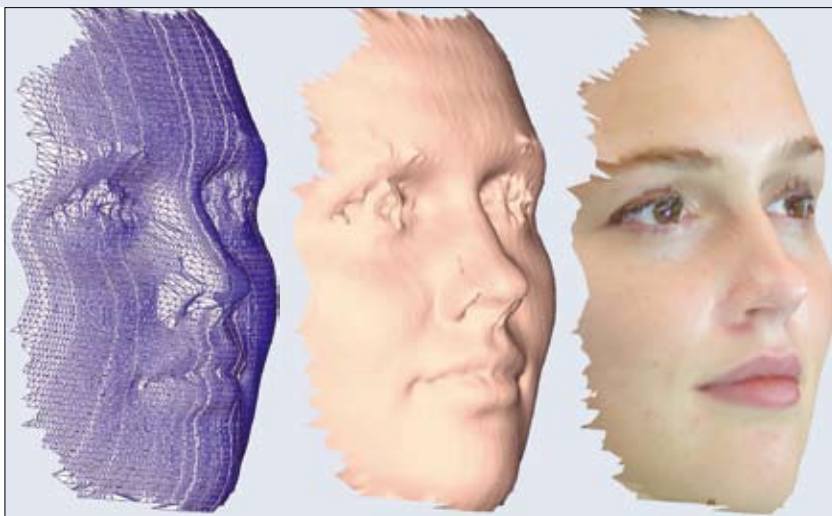


## Räumliche Scans mit Consumer-Hardware

# Mit viel Profil

Nach dem Structured-Light-Prinzip lässt sich ein kostengünstiges und zuverlässiges Verfahren für räumliche Scans erzielen. Voraussetzungen sind neben einem handelsüblichen Projektor und einer Digicam ein paar Kunstgriffe aus der mathematischen Algorithmen-Kiste. Philipp Fechteler



**Abbildung 1:** Das Ergebnis des Scans ist ein räumliches Gitter (Mesh, links). 3D-Software schattiert das Gitter entweder (Mitte) oder überzieht es mit einer Foto-Textur (rechts).

**3D-Scanner** aus Industrie und Forschung tasten räumliche Objekte meist mit einer Laseroptik und Spezialkameras ab. Bei der Investition etlicher Tausend Euro muss der Einsatzzweck des Scanners entsprechend einträglich sein. Das Berliner Fraunhofer Heinrich-Hertz-Institut hat nun ein Verfahren entwickelt, das 3D-Scans binnen weniger Sekunden mit einem handelsüblichen Projektor und einer Digitalkamera erstellt.

Das für Gesichter optimierte Verfahren basiert auf dem so genannten Structured-Light-Prinzip. Es projiziert ein Farbstreifenmuster auf die Haut, das die Digitalkamera aufnimmt. Die räumliche Tiefe berechnet es aus der Verzerrung des aufgenommenen Musters. Als Resultat entsteht ein Drahtgitter, das 3D-Programme auch als solides, schattiertes Objekt darstellen. Dient ein Foto des gescannten Gesichts als Textur, entsteht ein realistisch wirkendes räumliches Modell (**Abbildung 1**), das sich anders als zweidimensionale Fotos jedoch animieren lässt.

Die Herausforderung bei dem von **[1]** und **[2]** inspirierten Verfahren besteht zum größten Teil darin, das Signalausgang auszugleichen, das sich aus den schwankenden Reflexionseigenschaften der Haut, ungünstiger Beleuchtung und der Unvollkommenheit optischer Sensoren ergibt. Gelingt dies, steht ein Verfahren zur Verfügung, das gegenüber konventionellen Laserscans zwei Vorteile bietet: Es reicht eine einzelne Aufnahme mit der Digitalkamera, um es auch für bewegte Objekte einzusetzen. Außerdem funktioniert es kostengünstig mit Consumer-Hardware (**Abbildung 2**). Die wissenschaftlichen Details liefern **[3]** und **[4]**.

### Alles Linux - fast alles

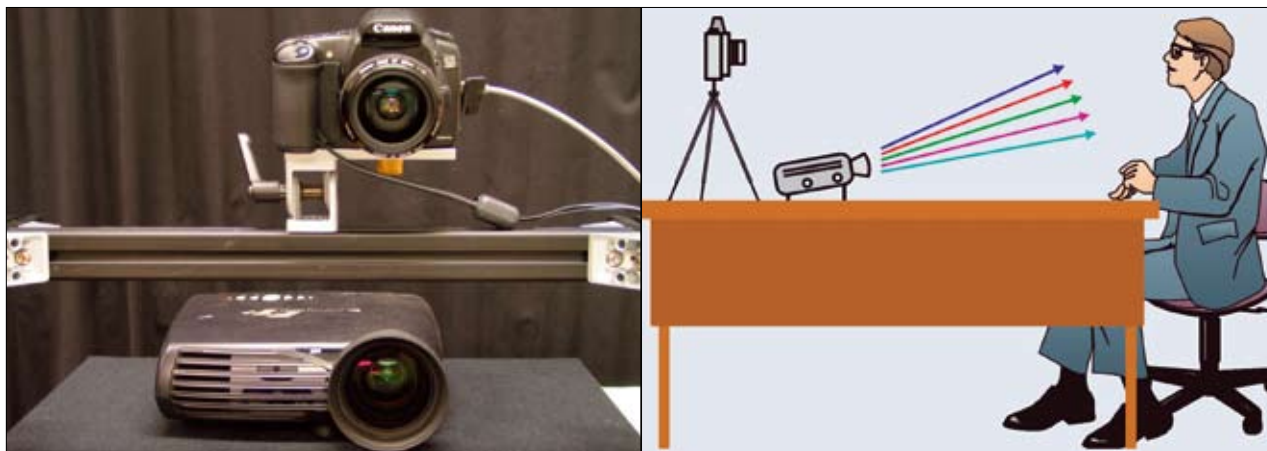
Die Entwicklung des Systems fand vollständig unter Linux mit Hilfe von Tools wie dem G++-Compiler und Vim statt. Zum Testen und Simulieren verschiedener Algorithmen benutzten die Entwickler R, eine Computersprache für statisti-

sches Rechnen **[5]**. Erste Prototypen implementierten sie mit der Computer-Vision-Bibliothek Vigna **[6]**, die mit C++-Templates arbeitet. Mit dem C++-Cross-Plattform-Toolkit Fox **[7]** programmierten sie die grafische Oberfläche. Dass die Entwicklung vollständig unter Linux stattfand, scheint dem fertigen System nichts zu nützen: Es lässt sich derzeit nur unter Windows betreiben, da der Treiber für die Softwaresteuerung der benutzten Kamera, einer Canon EOS 20D, nur für diese Plattform verfügbar ist.

### Ein projiziertes Muster mit bestimmten Eigenschaften

Das Ziel beim 3D-Scan nach dem Structured-Light-Prinzip besteht darin, in der Aufnahme jeden auf das räumliche Profil projizierten Punkt seinem Ursprung zuzuordnen. Projiziert man im Extremfall nur einen Punkt, ist der natürlich leicht zu identifizieren. Dann wären allerdings sehr viele Aufnahmen nötig, um das ganze Gesicht abzutasten. (Das sehr zuverlässige Ein-Punkt-Verfahren nehmen Experimente gern als Referenz.) Im anderen Extrem bestrahlt ein Projektor jeden Punkt der Szene mit andersfarbigem Licht. Hier reicht theoretisch eine Aufnahme. Die große Zahl der benötigten Farben macht eine Zuordnung in der Praxis jedoch unmöglich.

Die hier vorgestellte Lösung wählt ein für den Einsatzzweck optimiertes Verfahren: Das projizierte Muster besteht aus horizontalen, farbigen Linien (**Abbildung 3**). Zwischen ihnen liegen unbeleuchtete Streifen. Um die Zuordnung der Farbstreifen zu erleichtern, kommen nur sieben Farben mit voll gesättigten RGB-Farbkämen zum Einsatz: Die Grundfarben



**Abbildung 2a und 2b:** Um die räumliche Form des Gesichts zu ermitteln, reicht handelsübliche Hardware aus: Ein Projektor, der ein Muster auf das Gesicht des Probanden wirft, und eine hinreichend gut auflösende Digitalkamera, deren Bilder dann auszuwerten sind.

Rot, Grün und Blau, die Kombinationen zweier Grundfarben, Cyan, Magenta und Gelb, sowie Weiß, die Kombination aller drei Grundfarben.

Das Muster enthält also Streifen immer wiederkehrender Farben. Bei der Zuordnung von Ausgangsmuster und projizierter Linie hilft jedoch die Reihenfolge der Farben. Sie ist daher so gewählt, dass sie eine maximale Periode aufweist, die sieben Farben also möglichst selten in der gleichen Reihenfolge auftreten. Für ausreichenden Kontrast unterscheiden sich aufeinanderfolgende Streifen außerdem in mindestens zwei Farbkanälen.

## Subpixel-Streifenerkennung

Zunächst sind in der Aufnahme des mit dem Muster bestrahlten Gesichts die horizontalen Streifen zu lokalisieren. Die Grundfarben-Streifen erscheinen in der Aufnahme als Helligkeitsmaxima. Die Genauigkeit beim Bestimmen des Mittelpunkts der farbigen Linien steigert ein Subpixel-Parabel-Fitting (**Abbildung 4**). Es errechnet aus der Lage der hellsten Pixel eine Parabel – statt des hellsten Pixels zählt nun die Parabelspitze als Maximum. Das funktioniert selbst dann

noch gut, wenn ein übersteuerter optischer Sensor mehrere benachbarte Pixel mit maximaler Sättigung ausweist.

## Eine adaptive Farbklassifizierung kompensiert

Bei der Zuordnung vom ursprünglichen zum projizierten Streifen anhand der Farbe muss die Farbklassifizierung mit ungünstigem Umgebungslicht, mit durch Schweiß oder Makeup hervorgerufenen schwankenden Reflexionseigenschaften der Haut sowie mit einer nicht perfekt arbeitender Hardware (Kamera, Projektor) zurecht kommen.

Versuche haben gezeigt, dass sich die gemessenen Farben der erkannten Streifen in einer RGB-Raumdarstellung zwar grob entlang der Achsen verteilen, die von Schwarz in Richtung der gesättigten Vollkanal-Farben gehen (**Abbildung 5**). Der Verlauf der Achsen im Farbraum weicht in der Praxis je nach Umgebungslicht oder Hautbeschaffenheit allerdings stark von den geometrischen Achsen ab.

Solche Farbverschiebungen kompensiert ein Verfahren aus der Clusteranalyse, der Kmeans-Algorithmus. Er passt Lage und Verlauf der Achsen an die realen

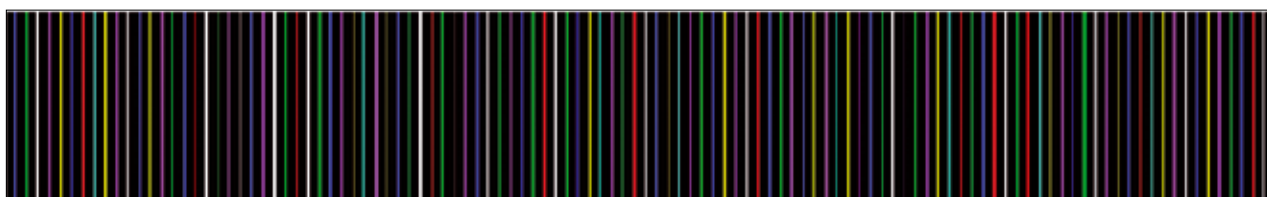
Lichtverhältnissen an. Die klassischen Kmeans-Verfahren liefern allerdings keine Linien, sondern Punkte, die Zentren von Werteklustern markieren. Das iterative Vorgehen lässt sich jedoch auch auf Prototyp-Linien anwenden, die die langgestreckten Cluster möglichst zentral durchlaufen (**Abbildung 7**):

- Klassifizierung mit aktuellen Parametern: Jedes Pixel bekommt die Farbe zugewiesen, deren Prototyp-Linie es am nächsten liegt.
- Anpassen der Parameter an die aktuelle Klassifizierung: Die Prototyp-Linien werden an den entstandenen Clustern neu ausgerichtet.

Der Vorgang ist beendet, sobald ein Durchlauf keine signifikanten Veränderungen mehr bewirkt. Im Allgemeinen ist dies nach fünf Iterationen der Fall.

## Ahnenforschung für Fortgeschrittene

Korrespondenzen zu bestimmen, also die erkannten Linien im projizierten Muster ihrem Ursprung zuzuordnen, bildet das Herzstück der 3D-Rekonstruktion. Dabei handelt es sich um ein typisches kombinatorisches Optimierungsproblem



**Abbildung 3:** Das projizierte Muster garantiert maximale Farbkontraste. Die Reihenfolge der nur sieben verschiedenen Farbstreifen ist auf möglichst seltene Wiederholung der Reihenfolge optimiert. Zwischen den Linien liegen jeweils unbeleuchtete Streifen.

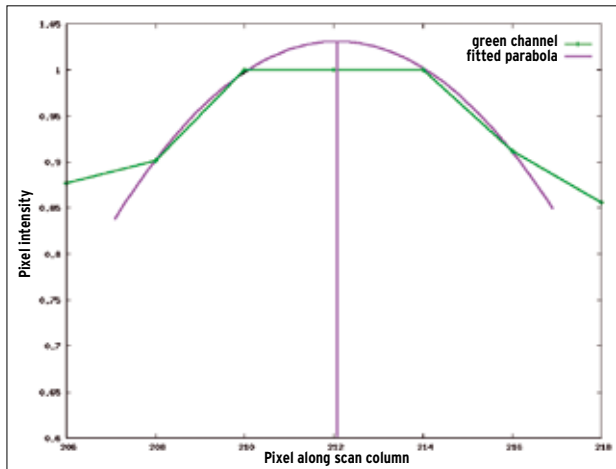


Abbildung 4: Ein Parabel-Fitting errechnet die Lage der Streifenzentren auf Subpixel genau und gleicht außerdem das Übersteuern des optischen Sensors aus.

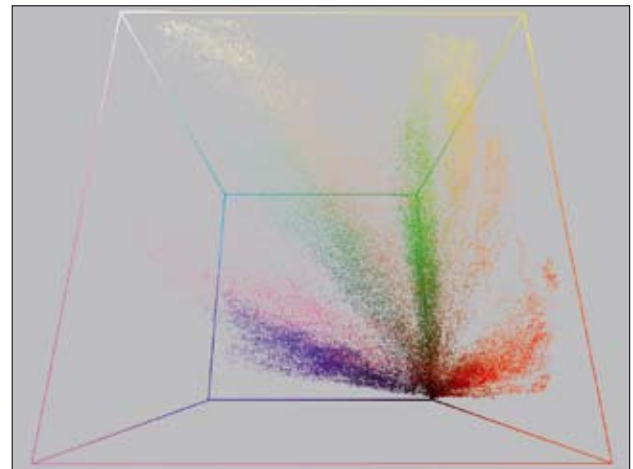


Abbildung 5: In einem Rot-Grün-Blau-Farbraum abgebildet, ballt sich die Dichte an den Achsen, also bei den Grundfarben, die den Streifenfarben entsprechen.

(siehe **Kasten „Kombinatorische Optimierungsprobleme“**). Wegen des Signalrauschens und der Tatsache, dass die Streifenfarben mehrfach wiederkehren, lässt sich ein einzelner Streifen seinem Pendant im projizierten Muster nicht eindeutig zuordnen. Erst mit einem übergreifenden Vergleich, der die farbliche Übereinstimmung und die Richtigkeit der Farbabfolge für alle Streifenpaare zugleich in Betracht zieht, klappt dies trotz einzelner Zuordnungsfehler.

Eine Kostenfunktion definiert ein Maß für die Güte einer Streifenzuordnung. Sie basiert auf der Summe der farblichen Abweichung und dem Kehrwert der Übereinstimmung bei der farblichen Abfolge. Die Lösung, die für alle Strei-

fen zusammengenommen am wenigsten „Kosten“ verursacht, bildet die Basis für das Errechnen der 3D-Koordinaten. Zur Lösung der Optimierungsaufgabe kommt die dynamische Programmierung (vergleiche **Kasten „Dynamische Programmierung“**) zum Tragen.

## Mit einer Vision zu den 3D-Koordinaten

Sind die Korrespondenzen zwischen den Streifen im Muster und in der Fotografie gefundenen, so lassen sich die 3D-Koordinaten mittels Triangulation bestimmen – hier realisiert mit Standard-Vision-Methoden [8]. Sie tasten das Gesicht entlang einer horizontalen Li-

nie ab. Neben der horizontalen Position bildet der Betrag, um den ein Streifen in der Aufnahme nach oben oder unten verschoben erscheint, die Basis, um die Raumkoordinaten zu bestimmen (**Abbildung 8**). Allerdings müssen zusätzlich die optischen Parameter von Kamera und Projektor bekannt sein, also deren Lage relativ zueinander und die Verzerrungen der Linsen. Beides misst eine dem eigentlichen Scan vorausgehende Kalibrierungsroutine [9].

## Ergebnis: Ein 3D-Modell

Noch bilden die ermittelten Punkte im Raum lediglich eine Wolke im Raum, noch kein 3D-Gitter mit festgelegten Ver-

### Kombinatorische Optimierungsprobleme

Eine der bekanntesten kombinatorischen Optimierungsaufgaben ist das Problem des Handlungsreisenden (Traveling-Salesman-Problem): Welche ist die kürzeste Route durch eine Anzahl von vorgegebenen Städten (**Abbildung 6**)? Kombinatorisch heißt die Optimierungsaufgabe deshalb, weil die Wahl einer Teilstrecke Auswirkungen auf die restliche Route hat. Die global-optimale Lösung ergibt sich aus einer Kombination von untereinander abhängigen Teillösungen.

Für eine global-optimale Lösung haben einerseits gewisse Bedingungen erfüllt zu sein, damit die Lösung überhaupt gültig ist: Der Handlungsreisende muss alle vorgegebenen Stationen erreichen. Die Menge potenzieller Lösungen besteht aus allen Reiserouten durch die Städte in jeder möglichen Reihenfolge. Eine optimale Lösung erzielt unter allen gültigen Lösungen den niedrigsten Wert für eine Kosten-

funktion, beim Handlungsreisenden der zurückzulegenden Reisedistanz.

Problematisch bei kombinatorischen Optimierungsaufgaben ist das schnelle Anwachsen der Menge potenzieller Lösungen, des so genannten Suchraums. Im Traveling-Salesman-Beispiel

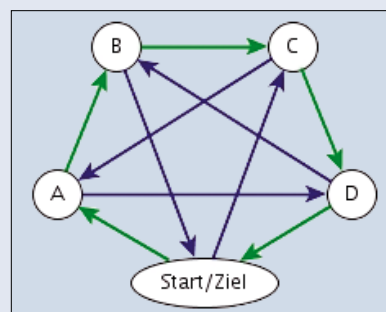


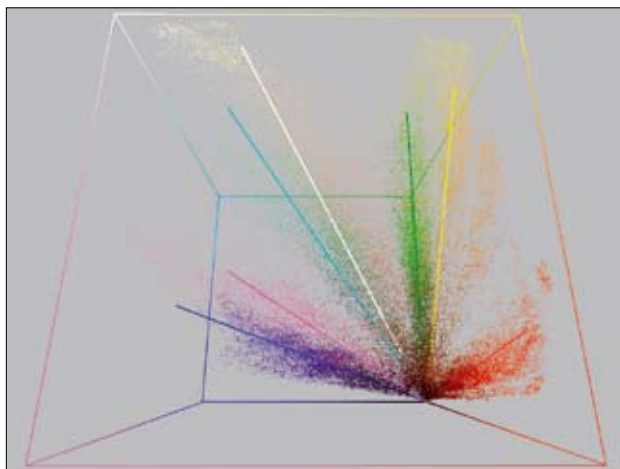
Abbildung 6: Welcher Weg ist der kürzeste? Die Zahl der Routenvarianten durch die Städte A bis D beträgt  $n$  Fakultät.

gibt es für  $n$  Städte genau  $n! = 1 * 2 * \dots * n$  potenzielle Lösungen.

Kombinatorische Optimierungsprobleme treten in den verschiedensten Disziplinen auf, beispielsweise:

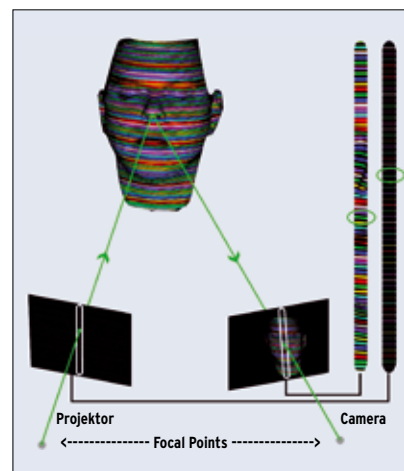
- Chipdesign: Planung der Verbindungen der Bauelemente bei minimalem Platzverbrauch.
- Genom-Sequenzieren: Zerlegen und Zusammenfügen der DNA-Stränge mit maximalem Informationsgehalt.

Zur Lösung solcher Probleme gibt es exakte und heuristische Verfahren. Exakte sind im Allgemeinen wesentlich rechenintensiver, finden dafür aber stets die global-optimale Lösung. Heuristische Verfahren erreichen lediglich eine gute Lösung, benötigen allerdings wesentlich weniger Ressourcen und stellen einen Kompromiss zwischen Ressourcenbedarf und Güte der gefundenen Lösung dar.



◀ **Abbildung 7:** Ein Kmeans-Verfahren verschiebt die Farbachsen so, dass sie zentral durch die Farbcluster laufen. Dies gleicht zum Beispiel Verschiebungen der Hautfarbe aus.

▶ **Abbildung 8:** Der Scan tastet das Gesicht horizontal ab. Aus dem Höhenversatz der Farbstreifen errechnet sich die räumliche Tiefe.



bindungslinien zwischen den Netzmaschen und einer definierten Oberfläche. Dass die Punkte jedoch allesamt auf vertikalen Scanlines liegen, erleichtert das Erzeugen eines Gitters erheblich. Das fertige Modell lässt sich im 3D-Viewer der Scansoftware betrachten.

Der hier vorgestellte Scanner ist auf Gesichter optimiert, lässt sich aber recht leicht für andere Objekte anpassen. Da zur reinen 3D-Rekonstruktion nur ein einziges Bild nötig ist, eignet er sich sogar für bewegte Objekte und allgemein für jedes Szenario, bei dem ein 3D-Scan ohne

große Wartezeit entstehen soll. Durch die intelligente Kombination von Signalverarbeitungsalgorithmen entstehen qualitativ hochwertige 3D-Modelle mit hoher Auflösung – selbst unter Einfluss von starkem Umgebungslicht. (pkr/jk) ■

#### Infos

- [1] D. Scharstein and R. Szeliski, „High-accuracy stereo depth maps using structured light“: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Volume 1, p. 195-202

#### Dynamische Programmierung

Als eine Methode, die Optimierungsprobleme effizient löst, gilt die dynamische Programmierung. Sie zerlegt die Aufgabenstellung in viele kleinere Teilprobleme, die sich einzeln lösen und zur global-optimalen Lösung kombinieren lassen. Wie das funktioniert, verdeutlichen die beiden folgenden Berechnungsalgorithmen für Fibonacci-Folgen, bei denen sich jedes folgende Element aus der Addition seiner beiden Vorgänger ergibt:

```
naiveFibo( x )
    if x < 2 return x
    return naiveFibo(x-1) + naiveFibo(x-2)
dpFibo( x )
    if x < 2 return x
    prev = 0, cur = 1
    repeat n-1 times
        tmp = prev + cur
        prev = cur
        cur = tmp
    return cur
```

»naiveFibo()« berechnet die x-te Fibonacci-Zahl rekursiv. Da jede Rekursionsebene die Werte aller Vorgänger bis zurück zu  $x = 1$  ermittelt, heißt ein solcher Algorithmen naiv. Für  $x = 30$  benötigt er bereits 328 380 Rekursionen.

Bei »dpFibo()« baut dagegen jeder Rechenschritt auf dem Ergebnis des vorigen Schleifendurchlaufs auf. Dazu kehrt der Algorithmus die Blickrichtung um: Statt die beiden Vorgänger für  $x-1$  und  $x-2$  zu berechnen, startet er mit  $x=0$  sowie  $x=1$  und errechnet die gewünschte Zahl Bottom-up. Dabei sind für  $x=30$  lediglich 29 Schleifendurchläufe nötig – die Vorgängerverte, die der naive Algorithmus mühsam über Rekursionen errechnet, braucht der dynamische bei diese Bewegungsrichtung durch die Zahlenfolge nur zwischenspeichern. Auf einem 2-GHz-Laptop dauert der rekursive Algorithmus etwa 6 Sekunden, der dynamische dagegen weniger als 0,5.

Voraussetzung für dynamische Programmierung ist, dass sich das Problem aus gleichartigen, nicht aus voneinander unabhängigen Unterproblemen zusammensetzt. Das Berechnen der Fibonacci-Zahlen ist eine sehr einfache Anwendung der dynamischen Programmierung. Ein anderes Beispiel dafür ist der Quicksort-Algorithmus, der Werte-Sets so oft in Gruppen mit Elementen unterhalb und oberhalb eines mittleren Wertes einteilt, bis die Größe der Gruppen auf 1 geschrumpft ist, die Sortierung also vollständig ist.

- [2] Li Zhang, Brian Curless and Steven M. Seitz, „Rapid shape acquisition using color structured light and multi-pass dynamic programming“: Proceedings of the 1st IEEE International Symposium on 3D Data

- [3] P. Fechteler and P. Eisert, „Adaptive Color Classification for Structured Light Systems“: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Workshop on 3D Face Processing

- [4] P. Fechteler, P. Eisert and J. Rurainsky, „Fast and High Resolution 3D Face Scanning“: Proceedings of the International Conference on Image Processing

- [5] R: [<http://www.r-project.org>]

- [6] Vigna: [<http://kogs-www.informatik.uni-hamburg.de/~koethe/vigna/>]

- [7] Fox-Toolkit: [<http://www.fox-toolkit.org>]

- [8] Olga Ebers (Studienarbeit 2004), „Überblick über aktuelle Verfahren zur Tiefenschätzung aus 2D-Video-Sequenzen“: Technische Universität Berlin, Fakultät Elektrotechnik und Informatik, Institut für Telekommunikationssysteme, Fachgebiet Nachrichtenübertragung

- [9] Peter Eisert, „Model-based Camera Calibration Using Analysis by Synthesis Techniques“: Proceedings of the 7th International Workshop Vision, Modeling and Visualization

#### Der Autor

Philipp Fechteler ist wissenschaftlicher Mitarbeiter am Berliner Fraunhofer Heinrich-Hertz-Institut (HHI). Dort arbeitet er in der Gruppe „Computer Vision & Graphics“ der Abteilung Bildsignalverarbeitung und beschäftigt sich mit Themen rund um Virtual und Augmented Reality, 3D-Rekonstruktion sowie der Analyse und Animation von Gesichtsausdrücken.