf{w}\|), \frac{\mathbf{w}}{\|\mathbf{w}\|}\sin(\|\mathbf{w}\|)]$ [Sol17].

The MLP is evaluated with two loss functions: $\mathcal{L}_1^{\mathrm{P}}$ for the relative translation error and $\mathcal{L}_2^{\mathrm{P}}$ for the rotation error, both using the L_1 distance.

6.2.4 Training and Optimization

The training loss is the sum of the two losses of the MLP and the five losses of the viewport estimation. With seven losses overall, balancing the multi-task learning process requires using different weights for each loss. As in [KC17, KGC18], an exponential mapping is used to weight the multiple loss functions. The learnable weighting parameters β_i $(i = 1, 2)$ for the MLP losses and γ_j $(j = 1, \ldots, 5)$ for the viewport estimation losses are optimized during training, taking into account the homoscedastic uncertainty of each task. The final loss is given as:

$$\mathcal{L} = \sum_{i=1}^{2} (\mathcal{L}_i^{\text{P}} e^{-\beta_i} + \beta_i) + \sum_{j=1}^{5} (\mathcal{L}_j^{\text{VP}} e^{-\gamma_j} + \gamma_j). \tag{6.3}$$

View sampling During training, the 6D pose $\mathbf{P} = (\mathbf{t}, \mathbf{R})$ and the camera matrix $\mathbf{K}$ of the query image $\mathbf{I}_\text{q}$ in the model $\mathcal{S}$ are assumed to be known. For each query, the viewport and relative pose offset are calculated against multiple references.

A total of s_1 randomly positioned panoramas are rendered around $\mathbf{t}$ within a radius of $\pm\, r_1$ in xy-direction and $\pm\, r_2$ in z-direction (upward). Additionally, s_2 *negative* examples are generated to enhance the network's ability to distinguish subtle room differences. Negative samples are sampled across the map, outside of radius r_1 around $\mathbf{t}$, likely in a different room. If a negative pose is sampled in a different room, both mask and target scores for all anchors are zero; in the same room, they are computed as for positive samples, yielding larger offsets. The *Pose* head is not evaluated for negative samples.

For datasets including localized panoramas [ZZL+20, CHL+21], as in later experiments, $\mathbf{R}$ and the FoV in $\mathbf{K}$ can be randomly sampled in predefined ranges (e.g., Euler angle ranges for $\mathbf{R}$). Perspective training views are resampled from panoramas. Varying $\mathbf{K}$ and $\mathbf{R}$ consistently generates new perspectives during training, providing effective augmentation that aids generalization to unseen scenes. Additionally, varying $\mathbf{K}$ allows generating enough variation to incorporate dynamic FoV handling (Sec. 6.2.3).

Viewport construction The ground truth for $\mathbf{V}_\text{p}$, required while training the CNN, is generated from the rendered depth maps of the panorama and the perspective image ($\mathbf{D}_\text{p} \in \mathbb{R}^{w_p \times h_p}$ and $\mathbf{D}_\text{c} \in \mathbb{R}^{w \times h}$), $\mathbf{K}$, and $\Delta\mathbf{P}$.

First, a 3D point map $\mathbf{Q}_\text{p} \in \mathbb{R}^{w_p \times h_p \times 3}$ is derived from $\mathbf{D}_\text{p}$, mapping each pixel's depth to a 3D point in the panorama camera's coordinate system. The inverse pose matrix transforms the points into the perspective camera's coordinate system, where (x_c, y_c) are the projected 2D coordinates, and π represents perspective division:

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = \pi(\mathbf{K} \cdot \mathbf{Q}_\text{p2c}) \quad \text{with} \quad \mathbf{Q}_\text{p2c} = \Delta\mathbf{P}^{-1} \cdot \mathbf{Q}_\text{p}. \tag{6.4}$$

Second, the binary viewport mask $\mathbf{V}_\text{p}$ is set to 1 at all locations where (x_c, y_c) lies within the perspective image bounds and $\mathbf{D}_\text{p}$ is consistent with $\mathbf{D}_\text{c}$ at the projected location within a small threshold ϵ. Finally, to smooth small inaccuracies, $\mathbf{V}_\text{p}$ is refined with a morphological closing operation with *wrap*-padding to preserve panorama edge continuity.

6.2.5 Inference

During inference, a set of synthetic panoramas $L = \{\mathbf{L}_i \mid \; i = 1, \dots, n\}$ represents the model $\mathcal{S}$ against which $\mathbf{I}_\mathrm{q}$ is matched. The panorama positions are determined using a regular 2D grid overlaid on the floor plan, with virtual cameras placed at a fixed height h_{pano} above the floor. The spacing between grid points, specified as xy_{pano}, is equal along both the x and the y-axes. Virtual cameras are exclusively sampled above floor polygons to ensure valid reference images inside rooms. Setting xy_{pano} as large as possible is advantageous in order to minimize the number of references, given that feature correlation and bounding box regression are executed for every panorama.

To avoid rendering during inference, reference panoramas can be pre-generated for a floor plan. Since the encoding of panoramas is independent of the image encoding, the entire set of references can be *DenseNet-121*-encoded in advance and cached, significantly streamlining the inference process.

To rank the panoramas and to determine the set of scores C, only the *BBox Class.* head is required. It outputs probabilities for each bounding box anchor in a view, and the highest probability (indicating the probability of the most likely viewport) serves as the score c_i to rank each panorama. Because only one match is permitted per reference, *non-maximum suppression* is not needed. The *Pose* head is evaluated for the top-n highest bounding box classification scores in C. The absolute poses are then computed following the approach outlined in Sec. 6.2.1.

Refinement Without changing the network architecture or training procedure, SPVLoc can refine an estimated pose by rendering a new reference panorama at the updated estimated position $\Delta\mathbf{t}_i + \mathbf{t}_i^*$ and then re-running the relative pose regression. Provided that this updated position is closer to the true position than the original reference position, the second iteration effectively simplifies the pose estimation task, enabling higher precision in the refined pose estimation.

6.3 Experiments

Several experiments validate the proposed method. Images are localized in scenes consisting of a single building floor with varying sizes and layouts that were not seen during training, and the quality of the estimated camera poses is assessed.

The **translation error** t_err (in cm) is the Euclidean distance between the estimated and ground truth 3D positions. The angular **rotation error** r_err (in degrees) is the geodesic distance between the estimated and ground truth 3D rotation matrices (Eq. (3.6)).

The **recall**, defined as the fraction of frames localized correctly within a threshold ($t_\mathrm{err} < \tau$ and in some cases $r_\mathrm{err} < \upsilon$), is reported for different threshold values. During retrieval, the top-3 best matches are selected, and recall metrics are provided for both the

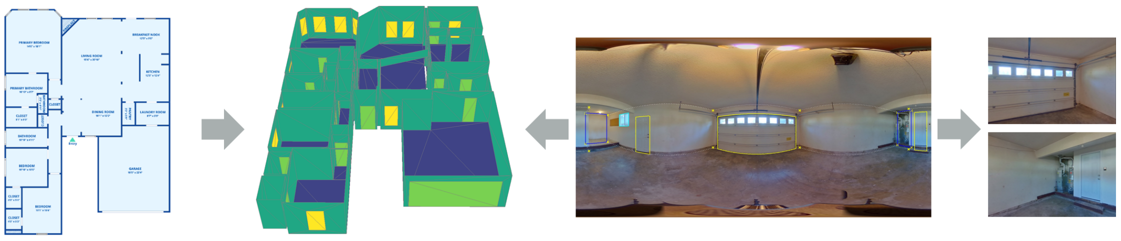


Figure 6.4: ZinD data preparation. Annotations generate 3D reference models (left), while resampled panoramas create perspective train and test images (right).

top-1 match and the best match among the top-3. Additionally, the **median rotation** and **translation errors** are reported for all poses where $t_{\mathrm{err}} < 1\,\mathrm{m}$.

6.3.1 Datasets

Two publicly available and widely used datasets are employed, both containing large quantities of data to train a scene-agnostic 6D localization network.

Structured3D (S3D) [ZZL+20] consists of 3,500 near-photorealistic professionally modeled indoor environments, each annotated with ground truth 3D structure information. It encompasses 21,835 panoramic images, each provided in both furnished and unfurnished configurations. The authors define a unified 3D representation of scenes, referred to as the S3D format, using geometric primitives—planes, lines, and junction points–along with matrices to capture their incidences and intersections. Semantic annotations provide interpretability with classes such as floor, ceiling, wall, door, and window.

Zillow Indoor (ZInD) [CHL+21] comprises 67,448 panorama images taken in 1,575 unfurnished residential homes, all globally aligned and registered to a floor plan. Each floor is treated as a separate scene, as is common in the literature [HJP22, MKB+22, CWVP24], resulting in 2,462 scenes. The 3D annotations of panoramas within a scene are suitable for transformation into the S3D representation format. However, this conversion requires custom processing, developed as part of this thesis, to automatically correct minor annotation inaccuracies and generate clean models. Prior to training, all data is transformed into the S3D format. While S3D includes door and window annotations as rectangular areas within walls, ZInD adds annotations for *openings*. These openings indicate where two rooms connect without a door, cutting through walls to create a passage between them. Openings increase the structural variation of the semantic renderings. Figure 6.4 shows an example reference model generated from Zillow annotations.

For both datasets, the official train-test scene split is used, with 238 scenes reserved for testing in ZInD and 250 scenes for testing in S3D. To enhance control over the FoV and angle variation, perspective test images are sampled from the RGB panoramas. They are registered in the 3D model, so the semantics are known for each perspective image. Test sets suitable for 6D pose estimation and structured evaluation are compiled, consisting of square perspective images with viewing angles of 60, 90, and 120°, respectively, with variations in roll and pitch, and random yaw angles for ZinD and S3D. Two criteria are applied to the sampled yaw to ensure meaningful views. First, the images contain a minimum of three semantic classes, like wall-floor-ceiling or wall-window-floor. Second, the camera faces inside the room rather than towards the next wall. The criteria are applied to the 60° images (4,420 for ZiND and 1,142 for S3D). Subsequent variations in opening angles and roll-pitch configurations are generated from this starting point.

Appendix A.4 contains more details on the test sets, including the distribution of scene sizes, camera height, visualizations of 3D models (Fig. A.7 on page 145), and overviews of the capture positions and test images for an example scene (see pages 142–144)

6.3.2 Experiment Setup

Training A model with FoV awareness (see Sec. 6.2.3) is trained for state-of-the-art comparison, while other experiments utilize a model trained with a fixed 90° FoV. The model with variable FoV samples viewing angles between 45 and 135° during training.

The sampling radii for panoramas are set to $r_1 = 1.4\,\text{m}$ and $r_2 = 0.3\,\text{m}$, with $s_1 = 4$ positive and $s_2 = 1$ negative panorama samples (see Sec. 6.2.4). The panorama renderings are created using a ray-tracing-based renderer [LADL18].

Perspective query images are resampled from loaded panoramas with random yaw, pitch, and roll angles (see Fig. 6.4). Pitch and roll angles are sampled from an interval of $\pm\,10°$. The batch size is set to 40, comprising 40 query images and 200 panoramas. The training is performed on a single NVIDIA A100 GPU. Query images with fewer than three semantic classes are disregarded during the loss calculation process. Training runs for approximately 42,000 steps with an initial learning rate of 2.5×10^{-4}, halved at 47% and 72% of the training process. Data augmentation includes random color jitter (brightness, contrast, saturation) and Gaussian blur to improve robustness to environmental variations.

Testing During testing, the grid is defined by $h_{pano} = 1.4\,\text{m}$ and $xy_{pano} = 1.2\,\text{m}$, sampling on a $1.2 \times 1.2\,\text{m}$ grid. Compared to the rendering-based approach LaLaLoc [HJRSP21], which uses a $0.5 \times 0.5\,\text{m}$ grid, the number of required samples is significantly reduced. Unless otherwise noted, the studies are conducted on a ZInD test set with roll/pitch variation of 10° and a 90° FoV.

6.3.3 Comparison with the State of the Art

SPVLoc is the first method to achieve 6D localization in unseen scenes using semantic 3D models. The state-of-the-art method for scene-independent global localization, LASER [MKB+22], takes perspective views as input and matches them against a semantic floor plan. Hence, it estimates only two positional and one rotational degree of freedom Despite this, it serves as a robust baseline for assessing the localization accuracy of SPVLoc. LASER expects zero pitch/roll angles in its input, which is respected in this comparison.

To compare with a 2D method, recall is evaluated based on 2D metrics. The 2D translation error, denoted as $t_{\text{err}}^{\text{xy}}$, is the Euclidean distance between the ground truth and estimated positions in the xy-plane. The 2D rotation error extracts the yaw angle $\psi = \text{atan2}(r_{10}, r_{00})$ from the rotation matrix $\mathbf{R}$, and is denoted as:

$$r_{\text{err}}^{\text{yaw}} = |\text{atan2}\left(\sin(\Delta\psi), \cos(\Delta\psi)\right)| \cdot \frac{180}{\pi} \quad \text{with} \quad \Delta\psi = \psi_{\text{gt}} - \psi_{\text{est}}. \tag{6.5}$$

Table 6.1 presents the results on the ZInD and S3D datasets. Notably, SPVLoc (*Ours*) outperforms LASER in localization accuracy across all three FoV settings. It achieves higher recall across all categories, an increased inlier rate, and a significantly lower median translation error. For example, for 90° FoV images from ZinD, the median translation error of 13.63 cm is nearly half that of Laser, with over three times more images localized within 10 cm and recall within a 1 m, 30° range improving from 80% to 90%. Similarly, for 90° S3D images, more than twice as many are localized within 10 cm (over half of the images), and recall within a 1 m, 30° range improves from 76% to 92%.

In terms of 2D rotation, LASER performs slightly better due to its direct estimation of yaw, while SPVLoc estimates a full 3D rotation. However, its capability to estimate camera height, roll, and yaw angles offers a distinct advantage. To further assess its robustness, SPVLoc is evaluated on test sets with roll and pitch variations of ± 10° (*Ours*†). The results suggest that these variations do not substantially impact the accuracy of the method for 2D localization.

Regarding further methods, LASER has demonstrated a significant performance improvement over PF-net [KHL18] and non-AI-based Monte-Carlo Localization (MCL)

		ZInD							**Structured3D** (Furnishing-Level : Full)						
Query	Method	<1 m med $t_{\text{err}}^{\text{xy}}$ (cm)	<1 m med $r_{\text{err}}^{\text{yaw}}$ (deg)	10 cm (%)	50 cm (%)	1 m (%)	1 m & 30° (%)	top-3 1 m (%)	<1 m med $t_{\text{err}}^{\text{xy}}$ (cm)	<1 m med $r_{\text{err}}^{\text{yaw}}$ (deg)	10 cm (%)	50 cm (%)	1 m (%)	1 m & 30° (%)	top-3 1 m (%)
Persp. 60°	LASER	33.53	1.03	4.05	47.24	64.52	63.53	78.64	17.60	0.84	16.46	63.22	68.39	67.34	86.08
	Ours	**17.16**	1.50	**21.04**	**81.33**	**86.76**	**86.09**	**94.12**	**12.25**	1.21	**34.94**	**87.13**	**89.49**	**88.79**	**96.15**
	Ours†	17.77	1.63	19.59	81.13	87.10	86.43	94.86	13.31	1.24	30.82	84.68	88.18	87.48	95.80
Persp. 90°	LASER	26.28	0.77	8.69	67.01	80.90	80.25	89.73	14.60	0.67	23.47	72.77	76.80	76.18	91.94
	Ours	**13.63**	1.34	**30.57**	**85.59**	**90.84**	**90.34**	**96.36**	**9.17**	0.83	**50.09**	**89.67**	**92.12**	**91.86**	**97.37**
	Ours†	14.25	1.37	29.59	85.90	91.33	90.95	96.97	10.09	0.91	45.62	89.32	92.12	91.94	96.85
Persp. 120°	LASER	23.10	0.78	12.69	75.09	85.63	85.20	92.38	12.90	0.76	29.07	77.76	82.14	81.35	94.13
	Ours	**14.78**	1.45	**29.05**	**85.14**	**91.72**	**91.27**	**96.92**	**11.69**	0.87	**39.49**	**88.00**	**92.12**	**91.94**	**97.37**
	Ours†	15.13	1.53	28.12	86.13	92.71	92.35	97.24	11.77	0.88	38.18	89.67	93.08	92.91	97.55

Table 6.1: Comparison with state-of-the-art baseline LASER [MKB+22] on ZInD and S3D. Median translation ($t_{\text{err}}^{\text{xy}}$) in cm, rotation error ($r_{\text{err}}^{\text{yaw}}$) in deg. for instances localized under 1 m. Reporting recall at various translation accuracy levels, recall for inliers (<1 m and < 30°), and top-3 recall at 1 m. *Ours*† refers to the model tested with a test set incorporating pitch and roll rotation variation (± 10°).

[DFBT99]. PF-net, originally developed for sequential time updates (such as in particle filters), struggles in single-image localization tasks. On the other hand, MCL, which relies on LIDAR data rather than images, neglects semantic annotations, which has been shown to substantially reduce performance [MKB+22]. The improvement of SPVLoc over LASER, therefore, also reflects a performance gain over these methods.

The slight differences compared to LASER's reported results arise from the use of a custom reproducible test set. LASER's original test samples random yaw angles, occasionally generating images of empty walls, hindering pose estimation. In contrast, the test set used here contains images with richer contextual information, leading to more inliers and slightly higher median translation errors.

6.3.4 Ablation Study

Starting from the full network (**base**) configuration, specific components are removed to evaluate their effect on 6D localization (Tab. 6.2). The first modification tests the impact of excluding perspective supervision (**-query dec.**) and the view segment task head (**-view dec.**) during training. Their exclusion reduces network performance, likely because the ability of the network to understand the image content and establish correspondences is diminished. Removing both (**-view/query dec.**) further reduces accuracy.

Next, negative samples from different rooms are excluded during training (**-neg. samples**), significantly reducing the number of correctly localized images. This supports the hypothesis that teaching the network to recognize areas where there is no match enhances prediction accuracy.

To alter the architecture, the image encoder *EfficientNet-S* is replaced with a smaller *ResNet-18* [HZRS16] (**Δ Resnet18**), resulting in a performance drop. This suggests an increased demand for network capacity to analyze the images. In the panorama encoder, replacing all convolutional layers with *Equiconv* [FLFPY+20] (**Δ Equiconv**), a

Abl. Type		Method	<1 m med t_{err} (cm)	<1m med r_{err} (deg)	10 cm (%)	50 cm (%)	1 m (%)	1 m & 30° (%)	top-3 1 m (%)
(Ours)		base	**14.31**	**2.05**	**26.88**	86.92	**91.90**	**91.61**	96.56
	+	refine	12.32	1.96	33.69	90.84	92.76	91.74	-
Loss	-	query dec.	15.70	2.23	21.97	85.50	90.79	90.43	96.45
	-	view dec.	15.19	2.21	23.60	85.36	90.45	89.82	96.49
	-	view/query dec.	16.22	2.19	21.56	84.50	89.59	89.12	96.36
Training	-	neg. samples	15.06	2.28	22.60	76.65	82.19	81.74	92.99
Arch.	Δ	Resnet18 (query)	15.56	2.56	20.88	86.56	90.72	90.32	96.56
	Δ	Equiconv (pano)	14.98	2.13	25.07	**87.44**	91.70	91.31	**96.97**
Input	+	normal	14.68	2.28	25.09	85.93	90.84	90.23	96.47
	+	depth	**14.06**	2.10	**27.60**	**88.05**	**92.15**	**91.79**	96.74

Table 6.2: Ablation study. SPVLoc variations compared on ZInD with 90° FoV.

specialized convolution for panorama images that remaps query locations from Euclidean to spherical space, does not result in a significant performance gain. This suggests that the capacity of the small *DenseNet-121* is sufficient for analyzing the low-resolution semantic inputs.

Additionally, extended inputs for the panorama encoder are evaluated by augmenting the rendered panoramas with additional modalities. While the addition of world coordinate normals (**+normals**) did not improve the result, including an extra depth map slightly enhanced the outcome (**+depth**). This suggests that there are cases where additional information is encoded in depth that cannot be inferred from semantics alone.

Lastly, refining the result of **base** (**+refine**) by rendering a new panorama at the top-1 position and re-estimating the pose significantly improves recall rates and reduces pose errors. This will be discussed in detail in a separate experiment in Sec. 6.3.6.

6.3.5 Qualitative Results

Selected outcomes of SPVLoc are displayed in Fig. 6.5. The upper row shows the camera images, while the row below displays renderings from the semantic 3D model with the estimated top-1 pose. The third row shows the top-1 reference panorama, the predicted viewport bounding box, and a semi-transparent overlay representing the estimated viewport mask. All green-boxed images are correctly localized for the top-1 match within a 1 m range. In the yellow box, the top-1 match fails, but the top-2 match is accurate. The red box example cannot be localized. Across all examples, a pose with high visual similarity is found. Ambiguous examples arise from similar layouts occurring in multiple rooms. Failure cases are attributed either to images lacking sufficient structural information for matching, or to outliers in the data, such as those taken on balconies or in dark cellars. *More examples are provided in Appendix A.5.*

SPVLoc evaluates the correspondence between a reference panorama and an input image by calculating a classification score for each reference. The method employs anchor-based bounding box estimation [LGG$^+$17], where each anchor is assigned a score representing its degree of correspondence with the input image. The reference panorama score is the maximum of all anchors.

In Fig. 6.6, reference positions are color-coded based on their normalized classification score, ranging between 0 and 1, and displayed in a gradient from red to green. The ground truth camera direction is shown with an arc pattern originating from the ground truth position. The estimated top-3 locations are displayed by smaller circles.

For both examples, three reference panoramas with high classification scores are shown. The exact pixel-based viewport estimated by the *Segment* head is shown as a semi-transparent white overlay. It can be observed that the reference panoramas achieving high scores capture the content of the input image to a large degree. For example, in Example 1, high scores are achieved for positions where the bay window, as well as the wall opening in the direction of the kitchen, can be seen in the panorama.

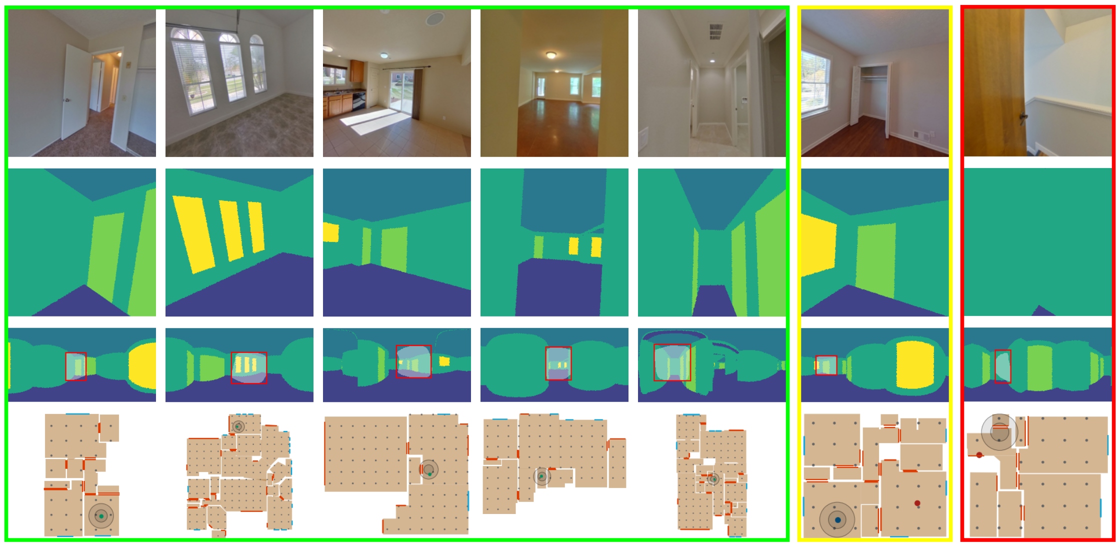


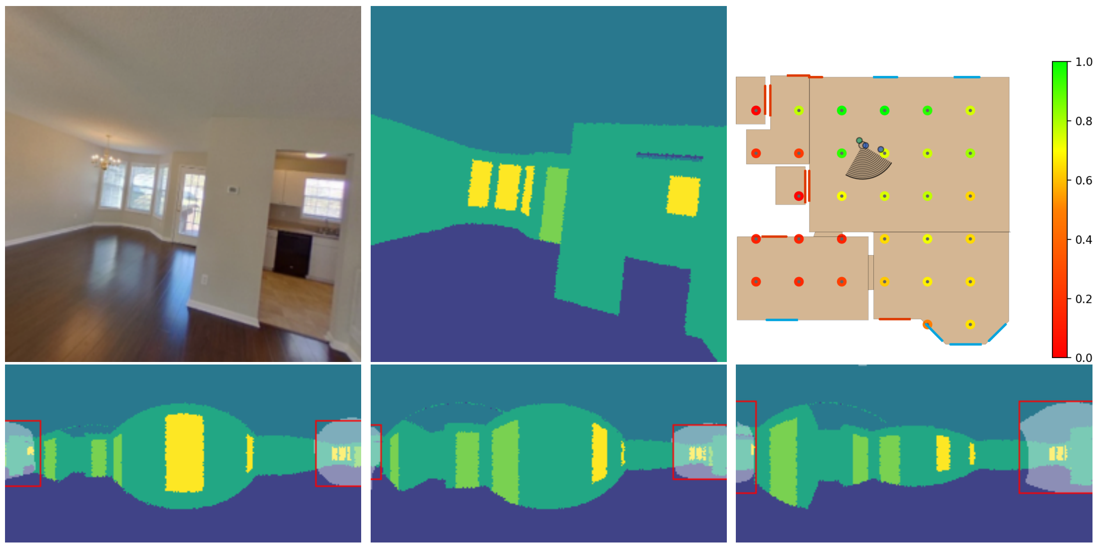
Figure 6.5: Qualitative localization results. Top to bottom – query, rendering with top-1 estimated pose, panorama with estimated viewport, map. Green box: success for top-1 match. Yellow box: success for top-2 match. Red box: failure case.

The selected reference panoramas show that the viewport of the camera image can be estimated from panorama positions with large in-between distances. Looking at the viewport in Example 2 (bottom row), the door and the adjoining wall are correctly detected by the viewport mask from different reference positions, despite perspective distortion.

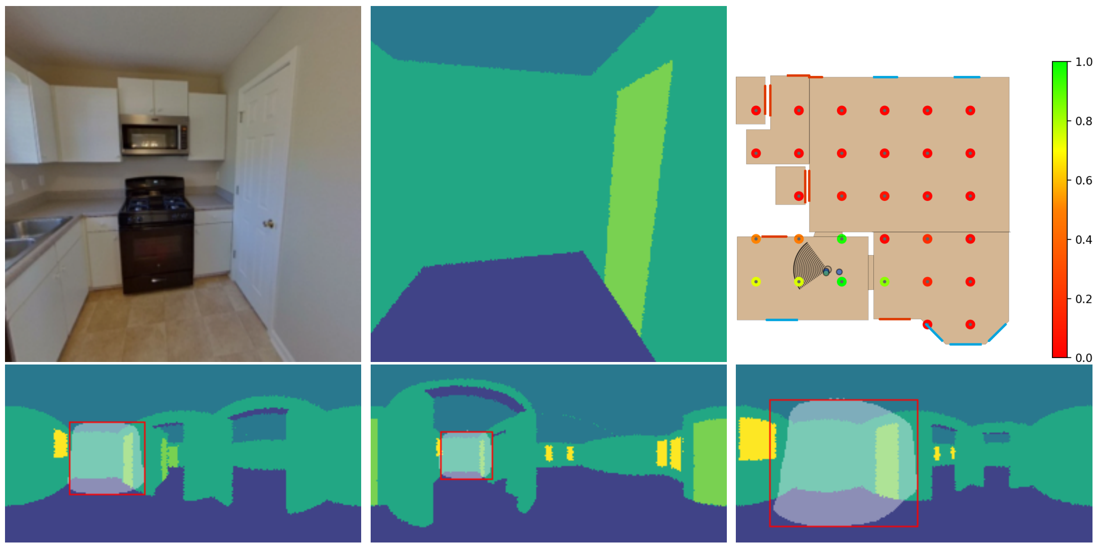
In conclusion, the depthwise correlation used for the matching demonstrates its ability to incorporate finely granulated details from the query encoder into the main network branch. It effectively distinguishes subtle distinctions in target images, maps from the photo to the semantic domain, and, in conjunction with the panoramas, successfully induces the correct 6D pose offset.

6.3.6 Influence of Sampling Grid Sizes and Refinement

Reference positions result from overlaying a grid onto the floorplan, and the grid density significantly affects SPVLoc's efficiency and inference time. Minimizing the number of references reduces rendering effort and requires the model to estimate larger offsets. For example, the increase of the grid spacing 0.7 m to 2 m reduces the number of samples per m^2 by approximately a factor of 10. A local grid per room (*Adv.*) is explored as an alternative to the global grid, which reduces the risk of entirely missing rooms. The local grids are centered on the room outline rather than the entire plan outline. Additionally, one camera is placed in the middle of each opening. Figure 6.7 shows different grids for an example plan.



(a) Example 1



(b) Example 2

Figure 6.6: Viewport score overview. For each example, the top row displays the input image on the left, the model rendered from the estimated top-1 pose in the middle, and a map where each reference position is color-coded relative to the estimated matching score. Here, the ground truth pose is shown with a stripe pattern, and the estimated top-3 positions are represented as dots (green for top-1). The second row shows three panorama images with high matching scores for the red bounding box and the pixel-based viewport displayed as a semi-transparent white overlay.

Sample dist.	Samples per m^2	Adv.	10 cm (%)	10 cm + refine	1 m (%)	1 m + refine
0.7	2.07		+1.00	+7.49	+1.86	+2.38
0.9	1.25		+0.95	+6.99	+1.79	+2.08
1.2	**0.70**		$\boxed{\mathbf{26.88}}$	+6.81	$\boxed{\mathbf{91.90}}$	+0.86
1.2	0.48	✓	+0.57	+6.95	−0.63	+0.61
1.5	0.45		−1.00	+6.63	−2.47	−1.02
1.5	0.31	✓	−1.04	+6.47	−2.67	−0.38
2.0	0.26		−4.19	+3.60	−9.05	−5.79
2.0	0.20	✓	−1.76	+5.81	−3.91	−1.18

Table 6.3: Grid size vs. accuracy. Deviation of 10 cm and 1 m recall from (boxed) baseline. The samples per m^2 are computed using the ZiND test scenes. *Adv.* indicates the use of local room-centered grids rather than a global floorplan-centered grid.

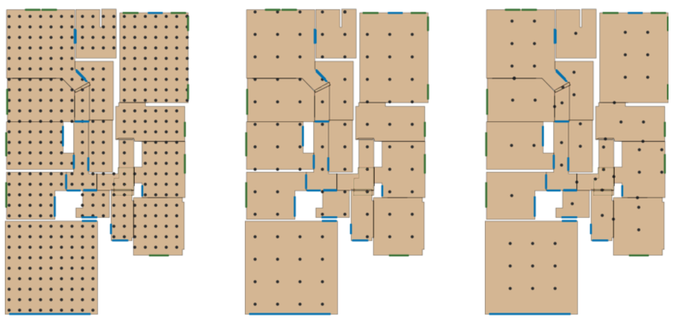


Figure 6.7: Reference positions for different grid sizes and types. Left: global grid (0.7 m), Middle: global grid (1.5 m), Right: local grid (1.5 m).

To analyze the impact of grid size on performance, various grid configurations are tested, Tab. 6.3 illustrates the relationships among grid size, global and local sampling, and refinement. The optional refinement step notably enhances the 10 cm recall across all configurations. For the largest local grid with refinement, the 1 m recall only decreases by 1.18% compared to the baseline configuration. Refinement is particularly useful when the distance between reference panorama positions is large, compensating for the cost of rendering an additional panorama during refinement. In summary, there exists a trade-off between refinement and dense versus sparse scene sampling; however, the method consistently delivers accurate results even with very sparse sampling, making it well-suited for large scenes.

Figure 6.8 presents two examples where the proposed refinement enhances the estimated pose. The zoom in the map shows that the estimated pose is closer to the ground truth than the initial estimate. The relative pose offset between the selected reference panorama

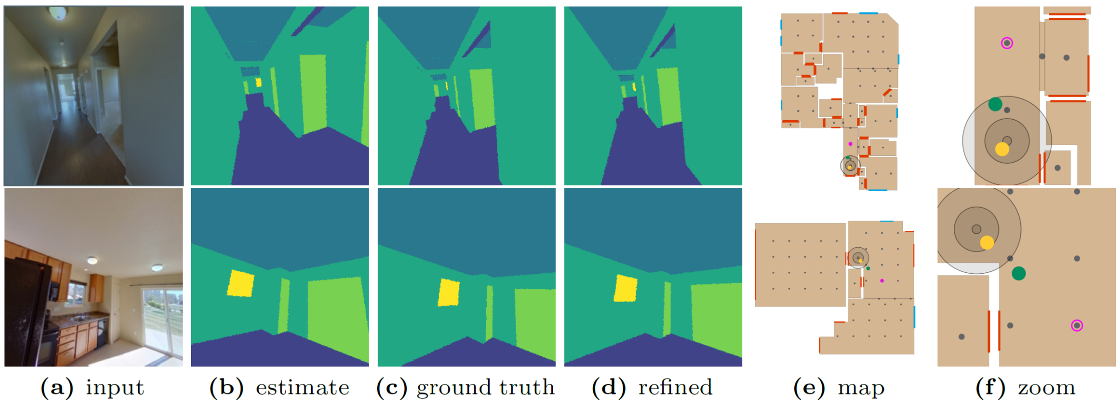


Figure 6.8: Demonstration of pose improvement via an iterative refinement step. On the map, the ground truth position is marked by gray circles with radii of 10 cm, 50 cm, and 100 cm. The estimated position is represented by a green dot, the refined position by a yellow dot, and the position of the selected reference panorama by a pink dot. The grid size is 150 cm, and a local grid (*Adv.*) is used.

position (pink) and the estimated initial position (green) is substantially larger than the distance between the initial (green) and refined positions (yellow).

During refinement, the query is matched against a single panorama only, leading to quick network inference. Beyond the scope of this study, the method could also be adapted for local tracking with a moving camera, where the panorama position is consistently located at the estimated position of the previous frame.

6.3.7 Influence of Focal Length

In this experiment, the performance difference between a network trained specifically for a known camera focal length and a focal length-independent network trained with random FoV, ranging from 45° to 135°, is examined. Table 6.4 compares the results of both networks exposed to images with 80° and 90° FoV.

The fixed focal length network performs slightly better when matching images captured with the FoV it is trained for, but drastically loses precision when tested with images of a different opening angle. In contrast, the focal length-independent network demonstrates the capability to handle FoV variations with constant accuracy. The significantly enhanced flexibility of the latter justifies the minor performance degradation in practical, real-world scenarios. In both cases, knowledge of basic sensor and lens properties is necessary, as the camera's viewing angle must be known either during training or testing.

Figure 6.9 shows an example of SPVLoc correctly estimating the pose of input images with different opening angles. Examining the estimated positions on the map reveals that all three images are correctly placed near the ground truth position, which remains

Input FoV	Train FoV	<1 m med t_{err} (cm)	<1 m med r_{err} (deg)	10 cm (%)	1 m (%)
90°	90°	**14.31**	**2.05**	**26.88**	**91.90**
	45 - 135°	16.11	2.20	21.81	91.45
80°	90°	37.51	2.37	1.67	90.14
	45 - 135°	**16.75**	**2.19**	**19.30**	**90.50**

Table 6.4: Comparison of fixed FoV and FoV-independent variants of SPVLoc. The fixed FoV network excels at a specific opening angle but loses precision as the FoV changes, while the FoV-independent network remains stable.

	Roll/Pitch Variation				
Metric	± 0°	± 5°	± 10°	± 15°	± 20°
<1m med. terr	14.46	14.88	14.88	15.21	15.91
<1m med. rerr	2.35	2.36	2.38	2.51	2.60
10cm (%)	24.64	23.94	24.46	23.69	22.74
1m (%)	91.45	91.47	91.99	92.38	92.60

Table 6.5: Effect of varying rotation angle ranges in test data. Performance degrades only slightly if the angles remain within the range seen during training (± 22°).

consistent across all three instances. Additionally, it is evident that the view rendered from the top-1 camera pose closely matches the input image.

The third column presents a comparison of the viewport estimated with the *Segment* head of the network (top, white transparent overlay) and the ground truth viewport (bottom), generated via perspective projection. By comparing the estimated viewport, the ground truth viewport, and the input image, it becomes apparent that the network successfully learns perspective projection and establishes visual correspondences between the input and the panorama.

6.3.8 Influence of Camera Rotation Angles

The range of pitch and roll angles the network is able to handle is determined by the variations of angles seen during training. In this experiment, the network is trained on ZInD with an increased angle variation of ± 22° and tested with images where the range of variation increases from 0 to 20° in steps of ± 5°. Table 6.5 shows only a slight increase in median errors for rotation and translation as the rotation angles of the test images increase. This demonstrates robust estimation, even for larger roll and pitch variations. Assuming a further increase in pitch, there will be a point where images predominantly capture the ceiling or floor, making localization impossible.

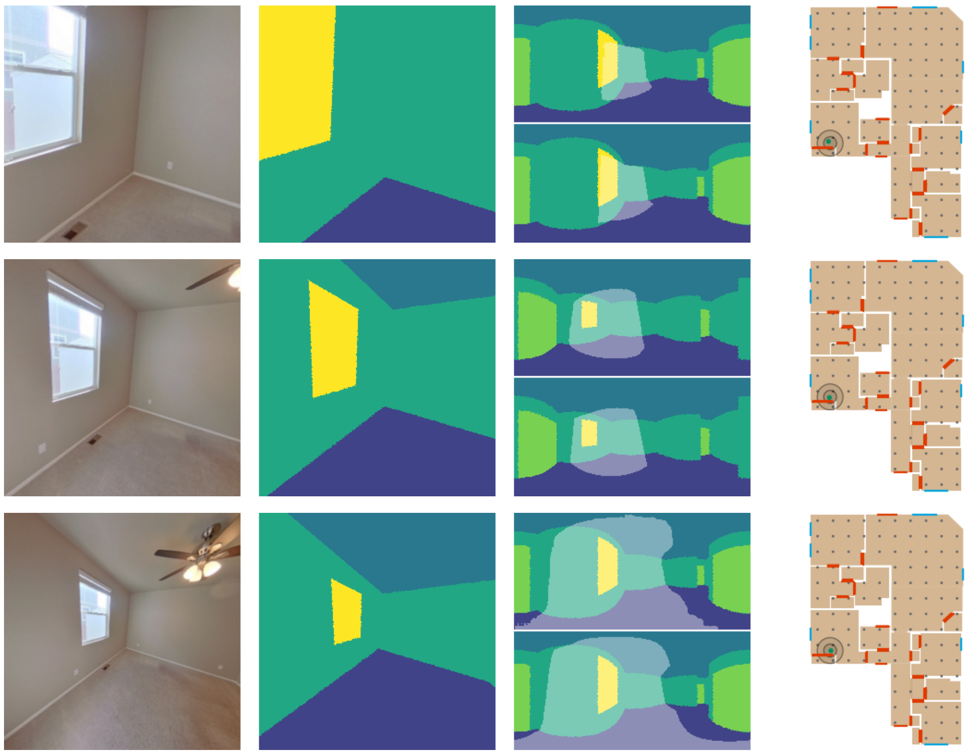


Figure 6.9: Localization results for images with varying fields of view (60°, 90°, 120° from top to bottom) captured from the same position: Input image in the first column, rendering with top-1 pose in the second column, comparison of estimated (top, white transparent overlay) and ground truth viewport mask (bottom) in the third column, the map in the fourth column. The ground truth position is marked by gray circles (50 cm and 100 cm radii), and the estimated position by a green dot.

6.3.9 Refinement for a different Aspect Ratio

Besides panoramic renderings, S3D [ZZL$^+$20] also comprises high-resolution perspective images in a 16:9 format with an 80° horizontal FoV. In contrast to resampling perspective training images from the 21,835 panoramas, this experiment demonstrates the use of perspective images contained in S3D in both training and testing.

Table 6.6 lists the results for the test sets, including 6405 photorealistically rendered perspective images from S3D test scenes, both with and without furniture. The pre-trained network, as in Tab. 6.1, is refined for 10 additional epochs with the S3D perspective images. During training, both *empty* and *furnished* images were used randomly. The images are zero-padded above and below to fit the square input of the network. With over 85% (6D, average of empty and furnished) of the images correctly localized within a 1 m threshold as the top-1 match and over 94% as the top-3 match, the network demonstrates its

Query	Aspect ratio	Furniture	Localization mode	<1m med t_{err} (cm)	<1 m med r_{err} (deg)	10 cm (%)	50 cm (%)	1 m (%)	1 m & 30° (%)	top-3 1 m (%)
Perspective 80° FoV	16:9	empty	2D	11.70	1.14	35.99	84.06	86.73	86.25	94.79
			6D	13.09	1.50	28.77	83.97	86.73	86.21	94.79
		furnished	2D	12.86	1.26	30.55	81.58	84.50	84.20	93.36
			6D	14.30	1.62	23.54	81.48	84.50	84.18	93.36

Table 6.6: Evaluation of SPVLoc on perspective images from S3D [ZZL+20]: A comparison is made between **furnished** and **unfurnished** configurations. For 2D localization, the listed results indicate the 2D localization errors ($t_{\text{err}}^{\text{xy}}$ in cm, $r_{\text{err}}^{\text{yaw}}$ in degrees), while for the 6D case, the 3D metrics (t_{err} in cm, r_{err} in degrees) are shown.

Method	Uses semantics	Reference model	10 cm (%)	50 cm (%)	1 m (%)	1 m & 30° (%)
F^{3}Loc [CWVP24]	no	2D	1.50	14.60	22.40	21.30
Ours	yes	3D	**30.55**	**81.58**	**84.50**	**84.20**

Table 6.7: Comparison of SPVLoc to the single-frame variant of F^{3}Loc using *furnished* perspective images from S3D. Recall is reported at different accuracy levels for 2D localization. Results for F^{3}Loc are taken from [CWVP24].

capability to handle the 16:9 test images efficiently. More than 26% of the images are localized correctly within a 10 cm threshold.

The strategy of zero-padding to a square shape could also be applied to train a network that operates independently of the image aspect ratio. By randomly adding symmetrical black bars of varying widths to the edges of the image during training, one can simulate inputs with different aspect ratios. To adapt the FoV scaling for a single network that handles both portrait and landscape images, one approach could be to select the larger of the two view angles (horizontal or vertical) as the network input.

To date, the only other work that provides localization results for the perspective images of S3D is F^{3}Loc [CWVP24]. This method performs 2D localization and relies solely on a 2D floor plan occupancy without semantic information. Table 6.7 compares the proposed method with the single-frame variant of F^{3}Loc. Although omitting semantic and height information may be unavoidable in certain situations (e.g., if not annotated in the floor plan), it significantly reduces the recall rate. The comparison demonstrates how beneficial using this information is for localization, with nearly four times higher 1 m & 30° recall and even greater improvements for 50 cm and 10 cm recall.

6.3.10 Robustness against Furniture

S3D contains renderings of rooms in *empty* and *furnished* configuration. Table 6.6 compares the estimation results for *empty* and *furnished* test images, but otherwise equal camera poses. The results indicate that the trained network is able to learn to ignore

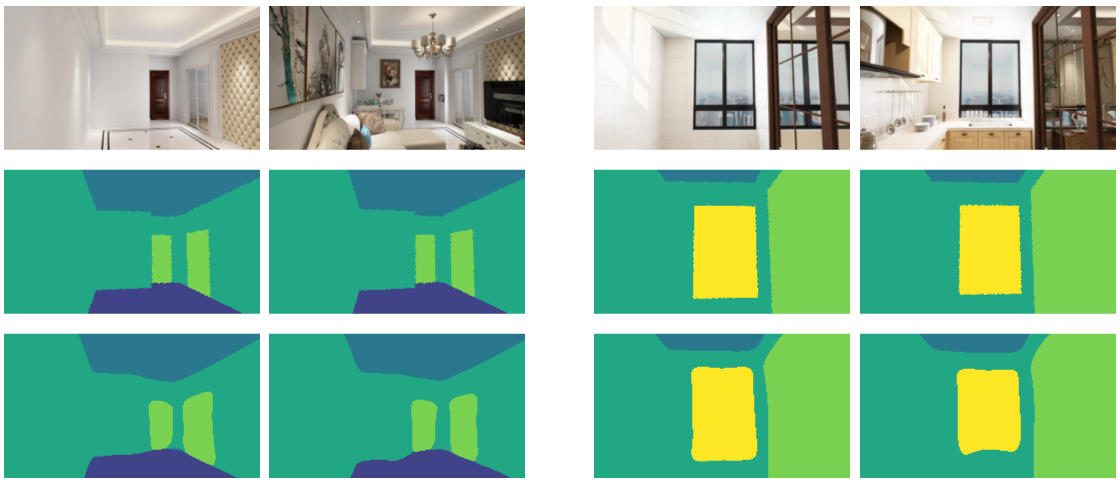
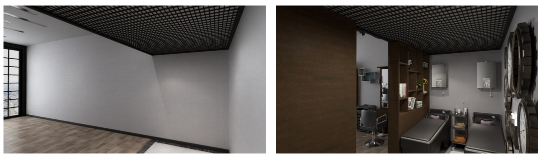
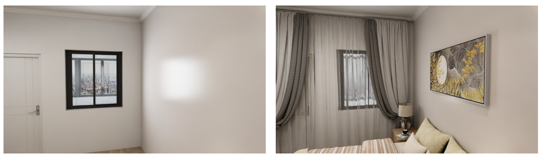


Figure 6.10: S3D examples with accurate localization for both *empty* and *furnished* images ($t_{\text{err}} < 25\,\text{cm}$, $r_{\text{err}} < 2°$). Input image in the first row, top-1 pose rendering in the second row, semantic *Query Decoder* output in the third row.



(a) additional wall modeled as furniture



(b) door behind curtain

Figure 6.11: Failure cases due to occluded room structure. In both examples from S3D, the *empty* image is localized accurately ($t_{\text{err}} < 25\,\text{cm}$, $r_{\text{err}} < 2°$), but localization fails for the *furnished* image due to important room structures being covered.

furniture in the images efficiently. This can also be seen by the output of the *Query Decoder*, which is displayed in the bottom row of Fig. 6.10. The decoding of image semantics is similar for the *empty* and *furnished* image and, in both cases, similar to the image rendered with ground truth pose. The slight decrease of around 2% in the 1 m recall can mainly be attributed to furniture that obscures the visible room structure and makes matching difficult, as shown in Fig. 6.11. For example, the information whether there is a door behind a shelf cannot be derived from an image when the door is invisible. The decrease of around 5% in the 10 cm recall can be explained, for instance, by wardrobes covering room edges, requiring the network to guess the invisible room structure.

6.3.11 Timing

In this evaluation, the measured time is divided into scene sampling/preparation time, which includes the sampling, rendering, and encoding of the panorama images, and inference/query time, which encompasses the encoding of the query image, feature correlation (including bounding box classification) and the *Pose* task head. Table 6.8 shows that SPVLoc exhibits considerably lower query time than LASER [MKB+22] and PF-net [KHL18], bringing the approach close to real-time performance for moderately sized scenes. Currently, the scene sampling time is slower than [MKB+22], mainly due to rendering on the CPU without parallelization [LADL18]. Parallelizing this process using GPU or CPU could significantly accelerate scene preparation. Once a scene is prepared, new images can be matched directly against the precomputed encoded panoramas.

Method	Sampling Time (s)	Query Fps
PF-net [KHL18]	48.95 ± 38.95	5.06 ± 1.77
LASER [MKB+22]	**0.97** ± 1.09	8.31 ± 0.64
Ours	1.66 ± 1.06	**28.63** ± 11.54

Query Module	Time (ms)
Eff.Net V2 S	4.4
Correlation	35.7
Pose	1.1

Table 6.8: Timing. Performance on ZInD using an NVIDIA Tesla V100, following the configurations outlined in Sec. 6.3.4, and compared against results from [MKB+22].

6.3.12 Limitations

In large spaces with repeating room layouts, the effectiveness of the approach may be constrained by the limited level of detail in the semantic reference model. Nevertheless, there is an opportunity to enhance the S3D format by incorporating additional semantic classes and more structural information. This could involve the inclusion of features such as staircases, galleries, or even more specific semantic details like sanitary installations or radiators. Importantly, input images need to depict sufficient room structure for effective matching. For example, extreme pitch angles often result in images dominated by the ceiling or floor, which lack distinct structural features and are therefore excluded from training and testing.

Generally, the system's performance depends on the range of poses seen during training, even though the log quaternion can theoretically represent all rotations and the translation offset can cover a broad 3D space. While roll angles larger than 20° are currently excluded from the experiments, they could be incorporated in future work, as rotation around the camera axis typically has minimal impact on image content. For camera height estimation, a relative offset is predicted. By training with virtual cameras across a broad height range, the system learns to output large height offsets. However, accurately localizing images with a camera positioned near the floor or ceiling is unlikely if the training data lacks real-world images captured at these heights (see Appendix A.4).

While ZiND generally annotates ceiling height, it is not always modeled correctly due to inaccuracies and limitations of the data model. For example, sloping roofs are not modeled in the S3D format, nor are they annotated in ZiND. During training, the network sees images where mismatches occur and learns to deal with those imperfections, e.g., by focusing on other visible elements like windows and doors. If no height information is available at inference time, approximately defined standard heights can be used as an alternative, although a slight loss of accuracy must be expected.

6.4 Chapter Summary and Outlook

This chapter has introduced a method for scene-independent model-based 6D localization for indoor scenes, which includes a novel method for multimodal image matching, combining panorama-to-perspective and RGB-to-semantic matching.

Learning relative pose estimation using a semantic 3D model offers a significant advantage over matching with localized image databases: it enables the model to focus on relevant information defined by the reference model while effectively disregarding non-modeled elements. This advantage was demonstrated in experiments with both furnished and unfurnished scenes using the S3D dataset.

The novel approach of representing 3D scenes using a set of semantic panoramas enables spaces to be captured as image representations that can be processed by a CNN while maximizing the amount of information per reference image. Experiments have demonstrated that perspective-to-panoramic matching can bridge wide baselines, allowing for sparse grids of references. However, the maximum allowed distances are constrained by the relative distances encountered during training. Future work could explore the potential for extending the camera baselines further. Related research has optimized the placement of panoramic cameras in indoor environments using linear integer programming [SP22]. Integrating this idea could minimize the number of panoramas required while ensuring that each region is represented in at least one image.

The matching of resampled perspective views from panoramas with rendered semantic panoramas is efficient for learning visual correspondences. Since both inputs can be generated from random parameters sampled on the fly, the network is continuously exposed to new combinations. The re-sampling technique simulates various camera parameters, such as the field of view, during image creation with precisely known ground truth. This enables training a model capable of handling different camera intrinsics by presenting sufficient variation, with only a simple network modification required. In other recent work, this principle is even used to train networks for single image camera calibration [VSLP24].

The calculation of the viewport revisits the concept of the projector as an inverse camera from Chapter 3. The camera *projects* its field of view onto the given geometry, much like a projector projects an image. This field of view can only be predicted by a neural network if it understands both the geometric structure of the image and that of the scene. Feature-correlation-based matching has proven efficient in establishing perspective relationships between inputs, while the use of anchor-based viewport bounding boxes provides a robust method for comparing matches across multiple references.

The localization accuracy and inference speed outperform state-of-the-art methods, and the inclusion of the 3D model reduces ambiguities compared to semantically annotated 2D maps. The additional information required is minimal: one metric height value for each semantic element, except for windows, which require two values (one for the window height and one for the sill height). These lightweight extra annotations also allow the network to estimate precise 6D poses, which can be helpful in future AR applications.

SPVLoc includes an optional refinement step in which a new panorama is rendered at the estimated position, and the position is re-estimated relative to this panorama. The experiments have shown that this improves the accuracy of the estimates, and tracking could also be implemented by estimating the pose of a new image against the previously rendered panorama. However, it would be interesting to model uncertainties across the entire plan to better handle ambiguous images within video sequences. This could be achieved, for example, using a particle filter [VDMDDFW00, KHL18] or a histogram filter [JB16, CWVP24]. However, an extension to probabilistic tracking requires a dataset encompassing both semantic models and sequential image data. Currently, no such dataset exists for real image data [CWVP24]. One possibility is to generate trajectories within 3D reconstructions, such as those found in the iGibson dataset [SXL+21]. Alternatively, modern rendering techniques like Gaussian Splatting [KKLD23, BHG+24] could be employed to create 3D scenes from the panorama images in the ZinD dataset.

Future work may focus on cases where additional information is present in the map, such as room labels and more detailed local information, e.g., the locations of stairs, electrical installations, and fire safety equipment. To account for this, these new classes could be directly incorporated into the training, although this would require a very labor-intensive expansion of the dataset. Foundation models like Grounding DINO [LZR+23] are trained on large, domain-crossing datasets and can capture semantic relationships in images without further training. Integrating foundation model features into the matching process is promising. This approach could involve allowing users to manually annotate the floor plan with an open set of classes and using their spatial activation in the image during perspective-to-panoramic matching. As a result, ambiguities in 3D models could be resolved using localized text prompts.

7 Selected Applications

While creating this thesis, I had the opportunity to explore both fundamental research and its translation into practical applications. Collaborative projects provided platforms to apply the developed pose estimation algorithms to address real-world challenges. Integrating these methods into various demonstrators offered invaluable insights, especially regarding their effective and efficient application, appropriate visualization, and connection with other AI systems. The following sections present the use cases, solutions, and conclusions, demonstrating the broad applicability of these approaches.

The methods are encapsulated in a configurable C++-based **AR toolbox**, developed for this thesis. This software facilitates the deployment of the methods in live demonstrators. It supports the configuration of AR scenes, including 3D visualizations overlaying objects, multi-object tracking, pose estimation, and projector-based tracking. The toolbox interfaces with cameras and communicates in real-time with other modules through WebSocket. Additionally, it projects content via a projector, applying necessary distortions to ensure accurate alignment with real-world geometry. With its versatile communication interfaces, the toolkit functions as both a data input (sensor) and output (endpoint) within the demonstrators. For example, hand interactions from external hand-tracking systems can be seamlessly integrated into the virtual scene.

This software is utilized in the demonstrators discussed in Sections 7.1 and 7.2, and it can be expanded to include the global localization discussed in Sec. 7.3. Since the global localization is implemented as a stand-alone module with a WebSocket interface, the AR toolbox can access it directly.

7.1 Visual Weld Inspection

As part of the *EASY COHMO* research project, an SAR system was developed to improve ergonomics and comfort during the visual inspection of large objects using human-robot collaboration. It enables simple and fast spatial documentation of inspection results, while providing intuitive control over both the robot and the inspection process [CSP+23].

The specific target scenario involves visually inspecting welds of a subframe, a heavy structural component of an automobile. This task is typically performed by a human expert who carefully examines the welds for defects such as cracks, porosity, or blowholes [MPG+22]. The visual inspection of weld seams is a demanding procedure that requires special training, attention, and considerable physical effort [KK07]. In the previous setup the workpiece is fixed in a rotatable frame, in which the position cannot be adjusted to

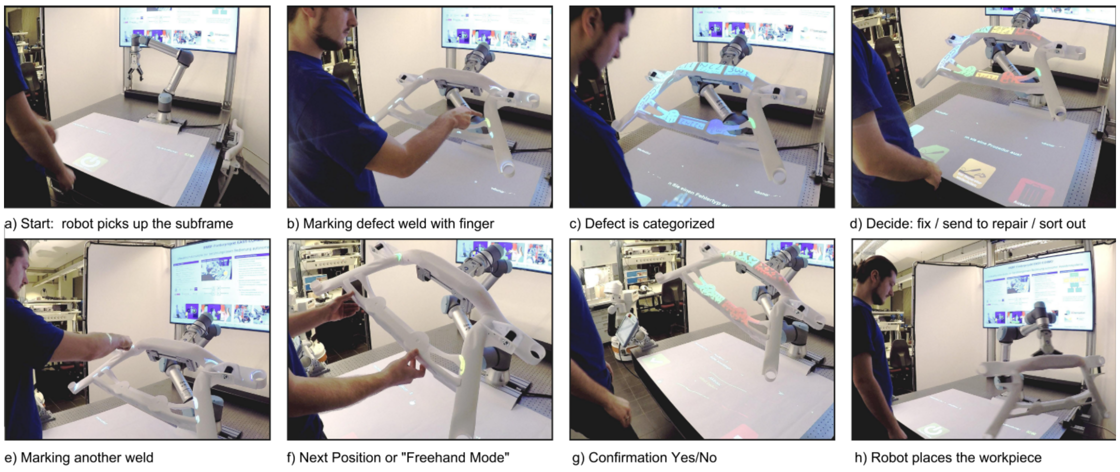

a) Start: robot picks up the subframe
b) Marking defect weld with finger
c) Defect is categorized
d) Decide: fix / send to repair / sort out
e) Marking another weld
f) Next Position or "Freehand Mode"
g) Confirmation Yes/No
h) Robot places the workpiece

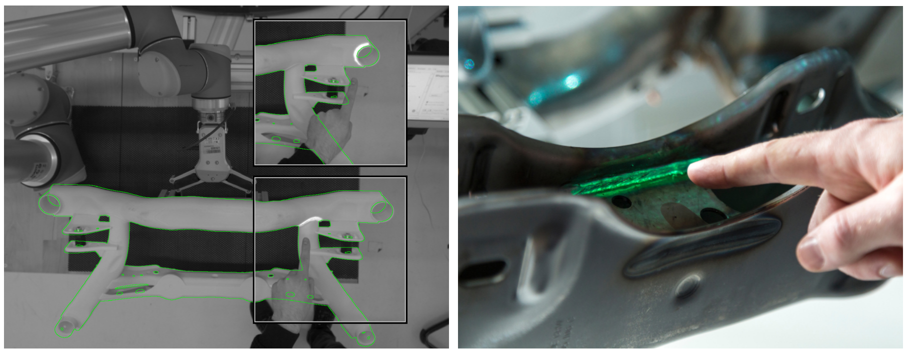
Figure 7.1: Inspection process of a car subframe using a mockup. The process involves a projected interface that moves with the object, enabled by real-time object tracking. Interaction is carried out through hand or voice commands. (Figure from [CSP+23])

user's preferences [CSP+23]. Also, there is no standardized documentation of the process. The old practice of measuring a weld seam with a measuring device, writing the results on a sheet of paper, transferring them to a computer database, and then printing a report is labor-intensive, error-prone, and slow [NHB11].

To simplify, accelerate, and digitize the task, a sensor-monitored workspace is created, where a robot arm presents the workpiece to the user, if needed, in various orientations. The user interacts directly with the workpiece through hand gestures and vocal inputs, while visual feedback is provided via precise projection mapping of information on both the 6D-tracked workpiece and the worktable. The chosen multimodal interaction approach for robot control, error detection, marking, and documentation follows established guidelines [RLL+04]. After finishing the inspection process, the inspection protocol is automatically available in digital form. The developed workflow is shown in Fig. 7.1.

In addition to the module discussed below, the setup includes software components for detecting head orientation, face position, hand gestures, hand position, and user identity, as well as for robot control, all connected via middleware [CSP+23].

The projection and registration module is implemented using technologies from this thesis. It visually links the work environment, consisting of the workpiece, a worktable, and the robot system, to the user. Utilizing shader lamps, the environment is visually altered by projecting newly rendered textures from the projector's viewpoint [RWLB01]. The projector and the camera form an intrinsically and extrinsically calibrated unit. The contour-based pose refinement and tracking (Sec. 3.2.4) is necessary to achieve accurately fitting augmentations even with fluctuating grasp positions. To optimize accuracy of the



(a) Contour alignment **(b)** Projection alignment

Figure 7.2: Weld selection process during visual inspection. In the camera image (a), the contour of the 3D model (green line) accurately matches the contour of the object. This allows for precise illumination of specific regions of the object (b). (Figure from [CSP+23])

projection mapping, the proposed extrinsic refinement (Sec. 3.3.4) initially optimizes the parameters of the projector-camera system, assuming a precisely known pose of a 3D object. Figure 7.2a exemplifies the registration between the 3D object and a 2D image, along with the alignment accuracy of the projection. Similarly, Fig. 7.2b demonstrates the visualization of a highlighted weld seam on the original metal workpiece. The system integrates three types of spatial augmentation.

1. **Augmentation of welds** Each weld is modeled as a separate texture layer using the texture coordinates of the 3D model (Fig. 7.3, right). Each layer is a binary image containing the exact path of the weld. External commands control the projected color. The module transforms incoming hand positions into object coordinates and triggers interactions with the weld if the hand is within a spherical area surrounding it.
2. **User interface on workpiece** An interface is projected onto the workpiece surface to enhance the interaction flow. Depending on the number of buttons needed, the 3D model is sliced vertically. For each slice, an axis-aligned bounding box represents the interaction region (Fig. 7.3, left). Hand interaction is handled as described above. All texture pixels in that box are found by projecting back to the texture space. The texture layer is filled with a pictograph of random size and orientation, ensuring readability from different angles and distances.
3. **User interface on worktable** A web-based interface is projected on the worktable. The projection is pre-distorted to create a rectangular image on the surface. The projection-plane refinement (Sec. 3.3.1) determines the exact position of the interface in space. The web-interface is rendered into the texture of a flat 3D object

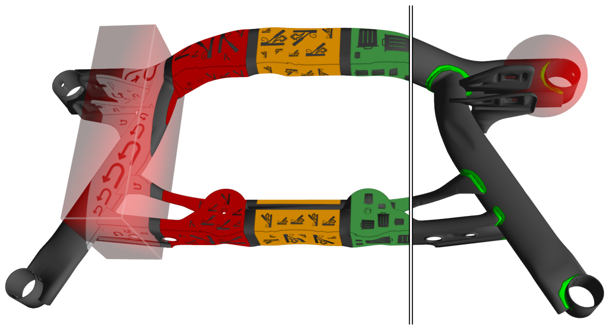


Figure 7.3: Projective annotations and interaction areas. Weld annotations (right) and menu items (left) are modeled as texture overlays. Hand interaction areas, exemplified by the red sphere and box, are geometric primitives aligned with the 3D mesh. (Figure from [CSP+23])

and updated at high frequency. The perpendicular projection of the hand's center onto the surface is converted into a 2D cursor with a metric height.

The system was evaluated for its suitability in the inspection task, focusing on the performance of different interaction modalities within a collaborative paper [CSP+23]. The evaluation includes three unimodal modes: *touch on worktable*, *touch on workpiece*, and *voice control*, as well as a *multimodal* mode that combines any of the unimodal modes. Weld selection is performed via touch interaction. 22 non-expert participants inspected 30 welds using all four modes. Constant object tracking is required because the workpiece height is adjusted for each participant, and touching the object, mounted on a rotating frame, could shift it. The SAR-based interaction proved to be faster than voice interaction, while worktable interaction was quicker than workpiece interaction, and multimodal interaction was the fastest. In terms of the System Usability Score (SUS) [B+96], all SAR-based scenarios exceeded the 80.3 mark, which is typically considered excellent, and outperformed pure voice input. User satisfaction, measured by the Kunin scale [Kun55], mirrored this trend.

In summary, the positive evaluation results show that the combination of projection and object tracking serves as essential components in creating an intuitive and easy-to-use multimodal SAR system. The dynamic setup requires precise projection mapping and reliable tracking, both of which are successfully achieved. The plane registration (Sec. 3.3.1) and extrinsic refinement (Sec. 3.3.4) enable an easy setup for precisely aligned projections. Contour-based tracking (Sec. 3.2.4) allows robust subframe tracking in real-time. As the object's contour remains clearly visible despite the projection, projector-based 6D tracking was omitted in favor of a higher frame rate. However, regarding the experiments (Sec. 3.4.2), it may be essential for future iterations with different workpieces and projections. The configurable software can easily be adapted for different SAR applications.

7.2 Guided Construction in Architecture

Integrating AR guidance into architectural assembly processes can improve efficiency and accuracy, particularly in complex structures [KHKA22]. As tasks become more challenging, the need for AI solutions increases, especially when distinguishing between similar objects or recalling their shapes is difficult [HBJ+21]. One application where this is particularly evident is the construction of gridshell roofs, where spherical free-form surfaces are assembled from steel and glass components. The shape of the construction is defined by the arrangement of steel node elements, which, for one building, are manufactured in numerous slightly different asymmetric shapes. The exact shape of the structure is generated via optimization processes that consider material, geometry, and topology, with specific objectives tailored to the design intent and practical requirements of fabrication and assembly [SMTT18, SKT19]. For example, the Chadstone Shopping Centre in Melbourne, Australia (Fig. 7.4) is covered by a freeform glass roof with an underlying steel structure consisting of 2,810 different rigid steel nodes interconnected with 5,186 steel rods [SEB+17]. The steel nodes are star-shaped, with multiple connection points for metal supports. Due to their asymmetry, incorrect orientation during assembly can hinder the successful completion of the structure. Additionally, distinguishing the elements by eye is difficult, yet choosing the correct next element is crucial. In practice, drawings, along with adhesive or printed identifiers placed on objects or connection points, are used to document and communicate construction progress [SKT19, KHKA22].

The technologies presented in this work support the implementation of AR to assist on-site construction workers. The multi-object pose estimation (Chapter 4) is fast enough to distinguish multiple objects in real-time AR. In the combined pipeline (Chapter 5), it enables stable tracking and reliable detection of similar target objects. Since buildings are digitally planned, all target geometries are available for generating synthetic training data. For visualization, glasses-based AR could be integrated into helmet visors.

The *DigitalTWIN* research project explored this construction scenario through a three-part AR demonstrator, with each station addressing a specific challenge. Presented as an exhibit in the *Forum Digitale Technologien* in Berlin and the *seele* showroom in Gersthofen, the demonstrator highlights the potential of AR in construction. The interfaces between the planning software, glasses-based visualization on Microsoft HoloLens, and the tracking and detection module (the contribution of this thesis) are designed for real-world adaptability. The demonstrator's three stations are described below.

Station 1: Steel node positioning A user's task is to move and rotate a full-scale, lightweight replica of a steel node to a designated target pose. The target pose is unknown and can only be found using guidance provided through AR glasses. Due to the minimally asymmetric shape, it would be very difficult to identify the correct pose, even with printed instructions. The target object is mounted on a height-adjustable frame with a ball joint for rotation (Fig. 7.5a). A camera captures the action from a bird's-eye view, and the 6D

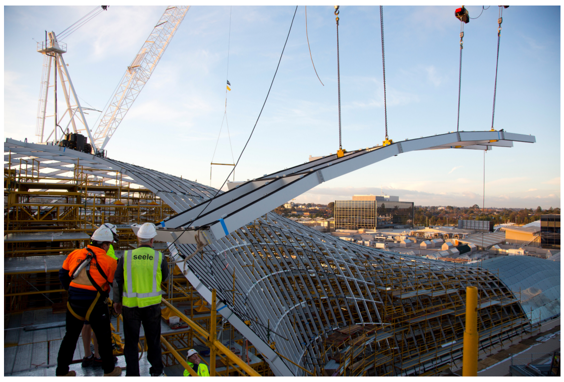
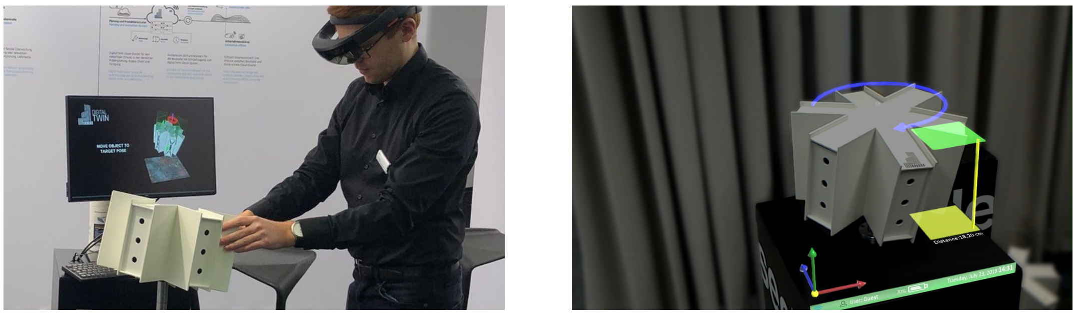


Figure 7.4: Construction work at the Chadstone Shopping Centre in Melbourne. A complex, free-form shell structure with a span of up to 44 meters is being built. Its design serves as a reference for the AR application. © seele

pose of the object is tracked without markers. The HoloLens visualization displays 3D arrows and planes indicating orientation and position offsets beside the object (Fig. 7.5b). As the object aligns with the target pose, the arrows shrink, and planes overlay. When the correct pose is reached within the allowed margins, the object is visually highlighted.

Object tracking operates at 60 frames per second (Sec. 3.2). Thanks to the high frame rate, the sharp geometric edges of the object, and the object's constrained path, the tracking remains stable even during fast movements. The object is large relative to the hands interacting with it, so as long as the node is not intentionally fully covered, no pose drift occurs. Estimated poses are sent to the visualization system via WebSocket. As the transmission and rendering on the glasses introduce only a few milliseconds of additional delay, visual feedback is nearly instantaneous. Accurate 3D visualization requires a reference coordinate system, initialized by scanning an AR marker on a table beneath the target object with both the glasses and the camera. The HoloLens tracks head movements using its internal sensors.

Station 2: Warehouse The *warehouse* station and the subsequent *construction site* station guide the construction of a gridshell roof using a miniaturized 3D-printed model. The 3D models of various components are generated by the architects' planning software and are utilized in computer-aided manufacturing processes, as shown in the upper row of Fig. 7.6. In parallel, they are also used to render images to train the pose estimation CNN. The user's task is to organize the objects according to their installation time during the construction process. A letter box structure on the table should be filled with the node elements in the correct order. In practical applications, false assignments can lead to construction errors in the worst case. Some objects appear so similar that they cannot be distinguished without assistance, even by a trained observer.



(a) Steel node positioning
© FraunhoferHHI/seele

(b) AR overlay in HoloLens

Figure 7.5: AR-guided steel node positioning. A replica of a gridshell facade component must be positioned and oriented accurately in an interactive demo. Its asymmetric shape makes guidance with 3D arrows essential for targeting the correct pose.

The planning software assigns sequence numbers to node elements in the digital model. This timetable is sent to the visualization unit, which maps elements to letterbox positions and overlays arrows guiding objects to their designated boxes.

Objects within the camera's view are identified using a combined CNN and local tracking method (Chapter 5). This method includes object identification, geometric verification, and local 6D tracking to stabilize the detection process. During this task, the accuracy benefits greatly from geometric pose verification. It helps to spot small differences that might be difficult for the synthetically trained neural network to detect. In the event of a false-positive validation, moving the objects resolves the perspective ambiguity, and reinitialization is triggered automatically.

Station 3: Construction site At the *construction site*, the node elements have to be mounted in the previously determined order. The outermost node elements are fixed to the table as the starting point, with the remaining 13 nodes and 42 connector sticks of three lengths spanning the roof between them. Since the sticks are easily distinguishable, their mounting does not require AI supervision. The construction frame is registered to the visualization unit, ensuring the target positions of node elements in 3D space remain fixed. A 3D arrow guides the handheld object toward its target, with an additional rotation arrow appearing for fine alignment when within a set distance threshold. The construction process and AR overlays are shown in Fig. 7.6.

Tracking and detection utilize the *far-range* tracking mode (Sec. 5.2.5): the identity of the moving object is known based on the desired assembly order, and only a part of the camera image is utilized for CNN-based identification. The local search helps to ignore objects already mounted to the connector sticks that are particularly hard to identify for the neural network. In contrast to Station 1, the relative movement of the object in the image, as well as the relative occlusion caused by hand interaction, is significantly

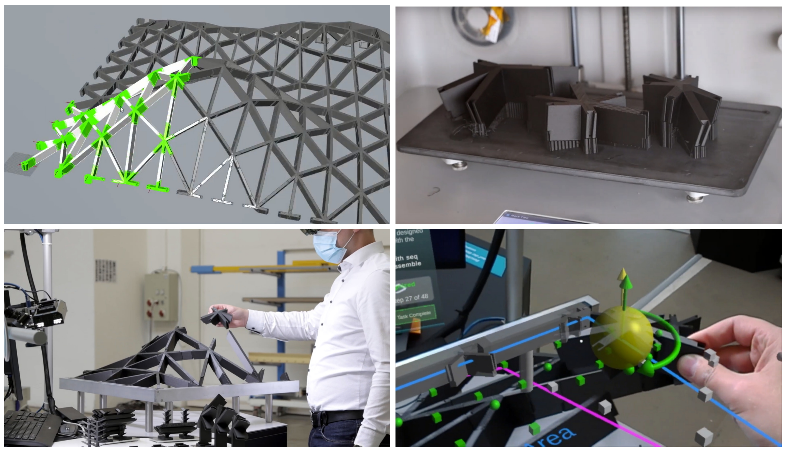


Figure 7.6: DigitalTWIN demonstrator for AR-supported construction. Above: Software planning of the construction. Manufacturing of components using a 3D printer. Below: Building the construction with AR support at the miniaturized *construction site*. Assembly is carried out with AR overlay. (Images from [see21])

increased. Therefore, the combined method with automatic reinitialization is essential to be capable of successfully solving the task. As the entire construction plan is stored on the server, the complete structure can be built using AR guidance.

In summary, the proposed methods integrate seamlessly into real-time AR systems with external visualization. The combined tracking, detection, and validation pipeline facilitates accurate, error-free sorting and montage of similar objects—something that individual methods cannot achieve. Applying this on a real-world construction site will pose additional challenges, such as camera placement. For manual object placement and sorting, a helmet-mounted camera may be suitable, while for assembling large structures, an additional crane-mounted camera may be required. Video processing can be performed on a 5G-enabled edge server, with only the visualization carried out on the mobile device. The presented workflows are also applicable in industrial assembly and sorting tasks [EGIB23, PRFG23], where stationary workplaces allow setups that more closely resemble the demonstrators.

7.3 Self Localization in Buildings

The camera localization method introduced in Chapter 6 globally registers camera images with a digital building model. A typical application involves the self-localization of users taking a photo, for example, with their cell phone in a specific environment. SPVLoc is scene-agnostic in indoor environments, provided that the new environments are structurally similar to those in the training data. Therefore, in contrast to other localization approaches for AR (e.g., [TOS+18, TBS+21, LGH22]), it does not require manual environment scanning and is designed to be well-suited for long-term re-localization.

However, the lack of building digitalization, resulting from the additional effort required and insufficient databases, makes high-quality reference models scarce [VW22]. In contrast, digital floor plans for most public buildings in Germany are archived by municipalities in non-digital format, with digital versions becoming increasingly available [Sen15]. Especially in projects designed before the advent of computer-aided design (CAD) tools, the documentation is typically in paper form, originating from manual drafts and later digitized through scanning. [PK04]. In the past, many studies focused on automatically reconstructing building models from floor plans, as outlined by Pizarro et al. [PHSS22].

The research project *BIMKIT* focused on automating the creation of accurate as-built 3D building models using AI technologies. The use case relevant to this thesis specifically involves generating reference models from existing data, including 2D scans of floor plans and on-site photos. Several AI solutions are combined into *service chains*, which are collections of fixed-function modules linked at their inputs and outputs to generate reference models by analyzing different data sources.

To illustrate this, consider an example service chain: First, a rasterized floor plan is analyzed to extract semantic information, such as the 2D locations of windows, doors, and walls [CBTHE23]. Second, the intermediate output is converted into a 3D format, specifically S3D, as understood by SPVLoc. Third, an indoor camera image is localized in the reconstructed environment. Fourth, specific objects in the image are localized using object detection or pose estimation. Finally, a coordinate transformation is performed to localize the detected objects within the 3D environment.

A similar pipeline can be employed to optimize and automate the process of fire safety inspections. Traditionally, a fire safety manager walks through a building with a checklist, and the low level of automation to date makes the process error-prone and time-consuming [HA23]. The research conducted by Aziz et al. [AKZS23, AK24] focuses on the detection of various types of fire safety equipment in images, and their detection module was integrated into the service chain. Detections are outputted as 2D bounding boxes using the YOLOv7 [WBL23]. Rather than estimating the 6D pose, it is sufficient to approximate the 3D position for each detection. This involves rendering the world coordinate image from the building model using the estimated camera pose and assigning each bounding box a world coordinate, for example, by querying its center.

Automating inspections with the proposed pipeline could eliminate the manual check marking performed by the fire safety manager, replacing it with the integration of a smartphone application into the process. Initially, a floor plan is uploaded and reconstructed. As the inspector walks through the building taking photos, information about the fire safety equipment is automatically extracted and transferred into the digital model. Using the localized detections, rule-based inspection validation can be automated online.

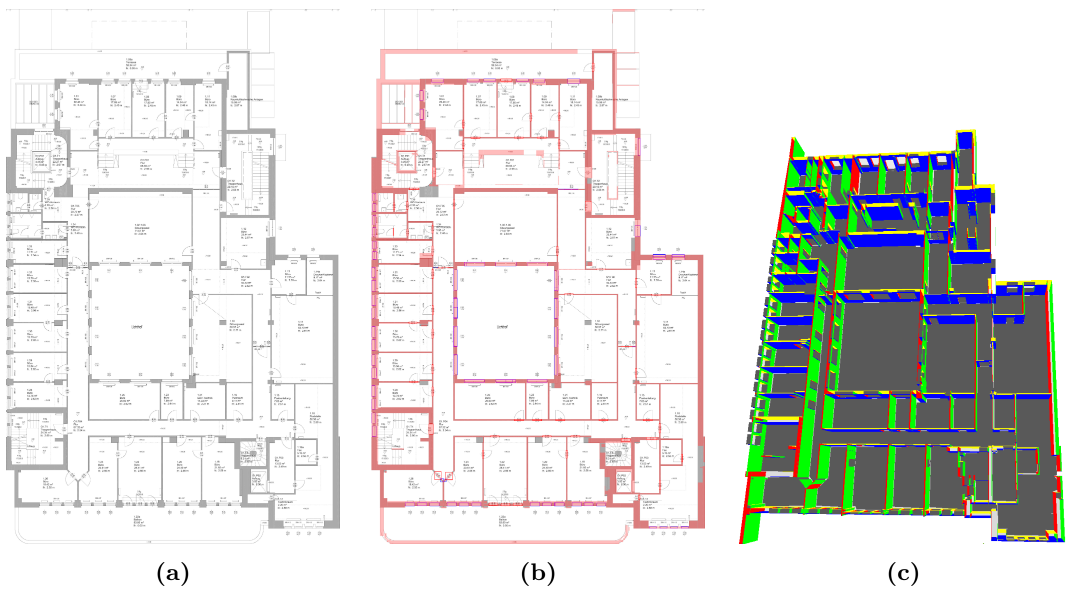


Figure 7.7: 3D model reconstruction from a floor plan for the purpose of localization. **(a)** shows the floorplan of the office building, **(b)** shows an overlay highlighting the extracted vectorized walls along with the identified doors and windows, and **(c)** shows the reconstructed 3D model. (Figure from [GCB23])

An advantage of the service chains is their modularity. For example, if available, the semantic building model could be generated from an existing BIM model instead of using AI floor plan analysis. Additionally, further analysis of detected objects, such as reading text labels [AK24] or estimating their 6D pose, is possible.

Insights into the application of SPVLoc in new domains Qualitative evaluations of the entire pipeline and its individual components were conducted using digital 2D plans and image data acquired from an empty floor of an office building [GCB23, AGK^{+}25].

Unlike the previous chapter's experiments, the model against which the localization is performed is generated by deep learning-based floor plan analysis [CBTHE23], shown in Fig. 7.7. This means the model is subject to detection errors; for example, the intersection over union (IoU) of the wall segmentation is 69.33%, indicating a good but not perfect reconstruction [GCB23]. Furthermore, room, window, and door heights are approximated with uniform default values, also limiting the accuracy of the 3D model.

Additionally, the model's generalization ability is challenged due to new domain gaps, namely the differences between the training data (single-family homes) and the test data (office complex), as well as differences in the capture equipment. Also, the test photos have a non-square aspect ratio. The pretrained SPVLoc (Sec. 6.3.3) is refined for 10 additional epochs on ZinD using random horizontal symmetric zero-padding to accommodate input

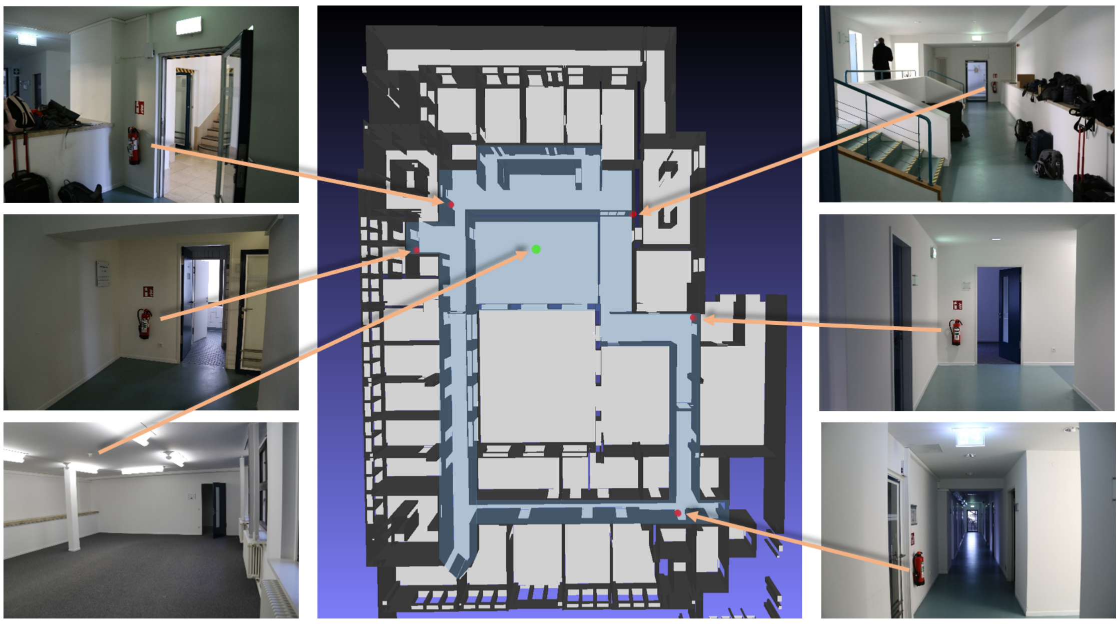


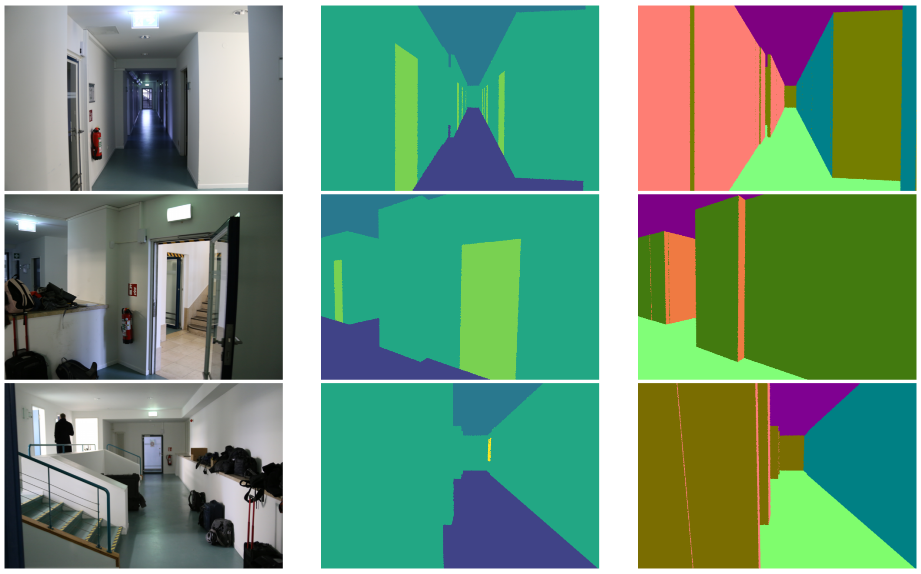
Figure 7.8: Localization of fire safety equipment via camera localization and object detection. The search area is highlighted in blue, with the detected smoke detector marked by a green sphere and the detected fire extinguishers by red spheres. The 6D camera poses are determined using SPVLoc.

images with varying aspect ratios. The camera's approximate horizontal FoV, determined by its focal length and sensor size, is required as input for inference.

A network trained on ZinD performs better on the new data than the network trained on S3D. This improvement can be attributed to the more diverse room shapes and real-world images in ZinD, which narrow the domain gap. However, the highly repetitive structures in the offices can lead to misplacements when matched across the entire map. Reducing the search space to specific areas of the floor significantly enhances performance. As mentioned earlier, planned extensions such as incorporating multi-image or video localization will help mitigate ambiguities in future work.

In a collaborative study [AGK+25], the camera localization accuracy was evaluated for images containing fire safety equipment. Although the exact 6D poses were not known, the locations of several depicted extinguishers and smoke detectors were. An object can only be placed at the correct location in the model if the camera is correctly localized. In the study, the search space was restricted to the all-around corridor and the large meeting room north of the atrium, covering approximately 300 square meters.

An overview of six input images and their estimated locations is shown in Fig. 7.8. In all cases, the detected fire safety equipment's location deviated by less than 1 meter from the ground truth. The results demonstrate the network's effectiveness in generalizing



(a) input images (b) rendered semantics (c) rendered normals

Figure 7.9: Examples for SPVLoc's robustness against model imperfections. Renderings of the model with the estimated top-1 pose are visualized.

to new environments. The layouts depicted in the images are distinct within the search space, allowing correct localization with the top-1 match.

Figure 7.9 illustrates SPVLoc's ability to handle 3D model imperfections. In the top row, there is a gap in the left wall and a missing door at the end of the corridor. In the middle row, the half-height wall is absent, as it was not depicted in the floor plan. In the bottom row, the half-height walls were detected as regular walls, resulting in a significantly different visual appearance, with the platform and railing on the left completely obscured. Additionally, the door at the end of the corridor is missing.

Despite these discrepancies, the method successfully localized the images, highlighting SPVLoc's robustness in focusing on key matchable regions and tolerance to missing elements. This demonstrates the network's potential to adapt to incomplete or inaccurate models, maintaining high localization accuracy even in visually altered environments. These results are promising, though future large-scale studies will need to validate the model's performance before it can be used in safety-critical applications.

8 Conclusion

This dissertation proposes novel solutions to key challenges in AR systems, focusing on 6D camera pose estimation, 6D object tracking, and the seamless integration of virtual content into real-world environments. The techniques developed effectively overlay and align 3D models with 2D images, while leveraging domain-crossing information—object silhouettes and their 3D shape edges, persistent semantic layouts, and actively projected known textures—to enhance the matching process between images and models. These are the core findings of this work:

Cross-domain features and new domains While contour-based 6D object tracking is a widely utilized technique, a major contribution of this thesis is the formulation of a new motion model and algorithm that extends its applicability to projector-camera systems. Projected textures behave differently from object textures during movement, and this behavior has been integrated into the mathematical framework of the analysis-by-synthesis approach. On the one hand, the simulation of projector intensity and self-shadows is designed to resemble the camera image; on the other hand, the projected contours are separated from the object contours to incorporate the new tracking terms in the appropriate regions.

The proposed CASAPose method emphasizes the use of object silhouettes and semantic object differences in CNN-based 6D position estimation to narrow the gap between synthetic training data and real-world test data. The neural network has access to the silhouette during 2D-3D keypoint regression, incorporating a new least-squares-based weight regression that allows the network to focus on contours during prediction. In the experiments, this approach led to improved accuracy compared to state-of-the-art methods at the time, which trained on synthetic data or on combinations of synthetic and unlabeled real-world data. An interesting finding is that the focus on silhouettes allowed the method to significantly outperform other approaches in the domain generalization test with sequences captured with different camera equipment, demonstrating reduced overfitting to the test dataset's characteristics.

The combined tracking system, involving deep learning and analysis-by-synthesis, uses geometric object fitting to validate the decisions of the neural network. This allows the correction of false positive results of the network caused by perspective ambiguities. When the tracking system encounters situations where objects appear indistinguishable due to similar shape projection, it resolves these ambiguities by utilizing pose validation in subsequent video frames. Additionally, the geometric matching process reduces the domain gap, as object contours of the 3D shape serve as domain-invariant features. Tests with 13 similar untextured objects also demonstrated the improved performance of a multi-object network with object-specific weights.

The thesis also introduces the first 6D system for global camera positioning using previously unseen semantic 3D models. This has been made possible by treating room shapes, as well as door and window positions, as domain-invariant features, and training a reliable mapping between images and 3D spaces on large-scale datasets. A novel approach is presented that directly matches semantic panoramic renderings with 2D photographs across domains, demonstrating that this method also generalizes to previously unseen scenes. The 2D-3D matching significantly improves the accuracy of pose determination compared to earlier plan-based localization methods.

Data and performance efficiency All presented concepts prioritize efficiency to ensure applicability in AR. For 6D object tracking, GPU-efficient shader techniques enable real-time frame rates. The closed-loop tracking, utilizing projection distortions, is also hardware-efficient: it uses only a simple projector-camera setup, eliminating the need for specialized capture hardware or marker placement. For 6D pose estimation, the CASAPose architecture is designed to be lightweight, extending its multi-object capacity with only a minimal increase in network size. The ability to estimate the poses of multiple objects in real-time is achieved by analyzing the entire image with a single-stage method, requiring minimal post-processing.

By balancing local tracking with global pose estimation within a unified system, GPU usage is reduced, and the CNN is activated only when necessary. The geometry-based pose refinement relaxes the realism constraints during the synthetic data generation process. This is particularly useful when the variation and number of objects being considered are high, limiting the time available for generating training datasets.

Lastly, the camera localization method SPVLoc achieves a high query rate by pre-computing scene features, significantly reducing computational requirements during inference. The maximum field of view of the synthetic equirectangular reference images, combined with training that enables the method to bridge large camera baselines, facilitates the use of sparse reference image grids. The small number of synthetic renderings thereby further increases efficiency. Furthermore, a localization method that does not require scene-specific image data or scene-specific training is inherently efficient. It eliminates the computationally intensive and time-consuming resources usually needed for this purpose.

Besides the experimental validation of the contributions, all approaches have been integrated into practically relevant demonstrators in case studies developed together with industry partners from different fields. Specifically, the SAR algorithms enhance projection alignment for visual inspection, while 6D pose estimation and tracking improve robustness in AR-assisted construction. Additionally, 6D image localization facilitates automated building digitalization and self-localization in new environments. Furthermore, various other applications can be envisioned. For example, 6D pose estimation with automatic validation is well-suited for robotic tasks, and model-based indoor localization can also enhance autonomous robot navigation. The combination of SAR object tracking with 6D pose estimation is particularly promising for mobile and handheld systems, especially as projectors become smaller and more integrated into future assistance technologies.

Outlook Among all the different aspects discussed in this work, the field of 6D object pose estimation has recently been experiencing the fastest progress. Since 2023, a notable trend has emerged toward eliminating object-specific training to enhance the algorithms' general applicability [HSL$^+$24], similar to the approach taken for localization in this thesis. Future work should focus on integrating the insights gathered here—such as the novel architectures and domain-invariant matching methods—with recent foundation models like SAM [KMR$^+$23] or DinoV2 [ODM$^+$23]. Ultimately, linking 3D geometry to a generic object detector could enhance the detector's ability to identify specific objects, while extracting class-adaptive weights from the geometry could improve performance using a unified decoder for keypoint detection.

Moreover, foundational models can play a crucial role in incorporating specific building details into the proposed 6D localization, as they are capable of identifying open sets of objects in images, offering a level of specificity that goes beyond what can be integrated into datasets like ZinD. Developing a network branch that focuses on domain-invariant features alongside a foundational model branch dedicated to prompted details will be a promising next step to integrate user-defined information into the reference models.

Lastly, a deeper integration of AI into the motion-focused aspects of this work is an important step for future research, particularly through the development of a combined deep 6D tracking and detection system or deep segmentation of projections in images. Both concepts have recently gained attention in the literature and hold significant potential to build on the insights of this thesis. In particular, the emergence of new datasets and ongoing advancements in synthetic dataset generation tools will provide the necessary foundation for further advancements.

A Appendix

A.1 Additional Loss Functions

Smooth $\mathbf{L_1}$ Loss The smooth L_1 loss [Gir15] is employed to regress the ground-truth unit vectors. Let O denote the set of visible objects, where $\mathcal{M}_l$ represents the object mask corresponding to the l-th object, and K_l denotes the set of 2D keypoints associated with the l-th object in O. The loss is defined as:

$$\begin{aligned} \mathcal{L}_{Vec} &= \sum_{l \in O} \frac{1}{|\mathcal{M}_l| \cdot |K_l|} \left(\sum_{\mathbf{k} \in K_l} \sum_{\mathbf{p} \in \mathcal{M}_l} \ell_1(\|\bar{\mathbf{v}}_k(\mathbf{p}) - \mathbf{v}_k(\mathbf{p})\|_1) \right), \\ \ell_1(a) &= 0.5a^2 \ \mathbf{1}(|a| < 1) + (|a| - 0.5) \ \mathbf{1}(|a| \geq 1) \end{aligned} \tag{A.1}$$

where $\mathbf{v}_k(\mathbf{p})$ represents an estimated unit direction vector between a pixel $\mathbf{p}$ and a keypoint $\mathbf{k}$, $\bar{\mathbf{v}}_k(\mathbf{p})$ represents a ground truth direction vector, and $\mathbf{1}(\cdot)$ is an indicator function. For each object, the term is normalized by dividing by the product of the number of pixels in the mask, $|\mathcal{M}_l|$, and the number of keypoints, $|K_l|$.

Differentiable Proxy Voting Loss (DPVL) DPVL [YZKL20] supports unit vector field prediction. The same angular error causes a ray passing through a pixel farther from a keypoint to deviate more significantly from the keypoint compared to a ray for a pixel closer to the keypoint. While the smooth L_1 loss penalizes both pixels equally, DPVL calculates the perpendicular distance from the ground-truth keypoint to the line defined by the pixel and its estimated direction vector, thereby encouraging more accurate vector predictions for distant pixels. Since DPVL ignores the sign of the estimated vectors, it is typically used in conjunction with the smooth L_1 loss. The loss is defined as:

$$\mathcal{L}_{PV} = \sum_{l \in O} \frac{1}{|\mathcal{M}_l| \cdot |K_l|} \left(\sum_{\mathbf{k} \in K_l} \sum_{\mathbf{p} \in \mathcal{M}_l} \ell_1 \left(\frac{|v_k^y k^x - v_k^x k^y + v_k^x p^y - v_k^y p^x|}{\sqrt{(v_k^x)^2 + (v_k^y)^2)}} \right) \right), \tag{A.2}$$

where $\mathbf{v}_k(\mathbf{p}) = (v_k^x, v_k^y)$, $\mathbf{p} = (p^x, p^y)$, $\mathbf{k} = (k^x, k^y)$ and the superscripts x, y represent horizontal and vertical coordinates respectively.

A.2 Additional Pose-Error Functions

The following is a shortened version of the description of pose error metrics used in the BOP Challenge 2020, adapted from the arXiv paper [Bop20a]:

Three error functions measure the error of an estimated pose $\hat{\mathbf{P}}$ with respect to the ground-truth pose $\bar{\mathbf{P}}$ of an object model $\mathcal{M}$. These functions, along with the *Average Recall* metric that combines the results into a single accuracy score, are defined below.

VSD (Visible Surface Discrepancy) VSD measures the misalignment of the visible part of the object surface, thereby it treats poses that result in indistinguishable shapes (ignoring color) as equivalent. To compute VSD, distance maps $\hat{\mathbf{D}}$ and $\bar{\mathbf{D}}$, which store the distance from the camera center to each visible 3D point, are generated by rendering $\mathcal{M}$ in the estimated pose $\hat{\mathbf{P}}$ and ground-truth pose $\bar{\mathbf{P}}$. Comparing these with the distance map $\mathbf{D_I}$ of the test image $\mathbf{I}$ results in the visibility masks $\hat{\mathbf{V}}$ and $\bar{\mathbf{V}}$, which mask the visible part of $\mathcal{M}$ in $\mathbf{I}$. Surface discrepancy is then measured for the visible object surface, with τ representing the allowed misalignment tolerance.

$$e_{\mathrm{VSD}}(\hat{\mathbf{D}}, \bar{\mathbf{D}}, \hat{\mathbf{V}}, \bar{\mathbf{V}}, \tau) = \mathrm{avg}_{p \in \hat{\mathbf{V}} \cup \bar{\mathbf{V}}} \begin{cases} 0 & \text{if } p \in \hat{\mathbf{V}} \cap \bar{\mathbf{V}} \wedge \left|\hat{\mathbf{D}}(p) - \bar{\mathbf{D}}(p)\right| < \tau \\ 1 & \text{otherwise} \end{cases} \tag{A.3}$$

MSSD (Maximum Symmetry-Aware Surface Distance) MSSD measures the maximum distance between model vertices, considering the set $S_{\mathcal{M}}$ of global symmetry transformations (identified as described in [Bop20a]) of the object model $\mathcal{M}$ consisting of the vertices $V_{\mathcal{M}}$. The maximum distance between model vertices is relevant for robotic manipulation, as it reflects the likelihood of a successful grasp. It is less affected by model geometry or surface sampling density than the ADD or ADD-S metric.

$$e_{\mathrm{MSSD}}(\hat{\mathbf{P}}, \bar{\mathbf{P}}, S_{\mathcal{M}}, V_{\mathcal{M}}) = \min_{\mathbf{S} \in S_{\mathcal{M}}} \max_{\mathbf{x} \in V_{\mathcal{M}}} \left\|\hat{\mathbf{P}}\mathbf{x} - \bar{\mathbf{P}}\mathbf{S}\mathbf{x}\right\| \tag{A.4}$$

MSPD (Maximum Symmetry-Aware Projection Distance) MSPD measures the maximum distance between model vertices projected into the image (denoted as $\mathrm{proj}(\cdot)$ for 2D projection) and is measured in pixels. As in MSSD, global object symmetries are considered, and the maximum distance is used to increase robustness against the geometry and sampling of the object model. Similar to the 2DP metric, it measures the perceivable discrepancy, which is especially important for AR applications where aligning the visual overlay takes precedence over alignment along the optical axis.

$$e_{\mathrm{MSPD}}(\hat{\mathbf{P}}, \bar{\mathbf{P}}, S_{\mathcal{M}}, V_{\mathcal{M}}) = \min_{\mathbf{S} \in S_{\mathcal{M}}} \max_{\mathbf{x} \in V_{\mathcal{M}}} \left\|\mathrm{proj}(\hat{\mathbf{P}}\mathbf{x}) - \mathrm{proj}(\bar{\mathbf{P}}\mathbf{S}\mathbf{x})\right\| \tag{A.5}$$

Average Recall Recall refers to the fraction of annotated object instances for which a correct pose is estimated, given a correctness threshold. The Average Recall for a function e, where $e \in \{e_{\text{VSD}}, e_{\text{MSSD}}, e_{\text{MSPD}}\}$, denoted as AR_e, is defined as the average of Recall rates calculated for multiple threshold settings θ_e, and, in the case of e_{VSD}, also for different misalignment tolerance settings τ. The combined accuracy of a method is measured by $\text{AR} = (\text{AR}_{\text{VSD}} + \text{AR}_{\text{MSSD}} + \text{AR}_{\text{MSPD}})/3$.

For AR_{VSD}, θ_{VSD} ranges from 0.05 to 0.5 with a step of 0.05 and τ ranges from 5% to 50% of the object diameter with a step of 5%. For AR_{MSSD}, θ_{MSSD} ranges from 5% to 50% of the object diameter with a 5% step. For AR_{MSPD}, θ_{MSPD} ranges from $5r$ to $50r$ with a $5r$ step, where $r = w/640$ and w is the image width in pixels.

A.3 Additional Object Pose Estimation Dataset Information

LINEMOD (LM) [HLI+12] consists of 15 RGB-D image sequences of various objects (15-30 cm in diameter) with minimal texture. Each sequence contains approximately 1,200 frames captured from the upper hemisphere, with $\pm 45°$ in-plane rotation. The poses of the objects are annotated relative to the camera, with distances ranging from 65 to 115 cm. A single object of interest is typically centered in the frame, surrounded by clutter from multiple other objects, but not significantly occluded. The camera, illumination, and desk setups remain consistent across all sequences, with ground truth poses determined using a marker board. However, the arrangement of clutter objects varies.

For pose estimation tasks, typically 13 of 15 sequences are used, excluding *cup* and *bowl* for which no watertight meshes are provided (e.g., following [PLH+19]). Fig. A.1 shows the 3D models for the remaining 13 target objects. Three objects from LM are also featured in the second sequence of the **HomebrewedDB (HB)** dataset, in a different setting (scene setup, illumination and camera intrinsics), which has been proposed for evaluating the generalization ability of pose estimation methods [KZSI19].

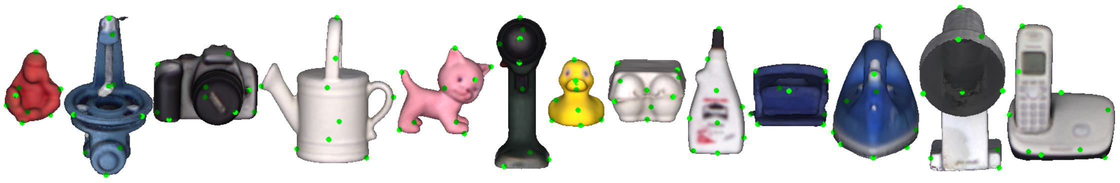


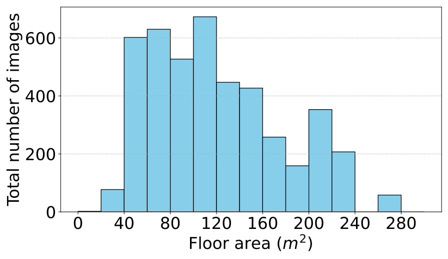
Figure A.1: Test objects of LM. From left to right: *ape, benchvise, cam, can, cat, drill, duck, eggbox, glue, holepuncher, iron, lamp, phone.* LM-O and the examined HB sequences show subsets of these objects. *Eggbox* and *glue* are considered symmetric. The 3D locations of 9 keypoints per object (not part of the original dataset), calculated using the Farthest Point Sampling [PLH+19], are shown as green spheres.

Occluded LINEMOD (LM-O) [BKM+14] is based on the *Benchvise* sequence of LINEMOD, with pose annotations for eight additional objects to create inter-object occlusions. This makes LM-O more suitable for multi-object pose estimation than the original LM. The BOP Challenge [HSD+20] uses a subset of 200 test images from LM-O and only considers object instances object instances with at least 10% of the projected surface area visible.

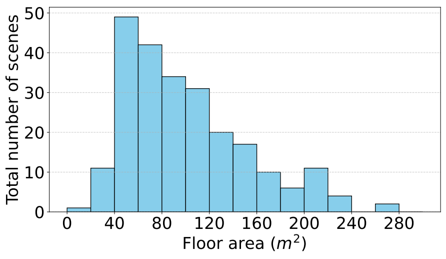
A.4 Additional Indoor Dataset Information

Zillow Indoor (ZInD) [CHL+21] As detailed in the main body of this thesis, the compiled test sets for ZinD consist of 4420 images for each combination of FoV and pitch/roll angle configuration. These images are captured across 238 test scenes, each representing a single unfurnished building floor.

The floor areas of the test scenes range from 17.91 m^2 to 278.37 m^2, with a median scene size of 90.30 m^2. Approximately 10% of the scenes have a floor area smaller than 46 m^2, while 10% exceed 177 m^2. Larger scenes contain more images, with about 5% of the images coming from the smallest 10% of scenes, and 18% from the largest 10% of scenes. The median floor area per image, accounting for the number of images per scene, is 111.86 m^2. The distribution of images and scenes across floor area ranges is shown in Fig. A.2. Figure A.2a shows the total images per range, while Figure A.2b shows the scenes per range. The distribution in Fig. A.2a is more relevant for assessing matching difficulty, as each image is matched against a single scene.



(a) Images per floor area ranges



(b) Scenes per floor area ranges

Figure A.2: Distribution of images and scenes across floor area ranges in the ZInD test sets.

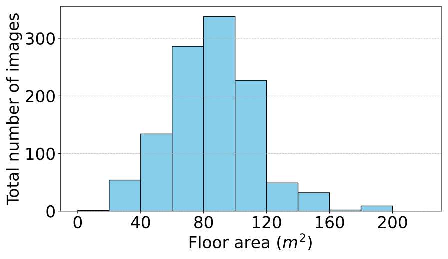
While the test image rotations are randomly sampled within specific ranges, the camera height is determined by the panorama images in the dataset. The median camera height is 1.43 m, with 95% of the samples ranging from 1.12 m to 1.67 m (Fig. A.4a).

Figure A.7 shows examples of the 3D models used for matching. The renderings highlight significant variation in scene layouts, including diverse room shapes and heights. In particular, the cut-out openings connecting the rooms contribute to this variability. Figure A.5 shows all camera positions for a scene, while Fig. A.6 displays the corresponding 90° FoV images, with pitch and roll sampled from ±10°.

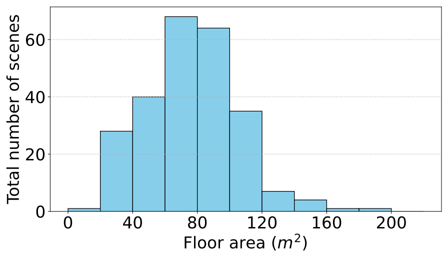
Structured3D (S3D) [ZZL+20] The floor areas in the 250 S3D test scenes have a median size of 74.55 m^2. 10% of the scenes have a floor area smaller than 37.3 m^2, while

10% exceed 109.9 m^2. The smallest scene is 17.21 m^2, and the largest scene (a single outlier) is 791.43 m^2. For the test sets used in Tab. 6.1, the median floor area per image is 87.68 m^2, with about 4% of the images coming from the smallest 10% of scenes and 14.5% from the largest 10% of scenes. The distribution of images and scenes across floor area ranges is shown in Fig. A.3. The camera heights are approximately equally distributed between 1.4 m and 1.8 m (Fig. A.4b).

The perspective images that come with S3D (used in Tab. 6.6 and Tab. 6.7) use the same scenes, and the distribution of floor area per image is similar, though with a higher total number of images. The camera heights are approximately normally distributed around 1.5 m (Fig. A.4c).

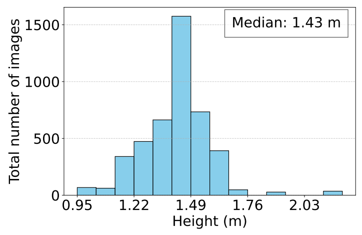


(a) Images per floor area ranges

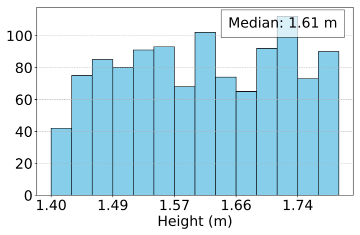


(b) Scenes per floor area ranges

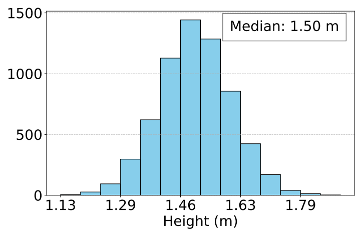
Figure A.3: Distribution of images and scenes across floor area ranges in the S3D test sets. A single outlier scene with a floor area exceeding 700 m^2 is excluded for better readability.



(a) ZInD



(b) S3D



(c) S3D (persp)

Figure A.4: Distribution of camera heights for ZInD and S3D test sets.

A.5 Additional Quantitative Results for SPVLoc

Figures A.5 and A.6 show additional qualitative results for SPVLoc, including an overview of all capture poses and reference images (90° FoV, ±10° pitch and roll) for a test scene from ZInD. Images are localized accurately across a wide range of scene configurations, including views through a narrow corridor (b, c, d, f, g, i), basement rooms with open drywall (e), glass doors leading to a balcony (a, m, q), a view captured in a built-in wardrobe (h), a bathroom (q, r), a colored kitchen (v), large wardrobe mirror doors (l, o, s), and built-in shelves next to a chimney (p, t). Out of 24 images, 23 are localized correctly within 50 cm, 18 within 25 cm, and 4 within 10 cm.

SPVLoc offers substantial robustness against model inaccuracies. The geometry of sloping roofs is not annotated in ZInD, and as a result, it is not represented in the model. Consequently, in several images (k, l, n, o, p, q, s, t, u, w, x), the room geometry in the models differs from the geometry visible in the images. Also, other minor model inaccuracies (e.g., near the opening in (w) or (x)) do not cause noticeable irritation. The only mislocalization (n) still finds a plausible camera pose in another room, where the mirrored part of the open door is interpreted as the door, and the ceiling height more accurately matches the visible ceiling than at the correct location.

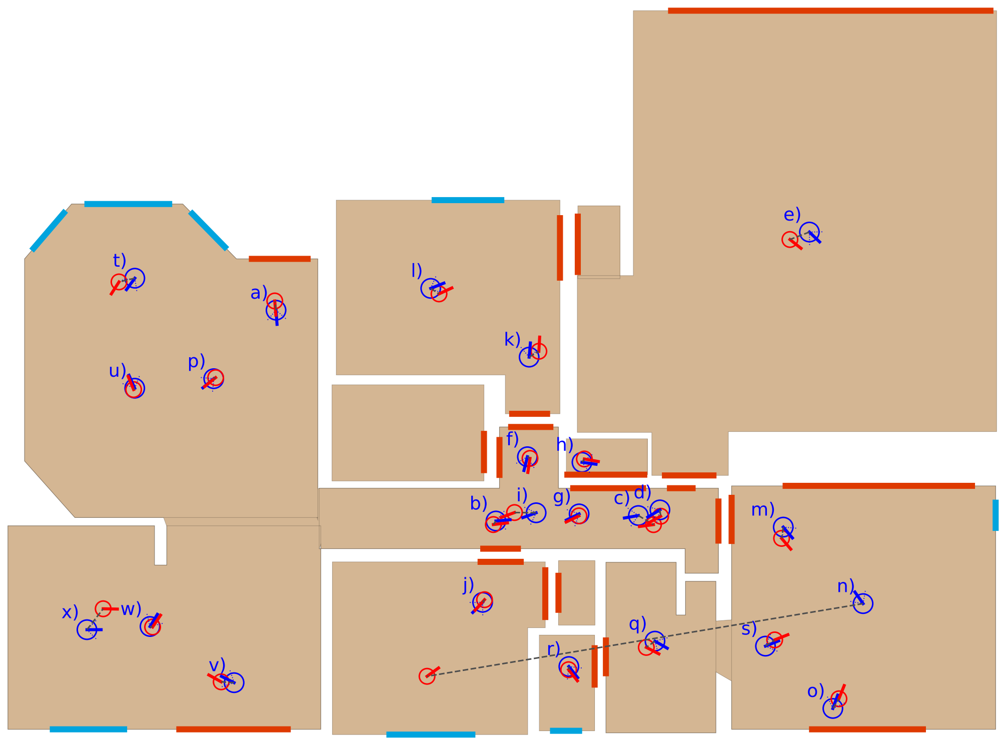
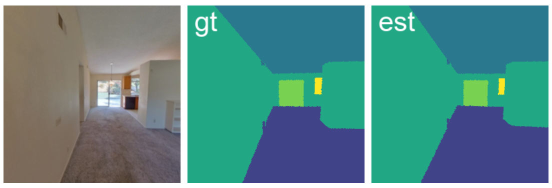
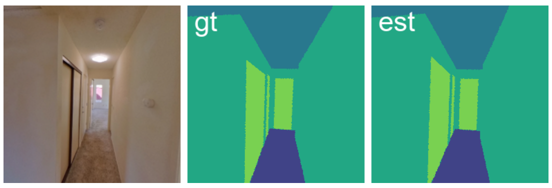


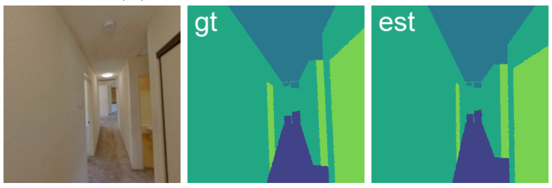
Figure A.5: Overview of capture positions and localization results for Scene 375 from ZInD. Ground truth poses are shown as blue circles, estimated poses as red circles, with corresponding poses connected by gray lines. The estimated orientation (yaw angle) is indicated by a colored line from the circle's center. Poses are estimated using SPVLoc with a fixed 90-degree FoV configuration, without refinement. See Fig. A.6 for the corresponding images and poses, and Fig. A.7e for the 3D model.



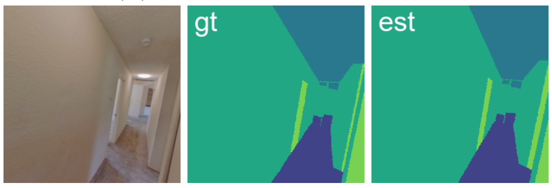
(a) $t_{\mathrm{err}} = 19.8\,\mathrm{cm}$, $r_{\mathrm{err}} = 1.6^\circ$



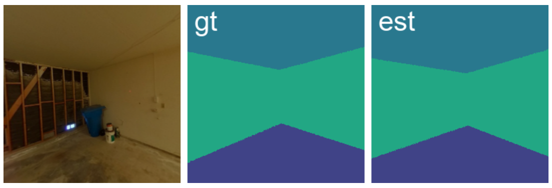
(b) $t_{\mathrm{err}} = 14.1\,\mathrm{cm}$, $r_{\mathrm{err}} = 2.7^\circ$



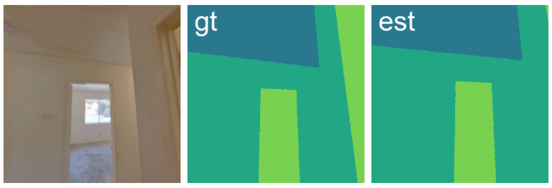
(c) $t_{\mathrm{err}} = 27.4\,\mathrm{cm}$, $r_{\mathrm{err}} = 2.9^\circ$



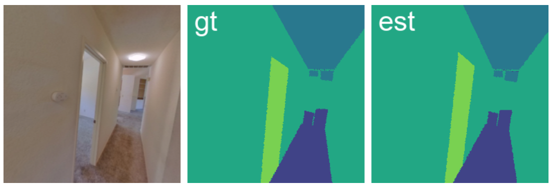
(d) $t_{\mathrm{err}} = 11.1\,\mathrm{cm}$, $r_{\mathrm{err}} = 1.8^\circ$



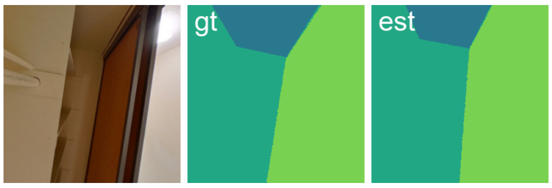
(e) $t_{\mathrm{err}} = 32.3\,\mathrm{cm}$, $r_{\mathrm{err}} = 5.7^\circ$



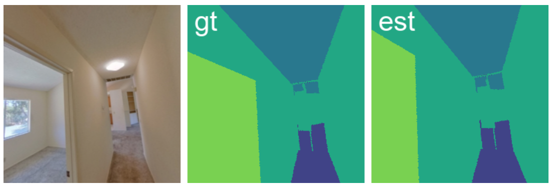
(f) $t_{\mathrm{err}} = 11.6\,\mathrm{cm}$, $r_{\mathrm{err}} = 5.3^\circ$



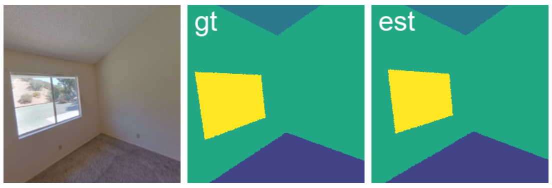
(g) $t_{\mathrm{err}} = 6.7\,\mathrm{cm}$, $r_{\mathrm{err}} = 1.1^\circ$



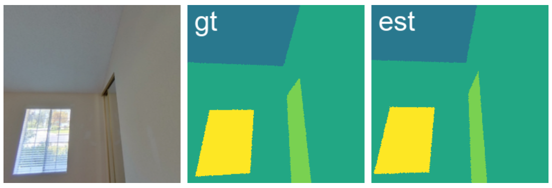
(h) $t_{\mathrm{err}} = 18.8\,\mathrm{cm}$, $r_{\mathrm{err}} = 6.3^\circ$



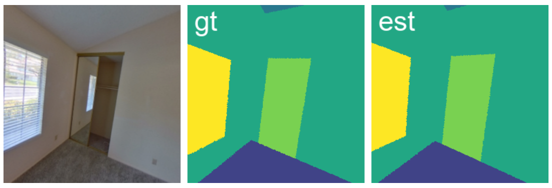
(i) $t_{\mathrm{err}} = 35.6\,\mathrm{cm}$, $r_{\mathrm{err}} = 1.1^\circ$



(j) $t_{\mathrm{err}} = 7.7\,\mathrm{cm}$, $r_{\mathrm{err}} = 2.7^\circ$

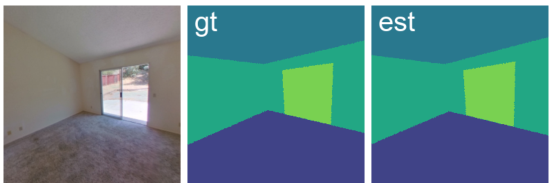


(k) $t_{\mathrm{err}} = 18.3\,\mathrm{cm}$, $r_{\mathrm{err}} = 1.9^\circ$

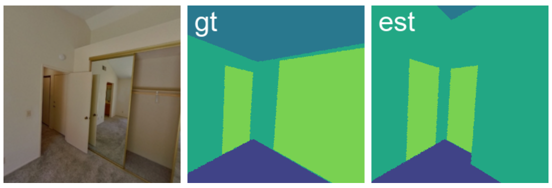


(l) $t_{\mathrm{err}} = 16.5\,\mathrm{cm}$, $r_{\mathrm{err}} = 3.2^\circ$

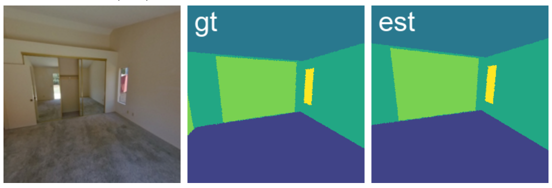
Figure A.6: Overview of all images for Scene 375 from ZInD, along with renderings using the ground truth (gt) and estimated (est) poses, and the corresponding pose errors. Poses were estimated using SPVLoc with a fixed 90-degree FoV configuration, without refinement. See Fig. A.5 for the corresponding floor plan and Fig. A.7e for the corresponding 3D model. *(Continued on next page)*



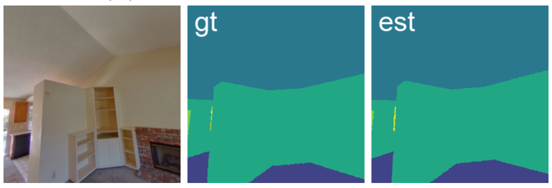
(m) $t_{\text{err}} = 18.4\,\text{cm}$, $r_{\text{err}} = 0.8°$



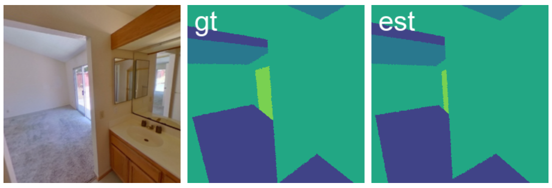
(n) $t_{\text{err}} = 685.7\,\text{cm}$, $r_{\text{err}} = 89.3°$



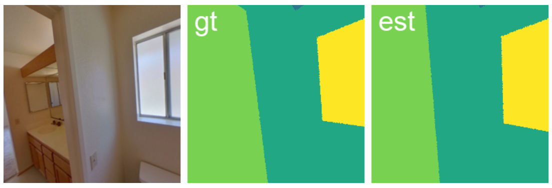
(o) $t_{\text{err}} = 18.7\,\text{cm}$, $r_{\text{err}} = 2.4°$



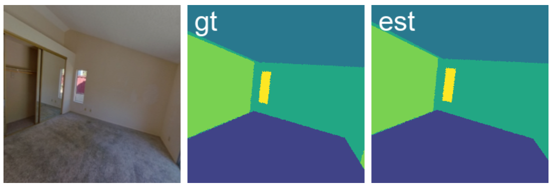
(p) $t_{\text{err}} = 7.5\,\text{cm}$, $r_{\text{err}} = 1.3°$



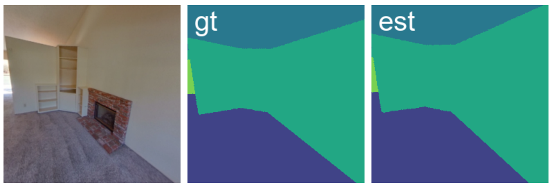
(q) $t_{\text{err}} = 19.9\,\text{cm}$, $r_{\text{err}} = 3.9°$



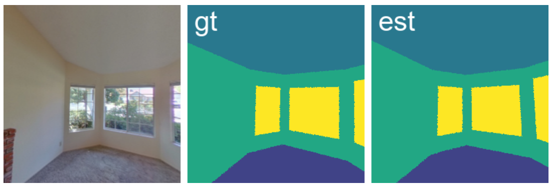
(r) $t_{\text{err}} = 5.7\,\text{cm}$, $r_{\text{err}} = 2.8°$



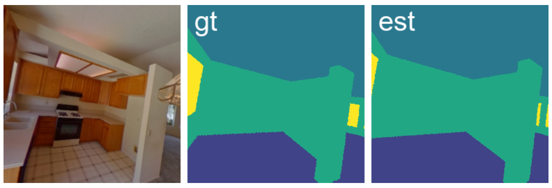
(s) $t_{\text{err}} = 20.2\,\text{cm}$, $r_{\text{err}} = 0.7°$



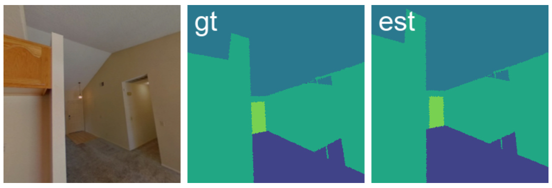
(t) $t_{\text{err}} = 26.5\,\text{cm}$, $r_{\text{err}} = 3.5°$



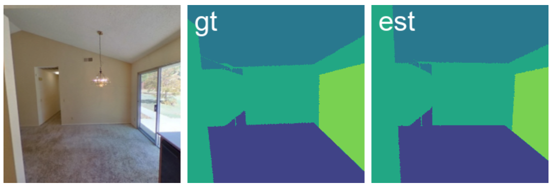
(u) $t_{\text{err}} = 13.8\,\text{cm}$, $r_{\text{err}} = 3.4°$



(v) $t_{\text{err}} = 21.8\,\text{cm}$, $r_{\text{err}} = 2.4°$

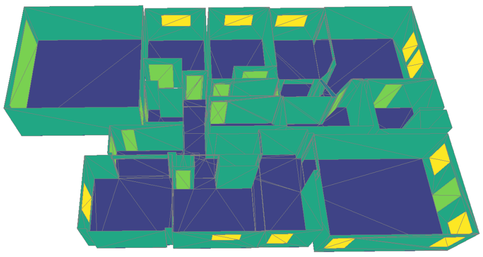


(w) $t_{\text{err}} = 18.8\,\text{cm}$, $r_{\text{err}} = 3.6°$

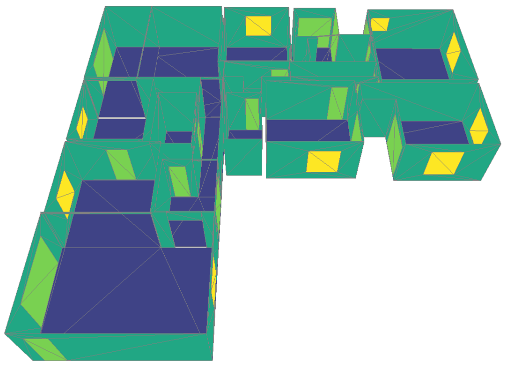


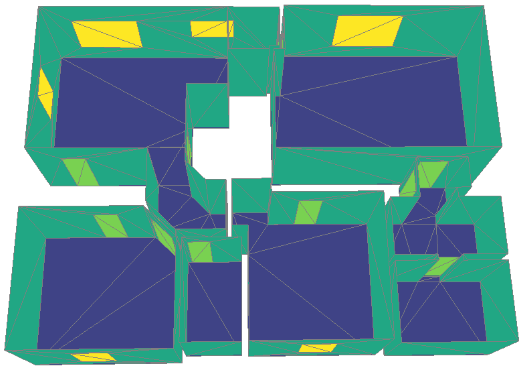
(x) $t_{\text{err}} = 41.9\,\text{cm}$, $r_{\text{err}} = 2.7°$

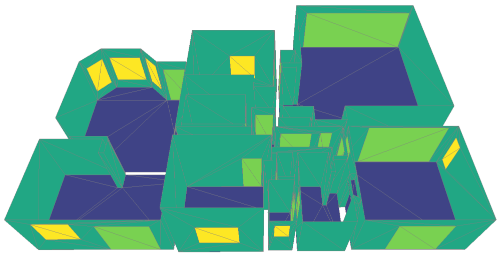
Figure A.6: *(Continuation.)* **Overview of all images for Scene 375 from ZInD**, along with renderings using the ground truth (gt) and estimated (est) poses, and the corresponding pose errors. Poses were estimated using SPVLoc with a fixed 90-degree FoV configuration, without refinement. See Fig. A.5 for the corresponding floor plan and Fig. A.7e for the corresponding 3D model.


(a) Scene 10, Floor 0

(b) Scene 675, Floor 1


(c) Scene 75, Floor 0


(d) Scene 165, Floor 1


(e) Scene 375, Floor 0

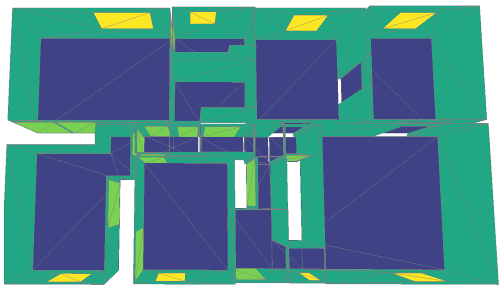

(f) Scene 905, Floor 0

Figure A.7: Example 3D scenes from the ZInD dataset. The scenes depict a non-Manhattan world (a 3D environment where walls, floors, and ceilings are not strictly aligned to orthogonal axes), with varying room heights, interconnected rooms forming complex geometries, and a range of window and door sizes and heights. Doors are colored light green, and windows are colored yellow.

Bibliography

[AFS$^+$11] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S.M. Seitz, and R. Szeliski. Building Rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[AFS$^+$18] P. Ammirato, C.Y. Fu, M. Shvets, J. Kosecka, and A.C. Berg. Target driven instance detection. *arXiv preprint arXiv:1803.04610*, 2018.

[AGK$^+$25] A. Aziz, N. Gard, M. König, P. Eisert, J.H. Heinbach, and L. Trost. Visual fire safety inspection framework using computer vision algorithms. *Journal of Computing in Civil Engineering*, 39(5):04025068, 2025.

[AGT$^+$16] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[AIS18] H. Asayama, D. Iwai, and K. Sato. Fabricating diminishable visual markers for geometric registration in projection mapping. *IEEE Transactions on Visualization and Computer Graphics*, 24(2):1091–1102, 2018.

[AK24] A. Aziz and M. König. Bilderkennungsmethoden für eine teilautomatisierte Inspektion von Brandschutzanlagen: Eine Studie unter Berücksichtigung von maschinellen Lernverfahren und der Integration von Expertenwissen. *Bautechnik*, 101(3):159–165, 2024.

[AKW19] D. Acharya, K. Khoshelham, and S. Winter. BIM-PoseNet: Indoor camera localisation using a 3D indoor model and deep learning from synthetic images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 150:245–258, 2019.

[AKZS23] A. Aziz, M. König, S. Zengraf, and J.U. Schulz. Instance segmentation of fire safety equipment using Mask R-CNN. *Advances in Information Technology in Civil and Building Engineering*, 357:121, 2023.

[ALCL21] L. Aing, W.N. Lie, J.C. Chiang, and G.S. Lin. InstancePose: Fast 6DoF pose estimation for multiple objects from a single RGB image. In

Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV) Workshops, 2021.

[AMMA16] R.S. Andersen, O. Madsen, T.B. Moeslund, and H.B. Amor. Projecting robot intentions into human environments. In *Proc. Int. Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2016.

[AOT10] S. Audet, M. Okutomi, and M. Tanaka. Direct image alignment of projector-camera systems with planar surfaces. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[ARKW19] D. Acharya, M. Ramezani, K. Khoshelham, and S. Winter. BIM-Tracker: A model-based visual tracking approach for indoor localisation using a 3D building model. *ISPRS Journal of Photogrammetry and Remote Sensing*, 150:157–171, 2019.

[ATK23] D. Acharya, C.J. Tatli, and K. Khoshelham. Synthetic-real image domain adaptation for indoor camera pose regression using a 3D model. *ISPRS Journal of Photogrammetry and Remote Sensing*, 202:405–421, 2023.

[ATM$^+$22] D. Acharya, R. Tennakoon, S. Muthu, K. Khoshelham, R. Hoseinnezhad, and A. Bab-Hadiashar. Single-image localisation using 3D models: Combining hierarchical edge maps and semantic segmentation for domain adaptation. *Automation in Construction*, 136:104–152, 2022.

[B$^+$96] J. Brooke et al. SUS-A quick and dirty usability scale. *Usability Evaluation in Industry*, 189(194):4–7, 1996.

[BB95] S.S. Beauchemin and J.L. Barron. The computation of optical flow. *ACM Computing Surveys*, 27(3):433–466, 1995.

[BBIG17] A.H. Bermano, M. Billeter, D. Iwai, and A. Grundhöfer. Makeup Lamps: Live augmentation of human faces via projection. *Computer Graphics Forum*, 36(2):311–323, 2017.

[BGK$^+$18] S. Brahmbhatt, J. Gu, K. Kim, J. Hays, and J. Kautz. Geometry-aware learning of maps for camera localization. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[BHG$^+$24] J. Bai, L. Huang, J. Guo, W. Gong, Y. Li, and Y. Guo. 360-GS: Layout-guided panoramic Gaussian splatting for indoor roaming. *arXiv preprint arXiv:2402.00763*, 2024.

[BHK19] F. Baek, I. Ha, and H. Kim. Augmented reality system for facility management using image-based indoor localization. *Automation in Construction*, 99:18–26, 2019.

[BHW^{+}24] D. Bauer, P. Hönig, J.B. Weibel, J. García-Rodríguez, M. Vincze, et al. Challenges for monocular 6D object pose estimation in robotics. *IEEE Transactions On Robotics*, 2024.

[BIG06] A. Ben-Israel and T.N. Greville. *Generalized inverses: theory and applications.* Springer Science & Business Media, 2006.

[BJ19] G. Billings and M. Johnson-Roberson. SilhoNet: An RGB method for 6D object pose estimation. *IEEE Robotics And Automation Letters*, 4(4):3727–3734, 2019.

[BKM^{+}14] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6D object pose estimation using 3D object coordinates. In *Proc. European Conf. on Computer Vision (ECCV)*, 2014.

[BMK^{+}16] E. Brachmann, F. Michel, A. Krull, M.Y. Yang, S. Gumhold, et al. Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[Bop20a] BOP: Benchmark for 6D object pose estimation. Submission: EPOS-BOP20-PBR/LM-O. `https://bop.felk.cvut.cz/sub_info/1382/`, 2020. [Online; accessed 26-March-2025].

[Bop20b] BOP: Benchmark for 6D object pose estimation. Submission: PVNet-CVPR19/LM-O. `https://bop.felk.cvut.cz/sub_info/1147/`, 2020. [Online; accessed 26-March-2025].

[BR08] C. Bibby and I. Reid. Robust real-time visual tracking using pixel-wise posteriors. In *Proc. European Conf. on Computer Vision (ECCV)*, 2008.

[BSM^{+}24] P. Banerjee, S. Shkodrani, P. Moulon, S. Hampali, F. Zhang, J. Fountain, E. Miller, S. Basol, R. Newcombe, R. Wang, J.J. Engel, and T. Hodaň. Introducing HOT3D: An egocentric dataset for 3D hand and object tracking. *arXiv preprint arXiv:2406.09598*, 2024.

[BTM15] L. Besharati Tabrizi and M. Mahvash. Augmented reality-guided neurosurgery: accuracy and intraoperative application of an image projection technique. *Journal Of Neurosurgery*, 123(1):1–6, 2015.

[BWS05] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal Of Computer Vision*, 61:211–231, 2005.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, PAMI-8(6):679–698, 1986.

[CBFAB94] P. Charbonnier, L. Blanc-Feraud, G. Aubert, and M. Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *Proc. IEEE Int. Conf. on Image Processing (ICIP)*, volume 2, 1994.

[CBTHE23] A. Cambeiro Barreiro, M. Trzeciakiewicz, A. Hilsmann, and P. Eisert. Automatic reconstruction of semantic 3D models from 2D floor plans. In *Proc. Int. Conf. on Machine Vision Applications (MVA)*, 2023.

[CCPB24] S. Chen, T. Cavallari, V.A. Prisacariu, and E. Brachmann. Map-relative pose regression for visual re-localization. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[CCZ$^+$20] B. Cheng, M.D. Collins, Y. Zhu, T. Liu, T.S. Huang, H. Adam, and L.C. Chen. Panoptic-DeepLab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[CGHSMR18] M. Chae, D. Ganhewa, D. Hunter-Smith, and W. Matthew Rozen. Direct augmented reality computed tomographic angiography technique (ARC): an innovation in preoperative imaging. *European Journal Of Plastic Surgery*, 41(4):415–420, 2018.

[CGS$^+$21] F. Cole, K. Genova, A. Sud, D. Vlasic, and Z. Zhang. Differentiable surface rendering via non-differentiable sampling. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2021.

[CHL$^+$21] S. Cruz, W. Hutchcroft, Y. Li, N. Khosravan, I. Boyadzhiev, and S.B. Kang. Zillow Indoor Dataset: Annotated floor plans with 360° panoramas and 3D room layouts. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[CMS$^+$20] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *Proc. European Conf. on Computer Vision (ECCV)*, 2020.

[CPC+20] B. Chen, A. Parra, J. Cao, N. Li, and T.J. Chin. End-to-end learnable geometric vision by backpropagating pnp optimization. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[CSP+23] P. Chojecki, D. Strazdas, D. Przewozny, N. Gard, D. Runde, N. Hoerner, A. Al-Hamadi, P. Eisert, and S. Bosse. Assessing the value of multimodal interfaces: A study on human–machine interaction in weld inspection workstations. *Sensors*, 23(11):5043, 2023.

[CWM22] L. Chen, Y. Wu, and D. Merhof. Instance segmentation of dense and overlapping objects via layering. In *Proc. British Machine Vision Conf. (BMVC)*, 2022.

[CWVP24] C. Chen, R. Wang, C. Vogel, and M. Pollefeys. F3Loc: Fusion and filtering for floorplan localization. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[Dem] L. Demes. CC0 Textures. `https://cc0-textures.com/`. [Online; accessed 26-March-2025].

[DFBT99] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 2, 1999.

[DG88] E.D. Dickmanns and V. Graefe. Dynamic monocular machine vision. *Machine Vision And Applications*, 1(4):223–240, 1988.

[DJW+20] Y. Dong, L. Ji, S. Wang, P. Gong, J. Yue, R. Shen, C. Chen, and Y. Zhang. Accurate 6DOF pose tracking for texture-less objects. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(5):1834–1848, 2020.

[DSK17] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2017.

[DSL+20] A. Dundar, K. Sapra, G. Liu, A. Tao, and B. Catanzaro. Panoptic-based image synthesis. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[DSW+19] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and H. Katam. BlenderProc. *arXiv preprint arXiv:1911.01911*, 2019.

[DWS$^+$19] M. Ding, Z. Wang, J. Sun, J. Shi, and P. Luo. CamNet: Coarse-to-fine retrieval for camera re-localization. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[EFR08] P. Eisert, P. Fechteler, and J. Rurainsky. 3-D tracking of shoes for Virtual Mirror applications. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[EG97] P. Eisert and B. Girod. Model-based 3D-motion estimation with illumination compensation. In *Proc. Sixth Int. Conf. on Image Processing and Its Applications*, 1997.

[EGIB23] M. Eswaran, A.K. Gulivindala, A.K. Inkulu, and M.R. Bahubalendruni. Augmented reality-based guidance in product assembly and maintenance/repair perspective: A state of the art review on challenges and opportunities. *Expert Systems with Applications*, 213:118983, 2023.

[EM20] J. Egger and T. Masood. Augmented reality in support of intelligent manufacturing–a systematic literature review. *Computers & Industrial Engineering*, 140:106195, 2020.

[Eve01] C.W. Everitt. Projective texture mapping. `https://developer.download.nvidia.com/assets/gamedev/docs/projective_texture_mapping.pdf`, 2001. [Online; accessed 26-March-2025].

[FB81] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[FJ07] H. Fuchs and T. Johnson. Real-time projector tracking on complex geometry using ordinary imagery. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[FLFPY$^+$20] C. Fernandez-Labrador, J.M. Facil, A. Perez-Yus, C. Demonceaux, J. Civera, and J.J. Guerrero. Corners for layout: End-to-end layout recovery from 360 images. *IEEE Robotics And Automation Letters*, 5(2):1255–1262, 2020.

[FMI20] K. Fukamizu, L. Miyashita, and M. Ishikawa. Elamorph projection: Deformation of 3D shape by dynamic projection mapping. In *Proc. IEEE Int. Symposium on Mixed and Augmented Reality (ISMAR)*, 2020.

[FZH$^+$22] Z. Fan, Y. Zhu, Y. He, Q. Sun, H. Liu, and J. He. Deep learning on monocular object pose detection and tracking: A comprehensive overview. *ACM Computing Surveys*, 55(4):1–40, 2022.

[GCB23] N. Gard and A. Cambeiro Barreiro. Towards automated digital building model generation from floorplans and on-site images. In *Proc. 34th Forum Bauinformatik*. Ruhr-Universität Bochum, 2023.

[GE18] N. Gard and P. Eisert. Markerless closed-loop projection plane tracking for mobile projector-camera systems. In *Proc. IEEE Int. Conf. on Image Processing (ICIP)*. IEEE, 2018.

[GEB15] L.A. Gatys, A.S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[GH12] K.M. Guttag and S. Hurley. Laser + LCOS projection-technology revolution. *Journal Of The Society For Information Display*, 20(5):279–285, 2012.

[GHE19] N. Gard, A. Hilsmann, and P. Eisert. Projection distortion-based object tracking in shader lamp scenarios. *IEEE Transactions on Visualization and Computer Graphics*, 25(11):3105–3113, 2019.

[GHE22a] N. Gard, A. Hilsmann, and P. Eisert. CASAPose: Class-adaptive and semantic-aware multi-object pose estimation. In *Proc. British Machine Vision Conf. (BMVC)*. BMVA Press, 2022.

[GHE22b] N. Gard, A. Hilsmann, and P. Eisert. Combining local and global pose estimation for precise tracking of similar objects. In *Proc. Int. Conf. on Computer Vision Theory and Applications (VISAPP)*. SCITEPRESS, 2022.

[GHE24] N. Gard, A. Hilsmann, and P. Eisert. SPVLoc: Semantic panoramic viewport matching for 6D camera localization in unseen environments. In *Proc. European Conf. on Computer Vision (ECCV)*. Springer, Cham, 2024.

[Gir15] R. Girshick. Fast R-CNN. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2015.

[GRJ$^+$19] N. Gard, J.C. Rosenthal, S. Jurk, A. Schneider, and P. Eisert. Image-based measurement by instrument tip tracking for tympanoplasty using digital surgical microscopy. In *SPIE Medical Imaging*, 2019.

[GRL18] A. Grabner, P.M. Roth, and V. Lepetit. 3D pose estimation and 3D model retrieval for objects in the wild. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[GRP22] F. Gorschlüter, P. Rojtberg, and T. Pöllabauer. A survey of 6D object detection based on 3D models for industrial applications. *Journal Of Imaging*, 8(3):53, 2022.

[HA23] J.H. Heinbach and A. Aziz. Visual partial inspection of fire safety equipment using machine learning. In *Proc. 34th Forum Bauinformatik*. Ruhr-Universität Bochum, 2023.

[HAZ+20] T. Hou, A. Ahmadyan, L. Zhang, J. Wei, and M. Grundmann. MobilePose: Real-time pose estimation for unseen objects with weak shape supervision. *arXiv preprint arXiv:2003.03522*, 2020.

[HB17] X. Huang and S.J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2017.

[HB22] R.L. Haugaard and A.G. Buch. SurfEmb: Dense and continuous correspondence distributions for object pose estimation with learnt surface embeddings. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[HBJ+21] V. Havard, D. Baudry, B. Jeanne, A. Louis, and X. Savatier. A use case study comparing augmented reality (AR) and electronic document-based maintenance instructions considering tasks complexity and operator competency level. *Virtual Reality*, pages 1–16, 2021.

[HBM20] T. Hodaň, D. Barath, and J. Matas. EPOS: Estimating 6D pose of objects with symmetries. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[HCL12] B. He, Z. Chen, and Y. Li. Calibration method for a central catadioptric-perspective camera system. *Journal of the Optical Society of America*, 29(11):2514–2524, 2012.

[HF10] S. Henderson and S. Feiner. Exploring the benefits of augmented reality documentation for maintenance and repair. *IEEE Transactions on Visualization and Computer Graphics*, 17(10):1355–1368, 2010.

[HFWS20] Y. Hu, P. Fua, W. Wang, and M. Salzmann. Single-stage 6D object pose estimation. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[HFZL20] Z. He, W. Feng, X. Zhao, and Y. Lv. 6D pose estimation of objects: Recent technologies and challenges. *Applied Sciences*, 11(1):228, 2020.

[HHFS19] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann. Segmentation-driven 6D object pose estimation. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[HHO$^+$17] T. Hodaň, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis. T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. In *Proc. IEEE Winter Conf. on Applications of Computer Vision (WACV)*, 2017.

[HJP22] H. Howard-Jenkins and V.A. Prisacariu. LaLaLoc++: Global floor plan comprehension for layout localisation in unvisited environments. In *Proc. European Conf. on Computer Vision (ECCV)*, 2022.

[HJRSP21] H. Howard-Jenkins, J.R. Ruiz-Sarmiento, and V.A. Prisacariu. LaLaLoc: Latent layout localisation in dynamic, unvisited environments. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2021.

[HK16] N. Hashimoto and D. Kobayashi. Dynamic spatial augmented reality with a single IR camera. In *Proc. ACM SIGGRAPH Posters*, 2016.

[HKPK18] I. Ha, H. Kim, S. Park, and H. Kim. Image retrieval using BIM and features from pretrained VGG network for indoor localization. *Building And Environment*, 140:23–31, 2018.

[HLI$^+$12] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *Proc. Asian Conf. on Computer Vision (ACCV)*, 2012.

[HLVDMW17] G. Huang, Z. Liu, L. Van Der Maaten, and K.Q. Weinberger. Densely connected convolutional networks. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[HS81] B. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 1981.

[HS90] C. Harris and C. Stennett. RAPID-a video rate object tracker. In *Proc. British Machine Vision Conf. (BMVC)*, 1990.

[HSD$^+$20] T. Hodaň, M. Sundermeyer, B. Drost, Y. Labbé, E. Brachmann, F. Michel, C. Rother, and J. Matas. BOP Challenge 2020 on 6D object localization. In *Proc. European Conf. on Computer Vision (ECCV) Workshops*, 2020.

[HSL21] B. Huang, T. Sun, and H. Ling. End-to-end full projector compensation. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 44(6):2953–2967, 2021.

[HSL$^+$24] T. Hodaň, M. Sundermeyer, Y. Labbe, V.N. Nguyen, G. Wang, E. Brachmann, B. Drost, V. Lepetit, C. Rother, and J. Matas. BOP Challenge 2023 on detection, segmentation and pose estimation of seen and unseen rigid objects. *arXiv preprint arXiv:2403.09799*, 2024.

[HVG$^+$19] T. Hodaň, V. Vineet, R. Gal, E. Shalev, J. Hanzelka, T. Connell, P. Urbina, S.N. Sinha, and B. Guenter. Photorealistic image synthesis for object instance detection. In *Proc. IEEE Int. Conf. on Image Processing (ICIP)*, 2019.

[HW77] P.W. Holland and R.E. Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics-theory and Methods*, 6(9):813–827, 1977.

[HYH13] M. Hodosh, P. Young, and J. Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal Of Artificial Intelligence Research*, 47:853–899, 2013.

[HZRS16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[HZSQ20] H. Huang, F. Zhong, Y. Sun, and X. Qin. An occlusion-aware edge-based method for monocular 3D object tracking using edge confidence. *Computer Graphics Forum*, 39(7):399–409, 2020.

[IMG22] M.T. Ibrahim, A. Majumder, and M. Gopi. Dynamic projection mapping on deformable stretchable materials using boundary tracking. *Computers & Graphics*, 103:61–74, 2022.

[IS15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2015.

[JB16] R. Jonschkowski and O. Brock. End-to-end learnable histogram filters. In *Workshop on Deep Learning for Action and Interaction at NIPS*, 2016.

[JGHE21] M. Janik, N. Gard, A. Hilsmann, and P. Eisert. Zero in on shape: A generic 2D-3D instance similarity metric learned from synthetic data. In *Proc. IEEE Int. Conf. on Image Processing (ICIP)*, 2021.

[JN21] J. Jacob and R. Nóbrega. Collaborative augmented reality for cultural heritage, tourist sites and museums: sharing visitors' experiences and interactions. In *Augmented reality in tourism, museums and heritage: A new technology to inform and entertain*. Springer, 2021.

[Jun20] A.B. Jung. imgaug. `https://github.com/aleju/imgaug`, 2020. [Online; accessed 26-March-2025].

[KB15] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2015.

[KC17] A. Kendall and R. Cipolla. Geometric loss functions for camera pose regression with deep learning. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[KCU$^+$22] D. Khan, Z. Cheng, H. Uchiyama, S. Ali, M. Asshad, and K. Kiyokawa. Recent advances in vision-based indoor navigation: A systematic literature review. *Computers & Graphics*, 104:24–45, 2022.

[KGC15] A. Kendall, M. Grimes, and R. Cipolla. PoseNet: A convolutional network for real-time 6-DoF camera relocalization. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2015.

[KGC18] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[KH14] D. Kobayashi and N. Hashimoto. Spatial augmented reality by using depth-based object tracking. In *Proc. ACM SIGGRAPH Posters*, 2014.

[KH19] S. Kagami and K. Hashimoto. Animated Stickies: Fast video projection mapping onto a markerless plane through a direct closed-loop alignment. *IEEE Transactions on Visualization and Computer Graphics*, 25(11):3094–3104, 2019.

[KHKA22] A.Z. Kolaei, E. Hedayati, M. Khanzadi, and G.G. Amiri. Challenges and opportunities of augmented reality during the construction phase. *Automation in Construction*, 143:104586, 2022.

[KHL18] P. Karkus, D. Hsu, and W.S. Lee. Particle filter networks with application to visual localization. In *Proc. Conf. on Robot Learning (CoRL)*, 2018.

[KK07] M. Kumru and P. Kılıcogulları. Process improvement through ergonomic design in welding shop of an automotive factory. In *10th QMOD Conference. Quality Management and Organiqatinal Development. Our Dreams of Excellence*, 2007.

[KKH15] R. Koizumi, D. Kobayashi, and N. Hashimoto. Acceleration of dynamic spatial augmented reality system with a depth camera. In *Proc. Int. Conf. on Cyberworlds (CW)*, 2015.

[KKLD23] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):139–1, 2023.

[KMR$^+$23] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A.C. Berg, W.Y. Lo, P. Dollár, and R. Girshick. Segment Anything. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2023.

[KP02] S.G. Krantz and H.R. Parks. *The implicit function theorem: history, theory, and applications.* Springer Science & Business Media, 2002.

[Kun55] T. Kunin. The construction of a new type of attitude measure. *Personnel Psychology*, 8(1):65–77, 1955.

[KWW17] J. Kern, M. Weinmann, and S. Wursthorn. Projector-based augmented reality for quality inspection of scanned objects. *International Archives Of Photogrammetry, Remote Sensing And Spatial Information Sciences*, 4(2):83–90, 2017.

[KZSI19] R. Kaskman, S. Zakharov, I. Shugurov, and S. Ilic. HomebrewedDB: RGB-D dataset for 6D pose estimation of 3D objects. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV) Workshops*, 2019.

[LADL18] T.M. Li, M. Aittala, F. Durand, and J. Lehtinen. Differentiable Monte Carlo ray tracing through edge sampling. *ACM Transactions on Graphics*, 37(6):222:1–222:11, 2018.

[LB14] M.M. Loper and M.J. Black. OpenDR: An approximate differentiable renderer. In *Proc. European Conf. on Computer Vision (ECCV) Workshops*, 2014.

[LCAS20] Y. Labbe, J. Carpentier, M. Aubry, and J. Sivic. CosyPose: Consistent multi-view multi-object 6D pose estimation. In *Proc. European Conf. on Computer Vision (ECCV) Workshops*, 2020.

[LGG+17] T.Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2017.

[LGH22] Y. Liu, W. Gao, and Z. Hu. A large-scale dataset for indoor visual localization with high-precision ground truth. *The International Journal Of Robotics Research*, 41(2):129–135, 2022.

[LHSJ21] Z. Li, Y. Hu, M. Salzmann, and X. Ji. SD-Pose: Semantic decomposition for cross-domain 6D object pose estimation. In *Proc. AAAI Conf. on Artificial Intelligence (AAAI)*, 2021.

[LK81] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI'81: 7th international joint conference on Artificial intelligence*, volume 2, pages 674–679, 1981.

[LLD17] L. Liu, H. Li, and Y. Dai. Efficient global 2D-3D matching for camera localization in a large-scale 3D map. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[LLM+18] R. Liu, J. Lehman, P. Molino, F.P. Such, E. Frank, A. Sergeev, and J. Yosinski. An intriguing failing of convolutional neural networks and the CoordConv solution. In *Proc. Neural Information Processing Systems (NeurIPS)*, 2018.

[LMNF09] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An accurate O(n) solution to the PnP problem. *International Journal Of Computer Vision*, 81(2):155–166, 2009.

[Low04] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal Of Computer Vision*, 60:91–110, 2004.

[LRS+18] G. Liu, F.A. Reda, K.J. Shih, T.C. Wang, A. Tao, and B. Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proc. European Conf. on Computer Vision (ECCV) Workshops*, 2018.

[LSK+15] C. Liu, A. Schwing, K. Kundu, R. Urtasun, and S. Fidler. Rent3D: Floor-plan priors for monocular layout estimation. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[LSR$^+$20] H. Liu, Y. Su, J. Rambach, A. Pagani, and D. Stricker. TGA: Two-level group attention for assembly state detection. In *Proc. IEEE Int. Symposium on Mixed and Augmented Reality (ISMAR)*, 2020.

[LSW$^+$18] G. Liu, K.J. Shih, T.C. Wang, F.A. Reda, K. Sapra, Z. Yu, A. Tao, and B. Catanzaro. Partial convolution based padding. *arXiv preprint arXiv:1811.11718*, 2018.

[LSY$^+$24] J. Liu, W. Sun, H. Yang, Z. Zeng, C. Liu, J. Zheng, X. Liu, H. Rahmani, N. Sebe, and A. Mian. Deep learning-based object pose estimation: A comprehensive survey. *arXiv preprint arXiv:2405.07801*, 2024.

[LTGD22] L. Lipson, Z. Teed, A. Goyal, and J. Deng. Coupled iterative refinement for 6D multi-object pose estimation. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[LWJ$^+$18] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox. DeepIM: Deep iterative matching for 6D pose estimation. In *Proc. European Conf. on Computer Vision (ECCV) Workshops*, 2018.

[LWJ19] Z. Li, G. Wang, and X. Ji. CDPN: Coordinates-based disentangled pose network for real-time RGB-based 6-DoF object pose estimation. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2019.

[LWZ$^+$22] J. Li, B. Wang, S. Zhu, X. Cao, F. Zhong, W. Chen, T. Li, J. Gu, and X. Qin. BCOT: A markerless high-precision 3D object tracking benchmark. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[LZR$^+$23] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, et al. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In *Proc. European Conf. on Computer Vision (ECCV)*, 2023.

[LZYZ21] X. Lv, S. Zhao, X. Yu, and B. Zhao. Residential floor plan recognition and reconstruction. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[MAR$^+$19] F. Manhardt, D.M. Arroyo, C. Rupprecht, B. Busam, T. Birdal, N. Navab, and F. Tombari. Explaining the ambiguity of object detection and 6D pose from visual data. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2019.

[Maz18] D. Mazzini. Guided upsampling network for real-time semantic segmentation. In *Proc. British Machine Vision Conf. (BMVC)*, 2018.

[ME20] T. Masood and J. Egger. Adopting augmented reality in the age of industrial digitalisation. *Computers In Industry*, 115:103112, 2020.

[MGB$^+$23] W. Morgenstern, N. Gard, S. Baumann, A. Hilsmann, and P. Eisert. X-maps: Direct depth lookup for event-based structured light systems. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2023.

[MGDRGLC$^+$20] J. Martin-Gutierrez, M.S. Del Rio Guerra, V. Lopez-Chao, R.H. Soto Gastelum, and J.F. Valenzuela Bojórquez. Augmented reality to facilitate learning of the acoustic guitar. *Applied Sciences*, 10(7):2425, 2020.

[MGGL21] J.P. Mercier, M. Garon, P. Giguere, and J.F. Lalonde. Deep template-based object instance detection. In *Proc. IEEE Winter Conf. on Applications of Computer Vision (WACV)*, 2021.

[MKB$^+$22] Z. Min, N. Khosravan, Z. Bessinger, M. Narayana, S.B. Kang, E. Dunn, and I. Boyadzhiev. LASER: Latent space rendering for 2D visual localization. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[MPG$^+$22] A.S. Madhvacharyula, A.V.S. Pavan, S. Gorthi, S. Chitral, N. Venkaiah, and D.V. Kiran. In situ detection of welding defects: A review. *Welding in the World*, 66(4):611–628, 2022.

[MRTMFH$^+$23] C.E. Mendoza-Ramírez, J.C. Tudon-Martinez, L.C. Félix-Herrán, J.d.J. Lozoya-Santos, and A. Vargas-Martínez. Augmented Reality: Survey. *Applied Sciences*, 13(18):10491, 2023.

[MSWT14] M.R. Marner, R.T. Smith, J.A. Walsh, and B.H. Thomas. Spatial user interfaces for large-scale projector-based augmented reality. *IEEE Computer Graphics and Applications*, 34(6):74–82, 2014.

[MT12] D. Moreno and G. Taubin. Simple, accurate, and robust projector-camera calibration. In *Proc. Int. Conf. on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, 2012.

[MYY22] I. Majil, M.T. Yang, and S. Yang. Augmented reality based interactive cooking guide. *Sensors*, 22(21):8290, 2022.

[NB17] T. Naseer and W. Burgard. Deep regression for monocular camera-based 6-DoF global localization in outdoor environments. In *Proc. Int. Conf. on Intelligent Robot Systems (IROS)*, 2017.

[NGP+23] V.N. Nguyen, T. Groueix, G. Ponimatkin, V. Lepetit, and T. Hodaň. CNOS: A strong baseline for CAD-based novel object segmentation. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV) Workshops*, 2023.

[NGSL24] V.N. Nguyen, T. Groueix, M. Salzmann, and V. Lepetit. GigaPose: Fast and robust novel object pose estimation via one correspondence. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[NHB11] J. Noruk, B. Holmes, and B. Bruss. Laser tool offers alternative for precise visual weld inspection. *Inspection Trends*, 14(4), 2011.

[NWI17] G. Narita, Y. Watanabe, and M. Ishikawa. Dynamic projection mapping onto deforming non-rigid surface using deformable dot cluster marker. *IEEE Transactions on Visualization and Computer Graphics*, 23(3):1235–1248, 2017.

[OB21] S. Orhan and Y. Baştanlar. Efficient search in a panoramic image database for long-term visual localization. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[ODM+23] M. Oquab, T. Darcet, T. Moutakanni, H.V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, R. Howes, P.Y. Huang, H. Xu, V. Sharma, S.W. Li, W. Galuba, M. Rabbat, M. Assran, N. Ballas, G. Synnaeve, I. Misra, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[OGB22] S. Orhan, J.J. Guerrero, and Y. Baştanlar. Semantic pose verification for outdoor visual localization with self-supervised contrastive learning. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2022.

[ÖLT+24] E.P. Örnek, Y. Labbé, B. Tekin, L. Ma, C. Keskin, C. Forster, and T. Hodaň. FoundPose: Unseen object pose estimation with foundation features. In *Proc. European Conf. on Computer Vision (ECCV)*, 2024.

[PB24] A.S. Periyasamy and S. Behnke. MOTPose: Multi-object 6D pose estimation for dynamic video sequences using attention-based temporal fusion. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2024.

[PHSS22] P.N. Pizarro, N. Hitschfeld, I. Sipiran, and J.M. Saavedra. Automatic floor plan analysis and recognition. *Automation in Construction*, 140, 2022.

[PIL19] G. Pitteri, S. Ilic, and V. Lepetit. CorNet: Generic 3D corners for 6D pose estimation of new objects without retraining. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2019.

[PJM22] J. Park, D. Jung, and B. Moon. Projector compensation framework using differentiable rendering. *IEEE Access*, 10:44461–44470, 2022.

[PK04] J. Park and Y.B. Kwon. Main wall recognition of architectural drawings using dimension extension line. In *Graphics Recognition. Recent Advances and Perspectives: 5th International Workshop, GREC 2003, Barcelona, Spain, July 30-31, 2003, Revised Selected Papers 5*, 2004.

[PK21] S. Park and H. Kim. 3DPlanNet: Generating 3D models from 2D floor plan images using ensemble methods. *Electronics*, 10(22):2729, 2021.

[PLH+19] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao. PVNet: Pixel-wise voting network for 6DoF pose estimation. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[PLS+15] M.K. Park, K.J. Lim, M.K. Seo, S.J. Jung, and K.H. Lee. Spatial augmented reality for product appearance design evaluation. *Journal Of Computational Design And Engineering*, 2(1):38–46, 2015.

[PLWZ19] T. Park, M. Liu, T. Wang, and J. Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[PMXF20] K. Park, A. Mousavian, Y. Xiang, and D. Fox. LatentFusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[PPDD16] T. Petković, T. Pribanic, M. Donlić, and N. D'Apuzzo. Software synchronization of projector and camera for structured light 3D body scanning. In *Proc. 3D Body Scanning Technologies (3DBST)*, 2016.

[PPV19] K. Park, T. Patten, and M. Vincze. Pix2Pose: Pixel-wise coordinate regression of objects for 6D pose estimation. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2019.

[PR12] V.A. Prisacariu and I.D. Reid. PWP3D: Real-time segmentation and tracking of 3D objects. *International Journal Of Computer Vision*, 98(3):335–354, 2012.

[PRFG23] T. Pöllabauer, F. Rücker, A. Franek, and F. Gorschlüter. Detection and pose estimation of flat, texture-less industry objects on HoloLens using synthetic training. In *Proc.Scandinavian Conference on Image Analysis (SCIA)*, 2023.

[PRH$^+$22] D. Przewozny, D. Runde, N. Hoerner, N. Gard, S. Bosse, and P. Chojecki. Comparison of uni- and multimodal interfaces for spatial interaction. In *Proc. ACM Symposium on Spatial User Interaction (SUI)*, 2022.

[RBW07] B. Rosenhahn, T. Brox, and J. Weickert. Three-dimensional shape knowledge for joint image segmentation and pose tracking. *International Journal Of Computer Vision*, 73:243–262, 2007.

[RKK16] C. Resch, P. Keitler, and G. Klinker. Sticky Projections-A model-based approach to interactive shader lamps tracking. *IEEE Transactions on Visualization and Computer Graphics*, 22(3):1291–1301, 2016.

[RL17] M. Rad and V. Lepetit. BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2017.

[RLL$^+$04] L.M. Reeves, J. Lai, J.A. Larson, S. Oviatt, T. Balaji, S. Buisine, P. Collings, P. Cohen, B. Kraal, J.C. Martin, et al. Guidelines for multimodal user interface design. *Communications of the ACM*, 47(1):57–59, 2004.

[RNK$^+$15] C. Resch, H. Naik, P. Keitler, S. Benkhardt, and G. Klinker. On-site semi-automatic calibration and registration of a projector-camera system using arbitrary objects with known geometry. *IEEE Transactions on Visualization and Computer Graphics*, 21(11):1211–1220, 2015.

[RRN$^+$20] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.Y. Lo, J. Johnson, and G. Gkioxari. Accelerating 3D deep learning with PyTorch3D. *arXiv preprint arXiv:2007.08501*, 2020.

[RVB18] N. Radwan, A. Valada, and W. Burgard. VLocNet++: Deep multitask learning for semantic visual localization and odometry. *IEEE Robotics And Automation Letters*, 3(4):4407–4414, 2018.

[RvBB$^+$03] R. Raskar, J. van Baar, P. Beardsley, T. Willwacher, S. Rao, and C. Forlines. iLamps: Geometrically aware and self-configuring projectors. In *Proc. ACM SIGGRAPH Conf. on Computer Graphics and Interactive Techniques*, 2003.

[RWLB01] R. Raskar, G. Welch, K. Low, and D. Bandyopadhyay. Shader Lamps: Animating real objects with image-based illumination. In *Proc. 12th Eurographics Workshop on Rendering*, 2001.

[SBK19] K. Sofiiuk, O. Barinova, and A. Konushin. AdaptIS: Adaptive instance selection network. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2019.

[SCA$^+$20] J. Sock, P. Castro, A. Armagan, G. Garcia-Hernando, and T.K. Kim. Tackling two challenges of 6D object pose estimation: Lack of real annotated RGB images and scalability to number of objects. *arXiv preprint arXiv:2003.12344v1*, 2020.

[SEB$^+$17] T. Spitzer, H.J. Eder, C. Bauchinger, I. Volkhausen, S. Peters, S. Engelsmann, C. Dengler, et al. Freeform grid-shell for Chadstone Shopping Centre, Melbourne. In *Proc. IASS Annual Symposia: IASS 2017 Hamburg Symposium*, 2017.

[see21] seele GmbH. DigitalTWIN | Demo | Live-Interaktion bei der Planung und Ausführung einer Gitterschalenkonstruktion. `https://www.youtube.com/watch?v=247inqcfhH4`, 2021. [Online; accessed 26-March-2025].

[SEG01] E. Steinbach, P. Eisert, and B. Girod. Model-based 3-D shape and motion estimation using sliding textures. In *Proc. Vision, Modeling, and Visualization (VMV)*, 2001.

[Sen15] Senatorin für Bau, Mobilität und Stadtentwicklung. Bremens Bauaktenarchiv wird komplett digitalisiert. `https://www.senatspressestelle.bremen.de/pressemitteilungen/bremens-bauaktenarchiv-wird-komplett-digitalisiert-135570`, 2015. [Online; accessed 26-March-2025].

[SF$^+$68] I. Sobel, G. Feldman, et al. A 3×3 isotropic gradient operator for image processing. *a talk at the Stanford Artificial Project*, pages 271–272, 1968.

[SHE17] C. Seibold, A. Hilsmann, and P. Eisert. Model-based motion blur estimation for the improvement of motion tracking. *Computer Vision And Image Understanding*, 160:45–56, 2017.

[SHL+23] M. Sundermeyer, T. Hodaň, Y. Labbé, G. Wang, E. Brachmann, B. Drost, C. Rother, and J. Matas. BOP Challenge 2022 on detection, segmentation and pose estimation of specific rigid objects. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 2785–2794, 2023.

[SHZ+18] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.C. Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[SKT19] F. Schmid, P. Kopriwa, and M. Teich. DigitalTWIN improving future gridshell installations. In *Proc. IASS Annual Symposia: IASS 2019 Barcelona Symposium*, 2019.

[SMD+18] M. Sundermeyer, Z.C. Marton, M. Durner, M. Brucker, and R. Triebel. Implicit 3D orientation learning for 6D object detection from RGB images. In *Proc. European Conf. on Computer Vision (ECCV)*, 2018.

[SMTT18] F. Schmid, S. Marinitsch, M. Teich, and C. Timm. Metal and glass grid-shell design: Flexible integration of digital design and fabrication tools. In *Proc. IASS Annual Symposia: IASS 2018 Boston Symposium*, 2018.

[Sol17] J. Sola. Quaternion kinematics for the error-state Kalman filter. *arXiv preprint arXiv:1711.02508*, 2017.

[SP22] S.R. Syu and C.H. Peng. Optimizing placements of 360-degree panoramic cameras in indoor environments by integer programming. *arXiv preprint arXiv:2211.07296*, 2022.

[SPP+13] B.K. Seo, H. Park, J.I. Park, S. Hinterstoisser, and S. Ilic. Optimal local searching for fast and robust textureless 3D object tracking in highly cluttered backgrounds. *IEEE Transactions on Visualization and Computer Graphics*, 20(1):99–110, 2013.

[SPS+20] M. Stoiber, M. Pfanne, K.H. Strobl, R. Triebel, and A. Albu-Schaeffer. A sparse Gaussian approach to region-based 6DoF object tracking. In *Proc. Asian Conf. on Computer Vision (ACCV)*, 2020.

[SPS+22] M. Stoiber, M. Pfanne, K.H. Strobl, R. Triebel, and A. Albu-Schäffer. SRT3D: A sparse region-based 3D object tracking approach for the real world. *International Journal Of Computer Vision*, 130(4):1008–1030, 2022.

[SRB10] D. Sun, S. Roth, and M.J. Black. Secrets of optical flow estimation and their principles. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[SRM$^+$19] Y. Su, J. Rambach, N. Minaskan, P. Lesur, A. Pagani, and D. Stricker. Deep multi-state object pose estimation for augmented reality assembly. In *Proc. IEEE Int. Symposium on Mixed and Augmented Reality (ISMAR)*, 2019.

[SSF$^+$22] Y. Su, M. Saleh, T. Fetzer, J. Rambach, N. Navab, B. Busam, D. Stricker, and F. Tombari. ZebraPose: Coarse to fine surface encoding for 6DoF object pose estimation. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[SSH20] C. Song, J. Song, and Q. Huang. HybridPose: 6D object pose estimation under hybrid representations. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[SSM01] R. Sukthankar, R.G. Stockton, and M.D. Mullin. Smarter presentations: Exploiting homography in camera-projector systems. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2001.

[SSY$^+$23] Z. Shi, H. Shi, K. Yang, Z. Yin, Y. Lin, and K. Wang. PanoVPR: Towards unified perspective-to-equirectangular visual place recognition via sliding windows across the panoramic view. In *Proc. IEEE 26th International Conf. on Intelligent Transportation Systems (ITSC)*, 2023.

[Stu02] P. Sturm. Mixing catadioptric and perspective cameras. In *Proceedings of the IEEE Workshop on Omnidirectional Vision 2002. Held in conjunction with ECCV'02*, 2002.

[SUL$^+$21] P.E. Sarlin, A. Unagar, M. Larsson, H. Germain, C. Toft, V. Larsson, M. Pollefeys, V. Lepetit, L. Hammarstrand, F. Kahl, et al. Back to the feature: Learning robust camera localization from pixels to pose. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[SXL$^+$21] B. Shen, F. Xia, C. Li, R. Martín-Martín, L. Fan, G. Wang, C. Pérez-D'Arpino, S. Buch, S. Srivastava, L. Tchapmi, et al. iGibson 1.0: A simulation environment for interactive tasks in large realistic scenes. In *Proc. Int. Conf. on Intelligent Robot Systems (IROS)*, 2021.

[SZZ$^+$21] X. Sun, J. Zhou, W. Zhang, Z. Wang, and Q. Yu. Robust monocular pose tracking of less-distinct objects based on contour-part model.

IEEE Transactions on Circuits and Systems for Video Technology, 31(11):4409–4421, 2021.

[Tai24] R. Tait. Ultimate gaming projector comparison. `https://www.thesmarthomehookup.com/ultimate-gaming-projector-comparison/`, 2024. The Hook Up - Smart Home Automation, [Online; accessed 26-March-2025].

[TAS+15] A. Torii, R. Arandjelovic, J. Sivic, M. Okutomi, and T. Pajdla. 24/7 place recognition by view synthesis. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[TBS+21] M.Ö. Türkoğlu, E. Brachmann, K. Schindler, G. Brostow, and A. Monszpart. Visual camera re-localization using graph neural networks and relative pose supervision. In *Proc. Int. Conf. on 3D Vision (3DV)*, 2021.

[TCC+21] Z. Tan, D. Chen, Q. Chu, M. Chai, J. Liao, M. He, L. Yuan, G. Hua, and N. Yu. Efficient semantic image synthesis via class-adaptive normalization. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 2021.

[TFR+17] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Proc. Int. Conf. on Intelligent Robot Systems (IROS)*, 2017.

[TL19] M. Tan and Q. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2019.

[TLPV21] S. Thalhammer, M. Leitner, T. Patten, and M. Vincze. PyraPose: Feature pyramids for fast and accurate object pose estimation under domain shift. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2021.

[TLZQ22] X. Tian, X. Lin, F. Zhong, and X. Qin. Large-displacement 3D object tracking with hybrid non-local optimization. In *Proc. European Conf. on Computer Vision (ECCV)*, 2022.

[TNT18] K. Tateno, N. Navab, and F. Tombari. Distortion-aware convolutional filters for dense prediction in panoramic images. In *Proc. European Conf. on Computer Vision (ECCV)*, 2018.

[TOS+18] H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla, and A. Torii. InLoc: Indoor visual localization with dense matching and view synthesis. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[TPV23] S. Thalhammer, T. Patten, and M. Vincze. COPE: End-to-end trainable constant runtime object pose estimation. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[Tra13] J. Traa. Least-squares intersection of lines. `http://web.archive.org/web/20190427061114/http://cal.cs.illinois.edu/~johannes/research/LS_line_intersect.pdf`, 2013. University of Illinois Urbana-Champaign (UIUC), [Online; accessed 26-March-2025].

[TSSC18] H. Tjaden, U. Schwanecke, E. Schömer, and D. Cremers. A region-based Gauss-Newton approach to real-time monocular multiple object tracking. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 41(8):1797–1812, 2018.

[TTB18] J. Tremblay, T. To, and S. Birchfield. Falling Things: A synthetic dataset for 3D object detection and pose estimation. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[TTM+18] T. To, J. Tremblay, D. McKay, Y. Yamaguchi, K. Leung, A. Balanon, J. Cheng, W. Hodge, and S. Birchfield. NDDS: NVIDIA deep learning dataset synthesizer. `https://github.com/NVIDIA/Dataset_Synthesizer`, 2018. [Online; accessed 26-March-2025].

[TTS+18] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. In *Proc. Conf. on Robot Learning (CoRL)*, 2018.

[UGM+18] A.E. Uva, M. Gattullo, V.M. Manghisi, D. Spagnulo, G.L. Cascella, and M. Fiorentino. Evaluating the effectiveness of spatial augmented reality in smart manufacturing: a solution for manual working stations. *The International Journal Of Advanced Manufacturing Technology*, 94:509–521, 2018.

[USS+17] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger. Sparsity invariant CNNs. In *Proc. Int. Conf. on 3D Vision (3DV)*, 2017.

[UVL16] D. Ulyanov, A. Vedaldi, and V.S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[VDMDDFW00] R. Van Der Merwe, A. Doucet, N. De Freitas, and E. Wan. The unscented particle filter. *Advances In Neural Information Processing Systems*, 13, 2000.

[VRCP17] R. Vassallo, A. Rankin, E.C. Chen, and T.M. Peters. Hologram stability evaluation for Microsoft HoloLens. In *Proc. SPIE Volume 10136*, 2017.

[VSLP24] A. Veicht, P.E. Sarlin, P. Lindenberger, and M. Pollefeys. GeoCalib: Single-image calibration with geometric optimization. In *Proc. European Conf. on Computer Vision (ECCV)*, 2024.

[VW22] T. Vorbeck and N. Wills. The current state of BIM on existing buildings: The case of Germany. `https://www.thm.de/wi/images/vXORD7JO.pdf`, 2022. [Online; accessed 26-March-2025].

[WBB05] J. Wolf, W. Burgard, and H. Burkhardt. Robust vision-based localization by combining an image-retrieval system with Monte Carlo localization. *IEEE Robotics And Automation Letters*, 21(2):208–216, 2005.

[WBL23] C.Y. Wang, A. Bochkovskiy, and H.Y.M. Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[WCX$^+$19] J. Wang, K. Chen, R. Xu, Z. Liu, C.C. Loy, and D. Lin. CARAFE: Content-aware reassembly of features. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2019.

[WFU15] S. Wang, S. Fidler, and R. Urtasun. Lost Shopping! Monocular localization in large indoor spaces. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2015.

[WKS$^+$20] X. Wang, T. Kong, C. Shen, Y. Jiang, and L. Li. SOLO: Segmenting objects by locations. In *Proc. European Conf. on Computer Vision (ECCV)*, 2020.

[WLH23] Y. Wang, H. Ling, and B. Huang. CompenHR: Efficient full compensation for high-resolution projector. In *Proc. IEEE Conf. on Virtual Reality and 3D User Interfaces (VR)*, 2023.

[WLT$^+$17] P.C. Wu, Y.Y. Lee, H.Y. Tseng, H.I. Ho, M.H. Yang, and S.Y. Chien. [POSTER] A benchmark dataset for 6DoF object pose tracking. In *Proc. IEEE Int. Symposium on Mixed and Augmented Reality (ISMAR)*, 2017.

[WML$^+$21] G. Wang, F. Manhardt, X. Liu, X. Ji, and F. Tombari. Occlusion-aware self-supervised monocular 6D object pose estimation. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 46(3):1788–1803, 2021.

[WMR24] C. Wu, Z. Montazeri, and T. Ritschel. Learning to rasterize differentiably. *Computer Graphics Forum*, 43(4):e15145, 2024.

[WMS$^+$20] G. Wang, F. Manhardt, J. Shao, X. Ji, N. Navab, and F. Tombari. Self6D: Self-supervised monocular 6D object pose estimation. In *Proc. European Conf. on Computer Vision (ECCV)*, 2020.

[WPYW20] Y. Wen, H. Pan, L. Yang, and W. Wang. Edge enhanced implicit orientation learning with geometric prior for 6D pose estimation. *IEEE Robotics And Automation Letters*, 5(3):4931–4938, 2020.

[WWZ$^+$15] G. Wang, B. Wang, F. Zhong, X. Qin, and B. Chen. Global optimal searching for textureless 3D object tracking. *The Visual Computer*, 31(6):979–988, 2015.

[WYZ$^+$23] L. Wang, S. Yan, J. Zhen, Y. Liu, M. Zhang, G. Zhang, and X. Zhou. Deep active contours for real-time 6-DoF object tracking. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2023.

[WZQ19] B. Wang, F. Zhong, and X. Qin. Robust edge-based 3D object tracking with direction-based pose validation. *Multimedia Tools And Applications*, 78(9):12307–12331, 2019.

[XG22] J. Xia and J. Gong. Precise indoor localization with 3D facility scan data. *Computer-Aided Civil And Infrastructure Engineering*, 37(10):1243–1259, 2022.

[XSNF18] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. In *Proc. Robotics: Science and Systems (RSS)*, 2018.

[YCL$^+$23] X. Yang, J. Cai, K. Li, X. Fan, and H. Cao. A monocular-based tracking framework for industrial augmented reality applications. *The International Journal Of Advanced Manufacturing Technology*, 128(5-6):2571–2588, 2023.

[YLY$^+$19] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T.S. Huang. Free-form image inpainting with gated convolution. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2019.

[YUS$^+$09] N. Yazawa, H. Uchiyama, H. Saito, M. Servieres, and G. Moreau. Image based view localization system retrieving from a panorama database by SURF. In *Proc. Int. Conf. on Machine Vision Applications (MVA)*, 2009.

[YYY21] Z. Yang, X. Yu, and Y. Yang. DSC-PoseNet: Learning 6DoF object pose estimation via dual-scale consistency. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[YZKL20] X. Yu, Z. Zhuang, P. Koniusz, and H. Li. 6DoF object pose estimation via differentiable proxy voting regularizer. In *Proc. British Machine Vision Conf. (BMVC)*, 2020.

[ZAA19] J. Zubizarreta, I. Aguinaga, and A. Amundarain. A framework for augmented reality guidance in industry. *The International Journal Of Advanced Manufacturing Technology*, 102(9):4095–4108, 2019.

[ZAQW20] P. Zhu, R. Abdal, Y. Qin, and P. Wonka. SEAN: Image synthesis with semantic region-adaptive normalization. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[ZBLC21] C. Zhang, I. Budvytis, S. Liwicki, and R. Cipolla. Lifted semantic graph embedding for omnidirectional place recognition. In *Proc. Int. Conf. on 3D Vision (3DV)*, 2021.

[ZH87] O.A. Zuniga and R.M. Haralick. Integrated directional derivative gradient operator. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(3):508–517, 1987.

[Zha97] Z. Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image And Vision Computing*, 15(1):59–76, 1997.

[ZLZ17] L. Zhong, M. Lu, and L. Zhang. A direct 3D object tracking method based on dynamic textured model rendering and extended dense feature fields. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(9):2302–2315, 2017.

[ZS10] A.R. Zamir and M. Shah. Accurate image localization based on Google maps Street View. In *Proc. European Conf. on Computer Vision (ECCV)*, 2010.

[ZSI19] S. Zakharov, I. Shugurov, and S. Ilic. DPOD: 6D pose object detector and refiner. In *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2019.

[ZSW13] F. Zheng, R. Schubert, and G. Weich. A general approach for closed-loop registration in AR. In *Proc. IEEE Int. Symposium on Mixed and Augmented Reality (ISMAR)*, 2013.

[ZZ19] L. Zhong and L. Zhang. A robust monocular 3D object tracking method combining statistical and photometric constraints. *International Journal Of Computer Vision*, 127(8):973–992, 2019.

[ZZG$^+$21] S. Zhang, W. Zhao, Z. Guan, X. Peng, and J. Peng. Keypoint-graph-driven learning framework for object pose estimation. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[ZZL$^+$20] J. Zheng, J. Zhang, J. Li, R. Tang, S. Gao, and Z. Zhou. Structured3D: A large photo-realistic dataset for structured 3D modeling. In *Proc. European Conf. on Computer Vision (ECCV)*, 2020.

Selbständigkeitserklärung

Ich erkläre, dass ich die Dissertation selbständig und nur unter Verwendung der von mir gemäß §7 Abs. 3 der Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät, veröffentlicht im Amtlichen Mitteilungsblatt der Humboldt-Universität zu Berlin Nr. 42/2018 am 11.07.2018 angegebenen Hilfsmittel angefertigt habe.

Berlin, den 28. März 2025 Niklas Gard