Estimating Position & Velocity in 3D Space from Monocular Video Sequences using a Deep Neural Network

Arturo Marban
Universitat Politécnica de Catalunya
08034 Barcelona, Spain
arturo.marban@upc.edu

Vignesh Srinivasan
Fraunhofer Heinrich Hertz Institute
10587 Berlin, Germany
vignesh.srinivasan@hhi.fraunhofer.de

Wojciech Samek
Fraunhofer Heinrich Hertz Institute
10587 Berlin, Germany
wojciech.samek@hhi.fraunhofer.de

Josep Fernández
Universitat Politécnica de Catalunya
08034 Barcelona, Spain
josep.fernandez@upc.edu

Alicia Casals
Universitat Politécnica de Catalunya
08034 Barcelona, Spain
alicia.casals@upc.edu

Abstract

This work describes a regression model based on Convolutional Neural Networks (CNN) and Long-Short Term Memory (LSTM) networks for tracking objects from monocular video sequences. The target application being pursued is Vision-Based Sensor Substitution (VBSS). In particular, the tool-tip position and velocity in 3D space of a pair of surgical robotic instruments (SRI) are estimated for three surgical tasks, namely suturing, needle-passing and knot-tying. The CNN extracts features from individual video frames and the LSTM network processes these features over time and continuously outputs a 12-dimensional vector with the estimated position and velocity values. A series of analyses and experiments are carried out in the regression model to reveal the benefits and drawbacks of different design choices. First, the impact of the loss function is investigated by adequately weighing the Root Mean Squared Error (RMSE) and Gradient Difference Loss (GDL), using the VGG16 neural network for feature extraction. Second, this analysis is extended to a Residual Neural Network designed for feature extraction, which has fewer parameters than the VGG16 model, resulting in a reduction of ∼96.44 % in the neural network size. Third, the impact of the number of time steps used to model the temporal information processed by the LSTM network is investigated. Finally, the capability of the regression model to generalize to the data related to "unseen" surgical tasks (unavailable in the training set) is evaluated. The aforesaid analyses are experimentally validated on the public dataset JIGSAWS. These analyses provide some guidelines for the design of a regression model in the context of VBSS, specifically when the objective is to estimate a set of 1D time series signals from video sequences.

1. Introduction

Embedding sensors in the instruments represents the most straightforward method for an accurate measurement of a physical variable, such as position or velocity. Nonetheless, for some applications, this approach is easier to implement in an experimental setup (i.e. in the laboratory) than in a real world scenario, as in Minimally Invasive Surgery (MIS). An alternative method, when sensor integration is not possible, is to rely on a model that implements the concept of Vision-Based Sensor Substitution (VBSS) [1] by processing video sequences recorded by a camera (monocular or stereo). In the context of Robotic Assisted Minimally Invasive Surgery (RAMIS), and specifically in the modeling of gestures and in skills assessment, this approach can be beneficial. In RAMIS, Surgical Robotic Instruments (SRI) mounted at the end-effector of slave robot manipulators, are teleoperated from a master console by a surgeon or trainee. The motion of the SRIs' tool-tip is highly correlated with surgical gestures (i.e. suturing). Therefore, an important step to perform surgical gesture classification strictly under

a vision-based approach relies on the detection and localization of SRIs from video sequences [2]. Subsequently, an action recognition model can take advantage of this information to automatically classify surgical gestures (i.e. suturing, knot-tying and needle-passing) from video data, as described in [1]. Recent advances in Deep Learning (DL) have shown that complex functions can be learned from data using a Deep Neural Network (DNN) in a supervised setting. Two of the most successful DNNs applied in the processing of data with spatial and temporal structure are Convolutional Neural Networks (CNN) and Long-Short Term Memory (LSTM) networks, respectively. Not only have CNNs obtained state-of-the-art results in image classification [3][4], but they have also shown superior results in transfer learning tasks. A CNN trained for image classification can also be used as an off-the-shelf feature extractor for a different task, which upon fine-tuning improves generalization performance [5][6]. In a similar note, LSTM networks highlight in the processing of sequences of data [7]. Video sequences can be interpreted as containing both spatial and temporal information. Therefore, a model which processes such data has to integrate both CNNs and LSTM networks in its design. In the present work, VBSS is investigated using a regression model that estimates the tool-tip position and velocity (both in 3D space) of a pair of SRIs from monocular video sequences. The requirement of processing velocity imposes an additional difficulty to the problem, making more evident the need for a model capable of processing spatiotemporal information. This complex relationship between video sequences and tool-tip position and velocity is learned from data by a regression model consisting of a CNN serially connected with an LSTM network.

1.1. Related Works

A regression model based on DNNs for processing video data should take into account its spatiotemporal structure. In domains such as action recognition, this is essential. For instance, a two stream CNN that processes RGB frames (spatial information) and a RGB representation of the optical flow (temporal information) is presented in [8]. In contrast, a 3D CNN designed with 3D filters (of size $3 \times 3 \times 3$) was designed for learning spatiotemporal features directly from RGB frames in [9]. The first model that integrates 3D CNN connected in series with an LSTM network was proposed in [10]. However, this model was validated only in a small dataset. More recently, different DNNs based on CNNs and LSTM networks have been designed and evaluated in larger and more complex datasets for action recognition, as described in [11] and [12].

DNNs have been applied to regression tasks in different domains. Human pose estimation from images has been investigated in [13]. In this work, a CNN was trained to regress upper joints' position using the standard $L2$ loss. A different approach is described in [14], by integrating a cascade of multiple CNNs in a single model. This model predicts an initial human pose from a full image and subsequently refines joint predictions by using higher resolution sub-images. The same application has been addressed in [15], with a robust function that avoids the influence of outliers during the training process. Nevertheless, [13]-[15] only consider the processing of spatial information (i.e. images), discarding the processing of temporal information. In contrast, the spatiotemporal structure of video data was taken into account in [16] by using a recurrent convolutional neural network. In this work, the authors address the problem of continuous shoulder pain intensity estimation from video sequences of human face expressions. Likewise, due to the sequential nature of video and audio data, [17] proposed a CNN connected in series with an LSTM network to estimate sound from silent video sequences.

In the medical domain, applications of DNNs to regression tasks are less common. In this regard, [1] introduces a CNN architecture for learning SRIs' position in 3D space from monocular video sequences. The estimated position values are subsequently used to feed an action recognition model based on a Latent Convolutional Skip Chain Conditional Random Field (LC-SC-CRF). Recently, [2] addressed the detection and localization of SRIs with a Region Proposal Network (RPN). This neural network operates based on a multi-modal framework, using two separate CNN streams for processing raw and optical flow video frames (both represented in RGB color space). Furthermore, the authors in [2] pointed out that a DNN approach is superior to the conventional hand-crafted feature based approaches in terms of precision and real-time requirements. However, the models described in [1] and [2], do not address the estimation of the tool-tip velocity in 3D space for each SRI. Moreover, the LSTM network is not considered in the regression model design.

1.2. Contributions

In the present work, a regression model that implements the concept of VBSS is investigated. This model estimates the tool-tip position and velocity of a pair of SRIs. These variables are computed in the 3D space from only monocular video sequences describing the motion of two SRIs. The position and velocity data are estimated for three surgical tasks, namely suturing, needle-passing and knot-tying. Although the present work is focused on the aforesaid application, it can be useful to address similar problems formulated in the context of regression. The contributions are listed as follows:

- In the context of RAMIS, [1] and [2] only address the SRIs' tool-tip position and consider feed-forward neural network architectures (i.e. CNN and RPN) in their proposed regression models. In the present work, the

estimation of the tool-tip velocity is included, and a CNN (for FE) in addition to an LSTM network are taken into account in the regression model. Therefore, each estimated variable is considered a 1D time series instead of a single real value.

- Four analyses were made in the regression model to reveal the best design practices, as well as its shortcomings:

  (i) The impact of using different loss functions in the regression model that consist of a fine-tuned VGG16 neural network [4] (pre-trained on 1.2 million images from ImageNet dataset [18]) for feature extraction (FE), serially connected with an LSTM network. The loss functions considered are the Root Mean Squared Error (RMSE) and Gradient Difference Loss (GDL).

  (ii) A Residual Neural Network (ResNet) is designed for FE to counteract the intense computational cost of using the VGG16 model, resulting in a reduction in the total number of parameters used. Subsequently, the ResNet is connected in series with an LSTM network, and evaluated with respect to the regression model that uses the VGG16 neural network for FE.

  (iii) The quality of the estimated signals is evaluated, by varying the number of time steps in the LSTM network, which takes as input feature vectors computed from the fine-tuned VGG16 model. This analysis provides hints about the trade-off between model complexity and the quality of the estimated signals.

  (iv) The capability of the regression model to generalize to "unseen" tasks is evaluated. This analysis is carried out by training the regression model (consisting of the VGG16 and LSTM networks) only on data related to suturing tasks (the "seen" task), and then evaluating its performance on needle-passing and knot-tying tasks.

2. A Regression Model Based on CNN+LSTM

The regression model investigated in this work consists of a CNN connected in series with an LSTM network. The CNN computes feature vectors Φ, from input video frames X. Afterward, the LSTM network processes Φ, and models their temporal information to produce the final output, Y. Such architecture is applied in the estimation of the tool-tip position and velocity in 3D space of two surgical robotic instruments (SRI), given as input only monocular video sequences. This architecture is illustrated in Figure 1, with a description of X, Y and Φ in Table 1, and was validated with analysis and experiments in the public dataset

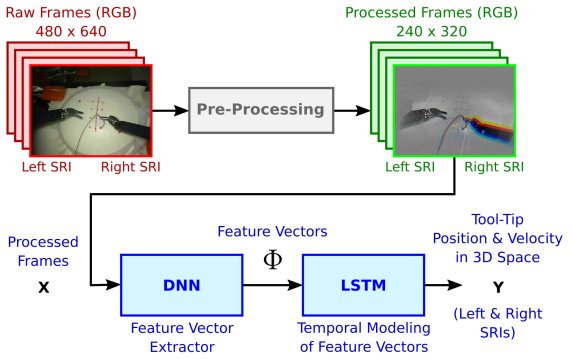

Figure 1. Regression model for the estimation of the position and velocity in 3D space for each surgical robotic instrument (SRI).

$X \in \Re^{H \times W \times C}$
H, W, C: Image height, width and channels, respectively.
$\Phi \in \Re^{N_{FV}}$
N_{FV}: Size of the feature vectors.
$Y = [x^l, y^l, z^l, v_x^l, v_y^l, v_z^l, x^r, y^r, z^r, v_x^r, v_y^r, v_z^r] \in \Re^{12}$
(x^k, y^k, z^k): Tool-tip position (in 3D space) for the SRI k.
(v_x^k, v_y^k, v_z^k): Tool-tip velocity (in 3D space) for the SRI k.
$k \in [l, r]$: l and r stand for left and right SRIs, respectively.

Table 1. Description of variables X, Y and Φ in the regression model.

JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS) [19] [20]. This dataset provides video sequences related to three different surgical tasks executed with a pair of SRIs. The surgical tasks are suturing (ST), knot-tying (KT) and needle-passing (NP). In addition, kinematic data (tool-tip position/velocity in 3D space) is available at each time instant for each SRI.

As Figure 1 shows, pre-processing operations are applied on the raw video frames. A mean frame was removed from each video and subsequently, three consecutive RGB frames, each one converted to grayscale, were concatenated resulting in a space-time image representation. Since the mean frame removal suppresses the static background present in video sequences [13], the space-time image representation corresponds to temporal derivatives as described in [17]. These preprocessing steps prevent the CNN from overfitting. Although these operations were performed off-line, they can be easily extended to real-time scenarios. A sample of raw frames and their corresponding processed versions for each surgical task are presented in Figure 2. Two CNNs were studied as feature extractors, namely the VGG16 neural network and a smaller network with residual layers (ResNet). The LSTM used in all the experiments is the variant with Coupled Input-Forget Gates (CIFG), which has been studied in [21] and suggested as

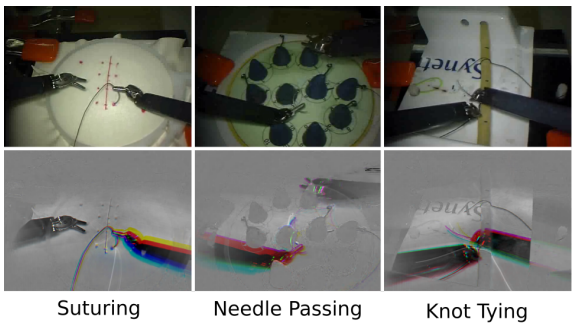

Figure 2. Sample of raw (top row) and processed (bottom row) frames for each surgical task.

an alternative model with fewer parameters than the Vanilla LSTM network with added peephole connections. The optimization of the whole regression model was performed in two stages. First, the CNN was optimized by taking the processed video frames as input, and the ground-truth signals as output. Subsequently, the LSTM network was optimized by taking as input and output, the feature vectors computed from the CNN and the ground-truth signals, respectively.

The four analyses performed on the regression model, are described in Sections 2.1-2.4.

2.1. Loss function analysis: RMSE & GDL

A loss function that takes into account the RMSE and GDL was investigated for the regression model. The RMSE represents a measure of the distance between ground-truth and estimated data, while the GDL penalizes the gradients of the ground-truth and estimated data. The gradients adopted here are the neighboring values differences. This simple form of the GDL has been studied in [22] to enhance the sharpness of the objects in the task of video frame prediction. The GDL can be helpful in the estimation of lower dimensional data, such as 1D time series signals. Equation (1) presents a loss function $\mathcal{L}$ describing these concepts. In this equation, a linear combination of the RMSE ($\mathcal{L}_{rmse}$) and GDL ($\mathcal{L}_{gdl}$) terms is weighed by $\alpha \in [0, 1]$. The definition for the RMSE appears in Equation (2) and GDL in Equation (3) with the parameter $\beta = \{1, 2\}$. Y and $\widehat{Y}$ correspond to ground truth and estimated signals, respectively. The indexes i and j iterate over M samples in the dataset and over N kinematic variables, respectively.

$$\mathcal{L}(Y, \widehat{Y}) = \alpha\, \mathcal{L}_{rmse}(Y, \widehat{Y}) + (1 - \alpha)\, \mathcal{L}_{gdl}(Y, \widehat{Y}) \quad (1)$$

$$\mathcal{L}_{rmse}(Y, \widehat{Y}) = \sum_{i}^{M} \Omega\left(\sqrt{\frac{1}{N} \sum_{j}^{N} (Y_i^{(j)} - \widehat{Y}_i^{(j)})^2} \right) \quad (2)$$

$$\mathcal{L}_{gdl}(Y, \widehat{Y}) = \sum_{i}^{M} \Omega\left(\sum_{j}^{N} \left| |Y_i^{(j)} - Y_{i-1}^{(j)}| - |\widehat{Y}_i^{(j)} - \widehat{Y}_{i-1}^{(j)}| \right|^{\beta} \right) \quad (3)$$

Model	DNN	α	$\Omega(r)$	
			RMSE Term	GDL Term
V1	VGG16	1.00	r	—
	LSTM	1.00	r	—
R1	ResNet	1.00	r	—
	LSTM	1.00	r	—
V2	VGG16	0.80	$\ln(r^2 + \epsilon)$	$\ln(r + \epsilon)$
	LSTM	0.75	r	r
R2	ResNet	0.80	$\ln(r^2 + \epsilon)$	$\ln(r + \epsilon)$
	LSTM	0.75	r	r

Table 2. Parameter α (Equation (1)) and transformation $\Omega(r)$ applied in the RMSE (Equation(2)) and GDL (Equation(3)) terms that define the loss function used to optimize the regression models V1, V2, R1 and R2. The model composed of the VGG16 and LSTM networks is optimized with a different set of loss functions as described by V1 and V2. These loss functions are also investigated in the model consisting of the ResNet and LSTM networks, resulting in R1 and R2.

Each term in the summation of Equation (2) and (3) is transformed by the function $\Omega(\cdot)$. In the simplest case, this function can be defined as $\Omega(r) = r$, where $r \in \Re$ stands for the residual. Other definitions are considered based on a logarithmic function $\Omega(r) = \ln(r^2 + \epsilon)$ with $r \in \Re$, or $\Omega(r) = \ln(r + \epsilon)$ with $r \in \Re_{\geq 0}$, where ϵ in the last two equations is a small positive constant. Table 2 shows the loss function used to optimize two models denoted as V1 and V2, given different values of the parameter α and definitions of the function $\Omega(\cdot)$ The parameter β in Equation (3), was set to 1.0. Following the illustration of Figure 1, the input data is defined by $X \in \Re^{224 \times 224 \times 3}$, and the feature vectors are computed from the fc7 layer of the VGG16 model, resulting in $\Phi \in \Re^{4096}$.

2.2. ResNet model as feature extractor

In recent years, ResNets have succeeded in computer vision tasks such as image recognition [23]. These models have "shortcut" connections that allow the design of deeper neural networks in comparison to a plain CNN. Furthermore, the optimization of a ResNet is easier than that of a plain CNN.

Based on these performances, a feature extractor has been designed with residual layers, which has fewer parameters (∼4.92 M) compared to the VGG16 model (∼138 M), resulting in a reduction of ∼96.44 % in the neural network size. This percentage was computed by taking into account all the parameters found in the convolutional and fully connected layers of each neural network. Each residual block in the ResNet model was designed according to [24], and dropout was found beneficial and applied between the convolutional layers as described in [25]. The proposed ResNet architecture is illustrated in Figure 3, and consists of a con-

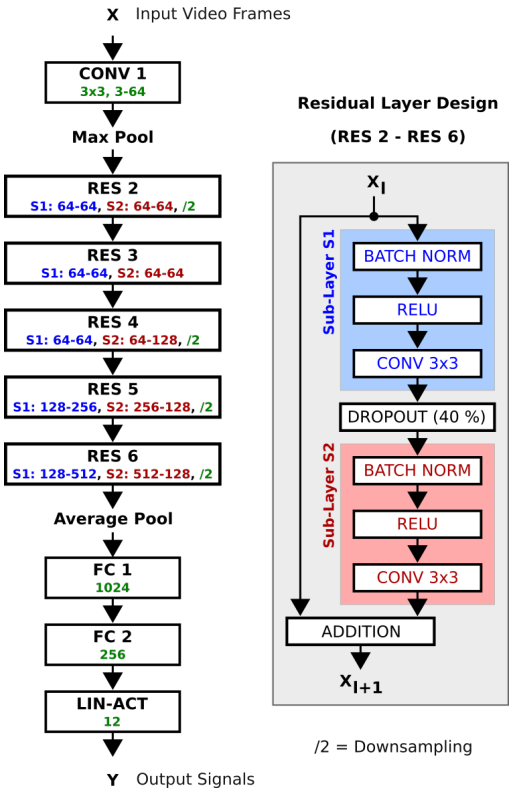

Figure 3. Architecture of the designed ResNet as feature extractor. The filter size and number of input-output feature maps are indicated in the first convolutional layer (CONV 1). Each residual layer (RES 2-6) is designed with two sub-layers, S1 and S2, and dropout (with probability of 40 %) is applied between them during training, as depicted in the diagram of the right side. In these layers, the number of input-output feature maps (for S1 and S2), as well as downsampling operations (/2) are shown. The size of each fully connected layer (FC 1-2) and the output layer (LIN-ACT) is also indicated.

volutional layer (CONV 1), followed by 5 residual layers (RES 2-6), 2 fully connected layers with ReLU as activation function (FC 1-2), in addition to an output layer with a linear activation function (LIN-ACT). Max and average pooling were applied after the first convolutional (CONV 1) and the last residual (RES 6) layers, respectively. Convolutions with a stride of 2 were used to downsample the dimensions of feature maps in the residual layers.

The dimensionality of the input video frames and target signals is $X \in \Re^{240\times320\times3}$ and $Y \in \Re^{12}$, respectively, as shown in the diagram of Figure 3. The feature vectors are computed by reshaping the output feature maps from the residual layer RES 6, after an average pooling operation is applied. Therefore, 128 feature maps of resolution 4×5 are reshaped as a single vector, resulting in $\Phi \in \Re^{2560}$. This neural network was analyzed with two loss functions as shown in Table 2, resulting in models R1 and R2.

2.3. Varying the time steps in the LSTM network

The quality of the estimated signals was evaluated by varying the number of time steps used in the LSTM network to process the feature vectors from the fine-tuned VGG16 model. These experiments were carried out by training the LSTM network at 32, 64, and 96 time steps. Both the VGG16 and LSTM networks were optimized with only the RMSE, by setting $\alpha = 1.0$ and $\Omega(r) = r$ in Equations (1) and (2), respectively. These parameters were selected based on the results of the experiment described in Section 2.1. As discussed later (in Section 3.1), a loss function which considers only the RMSE represents a reasonable design choice.

2.4. Generalization to "unseen" tasks

Finally, the capability of the regression model to deal with "unseen" tasks was evaluated. Specifically, a baseline model which consists of the VGG16 and LSTM networks was trained on data related to suturing tasks, and evaluated on knot-tying and needle-passing tasks. The two mentioned CNNs, were optimized with only the RMSE, by setting $\alpha = 1.0$ and $\Omega(r) = r$ in Equations (1) and (2), respectively.

3. Experiments & Results

The experiments were carried out on the JIGSAWS dataset, which consists of 206 video sequences of three surgical tasks, namely suturing (78), knot-tying (72) and needle-passing (56). Each frame in a video sequence is associated with a 12D vector of ground-truth position and velocity in 3D space for each SRI. The whole dataset was split in 75 % and 25 % as the training set and test set, respectively. Following the illustration of Figure 1, the input video frames for the VGG16 neural network are reshaped from 240×320 pixels to a square image (320×320) by replicating pixels, and subsequently resized to a resolution of 224×224 pixels. This strategy is used instead of centering and cropping the image, to avoid losing the location of the SRIs tool-tip on each video frame. In contrast, the ResNet model takes as input the processed video frames (240×320 pixels), without any further resizing.

The neural network models were implemented in Python, making use of the Google's open source machine learning framework, Tensorflow [26]. The experiments were carried out using two NVIDIA Titan X Graphic Processing Units (GPUs).

3.1. Impact of the objective function

This experiment was performed in two steps for each model V1 and V2 (see Table 2). In the first step, the pre-trained VGG16 model was fine-tuned over ∼100K iterations with the Root Mean Square Propagation (RMSProp) optimizer, using a batch size of 80 samples and learning rate

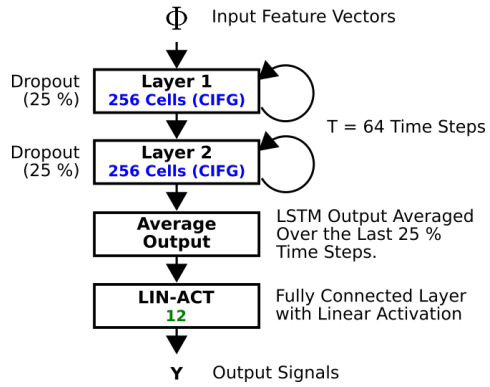

Figure 4. LSTM network design. Each cell is designed with Coupled Input-Forget Gates (CIFG).

of 1×10^{-5}. Dropout was applied in the fully connected layers fc6 and fc7, with a probability of 50 %. Subsequently, feature vectors of dimension 4096 were computed from the fc7 layer of the VGG16 model ($\Phi \in \Re^{4096}$). In the second step, an LSTM network whose design is shown in Figure 4, was optimized by taking as input these feature vectors. This LSTM network was trained over ∼160K iterations using the RMSProp optimizer, with a batch size of 250 samples and a learning rate of 0.0025. Additionally, dropout was applied at the output of each layer with a probability of 25 %.

Table 3 presents the Root Mean Squared Error (RMSE) and Pearson Correlation Coefficient (PCC) metrics computed on the test set for models V1 and V2. By examining these values, and specifically the PCC, it can be concluded that using the GDL does not provide an advantage (at least for this application). Optimizing both the VGG16 and LSTM networks with only the RMSE represents a reasonable choice. Moreover, in Table 3 it is appreciated that the PCC is higher for the position than for the velocity variables. This result can be justified by examining the shape of a sample of ground-truth and estimated signals, as depicted in Figure 6 for model V1. In this illustration, the position and velocity variables have a normalized amplitude (in the range +/-5). Furthermore, it can be appreciated that the position variables are smoother compared to the velocity variables.

3.2. ResNet model results

This experiment was carried out in two stages for each model R1 and R2 (see Table 2). First, the ResNet was trained from scratch over ∼140K iterations with the Root Mean Square Propagation (RMSProp) optimizer, using a batch size of 80 samples, and a learning rate of 0.5×10^{-4}. Subsequently, an LSTM network was trained to model a sequence of feature vectors computed from the ResNet ($\Phi \in \Re^{2560}$). The LSTM network design (see Figure 4) and hyper-parameters used during the optimization are the same than those described in Section 3.1. This neural network was trained over ∼180K iterations.

Table 4 presents a comparison between models R1 and R2, by providing the RMSE and PCC metrics computed on the estimated position and velocity in 3D space (data in the test set) for each SRI. By examining this data, the results are more favorable for the model R2 than R1. The model R2 has higher PCC and lower RMSE metrics for most of the variables. As discussed in Section 3.1, estimating the SRIs' tool-tip position is easier compared to velocity. This is quantitatively described in Table 4 by the higher and lower quality metrics for the position and velocity variables, respectively. An important observation supporting the better performance of model R2, is that including the GDL in the loss function results in more benefits when training a model from scratch (ResNet) than when using a pre-trained model (VGG16). Furthermore, model R2 is competitive with the baseline model V1, as depicted in Figure 5. In the last illustration, the metrics (RMSE and PCC) for the model R2 are deteriorated by a small margin with respect to model V1, as indicated by the percentage on top of each variable. Therefore, there is a compromise between the number of parameters in the CNN used for FE and the quality of the estimated signals.

3.3. Effect of varying the time steps in LSTM

The results of the regression model that consists of the VGG16 network as well as the LSTM network with 32, 64 and 96 time steps, optimized only with the RMSE are shown in Table 5. The LSTM network design (see Figure 4) as well as the hyper-parameters used for its optimization are the same as described in Section 3.1. By analyzing the RMSE and PCC presented in Table 5, the regression model that takes into account an LSTM network with 32 time steps is competitive with the model that considers 64 time steps. In contrast, an LSTM network optimized over 96 time steps does not provide an advantage over the other two mentioned models.

The aforesaid results shed light on the careful selection of the number of time steps used in the LSTM network. Intuitively, by increasing this hyper-parameter, the LSTM network should provide a better performance, and only a more expensive model is expected. However, this is not always the case, and a more economical model can meet the requirements (i.e. the LSTM network with 32 instead of 96 time steps).

3.4. Evaluation of the model for "unseen" tasks

This experiment was carried out in the conditions described for model V1 in Section 3.1, however, the training set consisted only of data related to suturing tasks. Afterward, the capability of the model to generalize to "unseen" tasks was evaluated by the metrics displayed in Figure 7. In

Model	Left Tool						Right Tool					
	x^l	y^l	z^l	v_x^l	v_y^l	v_z^l	x^r	y^r	z^r	v_x^r	v_y^r	v_z^r
RMSE (Lower values are better)												
V1	0.2577	0.1442	**0.1421**	**0.3801**	0.2126	**0.3068**	**0.0723**	**0.0646**	0.0666	0.2014	0.1158	**0.1763**
V2	**0.1975**	**0.1415**	0.1677	0.3896	**0.2051**	0.3138	0.0762	0.0650	**0.0553**	**0.1948**	**0.1118**	0.1795
PCC (Values close to 1.0 are better)												
V1	0.9419	**0.9599**	**0.9456**	**0.6949**	0.3647	**0.6403**	**0.8898**	**0.9177**	0.7871	**0.5002**	**0.2720**	**0.4929**
V2	**0.9497**	0.9444	0.9371	0.6599	**0.3798**	0.6025	0.8888	0.9159	**0.8049**	0.4913	0.2667	0.4332

Table 3. RMSE and PCC computed for models V1 and V2 (data in the test set) for each SRI (left & right)

this illustration, the RMSE and PCC metrics are better for the "seen" task, which is suturing, on the other hand, they are deteriorated by a wide margin for the "unseen" tasks, represented by needle-passing and knot-tying. These results indicate that the SRIs' motion required for each task is highly task-specific. Thus, a robust regression model for this application needs to take into account data related to all the surgical tasks.

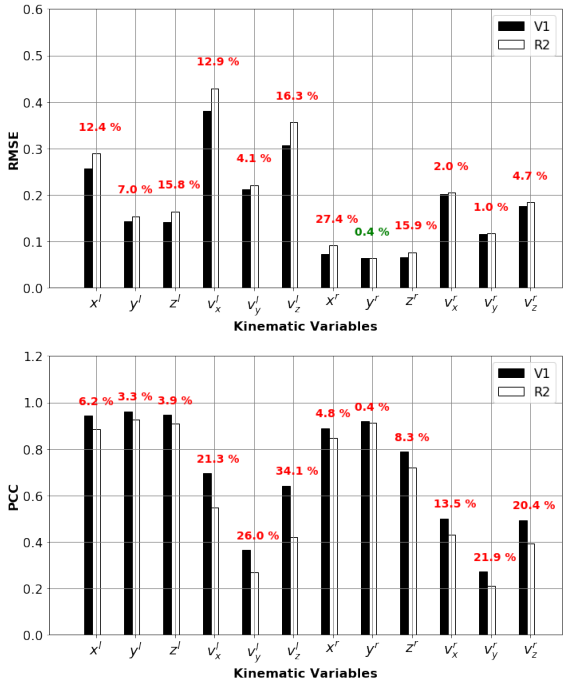

Figure 5. RMSE (top row) and PCC (bottom row) metrics for the regression models V1 and R2. The percentage in red/green color, describes the decrease/increase in the quality of the metrics for the model R2 with respect to the baseline model V1.

4. Conclusions

In the present work, a regression model based on DNNs for the application of VBSS was developed. It estimates the tool-tip position and velocity in 3D space of a pair of SRIs. The analyses made on the regression model reveal

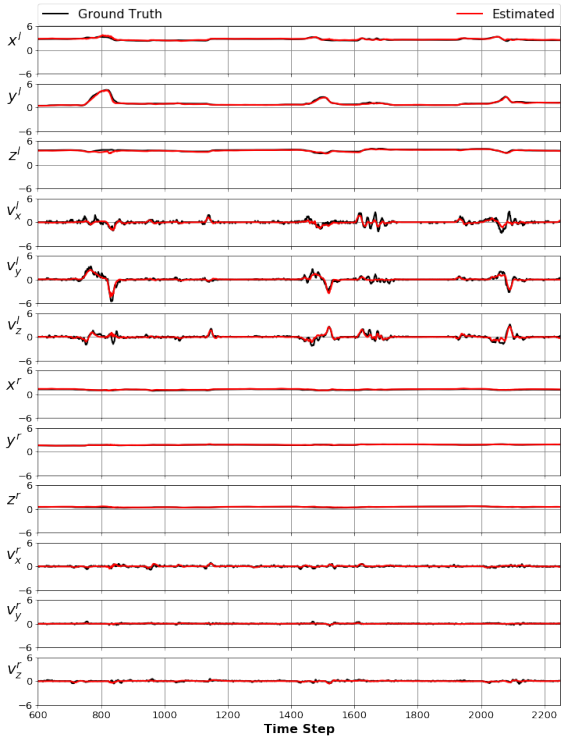

Figure 6. Estimated tool-tip position and velocity in 3D space for each SRI, related to a suturing task using the baseline model V1. The signals' amplitude is shown in a normalized space (in the range +/-5).

the benefits and drawbacks of different design choices.

Using the fine-tuned VGG16 neural network for FE represents the best design choice. However, a smaller CNN (in terms of parameters) designed with residual layers is competitive with the VGG16 model. The analysis of the loss function, highlights that including the GDL term is beneficial when the regression model takes into account a CNN trained from scratch and designed for FE (ResNet). In contrast, the advantage of using the GDL in the loss function is less clear when a pre-trained model is used for FE (fine-tuned VGG16 neural network).

Regarding the analysis of LSTM network, increasing the

Model	Left Tool						Right Tool					
	x^l	y^l	z^l	v_x^l	v_y^l	v_z^l	x^r	y^r	z^r	v_x^r	v_y^r	v_z^r
RMSE (Lower values are better)												
R1	0.3676	0.1883	0.1923	0.4612	**0.2187**	0.3613	0.0938	0.0862	0.0951	0.2149	0.1203	0.1910
R2	**0.2896**	**0.1542**	**0.1645**	**0.4292**	0.2213	**0.3567**	**0.0921**	**0.0643**	**0.0771**	**0.2054**	**0.1170**	**0.1845**
PCC (Values close to 1.0 are better)												
R1	0.8284	0.9197	0.8754	0.4823	**0.2902**	**0.4296**	0.8187	0.9011	**0.7640**	0.3730	0.1892	0.3516
R2	**0.8837**	**0.9278**	**0.9085**	**0.5469**	0.2698	0.4217	**0.8473**	**0.9136**	0.7216	**0.4325**	**0.2125**	**0.3923**

Table 4. RMSE and PCC computed for models R1 and R2 (data in the test set) for each SRI (left & right)

Time Steps	Left Tool						Right Tool					
	x^l	y^l	z^l	v_x^l	v_y^l	v_z^l	x^r	y^r	z^r	v_x^r	v_y^r	v_z^r
RMSE (Lower values are better)												
32	**0.2277**	**0.1268**	0.1422	**0.3635**	**0.1976**	**0.2934**	0.0913	0.0671	**0.0620**	**0.1924**	**0.1111**	**0.1669**
64	0.2577	0.1442	0.1421	0.3801	0.2126	0.3068	**0.0723**	**0.0646**	0.0666	0.2014	0.1158	0.1763
96	0.2644	0.1335	**0.1292**	0.3931	0.2121	0.3230	0.0885	0.0658	0.0723	0.1994	0.1124	0.1842
PCC (Values close to 1.0 are better)												
32	0.9334	0.9583	0.9481	**0.7119**	**0.4759**	**0.6692**	**0.8913**	**0.9194**	0.7534	**0.5511**	**0.3256**	**0.5530**
64	**0.9419**	**0.9599**	0.9456	0.6949	0.3647	0.6403	0.8898	0.9177	0.7871	0.5002	0.2720	0.4929
96	0.9341	0.9542	**0.9530**	0.6382	0.3534	0.5804	0.8863	0.9121	**0.7945**	0.4786	0.2506	0.4023

Table 5. Number of time steps used in the LSTM network, and their impact in the quality of the estimated signals (data in the test set) for each SRI (left & right), evaluated by the RMSE and PCC metrics.

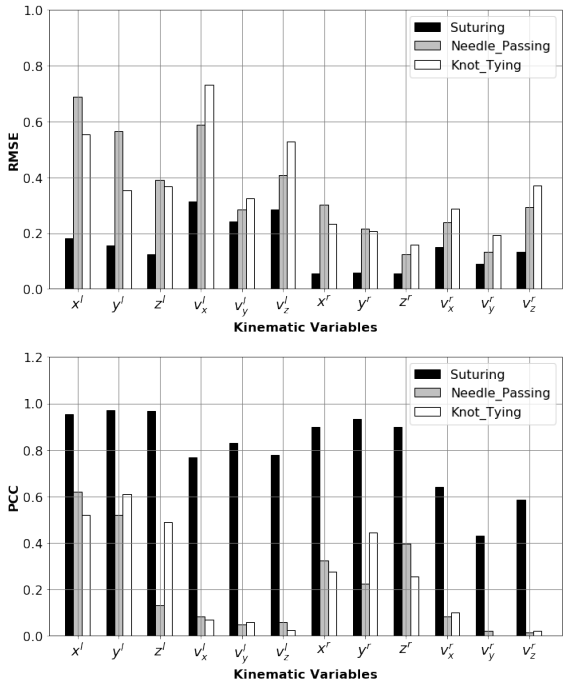

Figure 7. RMSE (top row) and PCC (bottom row) metrics computed on the regression model trained only on data related to suturing tasks, and evaluated on the "seen" (suturing) and "unseen" tasks (needle-passing and knot-tying).

number of time steps used to model sequential feature vectors, does not translate into an improvement in the quality of the estimated signals. Using 32 instead of 96 time steps in the LSTM network (to process the feature vectors from the VGG16 model), resulted in better accuracy (in terms of the RMSE and PCC) and a more economical model. Finally, the analysis related to the generalization of the regression model to "unseen" tasks, shows the importance of providing the DNN with data related to different surgical tasks during the training process.

It is important to notice that the regression model described in this work represents a generic framework, and it is not restricted to a specific application (i.e. SRIs' tool-tip position and velocity estimation). It can be extended to similar problems, where a non-linear mapping between monocular video sequences and 1D time series signals is required, i.e. tracking persons in video surveillance. As future work, this regression model is to be improved by interpreting its predictions and identifying its weaknesses (as done for image classification in [27]) using methods such as layer-wise relevance propagation [28]. Also, a semi-supervised approach represents an interesting avenue of research.

Acknowledgment

We thank the NVIDIA Corporation for supporting this research with a Titan X GPU card.

References

[1] C. Rupprecht, C. Lea, F. Tombari, N. Navab, and G. D. Hager. Sensor substitution for video-based action recognition. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5230–5237, 2016. 1, 2

[2] D. Sarikaya, J. Corso, and K. Guru. Detection and localization of robotic tools in robot-assisted surgery videos using deep neural networks for region proposal and detection. *IEEE Transactions on Medical Imaging*, PP(99):1–1, 2017. 2

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012. 2

[4] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 2, 3

[5] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems (NIPS)*, pages 3320–3328, 2014. 2

[6] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 806–813, 2014. 2

[7] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, Nov 1997. 2

[8] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems (NIPS)*, pages 568–576, 2014. 2

[9] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, Dec 2015. 2

[10] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt. *Sequential Deep Learning for Human Action Recognition*, pages 29–39. Springer Berlin Heidelberg, 2011. 2

[11] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4694–4702, June 2015. 2

[12] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634, 2015. 2

[13] T. Pfister, K. Simonyan, J. Charles, and A. Zisserman. Deep convolutional neural networks for efficient pose estimation in gesture videos. In *Asian Conference on Computer Vision (ACCV)*, pages 538–552. Springer, 2014. 2, 3

[14] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1653–1660, June 2014. 2

[15] V. Belagiannis, C. Rupprecht, G. Carneiro, and N. Navab. Robust optimization for deep regression. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2830–2838, Dec 2015. 2

[16] J. Zhou, X. Hong, F. Su, and G. Zhao. Recurrent convolutional neural network regression for continuous pain intensity estimation in video. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 84–92, June 2016. 2

[17] A. Owens, P. Isola, J. McDermott, A. Torralba, E. H. Adelson, and W. T. Freeman. Visually indicated sounds. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2405–2413, 2016. 2, 3

[18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 3

[19] Y. Gao, S. S. Vedula, C. E. Reiley, N. Ahmidi, B. Varadarajan, H. C. Lin, L. Tao, et al. Jhu-isi gesture and skill assessment working set (jigsaws): A surgical activity dataset for human motion modeling. In *MICCAI Workshop: M2CAI*, volume 3, 2014. 3

[20] N. Ahmidi, L. Tao, S. Sefati, Y. Gao, C. Lea, B. Bejar, L. Zappella, S. Khudanpur, R. Vidal, and G. D. Hager. A dataset and benchmarks for segmentation and recognition of gestures in robotic surgery. *IEEE Transactions on Biomedical Engineering*, PP(99):1–1, 2017. 3

[21] K. Greff, R. K. Srivastava, J. Koutnk, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99):1–11, 2016. 3

[22] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *CoRR*, abs/1511.05440, 2015. 4

[23] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 4

[24] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)*, pages 630–645. Springer, 2016. 4

[25] S. Zagoruyko and N. Komodakis. Wide residual networks. In *27th British Machine Vision Conference (BMVC)*, 2016. 4

[26] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016. 5

[27] S. Lapuschkin, A. Binder, G. Montavon, K.-R. Müller, and W. Samek. Analyzing classifiers: Fisher vectors and deep

neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2912–20, 2016. 8

[28] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7):e0130140, July 2015. 8