

# Deformation Tracking in Depth and Color Video

An Analysis by Synthesis Approach

Andreas Jordt

Dissertation  
zur Erlangung des akademischen Grades  
Doktor der Ingenieurwissenschaften  
(Dr.-Ing.)  
der Technischen Fakultät  
der Christian-Albrechts-Universität zu Kiel  
eingereicht im Jahr 2015

Kiel Computer Science Series (KCSS) 2015/6 v1.0 dated 2015-08-13>

ISSN 2193-6781 (print version)

ISSN 2194-6639 (electronic version)

Electronic version, updates, errata available via <https://www.informatik.uni-kiel.de/kcss>

The author can be contacted via [www.mip.informatik.uni-kiel.de](http://www.mip.informatik.uni-kiel.de)

Published by the Department of Computer Science, Kiel University

Multimedia Information Processing

Please cite as:

- ▷ Andreas Jordt. *Deformation Tracking in Depth and Color Video - An Analysis by Synthesis Approach* Number 2015/6 in Kiel Computer Science Series. Department of Computer Science, 2015. Dissertation, Faculty of Engineering, Kiel University.

```
@book{jordt15-diss,
  author   = {Andreas Jordt},
  title    = {Deformation Tracking in Depth and Color Video
             - An Analysis by Synthesis Approach},
  publisher = {Department of Computer Science, CAU Kiel},
  year     = {2015},
  number   = {2015/6},
  series   = {Kiel Computer Science Series},
  note     = {Dissertation, Faculty of Engineering,
             Kiel University.}
}
```

© 2015 by Andreas Jordt

# About this Series

The Kiel Computer Science Series (KCSS) covers dissertations, habilitation theses, lecture notes, textbooks, surveys, collections, handbooks, etc. written at the Department of Computer Science at Kiel University. It was initiated in 2011 to support authors in the dissemination of their work in electronic and printed form, without restricting their rights to their work. The series provides a unified appearance and aims at high-quality typography. The KCSS is an open access series; all series titles are electronically available free of charge at the department's website. In addition, authors are encouraged to make printed copies available at a reasonable price, typically with a print-on-demand service.

Please visit <http://www.informatik.uni-kiel.de/kcss> for more information, for instructions how to publish in the KCSS, and for access to all existing publications.

1. Gutachter: Prof. Dr.-Ing. Reinhard Koch  
Christian-Albrechts-Universität  
Kiel
2. Gutachter: Prof. Dr.-Ing. Bodo Rosenhahn  
Leibniz Universität  
Hannover

Datum der mündlichen Prüfung: 27. Mai 2015

# Zusammenfassung

Das Verfolgen sich verformender Objekte und die Rekonstruktion dieser Deformationen aus Bildfolgen sind aktuelle Forschungsthemen des maschinellen Sehens. Während sich starre Szenen ohne Verformungen sehr gut analysieren und rekonstruieren lassen, haben allgemeine Deformationen potenziell unendlich viele Unterbewegungen und damit mögliche Arten, diese zu parametrisieren — eine Tatsache, die die konkrete mathematische Formulierung der Zielsetzung sehr erschwert. Anders als bei klassischen Rekonstruktionen, die auf Farbdaten basieren, liefert die Kombination aus Tiefen- und Farbvideos eine zuverlässige Datengrundlage für Verfolgungsalgorithmen, die einerseits weniger Raum für Mehrdeutigkeiten lässt, andererseits aber auch neuer algorithmischer Herangehensweisen bedarf, um mit den unterschiedlichen Beschaffenheiten der Daten umgehen zu können. Diese Arbeit löst das Problem durch eine neue Methode zur Analyse der Deformationsverfolgung, die auf vielfältige Weise von üblichen Rekonstruktionsansätzen abweicht. Es wird gezeigt, dass der Analyse durch Synthese (AbS) Ansatz zur Lösung komplexer Deformationsprobleme geeignet ist, sehr gut mit Verdeckungen und Selbstverdeckungen umgehen kann und die Verfolgung in Echtzeit ermöglicht.



# Abstract

The tracking of deforming objects and the reconstruction of deformation in image sequences is one of the current research areas in computer vision. In contrast to rigid scenes, which can be analyzed and reconstructed very well, general deformations come with an infinite number of sub-movements and ways to parametrize them, which makes it very difficult to formulate discrete tracking goals. In contrast to the classic reconstructions based on color data alone, the combination of depth and color video provides tracking algorithms with a data foundation with less room for ambiguities, but also requires new algorithmic approaches to handle different entities and to exploit the available data. This thesis discusses an Analysis by Synthesis (AbS) scheme as an approach to the deformation tracking problem, a method that differs in many key aspects from common reconstruction schemes. It is demonstrated that AbS based deformation reconstruction can reconstruct complex deformations, deal with occlusions and self-occlusions, and can also be used for real-time tracking.





# Acknowledgements

I would like to address a big thanks to the people who supported me during my research and while writing this thesis. First and foremost I want to thank my mentor Prof. Reinhard Koch who provided me with this research topic and has been a great guide and supervisor. I have had six wonderful years at the Multimedia Information Processing (MIP) Group of the Kiel University in an enjoyable working atmosphere under his supervision. I would also like to thank Renate Staecker and Torge Storm of the MIP group for their continuous help and support and the scientific members and former members of the MIP group (Bogumil Bartczak, Kristine Bauer, Johannes Brünger, Sandro Esquivel, Oliver Fleischmann, Markus Franke, Anatol Frick, Anne Jordt, Daniel Jung, Falko Kellner, Arne Petersen, Stefan Reinhold, Ingo Schiller, Dominik Wolters, Robert Wulff, Claudius Zelenka, Lilian Zhang) for the countless fruitful discussions and help.

In addition, my thanks goes to my colleagues in the IRFO research project (Knud A. Andersen, Leon Bodenhagen, Andreas R. Fugl, Norbert Krüger, Martin M. Olsen, Henrik G. Petersen, Morten Willatzen) for the pleasant and enjoyable collaboration and for always making me feel welcome during our workshops in Denmark. I would also like to thank Jürgen Steimle for providing me with the opportunity to collaborate with the MIT and for creating the very successful FlexPad system with me, pushing the applicational side of my work to a whole new level. I thank Prof. Bodo Rosenhahn for reviewing my dissertation as well as Prof. Thomas Slawig and Prof. Reinhard von Hanxleden for being in my thesis committee.

Finally, a big thanks goes to my parents for always supporting me in my scientific ambitions and to my wife Anne for all the love and support during the writing process as well as for proofreading this thesis.



# Nomenclature

The symbols in this thesis are chosen with the intent to provide good readability, be distinguishable, and to maintain mathematical correctness. All variables are globally defined and keep their function throughout the entire thesis. The glossary contains a list of all symbols used, along with a precise mathematical definition (whenever possible) and a short description. In addition, the symbol fonts allow an intuitive categorization. Table 1 contains a list of fonts, categories and example symbols.

**Table 1.** A list of symbol styles used in this thesis.

<b>Example</b>	<b>Font</b>	<b>Group Description</b>
$u$	Italic Serif	Scalar values
$x$	Normal Serif	Letter values
$\mathbf{v}$	Normal Sans	Vectors $\in \mathbb{R}^n$
$\mathbf{v}$	Bold	Homogenous 3D vectors $\in \mathbb{P}^3$
$V$	Capital Normal	Sets (of any kind)
$\mathbf{A}$	Capital Bold	Matrices $\in \mathbb{R}^{n \times m}$
$\mathcal{F}$	Cap. Calligraphic	Functions

Very few exceptions to this notation style are made if it is supporting readability or maintaining common naming conventions (e. g., using  $\mathbf{P}$  for the probability function). Normal serif letters in an equation mean that these letters are the actual values, not representing a variable name. E.g. if there are two indexed version of a vector  $\mathbf{v}$ , namely  $\mathbf{v}_p$  and  $\mathbf{v}_q$ , these are actually defined instances in contrast to  $\mathbf{v}_s$ , with e. g.,  $s \in \{p, q\}$ .

In some cases, it is notation-wise convenient to directly access the components of a vector. In the case of a vector in  $\mathbb{R}^3$  or  $\mathbb{P}^3$ , the x-, y- and z-component will be notated via the corresponding index, i. e., the x-component of a vector  $\mathbf{v}$  will be notated as  $\mathbf{v}_x$ . The same holds for vectors in the  $\mathbb{R}^2$ , only that image

coordinates may be addressed via u- and v-index rather than x and y. The access of multiple elements at once, e. g., denoting the x- and y- components of a vector  $\mathbf{v}$  in  $\mathbb{R}^3$  will be notated as  $v_{(x,y)}$ .

Rotations in 3D space are given by matrices  $\mathbf{R} \in \mathbb{R}^{3 \times 3}$ . For rotations around the x-, y-, and z-axis, let  $\mathbf{R}_x$ ,  $\mathbf{R}_y$ , and  $\mathbf{R}_z$  be defined, such that

$$\mathbf{R}_x : \alpha \mapsto \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix},$$

$$\mathbf{R}_y : \alpha \mapsto \begin{pmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix},$$

and

$$\mathbf{R}_z : \alpha \mapsto \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

### Evaluation Preview

As the system introduced in this thesis is made out of various exchangeable components that, in most cases, can only be tested in conjunction, evaluation results are presented in Chapter 7, after the main components have been introduced. To nevertheless provide an idea about the capabilities of each component and the improvement gained by the tweaks described in each section, text boxes like this one will contain very short previews of the test results and of how the component at hand impacts the performance.





# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Related Work . . . . .	3
1.1.1	Feature Based Deformation Tracking . . . . .	3
1.1.2	Tracking without Features . . . . .	5
1.2	Overview . . . . .	7
<b>2</b>	<b>Sensing Depth</b>	<b>9</b>
2.1	Triangulation . . . . .	9
2.2	Active Range Cameras . . . . .	11
2.2.1	Structured Light . . . . .	11
2.2.2	Time-of-Flight Cameras . . . . .	13
2.2.3	Calibration . . . . .	15
<b>3</b>	<b>NURBS Deformation Models</b>	<b>17</b>
3.1	Non-Uniform Rational B-Splines . . . . .	18
3.2	Object Registration . . . . .	20
3.2.1	The <i>uvo</i> -Space . . . . .	20
3.2.2	NURBS Initialization . . . . .	22
3.3	Fast Evaluation Cache . . . . .	25
<b>4</b>	<b>Analysis by Synthesis</b>	<b>29</b>
4.1	Synthesis . . . . .	30
4.1.1	Geometric Projection . . . . .	31
4.1.2	Lens Distortion . . . . .	34
4.1.3	Rendering . . . . .	35
4.2	Analysis . . . . .	36
4.2.1	Color Error . . . . .	37
4.2.2	Depth Error . . . . .	38

## Contents

4.2.3	Stretch Constraint . . . . .	40
4.2.4	Error Fusion . . . . .	41
4.3	Prior Knowledge . . . . .	42
4.3.1	Spatial Constraints . . . . .	42
4.3.2	Temporal Constraints . . . . .	43
4.4	Sparsification . . . . .	44
4.4.1	Sparse Synthesis . . . . .	44
4.4.2	Randomized Online Sparsification . . . . .	48
<b>5</b>	<b>Optimization</b>	<b>51</b>
5.1	CMA-ES . . . . .	52
5.1.1	Covariance Matrix Adaptation . . . . .	53
5.1.2	Update Strategies and Evolution Path . . . . .	57
5.2	Accelerating CMA-ES with Parameter Space Knowledge . . . . .	63
5.2.1	Local Approximation of Constraints . . . . .	64
5.2.2	Optimized Initial Covariance Adaptation . . . . .	65
<b>6</b>	<b>Weighting Strategies</b>	<b>69</b>
6.1	Heuristic Adjustment . . . . .	69
6.2	Probabilistic Problem Formulations . . . . .	72
6.2.1	Polynomial Interpolated Gaussian Distribution . . . . .	73
<b>7</b>	<b>Evaluation</b>	<b>79</b>
7.1	Experiments on Synthetic Data . . . . .	80
7.1.1	Experiments on Complex Synthetic Geometry . . . . .	80
7.1.2	Experiments on Incomplete Synthetic Data . . . . .	81
7.2	Experiments on Real Data . . . . .	83
7.2.1	Real Data Experiments with Complex Object . . . . .	84
7.3	Relative Performance Experiments . . . . .	84
7.3.1	Real-time Tracking Experiments . . . . .	86
7.3.2	Scalability . . . . .	88
7.4	Random Sparsification Experiments . . . . .	91
7.4.1	Sparsification Noise . . . . .	91
7.5	Evaluation of Probabilistic Fitness . . . . .	95
7.5.1	Probabilistic Fitness on Synthetic Data . . . . .	95
7.5.2	Probabilistic Fitness on Real Data . . . . .	96



<b>8 Occlusion Handling</b>	<b>99</b>
8.1 Self-Occlusion Handling . . . . .	100
8.1.1 Noise Adaptation Heuristic . . . . .	101
8.1.2 Enforcing Contour Discontinuities . . . . .	103
8.2 Model-Based Occlusion Handling . . . . .	103
8.2.1 Assumptions and Models . . . . .	104
8.2.2 Evaluation . . . . .	106
8.3 Image Based Occlusion Handling . . . . .	110
8.3.1 FlexPad Material Filter . . . . .	111
8.3.2 Evaluation Example . . . . .	114
<b>9 Specialized Deformation Functions</b>	<b>117</b>
9.1 Rigid Transformations . . . . .	117
9.1.1 Rotation Parametrization . . . . .	118
9.1.2 Evaluation . . . . .	119
9.2 Estimating Physical Parameters . . . . .	119
9.2.1 Curvature Based Deformation . . . . .	119
9.2.2 Physical Plausibility Penalty . . . . .	121
9.2.3 Evaluation . . . . .	122
9.3 Real-Time Paper Tracking . . . . .	126
9.3.1 Paper Deformation Model . . . . .	126
9.3.2 Evaluation . . . . .	129
<b>10 Conclusion</b>	<b>131</b>
10.1 Traits of AbS . . . . .	131
10.2 Future Work . . . . .	132
<b>A CMA-ES</b>	<b>135</b>
A.1 Initialization . . . . .	135
A.2 Update . . . . .	137
A.3 Bootstrapped Initialization . . . . .	138
<b>B Brightness Normalization of the FlexPad System</b>	<b>141</b>
B.1 Surface Normal Compensation . . . . .	141
B.2 Distance Compensation . . . . .	143
<b>C Effects of Random Sparsification</b>	<b>145</b>
<b>Glossary</b>	<b>151</b>

Contents

**Acronyms** 159

**Bibliography** 161

# List of Figures

1.1	An overview showing the components and the data processing of the basic AbS systems discussed in this thesis. The functions and variables are introduced in the chapters in which the components are explained in detail. . . . .	8
2.1	Triangulation of three equally distributed points (green dots) does not result in equally distributed projections (red marks).	10
2.2	The same amount of uncertainty in the image plane (red area) can lead to large differences in the uncertainty when propagated (gray area around the black rays) to the reconstructed 3D point (green area). . . . .	10
2.3	A hierarchical classification of active range cameras. . . . .	11
2.4	The pattern of the <i>Kinect</i> in the infrared domain is made from randomly distributed dots. . . . .	12
2.5	Optical-shutter based ToF works by synchronously emitting light and opening a shutter on the sensor. Right: accumulated signal and incoming signal strength over time. . . . .	13
2.6	Continuous-wave ToF cameras work by emitting sinusoidally amplitude-modulated light and synchronized sensing. Right: synchronized measurement of the four phases and the incoming signal strength over time. . . . .	15
2.7	Images acquired from a ToF camera (Mesa Imaging SR 4000). Left: the depth image contains a distance value in each pixel. Right: the intensity image contains for each pixel the amplitude of the signal that was used to render the distance value provided in the depth image. . . . .	15

## List of Figures

3.1	A 2D example of a NURBS function with a 1D parameter space (NURBS line). Left: weighting functions $\mathcal{R}$ in the parameter space for every control point $\mathbf{p} \in P$ with example evaluation $u \in [0, 1]$ and entries of knot vector $\mathbf{k}$ . Right: the resulting 2D line in world coordinates $\mathcal{B}(\cdot, P)$ , shaped by the positions of the control points $\mathbf{p}$ . The example evaluation $\mathcal{B}(u, P)$ is depicted as the corresponding weighted linear combination of control point positions (purple lines). The thickness of each purple lines visualizes the influence of the control point given by the weighting for parameter $u$ . . . . .	19
3.2	A vertex $\mathbf{v}$ described by its position in the Euclidean space $(x, y, z)$ (left) and by its position $(u, v, o)$ relative to the NURBS $\mathcal{B}(\cdot, P)$ (right). . . . .	21
3.3	An example deformation of a triangle mesh using a NURBS deformation function (green) defined by $P$ (small gray dots): a) the mesh in its original state, b) the mesh subject to a mostly affine transformation, c) the mesh subject to a strong deformation. . . . .	22
3.4	A triangle mesh and a NURBS deformation function. Left: a purely affine initialization. Right: an affine initialization followed by an optimization of $P$ in respect to (3.2.5). . . . .	23
3.5	Performance comparison between cache generation, uncached, and cached NURBS function evaluations. Left: 10,000 function evaluations of a second degree NURBS function for various knot counts. Cached evaluation only takes around 1 ms (1.05 ms - 1.11 ms) Right: 10,000 function evaluations of a fourth degree NURBS function for various knot counts. Cached evaluation only takes around 2.5 ms (2.52 ms - 2.58 ms). . . . .	26
4.1	An overview over the analysis by synthesis approach. . . . .	31
4.2	Different triangle representations. Left: set of separate triangles stores redundant information if the triangles form a closed surface. Right: an indexed set of vertices contains the same triangle mesh information in a more efficient way. . . . .	32
4.3	A visualization of the pinhole projection and the camera coordinate system. The camera coordinate system is originated at the camera center $\mathbf{c}_s$ , the image plane is parallel to the x-y-plane of the camera coordinate system. . . . .	33

4.4	An example input image and the corresponding synthesis of the deformed object. . . . .	36
4.5	Left: an example Bayer pattern (GRBG) as it is used for CCD chips in cameras. Center: the result of the NN demosaicing algorithm shows block artifacts and colored edges. Right: the demosaicing algorithm AHD. . . . .	38
4.6	Warping a depth image into another depth image can be performed by back-projecting [HZ00] each pixel into 3D space and projecting it into the new camera center. . . . .	46
4.7	Warping a color image into a depth image can be performed by back-projecting each pixel of the destination depth image into 3D space and projecting it into the color camera center to associate a color value. . . . .	47
5.1	Example visualization of fitness function $\mathcal{F}$ (dark= low fitness, bright= high fitness). Left: initial distribution provided by mean $\mathbf{m}_0$ and the normalized covariance $\mathbf{C}_0$ scaled by distribution size $\sigma_0$ . Right (zoomed in): sample population of $\lambda$ parameter vectors $\Theta$ , distributed according to $\mathcal{N}(\mathbf{m}, \sigma, \mathbf{C})$ . . .	54
5.2	Example visualization of fitness function $\mathcal{F}$ (dark= low fitness, bright= high fitness). Left: selection of $\mu$ out of the $\lambda$ individuals based on their fitness $\mathcal{F}(\Theta)$ (green= selected, red= not selected). Right: calculation of a new mean $\mathbf{m}_{i+1}$ as the weighted sum of the $\mu$ selected samples. . . . .	55
5.3	Visualization of the covariance matrix update: Left: calculation based on the new mean $\mathbf{m}_{i+1}$ . Right: a covariance based on the old mean vector $\mathbf{m}_i$ yields a much better distribution for the following search iteration. . . . .	57
5.4	Visualization of the rank- $\mu$ update (left) compared to the rank-one update (right). Left: the distribution $\mathbf{C}_{i+1}^{\text{exp}}$ (green ellipse) leads to a widespread search in the parameter space. Right: faster but less exhaustive search is performed by the distribution $\mathbf{C}_{i+1}^{\text{evo}}$ (red ellipse), which is only based on the evolution path $\mathbf{e}_{i+1}$ (orange arrow), calculated from the recent mean movements (red trail). . . . .	60

List of Figures

5.5	Visualization of the anisotropic evolution path $\hat{e}$ (calculated from $\mathbf{m}_0, \dots, \mathbf{m}_{i+1}$ ) in different situations. Left: if the optimization steps continuously go in similar directions, the evolution path $\hat{e}$ is long, a hint to increase the distribution size $\sigma$ to save iterations. Center: uncorrelated movement directions lead to a medium sized evolution path $\hat{e}$ , indicating a proper search. Right: oscillating movements cause the evolution path to become small, a strong indication to reduce the distribution size $\sigma$ . . . . .	61
5.6	Example visualization of fitness function $\mathcal{F}$ (dark= low fitness, bright= high fitness). The blue line depicts the sub-manifold in which the stretch constraint is minimal. Left: in early iterations the distribution mean is not close to the optimum and the distribution size $\sigma$ is large. Right: when closing in on the optimum, the mean movements become smaller and/or anticorrelated (red lines) and the distribution size $\sigma$ starts shrinking. The evolution path $e$ shrinks as well but still contains the direction of the most recent, large steps. . . . .	64
5.7	Example visualization of fitness function $\mathcal{F}$ (dark= low fitness, bright= high fitness). The blue line depicts the sub-manifold in which the stretch constraint is minimal. Left: standard initialization of CMA-ES with $\mathbf{C}$ being a diagonal matrix. Right: bootstrapping $\mathbf{C}$ with the knowledge about the constraints and mean movements leads to a superior initial distribution. . . . .	67
6.1	Two examples of different entities providing the required data for the tracking: Top: the white, reflecting object on a white background can only be tracked properly because of the depth information, while the color information is rather useless. Bottom: a textured object lying flat on the ground is not well traceable in the depth input data. The color data provides the required information for tracking. . . . .	70
6.2	Plot of the average noise (standard deviation in mm) as it was measured by a central pixel for various distances (ground truth) and amplitude values as returned by an <i>SR4000</i> (distance compensated). The height is depicting the noise levels. For visualization purposes, the measurements have been connected to a 3D surface. . . . .	75

7.1	A synthetically generated, complex geometry, deformed and rendered as depth and color image sequence. The first line depicts the rendered color data, the second line shows the synthetic errors (invalid values) added to the input data to evaluate the stability of the algorithm on incomplete data. . .	80
7.2	An artificial object model with a complex geometry made from various boxes, rings, noisy surfaces, and scanned objects. . . .	81
7.3	A 3D mesh reconstructed from a <i>Kinect</i> color and depth image. The close distance to the <i>Kinect</i> causes holes in the depth information. . . . .	82
7.4	The RMS distance between the original deformed object vertices and the tracked object vertices for the 500 frame input sequence.	82
7.5	Pictures of a color and depth video sequence of a bag recorded with a <i>Kinect</i> camera. Top row: the color image output; center row: the depth image output; bottom row: a closeup of the resulting colored 3D mesh together with the NURBS surface (green) following the deformation of the bag. . . . .	83
7.6	Pictures of a deforming 3D mesh. The underlying deformation sequence was used to generate an artificial input data with known ground truth. The depicted sequence was acquired by the the algorithm discussed in Section 8.2. . . . .	84
7.7	Image sequence of a deformed teddy bear. The upper row shows the <i>Kinect</i> color image, the bottom row displays the deformed mesh that was retrieved from the first depth/color image pair of the sequence. . . . .	85
7.8	The RMS depth error for all vertices over all frames of the sequence depicted in Fig. 7.11 using 40 CMA-ES iterations. The standard initialization (red line) is compared to the propagation scheme introduced in Section 5.2 (green line). . . . .	86
7.9	The RMS depth error for all vertices of the sequence depicted in Fig. 7.11 using the standard initialization and every third frame with 60 CMA-ES iterations (red line). The green line shows the results for the new initialization method using every frame and only 20 CMA-ES iterations. . . . .	87

List of Figures

7.10	The average fitness value $f$ over all frames of a 500 frame sequence similar to 7.11 using 40 CMA-ES iterations (24 individuals). The red line shows the values if the covariance is not propagated, the green line shows the values of the same sequence using covariance propagation. Higher values stand for a lower error in this plot. . . . .	88
7.11	Top row: video sequence recorded with a <i>Kinect</i> camera of a flexible white object on white background. Center row: the resulting colored 3D mesh with the NURBS deformation function (green grid) and its control points (gray dots). Bottom row: deformation function of the input sequence applied to a box. . . . .	89
7.12	The input sequence of the deforming paper is processed in real-time with 25 Hz. The first row depicts close-ups of the color images of the input data, the second row the estimated object surface. . . . .	90
7.13	Average depth- (left) and color (right) errors in optimization runs using different random sparsification rates ranging from use of 100% of the vertices (red) to 1% (grey). 150 iterations and 16 CMA-ES individuals were used. For each iteration the (non-sparse!) depth and color error value is depicted to visualize the influence of sparsification on the real error. The error values do not refer to an absolute scale but serve the purpose of relative comparison only. . . . .	92
7.14	Average depth- (left) and color (right) errors in optimization runs using different random sparsification rates ranging from use of 100% of the vertices (red) to 1% (grey). In contrast to Figure 7.13, sparsified runs are plotted over the equivalent amount of computational work. As an example, iteration one without sparsification (red) is depicted with the same x value as iteration five using only 20% (blue) of the vertices. The amounts of computational work equivalent to 15 iterations without sparsification is depicted, using 16 CMA-ES individuals. The error values are calculated without sparsification to visualize the influence of sparsification on the real error. The error values do not refer to an absolute scale but serve the purpose of relative comparison only. . . . .	93



7.15	Noisy input data used in the online sparsification test. Left: depth and color input images of a simple box shape. Right: depth and color input images of the Stanford bunny model. . . . .	93
7.16	Average depth- (left) and color (right) errors in optimization runs using different random sparsification rates equivalent to the one depicted on Figure 7.13. . . . .	94
7.17	Average depth- (left) and color (right) errors in optimization runs using different random sparsification rates ranging from use of 100% of the vertices (red) to 1% (grey) equivalent to the one depicted on Figure 7.14. . . . .	94
7.18	The sum of depth and color errors as shown individually in Fig. 7.13 - Fig. 7.17 with an additional plot showing the result of (linearly) adjusted sparsification for the Box scenario (left) and the bunny sequence (right). . . . .	95
7.19	Left: a 3D representation of an artificial deformation sequence showing the virtual camera and the deforming object. Center: the depth image of the artificial rendering using projection parameters of a <i>Mesa Imaging SR4000</i> . Right: the depth image augmented with noise simulating input data of an <i>Mesa Imaging SR4000</i> . . . . .	96
7.20	The two test objects used in the real data tests. Left: the checker board has equally distributed areas of high and low reflection, Right: the black area in the center of the tracking object provides a special challenge for the tracking algorithm since the black area will cause an offset in the depth image. . . . .	97
8.1	Input material (left: color image, right: depth image) of a deformation sequence containing external occlusion (hands) as well as self-occlusion. . . . .	99
8.2	Illustration of the “edge problem” on a synthesis using z-buffer. Left: a flat object is projected onto a depth image. Right: if rotated to the edge, all projected pixels of the object can still map the same depth values in the image although the object pose does not represent the depth image content. The red pixels represent the reference pixel “missed” by the synthesis. . . . .	100

## List of Figures

8.3	Removing occlusions from the input data. Left: input color image and depth image. Right: 3D representation of the input data from which possible occlusion pixels have been removed. A green grid visualizes the deformation function of the tracking algorithm. . . . .	105
8.4	A synthetically generated, complex geometry, deformed and rendered as depth and color image sequence. The first line depicts the rendered color data, the second line shows the same data augmented by occluding boxes to evaluate the stability of the algorithm on occluded data. . . . .	106
8.5	The RMS distance between the original deformed object vertices and the tracked object vertices for the 500 frame input sequence.	107
8.6	Test input of the occlusion handling evaluation on real data using a bag manipulated by hands. The upper images in each row depict the color input data. The lower images show the corresponding depth images. . . . .	108
8.7	Test results of the occlusion handling method on real data using a bag manipulated by hands. The upper images in each row depict a 3D visualization of the depth and color input data from which image data was removed that could potentially occlude the object. The green grid visualizes the NURBS surface of the deformation function at its current state. The lower images depict the resulting deformation of the reference 3D mesh model.	109
8.8	The RMS depth error for all visible vertices over all frames of the bag-turning-sequence depicted in Fig. 7.6. . . . .	110
8.9	Three different materials captured by the <i>Kinect</i> IR camera. Left: skin, center: paper, right: dark cloth. . . . .	111
8.10	Left: photo of the FlexPad setup: a video projector and a <i>Kinect</i> sensor are mounted on the ceiling overlooking the interaction area. Right: a schematic view of the FlexPad setup. . . . .	112
8.11	A schematic overview over the material filter. From left to right: the IR input is analyzed by the intensity and gradient thresholds, yielding a mask image. The mask is applied to the depth input data and filtered by the convex-hull-fill filter. . .	113
8.12	Visualization of the interpolation filter, filling the local convex hull of already classified pixels. The filter is applied multiple times to generate a dense object area. The wrist parts in rows 1, 4, and 5 are not classified as skin because these were covered by cloth (sleeves). . . . .	114

8.13 6 input image pairs from the skin removal test. Top row: the input depth image. Center row: the input IR image. Bottom row: the resulting depth image in which the skin pixels have been removed. Note that the areas in which fingers might be classified wrong (e.g., too short) due to the convex-hull-fill filter, the actual depth value is the interpolated object distance, not the distance to the occluding hand. . . . . 115

9.1 A rendering of a flexible object held by a robot gripper. The difference in deflection between the undeformed object shape (red, transparent) and the picked up shape (gray, solid) allows to deduce material properties of the object. . . . . 120

9.2 An example deformation: the box is separated into one Section without deflection (white) and several Sections of varying curvature (colored). The undeformed box (upper left corner) is subject to varying deformations. In each Section, the applied curvature is constant (lower left). In addition, a global twist of the object can be applied (right) . . . . . 121

9.3 The FlexPad deformation space contains 8 non-linear bending deformations, parametrized by  $\mathbf{b}_0 - \mathbf{b}_7$ . Each deformation is bending half an object. . . . . 126

9.4 Left: a graph depicting the positive range of the z-mapping function. It shows the function that is applied to every z component  $v_z$  of each vertex  $\mathbf{v}$  (after the initial deformation  $\mathcal{D}'$ ), for  $\tau = -1$ ,  $\tau = 0$ , and  $\tau = 1$ . Right: a resulting example shape (x-z-plane only). Assuming an initial bending strength  $\mathbf{b}_1 = 1$ , this graph depicts results as deformed by  $\mathcal{D}''$  for  $\tau = -1$ ,  $\tau = 0$ , and  $\tau = 1$ . Figure 9.5 depicts 3D renderings of these examples. . . . . 127

9.5 Illustration of the z-mapping effect on the 3D mesh, already deformed by  $\mathcal{D}'$  with  $\mathbf{b}_1 = 1$ . A high  $\tau$  parameter raises the Section between the bended edge and the object center, while a low  $\tau$  lowers this area and translates the bending deformation to the object edge. . . . . 128

9.6 Example deformations visualizing the versatility of the 9 + 6 parameter FlexPad deformation model. . . . . 129

B.1 Pictogram of the components involved in the Phong lighting model. . . . . 142

List of Figures

C.1 Left: the fitness function maps all distributed individuals (green and red dots) to their fitness value (green and red lines). The distribution within the codomain of  $\mathcal{F}$  is notated as  $\|\sigma\mathbf{C}\|_{\mathcal{F}}$ . Right: the fitness function using sparsification  $\mathcal{F}_s$  adds an additional uncertainty to the fitness values  $\mathcal{F}_s(\Theta)$ , which can change the order of individuals and, hence, their weighting. . . . . 146

C.2 Left: two distributions of sparsification noise  $\sigma_s$  and the difference  $d$  between the expected values. The distribution overlap visualizes the events in which the sparsification noise causes a change in order, with its probability  $\mathbf{p}_{\text{change}}$  (Note:  $\mathbf{p}_{\text{change}}$  is plotted for intuitive visualization only and is not the numerically correct multiplication of the PDFs) Right: the same probability value can be calculated by reducing one distribution to a single result (red) and adding the subtracted variance to the other distribution (green). . . . . 147

# List of Tables

1	A list of symbol styles used in this thesis. . . . .	xi
7.1	Test results comparing the Euclidean fitness function to the likelihood fitness. The cells contain the average error and the standard deviation put in brackets. The rows contain results for the tests runs using the checker board (Fig. 7.20 left) and the stripe board (Fig. 7.20 right) as tracking target. The upper table contains the average deviation in mm using the NURBS deformation model, the bottom table contains the average maximum bending parameter using the FlexPad deformation model. . . . .	98
9.1	RMS Error values of relevant parameters on an experiment using a low stiffness object ( $E = 0.5 \text{ N/mm}^2$ ). . . . .	123
9.2	RMS Error values of relevant parameters on an experiment using a low stiffness object ( $E = 2.0 \text{ N/mm}^2$ ). . . . .	124
9.3	RMS Error values of relevant parameters on an experiment using a high stiffness object ( $E = 8.0 \text{ N/mm}^2$ ). . . . .	124
9.4	Test results on real objects: standard deviation of parameters for varying object thicknesses. . . . .	125
9.5	Pixel-wise RMS difference (error) between <i>Kinect</i> measurement and synthesized model for different shapes acquired at 90 cm and 150 cm object-to-sensor distance. The standard deviation of each error value is given in brackets. . . . .	130



# Introduction

The analysis and comprehension of real world events and objects via optical sensors is the focus of computer vision research, and reconstructing geometry and tracking movements have been subject to intense research in the last 40 years and continue to be so. Established approaches to geometric reconstruction from color camera footage, such as Structure from Motion (SfM), rely on the extraction and recognition of significant image patches (features). In a second step linear equation systems are formulated with the goal to associate these image patches with real world 3D points in a way that fits the extracted observation information, reconstructing the 3D geometry of the observed scene along the way. Many investigations have been made on the various parts of the SfM framework to improve all aspects of the algorithm. Even the inherent restriction to static environments has been loosened in the last fifteen years with the introducing of Non-Rigid Structure from Motion (NRSfM) as a way to deal with movements and deformations in a scene. Although these methods are very capable in theory, the lack of scene rigidity, which is the fundamental constraint SfM is based on, restricts their application to clean input data, for example by only using strong or hand selected feature points without outliers and complete feature trails throughout the sequence, etc.

The introduction and commercialization of depth cameras in the past ten to twenty years has led to research efforts causing depth-image based reconstruction of rigid scenes to become well understood, as one depth image provides enough information to reconstruct the visible geometry of a scene and succeeding depth images are easily and accurately registered to this geometry. But again, movements and deformations violate the underlying rigidity assumption, and deformation reconstruction based solely on depth images is an ill-posed problem. A sensible conclusion is to combine color and depth images to be able to avoid the under-determination of the problem

## 1. Introduction

formulation by providing lateral (texture) information simultaneously to depth data. But as most existing deformation reconstruction methods are still based on approaches derived from the entity-related restrictions and data fusion is not straight forward, the number of methods for depth and color based reconstruction is low. Hence, joint reconstruction from depth and color might benefit from fundamentally rethinking the way of approaching the problem.

The research documented in this thesis aims at doing so and provides an alternative framework for the reconstruction of scenes and objects that are moving and deforming along the way. Many aspects of this research have already been published by the author in the following articles:

- ▷ **High Resolution Object Deformation Reconstruction with Active Range Sensor** [JSBK10] by *Andreas Jordt, Ingo Schiller, Johannes Bruenger, and Reinhard Koch*, presented at the DAGM 2010. The article describes a feature-driven approach to deformation tracking using depth sensors.
- ▷ **Fast Tracking of Deformable Objects in Depth and Colour Video** [JK11] by *Andreas Jordt and Reinhard Koch*, presented at the BMVC 2011. The article introduces the Analysis by Synthesis (AbS) approach to deformation tracking for the first time.
- ▷ **An Outline for an intelligent System performing Peg-in-Hole Actions with flexible Objects** [JFB<sup>+</sup>11] by *Andreas Jordt, Andreas R. Fugl, Leon Bodenhausen, Morten Willatzen, Reinhard Koch, Henrik G. Petersen, Knud A. Andersen, Martin M. Olsen, and Norbert Krueger*, presented at the ICIRA 2011. The article shows the concept of AbS-based deformation tracking in the context of robotics for flexible object handling.
- ▷ **Estimation of Material Properties and Pose for Deformable Objects from Depth and Color Images** [FJP<sup>+</sup>12] by *Andreas R. Fugl, Andreas Jordt, Henrik G. Petersen, Morten Willatzen, and Reinhard Koch* presented at the DAGM/OAGM 2012. The article describes how AbS-based deformation reconstruction can be used to estimate material parameters of objects using optical sensors only.
- ▷ **Direct Model-based Tracking of 3D Object Deformations in Depth and Color Video** [JK12] by *Andreas Jordt and Reinhard Koch* published in the IJCV in 2012. The article gives an introduction to AbS-based deformation tracking and introduces an addaptive weighting strategy.



- ▷ **Flexpad: Highly Flexible Bending Interactions for Projected Hand-held Displays** [SJM13] by *Jürgen Steimle, Andreas Jordt, Pattie Maes* presented at the CHI 2013. The article introduces a system turning a sheet of paper into a flexible display using a video projector and a depth camera.
- ▷ **Reconstruction of Deformation from Depth and Color Video with Explicit Noise Models** [JK13] by *Andreas Jordt* and *Reinhard Koch* published in the book “Time-of-Flight and Depth Imaging: Sensors, Algorithms, and Applications” (LNCS 8200). The article shows how a probabilistic fitness function design can provide noise handling and automatic weighting for AbS-based deformation tracking.
- ▷ The patent **Verfahren zur echtzeitfähigen materialschätzung und zur materialbasierten Bildsegmentierung in elektronischen Bildsequenzen** [JSK13] by *Andreas Jordt, Jürgen Steimle, and Reinhard Koch* describes in detail how the FlexPad system [SJM13] manages to separate display surface from hands in the depth input for the AbS-based deformation tracking.

The following section will provide an overview on the existing work and will derive the motivation for the research of the remainder of this thesis.

## 1.1 Related Work

Although there are several algorithms in deformation reconstruction using silhouette information [CBK05, RKP<sup>+</sup>07, GSA<sup>+</sup>09], patchlet based reconstruction [DATSS07], or optical flow [HE09, dAST<sup>+</sup>08], most of the work on deformation tracking has been done utilizing feature points. Correspondences tracked throughout an image sequence provide a comfortable data foundation to formulate mathematically sound solutions to a tracking problem. A well known class of feature-based approaches that has been studied for over a decade is NRSfM [DSA07].

### 1.1.1 Feature Based Deformation Tracking

NRSfM is based on the factorization method, Carlo Tomasi and Takeo Kanade introduced in 1992 [TK92], which allows to simultaneously recover 3D shape and motion from a set of 2D feature correspondences, the SfM problem [HZ00], which was extended to the non-rigid case over the years. The essential SfM

## 1. Introduction

constraint, the assumption of a constant scene, was first relaxed to 'piece-wise' rigidity by Kanade et al. in 1994 [CK94], who allowed multiple moving elements by segmenting the features into separate movements.

In 2000, Bregler et al. [BHB00] introduced the NRSfM method, extending the pose space by a set of linear deformation dimensions spanned by basis shapes, so the factorization yields a weight to each deformation basis shape for every feature point in addition to its 3D position. To circumvent the problem of manually providing the set of basis shapes (for example by selecting a subset of images spanning the deformation space, see [BB98]) Torresani et al. [TYAB01] published a method for handling the ambiguities arising when the complete image set is used to span the deformation space by introducing rank constraints to the tracking matrix. Another approach was introduced in 2003 by Torresani et al. [SHB03], learning the deformation space based on the assumption that the shapes are Gaussian distributed around a mean shape. In 2006, Del Bue et al. [DBA06] extended the NRSfM approach to the stereo case.

A clear limitation to the linear basis shapes is the inability to model non-linear deformations like bending motions. In 2009, Fayad et al. [FDAA09] presented an extension to NRSfM by introducing bi-cubic basis shape deformations; however, the factorization method could not be extended to this case and was replaced by a non-linear optimization method. To handle even more complex deformations, the method was extended to a piecewise-quadratic reconstruction algorithm [FAB10]. A similar idea was introduced by Taylor et al. [TJK10] by dividing the NRSfM problem into multiple independent and rigid three-point reconstruction problems, connecting the independent solution afterwards in 3D space. A problem arising from splitting the reconstruction into many local sub-problems like this is the assignment of each feature point to a sub-problem. This task was addressed and formulated as a labeling problem by Russel et al. [RFA11].

These methods [FDAA09, FAB10, RFA11] surely blur the line between NRSfM and other feature based approaches since they do not use the classic factorization that is characteristic to NRSfM methods. Apart from NRSfM, many tracking algorithms for deformable objects have been introduced over the years:

In 1991, Delingette et al. [DHI91] introduced a reconstruction method for deforming surfaces, combining different types of features to a physically motivated energy term that is minimized by deforming a spherical object prototype. A similarly physically inspired approach was introduced by McInerney and Terzopoulos in 1993 [MT93], who used a finite element method to fit the

deformation model to the observed feature movements. A general problem of these approaches is the spatial smoothness enforced by the energy term that handles the noise along with the ambiguity problem (a theoretical discussion of these problems can be found in [SLF07]). In 2007, Salzmann et al. [SHF07] circumvent the spatial smoothness prior by introducing a temporal smoothness term along with a manually initialized surface mesh. In their method they directly minimized reprojection errors of the tracked features using Second Order Cone Programming (SOCP) [AG01]. Zhu et al. [ZHXL08] reformulated the task addressed by Salzmann as a convex optimization feasibility problem allowing a fast computation of the solution.

In the far less ambiguous stereo case, the initial shape can be estimated [PLF08], and correspondences can directly be tracked in 3D, as done by the extension of Salzmanns SOCP formulation to the stereo case by Shen et al. [SZL08, SMSL10]. Cagniart et al. [CBI09] proposed a mesh deformation method that does not track feature points over time but retrieves unregistered 3D points from a multiple camera setup using an optimization method similar to the Iterative Closest Point (ICP) algorithm [BM92].

A general problem of the feature based approaches is the limitation of the image data to features and the resulting decomposition of the processing into two stages: (1) the tracking of feature points and correspondences in the image and (2) the processing of correspondences into 3D pose, movement and deformation data. This decomposition entails three major issues.

- ▷ After step (1), most of the image information is discarded and does not influence the final result. The final solution does not necessarily fit the whole image information, only the extracted features including their inaccuracies.
- ▷ If there are multiple choices for a correspondence in (1), only the best match is passed to step (2). If this correspondence proves to be an outlier, it is discarded without regarding other possible solutions for a feature match.
- ▷ The inevitable generation of outliers in step (1) requires the reconstruction algorithm in step (2) to be able to handle these outliers. This entails an inferior convergence behavior of the overall algorithm since observations that do not fit the current estimate are more likely to be considered as an outlier than used to correct the estimate.

## 1.1.2 Tracking without Features

Another utilization of the image data can be achieved using silhouette information, which either requires a known background for subtraction [CBK05,

## 1. Introduction

RKP<sup>+</sup>07, GSA<sup>+</sup>09], or is subject to potential outliers similar to tracked feature points. More image information is used, when the optical flow field of an image is regarded. Hilsmann et al. [HE09] use the optical flow to deform an a priori known 3D triangle mesh in real-time from monocular video. Aguiar et al. [dAST<sup>+</sup>08, dATSS07] deform triangle meshes based on the scene flow [VBCK99] generated from the optical flow in the images of eight cameras.

However, optical flow and scene flow based methods suffer from a similar handicap as feature based methods. After the flow is estimated, the color information is discarded and not considered in the following process. In addition, dense optical flow assigns “interpolated” movement vectors to areas of constant color, although there might not be information about the real movement. In the following processing steps, it is not possible to determine the quality of the movement vectors. Furthermore, only one movement is assigned to each pixel, even for image sequences allowing various interpretations.

The most extensive usage of the input data is performed by direct methods, i. e., methods optimizing directly in the input data domain. AbS is such a direct approach. Applied to the deformation tracking problem it analyzes the likelihood of a certain deformation or movement estimate by synthesizing the input images for a solution candidate. The candidate’s fitness is judged by the difference between the synthesis and the real input data, thus avoiding the detour of computing the flow field or feature points explicitly.

One of the first works using analysis by image synthesis was done in 1985 by Netravali and Salz [NS85] for color images and in 1990 by Horn and Harris [HH91] for range images. They performed pose estimation by searching for the rigid transformation (six Degrees of Freedom (DoF)) that would map the current depth image as closely as possible into the succeeding one. A similar approach for 3D data without projections was proposed by Besl et al. [BM92] in 1992. It searches for the six DoF transformation between two surfaces by iteratively minimizing the Root Mean Square (RMS) distance between pairs of points from either surface closest to each other. This method became very popular as the ICP algorithm. A first AbS approach to deforming models was proposed by Koch [Koc93] in 1993 for the special case of modeling the upper part of the human body. An approach using generic deformable surfaces retrieved from range images was introduced by Yamamoto et al. [YBBR93]. They calculated a 3D range flow using an approach similar to Horn and Harris [HH91], but with a deformable net model that allowed the surface to deform locally by a range flow induced by the movement of the net links. In 2004, Bartoli et al. [BZ04] proposed a direct method, using radial basis functions instead of a discrete net for the deformation. The article also

provides a discussion on direct vs. feature based methods. In addition to this generic deformation functions, there has been a huge interest in specialized deformation functions for direct methods, such as for faces [MBB09, CGZZ10] or the human body [SFC<sup>+</sup>11].

In contrast to the methods defining a deformation function, parametric models such as used by level-set methods [OS88], active contours [CC91] or superquadrics [JLS00] try to provide a generic shape function which can be fit to 3D point cloud data. Superquadrics provide an implicit surface function, i. e., a “distance to surface” measure, which allows to efficiently minimize the overall distance of a superquadric to a point cloud. In 1998, Bardinet et al. [BCA98] proposed an extension to the superquadrics, by applying a deformation function to the parametric surface to increase the variety of shapes. Though still limited to star-convex shapes, this deformation introduced a non-trivial relationship between points in space and their closest point on the surface, inducing the need to iteratively approximate these. [BCA98] serves as a good example for the inevitable trade-off between shape complexity and ease of finding closest surface points in 3D. Direct methods (cf. [HH91]) based on image data circumvent this problem by reducing the “comparing domain” to a discrete set of projected pixels. In [JK11] a direct algorithm based on image data is introduced, performing fast deformation reconstruction of deforming objects based on depth and color video. It uses the complete image information provided, not only the data of selected feature points to estimate pose and deformation changes.

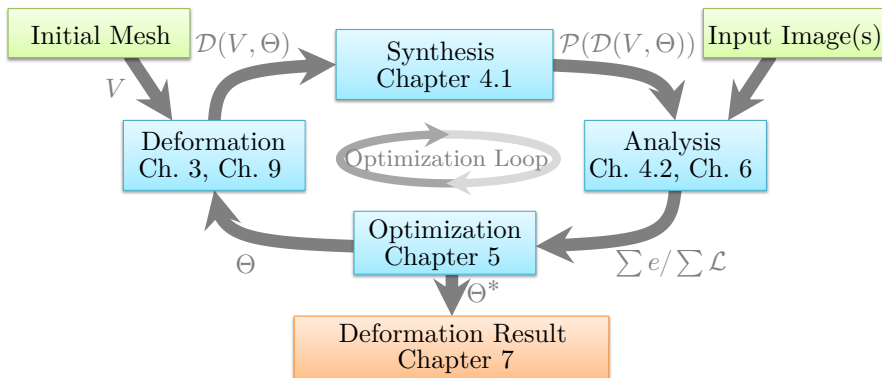
Direct methods, however, suffer from an inherent drawback, too: while correspondences provide a hint on the ongoing movement, direct methods have to estimate it by guessing solutions and improving them by analyzing the fit of these guesses. Hence, the direct approach for generic deformations can only produce global optima in combination with an optimization scheme able to efficiently handle high dimensional problems.

Although this is a difficult task, there are efficient optimization schemes like Covariance Matrix Adaptation - Evolution Strategy (CMA-ES) [Han06], able to handle high dimensional fitness functions that are neither smooth nor convex (see [ABH10] for benchmark results of CMA-ES in such environments).

## 1.2 Overview

This thesis will introduce an AbS-based approach to deformation tracking that is able to reconstruct complex deformations from depth and color video while also being applicable to real-time scenarios. The remainder of this thesis will be

## 1. Introduction



**Figure 1.1.** An overview showing the components and the data processing of the basic AbS systems discussed in this thesis. The functions and variables are introduced in the chapters in which the components are explained in detail.

structured as follows (see Fig. 1.1): Chapter 2 provides an introduction to depth sensing, with a focus on depth cameras, which play an important role in the AbS systems discussed here. Components of an AbS system are introduced by first discussing a Non-Uniform Rational B-Spline (NURBS)-based deformation model in Chapter 3, followed by Chapter 4 describing analysis and synthesis in detail. Chapter 5 will discuss the optimization scheme and Chapter 6 will complete the introduction of the base AbS method by providing approaches to handle the weighting of different entities. Chapter 7 contains a thorough evaluation of the AbS method introduced so far. The remaining chapters then extend on the base system by introducing occlusion handling (Chapter 8) and specializations for different tasks (Chapter 9). Chapter 10 will summarize this thesis and provide concluding statements.

# Sensing Depth

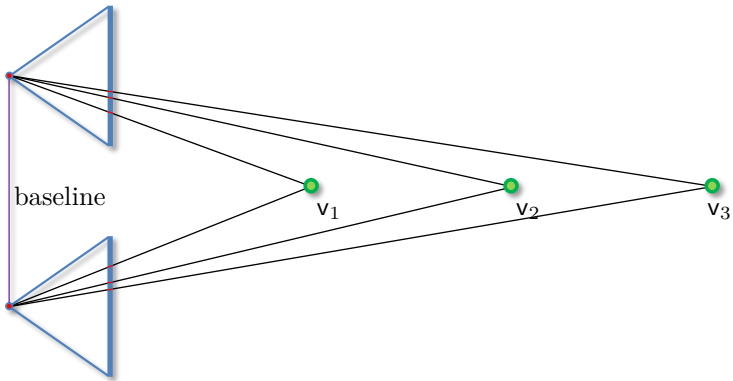
The sensing of depth information, especially the acquisition of depth images at high frame rates ( $\geq 25$  Hz) has gained an enormous amount of popularity in recent years due to the release of the first *Microsoft Kinect* camera in 2011. Although the research on depth-image based methods has increased since then, depth cameras have been around since the 1990s in an experimental stage, using the Time-of-Flight (ToF) technique, and as consumer products since 2005. Up until then, high frame rate depth image sequences were acquired only based on stereo- or multi-color camera setups as well as projector- camera setups. This Chapter will provide an overview of common depth sensing techniques, especially of active 3D acquisition methods.

## 2.1 Triangulation

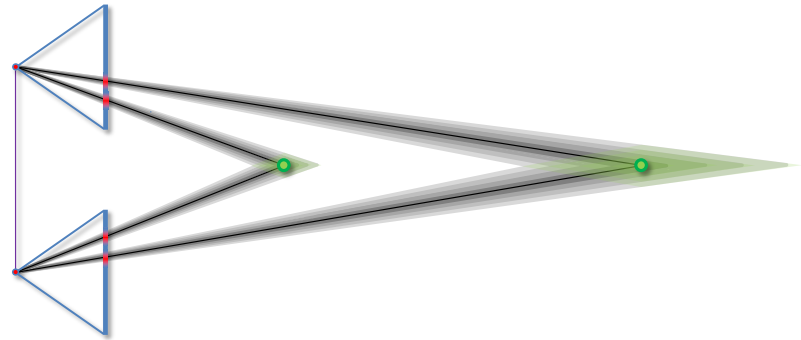
The basic principle of distance measurement with two or multiple color cameras is triangulation. The first *Microsoft Kinect* camera is a triangulation-based system, as well as the traditional stereo camera setup for depth estimation. Stereo camera systems work by finding a point in the image of each camera that corresponds to the same 3D point. If the system is calibrated (see Section 2.2.3), the angular information of the viewing rays along with the size of the baseline (Fig. 2.1) allows to calculate the positions of the 3D points ( $v_1 - v_3$  in Fig. 2.1).

One property of the triangulation that can be observed in Figure 2.1 is that the farther away a point is located from the camera centers, the less significant the position change becomes in the projections. Conversely, an observation in the projection domain with a given uncertainty can either lead to a small uncertainty in the reconstruction if the reconstructed point is

## 2. Sensing Depth



**Figure 2.1.** Triangulation of three equally distributed points (green dots) does not result in equally distributed projections (red marks).

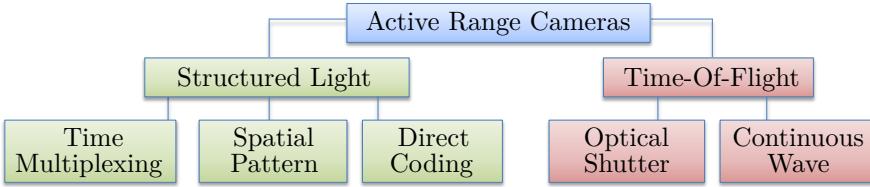


**Figure 2.2.** The same amount of uncertainty in the image plane (red area) can lead to large differences in the uncertainty when propagated (gray area around the black rays) to the reconstructed 3D point (green area).

close, or to a large uncertainty if the reconstructed point is far away from the camera, as depicted in Fig. 2.2. [HS95].

This is an important effect that will be addressed later in this thesis as the effect of noise in triangulation based reconstructions is discussed. A detailed introduction to triangulation in 3D reconstruction can be found in [BZ04].





**Figure 2.3.** A hierarchical classification of active range cameras.

## 2.2 Active Range Cameras

Capturing depth images at high frame rates has been the domain of stereo reconstruction for a long time. An alternative approach to stereo imaging provide camera setups that use active lighting to illuminate the scene with spatial and/or temporal patterns. These setups can be separated into two categories (see Fig. 2.3):

**Structured light** systems (Section 2.2.1) are based on triangulation and require a baseline (see Fig. 2.1) in order to calculate the distance information for each pixel, similar to stereo camera setups.

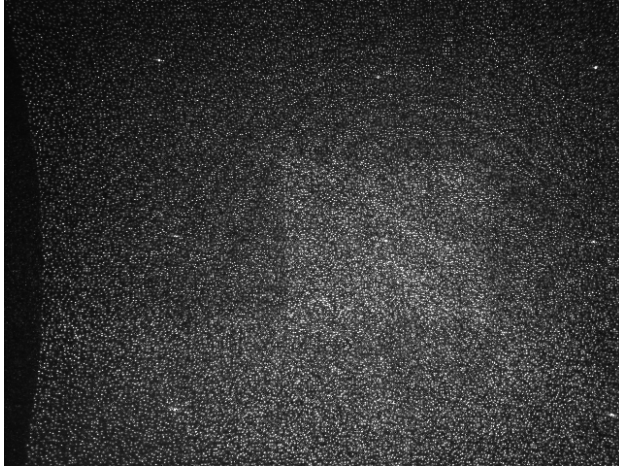
**Time-of-Flight** cameras (Section 2.2.2) utilize the time the light takes from its emitter to bounce of a surface and travel back to the camera to generate depth information for each camera pixel.

### 2.2.1 Structured Light

Structured light systems are similar to stereo camera setups. The difference is, that instead of triangulating between two cameras, one camera is replaced by a projector, emitting a pattern that can be located in the camera image. Many different techniques and patterns have been proposed over the years which can be roughly sorted into three categories:

**Direct-coding** methods use, e. g., sinusoidal patterns [GHL03] and allow a fast, direct estimation of depth for each pixel. Hence, these methods have a high effective resolution but are easily misled by vivid textures. In addition, a continuous coding can introduce ambiguities during the reconstruction process.

## 2. Sensing Depth

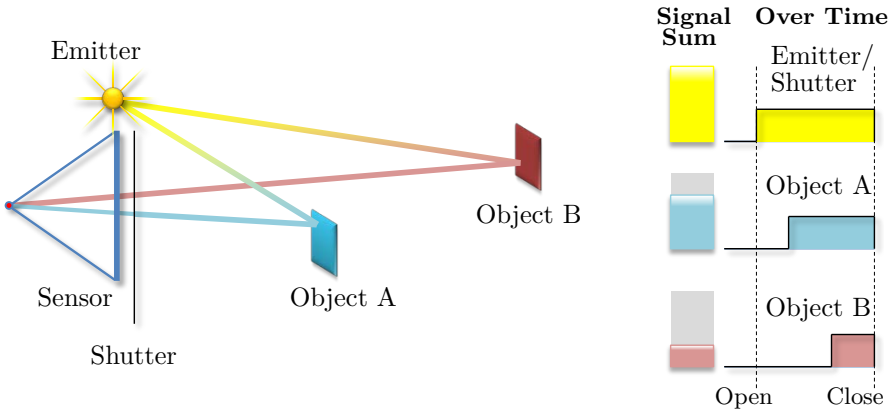


**Figure 2.4.** The pattern of the *Kinect* in the infrared domain is made from randomly distributed dots.

**Spatial-pattern based** methods use distinctive patterns that allow local retrievability. These methods do not necessarily suffer the ambiguity problems and texture sensitivity like direct coding methods. But since the depth of each pixel has to be calculated from the pattern in his vicinity, the effective lateral resolution is lower than the directly coded one.

**Time multiplexing** uses different patterns over time to reliably locate every pixel of the projector in the camera image. A very common way is the usage of gray codes [BER76] that are displayed by the projector over time. This approach only works for static scenes but allows an ambiguity-free and pixel-wise calculation of depth information.

For an in-depth survey on structured light systems refer to [SPB04]. The depth camera used for most of the experiments conducted in Chapter 7 is the *Microsoft Kinect* camera. It falls into the class of spatial-pattern based structured light systems and operates in the near-infra-red (IR) domain to provide depth images with a resolution of  $640 \times 480$  pixels at 30 frames per second (fps). In addition to the depth sensor, the *Kinect* also has a color camera built in. Because the pattern is projected in the IR domain, it does not interfere with the color image and allows to acquire depth and color video simultaneously.



**Figure 2.5.** Optical-shutter based ToF works by synchronously emitting light and opening a shutter on the sensor. Right: accumulated signal and incoming signal strength over time.

### 2.2.2 Time-of-Flight Cameras

Clocking light as it travels distances of few centimeters up to several meters is a challenging task and, although physically possible for small experimental setups, not directly feasible at a camera scale with reasonable resolution. Hence, ToF cameras perform indirect measurements of the effect caused by the temporal delay of the light as it travels from the emitter to the surface and into the camera. Like the *Kinect* camera, nearly all ToF cameras operate in the near IR domain, allowing to simultaneously capture depth and color images. There are two different approaches to measure this delay:

**Optical-shutter based ToF** cameras [IY01] illuminate the scene at a given moment and acquire an image at the same time (see Fig. 2.5). A synchronized shutter allows the sensor to acquire photons only for a short period of time. The distance of each pixel is then rendered from the amount of photons that entered each pixel on the sensor. A surface that is close to the camera will cause a high photon count, since the reflected light will arrive at the pixel shortly after the illumination has started. A surface farther away will result in a lower photon count because the pixel will receive no reflected light for a longer time period before the shutter ends the photon acquisition.

To compensate for differences in the surface reflectivity, usually a separate image is taken of the un gated, full illumination [YIM07]. To be less sensitive

## 2. Sensing Depth

to sensor noise, the acquisition is repeated several times, allowing the photon induced charges on the sensor to accumulate in the sensor. The *ZCam* by *3DV Systems* is using this technology.

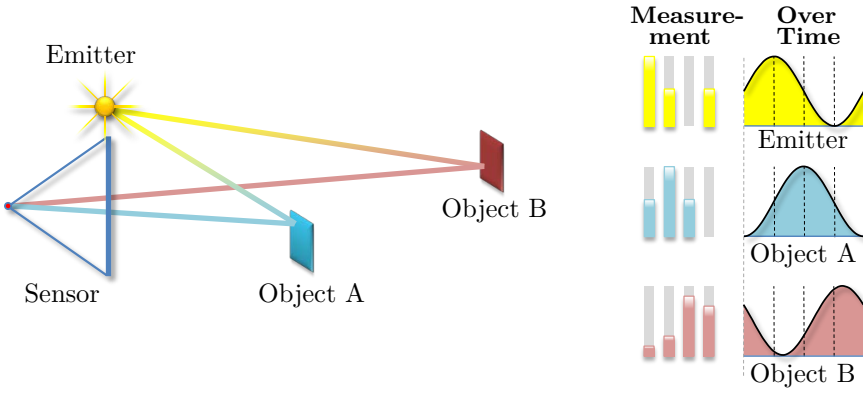
**Continuous-wave ToF** cameras work by modulating the amplitude of the emitted light in a sinusoidal way (see Fig. 2.6), usually at frequencies between 5 MHz and 80 MHz. Synchronized to the emitted light, the sensor of the ToF camera is capturing the incoming electrical charge generated by the photons in two separated bins, each accumulating the signal strength over  $180^\circ$  of the sinusoidal input. This is done for the phases  $0^\circ - 180^\circ$  and  $180^\circ - 360^\circ$  as well as  $90^\circ - 270^\circ$  and  $270^\circ - 450^\circ$ . Simplified, these measurements can be seen as approximations of the signal at  $45^\circ$ ,  $225^\circ$ ,  $135^\circ$ , and  $315^\circ$ . From these measurements, the underlying phase shift of the incoming signal can be retrieved (see Fig. 2.6 right), which has a linear relationship to the distance. The *CamCube* sensors by *PMD-Tech* and the *Swissranger* sensors by *Mesa Imaging* are using this technology. The *Kinect One*<sup>1</sup> uses a similar approach but with only three phase images. In addition, 3 sets of phase images are taken at different frequencies to detect and compensate for multi-path [JPMP14] effects. See [XPH13] for details on this method.

An important aspect of measuring distance with the ToF technique is that the actual measurement has a linear relationship to the result, i. e., in theory, a constant uncertainty in the measurement results lead to a constant uncertainty in the calculated distance. This is a fundamental difference to triangulation-based methods, which have an increasing uncertainty for increasing distances (see Fig. 2.2). In practice this advantage is lessened, since the reduced signal strength at longer distances causes the noise to increase accordingly. ToF cameras usually allow to retrieve the signal strength of each pixel along with the calculated depth value from the measurement images (see Fig. 2.7) and it can be used to provide an estimate about the accuracy of the distance measurement of each pixel. This property will be exploited in Chapter 6.2.

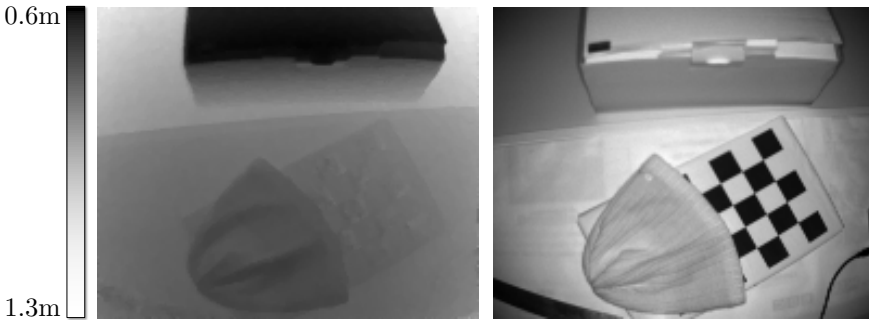
The acquisition of depth information using the ToF also entails a set of unique artifacts. The most obvious are caused by mixed depth information on edges and from multiple reflections in corners (the multi-path problem). A good review on these artifacts along with methods for removal can be found in [GCD11].

---

<sup>1</sup>The *Kinect One* is also referred to as *Kinect 2* or *Kinect 2.0*, not to be confused with the structured light sensor *Kinect*, also referred to as *Kinect 1*, *Kinect 1.0*, or *Kinect 360*.



**Figure 2.6.** Continuous-wave ToF cameras work by emitting sinusoidally amplitude-modulated light and synchronized sensing. Right: synchronized measurement of the four phases and the incoming signal strength over time.



**Figure 2.7.** Images acquired from a ToF camera (Mesa Imaging SR 4000). Left: the depth image contains a distance value in each pixel. Right: the intensity image contains for each pixel the amplitude of the signal that was used to render the distance value provided in the depth image.

### 2.2.3 Calibration

One of the advantages of the direct AbS approach introduced in this thesis is the ability to incorporate input signals from many different domains. AbS is capable of fusing any sort of measurement into the reconstruction pro-

## 2. Sensing Depth

cess such as depth sensors, color cameras, laser distance measurements, line cameras, light barriers; basically any input that can be synthesized from the parametrized model. While there are many sensor setups possible, this thesis will focus on arbitrary combinations of 2D projective sensors, i. e., cameras. In the following, whenever a set of cameras is expected to be fully calibrated, it is assumed that it's relative position and orientation is known (extrinsic parameters), as well the focal length, aspect ratio, principal point, and lens distortion of each camera (intrinsic parameters).

The tests and physical experiments conducted in this thesis were done using calibration information from the algorithm introduced by Schiller et al. [SBK08], a joint calibration method for depth and color cameras delivering extrinsic and intrinsic camera parameters, as well a lens distortion model (Bouget [Bou99]). In the following chapters, all relative camera poses and intrinsic parameters are expected to be known if not stated otherwise.

# NURBS Deformation Models

In the history of computer vision and computer graphics, various approaches to deformation modeling have been introduced. The most common model used in NRSfM decomposes the movement of features into the global pose change and a linear combination of deformation vectors. Here, the global 6 DoF pose is augmented by additional dimensions associated with deviations from the global pose. Hence, every point has not only its position in 3D, but also a scalar for each deformation dimension, describing how much it is (linearly) affected by the deformation at hand. Finding the position of each point in this high dimensional space is accomplished by a factorization method first introduced by Bregler [BHB00] for NRSfM, based on Tomasi and Kanades' Method [TK92] for SfM. The mathematically sound formulation requires the knowledge about the 2D movement of each feature point in the input sequence, which entails the problems discussed already in Section 1.1.1. Without features, some global deformation model function has to describe the movement of each point. For a triangle mesh, the most general deformation function imaginable is the unconstrained movement of each mesh vertex in all 3 dimensions, resulting in a parameter space with a dimensionality of 3 times the number of vertices. This is only feasible in special cases, e. g., meshes with a low vertex count and known optical 2D / 3D flow [SHF07], imposing, once again, the drawbacks discussed in 1.1.1.

Hence, to approach deformation tracking in a global way without any local information available, the deformation space has to be reduced to a tractable number of dimensions. This is an essential part of the deformation reconstruction process since it defines the trade-off between versatility and stability. This chapter will introduce NURBS surfaces [PT97a] and a NURBS-based deformation model as an example for a versatile deformation model for surfaces. Although the AbS approach introduced in this thesis can be

### 3. NURBS Deformation Models

used with many different deformation functions, the NURBS deformation serves as a good introduction example for the general concept as it is intuitive and applicable to a wide range of deformation tracking problems. Other deformation functions and more general approaches will be discussed in Chapter 9.

## 3.1 Non-Uniform Rational B-Splines

A good motivation for the usage of NURBS surfaces as a dimensional reduction is the empirical look at the domain of common object surfaces in general. In computer graphics and computer aided design (CAD) applications, NURBS have become a de-facto standard for modeling object surfaces [Bal08], combining versatility and ease of use. Besides an intuitive parametrization via 3D control points, NURBS are able to approximate any given 3D surface with arbitrary accuracy.

3D NURBS surfaces fall into the class of parameter surfaces, i. e., surface definitions given by functions from a 2D parameter space into the 3D coordinate system while the co-domain of each function resembles the set of surface points. Any surface point is a linear combination of control points. Control points can have any 3D position while the amount of influence on a parameter is given by its weighting function. These weighting functions can be understood as arranged in a non-uniform grid-like fashion in the parameter space. The distribution of the grid lines is determined by a set of “knots” for the two parameter space dimensions separately (see Fig. 3.1 (left) for a 1D visualization of knots and weighting functions).

The surface function for a set  $P$  of control points  $\mathbf{p} \in \mathbb{R}^3$  with degree  $d$  and  $m + 1$  knots per dimension is commonly defined [JK12, PT97a] for parameters  $(u, v) \in [0, 1]^2$  as:

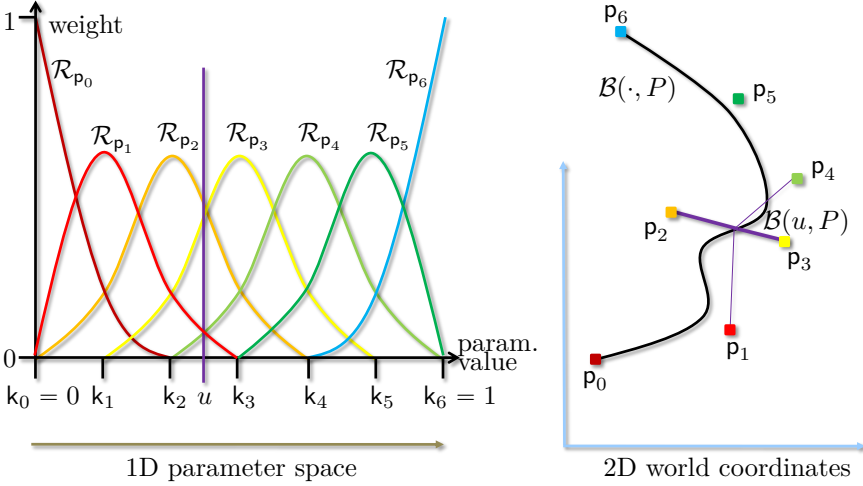
$$\mathcal{B}((u, v), P) := \sum_{\mathbf{p} \in P} \mathcal{R}_{\mathbf{p}}(u, v) \mathbf{p} \quad (3.1.1)$$

where  $\mathcal{R}_{\mathbf{p}}(u, v)$  is the weight function for each control point  $\mathbf{p}$ . Evaluated at a given parameter  $(u, v)$ ,  $\mathcal{R}_{\mathbf{p}}(u, v)$  provides the influence of  $\mathbf{p}$  to the corresponding surface point. Fig. 3.1 depicts an example of a NURBS function with a 1D parameter space and a 2D world coordinate system.

In the 2D/3D case that is addressed here, the control point grid in the parameter domain  $[0, 1]^2$  can be addressed by its row and column number. Given  $i \in \mathbb{N}$  as the row and  $j \in \mathbb{N}$  as the column number, the weighting



### 3.1. Non-Uniform Rational B-Splines



**Figure 3.1.** A 2D example of a NURBS function with a 1D parameter space (NURBS line). Left: weighting functions  $\mathcal{R}$  in the parameter space for every control point  $\mathbf{p} \in P$  with example evaluation  $u \in [0, 1]$  and entries of knot vector  $\mathbf{k}$ . Right: the resulting 2D line in world coordinates  $\mathcal{B}(\cdot, P)$ , shaped by the positions of the control points  $\mathbf{p}$ . The example evaluation  $\mathcal{B}(u, P)$  is depicted as the corresponding weighted linear combination of control point positions (purple lines). The thickness of each purple lines visualizes the influence of the control point given by the weighting for parameter  $u$ .

function  $\mathcal{R}_{\mathbf{p}}$  for each control point  $\mathbf{p} \in P$  is defined as:

$$\mathcal{R}_{\mathbf{p}}(u, v) = \frac{n_{i,d}(u)n_{j,d}(v)w_{i,j}}{\sum_{k=0}^m \sum_{l=0}^m n_{k,d}(u)n_{l,d}(v)w_{k,l}}. \quad (3.1.2)$$

$w$  describes the weight of each control point  $\mathbf{p}$  and for the remainder of this thesis, the NURBS weights will be set  $w_{i,j} = 1$  for all  $i, j \in \mathbb{N}_{\leq m}$  if not stated otherwise.  $n(\cdot)$  describes the weighting function, which, for each dimension of the parameter space, is recursively defined with respect to the knot vector  $\mathbf{k} = \{k_0, \dots, k_m\} \in [0, 1]^{m+1}$  of that dimension by:

$$n_{i,d}(u) = \frac{u - k_i}{k_{i+d} - k_i} n_{i,d-1}(u) + \frac{k_{i+d+1} - u}{k_{i+d+1} - k_{i+1}} n_{i+1,d-1}(u), \quad (3.1.3)$$

### 3. NURBS Deformation Models

for  $d > 0$  and by

$$n_{i,0}(u) = \begin{cases} 1 & \text{for } k_i \leq u < k_{i+1} \\ 0 & \text{otherwise} \end{cases} \quad (3.1.4)$$

as base clause. The knots  $\{k_0, \dots, k_m\}$  define the spatial influence of the control points on the surface in the parameter domain, hence they are used to, e. g., increase the density of control points in regions of higher surface information. But since the knot vectors are defined for each of the two surface dimensions separately, the usage of nonuniform knot vectors implies the separability of the surface information, which in our case cannot be assumed. Hence, the elements of the knot vector  $\{k_0, \dots, k_m\}$  will be equally distributed over  $[0, 1]$  in most situations.

## 3.2 Object Registration

For the usage on deforming objects, we are not directly interested in the modeling capabilities of NURBS surfaces, but in the surface space, i. e., the domain of possible surface shapes, as it obviously resembles a way to effectively parametrize the space of common surfaces. To reduce the number of dimensions of the deformation space, we exploit the huge variety of shapes that can be expressed by a rather small number of control points.

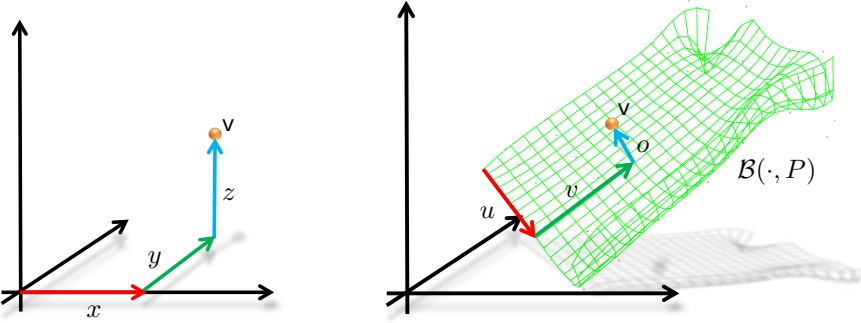
To avoid the loss of high frequency information by approximating the object, the NURBS surface function will not replace the object surface but only model its deformation. To do so, a NURBS surface approximation of the object mesh is generated (Section 3.2.2) and the triangle mesh vertices of the object are registered to the NURBS surface (Section 3.2.1).

### 3.2.1 The $uvo$ -Space

For an object shape given by a triangle mesh, the goal of the object registration is to find a representation of the vertices that is not rigidly connected to the euclidean 3D space, i. e., by  $\mathbf{v} = (x, y, z)$  coordinates, but connected to the NURBS surface. The  $uvo$ -Space is an example of how this can be achieved.

Let  $\mathbf{v}$  be a vertex of the given object, and let  $P_0$  be the initial set of control points. Let now  $u_{\min}$  and  $v_{\min}$  be found such that the distance between the NURBS surface and the vertex position becomes minimal:

$$(u_{\min}, v_{\min}) = \arg \min_{(u,v) \in [0,1]^2} \|\mathcal{B}((u, v), P_0) - \mathbf{v}\|_2. \quad (3.2.1)$$



**Figure 3.2.** A vertex  $\mathbf{v}$  described by its position in the Euclidean space  $(x, y, z)$  (left) and by its position  $(u, v, o)$  relative to the NURBS  $\mathcal{B}(\cdot, P)$  (right).

At this point, let the assumption be made that  $P_0$  is chosen reasonable, i. e., that the resulting minimization problem is convex, the NURBS surface is large enough such that the minimum lies within  $(0, 1)^2$  and not on the border, and the minimum is unique (see Section 3.2.2 for the initialization of  $P_0$ ). If  $\mathbf{v}$  is located on the NURBS surface, then  $\|\mathcal{B}((u_{\min}, v_{\min}), P_0) - \mathbf{v}\|_2$  would equal zero, however, in the general case  $\mathbf{v}$  will be located beneath or above the NURBS. Let  $o$  be the signed distance between the surface and  $\mathbf{v}$ , then

$$o = \pm \|\mathbf{v} - \mathcal{B}((u_{\min}, v_{\min}), P_0)\|_2 \quad (3.2.2)$$

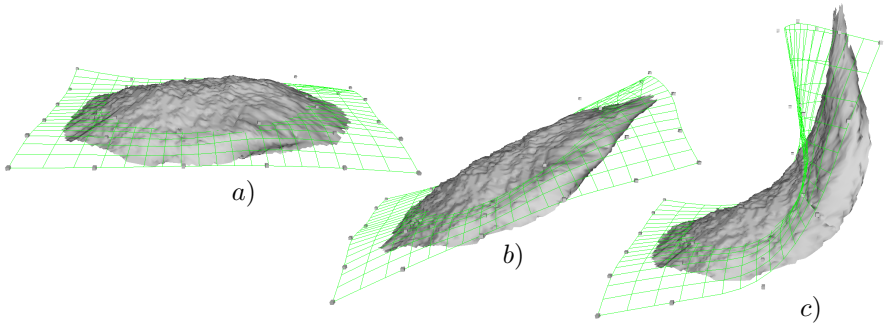
with the sign indicating if  $\mathbf{v}$  is above or below the surface. Because  $\mathcal{B}(\cdot, P_0)$  was chosen large enough, the vector pointing from the surface point  $\mathcal{B}((u_{\min}, v_{\min}), P_0)$  to  $\mathbf{v}$  is a scaled normal vector of  $\mathcal{B}(\cdot, P_0)$  at that point. This can easily be seen by looking at the surface point  $\mathcal{B}((u_{\min}, v_{\min}), P_0)$  as the orthographic projection to the tangential plane of  $\mathcal{B}(\cdot, P_0)$ . With  $\mathcal{B}^\perp : ([0, 1]^2, P) \rightarrow \mathbb{R}^3$  denoting the normal vector of the NURBS surface  $\mathcal{B}$ ,  $\mathbf{v}$  can be described using  $u$ ,  $v$ , and  $o$  only:

$$\mathbf{v} = \mathcal{B}((u_{\min}, v_{\min}), P_0) + o \cdot \mathcal{B}^\perp((u_{\min}, v_{\min}), P_0). \quad (3.2.3)$$

Figure 3.2 depicts the vertex  $\mathbf{v}$  described by  $(x, y, z)$  and via  $(u, v, o)$ .

For a given triangle mesh with its vertex set  $V$ , the corresponding  $(u, v, o)$  parameters can be calculated according to (3.2.1) and (3.2.2). For any set of control points  $P$  other than  $P_0$ , the deformation of the NURBS surface can

### 3. NURBS Deformation Models



**Figure 3.3.** An example deformation of a triangle mesh using a NURBS deformation function (green) defined by  $P$  (small gray dots): a) the mesh in its original state, b) the mesh subject to a mostly affine transformation, c) the mesh subject to a strong deformation.

be propagated to the vertex set using (3.2.3). Figure 3.3 depicts a triangle mesh example subject to a NURBS deformation function.

#### 3.2.2 NURBS Initialization

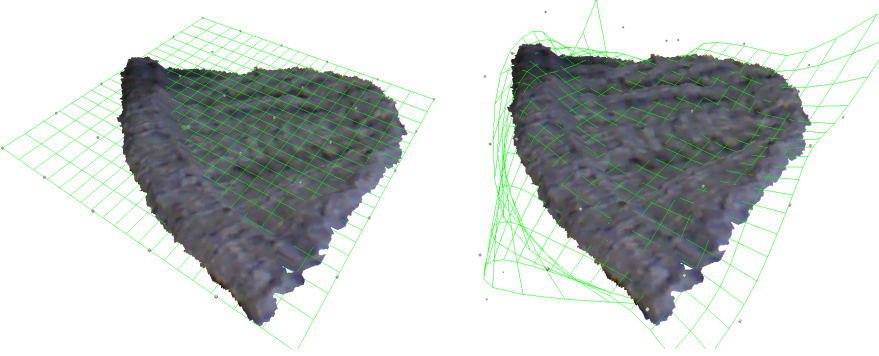
Prerequisite to a registration of the NURBS to a triangle mesh is the definition of the initial state of the NURBS described by the control point set  $P_0$ . There are multiple ways to do such an initialization, the most intuitive one being to minimize the RMS distance between  $V$  and  $\mathcal{B}(\cdot, P)$  by finding

$$P_0 = \arg \min_{P, \{(u_i, v_i): i=1, \dots, |V|\}} \sum_{v_i \in V} \|v_i - \mathcal{B}((u_i, v_i), P)\|_2, \quad (3.2.4)$$

which is equivalent to the more intuitive minimization problem

$$P_0 = \arg \min_P \sum_{v \in V} \min_{(u, v) \in (0, 1)^2} \|v - \mathcal{B}((u, v), P)\|_2. \quad (3.2.5)$$

This minimization yields a NURBS surface  $\mathcal{B}(\cdot, P)$  which interpolates the set of vertices  $V$  best. Although the optimization problem is not necessarily convex, its optimum usually is approximated by a local optimization of  $P_0$  (see [PT97b]) if a proper initial guess for  $P$  is available. At its core, the problem of finding such an initial guess is the task of creating a two dimensional manifold



**Figure 3.4.** A triangle mesh and a NURBS deformation function. Left: a purely affine initialization. Right: an affine initialization followed by an optimization of  $P$  in respect to (3.2.5).

(or mapping) in  $\mathbb{R}^3$  resembling the two dimensional surface sampled by the vertex set.

A common way to create this starting guess is to calculate the mean  $\bar{\mathbf{v}} \in \mathbb{R}^3$  and the covariance  $C \in \mathbb{R}^{3 \times 3}$  of the vertex set  $V$ :

$$\bar{\mathbf{v}} = \sum_{\mathbf{v} \in V} \frac{\mathbf{v}}{|V|}, \quad (3.2.6)$$

$$C = \begin{pmatrix} c_{x,x} & c_{x,y} & c_{x,z} \\ c_{y,x} & c_{y,y} & c_{y,z} \\ c_{z,x} & c_{z,y} & c_{z,z} \end{pmatrix} \quad (3.2.7)$$

with

$$c_{j,k} = \sum_{\mathbf{v} \in V} \frac{(\mathbf{v}_j - \bar{\mathbf{v}}_j)(\mathbf{v}_k - \bar{\mathbf{v}}_k)}{|V|}, \quad j, k \in \{x, y, z\}. \quad (3.2.8)$$

The  $x$ ,  $y$ ,  $z$ -indices in Equation (3.2.8) denote the corresponding  $x$ -,  $y$ -, or  $z$ -component of the vectors  $\bar{\mathbf{v}}$  and  $\mathbf{v}$ . Extracting the eigenvector  $\mathbf{e}$  with the smallest eigenvalue yields the direction in which  $V$  is distributed (spread) least. In other words, the plane defined by  $\bar{\mathbf{v}}$  and  $\mathbf{e}$ ,

$$\{\mathbf{a} \in \mathbb{R}^3 : (\mathbf{a} - \bar{\mathbf{v}}) \times \mathbf{e} = 0\} \quad (3.2.9)$$

### 3. NURBS Deformation Models

is the best affine approximation of  $V$  regarding the RMS distance between vertices and surface and a good initial guess for  $\mathcal{B}(\cdot, P)$ . If an affine approximation of the surface is not sufficient, this affine approximation can be refined by local minimization of (3.2.5). Figure 3.4 shows the results of the affine approximation along with an approximation using local optimization to determine  $P$ . Figure 3.4 also visualizes a problem that can occur when approximating triangle meshes with a NURBS surface: since the edges of the object in general do not directly fit the edge of the NURBS function, the outer control points are subject to overcompensation<sup>1</sup>, generating control points that lie far away from the object. This can slow down the optimization during the object tracking (see Chapter 5) as outer control points may have to cover long distances to describe simple rotations in the affine space.

This initialization is most suitable for situations in which the tracked object is provided as a triangle mesh beforehand. However, in most situations the tracked object is unknown and its geometry is extracted from the first frame of the tracking sequence. In this situation, a distribution does not need to be calculated, as the 2D subspace to spread the NURBS parameters in is already given by the depth camera, i. e., by the equivalence relation of its projective space, which can optionally be followed up by the local minimization of Equation (3.2.5).

In the following, let  $V_P \subset (0, 1)^2$  denote the indexed set of parameters containing the initialized and registered parameters for the indexed vertices in  $V$ , such that

$$V_P = \bigcup_{v_i \in V} \arg \min_{(u, v, o) \in (0, 1)^2 \times \mathbb{R}} |o = \pm \|v_i - \mathcal{B}((u, v), P)\|_2|. \quad (3.2.10)$$

Using a properly initialized set of control points  $P_0$ , the indexed parameter set is denoted by  $V_{P_0}$ . Changing the set of initial control points  $P_0$  to a differently placed but equally sized set of control points  $P$  will yield the deformed set of vertices  $\mathcal{B}(V_{P_0}, P)$ .

Since the NURBS deformation is just one possible implementation of a deformation function (other deformation functions are introduced in Chapter 9), let  $\mathcal{D} : \mathbb{R}^3 \times \mathbb{R}^n \rightarrow \mathbb{R}^3$  denote an abstract deformation function parametrized

---

<sup>1</sup>The optimization of control point locations  $\mathbf{p}$  in respect to a given set of vertex parameters and surface distances can cause outer control points to move far away from the object. This is caused by the potentially small amount of influence (weight)  $\mathcal{R}$  on the position of outer vertices, requiring large pose changes in  $\mathbf{p}$  to compensate for that (See Eq. (3.1.1)).

by an  $n$ -dimensional parameter vector  $\Theta$ , such that

$$\mathcal{D}(\cdot, \Theta) : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \quad (3.2.11)$$

maps every 3D point to its deformed position. Implemented as a NURBS based deformation function, this means that  $\Theta = P$  and

$$\mathcal{D}(v, \Theta) = \mathcal{B}((u, v), \Theta) + o \cdot \mathcal{B}^1((u, v), \Theta), \quad (3.2.12)$$

with

$$(u, v, o) = \arg \min_{(u, v, o) \in (0, 1)^2 \times \mathbb{R}} |o = \pm \|v_i - \mathcal{B}((u, v), P_0)\|_2|$$

for any  $v \in \mathbb{R}^3$  covered by the initialization  $\Theta_0 = P_0$ .

### Evaluation Preview

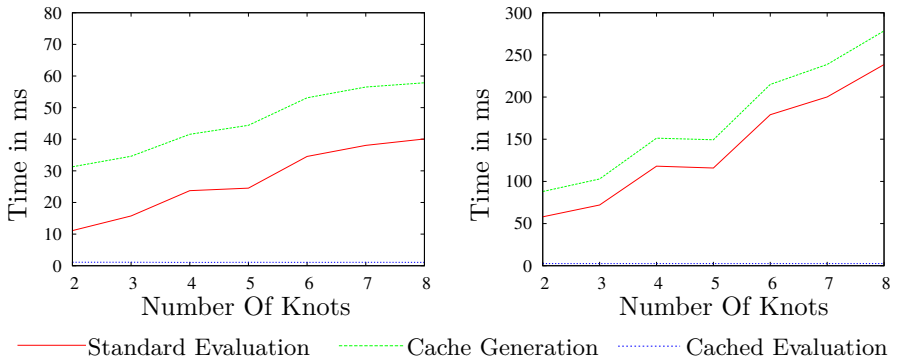
The NURBS deformation model introduced in this section has been well tested in many experiments. Its 2D topology makes it most suitable for surface reconstruction without consideration of the complete body of an object and it has been successfully tested using bags (Section 7.2), woolen hats [JK11], rubber objects 7.3 and paper 7.3.1. Nevertheless, it was also tested successfully on complex synthetic (Section 7.1.1) and complex real (Section 7.2.1) objects.

## 3.3 Fast Evaluation Cache

For a fast and efficient calculation of a deformed vertex set  $\{\mathcal{B}((u_i, v_i), \Theta) | i = 1, \dots, |V|\}$ , the evaluation of  $\mathcal{B}$  via equations (3.1.1) - (3.1.4) is not suitable because it involves recursive evaluation, introducing a large number of function calls for each deformed vertex, which should be avoided. One way of dealing with the problem is by flattening the evaluation to a function-call-free loop evaluation. However, a loop-based evaluation has a large number of operations as well, along with the loop-induced jump in the program code<sup>2</sup>. But a closer examination of the NURBS definition reveals a large optimization potential: the knot distribution and the weight of each control point is only changed

<sup>2</sup>Conditional jumps in a program can cause a temporary stall of operations in a modern Central Processing Unit (CPU) as their pipelined processing requires jump predictions. If a jump prediction fails, all pre-fetched and pre-calculated data is discarded and the processor restarts processing at the correct jump destination.

### 3. NURBS Deformation Models



**Figure 3.5.** Performance comparison between cache generation, uncached, and cached NURBS function evaluations. Left: 10,000 function evaluations of a second degree NURBS function for various knot counts. Cached evaluation only takes around 1 ms (1.05 ms - 1.11 ms) Right: 10,000 function evaluations of a fourth degree NURBS function for various knot counts. Cached evaluation only takes around 2.5 ms (2.52 ms - 2.58 ms).

during the initial object approximation, not during tracking because it would introduce a potential source of degeneration to the optimization problem and require a complete reassignment of the  $u, v, o$  parameters for each vertex which is computationally expensive. Having this fixed relationship allows for a drastic reduction of evaluation costs of  $\mathcal{B}$ , since equations (3.1.2) - (3.1.4) only depend on the choice of the knots  $k$  and the weights  $w$ .

Hence, after  $P_0$  is defined by the NURBS initialization and registration, the set of  $R_p(u, v)$  as defined in (3.1.2) can be calculated and stored for each combination of control point  $\mathbf{p} \in P$  and the tuple  $(u_{\min}, v_{\min})$  of each vertex  $\mathbf{v} \in V$ . Note that, given  $(u, v)$ ,  $\mathcal{R}$  is non-zero for only a few of the control points  $\mathbf{p}$ , allowing to efficiently store all relevant calculated values of  $\mathcal{R}$  in memory (NURBS cache). With the pre-calculated results available, the evaluation of  $\mathcal{B}$  is reduced to a small number of multiplications and additions to evaluate Equation (3.1.1) for each  $\mathbf{v} \in V$ .

Figure 3.5 depicts evaluation times of the cached and uncached evaluation for various knot counts of a second degree (left) and a fourth degree NURBS (right) function along with the cache generation time. Experiments were conducted using a single thread on a *Intel Xeon E3-1275* processor. It shows that the time it takes to generate the cache (green) is comparable with the



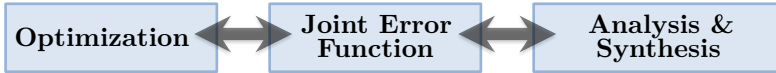
### 3.3. Fast Evaluation Cache

time an uncached evaluation takes (red). The cached evaluation (blue) then provides a speed up of up to factor 100. The visible indent in the curve at a knot count of five in both plots, which also appears on other processors, is most likely caused by compiler optimization, as it is not visible if compiler optimization is switched off.

In this chapter, a deformation function  $\mathcal{D}$  has been defined for vertex sets, parametrized by a deformation parameter vector  $\Theta$ . The following chapter will integrate this deformation function into the image synthesis process, which will then allow a direct comparison of input images with synthesized ones to evaluate  $\Theta$ .



# Analysis by Synthesis



In this chapter, the basic ideas of AbS will be discussed along with a generic application flow. The different parts of the algorithm are exchangeable to a large extent which allows AbS to handle very complex deformation tracking tasks with high accuracy on complicated data as well as real-time deformation tracking.

The general task solved by AbS is the estimation of parameters based on measurements that reflect a result of these parameters, e. g., to find the six DoF pose of a 3D object relative to a camera by analyzing the camera image. What sets apart AbS from other tracking and estimation methods is the direction in which data is processed: traditional tracking algorithms start by analyzing the image, deriving second level information such as features, contours, and movement. In a second step, the retrieved data is processed into a final result. AbS starts with a guess of the result (parameters) and synthesizes the input the camera would generate (e. g., by rendering an image) if these parameters would describe the actual object state. This synthesis is then compared to the real input data, which allows to evaluate the guessed parameters in the input domain. Hence, the analysis of many guesses allows to find the parameters fitting the input data best.

A drawback of the AbS approach is that it is in general not able to use local image information like movements or correspondences. At the same time AbS does not have to bother with problems like outliers or incorrect/inaccurate

## 4. Analysis by Synthesis

feature points. But the most appealing feature of AbS is the usage of the complete relevant image information, which allows AbS to provide results that are more accurate than feature based results. As an example, Schiller et al. [SBK08] used AbS in their checker-board based calibration method to refine the feature based result in a post processing step since it is independent of the inaccurate corner detection.

The three fundamental building blocks of the class of algorithms introduced in this thesis are as follows.

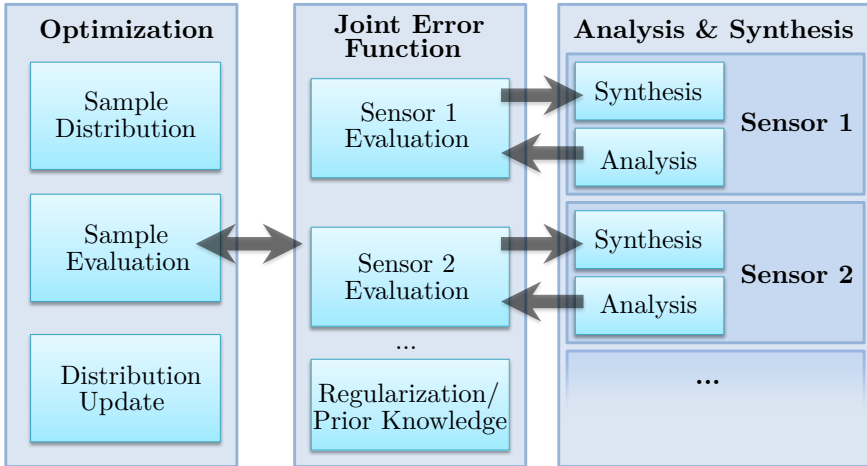
- ▷ The synthesis of the expected input data, providing an artificial input signal that can be compared to the actual sensor input (Chapter 4.1).
- ▷ The joint error function, fusing all these comparisons along with regularization and prior knowledge into one scalar error value (Chapter 4.2 + 4.3).
- ▷ The optimization method that is providing the global minimum of the error function and, hence, the most suitable parameter set with respect to the input data (Chapter 5).

A parameter guess of the optimizer is evaluated in the joint error function. The error function provides a weighted error sum over all input devices along with penalties derived from prior knowledge about the task at hand. Good examples for such priors are local or temporal smoothness as well as physical plausibility. After the parameter guess is evaluated, it is compared to other guesses to determine a set of new guesses in the parameter space that will minimize the joint error function. A most suitable optimization scheme for this optimization task is CMA-ES, which will be discussed in detail in Chapter 5. See Fig. 4.1 for a graphical overview.

In the following chapters, these components (Fig. 4.1) are explained from right to left, starting with the synthesis and its analysis.

### 4.1 Synthesis

In the generic case, an AbS algorithm can use any input data as long as its input can be generated artificially, i. e., synthesized. This thesis will focus on depth and color camera sensors, introducing first the well understood image synthesis as it is commonly used in the field of computer graphics [Gor12]. The 3D rendering capabilities of modern graphics cards allow to easily create hundreds of synthesized images per second. In this synthesis step, the geometry



**Figure 4.1.** An overview over the analysis by synthesis approach.

information passed by the optimizer to the error function will be converted into an image.

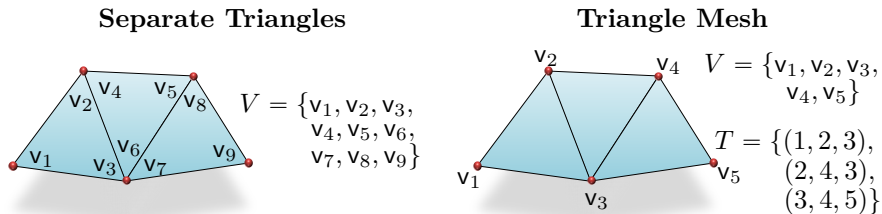
### 4.1.1 Geometric Projection

In computer graphics, the most common way of representing three dimensional objects are surface approximations by sets of triangles. A set of connected triangles (i. e., edge sharing) is also referred to as a triangle mesh. In most 3D applications, these triangle meshes are not described as separate triplets of 3D points (Fig. 4.2 left), but by an indexed set of 3D points, and a separate set of index triplets, containing the topology information of the 3D points (Fig. 4.2 right). This representation is not only memory efficient, it also allows fast processing, since each 3D point is only processed once when the object is rendered. According to the conventions of computer graphics, 3D points will be referred to as vertices in the remainder of this thesis.

Let  $V$  be a set of elements and  $T$  the corresponding set of indices. To generate a rendered image, i. e., an artificial camera image of a 3D object, a set of transformations from the 3D location to its 2D image coordinate<sup>1</sup> has

<sup>1</sup>This description of image rendering corresponds to the fixed function pipeline used for example in graphics cards for fast rendering. However, there are other rendering techniques, e. g., ray tracing [Suf07], following a different approach, which will not be covered here.

#### 4. Analysis by Synthesis



**Figure 4.2.** Different triangle representations. Left: set of separate triangles stores redundant information if the triangles form a closed surface. Right: an indexed set of vertices contains the same triangle mesh information in a more efficient way.

to be applied to each vertex  $v \in V$ , followed by filling the spaces between these 2D image coordinates according to the topology information in the index set  $T$ .

In the following, all three-dimensional points will be described as elements of the projective space  $\mathbb{P}^3$ . In a nutshell, the three-dimensional projective space  $\mathbb{P}^3$  is formed from the  $\mathbb{R}^4/0$  and allows to describe rotations, translation, scale changes and skews as linear operators, i. e., matrices in  $\mathbb{R}^{4 \times 4}$  (see [HZ00] for an introduction to the projective space<sup>2</sup>).

A point  $\mathbf{v} \in \mathbb{P}^3$  at  $(x, y, z) \in \mathbb{R}^3$  coordinates is given by the vector

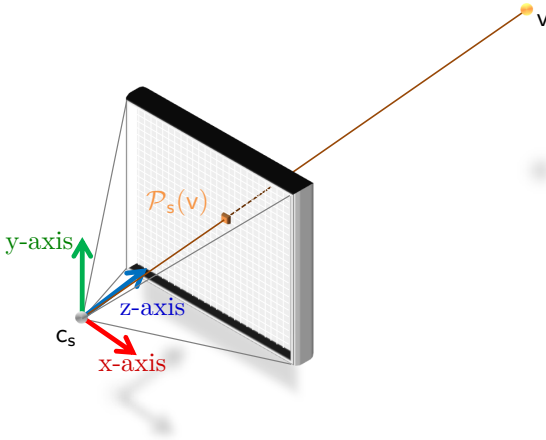
$$\mathbf{v} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}, \quad (4.1.1)$$

and an affine transformation is given by a matrix  $\mathbf{A} \in \mathbb{R}^{4 \times 4}$

$$\mathbf{A} = \begin{pmatrix} \mathbf{R}_{0,0} & \mathbf{R}_{0,1} & \mathbf{R}_{0,2} & t_0 \\ \mathbf{R}_{1,0} & \mathbf{R}_{1,1} & \mathbf{R}_{1,2} & t_1 \\ \mathbf{R}_{2,0} & \mathbf{R}_{2,1} & \mathbf{R}_{2,2} & t_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (4.1.2)$$

where the  $\mathbf{R} \in \mathbb{R}^{3 \times 3}$  components describe a rotation, scale change and skew transformation while the elements of  $\mathbf{t} \in \mathbb{R}^3$  describe a translation.

<sup>2</sup>The main contribution of the projective space, along with the linearity of the perspective projection, is the ability to describe points at infinity, a feature that will not be used in this context and, thus, will not be discussed in this thesis.



**Figure 4.3.** A visualization of the pinhole projection and the camera coordinate system. The camera coordinate system is originated at the camera center  $c_s$ , the image plane is parallel to the x-y-plane of the camera coordinate system.

To model how a 3D point  $\mathbf{v}$  is projected onto the image plane of a camera, the position and orientation (extrinsic parameters) have to be regarded as well as the intrinsic camera parameters. The extrinsic parameters are regarded by such an affine transformation  $\mathbf{A}_s$  from the world coordinate system to the camera coordinate system. The camera coordinate system is defined in the way that  $\mathbf{O}$ , i. e.,  $(0, 0, 0, 1)^T$  is equal to the optical center of the camera center, while the z-axis describes the optical axis. The x- and y-axes are parallel to the x- and y-axes of the image plane. This can be done by applying the matrix

$$\mathbf{A}_s = \begin{pmatrix} \mathbf{R}_s^T & -\mathbf{R}_s^T \mathbf{c}_s \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (4.1.3)$$

where  $\mathbf{R}_s$  describes the orientation of the camera in the world coordinate system and  $\mathbf{c}_s$  the optical center of the camera in the world coordinate system (see Fig. 4.3).

After the vertex  $\mathbf{v}$  is moved into the camera coordinate system, the intrinsic camera parameters determine the projection onto the image plane. The x- and y-axes are scaled by the focal lengths  $f_x$  and  $f_y$  of the camera. Image skew  $s \in \mathbb{R}$  and the image center (principal point)  $\rho \in \mathbb{R}^2$  are applied as well

## 4. Analysis by Synthesis

by a transformation described by the camera matrix  $\mathbf{K}$ :

$$\mathbf{K}_s = \begin{pmatrix} f_x & s & \rho_x & 0 \\ 0 & f_y & \rho_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad (4.1.4)$$

The resulting vector

$$\begin{pmatrix} u_d \\ v_d \\ d \end{pmatrix} = \mathbf{K}_s \mathbf{A}_s \mathbf{v} \quad (4.1.5)$$

now contains the image coordinates in a non-normalized form, i. e., the image coordinates  $u$  and  $v$  can be retrieved by normalizing the homogeneous component:

$$u = \frac{u_d}{d}, v = \frac{v_d}{d}. \quad (4.1.6)$$

### 4.1.2 Lens Distortion

While the camera matrix models the aspects of a pinhole camera, it does not regard the effects caused by the camera lenses. The most significant influence of the lenses with impact on 3D reconstruction accuracy is lens distortion. There are also other lens effects, e. g., chromatic aberration and blur (defocus), which are geometrically rather insignificant for the application of 3D reconstruction and will be neglected.

Lens distortion, as it is defined in e. g., [ASB00], is the deviation of the ray directions from the ideal pinhole model and it is described and compensated for by a distortion function  $\delta : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ . The distortion function is usually added to the undistorted image coordinates [FLP04]

$$\begin{aligned} u' &= u + \delta_u(u, v) \\ v' &= v + \delta_v(u, v) \end{aligned} \quad (4.1.7)$$

to receive the distorted image coordinates  $(u', v')$ .

The distortion effects of a lens can be separated into different classes by their physical source and, hence, their mathematical sub domain of modeling functions. The less significant class is the tangential distortion, a displacement of image elements with a function that is bilinear (and / or quadratic) in relation to  $u$  and  $v$ . Due to its usually small amount on most cameras it is often neglected, as it is in the remainder of this thesis. The more significant class of geometric image distortions is the set of radial distortions [HZ00], i. e.,



the displacement of image coordinates as a function of their distance to the principal point, as this distortion is directly caused by the shape of the lenses:

$$\begin{aligned} u'_r &= u + u \cdot \delta_r(\sqrt{u^2 + v^2}) \\ v'_r &= v + v \cdot \delta_r(\sqrt{u^2 + v^2}). \end{aligned} \quad (4.1.8)$$

There are multiple works on how to approximate  $\delta_r$  best, e. g., Tsai [Tsa87] and Brown [Bro71]. The results of this thesis were generated using a 6<sup>th</sup> order polynomial as the radial distortion approximation:

$$\delta_r(d) = k_1 \cdot d^2 + k_2 \cdot d^4 + k_3 \cdot d^6 \quad (4.1.9)$$

as described in [SBK08].

Many tests have shown that regarding only the first order term of the radial distortion already reduces average reprojection errors to 0.1 to 0.5 pixels on many cameras [FLP04], hence, using a 6<sup>th</sup> order polynomial is sufficient for the reconstruction application at hand.

The extrinsic and the intrinsic parameters of each camera are calibrated beforehand, using the calibration method described in 2.2.3. In the remainder of this thesis, all camera parameters are expected to be known. A set of extrinsic and intrinsic camera parameters is denoted by  $\mathbf{s}$ .

### 4.1.3 Rendering

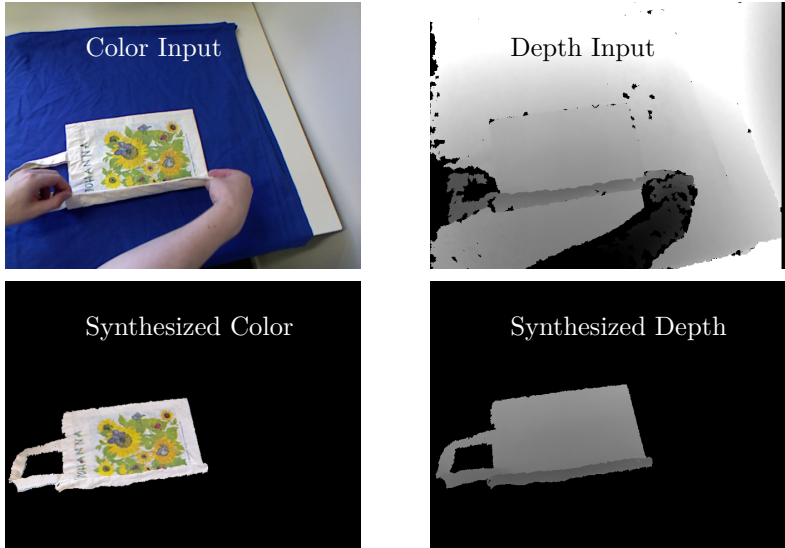
Using the process described in sections 4.1.1 and 4.1.2, a projection function  $\mathcal{P}$  can be defined, which maps a 3D vertex  $\mathbf{v}$  (see Fig. 4.3) to its projected image coordinates  $(u', v')$ :

$$\mathcal{P}_s(\mathbf{v}) = \delta_s(\mathbf{K}_s \mathbf{A}_s \mathbf{v}), \quad (4.1.10)$$

with  $\mathbf{v}$  being the homogeneous version of  $\mathbf{v}$ . To generate a synthesized image, the triangles defined in  $T$  can be drawn onto an image  $I$  according to the projected positions provided by  $\mathcal{P}$  (see [Gor12] for a detailed introduction to this rendering process). Images  $I$  will be denoted as functions from the image coordinate space  $\mathbb{R}^2$  to the content space, which can be  $\mathbb{N}_{<256}^n$ ,  $[0, 1]^n$ , or  $\mathbb{R}^n$ , depending on the content and the number of channels  $n$ .

For an undeformed object model given by a set of vertices  $V$  and a set of (textured) triangles  $T$ , a deformation result for a parameter set  $\Theta$  can be synthesized for a color camera  $\mathbf{s}$ , by rendering the triangle mesh with the

## 4. Analysis by Synthesis



**Figure 4.4.** An example input image and the corresponding synthesis of the deformed object.

image coordinates

$$\mathcal{P}_s(\mathbf{v}) = \delta_s(\mathbf{K}_s \mathbf{A}_s \mathcal{D}(\mathbf{v}, \Theta)) \quad (4.1.11)$$

for all  $\mathbf{v} \in V$  to an image  $I_{s,\Theta}$  (see (3.2.12) for a definition of  $\mathcal{D}$  using NURBS). In the following,  $I_{s,\Theta}^c$  will denote such a synthesized color image and  $I_{s,\Theta}^d$  the depth image, containing the camera-object distance value for every pixel instead of color information. Figure 4.4 (bottom) depicts a synthesized color image and a synthesized depth map. This direct connection between a parameter vector  $\Theta$  and the synthesized image  $I_{s,\Theta}^c$  will be used in the following Sections to analyze  $\Theta$  with respect to a given set of input images.

## 4.2 Analysis

Goal of the analysis step is to provide a fitness value for a deformation parameter vector  $\Theta$ , describing how plausible the parametrized deformation is with respect to a given set of information. This information can contain physical constraints, color images, depth images, as well as other sensor or

semantic information. Let  $S$  denote the set of these abstract constraints and sensors  $s$ . The overall fitness  $\mathcal{F}(\Theta)$  is calculated as the negative weighted sum over error functions  $e : \mathbb{R}^n \rightarrow \mathbb{R}_{\geq 0}$  derived from the input images, sensors, and constraints:

$$\mathcal{F}(\Theta) = - \sum_{s \in S} \varphi_s e_s(\Theta), \varphi_s \in \mathbb{R}_{\geq 0}. \quad (4.2.1)$$

As  $\mathcal{F}$  is only used for optimization purposes, the absolute values of the weights  $\varphi_s$  are irrelevant, only the ratio of these values affects the result. In the following Sections, various error functions for sensors and constraints are introduced. Choosing the weights  $\varphi$  will be discussed in detail in Chapter 6.

### 4.2.1 Color Error

For an input color image  $I_{s,\text{in}}^c$  (see Fig 4.4 top left), an intuitive error is calculated by the average pixel difference in the color space. Let  $\text{Dom}(I_{s,\Theta}^c)$  denote the set of discrete pixel positions  $(u, v)$  that are defined by the synthesis (object pixel), then:

$$e_c(\Theta) = \sum_{(u,v) \in \text{Dom}(I_{s,\Theta}^c)} \frac{\|I_{s,\Theta}^c(u, v) - I_{s,\text{in}}^c(u, v)\|_2}{|\text{Dom}(I_{s,\Theta}^c)|}. \quad (4.2.2)$$

Since only pixel positions are compared that have been actively synthesized in  $I_{s,\Theta}^c$ , no explicit segmentation of the input image is needed as it is implicitly performed by the estimated deformation  $\Theta$ .

One aspect of color images that is often forgotten in reconstruction algorithms is the fact that most color images already are interpretations of the actually sensed data. Most color cameras use only one sensor with a Bayer pattern [Bay76], so for each pixel only one color value is sensed. Demosaicing, i. e., reconstructing the full red-green-blue (RGB) information for each pixel has been subject to research for over 20 years. Simple approaches like Nearest Neighbor (NN) produce a large amount of artifacts (see Fig. 4.5 center), and while complex demosaicing algorithms like Directional Filtering (DF) [MAC07] and Adaptive Homogeneity-Directed Demosaicing (AHD) [HMP05] deliver visually plausible results (see Fig. 4.5 right), they are computationally expensive and still show signs of blurred and colored edges.

While the correct reconstruction of RGB values from Bayer pattern images is basically impossible, the artificial generation of Bayer pattern images from RGB data is trivial. Due to the AbS approach, no color image reconstruction

#### 4. Analysis by Synthesis



**Figure 4.5.** Left: an example Bayer pattern (GRBG) as it is used for CCD chips in cameras. Center: the result of the NN demosaicing algorithm shows block artifacts and colored edges. Right: the demosaicing algorithm AHD.

of the input data has to be acquired. Instead, the synthesized image can easily be converted into a Bayer pattern image, allowing a direct comparison to the input data. Even blurring and cross-talk effects of the Bayer pattern sensor can easily be modeled this way, while they are usually considered to be impossible to reconstruct in a demosaicing approach [MAC07].

#### 4.2.2 Depth Error

Similar to the analysis of color images, pixels of depth input images can be compared to an input image  $I_{s,\text{in}}^d$  from a depth sensor to evaluate a deformation. Defining  $D$  as the set of depth pixels valid in the synthesis as well as the input image

$$D = \text{Dom}(I_{s,\Theta}^d) \cap \text{Dom}(I_{s,\text{in}}^d), \quad (4.2.3)$$

the mean depth error is given by

$$e_d(\Theta) = \sum_{(u,v) \in D} \frac{\|I_{s,\Theta}^d(u,v) - I_{s,\text{in}}^d(u,v)\|_2}{|D|}. \quad (4.2.4)$$

A problem that occurs when formulating the depth error in this way, is the high errors introduced by discontinuities. If the object depicted in  $I_{s,\Theta}^d$  is shown in front of a distant background in  $I_{s,\text{in}}^d$ , any inaccuracy at the edges of the object will cause the error to become very high, as the difference between the synthesis and the input for these pixels is very large. In theory this is no problem, because there should always be a deformation  $\Theta$  causing  $\text{Dom}(I_{s,\Theta}^d)$  to only cover the object pixels in  $I_{s,\text{in}}^d$ . However, in real sequences, the chance

of inaccurate depth data and calibration, as well as insufficient modeling capabilities of the deformation model will often cause object contours to not match exactly in every pixel.

In order to prevent the optimization from bending object edges towards the background compensating the inaccuracies, a simple robustification can be introduced. Let  $d_{\max} \in \mathbb{R}$  be a threshold value for the depth difference, then the robustified version of (4.2.4) is given by

$$e_d(\Theta) = \sum_{(u,v) \in D} \frac{\min(d_{\max}, \|I_{s,\Theta}^d(u,v) - I_{s,\text{in}}^d(u,v)\|_2)}{|D|}. \quad (4.2.5)$$

Using the robustified depth error will still motivate the optimization to reduce the number of object pixels matched with the background, as each pixel contributes an error of  $d_{\max}$ , but it will keep the optimization from bending object edges towards the background, as the error remains constant at  $d_{\max}$ , not rewarding such an action.

Another problem using this error formulation is that it scales with the size of the scene that is observed, making it hard to interpret an error value, especially if compared to an average color error which does not scale with the scene. The solution here is to consider only depth difference ratios instead of the absolute differences, formulating  $e_d$  via

$$\begin{aligned} e_d(\Theta) &= \sum_{(u,v) \in D} \frac{1}{|D|} \min \left( d_{\max}, \left| \frac{I_{s,\Theta}^d(u,v) - I_{s,\text{in}}^d(u,v)}{I_{s,\text{in}}^d(u,v)} \right| \right) \\ &= \sum_{(u,v) \in D} \frac{1}{|D|} \min \left( d_{\max}, \left| \frac{I_{s,\Theta}^d(u,v)}{I_{s,\text{in}}^d(u,v)} - 1 \right| \right). \end{aligned} \quad (4.2.6)$$

This formulation also causes depth differences which are further away from the sensor to be weighted less than differences close to the sensor, implicitly regarding the accuracy behavior of most depth cameras<sup>3</sup>.

Another possible error source is the fact that the set of pixels  $D$  can be significantly reduced by finding a deformation reducing  $\text{Dom}(I_{s,\Theta}^d)$  (see eq. 4.2.3). This can happen for example if an object is thin and the deformation function is able to turn the thin side towards the sensors reducing the amount of projection pixels to a minimum. This does not necessarily lead to a false

---

<sup>3</sup>A more in-depth analysis of sensor specific noise characteristics and their incorporation into the analysis can be found in Section 6.2

## 4. Analysis by Synthesis

minimum in the error function  $e_d$ , but it creates the potential for one, as fewer pixels are usually matched more easily than large image areas. A solution to this problem is introduced in Chapter 8.1 as part of a self occlusion handling strategy.

### 4.2.3 Stretch Constraint

Formulating a deformation tracking problem by direct comparison with depth and color images can introduce singularities and degenerated solutions, as the consistency with the input images does not necessarily coincide with a plausible solution. If, e. g., a NURBS deformation model is used ( $\mathcal{D} = \mathcal{B} \dots$ ), the optimization might set the control points in  $P$  very close to each other, such that a rendering of  $\mathcal{D}(V, \Theta)$  degenerates to a single pixel, which can easily be matched with some point in the image.

This degeneration due to scale changes can easily be penalized by introducing a stretch error. Let  $\Psi \subset \mathbb{R}^3 \times \mathbb{R}^3$  denote the set of point pairs  $(\mathbf{v}, \mathbf{v}') \in \Psi$  that should maintain equidistancy over a sequence and during optimization. Let  $\Theta_0$  denote the initial, undeformed state, then

$$\|\mathcal{D}(\mathbf{v}, \Theta) - \mathcal{D}(\mathbf{v}', \Theta)\|_2 \stackrel{!}{=} \|\mathcal{D}(\mathbf{v}, \Theta_0) - \mathcal{D}(\mathbf{v}', \Theta_0)\|_2, \forall (\mathbf{v}, \mathbf{v}') \in \Psi. \quad (4.2.7)$$

for every  $\Theta$ .

The most intuitive way to choose  $\Psi$  is to assemble all triangle edges of an object, i. e.,

$$\Psi = \bigcup_{(t_0, t_1, t_2) \in T} \{(\mathbf{v}_{t_0}, \mathbf{v}_{t_1}), (\mathbf{v}_{t_1}, \mathbf{v}_{t_2}), (\mathbf{v}_{t_2}, \mathbf{v}_{t_0})\}, \quad (4.2.8)$$

with  $\mathbf{v}_i \in V$  denoting the indexed elements in  $V$ . Choosing  $\Psi$  this way keeps the object surface from stretching and shrinking but it is also computationally expensive to calculate for large meshes. A more feasible  $\Psi$  can be generated by selecting only a small number of relevant surface points distributed over the object surface, or, in case of  $\mathcal{D} = \mathcal{B} \dots$ , on the NURBS surface. Figure 3.3 and Figure 3.4 show such a subset of points as grid points of the green NURBS visualization.

Taking the NURBS deformation space as an example, the manifold within  $\mathbb{R}^n$  in which a  $\Theta$  (or  $P$ ) meets the stretch condition described in (4.2.7) has about half the number of dimensions of the  $\mathbb{R}^n$  it is embedded in. Using measure theory [Bau01], one can easily see that navigating  $\Theta$  through the

manifold during optimization is at least difficult, given its non-linear nature caused by the euclidean metric in (4.2.7). Given a certain amount of inaccuracy it is even impossible to tap into the manifold, because its measure equals zero within the  $\mathbb{R}^n$ . A projection onto the manifold in each step as it is done by Lagrange multipliers [Ber95] is also not feasible due to the lack of a projection function in the generic case.

An easy solution to this problem is to relax the condition in (4.2.7) by defining a stretch error  $e_s$ , describing the distance of a current guess  $\Theta$  to the manifold:

$$e_s(\Theta) = \sum_{(v,v') \in \Psi} \left| \frac{\|\mathcal{D}(v, \Theta) - \mathcal{D}(v', \Theta)\|_2 - \|\mathcal{D}(v, \Theta_0) - \mathcal{D}(v', \Theta_0)\|_2}{|\Psi|} \right|. \quad (4.2.9)$$

This will allow the optimization to move outside of the stretch constraint manifold while still trying to find a solution as close as possible to the constraint. To become independent to the scale of the scene, a normalization analog to the depth error (4.2.6) can be performed by setting:

$$e_s(\Theta) = \sum_{(v,v') \in \Psi} \frac{1}{|\Psi|} \left| \frac{\|\mathcal{D}(v, \Theta) - \mathcal{D}(v', \Theta)\|_2}{\|\mathcal{D}(v, \Theta_0) - \mathcal{D}(v', \Theta_0)\|_2} - 1 \right|. \quad (4.2.10)$$

### Evaluation Preview

The NURBS deformation model combined with the synthesis and analysis methods introduced in Chapter 4.1 and 4.2 are the default components of the AbS system introduced in this thesis and have been tested thoroughly. Many of the experiments described in Chapter 7 demonstrate the validity and versatility using artificial and real data. The system is able to e. g., track cloth, teddy bears, and deforming rubber objects.

## 4.2.4 Error Fusion

Having multiple error functions with different modalities introduces the challenge of combining these error values in a reasonable way. The approach denoted in (4.2.1) formulates the problem as choosing the weights  $\varphi_s$  for each sensor or constraint. This might be done empirically, adding prior knowledge to the algorithm, but nevertheless, a more sound, automatic, and data driven approach to finding these weights is desirable. In Chapter 6 two concepts for

## 4. Analysis by Synthesis

choosing the weights  $\varphi_s$  will be discussed: automatic weight adjustment and statistical accuracy modeling.

### 4.3 Prior Knowledge

In many deformation tracking applications, some information about the tracked object or the environment is already known. One way to go about the integration of this prior knowledge is to incorporate it into the design of the deformation function  $\mathcal{D}$ , which may be tedious but can result in very efficient tracking algorithms (see Section 9.3).

Due to the direct approach, prior knowledge can also be implemented in a straight forward way by simply penalizing parameters that yield synthesized states not conforming to the prior knowledge. This is done with regard to the optimization scheme described in Chapter 5.

#### 4.3.1 Spatial Constraints

A simple example of such a spatial constraint is a table top or the floor. If the system is aware of such a geometrical entity it can strictly penalize solution which yield 3D points inside or below the geometric area. As the optimization does not take the actual value of fitness function  $\mathcal{F}$  into account, but only the relative order in magnitude between a small set of evaluations (see Chapter 5), such a strict penalty can be established by setting the error value to the largest value it can express (e. g., either  $\infty$  or about  $10^{308}$  for double-precision floating point variables), making sure that it is ranked least in that order:

$$e_g(\Theta) = \begin{cases} \infty & \mathcal{D}(V, \Theta) \cup G \neq \emptyset \\ 0 & \mathcal{D}(V, \Theta) \cup G = \emptyset \end{cases}, \quad (4.3.1)$$

with  $G \subset \mathbb{R}^3$  describing the forbidden geometrical area. Such a geometry can be arbitrarily complex, as long as a collision with a deformed object can be calculated in a reasonable amount of time. And since  $G$  may vary from frame to frame, known moving objects can be modeled this way as well. If  $G$  is given as a line in space, it can even model a light barrier with result  $l \in \{\text{true}, \text{false}\}$  such that:



$$e_1(\Theta) = \begin{cases} \infty & \mathcal{D}(V, \Theta) \cup G \neq \emptyset \wedge \neg l \\ \infty & \mathcal{D}(V, \Theta) \cup G = \emptyset \wedge l \\ 0 & \mathcal{D}(V, \Theta) \cup G = \emptyset \wedge \neg l \\ 0 & \mathcal{D}(V, \Theta) \cup G \neq \emptyset \wedge l \end{cases}. \quad (4.3.2)$$

Multiple barriers or sensors with a similar output can be modeled analogously, such that  $e_1$  yields a high value if one of the sensor conditions is not met, and zero otherwise.

### 4.3.2 Temporal Constraints

Another constraint that is often used in deformation tracking is temporal consistency. This can be achieved in multiple ways, for example by defining a maximum speed to all vertices, resulting in a geometrical area for each vertex around the best estimate of the preceding frame for that vertex and the size rendered from the speed limit and the time that has passed between frames. Unlike a geometrical constraint, this limitation should be applied in a soft way, since it is a heuristic with no hard line between plausible and impossible. Let  $d_{\text{free}}$  denote the distance in which each vertex is allowed to be located without being penalized, then a temporal error can be described by:

$$e_t(\Theta) = \sum_{v \in V} \max(0, \|\mathcal{D}(v, \Theta) - \mathcal{D}(v, \Theta')\|_2 - d_{\text{free}}), \quad (4.3.3)$$

$\Theta'$  denoting the best guess of the preceding frame.

A more natural way to describe temporal consistency is by emulating inertia, i. e., formulating a term for momentum conservation. In the most simple way, this error term involves the result of the preceding frame  $\Theta'$  as well as the result of the frame before that,  $\Theta''$ . For a vertex  $v \in V$ , the momentum can be easily calculated by  $(\mathcal{D}(v, \Theta') - \mathcal{D}(v, \Theta''))$  with respect to the time passed between frames. Assuming a constant frame rate, an inertia error can be formulated via:

$$e_i(\Theta) = \sum_{v \in V} \max(0, \|2\mathcal{D}(v, \Theta') - \mathcal{D}(v, \Theta'') - \mathcal{D}(v, \Theta)\|_2 - i_{\text{free}}). \quad (4.3.4)$$

Similar to  $d_{\text{free}}$  in 4.3.3,  $i_{\text{free}}$  describes the amount of unpenalized change in momentum. Such a constraint can be used to lessen the effects of high frequency sensor noise as it is produced by ToF cameras.

## 4.4 Sparsification

One important aspect which sets the AbS approach described in this thesis apart from other direct methods is its ability to work in real-time on complex data. AbS usually has to deal with a large optimization space and complex parameter evaluation, as every evaluation entails the generation of a complete set of images (one for each sensor). To perform fast tracking nevertheless, all elements of the AbS algorithm have to be fast and efficient. It requires an efficient deformation function (see Chapter 3.3 or Chapter 9.3.1), a fast optimization scheme, which performs as few fitness function evaluations as possible (see Chapter 5), and a fast way to synthesize and analyze the images.

Modern Graphics Processing Units (GPUs) are very powerful in generating these synthetic images, but are inferior to the CPU when it comes to evaluation of hierarchical or otherwise complexly associated data terms. Performing a complex deformation on the CPU, however, and then transferring the deformed set of vertices to the graphics card for rendering does not allow for an efficient evaluation either, as the transfer speed between Random Access Memory (RAM) and graphics memory is limited. Synthesizing the images on the CPU via software rendering, however, performs even worse.

The solution to this problem is a simplification of the rendering process such that it can easily be performed by the CPU. The idea behind the sparse synthesis is to only synthesize a set of significant points of the object rather than the complete set of triangles. As the triangles are only interpolations between the vertex coordinates, these significant points are given by the set of vertices  $V$ , which, if necessary, can be sparsified even more by removing randomly or heuristically chosen vertices from it. A very adaptive and efficient way to approach the sparsification task is introduced in Section 4.4.2.

### 4.4.1 Sparse Synthesis

With  $V$  as the sparsified set of object vertices,  $I_{s,in}^d$  denoting the depth input image and  $\mathcal{P}_s$  providing the projection function (see (4.1.10)), the sum over all differences of relevant pixels can be calculated without explicitly creating a synthesized image. This is done by calculating the difference between the deformed vertex - camera distance  $\|\mathbf{c}_s - \mathcal{D}(\mathbf{v}, \Theta)\|_2$  and the depth value inside the input image representing the measurement for such a vertex  $I_{s,in}^d(\mathcal{P}_s(\mathcal{D}(\mathbf{v}, \Theta)))$ :

$$e_d(\Theta) = \sum_{\mathbf{v} \in V} \left| \frac{\|\mathbf{c}_s - \mathcal{D}(\mathbf{v}, \Theta)\|_2 - I_{s,in}^d(\mathcal{P}_s(\mathcal{D}(\mathbf{v}, \Theta)))}{|V|} \right|. \quad (4.4.1)$$

Note that, although only a few image pixels may influence the error value of the final result  $\Theta^*$ , all relevant pixels are being probed during the optimization process, as the image coordinates  $\mathcal{P}_s(\mathcal{D}(v, \Theta))$  change according to the deformation described by  $\Theta$ .

Similar to (4.2.5), the error term in 4.4.1 is robustified:

$$e_d(\Theta) = \sum_{v \in V} \max \left( d_{\max}, \left| \frac{\|c_s - \mathcal{D}(v, \Theta)\|_2 - I_{s,\text{in}}^d(\mathcal{P}_s(\mathcal{D}(v, \Theta)))}{|V|} \right| \right). \quad (4.4.2)$$

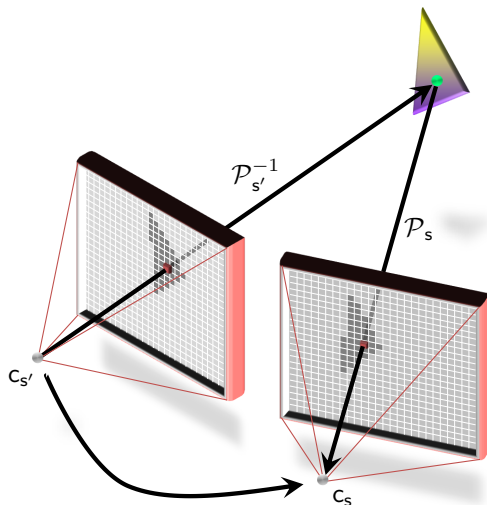
To reduce the number of operations required to calculate  $\mathcal{P}_s$ , it is possible to transform the problem by rotating and translating the world coordinate system such that it matches the camera coordinate system, i. e.,  $\mathbf{A}_s = \mathbf{1}$ ,  $\mathbf{R}_s = \mathbf{1}$ , and  $c_s = \mathbf{0}$ . If the image distortion is removed from the input image (see [SBK08]), i. e.,  $\delta_s = \mathbf{1}$ , the projection function  $\mathcal{P}_s$  (see Eq. (4.1.10)) becomes very simple, so that (4.4.2) can be rewritten as:

$$e_d(\Theta) = \frac{1}{|V|} \sum_{v \in V} \max \left( d_{\max}, \left| \|\mathcal{D}(v, \Theta)\|_2 - I_{s,\text{in}}^d \left( \frac{f_s}{\mathcal{D}(v, \Theta)_z} \mathcal{D}(v, \Theta)_{(x,y)} \right) \right| \right). \quad (4.4.3)$$

This means that the entire synthesis and analysis step has been reduced to one deformation function call, one subtraction, four multiplications, and one conditional for each vertex in  $V$ . Efficiently implemented, this function can evaluate hundreds or thousands of vertices and still allow to optimize with several fps (see Chapter 7).

The efficient evaluation is achieved by setting  $\mathbf{A}_s$  to the world coordinate system, which implies, that any other sensor that should be incorporated is either evaluated less efficiently or has to have the same camera center as the depth camera described by  $s$ . Changing the camera perspective of an input image with different camera settings  $s'$  can be done by warping the image [CS09] or transforming it in 3D. For depth images, this is simply done by  $\mathcal{P}_{s'}^{-1}$  back-projecting [HZ00] each pixel into 3D space, implicitly applying the rigid transformation between the source and the destination sensor and  $\mathcal{P}_s$  projecting it back onto the image plane (see Figure 4.6). The value inserted in the target depth image is not the corresponding value of that image pixel in  $I_{s'}^d$ , but the actual distance between the back-projected point and the camera center  $c_s$ .

#### 4. Analysis by Synthesis

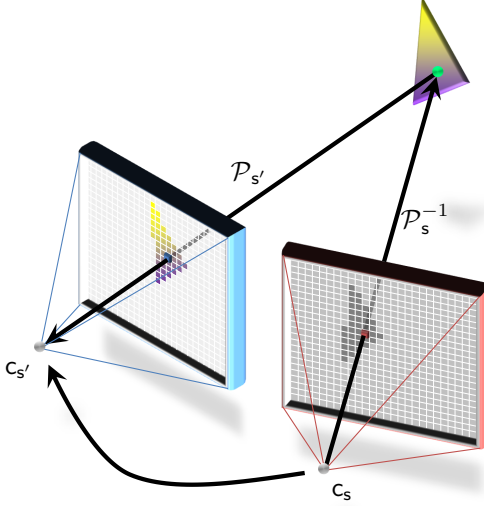


**Figure 4.6.** Warping a depth image into another depth image can be performed by back-projecting [HZ00] each pixel into 3D space and projecting it into the new camera center.

For color cameras, at least one depth sensor is required to directly perform a camera center transformation in 3D. Warping the color image at  $s'$  into the depth image at  $s$  can be done by  $\mathcal{P}_s^{-1}$  back-projecting the (destination) depth image into 3D and  $\mathcal{P}_{s'}$  projecting each pixel into the image plane of the color camera. This allows to look up a color value for each (valid) depth image pixel, yielding a color image from the depth sensor perspective (see Fig. 4.7).

In a standard setup, consisting of a depth- and a color sensor (as e.g., in a *Kinect*), the color image is warped into the depth image, which is assumed to be aligned with and located at the world coordinate system. Note that warping the color image can introduce aliasing and quantization effects and is mutual exclusive to the explicit Bayer pattern handling described in Section 4.2.1.

Let  $C$  be the indexed set of vertex colors, such that each  $(r_i, g_i, b_i) \in C$  denotes the color of the vertex  $v_i \in V$ . With an input color image  $I_{s,\text{in}}^c$  at hand, which has been warped into the camera coordinate system of  $s$ , the sparse color error can be denoted as the direct Sum of Absolute Differences



**Figure 4.7.** Warping a color image into a depth image can be performed by back-projecting each pixel of the destination depth image into 3D space and projecting it into the color camera center to associate a color value.

(SAD) in the color space, similar to 4.4.1:

$$e_c(\Theta) = \sum_{\mathbf{v}_i \in V} \frac{\|(r_i, g_i, b_i) - I_{s, \text{in}}^d(\mathcal{P}_s(\mathcal{D}(\mathbf{v}_i, \Theta)))\|_1}{|V|}, \quad (4.4.4)$$

which can be written analogously to 4.4.3 as

$$e_c(\Theta) = \frac{1}{|V|} \sum_{\mathbf{v}_i \in V} \left\| (r_i, g_i, b_i) - I_{s, \text{in}}^c \left( \frac{f_s}{\mathcal{D}(\mathbf{v}_i, \Theta)_z} \mathcal{D}(\mathbf{v}_i, \Theta)_{(x,y)} \right) \right\|_1. \quad (4.4.5)$$

Because the image coordinate of each vertex calculated in (4.4.3) can be reused in (4.4.5), the color error calculation only costs three additional subtractions, two additions and three absolute value calculations.

In general, the sparse synthesis may be less accurate than AbS performed by complete rendering and image based comparison of color and depth values, but the big advantage of the sparse synthesis is its speed (it allows real-time processing) as well as the immunity from singularities in the fitness function

## 4. Analysis by Synthesis

that occur when the deformation minimizes the number of pixels the object is projected on (already discussed in Section 4.2.2), as the normalization is not performed by the number of pixels (see Eq. (4.2.2) and (4.2.4)) varying with  $\Theta$ , but by  $|V|$  which is constant over the complete optimization.

### Evaluation Preview

Sparse synthesis has been used in the tests of Section 7.3.1, as well as in the FlexPad system introduced in 9.3 (See Section 9.3.2 for evaluation results), proving the resulting real time capabilities of the AbS system while maintaining accurate tracking results.

### 4.4.2 Randomized Online Sparsification

The reduction of a triangle mesh to its vertices as it is done by the sparse synthesis does not necessarily have a large impact on the resulting problem formulation. As an example: if an object is tracked in a color- and depth image sequence and the reference object is directly extracted from the initial image, each pixel of this initial image is converted into a vertex. Synthesizing the sensor output via a proper 3D rendering of the corresponding triangle mesh will result in the very same image as rendering the colored vertex set generated by sparse synthesis from the image data. The synthesis only becomes visibly sparse as the object is moving further towards the camera or performs deformations that stretch parts of the object.

The speed-up capabilities of sparse synthesis come to the fore when reference point sets (models) are used that are significantly reduced in their number (see e. g., FlexPad in Section 9.3). Unlike the synthesis using traditional rendering methods, the sparse synthesis has a linear relation between processing time and the number of reference vertices. But as any removal of data potentially effects the quality of the result, a predetermined sparsification of the reference model is not the optimal solution and would stray from the initial motivation to use the complete available information.

One solution to this problem is to use “randomized online sparsification”, i. e., to randomly determine a subset of vertices each time a sparse synthesis is generated. This is very easily implemented, by exchanging the vertex sets  $V$  in Equation (4.4.3) and (4.4.5) by randomly chosen subsets every time the fitness function is evaluated. This will introduce a certain error to the fitness function, but as this error has no systematic nature (each vertex is picked with an equal chance each evaluation), it has much more the nature of a Gaussian noise than an actual error. If this noise is parametrized by

its deviation  $\sigma_s$ , Appendix C deduces how the effects of sparsification can be estimated in their effects to the overall optimization process, allowing to deduce sparsification rates from required optimization stability. In fact, as will be discussed in Section 5.1.1, there is a high probability that random online sparsification does not affect the end result at all, if the sparsification rate is chosen accordingly. These sparsification rates not only allow to estimate the sparsification induced error for each optimization step, but also allow to adjust the sparsification rate online during the optimization process at each iteration, providing a performance boost without requiring to trade-off against a lower accuracy.

### **Evaluation Preview**

The randomized online sparsification is theoretically discussed in Appendix C and tested in Section 7.3.1. In these tests it is shown that even for noisy input data sparsification rates of 5% or even 1% yield comparable results. Adapting sparsification during the optimization even allows to maintain high accuracy by reducing the computational work to 10% and less.





# Optimization

Once the fitness criteria are defined completely, the goal of the AbS algorithm can be formulated as finding the deformation parameters  $\Theta^*$  that fit the observations given by a set of input images and other constraints best. Using the fitness function  $\mathcal{F}$  as defined in Chapter 4, the task can be notated as finding

$$\Theta^* = \arg \max_{\Theta \in \mathbf{R}^n} \mathcal{F}(\Theta). \quad (5.0.1)$$

For large and complex deformations the number of parameters in  $\Theta$  can easily become one hundred and more. The common approach to optimize non-linear problems like this is to perform a local linearization of the fitness function to determine in which direction of the current vantage point the optimum is located. This local optimization works under the assumption that the fitness function is convex or at least convex enough to determine the optimum by locally derived steps.

Unfortunately, the problem at hand cannot assumed to be convex, as it is directly based on the input images which contain noise, discontinuities, and other features that impede successful local optimization on real data. Tests have shown that local optimization e. g., with a Levenberg-Marquardt algorithm [Mar63] lead to insufficient tracking results. To efficiently optimize  $\mathcal{F}$  nevertheless, a fast, global optimization scheme is required that is able to handle noisy and non-convex data in high-dimensional problems. One algorithm that proved to be very powerful in this regard<sup>1</sup> is the CMA-ES algorithm [HR10], which will be used and adapted to optimize the AbS fitness function.

---

<sup>1</sup>CMA-ES is the winner of the 2009 Black-Box Optimization Benchmarking Competition (BBOB)

## 5. Optimization

The following Section 5.1 will provide a technical introduction to the CMA-ES algorithm and will in most parts follow the motivation and definitions given by Hansen et al. in [Han06]. For a more in-depth motivation of the optimizer components, refer to [HO01, Han06]<sup>2</sup>. Section 5.2 will then describe the extensions to the CMA-ES method to optimize its application to the AbS problem.

### 5.1 CMA-ES

The family of distribution estimation algorithms falls into the class of global distribution-based optimizers, which, instead of following a gradient or local linearizations of the fitness function, describe their state by a distribution. In the case of CMA-ES, it is a multivariate Gaussian (mutation-) distribution, which in most cases is randomly sampled with respect to that distribution to analyse the fitness function. CMA-ES was proposed in 2001 by Hansen and Ostermeier [HO01] and, originating from evolutionary computation, it introduced a very efficient concept of parametrizing the mutation distributions in a self-adapting fashion. Mutation distributions define the probability space for mutations within a population of fitness function samples (also referred to as individuals in classical, evolutionary computation). While the configuration of these distributions usually requires an appropriate choice of meta-parameters, CMA-ES manages a complete self-adaptation, which proves to be very efficient [HR10] and will be discussed in Section 5.1.2 (derandomization, cumulation).

In contrast to other global optimization schemes like Particle Swarm Optimization (PSO) [KE95], CMA-ES does not keep track of single individuals over multiple iterations. Instead it regenerates its population in every iteration based on the current mutation distribution, which is given by a Gaussian distribution with a mean value  $\mathbf{m} \in \mathbb{R}^n$  and a covariance matrix  $\tilde{\mathbf{C}} \in \mathbb{R}^{n \times n}$  for a fitness function  $\mathcal{F} : \mathbb{R}^n \rightarrow \mathbb{R}$ .  $\mathbf{m}$  and  $\tilde{\mathbf{C}}$  are updated during every iteration based on current and past evaluation results.

Formulating the mutation distribution as a multivariate Gaussian distribution allows CMA-ES to be much more scalable with respect to the number of dimensions than optimizers that use more complex distribution density approaches, e. g., CONDENSATION [IB98]. It also provides useful analogies to other, more local optimization schemes as the covariance matrix can be

---

<sup>2</sup>As an addition to [HO01, Han06], a good CMA-ES tutorial by Nicolaus Hansen can also be found at <https://www.lri.fr/~hansen/cmatutorial.pdf>

interpreted as an approximation of the inverse Hessian matrix, used for step size control in optimizers like the Levenberg-Marquardt algorithm [Mar63].

CMA-ES is coordinate system free, i. e., invariant to rotations in the parameter space, which is an important feature for the class of applications discussed in this thesis. This is not a self-evident property for an evolutionary optimization algorithm, because cross-over operations, a very common recombination strategy in evolutionary computing, is coordinate system dependent, implicitly assuming a certain degree of separability of the fitness function. For deformation and transformation parameters, which more or less directly relate to points in the 3D space, this assumption does obviously not hold, as these point sets can have an arbitrary orientation in 3D space.

### 5.1.1 Covariance Matrix Adaptation

Let  $\mathbf{m}_0$  denote the initial mean vector in the parameter space and let  $\tilde{\mathbf{C}}_0$  be the initial distribution accordingly. If no further information about the search space is available, the covariance matrix is usually initialized as a diagonal matrix containing the expected variances of each parameter variable.

For an iteration  $i$ , let  $\mathbf{m}_i$  and  $\tilde{\mathbf{C}}_i$  describe the distribution of that optimization iteration. Let  $\mathcal{N}_{(\mathbf{m}, \tilde{\mathbf{C}})} : \mathbb{R}^n \rightarrow [0, 1]$  denote the probability density function (PDF) of the normal distribution around  $\mathbf{m}$  with covariance  $\tilde{\mathbf{C}}$ ,

$$\mathcal{N}_{(\mathbf{m}, \tilde{\mathbf{C}})}(\Theta) = \frac{1}{\sqrt{(2\pi)^n \det(\tilde{\mathbf{C}})}} \exp\left(-\frac{1}{2}(\Theta - \mathbf{m})^T \tilde{\mathbf{C}}^{-1}(\Theta - \mathbf{m})\right). \quad (5.1.1)$$

For a more convenient handling of the covariance  $\tilde{\mathbf{C}}$  in the following Sections, let  $\tilde{\mathbf{C}}$  be decomposed into its size, given by the (induced) norm  $\sigma^2 = \|\tilde{\mathbf{C}}\|$  and the normalized matrix  $\mathbf{C}$  with  $\det(\mathbf{C}) = 1$ , such that  $\sigma^2 \mathbf{C} = \tilde{\mathbf{C}}$  and let

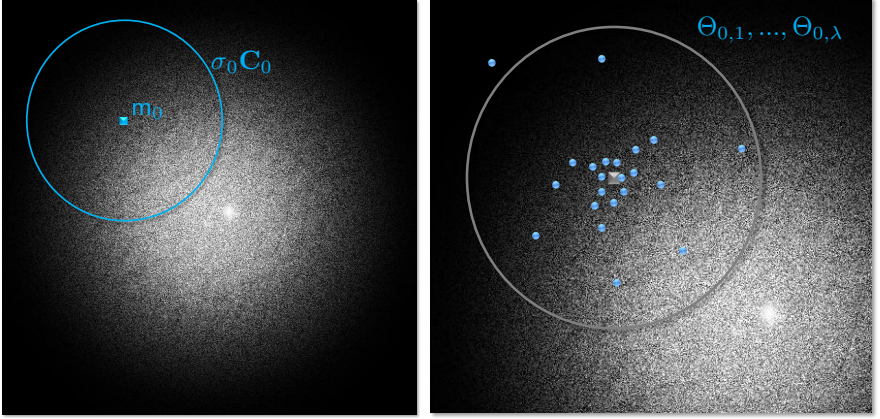
$$\mathcal{N}_{(\mathbf{m}, \sigma, \mathbf{C})}(\Theta) = \frac{1}{\sqrt{(2\pi\sigma^2)^n}} \exp\left(-\frac{1}{2\sigma^2}(\Theta - \mathbf{m})^T \mathbf{C}^{-1}(\Theta - \mathbf{m})\right). \quad (5.1.2)$$

Figure 5.1 (left) depicts an example for an initial distribution.

#### Mean Update

For a given population size  $\lambda \in \mathbb{N}$ , the translation from iteration  $i$  to  $i + 1$  is made by first distributing  $\Theta_{i,0}, \dots, \Theta_{i,\lambda-1}$  according to  $\mathcal{N}_{(\mathbf{m}_i, \sigma_i, \mathbf{C}_i)}$  and

## 5. Optimization



**Figure 5.1.** Example visualization of fitness function  $\mathcal{F}$  (dark= low fitness, bright= high fitness). Left: initial distribution provided by mean  $\mathbf{m}_0$  and the normalized covariance  $\mathbf{C}_0$  scaled by distribution size  $\sigma_0$ . Right (zoomed in): sample population of  $\lambda$  parameter vectors  $\Theta$ , distributed according to  $\mathcal{N}_{(\mathbf{m}, \sigma, \mathbf{C})}$

evaluating the fitness function  $\mathcal{F}$  for each of these parameters (see Fig. 5.1 (right)). The resulting fitness values are sorted by their magnitude afterwards. For easier notation (w.l.o.g) let the indexing of the parameter vectors  $\Theta_{i,j}$ ,  $j = 0, \dots, \lambda - 1$  correspond to the order of the resulting fitness values in each iteration  $i$ , such that

$$\mathcal{F}(\Theta_{i,0}) \leq \mathcal{F}(\Theta_{i,1}) \leq \dots \leq \mathcal{F}(\Theta_{i,\lambda-1}). \quad (5.1.3)$$

A vector of constant weights  $\mathbf{w} = (w_0, \dots, w_{\lambda-1}) \in [0, 1]^\lambda$  with

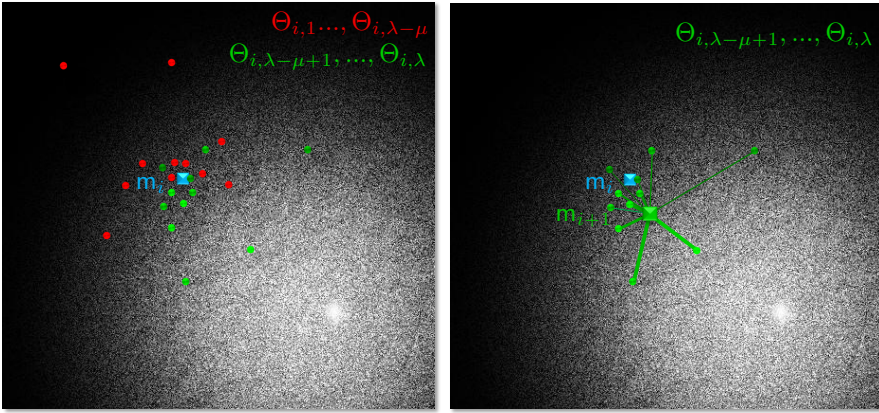
$$w_0 \leq w_1 \leq \dots \leq w_{\lambda-1}, \quad (5.1.4)$$

and

$$\sum_{j=0}^{\lambda-1} w_j = 1, \quad (5.1.5)$$

is then used to define the new mean value as

$$\mathbf{m}_{i+1} = \sum_{j=0}^{\lambda-1} w_j \Theta_{i,j}. \quad (5.1.6)$$



**Figure 5.2.** Example visualization of fitness function  $\mathcal{F}$  (dark= low fitness, bright= high fitness). Left: selection of  $\mu$  out of the  $\lambda$  individuals based on their fitness  $\mathcal{F}(\Theta)$  (green= selected, red= not selected). Right: calculation of a new mean  $\mathbf{m}_{i+1}$  as the weighted sum of the  $\mu$  selected samples.

Figure 5.2 (right) shows a visualization of the calculation of  $\mathbf{m}_{i+1}$ .

Selection, as a fundamental part of evolutionary algorithms, is provided by the selection index  $\mu \in \mathbf{N}_{\leq \lambda}$  indicating how many weights  $w$  are nonzero:

$$\forall j < \lambda - \mu : w_j = 0. \quad (5.1.7)$$

Figure 5.2 (left) depicts an example selection of samples. A reasonable choice for  $\mu$  is  $\mu = \frac{\lambda}{2}$  [Han06] with the weights of the selected samples being

$$\forall j < \lambda : w_j = \begin{cases} 2 \frac{\mu-j}{\mu^2+\mu} & j \geq \mu \\ 0 & \text{else.} \end{cases} \quad (5.1.8)$$

Note that the actual value of  $\mathcal{F}$  is not used within the calculation of  $\mathbf{m}_{i+1}$ , only the order of the results. Since  $\leq$  can be defined on  $\mathbb{R} \cup \{-\infty, \infty\}$ , the optimization is still well defined using the error definitions of Section 4.3, which may yield infinity<sup>3</sup> to mark forbidden areas in the search space. Figure 5.2 also visualizes how the distribution center is not penetrating such forbidden areas, assuming the area boundaries are shaped sufficiently smooth. The new

<sup>3</sup>In the implementation infinity may be replaced by the maximum / minimum value of the corresponding data type.

## 5. Optimization

mean is always within the convex hull of the selected individuals, which are all on the outside of the forbidden area.

Concerning the randomized online sparsification of the synthesis (Section 4.4.2), the disregard of the actual fitness function value entails that the noise in the fitness function of a sparsified synthesis has no effect on the optimization as long as it does not change the order described in equation (5.1.3) of the first  $\mu$  parameters evaluations. One can estimate the probability of this happening by

$$\mathbf{P}(\exists j \in \mathbb{N}_{[\lambda-\mu, \lambda-1]} : \mathcal{F}_s(\Theta_{i,j}) > \mathcal{F}_s(\Theta_{i,j+1})) = \sum_{j=\mu}^{\lambda-1} \frac{j}{2} \int_1^{\infty} \mathcal{N}\left(0, \frac{\sigma_s}{\sigma \|\mathbf{C}\|_{\mathcal{F}}}\right), \quad (5.1.9)$$

with  $\mathcal{F}_s$  as the sparse fitness function and  $\sigma_s$  denoting the noise induced to the fitness function by random sparsification and  $\sigma \|\mathbf{C}\|_{\mathcal{F}}$  denoting the current deviation of the CMA-ES population in the fitness value domain. The deduction of (5.1.9) and an evaluation example can be found in Appendix C.

### Covariance Matrix Update

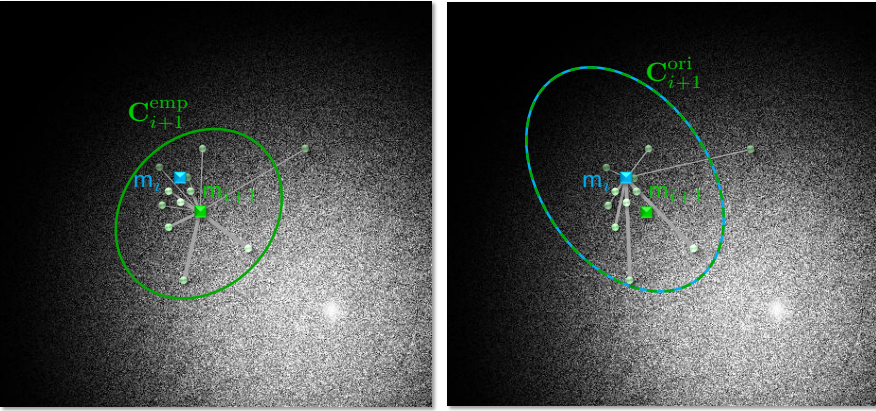
The goal in updating the covariance matrix is to improve the distribution such that it maximizes the chance of sampling high fitness values in  $\mathcal{F}$ . Under the assumption that the set of individuals is large enough to completely estimate such a distribution, the covariance  $\tilde{\mathbf{C}}_{i+1}$  can be calculated from the empirical data gathered by the sample evaluation as the covariance of the population after selection (individuals with index  $(\lambda - \mu + 1)$  to  $(\lambda)$ ):

$$\tilde{\mathbf{C}}_{i+1}^{\text{emp}} = \frac{1}{\mu - 1} \sum_{j=\lambda-\mu}^{\lambda-1} \left( \Theta_{i,j} - \frac{1}{\mu} \sum_{g=\lambda-\mu+1}^{\lambda} \Theta_{i,g} \right) \left( \Theta_{i,j} - \frac{1}{\mu} \sum_{g=\lambda-\mu+1}^{\lambda} \Theta_{i,g} \right)^T \quad (5.1.10)$$

as the empirical, unbiased covariance or, using  $\mathbf{m}_i$ , the original mean value of the distribution

$$\tilde{\mathbf{C}}_{i+1}^{\text{ori}} = \frac{1}{\mu} \sum_{j=\lambda-\mu}^{\lambda-1} (\Theta_{i,j} - \mathbf{m}_i) (\Theta_{i,j} - \mathbf{m}_i)^T. \quad (5.1.11)$$

While  $\tilde{\mathbf{C}}^{\text{emp}}$  is the more accurate distribution estimation based on the selected individuals, it does not regard the samples removed in the selection process.



**Figure 5.3.** Visualization of the covariance matrix update: Left: calculation based on the new mean  $\mathbf{m}_{i+1}$ . Right: a covariance based on the old mean vector  $\mathbf{m}_i$  yields a much better distribution for the following search iteration.

In contrast,  $\tilde{\mathbf{C}}^{\text{ori}}$  contains this information as the “bias” between the complete and the selected population. Figure 5.3 visualizes, how the usage of  $\tilde{\mathbf{C}}^{\text{ori}}$  is more suited for optimizing the search space for the next iteration than  $\tilde{\mathbf{C}}^{\text{emp}}$ .  $\tilde{\mathbf{C}}^{\text{emp}}$  does not regard the direction in which the selected samples are located with respect to the distribution center. Note that the resulting distribution for the next iteration  $\mathcal{N}_{(\mathbf{m}_{i+1}, \tilde{\mathbf{C}}_{i+1})}$  will be centered around  $\mathbf{m}_{i+1}$  in contrast to the  $\mathbf{m}_i$ -centered visualization in Figure 5.3 (right).

To regard the order of the fitness values in the distribution, a weighting, similar to the calculation of the new mean in (5.1.6), can be introduced to Equation (5.1.11):

$$\tilde{\mathbf{C}}_{i+1}^{\text{w}} = \sum_{j=\lambda-\mu}^{\lambda-1} \frac{w_j}{\sigma_i^2} (\Theta_{i,j} - \mathbf{m}_i) (\Theta_{i,j} - \mathbf{m}_i)^T. \quad (5.1.12)$$

## 5.1.2 Update Strategies and Evolution Path

The covariance update  $\tilde{\mathbf{C}}_i \rightarrow \tilde{\mathbf{C}}_{i+1}^{\text{w}}$  is calculated under the assumption that the population of iteration  $i$  is large enough to analyze the vicinity of  $\mathbf{m}_i$ . While this is a very thorough and robust approach, it also lacks efficiency because the search space has to be re-sampled at every iteration by a large number

## 5. Optimization

of function evaluations. Efficiency is introduced to CMA-ES by reducing the population size  $\lambda$  from multiples of  $n$  (dimension of  $\Theta$ ) to a number below  $n$ . To compensate for the missing information about the search space,  $\mathbf{C}$  is not completely recalculated in every iteration but updated to a certain amount. The underlying assumption is, that  $\mathbf{C}_i$  is the approximation of the inverse Hessian matrix of  $\mathcal{F}$  at  $\mathbf{m}_i$ , but of a less noisy and more convex version of  $\mathcal{F}$ , as the effects of noise and local minima average out over the area of distribution. Hence, it is assumed that  $\mathbf{C}$  does not change significantly in its vicinity.

### Exponential Covariance Adaptation

The weighted covariance over all selected samples with respect to their distribution mean can be written as

$$\tilde{\mathbf{C}}_{i+1}^{\text{avg}} = \frac{1}{i+1} \sum_{k=0}^i \sum_{j=\lambda-\mu}^{\lambda-1} \frac{w_j}{\sigma_k^2} (\Theta_{k,j} - \mathbf{m}_k) (\Theta_{k,j} - \mathbf{m}_k)^T. \quad (5.1.13)$$

To avoid a higher impact of large distributions, i. e., with a large  $\sigma$ , compared to smaller distributions, the normalized covariance matrices of each iteration can be used to provide a better scale invariant average:

$$\mathbf{C}_{i+1}^{\text{avg}} = \frac{1}{i+1} \sum_{k=0}^i \mathbf{C}_k^{\text{w}} = \frac{1}{i+1} \sum_{k=0}^i \sum_{j=\lambda-\mu}^{\lambda-1} \frac{w_j}{\sigma_k^2} (\Theta_{k,j} - \mathbf{m}_k) (\Theta_{k,j} - \mathbf{m}_k)^T. \quad (5.1.14)$$

But averaging over all past distribution estimates in such a way would slow the adaptation process down significantly, since mean  $\mathbf{m}$  and scale  $\sigma$  of the distribution change. As the iteration count increases, local properties of the search space will require more and more iterations to become dominant in  $\mathbf{C}^{\text{avg}}$ .

A more suitable way of averaging is performed by having a weighted average, which is (exponentially) prioritizing recent estimators over older ones:

$$\mathbf{C}_{i+1}^{\text{exp}} = (1 - c_\mu)^{(i+1)} \mathbf{C}_0^{\text{w}} + c_\mu \sum_{k=1}^i (1 - c_\mu)^{(i-k)} \mathbf{C}_k^{\text{w}} \quad (5.1.15)$$

or in a more implementation-oriented, recursive way:

$$\mathbf{C}_{i+1}^{\text{exp}} = (1 - c_\mu) \mathbf{C}_i^{\text{exp}} + c_\mu \mathbf{C}_{(i+1)}^{\text{w}}. \quad (5.1.16)$$



$c_\mu \in (0, 1]$  denotes the learning rate of the sequence, i. e., the weight a current estimation  $\mathbf{C}^w$  has on the updated average. Looking at it the other way around, the inverse learning rate  $c_\mu^{-1}$  is also referred to as the *backward time horizon* as it indicates how many of the most recent iterations have to be summed up to gain about 63% of the influence on the averaged covariance  $\mathbf{C}_{i+1}^{\text{exp}}$ .

### Evolution Path

The distribution adaptation as defined in (5.1.16) does not regard the sign of the deviation vectors  $(\Theta - \mathbf{m})$  (see also Fig. 5.3 (right)), as Gaussian distributions are always point-symmetric. To exploit directional information nevertheless, CMA-ES keeps track of the direction the mean vector moves in over the iterations by summing up the weighted mean values of the movement directions of recent optimization steps to a vector  $\mathbf{e} \in \mathbb{R}^n$ , a strategy referred to as *cumulation*. The weighting of the direction vectors of each iteration is exponentially, similar to (5.1.16), where  $\mathbf{e}_0 = \mathbf{0}$  and

$$\mathbf{e}_{i+1} = (1 - c_e)\mathbf{e}_i + \sqrt{c_e(2 - c_e)}\|\mathbf{w}\|_2^{-2} \frac{\mathbf{m}_{i+1} - \mathbf{m}_i}{\sigma_i}, \quad (5.1.17)$$

where  $c_e \in (0, 1]$  denotes the learning rate, i. e.,  $c_e^{-1}$  is the *backward time horizon* of the evolution path  $\mathbf{e}$ , similar to  $c_\mu$ . Note that  $\sqrt{c_e(2 - c_e)}$  serves as a normalization counter part to  $(1 - c_e)^2$  in the  $\|\cdot\|_2$  norm, as  $(1 - c_e)^2 + \sqrt{c_e(2 - c_e)}^2 = 1$  and  $\|\mathbf{w}\|_2^{-1}$  serves as an approximation for the sum of the weighted selected individuals<sup>4</sup>, i. e.,  $\|\mathbf{w}\|_2^{-2}\mathcal{N}(0, \mathbf{C}) \sim \sum_{j=0}^{\mu-1} w_j \mathcal{N}(0, \mathbf{C})$ .

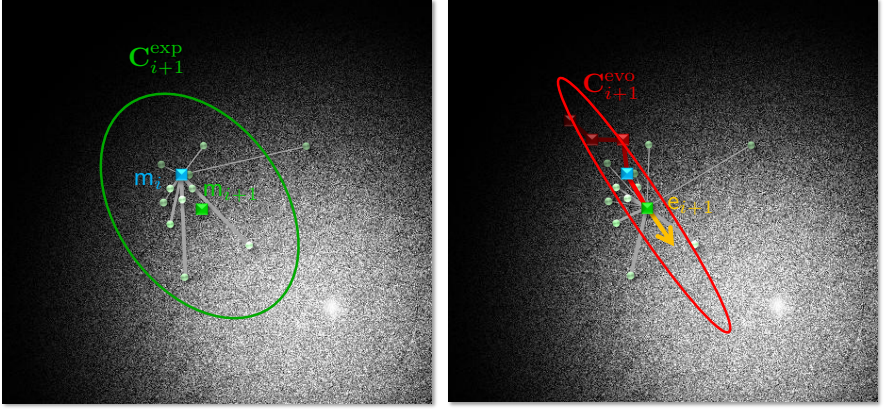
The resulting evolution path vector  $\mathbf{e}_i$  for an iteration  $i$  can be interpreted in various ways, either as the direction in each dimension that was favored by the selection in the recent updates, or an approximation of the gradient of  $\mathcal{F}$ , or the current movement direction, or a “super sample”, an averaged, weighted mix from all selected individuals (see definition of  $\mathbf{m}$ ). The latter can be used to formulate an alternative update step compared to (5.1.16). Given the covariance  $\mathbf{C}_i^{\text{evo}}$  at iteration  $i$ , the learning rate  $c_1 \in (0, 1]$ , and the corresponding evolution path  $\mathbf{e}_i$ , the update

$$\mathbf{C}_{i+1}^{\text{evo}} = (1 - c_1)\mathbf{C}_i^{\text{evo}} + c_1\mathbf{e}_{i+1}\mathbf{e}_{i+1}^T \quad (5.1.18)$$

---

<sup>4</sup>The scalar  $\mu_{\text{eff}} = \|\mathbf{w}\|_2^{-2} = \left(\frac{\|\mathbf{w}\|_1}{\|\mathbf{w}\|_2}\right)^2$  is also referred to as the *variance effective selection mass*, an indicator for the “size” an update based on the recombination weights  $\mathbf{w}$  introduces to the covariance.

## 5. Optimization



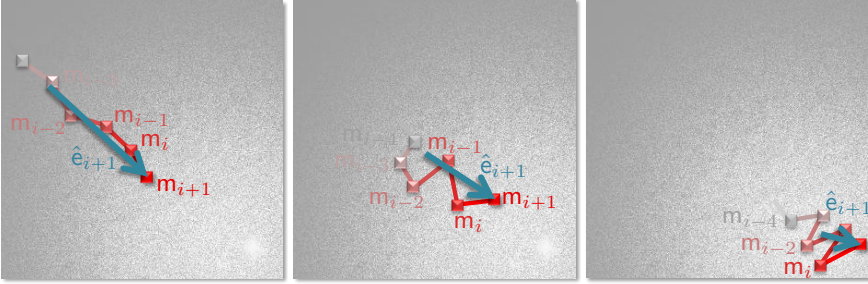
**Figure 5.4.** Visualization of the rank- $\mu$  update (left) compared to the rank-one update (right). Left: the distribution  $\mathbf{C}_{i+1}^{\text{exp}}$  (green ellipse) leads to a widespread search in the parameter space. Right: faster but less exhaustive search is performed by the distribution  $\mathbf{C}_{i+1}^{\text{evo}}$  (red ellipse), which is only based on the evolution path  $\mathbf{e}_{i+1}$  (orange arrow), calculated from the recent mean movements (red trail).

will shape the covariance to stretch along the direction of the evolution path. Based on the number of vectors the update matrix is generated from, Equation (5.1.18) is referred to as the rank-one update, whereas (5.1.16) is called the rank- $\mu$  update. And as the evolution path is calculated from the mean movement, which is defined by a weighted sum of selected individuals, its update direction is basically applying the currently best direction guess.

Hence, the rank-one and the rank- $\mu$  update are two oppositional approaches to sample the search space during optimization. Whereas the rank-one update describes a fast and greedy focus on the best guess, the rank- $\mu$  update tends to spread the search space into all prospective directions and performs a slow, save, and more exhausting search. Figure 5.4 provides a visualization of both update types.

The performed update step is then a combination of both approaches to balance between fast and thorough searching:

$$\mathbf{C}_{i+1} = (1 - c_\mu - c_1)\mathbf{C}_i + c_1\mathbf{e}_{i+1}\mathbf{e}_{i+1}^T + c_\mu \sum_{j=0}^{\lambda-1} \frac{w_j}{\sigma_i^2} (\Theta_{i,j} - \mathbf{m}_i)(\Theta_{i,j} - \mathbf{m}_i)^T. \quad (5.1.19)$$



**Figure 5.5.** Visualization of the anisotropic evolution path  $\hat{e}$  (calculated from  $m_0, \dots, m_{i+1}$ ) in different situations. Left: if the optimization steps continuously go in similar directions, the evolution path  $\hat{e}$  is long, a hint to increase the distribution size  $\sigma$  to save iterations. Center: uncorrelated movement directions lead to a medium sized evolution path  $\hat{e}$ , indicating a proper search. Right: oscillating movements cause the evolution path to become small, a strong indication to reduce the distribution size  $\sigma$ .

Note that the rank-one update will only have a large effect if the evolution path continuously points in similar directions. If no clear movement of the mean vector is present, the evolution path will degenerate to  $\mathbf{0}$ , leading to a more widespread distribution dominated by the rank- $\mu$  update.

### Adaptation of the Search Scale

The idea of the evolution path as an indicator on how well the main direction of the search performs is already implicitly used in (5.1.19). But as it does not only indicate the performance of the rank-one update but the entire search, it can be exploited further to speed up the convergence. It can be assumed that a long evolution path, caused by consecutive steps in the same direction, indicates that the current distribution shape is well chosen (Fig. 5.5 left). It also implies that the distribution could be enlarged to cover a larger distance in the parameter space using fewer iterations. On the other hand, a medium length evolution path implies constantly changing (uncorrelated) step directions (Fig. 5.5 center), an indication for a thorough search of the parameter space. A short evolution path, however, is given if the distribution mean is jumping back and forth on the same direction or not moving at all (Fig. 5.5 right), which are both signs hinting at a distribution that might be too coarse.

## 5. Optimization

Following this idea, the distribution scale  $\sigma$  should be increased if the evolution path is long and decreased if the evolution path is short. Whether the path is long or short can be determined by comparing it to the average path length on uncorrelated moving directions, the medium path size indicating a thorough search. Under an  $n$ -dimensional normal distribution, this average, or more precisely, the expected vector length is given by

$$\chi_n = \mathbf{E} \left( \|\mathcal{N}(\mathbf{0}^n, \mathbf{1}^{n \times n})\|_2 \right). \quad (5.1.20)$$

Unfortunately, the evolution path  $\|\mathbf{e}\|_2$  cannot be compared to  $\chi_n$ , as it is correlated by all recent covariances  $\mathbf{C}$  of the past iterations. So, to calculate an uncorrelated path length, a second evolution path  $\hat{\mathbf{e}}$  has to be defined similar to (5.1.17), which only takes the de-correlated movement into account by applying  $\mathbf{C}_i^{-\frac{1}{2}}$  to the current mean movement:

$$\hat{\mathbf{e}}_{i+1} = (1 - c_{\hat{\mathbf{e}}})\hat{\mathbf{e}}_i + \sqrt{c_{\hat{\mathbf{e}}}(2 - c_{\hat{\mathbf{e}}})} \|\mathbf{w}\|_2^{-1} \mathbf{C}_i^{-\frac{1}{2}} \frac{\mathbf{m}_{i+1} - \mathbf{m}_i}{\sigma_i}. \quad (5.1.21)$$

with  $c_{\hat{\mathbf{e}}}$  defining the learning rate of the adaptation similar to  $c_e$ .  $\mathbf{C}_i^{-\frac{1}{2}}$  denotes the inverse of  $\mathbf{C}_i$  with the square root applied to each eigenvalue, as the de-correlation has to be done using the standard deviation, not the variance.

The length of the anisotropic evolution path  $\|\hat{\mathbf{e}}_{i+1}\|_2$  can now be compared to  $\chi_n$ , the expected length under uncorrelated movement. This is done by a multiplication with the scaling factor

$$\sigma_{i+1} = \sigma_i \exp \left( \frac{c_{\hat{\mathbf{e}}}}{q} \left( \frac{\|\hat{\mathbf{e}}_{i+1}\|}{\chi_n} - 1 \right) \right), \quad (5.1.22)$$

using the damping factor  $q \approx 1$ , given by

$$q = 1 + 2 \max \left( 0, \sqrt{\frac{\|\mathbf{w}\|_2^{-2} - 1}{n + 1}} - 1 + c_{\hat{\mathbf{e}}} \right). \quad (5.1.23)$$

Please refer to [Han98] for a motivation on how to chose the damping factor  $q$  and the learning rates  $c_\mu$ ,  $c_1, c_e$ , and  $c_{\hat{\mathbf{e}}}$  or refer to Appendix A for the definition that is used in the experiments of this thesis.

At this point, all components of a working AbS system have been introduced: a 3D-model of the tracked object is either given or generated from the initial input images (Section 3.2.2). Each sensor  $\mathbf{s}$  observing the scene

## 5.2. Accelerating CMA-ES with Parameter Space Knowledge

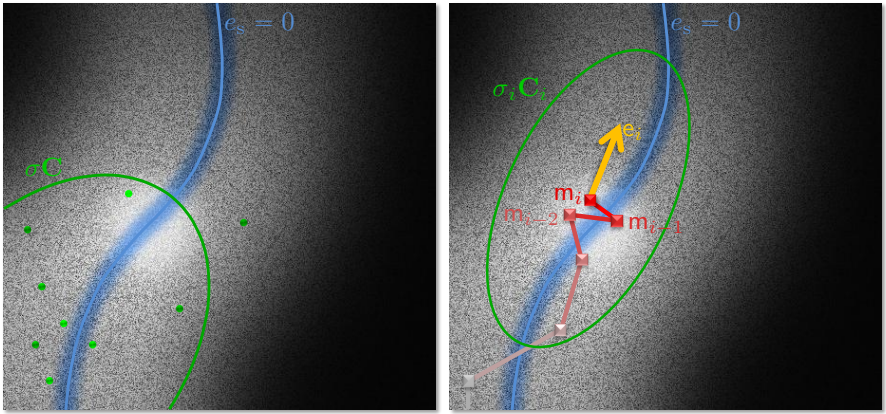
provides an image  $I_{s,\text{in}}^c$  or  $I_{s,\text{in}}^d$  for each frame. A deformation model function  $\mathcal{D}$  is defined (Section 3.2.2), providing a deformed 3D model for each parameter guess  $\Theta$ . Synthesis (Section 4.1) allows to render images  $I_{s,\Theta}^c$  or  $I_{s,\Theta}^d$  for each camera  $s$  and parameter guess  $\Theta$ . The analysis step (Section 4.2) then compares all input images  $I_{s,\text{in}}^c$ ,  $I_{s,\text{in}}^d$  to their synthesized versions  $I_{s,\Theta}^c$ ,  $I_{s,\Theta}^d$ , summarizing (Section 4.2.4) the comparison into one scalar fitness value in  $\mathcal{F}$ , together with constraints implementing prior knowledge (Section 4.3). The resulting fitness function  $\mathcal{F}$  can then be optimized for every frame using CMA-ES (Section 5.1), yielding the optimal deformation parameter  $\Theta^*$ .

## 5.2 Accelerating CMA-ES with Parameter Space Knowledge

Taking a closer look at the deformation model that was used up until now, it is worth mentioning that it is not only defined by the deformation function  $\mathcal{D}$ , but also by the additional constraints in the parameter space. For the deformation at hand this is the stretch constraint. Without this additional constraint, the control points could be arbitrarily chosen in 3D space, not representing the deformation that one wishes to model. Hence, the deformation model consists of an explicit part ( $\mathcal{D}$ ) and an implicit part ( $e_s = 0$ , see (4.2.10)), defining a subspace (or manifold) in which the solution should be found. Although this implies a larger number of problem dimensions than necessary (the problem could have been formulated completely explicitly within the constraint manifold), it allows an easier problem description and prevents the search space from being too complex and cluttered for efficient optimization. Having additional dimensions for a more convenient optimization process is a very common tool in applied mathematics and computer science and can be found in many methods (e. g., the kernel trick of support vector machines [BGV92]).

Usually this problem class (optimization in a sub-manifold) is tackled using techniques like Lagrange multipliers [Ber96] that allow to mitigate the effects of a higher dimension count. Unfortunately, these methods cannot be applied efficiently due to the missing derivatives of the fitness  $\mathcal{F}$  and the noisy character of the function. And although CMA-ES scales very well with the number of dimensions, optimizing it for the current deformation model can still speed up the optimization process significantly. In the following Sections 5.2.1 and 5.2.2, a system is introduced to retrieve a local approximation of the

## 5. Optimization



**Figure 5.6.** Example visualization of fitness function  $\mathcal{F}$  (dark= low fitness, bright= high fitness). The blue line depicts the sub-manifold in which the stretch constraint is minimal. Left: in early iterations the distribution mean is not close to the optimum and the distribution size  $\sigma$  is large. Right: when closing in on the optimum, the mean movements become smaller and/or anticorrelated (red lines) and the distribution size  $\sigma$  starts shrinking. The evolution path  $\mathbf{e}$  shrinks as well but still contains the direction of the most recent, large steps.

constraints and to apply this information increasing the convergence speed of CMA-ES.

### 5.2.1 Local Approximation of Constraints

In practice, the optimization for solving a tracking problem like the one at hand takes place in two stages: at the initial stage, CMA-ES will start with a large distribution size to move its mean roughly in the direction of the new minimum, picking up momentum, leading to evolution paths ( $\mathbf{e}$  and  $\hat{\mathbf{e}}$ ) with long or medium length while maintaining a large step size  $\sigma$  (see Fig. 5.6 left). The second stage is dominated by a decreasing step size  $\sigma$  as the solution is roughly reached but still has to be refined. At this stage, the evolution paths are shrinking and the search distribution narrows down around the final optimum (see Fig. 5.6 right).

At this second stage, around the time in which the length of the anisotropic evolution path  $c_{\hat{\mathbf{e}}}$  drops below the expected length of a normal distribution  $\chi_n$ , the uncorrelated search pattern becomes anticorrelated and the step size

## 5.2. Accelerating CMA-ES with Parameter Space Knowledge

$\sigma$  starts to decrease. There are three properties of the CMA-ES state that allow exploitation.

- ▷ The movement of the mean is small in relation to the distribution size (due to the regularization in evolution path and the step size) and the current estimate is (most probably) fairly close to the final result, as are the means in between.
- ▷ The anisotropic evolution path becomes smaller, reducing the distribution size, causing the rank- $\mu$  updates to contain more local information, and in conjunction with the slow moving mean, this leads to a very thoroughly probed covariance for the current region.
- ▷ The isotropic evolution path becomes smaller, but due to the uncorrelated movement of the mean vector, still contains the main direction of the current optimization step, continuously adding it to the covariance via the rank-one update.

Let  $i^*$  denote the iteration that is characterized by these aspects. At  $i^*$ , the covariance matrix contains two main influences: The rank- $\mu$  update, as an approximation of the inverse hessian<sup>5</sup> at this location, and the rank-one update which is mainly influenced by the direction of the result from the previous frame and the current frame.

### 5.2.2 Optimized Initial Covariance Adaptation

It is possible to utilize the CMA-ES state at iteration  $i^*$  for each succeeding frame in an image sequence, if two assumptions are met:

- ▷ The manifold constraining the search space has to be a dominant part of the fitness function so that it can influence  $\mathbf{C}$  via the rank- $\mu$  update significantly,
- ▷ and the linear approximation of the manifold contained in the covariance  $\mathbf{C}_{i^*}$  should be reasonably well for the distance in the parameter space that is covered from frame to frame.

---

<sup>5</sup>One can interpret the rank-one update as a natural gradient descent, with the natural gradient defined as the inverse Fisher matrix multiplied with the gradient and the Jacobian matrix. The Fisher matrix is often the negative Hessian of the Shannon entropy, thus allowing  $\mathbf{C}$  to be interpreted as an approximation of the inverse hessian of  $\mathcal{F}$ .

## 5. Optimization

The influence of the rank-one update on the covariance, dominated by the parameter movement from the recent frame to the current frame, can be used to bootstrap the initialization of the succeeding frame. In the remainder of this Section, let the high index  $f$  specify the frame of the CMA-ES optimization, i. e., the covariance at CMA-ES-iteration  $i$  of frame  $f$  is denoted by  $\mathbf{C}_i^f$ .

It is obviously advantageous to initialize the distribution of a frame around the optimum of the preceding frame:

$$\mathbf{m}_0^{f+1} = \Theta^{*f}. \quad (5.2.1)$$

As an initial distribution size, the overall step size of the preceding frame serves as an appropriate prediction. But as this can easily be 0, e. g., if the first frames of a sequence contain no movement, a minimum distribution size has to be defined. Let  $\theta$  be this lower bound, than  $\sigma$  is initialized by

$$\sigma_0^{f+1} = \max\{\theta, \|\mathbf{m}_0^f - \Theta^{*f}\|_2\}. \quad (5.2.2)$$

As  $\mathbf{C}_{i*}^f$  provides a good initial guess on a successful search distribution for frame  $f + 1$ , it is used for the initialization of  $\mathbf{C}_0^{f+1}$ . But, again, to prevent this matrix from degenerating, it is important to define a minimum value  $\epsilon$  for the eigenvalues of the covariance. If no minimal eigenvalue is enforced, a linear subspace can easily be excluded completely from all optimizations because it is not regarded by the sample distribution and, thus, has no way to recover itself.

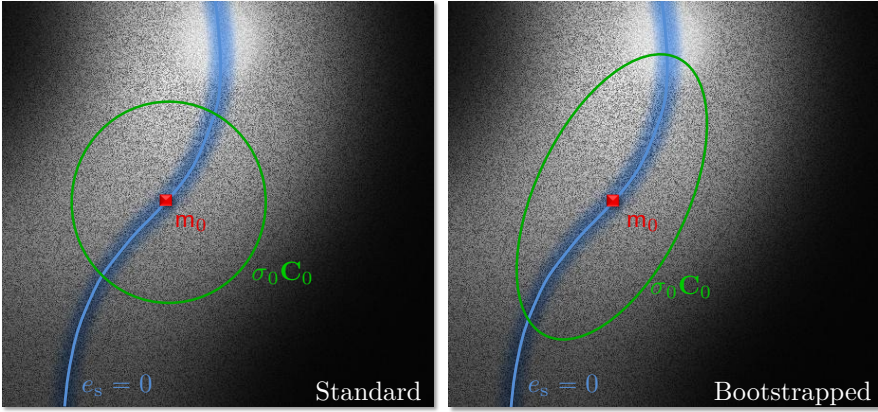
The covariance matrix  $\mathbf{C}$  is normalized, so  $\epsilon$  has to be chosen according to the problem dimension  $n$ , not the actual search distribution size. In the practical evaluation (see chapter 7), a maximum ratio of 20 between the largest and the smallest eigenvalue of  $\mathbf{C}$  yielded good results throughout all experiments, i. e.,  $\epsilon = \frac{1}{20}$ . To enforce this constraint, the reference distribution matrix  $\mathbf{C}_{i*}^f$  has to be decomposed so the individual eigenvalues can be accessed. As  $\mathbf{C}$  is always real, symmetric, and semi-definite, a Singular Value Decomposition (SVD) yields

$$\mathbf{C}_{i*}^f = U \text{diag}(\xi) U^T, \quad (5.2.3)$$

with  $U \in \mathbb{R}^{n \times n}$  and  $\xi \in \mathbb{R}$  containing the eigenvalues of  $\mathbf{C}$ . Note that (5.2.3) does not have to be calculated separately for the bootstrap, but is already present as it is required for distributing the samples according to  $\mathcal{N}_{(\mathbf{m}, \tilde{\mathbf{C}})}(\Theta)$ , so no additional computational work is required. The maximum ratio can now



## 5.2. Accelerating CMA-ES with Parameter Space Knowledge



**Figure 5.7.** Example visualization of fitness function  $\mathcal{F}$  (dark= low fitness, bright= high fitness). The blue line depicts the sub-manifold in which the stretch constraint is minimal. Left: standard initialization of CMA-ES with  $\mathbf{C}$  being a diagonal matrix. Right: bootstrapping  $\mathbf{C}$  with the knowledge about the constraints and mean movements leads to a superior initial distribution.

be enforced using  $\xi' \in \mathbb{R}^n$  with

$$\xi'_m = \max\{\epsilon \max_{k \leq n} \{\xi_k\}, \xi_m\} \forall m \leq n, \quad (5.2.4)$$

in its normalized form  $\xi'_m \|\xi'_m\|_2^{-1}$  to compose  $\mathbf{C}_0^{f+1}$ :

$$\mathbf{C}_0^{f+1} = U \text{diag}(\xi'_m \|\xi'_m\|_2^{-1}) U^T. \quad (5.2.5)$$

To maintain a consistent CMA-ES state, the two evolution paths are initialized according to their state at iteration  $i^*$ ,

$$\mathbf{e}_0^{f+1} = \mathbf{e}_{i^*}^f \quad (5.2.6)$$

and

$$\hat{\mathbf{e}}_0^{f+1} = \hat{\mathbf{e}}_{i^*}^f. \quad (5.2.7)$$

Figure 5.7 visualizes the effect of a bootstrapped initial distribution (see also Figure 5.6).

## 5. Optimization

### **Evaluation Preview**

Initializing the covariance matrix by prior knowledge as discussed in this Section leads to a significant improvement of the convergence speed. Section 7.3 contains results of multiple tests on real data showing a faster optimization in the early optimization stage (90% better fitness value after 15 iterations, 50% better fitness values after 30 iterations), but also a slight improvement of the general optimization performance.

# Weighting Strategies

Combining the fitness values of multiple, different sensors and constraints to one overall fitness is usually done by creating a weighted sum of all component errors (see Section 4.2.4). But determining these weights is not yet well defined. For many situations, it is appropriate to manually set these parameters based on experiments, as the algorithm is not very sensible to inadequate weight values. However, there are ways to automatically determine, adjust or deduce these weights. In the following, two strategies are introduced and evaluated.

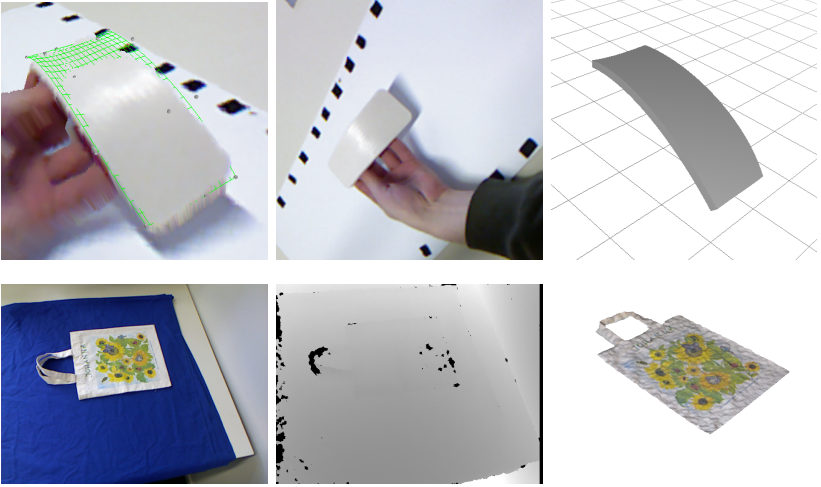
## 6.1 Heuristic Adjustment

One of the standard setups this thesis discusses is the combination of one or multiple depth cameras with one or multiple color cameras. The fitness function  $\mathcal{F}$  of such a setup thus consists of three components: constraints, depth fitness, and color fitness. As the constraints do not follow any specific scale but, in general become zero if their condition is met, one can simply eliminate the constraint weight by reformulating the fitness function to

$$\mathcal{F}(\Theta) = -(e_s(\Theta) + 1) \left( \sum_{s \in S_d} \varphi_s e_s(\Theta) + \sum_{s \in S_c} \varphi_s e_s(\Theta) + 1 \right), \quad (6.1.1)$$

where  $S_d$  denotes the set of depth sensors in  $S$  and  $S_c$  denotes the color sensors. Since depth errors are all calculated from the same domain (as well as color errors), the weighting problem can be reduced to finding the balance

## 6. Weighting Strategies



**Figure 6.1.** Two examples of different entities providing the required data for the tracking: Top: the white, reflecting object on a white background can only be tracked properly because of the depth information, while the color information is rather useless. Bottom: a textured object lying flat on the ground is not well traceable in the depth input data. The color data provides the required information for tracking.

between color and depth errors:

$$\mathcal{F}(\Theta) = -(e_s(\Theta) + 1) \left( \varphi_d \sum_{s \in S_d} e_s(\Theta) + \varphi_c \sum_{s \in S_c} e_s(\Theta) + 1 \right). \quad (6.1.2)$$

And as the system is only interested in the minimal argument of the fitness function, not the actual fitness itself, the absolute value can be scaled arbitrarily without effecting the optimization process and its result. Let  $\varphi_{cd} = \frac{\varphi_c}{\varphi_d}$ , then the same argument minimum for  $\Theta$  can be achieved by dividing the complete definition by  $\varphi_d$ :

$$\mathcal{F}(\Theta) = -(e_s(\Theta) + 1) \left( \sum_{s \in S_d} e_s(\Theta) + \varphi_{cd} \sum_{s \in S_c} e_s(\Theta) + 1 \right). \quad (6.1.3)$$

This reduces the weighting problem to the task of finding an appropriate weight balance  $\varphi_{cd}$

$e_c$  and  $e_d$  both have a linear characteristic with respect to the number of “falsely matched” pixels and both functions yield 0 for a perfect fit and perfect input data. This means that for the perfect deformation  $\Theta^*$ , the error caused by noise in the depth and color image is  $e_d(\Theta^*)$  and  $e_c(\Theta^*)$ . Following the principle of calculating a maximum likelihood based on variances calculated by these noise ratios, the resulting weight  $\varphi_c$  should be equal to  $\frac{\sum e_d(\Theta^*)}{\sum e_c(\Theta^*)}$ .

Although this definition can be statistically deduced, it is based on the assumption of  $\Theta^*$  being the perfect match, i. e., for every frame it is assumed that the preceding frames were matched correctly. So in a situation as depicted in Fig. 6.1 (bottom row), in which the color information is valuable whereas the depth information is rather useless,  $e_c(\Theta^*)$  is likely to yield increased error values, even for a good solution  $\Theta^*$ , and  $e_d$  remains constantly low. This would cause  $\varphi_{cd}$  to decrease, leading to a fit much less influenced by the color, causing  $\varphi_{cd}$  to decrease further. A similar example for an increasing  $\varphi_{cd}$  can be found in Fig. 6.1 (top row), depicting the tracking of a white object on white background.

To counter this behavior, a second aspect is considered in addition to the noise levels: the relative information scale of the error values. For a finite set of vicinity parameters  $\Omega \subset \mathbb{R}^n$  distributed around  $\Theta^*$ , let

$$H_d(\Omega) = \sum_{\Theta \in \Omega} \frac{e_d(\Theta)}{|\Omega|e_d(\Theta^*)} \quad (6.1.4)$$

and

$$H_c(\Omega) = \sum_{\Theta \in \Omega} \frac{e_c(\Theta)}{|\Omega|e_c(\Theta^*)}. \quad (6.1.5)$$

$H_c$  and  $H_d$  provide a measure for the scale of the error values by determining the currently best fit in respect to their “noise levels”, by calculating the average ratio in fitness between the average parameter in the vicinity and the optimum. Hence, the ratio  $\frac{H_c}{H_d}$  is a measure on how valuable the color error is in relation to the depth error. As the aim is to equally use color and depth information,  $\varphi_{cd}$  should be chosen such that

$$H_d = \varphi_{cd}H_c. \quad (6.1.6)$$

The resulting  $H_c$  and  $H_d$  are a measure on the scale of the error values used to determine the currently best fit with respect to their “noise levels”. Consequently, the ratio  $\varphi_{cd} = \frac{H_c}{H_d}$  is a measure on the scale of the color

## 6. Weighting Strategies

error in relation to the depth error. And after every frame of a sequence, the color-depth-weighting ratio can be updated to average over the ratios satisfying (6.1.6) of all frames already processed. This can be notated as:

$$\lambda_c \rightarrow \frac{n-1}{n} \lambda_c + \frac{1}{n} \frac{H_c}{H_d}, \quad (6.1.7)$$

for  $n$  being the number of the current frame. This weighting technique has been applied in the experiments depicted in Figure 7.3, 7.5, 7.6, 8.1, and 8.3.

## 6.2 Probabilistic Problem Formulations

The heuristic adjustment of weights, although working well, is limited to balancing between different entities. In this Section, an alternative method of determining weights is proposed (first introduced in [JK13]) that allows to weight every measurement, i. e., every sensor value and pixel input separately. The benefit of pixel-wise weighting is demonstrated on a ToF sensor, which is generally known to have varying noise characteristics throughout the pixels.

In the area of sensor fusion, statistical modeling is common because it allows to define a global likelihood based on the expected distribution of each sensor input. This means if a noise model is present for every input data, the global likelihood over an arbitrarily large set of different inputs from different domains is still well defined. A likelihood-based AbS method is able to combine the input of different types of depth cameras (*Kinect* and ToF) along with color information and even binary information like light-barrier feedback, regarding sensor specific shortcomings.

But modeling the noise of a sensor is not straight forward. Especially for more complex capturing systems like time-of-flight cameras, the problem of finding a suitable model is tedious and not sufficiently solved yet. The problem is that errors may depend on various environmental influences, correlated background noise within and from outside the camera housing, overexposure [LNL<sup>+</sup>13], and geometrical constellations that introduce multi-path effects [JPMP14] as well as material properties [GAL07]. Some of these effects can be modeled well, although a direct compensation is not always possible or only with a reduction of image information. The very advantage of AbS methods is that there is no need for any explicit compensation as long as the noise model can be given, i. e., the PDF can be calculated.

Let  $\mathbf{P} : \Omega \rightarrow [0, 1]$  be the PDF for a distance measurement event  $x_{d_m} \in \Omega$  given the real distance  $x_{d_r}$ , i. e.,  $\mathbf{P}(x_{d_m} | x_{d_r})$ , providing the relative likelihood

## 6.2. Probabilistic Problem Formulations

of  $x_{d_r}$  when the measurement event  $x_{d_m}$  is present, which can be notated as  $\mathcal{L}(x_{d_r}|x_{d_m})$ .

There has been the common observation, that the amplitude of the ToF signal can provide a hint on the accuracy of a measurement (see [LNL<sup>+</sup>13, SBH11, RDP<sup>+</sup>11]). Let this be regarded by the likelihood function as well. Given the additional amplitude measurement event  $x_a$ , the likelihood  $\mathcal{L}(x_{d_r}|x_{d_m}, x_a)$  is augmented by the additional conditional  $x_a$ , regarding the amplitude measurement. The slight inaccuracy of regarding the amplitude measurement instead of the actual amplitude helps to create a system free of hidden random variables, since the amplitude is not rendered by the synthesis step. It can be assumed that the effect of exchanging the actual amplitude by its measurement is neglectable. For a color sensor, the likelihood can be denoted in a similar fashion, by calculating the probability of the color of a vertex projected onto sensor given the measured color at this point in the image.

The overall goal can now be formulated as finding the deformation parameters  $\Theta^*$  causing the solution with the highest likelihood over all measurements over all sensors

$$\Theta^* = \arg \max_{\Theta} \sum_{v \in V} \sum_{s \in S} \log(\mathcal{L}_{v,s}) \quad (6.2.1)$$

where  $\mathcal{L}_{v,s}$  denotes the likelihood function for the pixel onto which  $\mathcal{D}(v, \Theta)$  would be projected in sensor  $s$ , evaluated using the given synthesized distance ( $\|\mathcal{D}(v) - c\|_2$ ) (or vertex color) and the measured distance and amplitude (or color) of the pixel at image coordinate  $\mathcal{P}(\mathcal{D}(v, \Theta))$ .

In order to test the explicit noise handling of the likelihood-based system, a set of synthetic and real tests have been performed. The goal of these tests is not to evaluate a certain noise model, but to demonstrate that explicit noise handling does work in AbS methods and that it can greatly improve the fitting results. For this reason, a rather simple Gaussian-distribution-based approach has been implemented for the test cases, which can be replaced by any, more complex model based, e. g., on [RDP<sup>+</sup>11, Fal07].

### 6.2.1 Polynomial Interpolated Gaussian Distribution

The implemented distribution, approximates the ToF measurement noise by a Gaussian distribution per pixel with respect to depth and amplitude image. Mean value and deviation are calculated based on distance and amplitude measurements. Let  $\mu_p$  be the function to describe the average offset of the measurement for a pixel  $p$ , and let  $\sigma_p$  describe the average deviation of a measurement. Then the PDF of the approximating Gaussian distribution is

## 6. Weighting Strategies

given by

$$\phi(x)_p = \frac{1}{\sqrt{2\pi\sigma_p^2}} e^{-\frac{(x-\mu_p)^2}{2\sigma_p^2}}. \quad (6.2.2)$$

for one measurement of pixel  $p$ .

The precision and the noise levels of ToF measurements depend on a rather large number of variables, including the integration time, the modulation frequency, sensor and illumination properties, the observed material and the surface normal, the scene geometry, and the distance of each measurement. For a ToF camera with a fixed integration time setting and sensor properties, the most influential variable aspects remain the amount of reflected light and the distance to the measurements target. Both aspects can be estimated by the actual measurement, as the distance measurement of each pixel obviously correlates with the actual camera-object-distance and the amplitude image of the ToF measurement provides a hint on the amount of light reflected by the object for the measurement at hand subject to the distance.

To account for the influence of distance and amplitude on the measurement, fifteen static ToF camera sequences have been acquired using five different distances to a large reference object and three different object materials each, to change the reflectivity of the measured surface. The measurements have been performed using a *Mesa Imaging SR4000*<sup>1</sup>. The ground truth for all measurements is known. For each pixel and each combination of amplitude and distance, the mean measurement and the deviation of the measurements (over time) is available. Figure 6.2 depicts the measurements for a central pixel of the ToF camera to demonstrate the relation between distance, amplitude and noise. Note that the amplitude values of the *SR4000* are preprocessed with a disclosed factor, containing  $\frac{1}{d^2}$  with  $d$  being the distance measurement.

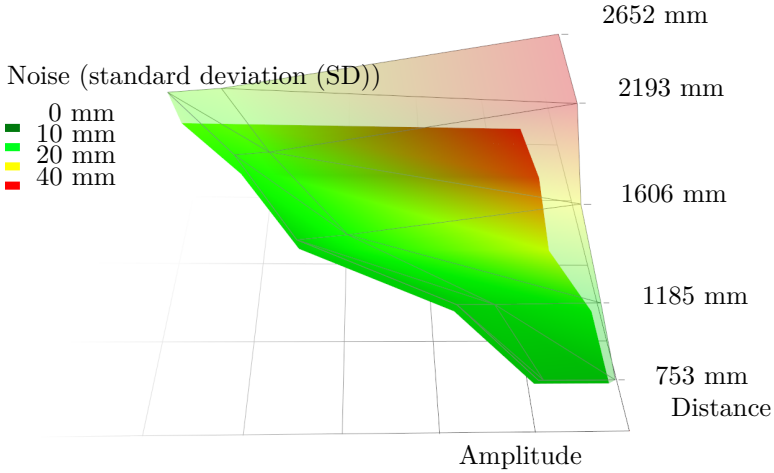
In order to apply the measured distribution to the analysis step, the likelihood function of each measurement is rendered based on the deviation  $\sigma_p$  and the mean  $m_{(u,v)}$ , which interpolate the measured mean and deviation values for the synthesized depth measurement and the amplitude measurement of the pixel  $(u, v)$ . Hence,  $\sigma_{(u,v)}$  and  $m_{(u,v)}$  become functions depending on the actual distance  $d$  and the measured amplitude  $a$ , so (6.2.2) becomes

$$\phi_{(u,v),a}(d) = \frac{1}{\sqrt{2\pi\sigma_{(u,v)}(d,a)^2}} \exp\left(-\frac{1(d - m_{(u,v)}(d,a))^2}{2\sigma_{(u,v)}(d,a)^2}\right). \quad (6.2.3)$$

---

<sup>1</sup>For technical details on the *SR4000* visit <http://www.mesa-imaging.ch>





**Figure 6.2.** Plot of the average noise (standard deviation in mm) as it was measured by a central pixel for various distances (ground truth) and amplitude values as returned by an *SR4000* (distance compensated). The height is depicting the noise levels. For visualization purposes, the measurements have been connected to a 3D surface.

The functions  $m_{(u,v)}$  and  $\sigma_{(u,v)}$  interpolate the actual mean and deviation values of the measurements made by pixel  $(u, v)$  with available ground truth based on the synthesized distance  $d$  and the amplitude  $a$ , i. e.,  $m_{(u,v)}$  and  $\sigma_{(u,v)}$  are polynomial functions interpolating the discrete measurements retrieved by the real experiments with available ground truth. In a way, this error modeling follows the spirit of the reflectivity calibration by Lindner et al. [LSKK10]. If  $(u, v)$  is written as a function providing the actual pixel to an image coordinate, then the optimization goal is to find a  $\Theta^*$  that maximizes the likelihood over all pixels given by

$$\Theta^* = \arg \max_{\Theta} \sum_{v \in V} \log(\phi_{(u,v), I^{\text{amp}}(u,v)}(\|\mathcal{D}(v, \Theta) - c\|_2 - I^d(u, v))) \quad (6.2.4)$$

## 6. Weighting Strategies

with

$$(u, v) = \lfloor \mathcal{P}(\mathcal{D}(\mathbf{v}, \Theta)) + \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \rfloor \quad (6.2.5)$$

and  $\mathbf{c}$  denoting the camera center,  $I^{\text{d}}$  and  $I^{\text{amp}}$  describing the actual depth and the amplitude image. To examine the components of (6.2.4), it is helpful to recapitulate the overall tracking process (see also Fig. 4.1):

- ▷ The goal is to search for the model parameters  $\Theta^*$  fitting the depth input image  $I^{\text{d}}$  with respect to the amplitude input image  $I^{\text{amp}}$ .
- ▷ To optimize the parameters, CMA-ES generates a guess  $\Theta$  that is evaluated.
- ▷  $\Theta$  is evaluated by a deformation function  $\mathcal{D}(\cdot, \Theta)$ , which allows to generate the set of deformed vertices  $\mathcal{D}(V, \Theta)$ .
- ▷ Each deformed vertex  $\mathcal{D}(\mathbf{v}, \Theta)$  is projected at the image coordinate given by  $\mathcal{P}(\mathcal{D}(\mathbf{v}, \Theta))$  and visible in pixel  $(u, v)$  (see (6.2.5)).
- ▷ If  $\Theta$  is a correct guess, then the distance measurement  $I^{\text{d}}(\mathcal{P}(\mathcal{D}(\mathbf{v}, \Theta)))$  of the pixel at image coordinate  $\mathcal{P}(\mathcal{D}(\mathbf{v}, \Theta))$  would correspond to the distribution of pixel  $(u, v)$  given a real distance  $\|\mathcal{D}(\mathbf{v}, \Theta) - \mathbf{c}\|_2$  and a current amplitude of that pixel  $I^{\text{amp}}(u, v)$ .
- ▷ This distribution is described by  $m_{(u,v)}(I^{\text{amp}}(u, v))$  and  $\sigma_{(u,v)}(I^{\text{amp}}(u, v))$ , the mean and deviation offset retrieved by interpolating the sample measurements with available ground truth.
- ▷ Summing up the log-Likelihood over all vertices  $\mathbf{v}$  yields how likely it is that  $\Theta$  represents the deformation visible in the input images  $I^{\text{d}}$  and  $I^{\text{amp}}$ .
- ▷ This value is returned to CMA-ES as evaluation of  $\Theta$  (equation 6.2.4).

Equation 6.2.4 can easily be extended to multiple depth and color sensors, by summing up all likelihood functions as suggested in (6.2.1). Let  $S$  be a set of sensors  $\mathbf{s}$  and let  $I_{\mathbf{s}}^{\text{d}}$  and  $I_{\mathbf{s}}^{\text{amp}}$  be the depth and amplitude image of a sensor  $\mathbf{s}$ , then

$$\Theta^* = \arg \max_{\Theta} \sum_{\mathbf{s} \in S} \sum_{\mathbf{v} \in V} \log(\phi_{(u,v), I_{\mathbf{s}}^{\text{amp}}(u,v)}(\|\mathcal{D}(\mathbf{v}, \Theta) - \mathbf{c}_{\mathbf{s}}\|_2 - I_{\mathbf{s}}^{\text{d}}(u, v))) \quad (6.2.6)$$

with

$$(u, v) = \lfloor \mathcal{P}_{\mathbf{s}}(\mathcal{D}(\mathbf{v}, \Theta)) + \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \rfloor \quad (6.2.7)$$

## 6.2. Probabilistic Problem Formulations

providing the likelihood of a solution  $\Theta$  regarding all sensors, with  $\mathbf{c}_s$  denoting the camera center of each sensor and  $\mathcal{P}_s$  denoting the corresponding projection. For color cameras, the likelihood can be computed analogously using the vertex color instead of the camera-to-vertex distance.

### **Evaluation Preview**

The evaluation results discussed in Section 7.5 show that a probabilistic formulation of the tracking problem allows to compensate for strong sensor noise, if a viable noise model is at hand. For ToF cameras, such a model can also be learned to a certain extend.

At this point, the complete AbS chain has been introduced. The next chapter will show evaluation results on synthetic and real data experiments, testing it for accuracy, stability, and efficiency.



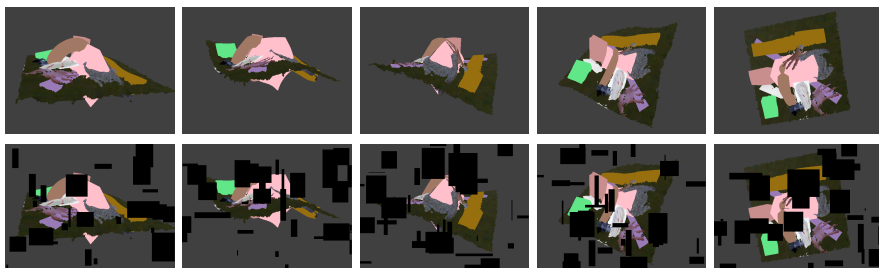
# Evaluation

A number of experiments have been performed using synthetic (7.1) and real data (7.2 - 7.3.1) to evaluate the various aspects of the AbS tracking algorithm. In Section 7.1 the tracking of a synthetically deformed, simple object under optimal conditions was investigated. Section 7.1.1 shows a similar test with a far more complex object. In Section 7.1.2, some of the input data was blackened to test the method on incomplete data.

In addition, the performance on real data has been evaluated, testing the algorithm on input data, motion blur, incomplete data, and large changes in object distance (7.2). Section 7.2.1 shows the tracking of a more complex geometry and in Section 7.3 the performance gain from the CMA-ES initialization scheme (5.2) is evaluated. Section 7.3 shows the results of a real-time test at 25 Hz frame rate. Section 7.3.2 discusses the scalability and the limitations of the NURBS-based deformation function.

In the following, the number of NURBS control points, the number of object vertices, the object size and the average distance between the object and the camera will be provided for each test. In addition, the number of optimization cycles (CMA-ES iterations) and the number of fitness function evaluations per iteration (CMA-ES individuals) will be given. Note that the search space is three times the number of control points, e. g., for  $7 \times 7$  control points the search space has 147 dimensions. In Section 7.3, empirical data to estimate the runtime / frame rate based on vertex count, iterations, and individual count will be given. Section 7.4 will analyze the impact of random online sparsification on different tracking targets using constant and adapting sparsification rates.

## 7. Evaluation



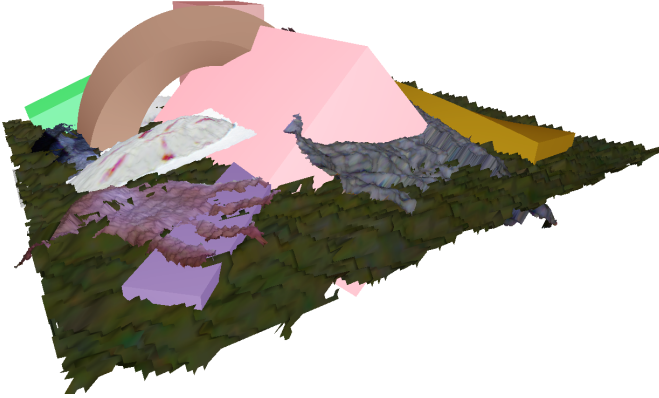
**Figure 7.1.** A synthetically generated, complex geometry, deformed and rendered as depth and color image sequence. The first line depicts the rendered color data, the second line shows the synthetic errors (invalid values) added to the input data to evaluate the stability of the algorithm on incomplete data.

### 7.1 Experiments on Synthetic Data

The absolute accuracy under massive deformation has been tested using a rendered scene with available ground truth. A scene of a deforming bag (size  $42 \times 26$  cm, distance to camera 70 cm) has been generated by applying a deformation sequence of a bag (see Fig. 7.6) to a 3D model of a bag (consisting of 117,000 vertices). This way, ground truth data about every object position in every frame is available for a deformation sequence. The deforming 3D model was rendered using the projection data from the *Kinect* calibration. The rendered depth and color video streams were then used as input data for a tracking sequence using the same object mesh. This test allows to compare every vertex position produced by the tracking algorithm with the vertex positions used to render the input sequence. For the complete sequence, the average RMS distance between the input vertices and the tracked vertices was 2.89 mm, using a NURBS function with  $7 \times 7$  control points, 350 CMA-ES iterations and 96 CMA-ES individuals.

#### 7.1.1 Experiments on Complex Synthetic Geometry

To test the algorithm on a more complex geometry, an artificial object (Fig. 7.2) was created from various boxes, rings, random noise surfaces, and scanned object meshes, a mesh made of 113,000 vertices and with a size of  $60 \times 60 \times 40$  cm. A 500 frame sequence was generated in which this object was rotated, translated, and deformed. Figure 7.1 shows a selection of renderings



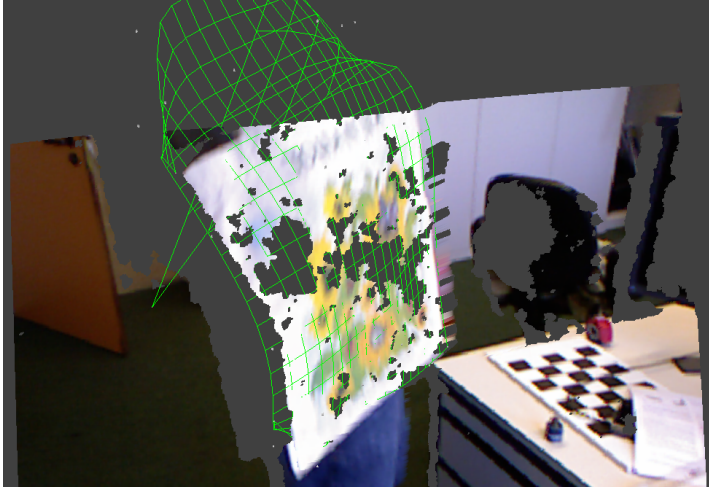
**Figure 7.2.** An artificial object model with a complex geometry made from various boxes, rings, noisy surfaces, and scanned objects.

of this sequence. The sequence was tracked using a NURBS function with  $5 \times 5$  control points, 219 CMA-ES iterations (in average), and 56 CMA-ES individuals. For the complete sequence, the RMS distance between the input vertices and the tracked vertices was 4.49 mm. The red line in Figure 7.4 shows the RMS error for each frame of the sequence. From frame 300 to 354, massive self occlusion is causing the average distance over all frames to increase. Ignoring these frames, the average distance is 4.08 mm.

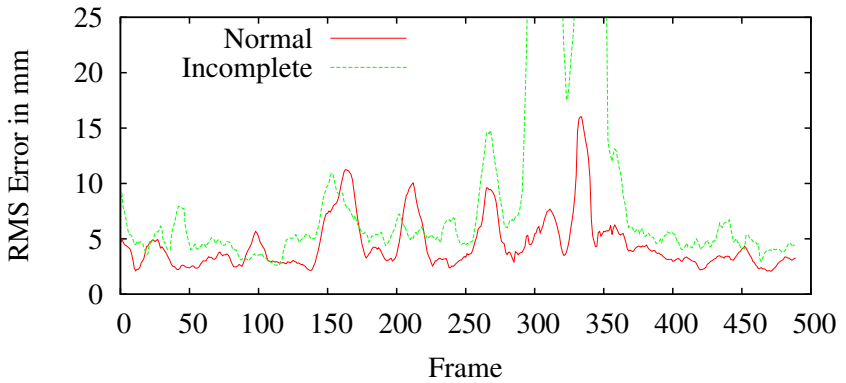
### 7.1.2 Experiments on Incomplete Synthetic Data

Since depth sensors like the *Kinect* camera often provide incomplete depth data (see e. g., Fig. 7.3), the algorithm was tested using the same input data as described in Section 7.1.1 but with random black boxes added, covering twenty percent of the input data in average. The second line of Figure 7.1 displays the blackened areas in the input data. The RMS distance between the input vertices and the tracked vertices was 12.93 mm. Ignoring the 59 frames of massive self occlusion after frame 295, the RMS distance would be 5.71 mm. The green line in Figure 7.4 shows the RMS error for each frame of the sequence.

## 7. Evaluation

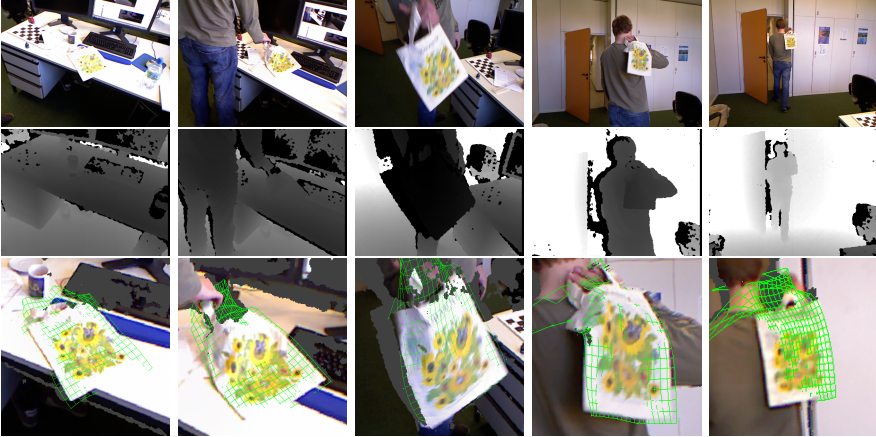


**Figure 7.3.** A 3D mesh reconstructed from a *Kinect* color and depth image. The close distance to the *Kinect* causes holes in the depth information.



**Figure 7.4.** The RMS distance between the original deformed object vertices and the tracked object vertices for the 500 frame input sequence.





**Figure 7.5.** Pictures of a color and depth video sequence of a bag recorded with a *Kinect* camera. Top row: the color image output; center row: the depth image output; bottom row: a closeup of the resulting colored 3D mesh together with the NURBS surface (green) following the deformation of the bag.

## 7.2 Experiments on Real Data

The algorithm has been tested on different types of real world scenes to validate the convergence of the tracking algorithm.

Figure 7.5 shows pictures of a sequence recorded with a *Kinect* camera in which a bag is dragged along a table and then thrown over the shoulder. The input data is challenging in a number of ways: the camera is moving and the first two color images (top row, image one and two from the left) show severe overexposure due to the auto-gain of the *Kinect* camera. Column three and four in Figure 7.5 show motion blur that occurs at fast movements of the camera and/or the object. Column five displays the effect of very low color and depth resolution when objects move far away from the camera. Figure 7.3 depicts a 3D mesh reconstruction of the same sequence, showing the effect of errors in the depth image (holes in the mesh), caused by the object being too close to the *Kinect* camera. In addition to that, the rotation movement of the camera is too slow to keep the fast moving bag completely inside the image, so parts of the tracked object leave the depth and color images.

As the closeup of the rendered NURBS surfaces in the 3D mesh reconstruction of the sequence show (Fig. 7.5 bottom row, Fig. 7.3), the NURBS surface

## 7. Evaluation



**Figure 7.6.** Pictures of a deforming 3D mesh. The underlying deformation sequence was used to generate an artificial input data with known ground truth. The depicted sequence was acquired by the the algorithm discussed in Section 8.2.

remains precisely on the object throughout the video sequence, following every deformation and transformation in the input data. The results have been produced offline using a  $7 \times 7$  NURBS function, 350 CMA-ES iterations and 96 CMA-ES individuals.

### 7.2.1 Real Data Experiments with Complex Object

A test on a geometrically more complex object is depicted in Figure 7.7. A ( $24 \times 29$  cm) teddy bear was deformed and recorded with a *Kinect* camera at an average distance of 68 cm. The first depth and color images of the sequence were used to generate the 3D model (33500 vertices) for the tracking algorithm. A  $6 \times 6$  control point NURBS surface was used as deformation function and 750 CMA-ES iterations using 120 CMA-ES individuals produced an average RMS depth error for the complete sequence of 2.59 mm.

## 7.3 Relative Performance Experiments

This Section will discuss the relative gain of speed using the covariance propagation introduced in Section 5.2. To measure the increased performance



**Figure 7.7.** Image sequence of a deformed teddy bear. The upper row shows the *Kinect* color image, the bottom row displays the deformed mesh that was retrieved from the first depth/color image pair of the sequence.

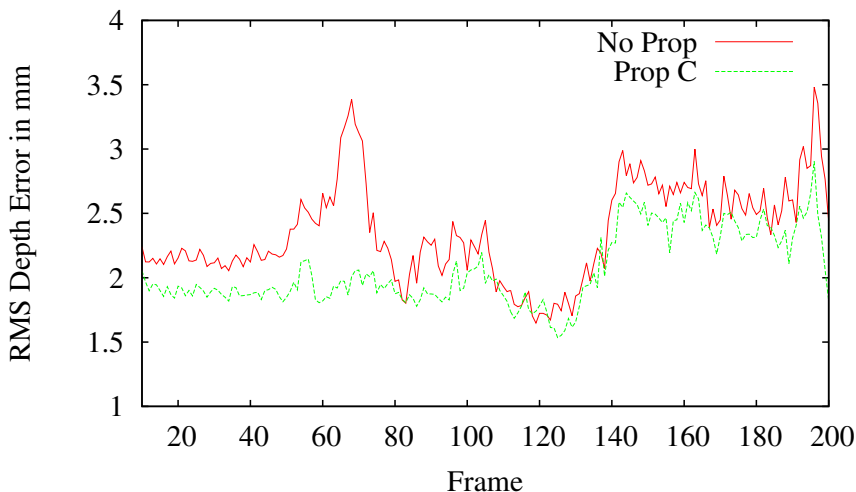
gained by the covariance propagation, the original CMA-ES method using the standard initialization is compared to the one introduced in Section 5.2 on the exact same input data. Fig. 7.11 depicts this input sequence.

The time one frame takes to be processed is approximately linearly dependent on the number of CMA-ES iterations performed, so reducing the number of iterations to one third will yield a frame rate three times as high. The goal is to show that for low iteration numbers, the new initialization scheme will lead to a frame rate three times as high compared to the standard initialization while providing equivalent results.

The sequence depicted in Fig. 7.11 is given as the tracking input. The tracker using the standard initialization performs an optimization on every third frame (simulating 2 drop-frames<sup>1</sup> for each tracked frame), making 60 CMA-ES iterations for each frame (red line in Fig. 7.9). Less iterations causes the algorithm to loose track of the object during the sequence. Hence, 60 is the minimum number of iterations on this sequence using the old initialization. The algorithm using the new initialization method is performing the tracking on every frame, performing only 20 CMA-ES iterations for each frame (green line in Fig. 7.9). It shows that the new method still outperforms the one using

<sup>1</sup>In this real-time context, a drop-frame is a frame that could not be processed because the processing time of a preceding frame is exceeding its time-frame, causing the algorithm to skip the frame completely as a newer frame is available once the processing of the preceding frame is finished.

## 7. Evaluation



**Figure 7.8.** The RMS depth error for all vertices over all frames of the sequence depicted in Fig. 7.11 using 40 CMA-ES iterations. The standard initialization (red line) is compared to the propagation scheme introduced in Section 5.2 (green line).

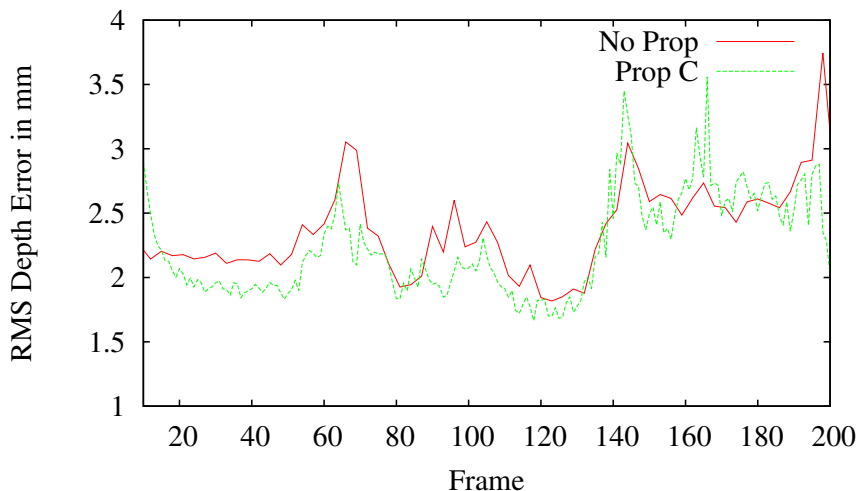
the standard initialization in most of the frames, using only a third of the number of iterations and hence yielding results three times as fast.

To visualize the advantage of the covariance propagation during the optimization process, the fitness value at each iteration at every frame was recorded for a deformation sequence similar to the one depicted in Fig. 7.11. Only 40 CMA-ES iterations were performed per frame with 24 individuals. Fig. 7.10 shows the fitness value at each iteration averaged over the 500 frames of the sequence for the tracking with (green line) and without (red line) covariance propagation. The plot (7.8) visualizes the advantage, the new initialization provides to the optimization process.

### 7.3.1 Real-time Tracking Experiments

To test the overall speed on an implementation of the proposed algorithm, a real-time test was performed. The time one frame takes to be calculated is linearly dependent on several factors: the number of vertices of the reference object, the number of CMA-ES individuals and the number of iterations. This section already showed that propagating the covariances can minimize

### 7.3. Relative Performance Experiments



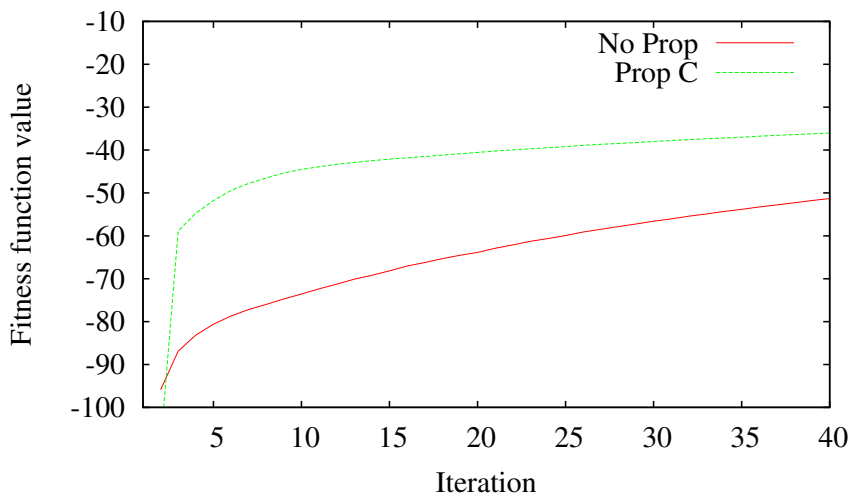
**Figure 7.9.** The RMS depth error for all vertices of the sequence depicted in Fig. 7.11 using the standard initialization and every third frame with 60 CMA-ES iterations (red line). The green line shows the results for the new initialization method using every frame and only 20 CMA-ES iterations.

the number of iterations if the fitness is constantly updated. This means the optimization becomes in a way continuous with permanent updates of the fitness function, as soon as a new input image is available. Nevertheless, the number of CMA-ES individuals and the number of vertices of the reference object should be chosen carefully to be able to keep track of the deformation in the input data.

For this real-time experiment, a rather sparse object representation by 225 vertices and a NURBS function with  $4 \times 4$  control points was chosen. Although an individual count of  $4 + \lceil 3 * \log(n) \rceil$  is recommended for CMA-ES,  $n$  being the number of dimensions of the search space, an individual count equal to the number of search dimensions always produced results more efficiently in this deformation tracking application. Hence, 48 individuals were used in this experiment.

The first row in Figure 7.12 depicts the input sequence that is processed, the second row shows the estimated deformation calculated in real-time. The processing frame rate was 25 Hz on an Intel i7 computer with 2.8 GHz. While the optimization results look adequate, the model error remains large due

## 7. Evaluation



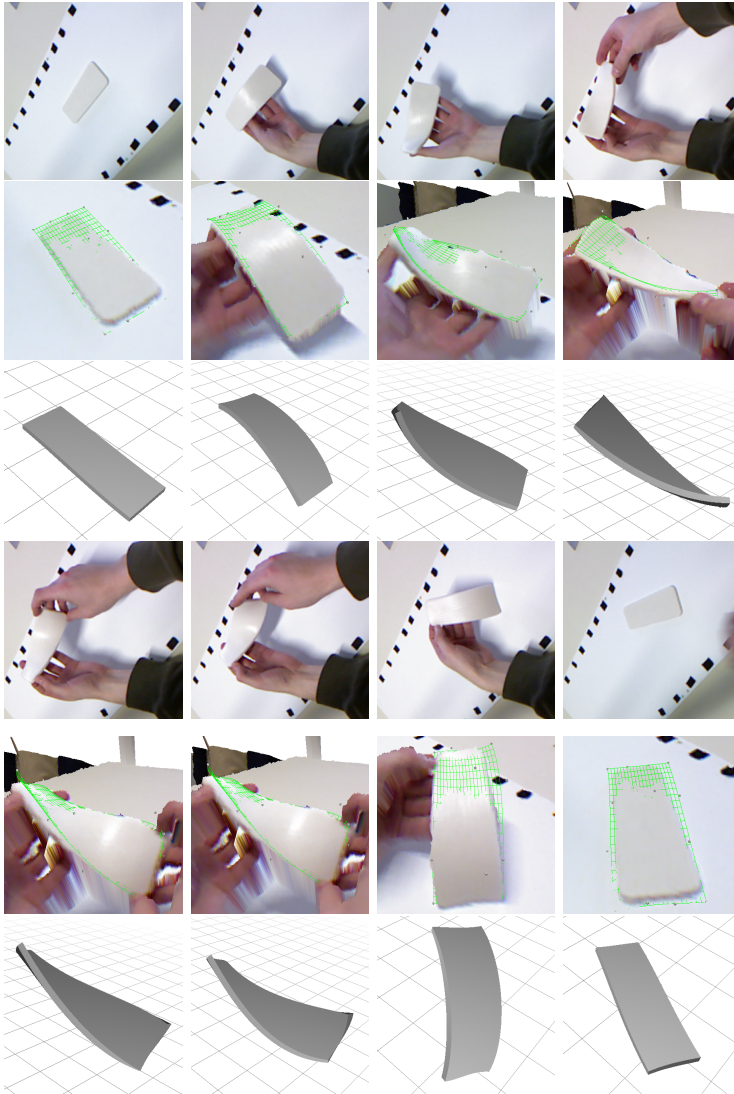
**Figure 7.10.** The average fitness value  $f$  over all frames of a 500 frame sequence similar to 7.11 using 40 CMA-ES iterations (24 individuals). The red line shows the values if the covariance is not propagated, the green line shows the values of the same sequence using covariance propagation. Higher values stand for a lower error in this plot.

to the few optimization iterations. For high accuracy and high frame rates tracking a sheet of paper, the generic NURBS function has to be replaced by a specialized deformation function with fewer dimensions, which will be introduced and tested in Section 9.3.

### 7.3.2 Scalability

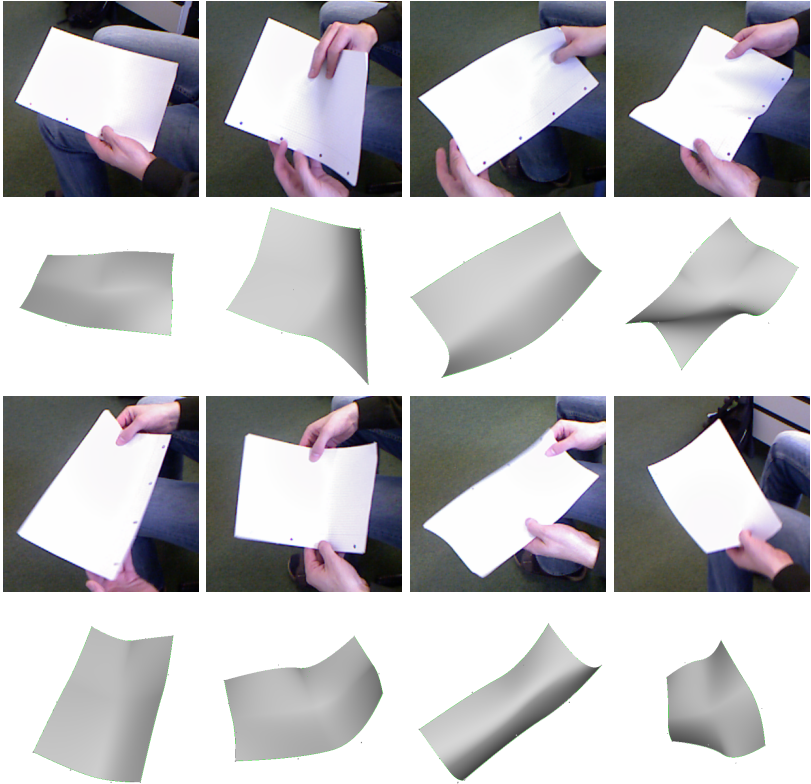
To evaluate the scalability, a number of experiments with increasing numbers of control points have been conducted. They showed that up to  $8 \times 8$  control points can be handled well by the proposed system. For complex geometries, using  $10 \times 10$  control points (300 dimensional search space) results in a very complicated optimization task. It is possible to handle such problems by significantly increasing the number of CMA-ES individuals, resulting in very long optimization times. Hence, modeling human poses or objects with a large number of joints is better done using specialized deformation functions, which

### 7.3. Relative Performance Experiments



**Figure 7.11.** Top row: video sequence recorded with a *Kinect* camera of a flexible white object on white background. Center row: the resulting colored 3D mesh with the NURBS deformation function (green grid) and its control points (gray dots). Bottom row: deformation function of the input sequence applied to a box.

## 7. Evaluation



**Figure 7.12.** The input sequence of the deforming paper is processed in real-time with 25 Hz. The first row depicts close-ups of the color images of the input data, the second row the estimated object surface.



can easily be integrated into the proposed system by exchanging the function  $\mathcal{D}$  (see Chapter 9).

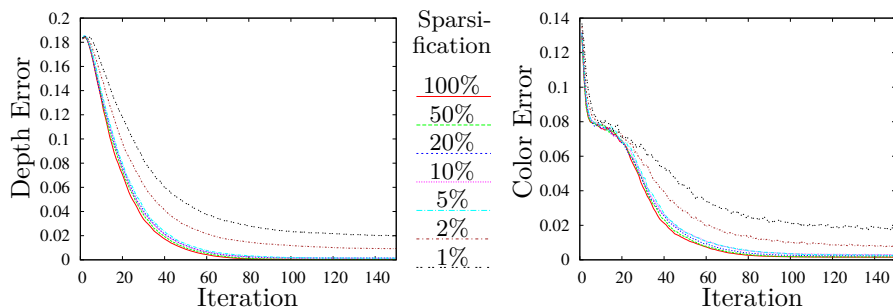
## 7.4 Random Sparsification Experiments

Testing the effect of randomized online sparsification is a complex task, since the error rates of sparsification are depending on a number of variables, e. g., object geometry and complexity, the background scene, deformation function and complexity of deformations, frame rates, and the optimization state and configuration. The relationship between the fitness noise induced by sparsification and the optimization process can be approximated mathematically (see Appendix C) but the relationship between the sparsification rate and the fitness noise induced by it cannot be estimated so easily, as it depends on the factors mentioned before. This section will show how a simple linear relationship between the iteration count and the sparsification rate can already improve the optimization performance significantly.

### 7.4.1 Sparsification Noise

Section 4.4.2 has introduced a method to additionally sparsify the synthesis model randomly online during the synthesis process to reduce the overall computational costs. To test the effectiveness of random sparsification for fast optimization, tests with various sparsification rates have been performed on two different tracking scenarios. While the optimizer is given the sparsified fitness values, the evaluation is done on the correct fitness values without sparsification error to analyze the sparsification effects on the optimization process. As a first test, a simple box shape is tracked (see Figure 7.15 (left)). Figure 7.13 depicts the depth- and color error values of a tracking sequence averaged over all 200 frames of the sequence. The tracked object mesh contains 6,777 vertices, 16 CMA-ES individuals were evaluated in each iteration. The input data has been artificially generated using a projection equivalent in resolution and intrinsic parameters to a *Kinect* camera. To simulate the inaccuracy of real input data, Gaussian noise has been added to the input data (3 mm standard deviation on the depth data, 5 color values standard deviation on each of the RGB channels (0 - 255)). The simulated object had a size of  $160 \times 225 \times 125 \text{ mm}^3$  and the average camera-to-object distance throughout the sequence was 515 mm. The plot shows that a considerable amount of vertices can be randomly removed from synthesis without it having

## 7. Evaluation



**Figure 7.13.** Average depth- (left) and color (right) errors in optimization runs using different random sparsification rates ranging from use of 100% of the vertices (red) to 1% (grey). 150 iterations and 16 CMA-ES individuals were used. For each iteration the (non-sparse!) depth and color error value is depicted to visualize the influence of sparsification on the real error. The error values do not refer to an absolute scale but serve the purpose of relative comparison only.

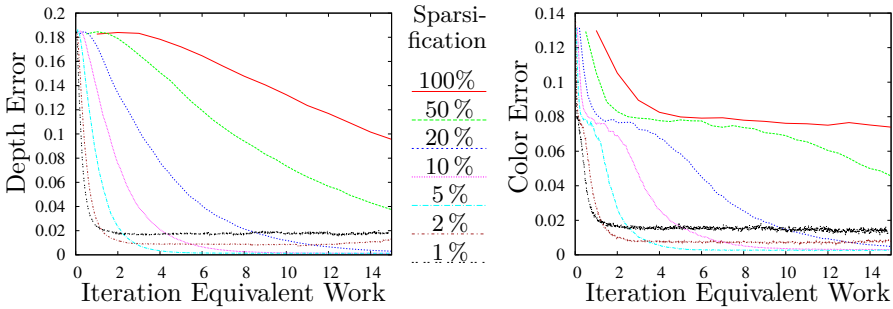
a large impact on the optimization, as e. g., using only 5% of the vertex set still yields comparable results.

The error values depicted in Figure 7.13 are plotted over the iteration count, but since the amount of computational work in each iteration is linear to the sparsification rate, plotting the error values over actual work better visualizes the effects of sparsification. Figure 7.14 depicts the same optimization data, but it is plotted over actual work, instead of iterations. The unit used in this Figure (Equivalent Iteration Work) is the iteration multiplied by the sparsification rate. This means that, as an example, iteration one using 100% of the vertices is displayed on the same x-position in the plot as iteration five using only 20% of the vertices ( $1 \cdot 0.2^{-1}$ ). Hence, Figure 7.14 shows the actual trade-off of sparsification: the optimization converges faster but is more inaccurate.

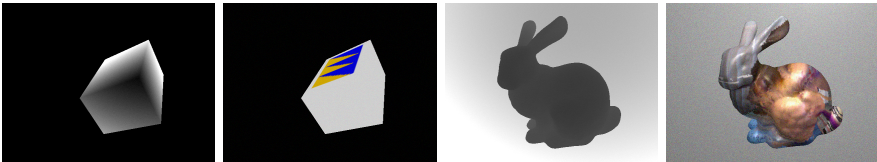
To also verify this result for more complex object shapes and smaller signal to noise ratios, a similar test has been performed using the Stanford Bunny<sup>2</sup> as a reference object (see Figure 7.15 (right)) and a Gaussian image noise with a deviation of 8 mm in the depth image and 15 color values on each RGB

<sup>2</sup>The Stanford Bunny is a computer model widely used in the computer vision industry by the Stanford University Computer Graphics Laboratory. Please refer to <http://graphics.stanford.edu/data/3Dscanrep/> for details.

## 7.4. Random Sparsification Experiments



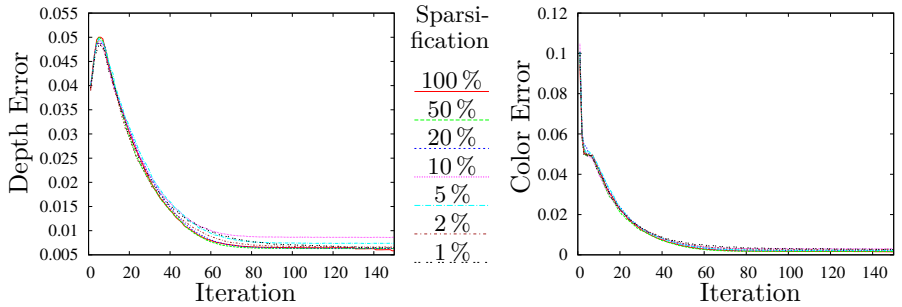
**Figure 7.14.** Average depth- (left) and color (right) errors in optimization runs using different random sparsification rates ranging from use of 100% of the vertices (red) to 1% (grey). In contrast to Figure 7.13, sparsified runs are plotted over the equivalent amount of computational work. As an example, iteration one without sparsification (red) is depicted with the same x value as iteration five using only 20% of the vertices. The amounts of computational work equivalent to 15 iterations without sparsification is depicted, using 16 CMA-ES individuals. The error values are calculated without sparsification to visualize the influence of sparsification on the real error. The error values do not refer to an absolute scale but serve the purpose of relative comparison only.



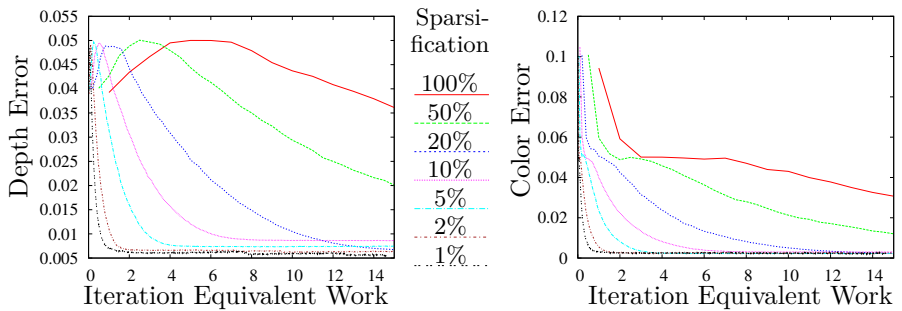
**Figure 7.15.** Noisy input data used in the online sparsification test. Left: depth and color input images of a simple box shape. Right: depth and color input images of the Stanford bunny model.

channel. The bunny model used in this test had a total of 83264 vertices. The results, depicted on Figure 7.16 and 7.17, show an even smaller impact of the sparsification on the optimization process. Even when only using 1% of the vertex set in each evaluation, the optimization results do not vary significantly from the full synthesis results, which is a consequence of the high vertex count of the object compared to its low frequency surface.

## 7. Evaluation



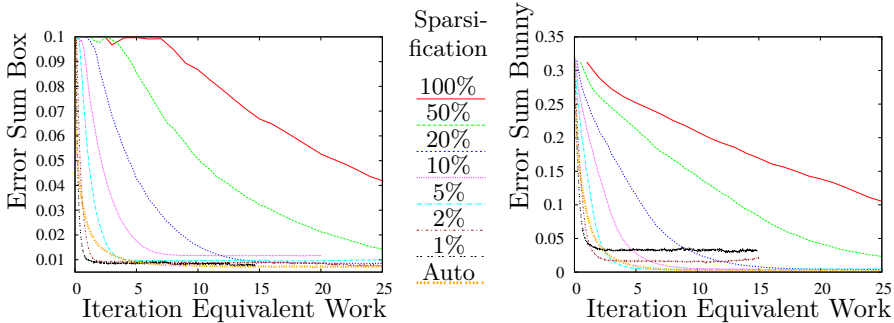
**Figure 7.16.** Average depth- (left) and color (right) errors in optimization runs using different random sparsification rates equivalent to the one depicted on Figure 7.13.



**Figure 7.17.** Average depth- (left) and color (right) errors in optimization runs using different random sparsification rates ranging from use of 100% of the vertices (red) to 1% (grey) equivalent to the one depicted on Figure 7.14.

To combine efficient optimization and high accuracy the sparsification factor can be adjusted between iterations. This allows to use high sparsification rates for the first iteration, moving the mean roughly near the optimum. Close to the optimum a less sparse synthesis then allows to achieve high accuracies. The sparsification rate can either be chosen by the iteration count, e. g., with a linear factor between sparsification rate and iteration count, or, for higher accuracy, based on the ratio between deviation  $\sigma_s$  (sparsification noise) of  $\mathcal{F}_s(\Theta_0)$  and the distribution of individuals in the fitness domain

## 7.5. Evaluation of Probabilistic Fitness



**Figure 7.18.** The sum of depth and color errors as shown individually in Fig. 7.13 - Fig. 7.17 with an additional plot showing the result of (linearly) adjusted sparsification for the Box scenario (left) and the bunny sequence (right).

$\mathcal{F}(\mathcal{N}_{(m,\sigma,C)}(\Theta_i))$  as recommended in Appendix C. Figure 7.18 shows test results of adjusting the sparsification factor linearly in comparison to the static sparsification test in Fig. 7.13 - Fig. 7.17. The optimization plot (orange) shows that the adjusted sparsification yields results efficiently while maintaining the accuracy of the full synthesis.

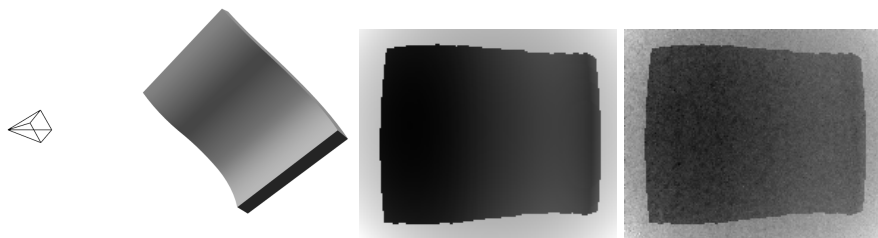
## 7.5 Evaluation of Probabilistic Fitness

This section evaluates AbS approach using the probabilistic fitness function design introduced in chapter 6.

### 7.5.1 Probabilistic Fitness on Synthetic Data

To evaluate the distribution-based approach to AbS for ToF cameras, an artificial deformation scene was generated (see Fig. 7.19) using the noise model as introduced in 6.2.1. Two sets of experiments have been conducted, one using the NURBS deformation model as described in Section 3, and one using the FlexPad deformation described in 9.3. Each of these tests was performed using the sparse synthesis as introduced in Section 4.4.1 and was compared to the new likelihood-based fitness. Note that these tests are not suitable to validate the noise model itself, but how the algorithm performs if the noise model is known.

## 7. Evaluation



**Figure 7.19.** Left: a 3D representation of an artificial deformation sequence showing the virtual camera and the deforming object. Center: the depth image of the artificial rendering using projection parameters of a *Mesa Imaging SR4000*. Right: the depth image augmented with noise simulating input data of an *Mesa Imaging SR4000*.

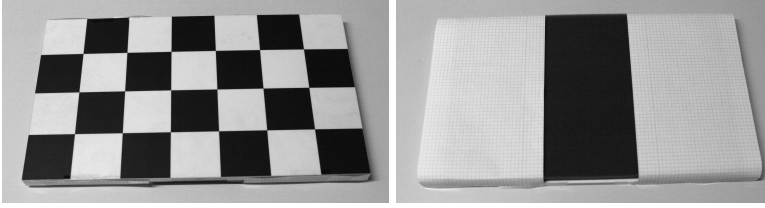
The experiments show that the distribution based AbS system outperforms the Euclidean-based AbS approach for known noise levels. The ratio between the accuracies of the two systems can be driven arbitrarily high just by choosing the noise model accordingly, as the Euclidean error function has no possibility to compensate for any mean offset in the data set, e. g., shifting the complete measurements by 10 cm results in an error of the Euclidean AbS raised by 10 cm whereas the probability-based AbS compensates for the shift in the mean function  $m$ .

To test the influence of applied knowledge about deviation in an isolated way, a synthetic sequence was generated applying artificial noise based on the pixel wise deviation measured in a real ToF camera. Using 10 different geometries, each tracked over a sequence of 500 frames, the deviation of the NURBS control points from the actual ground truth data was in average 1.92 times (SD 0.35) as high using the Euclidean-based AbS as it was using the distribution based AbS. This shows that even without known mean offset, the tracking can profit from the knowledge of the deviation as the likelihood-based AbS outperforms the Euclidean AbS.

### 7.5.2 Probabilistic Fitness on Real Data

The evaluation of deformation tracking algorithms on real data is always a difficult task. The reason is the lack of available ground truth for real scenes containing deforming objects and the missing generic model covering all possible deformations. An algorithm producing low reproduction errors does not necessarily yield true results. Just like optical flow can be used for

## 7.5. Evaluation of Probabilistic Fitness



**Figure 7.20.** The two test objects used in the real data tests. Left: the checker board has equally distributed areas of high and low reflection, Right: the black area in the center of the tracking object provides a special challenge for the tracking algorithm since the black area will cause an offset in the depth image.

efficient image compression, but does not always describe the real motion in the scene, since it does not use an underlying model to constrain itself to plausible movements.

To be able to offer comparable results nevertheless, the tests on real data in this article were performed using the NURBS deformation model as well as the FlexPad deformation model, which already proved the ability to model certain deformation scenes in a simulated setup as well as in real tests using a structured light system. To offer ground truth, the physical objects were not deformed, so any tracked deformation can be evaluated as deviation from ground truth. It is important to note that neither of the models encourages the optimizer to yield undeformed results by e. g., penalizing large deformations.

The tests were performed using two objects with known geometry, a checker board and a stiff white object with a black center, especially designed to show how well the noise model can handle systematic errors caused by varying reflection properties in the time of flight measurements. Figure 7.20 depicts the tracking targets.

A sequence of 500 images was recorded of each object while changing the object position and orientation. This sequence was used as input data for the tracking algorithm. Table 7.1 shows the average error over all sequence images. The error in the upper table was calculated by measuring the average deviation of the NURBS control points from a planar alignment, i. e., the average distance in mm over all control points from the plane that minimizes the distance to the control points. A NURBS function of degree 2 with  $4 \times 4$  control points was used, optimized with 1,000 CMA-ES iterations using 32 individuals.

## 7. Evaluation

**Table 7.1.** Test results comparing the Euclidean fitness function to the likelihood fitness. The cells contain the average error and the standard deviation put in brackets. The rows contain results for the tests runs using the checker board (Fig. 7.20 left) and the stripe board (Fig. 7.20 right) as tracking target. The upper table contains the average deviation in mm using the NURBS deformation model, the bottom table contains the average maximum bending parameter using the FlexPad deformation model.

Error using NURBS model		
	Euclidean	Likelihood
checker board	12.83 (5.31)	10.21 (5.49)
stripe board	35.71 (18.35)	17.08 (7.76)

Error using FlexPad model		
	Euclidean	Likelihood
checker board	0.16 (0.11)	0.12 (0.09)
stripe board	0.92 (0.23)	0.24 (0.17)

In contrast to the NURBS-based tracking, the Flexpad tracking delivers pose estimation and deformation values separately. To evaluate the tracking results, the bending parameter containing the maximum absolute value was considered in every frame. Bending parameters range from -2.0 (full bending away from the camera) to +2.0 (full bending towards the camera). Table 7.1 (bottom) contains the average maximum bending parameter for the two test sequences. The tracking used 400 CMA-ES iterations with sixteen individuals for each frame.

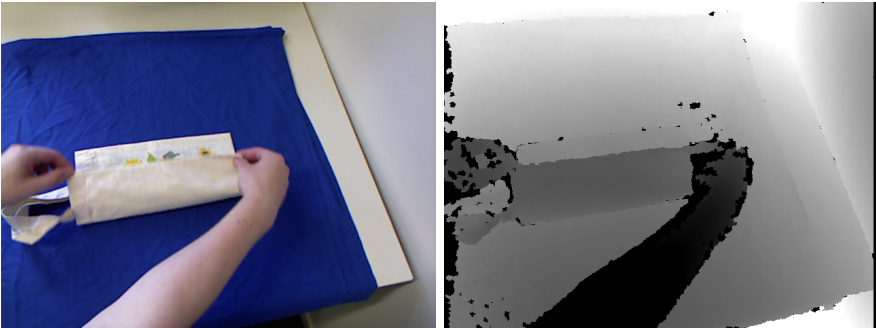
Summarising the results of the weighting strategies, a probabilistic formulation of the tracking problem allows to compensate for strong sensor noise, if a viable noise model is at hand. For ToF cameras, such a model can also be learned to a certain extend. For setups using sensors with very low noise, or noise that follows an unknown non-Gaussian distribution, the automatic adjustment proposed in Section 6.1 provides a parameter free alternative.

This chapter concludes the introduction and evaluation of the basic AbS pipeline. The following chapters will expand on this system by discussing methods for occlusion handling (Chapter 8) and by introducing alternative deformation functions (Chapter 9).



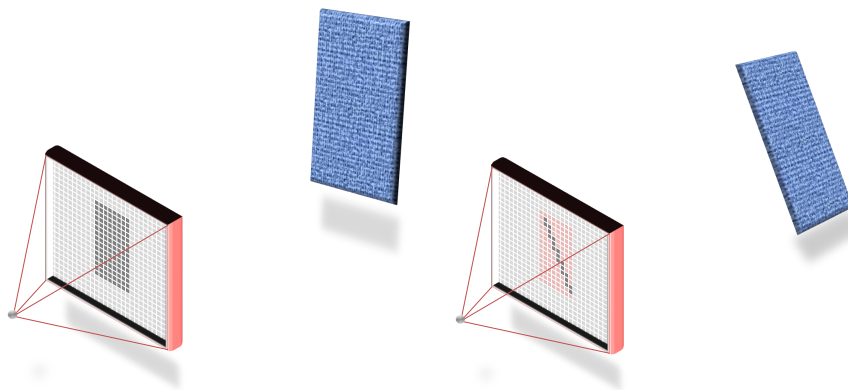
# Occlusion Handling

When transferring the theoretical problem of deformation tracking to its application in the real world, one of the major challenges arising is the problem of occlusion handling. Occlusions can occur in two forms: self-occlusion and occlusion by other objects. Both types of occlusion are usually present in a deformation sequence, as deforming moving parts usually occlude each other in one way or another. In addition, the process of deforming is often induced by extrinsic manipulators, e. g., hands, robot grippers, etc., which occlude the object as well, if the scene is not deliberately designed to avoid occlusions. Figure 8.1 depicts an example of a combination of both types of occlusion. In this chapter, solutions for both types of occlusions are introduced, benefiting greatly from the direct nature of the AbS approach.



**Figure 8.1.** Input material (left: color image, right: depth image) of a deformation sequence containing external occlusion (hands) as well as self-occlusion.

## 8. Occlusion Handling



**Figure 8.2.** Illustration of the “edge problem” on a synthesis using z-buffer. Left: a flat object is projected onto a depth image. Right: if rotated to the edge, all projected pixels of the object can still map the same depth values in the image although the object pose does not represent the depth image content. The red pixels represent the reference pixel “missed” by the synthesis.

### 8.1 Self-Occlusion Handling

Self-occlusions handling can be handled by the AbS system completely by simulating it in the synthesis stage. In case of rendering color and depth images, the self-occlusions can be implemented into the synthesis easily via z-buffering [WW91], which is the standard occlusion handling method and available on all modern graphics cards. But although the z-buffering allows a proper synthesis, it does not regard the problems arising during the analysis. Figure 8.2 (left) depicts a situation in which an object is projected into a depth camera as a synthesis step. In Figure 8.2 (right), the same object is rotated by 90 degrees so its edge is projected. Because of the z-buffering in the rendering pipeline, the depth values of the object’s front edge are correctly rendered in the image, matching the values of the initial pose, although the object pose does not match the depicted one (red pixels in Fig 8.2 right). During real data experiments it showed that this “edge pose” can have a lower average pixel error compared to the original pose because the deformation function can adapt more easily to the lower number of pixels for noisy input data. Objects without edges (e. g., cubes, spheres) may be less deceptive in this regard, but, depending on the deformation function, even these objects

may be squeezed to synthesize a thinner silhouette than is actually visible in the input image. This Section will introduce two methods to avoid such a behavior.

### 8.1.1 Noise Adaptation Heuristic

An intuitive approach to counter this problem is to perform an object segmentation in the input data prior to the optimization and force the synthesis to cover the segmented image areas by penalizing unmapped pixels. Unfortunately, such a segmentation requires a known silhouette, which is unavailable prior to the optimization. In addition, such a penalty would contradict the very fundamental idea of AbS, as it applies an interpretation to the input data prior to the actual matching process (see Section 1.1.1 for a discussion on this problem).

Because z-buffering is obviously required for proper self-occlusion handling in the synthesis, the analysis step has to be extended such that it prefers a complete object match over an “edge match”, only without making any interpretations of or assumptions about the image content. As a first approximation, it is assumed that each of the  $p$  pixels (or measurements) in the input image of one camera is subject to uncorrelated Gaussian noise of the same deviation  $\sigma$  and a mean at the correct value  $m$ . Hence, the expected error is

$$\begin{aligned}
 \left\| \sum_{i=1}^p |m_i - \mathbf{E}(m_i, \sigma)| \right\|_2 &= \left\| \sum_{i=1}^p |\mathbf{E}(0, \sigma)| \right\|_2 \\
 &= \mathbf{E} \left( \|\mathcal{N}(\mathbf{0}^p, \sigma \mathbf{1}^{p \times p})\|_2 \right) \\
 &= \sigma \mathbf{E} \left( \|\mathcal{N}(\mathbf{0}^p, \mathbf{1}^{p \times p})\|_2 \right) \\
 &= \sigma \chi_p
 \end{aligned} \tag{8.1.1}$$

See 5.1.20 for a definition of  $\chi_p$  and A.1.8 for a numeric approximation of it.

Not all aspects of the sensor induced noise can be modeled in an uncorrelated way<sup>1</sup> and differences between synthesis and input data may also stem from an insufficient deformation model (causing highly correlated discrepancies). To regard this fact, let the covariance  $\mathbf{C} \in \mathbb{R}^{n \times n}$  model this correlated noise. Due to the linearity of the involved components, (8.1.1) also applies to

---

<sup>1</sup>E.g. *Kinect* depth sensors yield highly correlated noise due to the region based depth estimation

## 8. Occlusion Handling

any normalized covariance, which means that

$$\mathbf{E}(\|\mathcal{N}(\mathbf{O}^p, \sigma \mathbf{C})\|_2) = \sigma \chi_p, \quad (8.1.2)$$

for  $\|\mathbf{C}\| = 1$ .

The “edge effect” is mainly caused by the lower number of pixels to which the deformation model can adapt more easily. Under the assumption, that the deformation model  $\mathcal{D}$  can be linearly approximated reasonably well in the parameter space vicinity  $\sigma \mathbf{C}$ , the optimum synthesis of the noisy data can compensate for  $n$  (dimension of the parameter space) of the  $p$  pixel measurements, reducing the noise at the optimum to  $\sigma \chi_{(p-n)}$ .

To counter this effect, a compensation factor  $c(p, n)$  can be defined such that, when multiplied to the average pixel error, it causes the fitness to remain equal for varying pixel counts. Let  $\Theta_1$  and  $\Theta_2$  denote two parameter vectors with different pixel counts  $p_1$  and  $p_2$ . Under the assumption that both parameter vectors describe perfect fits that only deviate from a zero fitness value (no errors) due to the noise in the input image, then  $c(p, n)$  is supposed to satisfy

$$c(p_1, n) \mathcal{F}(\Theta_1) = c(p_2, n) \mathcal{F}(\Theta_2), \quad (8.1.3)$$

which, using the introduced noise adaptation model is equivalent to

$$c(p_1, n) \sigma \chi_{(p_1-n)} = c(p_2, n) \sigma \chi_{(p_2-n)}. \quad (8.1.4)$$

As the optimization problem is independent of the scale of the fitness, it is legitimate to set  $c(p_2, n) = 1$  as a reference value, which implies

$$c(p_1, n) = \frac{c(p_2, n) \sigma \chi_{(p_2-n)}}{\sigma \chi_{(p_1-n)}} = \frac{\sigma \chi_{(p_2-n)}}{\sigma \chi_{(p_1-n)}}. \quad (8.1.5)$$

This formulation reveals that the actual noise level  $\sigma$  is irrelevant as the fraction is reduced. For higher pixel counts,  $\chi$  is approximating  $\sqrt{\cdot}$  very closely, and it follows that

$$c(p_1, n) = \frac{\sigma \chi_{(p_1-n)}}{\sigma \chi_{(p_2-n)}} = \frac{\chi_{(p_1-n)}}{\chi_{(p_2-n)}} \approx \sqrt{\frac{p_1-n}{p_2-n}}. \quad (8.1.6)$$

This factor is easy to calculate and removes the advantage of the “edge match”. In practice, adding this factor in the synthesis step greatly reduces the number

of times in which the optimizer favors an “edge match” over the true match. An evaluation of this extension on real data can be found in Section 8.2.2.

### 8.1.2 Enforcing Contour Discontinuities

The noise adaptation heuristic is a very generic way to circumvent trivial matches with low pixel counts and it can be applied to every AbS instance without hesitation, as its effect to the overall fitness is intentionally minimal. But in some situations this minimal effect might not be enough to prevent wrong trivial matches. In these cases, additional information about the reconstruction problem has to be exploited.

A property that can be found in many objects is a certain degree of thickness, causing depth discontinuities at the object contours. A heuristic enforcing the complete contour of an object to be discontinuous might not apply to many situations, but one that requires e. g., 50% of the contour to have a minimum discontinuity can be applied in most cases. This discontinuity can also be used in the color space, as the object contour can be retrieved from the color synthesis as well.

As the *Kinect* sensor as well as ToF cameras have problems at depth discontinuities, it is a viable strategy to calculate the discontinuities directly at the border pixels but with a distance according to the contour uncertainty of the device (2 to 3 pixels using a *Kinect*), so a wrong distance measurement directly at the object edge does not enforce the contour to follow it.

## 8.2 Model-Based Occlusion Handling

In conventional tracking methods, the handling of (external) occlusion is straight forward. Each feature descriptor in the image matching a feature descriptor associated with the tracked object is used for the object reconstruction, all other feature information is considered to be part of the environment or an occlusion. AbS does not provide such an easy handle on the occlusions problem, but it still allows efficient occlusion handling if a few assumptions are made about the object movement.

One of the great features of the AbS approach is the capability to cope with incomplete image data (see Section 7.1.2). Hence, occlusion handling can be performed by removing image areas that occlude or might occlude the object from the input data. Although this can be seen as a contradiction to the AbS philosophy about not touching the input data, it can be done in a way without “interpreting” the incoming data, all that is required is

## 8. Occlusion Handling

an assumption about the maximum object movement speed and the color constancy of the object and reliance in the match of the preceding frame.

### 8.2.1 Assumptions and Models

Occlusions are image areas depicting a point that does not belong to the tracked object but is positioned in front of it in at least one sensor. Consequently, as occlusions are sensor specific, the occlusion handling will be done separately for each sensor. Every pixel in every input sensor has to undergo a test for being a potential object pixel, and if not, if it occludes the real object. To allow a more formal definition, it is required to formulate two assumptions about the tracked object:

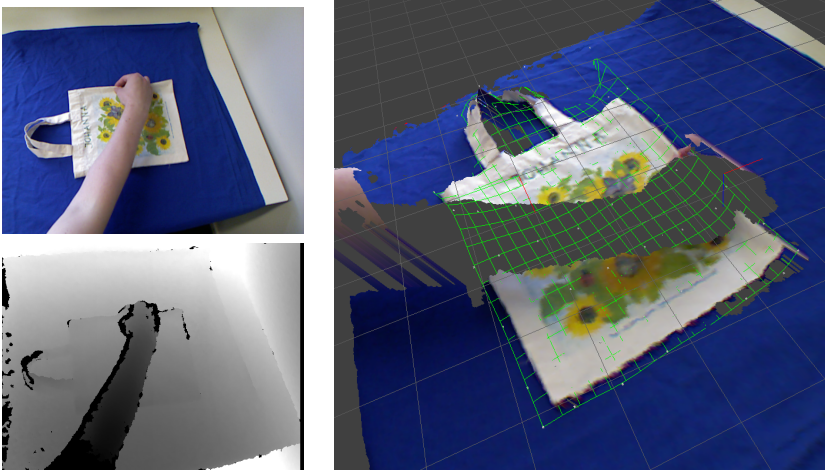
- ▷ no part of the object will exceed a maximum speed  $\gamma_{\max}$  during the tracking,
- ▷ there is a vicinity  $\Omega(c) \subset [0, 1]^3$  in the color space  $[0, 1]^3$  for each color  $c \in [0, 1]^3$ , providing the maximum change in color, an object point for a given original color  $c$  can undergo in the image sequence due to changes in lighting and perspective.

This leads to the assumptions that, for a given frame rate  $f$ , the 3D vertex movement between two consecutive frames satisfies an upper bound, i. e., for a vertex  $\mathbf{v} \in \mathbb{R}^3$  in one frame and its position  $\mathbf{v}' \in \mathbb{R}^3$  in the next frame

$$\|\mathbf{v}' - \mathbf{v}\|_2 \leq \frac{\gamma_{\max}}{f}. \quad (8.2.1)$$

In a similar way, the color  $c$  of a pixel not being occluded is assumed to be fairly constant, yielding the assumption that for any pixel in the input image representing the object, a vertex can be projected into the pixel vicinity that has a color within  $\Omega(c) \subset [0, 1]^3$ .

Hence, 3D and color criteria are defined allowing a preliminary classification of image pixels. A set of rules can be applied to the input image to remove non-object pixels potentially occluding the object. Removing input areas supposed to occlude the actual object from the input data can be done without leaving unfeasible data to the main tracking algorithm, because of the ability of the algorithm to cope with holes in the input images (see Section 7.1.2). If  $\gamma_{\max}$  is set to twice the amount of movement from previous frames, 8.2.1 can be used to calculate a 3D region and a corresponding image area for each vertex, in which the projection can be found in the current frame. Given the color error  $f_c$  for the vertices, it is appropriate to assume that most of the vertices



**Figure 8.3.** Removing occlusions from the input data. Left: input color image and depth image. Right: 3D representation of the input data from which possible occlusion pixels have been removed. A green grid visualizes the deformation function of the tracking algorithm.

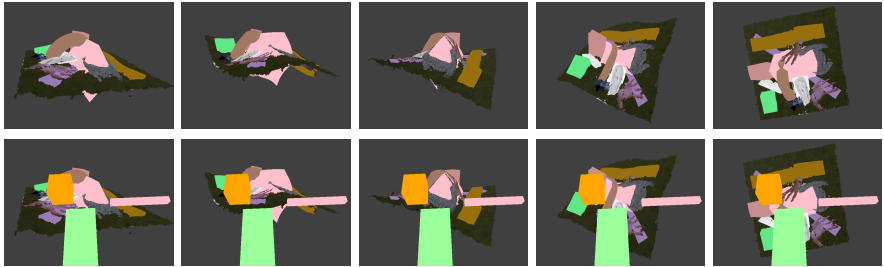
have a color error below twice the average color error of the preceding frame. This is equivalent to a  $2\sigma$  threshold to separate an object pixel color from colors of a potential occlusion.

These assumptions allow to formulate a two-step occlusion classifier. In a first step, each pixel in the depth and color image is assigned one of these states:

1. not classified,
2. depth and color are within the projected area in the image of at least one vertex,
3. only the depth value is within the projected area of an object vertex in the image but the color value does not fit, or
4. the pixel seems to be part of an occluding object.

Starting with every pixel set to state 1), for every vertex  $v$  in  $V$  and every pixel  $[x, y]$  in the vicinity (as defined above) of the projection of  $v$ , the following rules are processed.

## 8. Occlusion Handling



**Figure 8.4.** A synthetically generated, complex geometry, deformed and rendered as depth and color image sequence. The first line depicts the rendered color data, the second line shows the same data augmented by occluding boxes to evaluate the stability of the algorithm on occluded data.

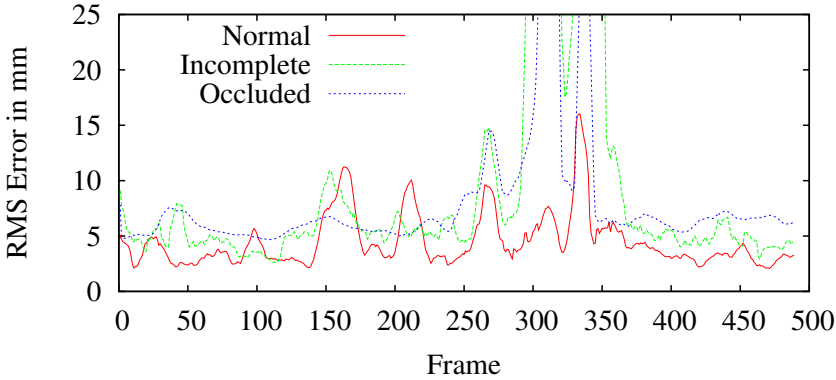
- ▷ If  $[x, y]$  is within the projection of the three dimensional vicinity of  $v$  given by 8.2.1 according to its recent movement and the color of  $v$  is in the vicinity of the color at  $[x, y]$ , then set the pixel state of  $[x, y]$  to 2).
- ▷ If the state of  $[x, y]$  is not 2) and  $[x, y]$  is within the projection of the three dimensional vicinity of  $v$  but the color of  $v$  is not in the vicinity of the color at  $[x, y]$ , then set the pixel state of  $[x, y]$  to 3).
- ▷ If the state of  $[x, y]$  is not 2) the depth value of  $[x, y]$  is outside the projected vicinity of  $v$  such that it is in front of  $v$ , then set the pixel state of  $[x, y]$  to 4).

This procedure computes a classification into certain object pixels 2), uncertain pixels 3) and certain occlusion pixels 4). In a second step, every pixel classified as 3) in the vicinity of a pixel classified as 4) is also set to 4). Finally, every pixel classified as 4) is removed from the input data.

### 8.2.2 Evaluation

To evaluate the proposed occlusion handling method, tests on synthetic data with available ground truth have been conducted, as well as an evaluation on real world data. Based on the data from Section 7.1.1, three 3D boxes were placed within the scene (see bottom line in Fig. 8.4) occluding the deforming object. Unlike in the blackened areas in Section 7.1.2, these added elements provide valid depth data. The RMS distance between the input vertices and the tracked vertices was 8.59 mm. Again, ignoring the 59-frame part from frame





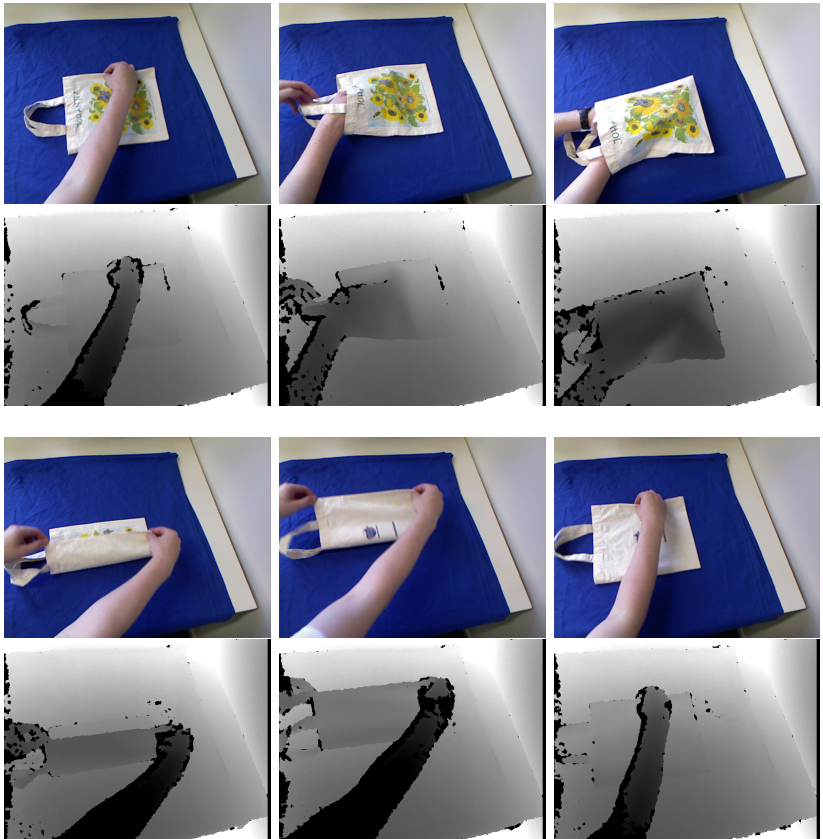
**Figure 8.5.** The RMS distance between the original deformed object vertices and the tracked object vertices for the 500 frame input sequence.

295 to 354 leads to an average distance of 6.12 mm. The blue line in Figure 8.5 shows the RMS error for each frame of the sequence. As a comparison, the evaluation of the experiment with deleted image information is depicted in Figure 8.5 as well. It shows that occluding objects do not influence the overall tracking process but only cause a loss of accuracy which is equivalent to the missing image data in the sequence.

For the evaluation of the occlusion handling along with the self-occlusion handling proposed in Section 8.1.1 on real data, a sequence of a deforming bag was captured using a *Kinect* camera. Figure 8.6 depicts a selection of input images.

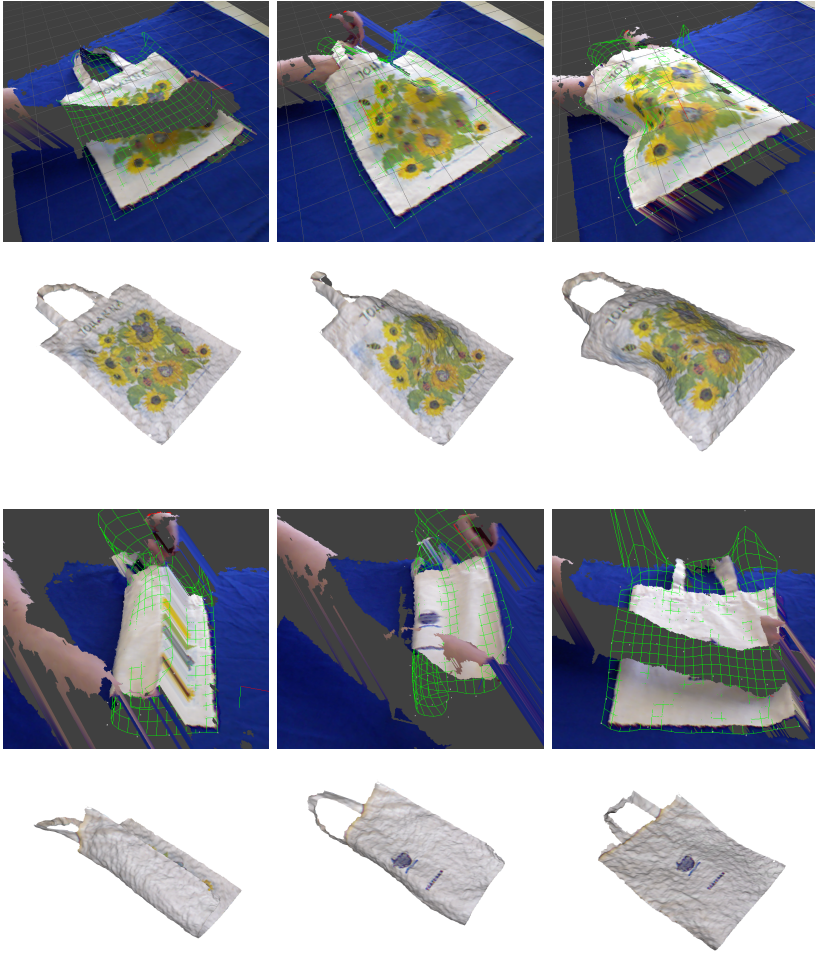
The sequence shows how a bag is manipulated by hands, which leads to massive occlusions of the tracked object. To evaluate the self-occlusion handling, the object is turned on its back during the sequence. For the generation of the 3D visualization shown in Figure 8.7 (upper rows), the depth image filtered by the occlusion algorithm was back-projected into 3D space and textured using the color camera input image. It is visible how the occlusion handling removes the parts of the hands and arms that would occlude the tracked object. The deformation function is visualized as well by showing the underlying NURBS surface (green grid). The reference object model deformed by the deformation function is shown in Figure 8.7 (bottom rows) how it follows the deformation of the real bag. 350 CMA-ES iterations using 96 CMA-ES individuals were processed offline on a  $7 \times 7$  NURBS function. The self

## 8. Occlusion Handling



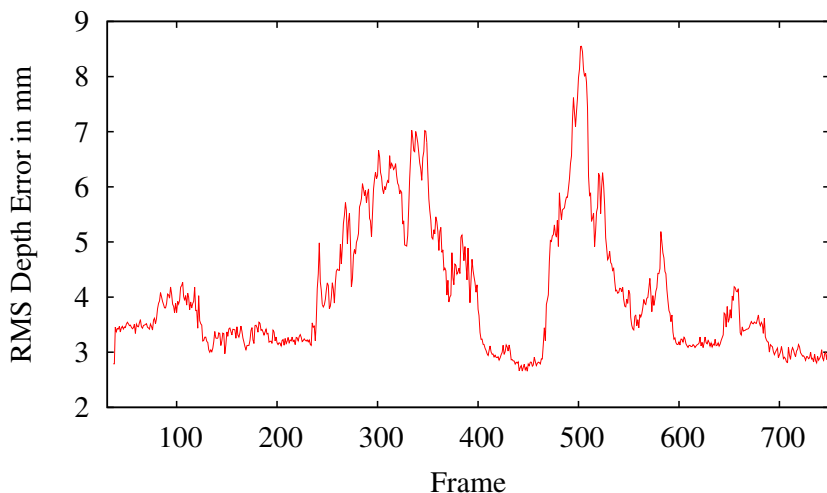
**Figure 8.6.** Test input of the occlusion handling evaluation on real data using a bag manipulated by hands. The upper images in each row depict the color input data. The lower images show the corresponding depth images.

## 8.2. Model-Based Occlusion Handling



**Figure 8.7.** Test results of the occlusion handling method on real data using a bag manipulated by hands. The upper images in each row depict a 3D visualization of the depth and color input data from which image data was removed that could potentially occlude the object. The green grid visualizes the NURBS surface of the deformation function at its current state. The lower images depict the resulting deformation of the reference 3D mesh model.

## 8. Occlusion Handling

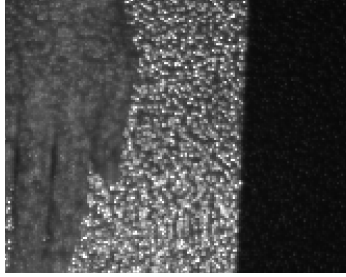


**Figure 8.8.** The RMS depth error for all visible vertices over all frames of the bag-turning-sequence depicted in Fig. 7.6.

occlusion works as well, as the object is turned correctly from one face to the other. Fig. 8.8 shows a plot of the average depth error for every frame of the sequence. The average error over the complete sequence is 3.98 mm.

### 8.3 Image Based Occlusion Handling

The color input data provides an important clue to the occlusion handling, as other objects directly interacting with the tracked one can not be segmented by depth information, but only based on color. If no color information is available, additional knowledge has to be used to label occluding image parts. Again, the ability of the AbS approach to easily cope with missing image information reduces the occlusion handling task down to marking the occluding pixels in the input data. The next Section will introduce an example application, in which no color information is available, and a blank sheet of paper is deformed by hands, heavily occluding the tracked object.



**Figure 8.9.** Three different materials captured by the *Kinect* IR camera. Left: skin, center: paper, right: dark cloth.

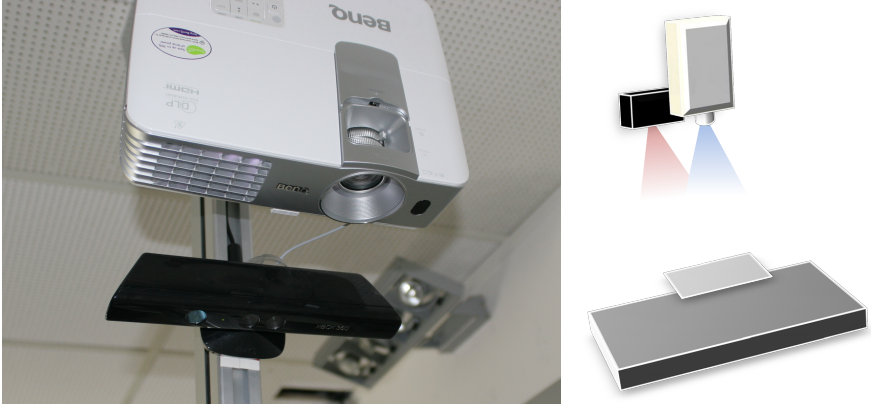
### 8.3.1 FlexPad Material Filter

The following method was first introduced as a part of the tracking component of the FlexPad system [SJM13] and was patented under [JSK13]. The FlexPad system is a human computer interface in form of a virtual, deformable display. The actual display information is projected onto the display surface (which can be a sheet of paper or foam) using a video projector. To locate the paper in 3D space and determine its deformation, a *Kinect* sensor is placed next to the projector. Figure 8.10 depicts a schematic drawing of the setup.

To get an accurate reconstruction of the paper shape, the hands have to be removed from the input images prior to the actual optimization (described in Section 9.3). But since the system is projecting arbitrary content onto the object surface and the interacting hands, color information cannot be used for segmentation. Based on depth information alone, efficient segmentation between the hands and the deformable object is difficult due to missing prior knowledge about the object shape and the complexity of the human hands.

The key to separating hands from object is provided by the optional IR camera information of the *Kinect* device. As explained in Section 2.2.1, the *Kinect* captures depth information by triangulating between an IR point pattern projector, the object surface, and an IR camera, observing the disparity of the dot pattern on the surface in the image. It is easily understood, that different materials show different reflection behaviors of the IR pattern in the camera image. Materials with a high reflectivity in the IR domain provide a brighter response of the pattern than materials with less reflectivity. But in addition to the reflectivity property, the dot pattern also allows to estimate the translucency of the object surface. Figure 8.9 displays the effect on three different materials. High translucency is not seen very often in every-day

## 8. Occlusion Handling



**Figure 8.10.** Left: photo of the FlexPad setup: a video projector and a *Kinect* sensor are mounted on the ceiling overlooking the interaction area. Right: a schematic view of the FlexPad setup.

objects, but the human skin is one of these materials, allowing light to partly penetrate it while it is scattered multiple times. The visible result is a halo around each dot of a pattern in the image, blurring the distinct point features.

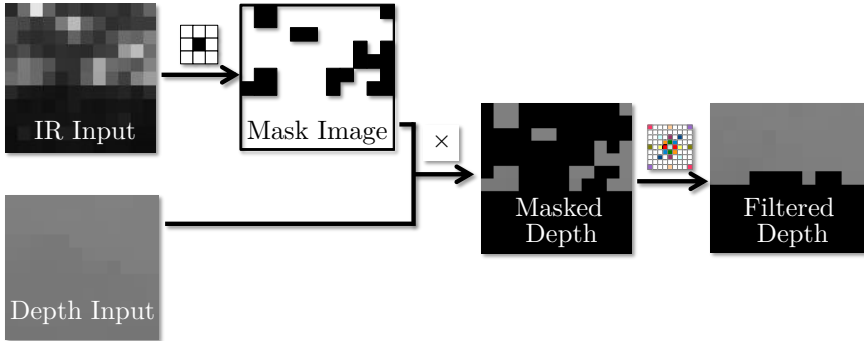
The FlexPad system takes advantage of these effects by analyzing the intensity of the pattern as well as the amount of softening around the pattern dots, i. e., evaluating the amplitude and the gradients of the image. Let  $(u, v)$  denote an image coordinate in the IR image  $I^{\text{ir}}$ , then

$$\eta_{\text{int}}((u, v), I^{\text{ir}}) = \max\{I^{\text{ir}}(u + i, v + j) : (i, j) \in \{-1, 0, 1\}^2\}. \quad (8.3.1)$$

$\eta_{\text{int}}$  describes the maximum intensity of a  $3 \times 3$  window around a pixel which serves as a first approximation of the surface reflectivity at that surface location. In a similar way, a translucency index can be acquired by evaluating

$$\eta_{\text{grad}}((u, v), I^{\text{ir}}) = \max\{|I^{\text{ir}}(u, v) - I^{\text{ir}}(u + i, v + j)| : (i, j) \in \{-1, 0, 1\}^2\}. \quad (8.3.2)$$

Since the geometry of the evaluated surface is known from the depth image, the influence of changes in surface normals and distances between projector, surface, and camera on the pixel brightness can be compensated by applying the factor  $|\mathbf{v}|^3(\mathbf{v} \cdot \mathbf{n})^{-1}$  to each IR measurement with  $\mathbf{v}$  being the observed



**Figure 8.11.** A schematic overview over the material filter. From left to right: the IR input is analyzed by the intensity and gradient thresholds, yielding a mask image. The mask is applied to the depth input data and filtered by the convex-hull-fill filter.

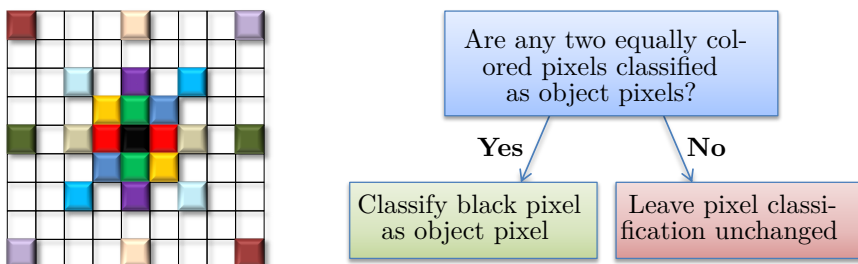
surface point in camera coordinates and  $\mathbf{n}$  the corresponding surface normal. See Appendix B for a complete deduction of the compensation equation.

Analyzing these two descriptors ( $\eta_{\text{int}}$  and  $\eta_{\text{grad}}$ ), allows to make a pixel-wise decision between the well reflecting, non-translucent paper, and the medium reflecting, translucent skin of the user by posting thresholds for  $\eta_{\text{int}}$  and  $\eta_{\text{grad}}$ . Unfortunately, not every pixel yields a correct classification, most pixels are in between dots and others suffer from aliasing effects as they share one dot on up to four pixels. The effect is, that only few pixels of the IR image are classified as object pixels. The holes between the classified pixels have to be removed in order to serve as input data for the AbS system. Note, that at this point only little of the available depth information is lost, as the pixel-wise depth estimation of the *Kinect* triangulation is interpolating implicitly between the visible dots in the image as well.

Figure 8.11 shows a magnified part of the IR image and the depth image and the processing stages. The masked depth image, which is only a sparse representation of the classified-image area, is filled in a last filter step. Figure 8.12 depicts a visualization describing how this fill-filter works: assuming the projection of the tracked object is convex in a local vicinity of each object pixel, the gaps between classified object pixels can be closed by filling the spaces in between, i. e., a pixel should be classified as object pixel if it lies directly in between two other object pixels.

The approach of the fill-filter is to check for pixels already classified as being part of the object in a given window around a center pixel. If two pixels

## 8. Occlusion Handling



**Figure 8.12.** Visualization of the interpolation filter, filling the local convex hull of already classified pixels. The filter is applied multiple times to generate a dense object area. The wrist parts in rows 1, 4, and 5 are not classified as skin because these were covered by cloth (sleeves).

at symmetric positions with respect to the center pixel are classified as object pixels, then the center pixel is inside (or at the edge) of the convex hull of classified pixels and is marked as object pixel as well. The depth value assigned to this pixel is not the depth value from the original image, but the average depth of the pixels used for the object classification. This way, the filter does not run into the problem of reactivating non-object image parts of the depth video (e.g., by interpolating across a finger tip). Instead such image areas are filled with interpolated values of the object distance.

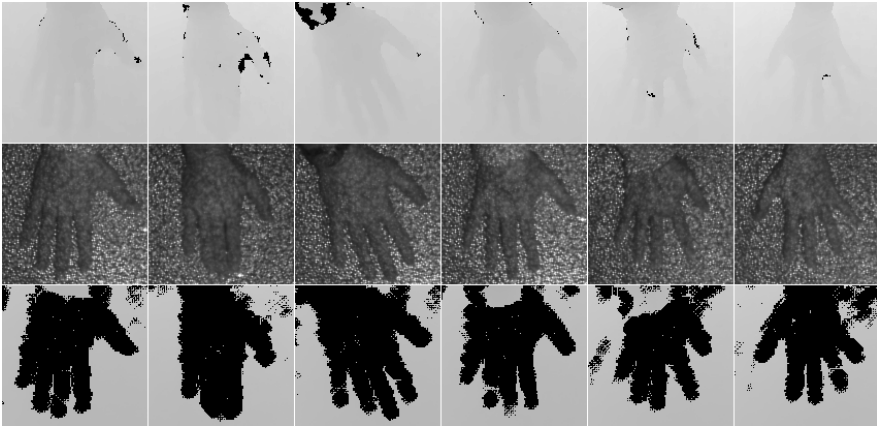
To make this classification process efficient, not all symmetric pixel combinations are tested, but only a selected set that is designed to provide a dense result if the filter is applied multiple times (see Fig. 8.12 for a visualization of the filter design).

### 8.3.2 Evaluation Example

The skin removal filter was evaluated by testing it on 10 persons of varying age (25 - 54) and different skin tones without adjusting any of the intensity or gradient thresholds during the test. The evaluation of the test results yielded a removal of over 99% of the skin pixels while keeping most of the underlying object in the image. Figure 8.13 shows a selection of test input images along with their segmentation results. Later experiments showed that



### 8.3. Image Based Occlusion Handling



**Figure 8.13.** 6 input image pairs from the skin removal test. Top row: the input depth image. Center row: the input IR image. Bottom row: the resulting depth image in which the skin pixels have been removed. Note that the areas in which fingers might be classified wrong (e. g., too short) due to the convex-hull-fill filter, the actual depth value is the interpolated object distance, not the distance to the occluding hand.

up to 6 different materials can be classified pixel-wise if additional knowledge about the dot pattern is used.



# Specialized Deformation Functions

Chapters 3, 4, and 5 introduced a versatile tracking method for flexible objects, delivering excellent test results on synthetic and real data (see Chapter 7). But thanks to the exchangeability of the synthesis, the analysis, the constraints, and the optimization parameters, the approach in these chapters should be understood as more of an example instance for a certain class of deformation tracking algorithms. This chapter will introduce a number of algorithms following the same approach while customizing one or more parts of the components to meet certain problem specific conditions.

## 9.1 Rigid Transformations

The most common transformation of vertices in computer graphics and vision is the rigid transformation, i. e., global translation and rotation applied to all vertices of a triangle mesh. Although not a deforming transformation, rigid transformations are included as a subset of the deformation space of most of the transformations introduced in this thesis. In the deformation functions introduced in this chapter, this is done explicitly, i. e., with a dedicated set of parameters. Although rigid tracking of rigidly transforming objects is simple compared to complex deformations, this Section will evaluate the AbS performance for the sake of completeness. In addition, it is worth while taking a closer look at the parametrization of the rigid transformation, because there are multiple ways to represent the rotational part of the transformation.

## 9. Specialized Deformation Functions

### 9.1.1 Rotation Parametrization

The two most common parametrizations for orientations in 3D space are Euler angles and quaternions [PW82]. In most situations quaternions are the preferred choice, as they provide a continuous parametrization of the rotational space and do not suffer from the gimbal lock problem [MSZ94]. Euler angles have. But as quaternions require 4 parameters to be estimated, compared to only 3 using Euler angles, estimating the correct rotation in a tracking method implies one more dimension added to the search space. In addition, the unit-sphere constraint on the quaternion parameters is not enforced easily on CMA-ES. If the constraint is ignored completely, and the parameter guesses are simply normalized in the evaluation, the parameter scale will have no effect on the fitness at all. This is unfavorable, since the CMA-ES covariance matrix  $\mathbf{C}$  as an approximation of the inverse Hessian (see Section 5.1) will develop an eigenvalue that is constantly growing towards infinity, leading to potential numeric evaluation problems of the matrix. On the other hand, introducing a constraint to keep the quaternion parameters normalized is not trivial, since the stricter the penalty on the quaternion scale, the more the search space is reduced to the valid parameter constellations. But at the same time, a high penalty shadows the effect of all other (non-quaternion) parameters (leading to a higher number of iterations) and it will do so across the complete set of iterations, as the covariance matrix represents a linearization of the search space, and, hence, can never be valid for a sub-domain of constant curvature as the unit sphere of the quaternions is.

Having this in mind, the Euler angles seem to be the more suitable parametrization, especially since the gimbal lock problem can be avoided in the case of tracking in a sequence: gimbal lock occurs if one axis has been rotated by  $90^\circ$  such that the other two axis align. In that case, the degree of freedom is locally reduced to two, a situation not desirable during the optimization. But as one can expect the tracked object not to turn more than  $90^\circ$  between two consecutive frames, the problem can be solved by estimating the orientation in an incremental way, i. e., by defining the identity rotation to be the orientation result of the previous frame. In addition, a “gimbal-lock-free” start parametrization should be chosen rotating around all three axes (e. g., not ZYZ or ZXZ). Hence, for a rigid transformation, the parameter vector is given by

$$\Theta = (\mathbf{t}_x, \mathbf{t}_y, \mathbf{t}_z, r_x, r_y, r_z) \quad (9.1.1)$$

## 9.2. Estimating Physical Parameters

with  $\mathbf{t} \in \mathbb{R}^3$  containing the translational parameters and  $\mathbf{r} \in \mathbb{R}^3$  containing the Euler angles of the rotation, the rigid deformation function is given by

$$\mathcal{D}(\mathbf{v}, \Theta) = \mathbf{R}_z(r_z)\mathbf{R}_y(r_y)\mathbf{R}_x(r_x)\mathbf{v} + \mathbf{t}. \quad (9.1.2)$$

### 9.1.2 Evaluation

The rigid transformation is not evaluated separately in this thesis, as it is part of the deformation functions introduced in the following chapters (9.2 and 9.3). The evaluation of these deformation functions can be found in Section 9.2.3 and Section 9.3.2.

## 9.2 Estimating Physical Parameters

One of the customized versions of this AbS deformation reconstruction method aims at estimating the physical properties of an object based on its deflection when picked up (introduced in [FJP<sup>+</sup>12]). It is assumed that the object consists of one material only. From a resting position on a straight surface it is picked up by a robot gripper. The observed change in the object shape moving from a resting position on a straight surface to a deflected state, is used to calculate Young's modulus<sup>1</sup> and Poisson's ratio<sup>2</sup>, the variables describing the stiffness of a homogenous material. Since  $\nu$  (Poisson's ratio) and  $E$  (Young's modulus) cannot be determined directly from the object deflection, they are added to the optimization parameters. These two additional parameters are evaluated by a physical plausibility penalty, which is added to the overall fitness.

### 9.2.1 Curvature Based Deformation

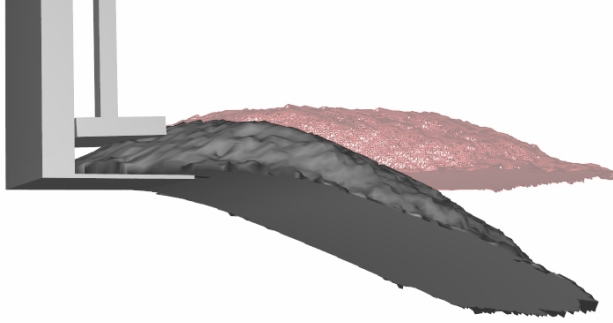
For gripper-based interaction, it is possible to reduce the set of generic deformations to deflections typical for this scenario. In order to do so, the undeformed object model is divided into Sections. The first Section describes the area in which the gripper directly touches the object surface, the remaining Sections are equally distributed over the object part hanging loose under the

---

<sup>1</sup>Young's modulus is defined as the normal stress divided by the linear strain (definition according to [NJK09]) inside the object material

<sup>2</sup>Poisson's ratio is the ratio of latitudinal to longitudinal strain, which describes the extent to which a material is distorted in a direction perpendicular to an applied stress (definition according to [All99])

## 9. Specialized Deformation Functions



**Figure 9.1.** A rendering of a flexible object held by a robot gripper. The difference in deflection between the undeformed object shape (red, transparent) and the picked up shape (gray, solid) allows to deduce material properties of the object.

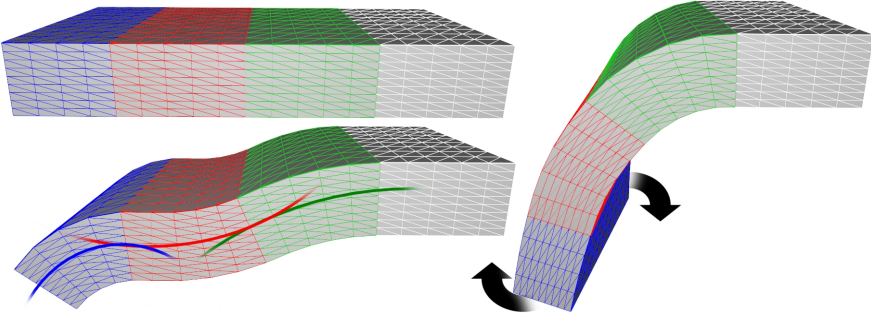
influence of gravity. The gripper Section is not visible to the optical sensors, but as it is rigidly connected to the robot, the pose of this object part is well known. Outside of the gripper Section, the object is subject to gravity and, thus, deflecting. Each of these Sections is subject to a constant curvature operator moving the vertices of the Section. Each vertex  $\mathbf{v}$  in a Section of size  $x_{size}$  is manipulated by the mapping

$$\mathbf{v} \mapsto \begin{pmatrix} \mathbf{v}_x \cos(\alpha) - (\mathbf{v}_z - r) \sin(\alpha) \\ \mathbf{v}_y \\ (\mathbf{v}_z - r) \cos(\alpha) + \mathbf{v}_x \sin(\alpha) + r \end{pmatrix}, \alpha = \frac{\mathbf{v}_x}{2\pi(r - \mathbf{v}_z)}, r = \frac{x_{size}}{\beta} \quad (9.2.1)$$

with a bending angle of  $\beta \neq 0$  (see lower left mesh in Figure 9.2 ) in addition to the affine transformations caused by the bending of preceding Sections.

The number of Sections determines how complex the deformations are that can be described by the parametrization. In Section 9.2.3, the necessary number of Sections required for good results is discussed.

Preceding the deflection, a twisting deformation is performed, a rotation around the x-axis (length) of the object by a value linearly depending on the x-value of the vertex (Fig. 9.2 right) and a twisting strength  $t$ . In addition, 6 degrees of freedom for translation and rotation in 3D are defined. Let  $\beta_1, \beta_2, \dots, \beta_n$  denote the  $n$  deflection parameters,  $t$  denote the object twist,



**Figure 9.2.** An example deformation: the box is separated into one Section without deflection (white) and several Sections of varying curvature (colored). The undeformed box (upper left corner) is subject to varying deformations. In each Section, the applied curvature is constant (lower left). In addition, a global twist of the object can be applied (right)

$x, y, z$  the translation parameters, and  $\gamma_x, \gamma_y, \gamma_z$  the rotation of the object in Euler angles. Then a deformation state can be formulated by the state vector

$$\Theta = (x, y, z, \gamma_x, \gamma_y, \gamma_z, t, \beta_1, \beta_2, \dots, \beta_n). \quad (9.2.2)$$

### 9.2.2 Physical Plausibility Penalty

The reason for choosing Section wise constant curvature deflection, is that this well defined deflection curve can easily be compared to an abstract physical model, by modeling it as a 1D static deformation. This 1D static deformation in an isotropic, linearly elastic material can be modeled by the Euler-Bernoulli model<sup>3</sup>. As the geometry of the tracked object is known, assuming a given Poissons's ratio  $\nu$  and Young's modulus  $E$ , the expected deformation under gravity can be calculated efficiently. The difference in deflection between the guess (given by  $\beta_1, \beta_2, \dots, \beta_n$ ) and the physical model (given by  $\nu$  and  $E$ ) can now be compared directly at discrete points along the deflection curve. But since the relation between  $\nu$  and  $E$  is depending on the mass density of the object, which can't be extracted from visual data, it does not make sense to estimate both parameters simultaneously if no information about the weight

<sup>3</sup>See [FJP<sup>+</sup>12] for a detailed description of the physics modeling in this setup. For an in-depth look at the Euler-Bernoulli model, see [HBW99].

## 9. Specialized Deformation Functions

is available<sup>4</sup>. For this reason, Young's modulus  $E$  will be estimated as the variable stiffness parameter and Poisson's ratio modules will not be considered in the following experiments. Hence, the parameter vector, augmented by  $E$ ,

$$\Theta = (x, y, z, \gamma_x, \gamma_y, \gamma_z, t, E, \beta_1, \beta_2, \dots, \beta_n), \quad (9.2.3)$$

allows to simultaneously estimate the deflection from the input images and the corresponding physical parameter  $\nu$  of the material using the CMA-ES optimization. Let  $y_\beta$  denote the curve function modeled by  $\{\beta_1, \beta_2, \dots, \beta_n\}$ , and let  $y_\nu$ , then the geometric distance  $e_b$  between these two models can be described by

$$e_b(\Theta) = \frac{1}{|S|} \sum_{s \in S} (y_E(s) - y_\beta(s))^2, \quad (9.2.4)$$

$S \subset [0, 1]$  describing the set of points along the object at which the deflection is compared. In conjunction, the depth error  $e_d$  and the color error  $e_c$  introduced in sections 4.2.1 and 4.2.2 and the overall fitness function can (according to (4.2.1)) be written as

$$\mathcal{F}(\Theta) = -(\varphi_d e_d(\Theta) + \varphi_c e_c(\Theta) + \varphi_b e_b(\Theta)), \quad (9.2.5)$$

$\varphi \in \mathbb{R}_{\geq 0}$  denoting the weights of the error terms.

### 9.2.3 Evaluation

To evaluate the algorithm, tests on synthetic data and real data have been performed.

#### Experiments on Synthetic Data

The input data for the synthetic test has been generated by the deformation of cuboid shapes, using a Neo-Hookean deformation model [Riv48] as implemented by the Finite-Element modeling software COMSOL<sup>5</sup> 3.5a in MEMS-mode. The simulated material had a size of  $140 \times 65 \times 10$  mm. To generate the input image, a high resolution triangle mesh (10,000 triangles) was generated from the deflection curve provided by the finite-element software.

---

<sup>4</sup>The weight of an object is not the only variable to link Poisson's ratio and Young's modulus for a known geometry. The mass density can also be acquired by performing e. g., pressure tests. But no visual inspection of the object allows to determine both variables at the same time.

<sup>5</sup>COMSOL Multiphysics, <http://www.comsol.com>



## 9.2. Estimating Physical Parameters

**Table 9.1.** RMS Error values of relevant parameters on an experiment using a low stiffness object ( $E = 0.5 \text{ N/mm}^2$ ).

RMS Pose Error						
Sections	$x$ (mm)	$y$ (mm)	$z$ (mm)	$\gamma_x$	$\gamma_y$	$\gamma_z$
<b>4</b>	0.250	1.386	0.438	$0.372^\circ$	$0.0002^\circ$	$0.575^\circ$
<b>8</b>	0.267	0.086	0.147	$0.056^\circ$	$0.0001^\circ$	$0.036^\circ$
<b>12</b>	0.506	0.118	0.122	$0.076^\circ$	$0.0005^\circ$	$0.052^\circ$

Twist, Offset, Young's Modulus Error			
Sections	$t$	offset(mm)	$E$
<b>4</b>	$0.013^\circ$	5.079	0.060
<b>8</b>	$0.001^\circ$	4.945	0.062
<b>12</b>	$0.002^\circ$	4.695	0.059

This model was rendered using the calibration data of the *Kinect* including lens distortion. A Gaussian distributed noise with a SD of 0.5 mm (1.7% for the object distance of 30 cm) was added to the depth image to simulate the *Kinect* camera noise.

The tests have been performed on three different, homogeneous materials: low stiffness ( $E = 0.5 \text{ N/mm}^2$ ,  $\rho = 1.155 \cdot 10^{-6} \text{ kg/mm}^3$ ), medium stiffness ( $E = 2.0 \text{ N/mm}^2$ ,  $\rho = 1.155 \cdot 10^{-6} \text{ kg/mm}^3$ ), and high stiffness ( $E = 8.0 \text{ N/mm}^2$ ,  $\rho = 1.155 \cdot 10^{-6} \text{ kg/mm}^3$ ). Each of the materials has been tested using a varying number of Sections of constant curvature (see Section 9.2.1), to determine the influence of this hyper parameter. Each test result is the average value of 50 test runs. The following tables show the RMS error from the ground truth data for the relevant components of  $\Theta'$ . The position and orientation of the objects have not been fixed to also test the ability of simultaneous pose estimation.

The test results displayed in tables 9.1 - 9.3 show that the object pose can be retrieved accurately, as well as the material parameters if the object is not too stiff. For very stiff objects, the RMS of the gripper offset becomes larger (and hence Young's modulus  $E$ ), which is reasonable, since for completely stiff objects, this value is undefined. Furthermore, it shows that the quality of the results increases considerable for some values when switching from four sections of constant curvature to eight sections, while for e. g., twelve sections the model tends to over-fit in some cases. Therefore, dividing the model into eight sections is a good compromise between versatility and stability.

## 9. Specialized Deformation Functions

**Table 9.2.** RMS Error values of relevant parameters on an experiment using a low stiffness object ( $E = 2.0 \text{ N/mm}^2$ ).

RMS Pose Error						
Sections	$x$ (mm)	$y$ (mm)	$z$ (mm)	$\gamma_x$	$\gamma_y$	$\gamma_z$
<b>4</b>	0.920	0.127	0.519	$0.092^\circ$	$0.0001^\circ$	$0.052^\circ$
<b>8</b>	0.726	0.110	0.025	$0.078^\circ$	$0.0002^\circ$	$0.044^\circ$
<b>12</b>	0.708	0.063	0.034	$0.061^\circ$	$0.001^\circ$	$0.028^\circ$

Twist, Offset, Young's Modulus Error			
Sections	$t$	offset(mm)	$E$
<b>4</b>	$0.003^\circ$	2.415	0.103
<b>8</b>	$0.002^\circ$	2.638	0.116
<b>12</b>	$0.001^\circ$	2.766	0.121

**Table 9.3.** RMS Error values of relevant parameters on an experiment using a high stiffness object ( $E = 8.0 \text{ N/mm}^2$ ).

RMS Pose Error						
Sections	$x$ (mm)	$y$ (mm)	$z$ (mm)	$\gamma_x$	$\gamma_y$	$\gamma_z$
<b>4</b>	0.979	0.083	0.235	$0.250^\circ$	$0.002^\circ$	$0.030^\circ$
<b>8</b>	0.358	0.115	0.202	$0.143^\circ$	$0.001^\circ$	$0.043^\circ$
<b>12</b>	1.160	0.245	0.170	$0.154^\circ$	$0.004^\circ$	$0.080^\circ$

Twist, Offset, Young's Modulus Error			
Sections	$t$	offset(mm)	$E$
<b>4</b>	$0.004^\circ$	11.70	2.08
<b>8</b>	$0.004^\circ$	11.10	1.98
<b>12</b>	$0.003^\circ$	12.27	2.40

## 9.2. Estimating Physical Parameters

**Table 9.4.** Test results on real objects: standard deviation of parameters for varying object thicknesses.

RMS Pose Error						
Thickness	$x$ (mm)	$y$ (mm)	$z$ (mm)	$\gamma_x$	$\gamma_y$	$\gamma_z$
7 mm	1.304	1.082	0.705	0.509°	0.002°	1.161°
10 mm	1.320	1.722	0.368	0.341°	0.001°	0.707°
13 mm	0.162	1.899	0.074	0.058°	0.001°	0.174°

Twist, Offset, Young's Modulus Error			
Thickness	$t$	offset(mm)	$E$
7 mm	0.009°	2.305	0.040
10 mm	0.006°	1.855	0.029
13 mm	0.001°	0.367	0.007

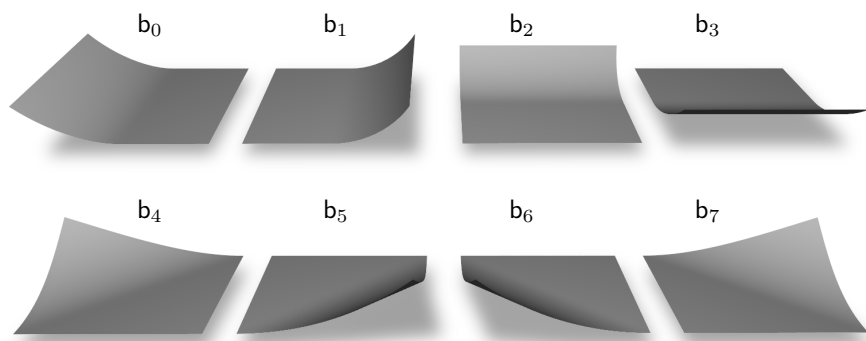
### Experiments on Real Data

The real data experiments have been performed using a *Microsoft Kinect* camera with a resolution of  $640 \times 480$  px in depth and color images. The distance between object and camera has been 40 cm. As a consequence of the results discussed in previous Section, the number of Sections of constant curvature has been set to eight.

To analyze the object material, three test objects have been made from the same material but different thickness, and hence, these objects show varying deflection curves. The material is a Dow Corning Silastic 3481 silicone rubber molded using Silastic 81-R hardener. The mass density for this is given to  $\rho = 1.200 \cdot 10^{-6}$  kg/mm<sup>3</sup>. Reference values for Young's modulus have been estimated by tensile tests using an industrial robot and a PASCO PS-2189 force sensor. The 7 mm object was estimated to a value of  $E = 0.50$  N/mm<sup>2</sup>, while the 10 mm and 13 mm were estimated to  $E = 0.45$  N/mm<sup>2</sup>.

Table 9.4 lists the SD in the optimization results over a sequence of 50 *Kinect* images. The data shows slightly higher error values compared to the synthetic test results. This can be explained by an increased distance between camera and object and the non-Gaussian nature of the noise in the depth values of the *Kinect* camera. Especially the higher deviations in  $\gamma_z$  can be explained by the inaccurate perception of depth discontinuities by the *Kinect* camera. The averaged estimate for the object of 7 mm thickness estimated by the algorithm was  $E = 0.40$  N/mm<sup>2</sup> (0.50 N/mm<sup>2</sup> in the tensile test), for the

## 9. Specialized Deformation Functions



**Figure 9.3.** The FlexPad deformation space contains 8 non-linear bending deformations, parametrized by  $b_0 - b_7$ . Each deformation is bending half an object.

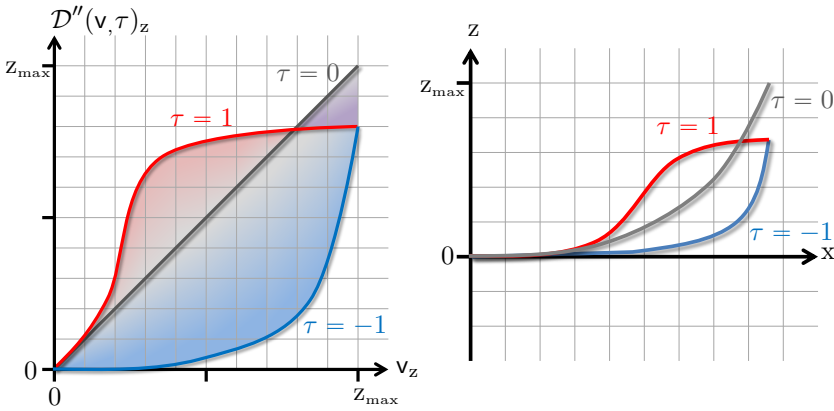
object of 10 mm thickness it was  $E = 0.47 \text{ N/mm}^2$  ( $0.45 \text{ N/mm}^2$  in the tensile test), and for the 13 mm strong object it was  $E = 0.51 \text{ N/mm}^2$  ( $0.45 \text{ N/mm}^2$  in the tensile test). The real data tests show that the actual material parameters have been acquired with an average error of  $E = 0.1 \text{ N/mm}^2$  and below.

### 9.3 Real-Time Paper Tracking

Section 8.3 already discussed the preprocessing steps of the FlexPad system for tracking paper sheets that are deformed by hands. This section will introduce the deformation function of the FlexPad system (first introduced in [SJM13]). One of the crucial aspects of the system is real-time processing, since low frame rates or latency may compromise the interaction experience. For this reason, a special deformation function was designed for FlexPad that enables a low dimension count, leading to efficiency and robustness, while still maintaining the basic deformation set a sheet of paper can undergo.

#### 9.3.1 Paper Deformation Model

The FlexPad deformation model is made out of 3 deformation stages  $\mathcal{D}'$ ,  $\mathcal{D}''$ , and  $\mathcal{D}'''$ , which are applied to the initial paper model. The initial model is given by a rectangular triangle mesh in the x-y-plane at  $z = 0$ . The

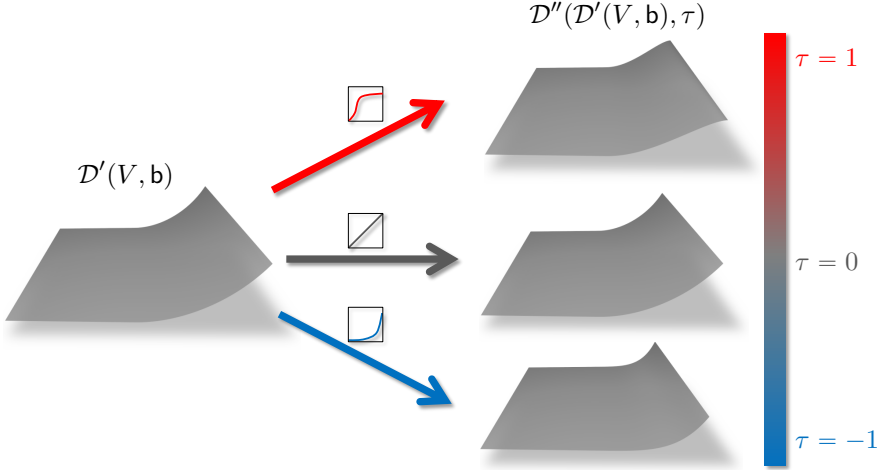


**Figure 9.4.** Left: a graph depicting the positive range of the  $z$ -mapping function. It shows the function that is applied to every  $z$  component  $v_z$  of each vertex  $v$  (after the initial deformation  $\mathcal{D}'$ ), for  $\tau = -1$ ,  $\tau = 0$ , and  $\tau = 1$ . Right: a resulting example shape ( $x$ - $z$ -plane only). Assuming an initial bending strength  $b_1 = 1$ , this graph depicts results as deformed by  $\mathcal{D}''$  for  $\tau = -1$ ,  $\tau = 0$ , and  $\tau = 1$ . Figure 9.5 depicts 3D renderings of these examples.

first deformation stage  $\mathcal{D}'(\cdot, \mathbf{b})$  applies 8 bending transformations to the triangle mesh, parametrized by  $\mathbf{b}_0 - \mathbf{b}_7$ . Each bending deformation affects half the object, such that the area of effect is either given by a separation line perpendicular to one of the edges or by one of the diagonals. Figure 9.3 depicts all 8 initial bendings. The bending transformation is identical to the one introduced in Section 9.2.1. To ensure the fast evaluation, the bending equation is not evaluated during the processing, but a look-up table is generated (5 entries per bending) for each parameter containing pre-evaluated deformations. During the real-time processing, the deformation is performed by linearly interpolating between the nearest pre-evaluated look-up table entries.

The second deformation stage  $\mathcal{D}''$  applies a mapping function to the  $z$ -coordinates (height) of the mesh vertices. It does not change the  $x$  and  $y$  components of the vertices. This stage is only controlled by one parameter  $\tau \in [-1, 1]$  that is changing the shape of the function. If parametrized with 0, the mapping does not change the  $z$ -values (identity function). For values  $< 0$  the mapping becomes parabola-like, for values  $> 0$  it approximates the shape of an 'S'. Figure 9.4 left depicts the  $z$ -mapping function for  $\tau = -1$ ,  $\tau = 0$ , and

## 9. Specialized Deformation Functions



**Figure 9.5.** Illustration of the z-mapping effect on the 3D mesh, already deformed by  $\mathcal{D}'$  with  $\mathbf{b}_1 = 1$ . A high  $\tau$  parameter raises the Section between the bended edge and the object center, while a low  $\tau$  lowers this area and translates the bending deformation to the object edge.

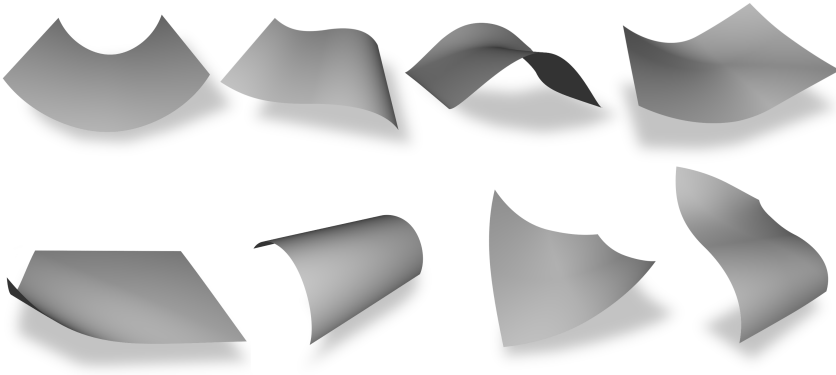
$\tau = 1$ . Figure 9.4 (right) depicts the resulting deformations for the example of  $\mathbf{b}_0 = 0$ ,  $\mathbf{b}_1 = 1$  and  $\mathbf{b}_{>1} = 0$  for the different  $\tau$ . The resulting deformed 3D meshes are depicted in Figure 9.5, illustrating the function of the z-mapping, which is to provide a better adaptability to bendings of the paper. Experiments have shown that this one parameter greatly improves the robustness of the overall tracking, as it prevents the optimizer from compensating the simplicity of the bending model by adjusting the global pose of the object.

The third transformation  $\mathcal{D}'''$  applied to the object is a rigid transformation as defined in Section 9.1, i. e., translation in x-, y-, and z-direction as well as a rotation around all three axes. The final deformation is then given by

$$\mathcal{D} = \mathcal{D}' \circ \mathcal{D}'' \circ \mathcal{D}''' . \quad (9.3.1)$$

A parameter vector in the 15 dimensional parameter space of  $\mathcal{D}$  is given by

$$\Theta = (\mathbf{b}_0, \dots, \mathbf{b}_7, \tau, \mathbf{t}_x, \mathbf{t}_y, \mathbf{t}_z, \mathbf{r}_x, \mathbf{r}_y, \mathbf{r}_z), \quad (9.3.2)$$



**Figure 9.6.** Example deformations visualizing the versatility of the 9 + 6 parameter FlexPad deformation model.

with  $\mathbf{t}$  representing the translation of the rigid transformation and  $\mathbf{r}$  the rotation in Euler angles  $\mathbf{R}_x(r_x)$ ,  $\mathbf{R}_y(r_y)$ , and  $\mathbf{R}_z(r_z)$ .









### 9.3.2 Evaluation

The FlexPad tracking system is compiled using the depth image of the *Kinect* mounted above the user (see Section 8.3 for a description of the FlexPad hardware setup), with the preprocessing step described in Section 8.3 and sparse synthesis (Section 4.4.1) with  $25 \times 25$  vertices. The number of CMA-ES iterations varies between 50 (at 25 fps on an up-to-date desktop computer) and 200 iterations (at eight fps). The number of iterations is determined by the current movement of the tracked object. For fast movements, the iteration count is lowered to keep track of the object, while for slow movements, the iteration count is increased to add accuracy. Table 9.5 contains a set of eight representative deformations that have been tracked. To evaluate the tracking data, the RMS difference between each synthesized object pixel and the actual depth measurement is given for all of these deformations at two different object-to-sensor distances.

**Application Experiments** The tracking system introduced in this Section was designed for FlexPad, an interactive system that uses flexible sheet-like objects (paper, foam, etc.) to deliver the experience of having a flexible display

## 9. Specialized Deformation Functions

**Table 9.5.** Pixel-wise RMS difference (error) between *Kinect* measurement and synthesized model for different shapes acquired at 90 cm and 150 cm object-to-sensor distance. The standard deviation of each error value is given in brackets.

Shape				
Sensor Distance				
<b>90 cm</b>	2.10 mm (1.1)	1.91 mm (1.2)	2.67 mm (1.6)	4.58 mm (1.9)
<b>150 cm</b>	2.64 mm (1.3)	3.07 mm (1.7)	6.10 mm (4.2)	5.45 mm (2.7)
Shape				
Sensor Distance				
<b>90 cm</b>	4.82 mm (2.2)	4.93 mm (2.1)	5.39 mm (2.2)	2.41 mm (1.2)
<b>150 cm</b>	5.15 mm (2.5)	6.38 mm (3.8)	7.30 mm (4.1)	4.56 mm (2.3)

by projecting content onto the flexible sheet using a video projector. The real-time tracking allows for an intuitive, easy to use, interactive display experience. The spatial awareness of the display allows for a variety of input and visualization applications. Please refer to [SJM13] for a detailed description of the FlexPad user studies.



# Conclusion

In this thesis, the tracking of deforming objects and the reconstruction of that deformation has been addressed. In contrast to rigid movements, which can be analyzed and reconstructed very well, general deformations can come with an infinite number of sub-movements and ways to parametrize them, which makes it very difficult to formulate tracking goals. In contrast to the classic reconstructions based on color data alone, the combination of depth and color video used in this thesis provides a data foundation with less room for ambiguities. The AbS approach is able to exploit the complete information out of the various entities while being able to regard sensor specific benefits and shortcomings.

## 10.1 Traits of AbS

AbS differs in many key aspects from common reconstruction schemes, giving it a unique set of advantages compared to feature-based approaches or to locally optimizing geometry registration methods.

The main advantages are:

- ▷ No two-stage processing is required. The problem is solved directly on the data without potentially error-prone preprocessing like feature detection.
- ▷ AbS uses the complete image information during the optimization process. No image parts are disregarded due to lack of texture or structure.
- ▷ Data from different domains like color-, depth-, or IR-images can easily be used simultaneously without complex fusion strategies. A heuristic weight

## 10. Conclusion

or a noise model of each sensor is sufficient to determine a common, optimal solution.

- ▷ A priori knowledge is applied in a straight forward manner, allowing easy compilation and adjustment of any AbS-based tracking or reconstruction. Known physical limitations of the environment or the object can be formulated directly to confine the solution to valid regions.
- ▷ Since image generation effects only have to be synthesized instead of being analyzed and removed, the modeling of “non-invertible” effects is possible and intuitive. Restriction to Bayer pattern, cross talk between pixels, image blur, and various other artifacts can easily be generated artificially even when an analysis or complete removal is impossible. Hence, approaches purely based on analysis might not be able to model these effects, while AbS can regard these effects without the need for complex reconstruction.

However, one has to be aware of the restrictions, which (yet) limit the area of application of AbS in computer vision:

- ▷ The reconstruction of scenes solely based on color data is not feasible, because the problem dimensions are very high. CMA-ES does provide very good results for dimension counts below 100, and works reasonably well on below 1,000 dimensions. But reconstructing for example a  $1024 \times 768$  depth map is an optimization problem with more than 700,000 dimensions, a number that can not be handled well by CMA-ES.
- ▷ The AbS approach is computationally expensive compared to feature-based approaches. In situations where computational power is valuable and the tracking or reconstruction problem can successfully be performed by feature-based methods, there is no need to apply AbS.
- ▷ The AbS system as introduced in this thesis lacks a generic deformation model that can be applied to unknown scenes. Scene-flow based systems or feature driven approaches can perform spatial interpolation between detected movements. AbS has to first propose a transformation along with the parametrization before it can be validated, making the retrieval of the scene deformation difficult without prior knowledge.

## 10.2 Future Work

Direct deformation tracking in its various implementations discussed in this thesis is a very powerful tool to perform reconstructions, even for complex

geometries and subject to real-time requirements. However, if the two main limitations of the method, the general limitation of global optimization and the lack of a missing generic, adaptable deformation model, can be overcome, it is possible to push the approach even further towards autonomous scene understanding from depth and color video.

### **Improving Optimizer Performance**

Achieving a higher performance and scalability with CMA-ES has been investigated by several work groups in recent years. Despite many insights gained from the detailed analyses and a large number of proposed advancements regarding CMA-ES, the overall performance increase is limited to certain scenarios and function types or entails a disproportional increase of required computation.

A promising approach to increase the Abs potential nevertheless is to use a posterior knowledge to bootstrap the optimization. Analyzing the image movement might allow to intensify the search in regions of possible solutions and to disregard search directions to which there are no hints in the image data. If the search distribution can be conventionalized accordingly, the dimension count could be increased significantly without the need to stray from a completely direct formulation of the fitness function.

### **Generic Deformation**

The NURBS-based deformation as it is described in this thesis allows to track a large number of deformations, but clearly has limitations. The separable distribution of control points in the parameter domain significantly reduces the adaptability to local resolution. The limitation to a 2D topology also limits the application areas. To achieve complete scene understanding, a volumetric transformation or a topology-independent deformation is required.

### **Model Learning**

The last missing element to autonomous, non-rigid scene reconstruction is the accumulation of the scene geometry throughout the sequence. Rigid and moving objects alike usually show only a subset of their geometry in each sensor. A system able to accumulate geometry information and to distinguish between

## 10. Conclusion

object structure and deformation will allow the complete reconstruction of scene sequences.

The goals listed in this future work section are subject to the DFG<sup>1</sup> funded research program “Reconstruction of complex deformations in 3D scenes from depth- and color data” and will be investigated in the near future.

---

<sup>1</sup>DFG stands for “Deutsche Forschungsgemeinschaft”, i. e., German Research Foundation

## Appendix A

# CMA-ES

Combining the elements introduced in 5.1.1 and 5.1.2 yields the update instructions as they are used in the implementation for this thesis. Given the  $n$ -dimensional fitness function  $\mathcal{F}$ , a start guess  $\mathbf{m}_0$ , a distribution scale  $\sigma_0$ , and a population size  $\lambda$ , CMA-ES performs the following steps:

### A.1 Initialization

The following definitions are taken from [Han06], except for the definitions of  $q, c_e, c_{\hat{e}}, c_1$ , and  $c_\mu$  which are taken from the example implementation of CMA-ES by Nicolaus Hansen available at <https://www.lri.fr/~hansen/>. The Weight vector  $\mathbf{w}$  is set to

$$\forall j \in \mathbb{N}_{<\lambda} : \mathbf{w}_j = \begin{cases} 2 \frac{\mu-j+1}{\mu^2+\mu} & j > \lambda - \mu \\ 0 & \text{else.} \end{cases} \quad (\text{A.1.1})$$

The variance effective selection mass is set to

$$\mu_{\text{eff}} = \|\mathbf{w}\|_2^{-2} = \left( \frac{\|\mathbf{w}\|_1}{\|\mathbf{w}\|_2} \right)^2. \quad (\text{A.1.2})$$

The weight of the rank-one update is set to

$$c_1 = \frac{2}{(n + 1.3)^2 + \mu_{\text{eff}}}. \quad (\text{A.1.3})$$

## A. CMA-ES

The weight of the rank- $\mu$  update is set to

$$c_\mu = \min \left( 1 - c_1, 2 \frac{\frac{\mu_{\text{eff}} - 2 + 1}{\mu_{\text{eff}}}}{(n + 2)^2 + \mu_{\text{eff}}} \right). \quad (\text{A.1.4})$$

The weight of the isotropic evolution path update is set to

$$c_e = \frac{4 + \frac{\mu_{\text{eff}}}{n}}{n + 4 + 2 \frac{\mu_{\text{eff}}}{n}}. \quad (\text{A.1.5})$$

The weight of the anisotropic evolution path update is set to

$$c_{\hat{e}} = \frac{n + 2}{n + \mu_{\text{eff}} + 5}. \quad (\text{A.1.6})$$

The damping factor for the step size update is set to

$$q = 1 + 2 \max \left( 0, \sqrt{\frac{\mu_{\text{eff}} - 1}{n + 1}} - 1 + c_{\hat{e}} \right). \quad (\text{A.1.7})$$

$\mathbf{E} (\|\mathcal{N}(\mathbf{0}^n, \mathbf{1}^{n \times n})\|_2)$ , the expected length of a normally distributed random vector, is approximated via

$$\chi_n = \sqrt{n} \left( 1 - \frac{1}{4n} + \frac{1}{21n^2} \right). \quad (\text{A.1.8})$$

The initial normalized covariance set to the identity matrix

$$\mathbf{C}_0 = \mathbf{1}. \quad (\text{A.1.9})$$

The isotropic evolution path vector set to the zero vector

$$\mathbf{e}_0 = \mathbf{0}. \quad (\text{A.1.10})$$

The anisotropic evolution path vector set to the zero vector

$$\hat{\mathbf{e}}_0 = \mathbf{0}. \quad (\text{A.1.11})$$

## A.2 Update

The following steps have to be performed in each iteration  $i$  of the algorithm: Distribute  $\Theta_0, \dots, \Theta_{\lambda-1}$  according to  $\mathcal{N}(\mathbf{m}_i, \sigma_i \mathbf{C}_i)$  and sort them such that

$$\mathcal{F}(\Theta_0) \leq \mathcal{F}(\Theta_1) \leq \dots \leq \mathcal{F}(\Theta_{\lambda-1}). \quad (\text{A.2.1})$$

Set the current optimum to the best evaluated parameter ever

$$\Theta^* \rightarrow \arg \max_{\Theta \in \{\Theta^*, \Theta_0, \Theta_1, \dots, \Theta_{\lambda-1}\}} \mathcal{F}(\Theta). \quad (\text{A.2.2})$$

Set the new distribution mean to

$$\mathbf{m}_{i+1} = \sum_{j=0}^{\lambda-1} w_j \Theta_j. \quad (\text{A.2.3})$$

Set the isotropic evolution path vector to

$$\mathbf{e}_{i+1} = (1 - c_e) \mathbf{e}_i + \sqrt{c_e(2 - c_e) \mu_{\text{eff}}} \frac{\mathbf{m}_{i+1} - \mathbf{m}_i}{\sigma_i}. \quad (\text{A.2.4})$$

Set the anisotropic evolution path vector to

$$\hat{\mathbf{e}}_{i+1} = (1 - c_{\hat{e}}) \hat{\mathbf{e}}_i + \sqrt{c_{\hat{e}}(2 - c_{\hat{e}}) \mu_{\text{eff}}} \mathbf{C}_i^{-\frac{1}{2}} \frac{\mathbf{m}_{i+1} - \mathbf{m}_i}{\sigma_i}. \quad (\text{A.2.5})$$

Set the covariance matrix to

$$\mathbf{C}_{i+1} = (1 - c_\mu - c_1) \mathbf{C}_i + c_1 \mathbf{e}_{i+1} \mathbf{e}_{i+1}^T + c_\mu \sum_{j=0}^{\lambda-1} w_j (\Theta_j - \mathbf{m}_i) (\Theta_j - \mathbf{m}_i)^T. \quad (\text{A.2.6})$$

Set the step size (covariance scale) to

$$\sigma_{i+1} = \sigma_i \exp \left( \frac{c_{\hat{e}}}{q} \left( \frac{\|\hat{\mathbf{e}}_{i+1}\|}{\chi_n} - 1 \right) \right). \quad (\text{A.2.7})$$

### A.3 Bootstrapped Initialization

In this subsection, the CMA-ES components will be denoted with an additional superscript to indicate the frame. The following bootstrapping describes the initialization of frame  $f + 1$  after the complete optimization of frame  $f$  has been processed. For a sequence of input images, each optimization problem (except for the first frame) can be initialized using parameter space knowledge from the preceding optimization path (see Section 5.2.1).

Set the reference iteration  $i^*$  of frame  $f$  to the last frame of the optimization with a correlated movement:

$$i^* = \max\{i : \|\hat{\mathbf{e}}_i^f\| > \chi_n\} \quad (\text{A.3.1})$$

Set the new distribution mean to the optimum of the preceding frame

$$\mathbf{m}_0^{f+1} = \Theta^{*f}. \quad (\text{A.3.2})$$

Set the step size (covariance scale) to the calculated step size of the recent frame, regarding the minimum step size  $\theta$  to avoid degeneration

$$\sigma_0^{f+1} = \max\{\theta, \|\mathbf{m}_0^f - \Theta^{*f}\|_2\}. \quad (\text{A.3.3})$$

Perform an SVD<sup>1</sup> of  $\mathbf{C}_{i^*}^f$ , such that  $\mathbf{C}_{i^*}^f = U \text{diag}(\xi) U^T$ . Define the robustified diagonal entries (eigenvalues) via

$$\xi'_n = \max\{\epsilon, \xi_n\} \forall n \quad (\text{A.3.4})$$

and initialize the distribution with the normalized<sup>2</sup> robustified distribution of the reference frame

$$\mathbf{C}_0^{f+1} = \frac{U \text{diag}(\xi') U^T}{\det(U \text{diag}(\xi') U^T)}. \quad (\text{A.3.5})$$

Set the isotropic evolution path to its value at the reference iteration of the most recent frame

$$\mathbf{e}_0^{f+1} = \mathbf{e}_{i^*}^f. \quad (\text{A.3.6})$$

---

<sup>1</sup>This decomposition does not have to be calculated especially for this bootstrapping, as it is already required in each iteration to generate the normally distributed random variables with respect to  $\mathcal{N}(\mathbf{m}_i, \sigma_i \mathbf{C}_i)$ . Since  $\mathbf{C}$  is real, symmetric, and positive semi-definite, the general term  $U \text{diag}(\xi) V^*$  can be written as  $U \text{diag}(\xi) U^T$ .

<sup>2</sup>The normalization by the new matrix scale (determinant) can be neglected in practice, as the impact of the robustification on the overall scale is usually very small.



### A.3. Bootstrapped Initialization

Set the anisotropic evolution path to its value at the reference iteration of the most recent frame

$$\hat{\mathbf{e}}_0^{f+1} = \hat{\mathbf{e}}_{i^*}^f. \quad (\text{A.3.7})$$



# Brightness Normalization of the FlexPad System

The color (or intensity) information in each pixel of a camera image is subject to many influences: the material of the projected surface, its geometry, as well as various light sources and more subtle physical effects play into the value of each pixel in the image. The fact that a large set of influences are condensed into one value makes it difficult or even impossible to reconstruct these various parameters from the measurement of a pixel.

But in an IR image of a *Kinect* camera, many of these influences are either constant or can be estimated. The following Section introduces a way to remove the effects of changes in surface distance and orientation.

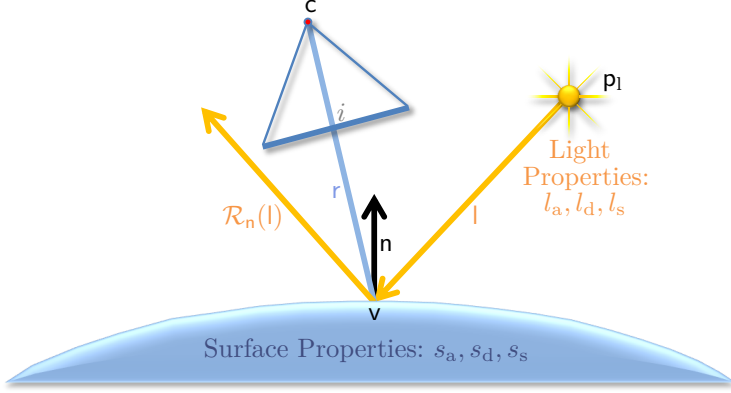
## B.1 Surface Normal Compensation

The Phong lighting model [Pho75] is the standard model in computer graphics to describe the general reflection behavior of light on surfaces. It groups reflection into three categories: ambient, diffuse, and specular reflections. Hence, a material is (locally) defined by its ambient  $s_a$ , diffuse  $s_d$ , and specular  $s_s$  reflectivity. The visible color of a surface point in the eye of a viewer is given by the sum of the ambient, diffuse, and specular reflections of all lights present. The model is extended straight forward to multiple color channels, but in the case of the IR image, the monochrome model is sufficient.

Let  $l$  denote the direction of the incoming light at the surface,  $n$  the surface normal,  $r$  the vector pointing from the surface to the viewer, and

$$\mathcal{R}_n(l) = 2(l \cdot n)n - l \tag{B.1.1}$$

## B. Brightness Normalization of the FlexPad System



**Figure B.1.** Pictogram of the components involved in the Phong lighting model.

the direction of the light, when directly reflected on the surface. The Phong illumination  $i$  is given by the equation:

$$i = s_a l_a + \sum_{(l, l_d, l_s) \in L} (s_d (l \cdot n) l_d + s_s (\mathcal{R}_n(l) \cdot r)^\alpha l_s), \quad (\text{B.1.2})$$

with  $\alpha \in \mathbb{R}_{\geq 1}$  denoting a parameter to adjust the specular reflection and  $L$  being the set of lights. For a surface point at position  $v \in \mathbb{R}^3$ , a camera at  $c \in \mathbb{R}^3$ , and a light at  $p_l \in \mathbb{R}^3$ , this translates to (see Fig. B.1)

$$s_a l_a + \sum_{(p_l, l_d, l_s) \in L} \left( s_d \left( \frac{v - p_l}{|v - p_l|} \cdot n \right) l_d + s_s \left( \mathcal{R}_n \left( \frac{v - p_l}{|v - p_l|} \right) \cdot \left( \frac{c - v}{|c - v|} \right) \right)^\alpha l_s \right). \quad (\text{B.1.3})$$

In the case of a *Kinect* IR image, this equation can be simplified by customizing it to the setup at hand:

- ▷  $l_a = 0$ . The amount of indoor, IR light is neglectable in general, as there usually are no near IR light sources and windows tend to block most of the IR light.
- ▷  $|L| = 1$ . The only light source in the IR domain is the *Kinect* projector.

## B.2. Distance Compensation

- ▷  $s_s = 0$ . The materials observed are skin, paper and cloth (clothing). All of these materials have nearly zero specular reflection properties, which allows to neglect the specular components.
- ▷  $\mathbf{p}_1 = \mathbf{c}$ . The *Kinect* projector position equals the camera position. Physically, these positions are 7 cm apart, which is nearly the same position in the scale of the complete setup.

This allows to formulate a simpler version of (B.1.3):

$$i = s_d \left( \frac{\mathbf{v} - \mathbf{p}_1}{|\mathbf{v} - \mathbf{p}_1|} \cdot \mathbf{n} \right) l_d \quad (\text{B.1.4})$$

which in the camera coordinate system ( $\mathbf{p}_1 = \mathbf{c} = \mathbf{O}$ ) is equivalent to

$$i = s_d (\mathbf{v} |\mathbf{v}|^{-1} \cdot \mathbf{n}) l_d. \quad (\text{B.1.5})$$

To remove the effect, the surface normal has on the reflection towards the camera center,  $i$  is simply multiplied by the inverse term that is depending on the surface normal  $\mathbf{n}$  and the surface point position  $\mathbf{v}$ :

$$i_{\text{normal\_ori}} = s_d l_d = \frac{i}{\mathbf{v} |\mathbf{v}|^{-1} \cdot \mathbf{n}}. \quad (\text{B.1.6})$$

$i_{\text{normal\_ori}}$  can be evaluated directly from the measurements available,  $i$  corresponds to the measured IR intensity,  $\mathbf{v}$  is retrieved from the depth image, as well as  $\mathbf{n}$  is calculated from the local vicinity in the depth image.

## B.2 Distance Compensation

To compensate for all positional effects on the measured intensity, it is useful to take a look at the way the light covers: it is emitted by the projector, travels through air, is reflected by the surface, and travels to the camera, where its intensity is measured. The projector can be assumed to provide constant illumination, but the way from the projector to the object has to be regarded. The reflection is discussed in Section B.1, and the way to the camera receives has to be regarded. The camera itself is assumed to have a linear relationship between incoming light and the measured brightness value.

## B. Brightness Normalization of the FlexPad System

If a light source at position  $\mathbf{p}_1$  emits light  $i_e$  into 3D space, the light intensity  $i_o$  at any light receiving point  $\mathbf{p}_o$  is attenuated by the squared distance:

$$i_o = \frac{i_e}{|\mathbf{p}_1 - \mathbf{p}_o|^2}. \quad (\text{B.2.1})$$

This equation<sup>1</sup> can easily be deduced from the energy conservation of light, e. g., by setting the incoming light on any sphere around  $\mathbf{p}_1$  to receive a constant amount of light, independent of its radius.

The way from the object to the camera has no measurable effect on the intensity. This is due to the fact that the surface area each camera pixel covers grows with distance in the same ratio as the intensity of a constant surface area decreases with distance. Hence, the effect modeled in B.2.1 is the only one induced by a change of distance.

The complete compensation is given by combining (B.1.6) and (B.2.1) with  $\mathbf{p}_1 = \mathbf{c} = \mathbf{O}$ :

$$i_{\text{normalized}} = \frac{i|\mathbf{v}|^2}{|\mathbf{v}|^{-1}(\mathbf{v} \cdot \mathbf{n})} = \frac{i|\mathbf{v}|^3}{\mathbf{v} \cdot \mathbf{n}}. \quad (\text{B.2.2})$$

---

<sup>1</sup>The singularity in (B.2.1) for  $\mathbf{p}_1 = \mathbf{p}_o$  is caused by the simplification of reducing areas to points in 3D space. Points that are infinitely small can never emit any measurable amount of light. The resulting attenuation is correct, nevertheless.

# Effects of Random Sparsification

In this Section, the probability of the random sparsification influencing the optimization process is analyzed. As discussed in Section 5.1.1, this happens when the noise introduced by the sparsification leads to a change in order of the  $\mu$  selected individuals and the selection itself. Starting with a simplified scenario, it is assumed that  $\mathcal{F}$  is linear and the population size  $\lambda = 2$ .

The normal distribution of CMA-ES in the parameter space is given by the mean value  $\mathbf{m}$  and covariance  $\sigma\mathbf{C}$ . Let  $\Theta_0$  and  $\Theta_1$  be these two individuals, then

$$\Theta_0, \Theta_1 \sim \mathcal{N}(\mathbf{m}, \sigma\mathbf{C}). \tag{C.0.1}$$

The evaluated values are given by

$$\mathcal{F}(\Theta_0), \mathcal{F}(\Theta_1) \sim \mathcal{F}(\mathcal{N}(\mathbf{m}, \sigma\mathbf{C})) = \mathcal{N}(\mathcal{F}(\mathbf{m}), \sigma\mathcal{F}(\mathbf{C})). \tag{C.0.2}$$

Random sparsification introduces noise to the fitness function that can be assumed to have an expected offset of 0 (see 4.4.2 for the motivation of this assumption). The probability of  $\Theta_0$  and  $\Theta_1$  changing places in the order induced by  $\mathcal{F}$  when evaluated by the sparsified fitness  $\mathcal{F}_s$  instead is given by

$$\mathbf{P}(\mathcal{F}(\Theta_0) > \mathcal{F}(\Theta_1), \mathcal{F}_s(\Theta_0) < \mathcal{F}_s(\Theta_1)) \tag{C.0.3}$$

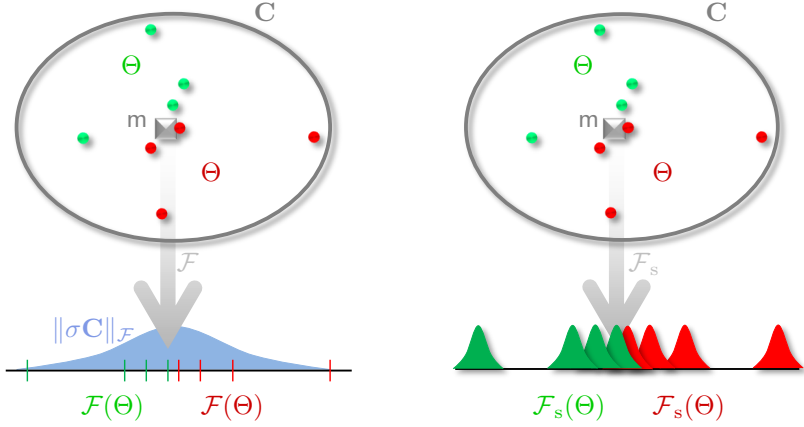
$$+\mathbf{P}(\mathcal{F}(\Theta_0) < \mathcal{F}(\Theta_1), \mathcal{F}_s(\Theta_0) > \mathcal{F}_s(\Theta_1)) \tag{C.0.4}$$

which equals

$$\mathbf{P}(\mathcal{F}(\Theta_0) > \mathcal{F}(\Theta_1) | \mathcal{F}_s(\Theta_0) < \mathcal{F}_s(\Theta_1)) \tag{C.0.5}$$

due to the symmetric nature of the distribution.

### C. Effects of Random Sparsification



**Figure C.1.** Left: the fitness function maps all distributed individuals (green and red dots) to their fitness value (green and red lines). The distribution within the codomain of  $\mathcal{F}$  is notated as  $\|\sigma\mathbf{C}\|_{\mathcal{F}}$ . Right: the fitness function using sparsification  $\mathcal{F}_s$  adds an additional uncertainty to the fitness values  $\mathcal{F}_s(\Theta)$ , which can change the order of individuals and, hence, their weighting.

The distance in the fitness domain between the evaluated  $\Theta_0$  and  $\Theta_1$  can be notated following the addition rule for normally distributed random variables:

$$d = \mathcal{F}(\Theta_0) - \mathcal{F}(\Theta_1) \sim \mathcal{N}(0, 2(\|\sigma\mathbf{C}\|_{\mathcal{F}})) \quad (\text{C.0.6})$$

with  $\|\cdot\|_{\mathcal{F}}$  denoting the norm<sup>1</sup> induced by  $\mathcal{F}$ , and accordingly

$$d_s = \mathcal{F}_s(\Theta_0) - \mathcal{F}_s(\Theta_1). \quad (\text{C.0.7})$$

The probability of  $\Theta_0$  and  $\Theta_1$  changing places in the order because of sparsification noise is the same as  $d$  and  $d_s$  having different signs, i. e.,

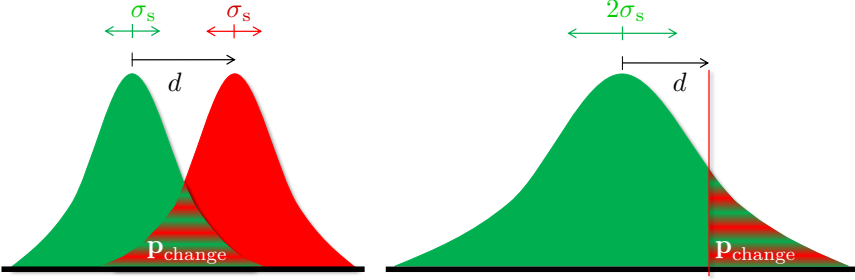
$$\mathbf{P}(\mathcal{F}(\Theta_0) > \mathcal{F}(\Theta_1) | \mathcal{F}_s(\Theta_0) < \mathcal{F}_s(\Theta_1)) = \mathbf{P}(\text{sign}(d_s) \neq \text{sign}(d)) \quad (\text{C.0.8})$$

Let it be assumed that this sparsification noise is normally distributed as well, with a deviation given by  $\sigma_s$ :

$$\mathcal{F}_s(\Theta_0) \sim \mathcal{N}(\mathcal{F}(\Theta_0), \sigma_s) \quad (\text{C.0.9})$$

<sup>1</sup>The  $\mathcal{F}$  induced norm on a matrix  $\|\mathbf{C}\|_{\mathcal{F}}$  is in this case given by the operator norm of  $\|\mathcal{F}(\mathbf{C})\|_{\text{op}}$  with  $\mathcal{F}(\mathbf{C})$  denoting the vector wise application of  $\mathcal{F}$  to  $\mathbf{C}$ .





**Figure C.2.** Left: two distributions of sparsification noise  $\sigma_s$  and the difference  $d$  between the expected values. The distribution overlap visualizes the events in which the sparsification noise causes a change in order, with its probability  $\mathbf{p}_{\text{change}}$  (Note:  $\mathbf{p}_{\text{change}}$  is plotted for intuitive visualization only and is not the numerically correct multiplication of the PDFs) Right: the same probability value can be calculated by reducing one distribution to a single result (red) and adding the subtracted variance to the other distribution (green).

and

$$\mathcal{F}_s(\Theta_1) \sim \mathcal{N}(\mathcal{F}(\Theta_1), \sigma_s). \quad (\text{C.0.10})$$

It is possible to rewrite  $d_s$  as

$$d_s = d + \Delta_s \quad (\text{C.0.11})$$

with

$$\Delta_s \sim \mathcal{N}(0, 2\sigma_s), \quad (\text{C.0.12})$$

which allows to formulate the implication that

$$\begin{aligned} \mathbf{P}(\text{sign}(d_s) \neq \text{sign}(d)) &= \mathbf{P}(|\Delta_s| > |d|) \mathbf{P}(\text{sign}(\Delta_s) \neq \text{sign}(d)) \\ &= \frac{1}{2} \mathbf{P}\left(\frac{|\Delta_s|}{|d|} > 1\right). \end{aligned}$$

Using the distributions derived in (C.0.6) and (C.0.12), this equals

$$\mathbf{p}_{\text{change}} = \frac{1}{2} \int_1^\infty \mathcal{N}\left(0, \frac{\sigma_s}{\|\sigma \mathbf{C}\|_{\mathcal{F}}}\right). \quad (\text{C.0.13})$$

### C. Effects of Random Sparsification

So, if e. g., a 3% error rate is required, i. e., only 3% of the iterations should be affected by the sparsification at all, this is met by choosing  $\sigma_s$  according to the  $2\sigma$ -bound of the distribution at hand<sup>2</sup>:

$$1 = 2 \frac{\sigma_s}{\|\sigma\mathbf{C}\|_{\mathcal{F}}}, \quad (\text{C.0.14})$$

which implies

$$\sigma_s = \frac{\|\sigma\mathbf{C}\|_{\mathcal{F}}}{2}. \quad (\text{C.0.15})$$

The fitness function  $\mathcal{F}$  was assumed to be linear, which obviously is not given in a real application scenario. But, in case of a real application, the term  $\|\sigma\mathbf{C}\|_{\mathcal{F}}$  is easily estimated.  $\sigma\mathbf{C}$  being the current distribution of individuals, the term  $\|\sigma\mathbf{C}\|_{\mathcal{F}}$  describes the range (deviation) of fitness values, evaluated by the current<sup>3</sup> distribution, which is the variance in the results of the current generation of individuals:

$$\|\sigma\mathbf{C}\|_{\mathcal{F}} \approx \sqrt{\sum_{i=0}^{\lambda-1} \frac{(\mathcal{F}(\Theta_i) - m)^2}{\lambda}}, m = \sum_{i=0}^{\lambda-1} \frac{\mathcal{F}(\Theta_i)}{\lambda}. \quad (\text{C.0.16})$$

This evaluation serves as an approximation for an unscented transform [JJU04], as it allows to propagate  $\sigma\mathbf{C}$  through  $\mathcal{F}$ . It works without any additional computational efforts by using the already evaluated individuals  $\mathcal{F}(\Theta_i)$ , and since the codomain of  $\mathcal{F}$  is one-dimensional, only few individuals are required.

So for a given  $\sigma$ -bound, an approximation can be given on how to choose  $\sigma_s$  to satisfy the error rate. Note that the sparsification noise  $\sigma_s$  is not set directly but depends on the complexity of the reference object and the amount of sparsification. A simple shape that is represented by many vertices will allow a much higher rate of sparsification than a complex shape represented by few vertices for the same error rate. This entails, that the association between sparsification rate and  $\sigma_s$  has to be determined by testing different sparsification rates and cannot be derived mathematically.

For more than 2 individuals, the likelihood of a change in the order increases, as the probability  $p_{\text{change}}$  (C.0.13) has to be considered for every

<sup>2</sup> The  $2\sigma$ -bound delivers a probability of  $\approx 4.4\%$  (error rate) which is multiplied by  $\frac{1}{2}$  according to (C.0.13).

<sup>3</sup>In the linearized case, there is no real “current”, as all derivatives of the fitness function are constant. For a real fitness function, the area around the current mean should be regarded.

pair of individuals. So having an arbitrary population size  $\lambda$ , there are  $\sum_{i=1}^{\lambda-1} i$  pairs in each generation, but the order in between individuals that are not selected is irrelevant, which are  $\sum_{i=1}^{\mu-1} i$ . So for arbitrary population sizes, this yields a chance of

$$\left( \sum_{i=1}^{\lambda-1} i - \sum_{i=1}^{\mu-1} i \right) p_{\text{change}} = \sum_{i=\mu}^{\lambda-1} \frac{i}{2} \int_1^{\infty} \mathcal{N}\left(0, \frac{\sigma_s}{\|\sigma\mathbf{C}\|_{\mathcal{F}}}\right) \quad (\text{C.0.17})$$

## Evaluation

As an example: for a population of 8 individuals, the chance (C.0.13) increases by a factor of  $28 - 6 = 22$  (C.0.17). To keep the error rate below the requested threshold of 3%, the sparsification noise has to be decreased such that it satisfies the  $3\sigma$  bound, i. e.,

$$\sigma_s = \frac{\|\sigma\mathbf{C}\|_{\mathcal{F}}}{3}. \quad (\text{C.0.18})$$

As the  $3\sigma$  bound has an error below 0.2%, the overall probability of a position change is  $22 \cdot \frac{1}{2} \cdot 0.2 = 2.2\%$ .  $\|\sigma\mathbf{C}\|_{\mathcal{F}}$  is given by evaluating (C.0.16).

Instead of estimating upper  $\sigma$ -bounds, one could formulate the direct connection between the sparsification noise  $\sigma_s$ , the number of individuals, and the current distribution  $\|\sigma\mathbf{C}\|_{\mathcal{F}}$ , but this would require an evaluation of (C.0.13), which is not feasible due to the nature of the cumulative distribution function (CDF) of the Gaussian distribution. Experimental results of the random sparsification can be found in Section 7.4.



# Glossary

<b>0</b>	the zero vector or function	33, 45, 59–61, 91, 92, 142, 147, 148
<b>1</b>	the identity function, matrix or element	45, 61, 91, 142
<b>A</b>	$\in \mathbb{R}^{4 \times 4}$ , an affine transformation (translation and rotation) of homogenous coordinates in the $\mathbb{P}^3$	32–35, 45, 46
<b>B</b>	$: ([0, 1]^2, \mathfrak{P}(\mathbb{R}^3)) \rightarrow \mathbb{R}^3$ , NURBS function, taking a (two dimensional) parameter and a set of control points as arguments	18, 21–26, 40, 41
<b>b</b>	$\in [-1, 1]$ , vector of indexed bending parameter ( $\mathbf{b}_0 - \mathbf{b}_7$ ) of the FlexPad deformation function	119, 120
<b><math>\mathcal{B}^\perp</math></b>	$: ([0, 1]^2, \mathfrak{P}(\mathbb{R}^3)) \rightarrow \mathbb{R}^3$ , function yielding the normal vector to the surface described by $\mathcal{B}(\cdot, P)$	21, 25
<b>C</b>	$\subset \mathbb{N}_{<256}^3$ , indexed set (analog to $V$ ) containing vertex colors	47
<b>C</b>	$\in \mathbb{R}^{n \times n}$ , the covariance matrix of a Gaussian distribution	52, 53, 56–60, 62, 65–67, 87, 92, 110, 142–144, 149–153

## Glossary

$\mathbf{c}$	$\in \mathbb{R}^3$ , the optical center of a camera as a 3D position	33, 45, 47, 129, 131, 132, 146–148
$c$	$\in \mathbb{R}_{\geq 0}^2 \rightarrow \mathbb{R}_{\geq 0}$ , noise compensation function	92
$c_{\hat{\mathbf{e}}}$	$\in (0, 1]$ the learning rate for the anisotropic evolution path of the CMA-ES algorithm	62, 64, 141–143
$c_e$	$\in (0, 1]$ the learning rate for the isotropic evolution path of the CMA-ES algorithm	59, 62, 141–143
$c_\mu$	$\in (0, 1]$ the rank $\mu$ learning rate for the covariance of the CMA-ES algorithm	58–60, 62, 141, 143
$c_1$	$\in (0, 1]$ the rank one learning rate for the covariance of the CMA-ES algorithm	59, 60, 62, 141, 143
$\mathcal{D}$	$: \mathbb{R}^3 \times \mathbb{R}^n \rightarrow \mathbb{R}^3$ , an abstract deformation function, usually parametrized by a parameter vector $\Theta$ such that $\mathcal{D}(\cdot, \Theta) : \mathbb{R}^3 \rightarrow \mathbb{R}^3$	24, 25, 27, 35, 40–46, 48, 62, 63, 92, 111, 119–121, 129, 131, 132
$D$	$\subset \mathbb{R}^2$ , the set of valid depth pixels used for depth image analysis	39, 40
$\delta$	$: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ a camera lens distortion function	34, 35, 45
$d$	$\in \mathbb{R}$ , a depth value or distance in 3D	128–131
$d_{\max}$	$\in \mathbb{R}$ , robustification threshold of the depth error (maximum depth difference considered)	39, 40, 45, 46
Dom	the domain of a function, for input images it denotes the set of properly defined pixels	37, 39, 40
$\mathbf{E}$	$: \{\Omega \rightarrow [0, 1]\} \rightarrow \Omega$ the expected mean of a distribution	61, 91, 92, 142
$E$	$\in \mathbb{R}$ , Young’s modulus, see [NJK09] for a definition	111, 114, 116–118
$\hat{\mathbf{e}}$	$\in \mathbb{R}^n$ the anisotropic CMA-ES evolution path	62, 63, 67, 142–144
$\mathbf{e}$	$\in \mathbb{R}^n$ the isotropic CMA-ES evolution path	59, 60, 62, 63, 67, 142–144

$e$	$\in \mathbb{R}_{>0}$ an error value	36, 37, 39–41, 43– 46, 48, 63, 114, 115, 125–127
$\epsilon$	$\in \mathbb{R}_{>0}$ the minimum eigenvalue of the robustified version of the CMA-ES covariance	66, 67
$\mathcal{F}$	$: \mathbb{R}^n \rightarrow \mathbb{R}$ , a fitness function yielding the fitness value for a given $n$ -dimensional deformation vector $\Theta$	36, 37, 43, 51–57, 59, 62, 63, 65, 87, 92, 115, 125, 126, 141, 142, 149–153
$f_s$	$\in \mathbb{R}$ , the focal length parameter of the camera parameter vector $\mathbf{s}$	46, 48
$f$	$\in \mathbb{R}$ , the focal length of a camera	33, 34
$G$	$\in \mathbb{R}^3$ , a geometric sub set that should not be occupied by object vertices	43
$\gamma$	$\in \mathbb{R}_{\geq 0}$ , speed parameter used in the occlusion handling	94
$H$	$\in \mathbb{R}$ , a measure for the information of a modality for the optimization	126–128
$I$	$: \mathbb{R}^2 \rightarrow \mathbb{R}^n$ an image, given as a function yielding an $n$ -dimensional image content for a given image coordinate	35, 47, 131, 132
$I^{\text{amp}}$	$: \mathbb{R}^2 \rightarrow \mathbb{R}$ an amplitude image of a ToF sensor	131, 132
$I_{s,\text{in}}^c$	$: \mathbb{R}^2 \rightarrow \mathbb{N}_{<256}^3$ an input RGB color image $I$ from a camera with parameters $\mathbf{s}$	37, 48, 62
$I_{s,\Theta}^c$	$: \mathbb{R}^2 \rightarrow \mathbb{N}_{<256}^3$ a synthesized RGB color image $I$ , parametrized by camera parameters $\mathbf{s}$ and deformation parameters $\Theta$	36, 37, 62
$I_{s,\text{in}}^d$	$: \mathbb{R}^2 \rightarrow \mathbb{R}$ an input depth image $I$ , from a depth sensor with parameters $\mathbf{s}$	39, 40, 45, 46, 48, 62
$I_{s,\Theta}^d$	$: \mathbb{R}^2 \rightarrow \mathbb{R}$ a synthesized depth image $I$ , parametrized by camera parameters $\mathbf{s}$ and deformation parameters $\Theta$	36, 39, 40, 62

## Glossary

$I^{\text{ir}}$	: $\mathbb{R}^2 \rightarrow \mathbb{R}$ an infra-red image of the <i>Kinect</i> sensor, given as a function yielding the ir-response for a given image coordinate	103, 104
$\mathbf{K}$	$\in \mathbb{R}^{4 \times 3}$ , the K-matrix (camera matrix).	34, 35
$k$	or $k_i \in \mathbb{R}$ , (indexed) camera distortion parameter	35
$k$	$\in (0, 1)$ , a knot value of a NURBS function	19, 20, 26
$\mathcal{L}$	: $\Omega \rightarrow [0, 1]$ , the likelihood function of an event	129
$\varphi$	$\in \mathbb{R}_{>0}$ the weight of an error value $e$ in the fitness function $\mathcal{F}$	37, 42, 115, 125–127
$\mathbf{m}$	$\in \mathbb{R}^n$ , the mean vector of a Gaussian distribution	52–60, 62, 65–67, 87, 141–144, 149
$m$	$\in \mathbb{R}$ , the mean value of a Gaussian distribution	130–132, 134
$\mu$	$\in \mathbb{N}_{\leq \lambda}$ the number of CMA-ES individuals selected in each iteration	54–56, 58–60, 64, 65, 141, 149, 152
$\mu_{\text{eff}}$	$= \frac{1}{\ \mu\ ^2}$ the variance effective selection mass in the CMA-ES algorithm	59, 141–143
$\mathcal{N}$	: $\Omega \rightarrow [0, 1]$ , the normal distribution	53, 56, 59, 61, 67, 87, 91, 92, 142, 144, 149–152
$\mathbb{N}$	the set of natural numbers	18, 19, 35, 53, 54, 56, 141
$\nu$	$\in \mathbb{R}$ , Poisson’s ratio, see [All99] for a definition	111, 114
$\Omega$	a set of events (in the sense of probability theory)	128
$o$	$\in \mathbb{R}$ , offset (signed distance) to a NURBS surface	20–22, 24–26
$P$	$\subset \mathbb{R}^3$ , a set of NURBS control points	18, 19, 22–26, 40, 41
$\mathbf{P}$	: $\Omega \rightarrow [0, 1]$ , the probability of an event	ix, 56, 128, 149–151



$\mathbb{P}^3$	$\equiv \mathbb{R}^4/0$ , the three dimensional projective space	ix, 32
$\mathbf{p}$	$\in P \subset \mathbb{R}^3$ , a NURBS control point	18, 19, 23, 26
$p$	$\in \mathbb{N}$ the number of pixels an object is projected onto by the rendering	91, 92
$\rho$	$\in \mathbb{R}^2$ , principal point of a perspective projection described by $\mathbf{K}$	34
$\phi$	$:\mathbb{R} \rightarrow [0, 1]$ , Probability density function for a ToF measurement	130–132
$\mathbf{p}$	$\in [0, 1]$ , a variable containing a probability value	151
$\lambda$	$\in \mathbb{N}$ the number of CMA-ES individuals	53–58, 60, 141–143, 149, 152
$\mathcal{P}$	$:\mathbb{R}^3 \rightarrow \mathbb{R}^2$ or $\mathbb{P}^3 \rightarrow \mathbb{R}^2$ , the projection function from 3D space into image coordinates	35, 46, 47, 129, 131, 132
$\mathcal{P}_s$	$:\mathbb{R}^3 \rightarrow \mathbb{R}^2$ or $\mathbb{P}^3 \rightarrow \mathbb{R}^2$ , the projection function $\mathcal{P}$ with respect to parameters $\mathbf{s}$	45, 48
$\Psi$	$\subset \mathbb{R}^3 \times \mathbb{R}^3$ , the set of 3D point pairs defining a stretch constraint	40, 41
$P_0$	$\subset \mathbb{R}^3$ , a control point set $P$ directly after initialization	20–22, 24–26
$q$	$\in (0, 1]$ damping factor for the step size $\sigma$ of the CMA-ES algorithm	62, 141–143
$\mathcal{R}$	or $\mathcal{R}_p : [0, 1]^2 \rightarrow [0, 1]$ , weighting function for a NURBS control point $\mathbf{p}$	18, 19, 23, 26
$\mathbf{R}$	$\in \mathbb{R}^{3 \times 3}$ , a rotation matrix for points in 3D space	x, 32, 33, 45
$\mathbb{R}$	the set (field) of real numbers	ix, x, 18, 21, 23–25, 32–37, 39, 41, 43, 51–53, 55, 59, 67, 92, 94, 110, 115, 126, 146
$\mathbf{R}_x$	$:\mathbb{R} \rightarrow \mathbb{R}^{3 \times 3}$ , a function yielding a matrix for a rotation around the x axis by a given angle	x, 111, 121

## Glossary

$\mathbf{R}_y$	: $\mathbb{R} \rightarrow \mathbb{R}^{3 \times 3}$ , a function yielding a matrix for a rotation around the y axis by a given angle	x, 111, 121
$\mathbf{R}_z$	: $\mathbb{R} \rightarrow \mathbb{R}^{3 \times 3}$ , a function yielding a matrix for a rotation around the z axis by a given angle	x, 111, 121
$S$	a set of abstract indexing elements $\mathbf{s}$ for sensors and constraints	36, 37, 125, 126, 129, 132
$\mathbf{s}$	$\in \mathbb{R}^{3 \times 3} \times \mathbb{R}^3 \times \mathbb{R}^{4 \times 2} \times \mathbb{R}^n$ , a sensor described by a set of camera parameters containing a camera orientation $\mathbf{R}$ , center $\mathbf{c}$ , K-matrix $\mathbf{K}$ , and distortion parameters $k_i$	33–37, 45–48, 62, 125, 126, 129, 132
$\sigma$	$\in \mathbb{R}_{\geq 0}$ , variance of a distribution	53, 56–62, 64, 66, 87, 91, 92, 130–132, 141–144, 149, 151–153
$s$	$\in \mathbb{R}$ , the amount of skew of introduced to a projection by the camera matrix $\mathbf{K}$	34
$\sigma_s$	$\in \mathbb{R}_{\geq 0}$ , deviation of the noise in $\mathcal{F}$ introduced by sparsification	49, 56, 87, 150–153
$\Theta$	$\in \mathbb{R}^n$ a parameter vector, parametrizing the abstract deformation function $\mathcal{D}$ . When indexed, $\Theta_{i,j}$ denotes the $j^{\text{th}}$ parameter sample of the $i^{\text{th}}$ CMA-ES iteration	25, 27, 35–37, 39–41, 43–46, 48, 51, 53, 54, 56–60, 62, 65–67, 87, 92, 110, 111, 114, 115, 120, 125–127, 129, 131, 132, 142–144, 149, 150, 152
$T$	$\subset \mathbb{N}^3$ , a set of vertex indices defining a triangle mesh topology	31, 32, 35, 41

$\theta$	$\in \mathbb{R}_{>0}$ the minimum value to initialize CMA-ES step size $\sigma$	66, 144
$\mathbf{t}$	$\in \mathbb{R}^3$ , a vector describing a translation in 3D space	32, 33, 110, 111, 120, 121
$\tau$	$\in [-1, 1]$ , z-mapping parameter of the FlexPad deformation function	119, 120
$u$	$\in \mathbb{R}$ , first component of an image coordinate describing a position in the image plane	34, 35, 37, 39, 40, 103, 104, 130–132
$u$	$\in [0, 1]$ , first parameter in the NURBS parameter space	18–22, 24–26
$u'$	$\in \mathbb{R}$ , distorted first component of an image coordinate describing a position in the image plane	34, 35
$V$	$\subset \mathbb{R}^3$ , an (indexed) set of 3D points or vertices $\mathbf{v} (v_i)$	22–26, 35, 40, 41, 43–46, 48, 49, 95, 129, 131, 132
$\mathbf{v}$	$\in \mathbb{P}^3$ , a 3D point or vertex in homogenous coordinates	32–35
$\mathbf{v}$	$\in \mathbb{R}^3$ , a 3D point or vertex	20–26, 32, 35, 40, 41, 43–46, 48, 94–96, 104, 111, 112, 129, 131, 132, 146–148
$v$	$\in \mathbb{R}$ , second component of an image coordinate describing a position in the image plane	34, 35, 37, 39, 40, 103, 104, 130–132
$v$	$\in [0, 1]$ , second parameter in the NURBS parameter space	18–22, 24–26

## Glossary

$v'$	$\in \mathbf{R}$ , distorted second component of an image coordinate describing a position in the image plane	34, 35
$V_P$	$\subset [0, 1]^2$ , the set of indexed parameters for each indexed vertex in $V$ with respect to $P$	24
$V_{P_0}$	$\subset [0, 1]^2$ , the set of indexed parameters for each indexed vertex in $V$ after initialization (using $P_0$ )	24
$\mathbf{w}$	$\in [0, 1]^n$ , recombination weight vector of the CMA-ES algorithm	54, 55, 57–60, 62, 141, 143
$w$	$\in \mathbf{R}$ , a NURBS control point weight	19, 26
$x$	$\in \Omega$ , an event	128, 129
$\chi$	$\in \mathbf{R}$ with $\chi_n$ denoting the expected length of an $n$ -dimensional vector under $\mathcal{N}(\mathbf{0}, \mathbf{1})$	61, 62, 64, 91, 92, 142, 143
$y$	$: [0, 1] \rightarrow \mathbf{R}$ , a function of deflection mapping from the distance relative to the object length to the amount of deflection at that point	114

# Acronyms

AbS	Analysis by Synthesis	2, 3, 6, 8, 15, 17, 29, 30, 37, 42, 44, 48, 51, 52, 62, 67, 69, 87, 89, 90, 93, 97, 104, 109, 111, 128, 129, 133, 134, 137–139
AHD	Adaptive Homogeneity-Directed Demosaicing	37
BBOB	Black-Box Optimization Benchmarking Competition	51
CAD	computer aided design	18
CDF	cumulative distribution function	153
CMA-ES	Covariance Matrix Adaptation - Evolution Strategy	7, 30, 51, 52, 56, 57, 59, 62–65, 67, 69–71, 75, 76, 79, 83, 84, 97, 110, 114, 121, 132, 135, 138, 139, 141, 143, 149

## Acronyms

CPU	Central Processing Unit	25, 44
DF	Directional Filtering	37
DoF	Degrees of Freedom	6, 17, 29
fps	frames per second	12, 46, 121
GPU	Graphics Processing Unit	44
ICP	Iterative Closest Point	5, 6
IR	infra-red	12, 13, 102–105, 137, 145–147
NN	Nearest Neighbor	37
NRSfM	Non-Rigid Structure from Motion	1, 3, 4, 17
NURBS	Non-Uniform Rational B-Spline	8, 17, 18, 20–26, 35, 40–42, 69–71, 75, 79, 97, 135, 136, 139
PDF	probability density function	53, 128, 129
PSO	Particle Swarm Optimization	52
RAM	Random Access Memory	44
RGB	red-green-blue	37, 84, 86
RMS	Root Mean Square	6, 22, 23, 70–72, 75, 97, 115, 121
SAD	Sum of Absolute Differences	48
SD	standard deviation	115, 117, 131, 134
SfM	Structure from Motion	1, 3, 4, 17
SOCP	Second Order Cone Programming	5
SVD	Singular Value Decomposition	67, 144
ToF	Time-of-Flight	9, 13, 14, 44, 93, 128–130, 133–135

# Bibliography

- [ABH10] Anne Auger, Dimo Brockhoff, and Nikolaus Hansen. Benchmarking the  $(1,\lambda)$ -CMA-ES With Mirrored Sampling and Sequential Selection on the Noisy BBOB-2010 Testbed. In *GECCO workshop on Black-Box Optimization Benchmarking (BBOB'2010)*, pages 1625–1632. ACM, 2010.
- [AG01] Farid Alizadeh and Donald Goldfarb. Second-order cone programming. *Mathematical Programming*, 95:3–51, 2001.
- [All99] Michael Allaby. *A dictionary of earth sciences*. Oxford paperback reference. Oxford University Press, 1999.
- [ASB00] Xavier Armangué, Joaquim Salvi, and Joan Batlle. A comparative review of camera calibrating methods with accuracy evaluation. *Pattern Recognition*, 35:1617–1635, 2000.
- [Bal08] Ákos Balázs. *Tessellation and rendering of trimmed NURBS models in scene graph systems*. Dissertation, Universität Bonn, September 2008.
- [Bau01] Heinz Bauer. *Measure and Integration Theory*. De Gruyter studies in mathematics. W. de Gruyter, 2001.
- [Bay76] Bryce E. Bayer. Color imaging array, 1976. US Patent 3,971,065.
- [BB98] Benedicte Bascle and Andrew Blake. Separability of pose and expression in facial tracking and animation. In *Proceedings of the Sixth International Conference on Computer Vision, ICCV '98*, pages 323–, Washington, DC, USA, 1998. IEEE Computer Society.

## Bibliography

- [BCA98] Eric Bardinet, Laurent D. Cohen, and Nicholas Ayache. A parametric deformable model to fit unstructured 3d data. *Computer Vision and Image Understanding*, 71(1):39–54, 1998.
- [BER76] James R. Bitner, Gideon Ehrlich, and Edward M. Reingold. Efficient generation of the binary reflected gray code and its applications. *Communications of the ACM*, 19(9):517–521, September 1976.
- [Ber95] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1995.
- [Ber96] Dimitri P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*. Athena Scientific, 1 edition, 1996.
- [BGV92] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT '92*, pages 144–152, New York, NY, USA, 1992. ACM.
- [BHB00] Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3d shape from image streams. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2690–2696. IEEE Computer Society, 2000.
- [BM92] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, February 1992.
- [Bou99] Jean-Yves Bouguet. *Visual methods for three-dimensional modeling*. PhD thesis, Pasadena, CA, USA, 1999. AAI9941097.
- [Bro71] Duane C. Brown. Close-range camera calibration. *Photogrammetric Engineering*, 37(8):855–866, 1971.
- [BZ04] Adrien Bartoli and Andrew Zisserman. Direct estimation of non-rigid registration. In *In British Machine Vision Conference*, 2004.
- [CBI09] Cedric Cagniar, Edmond Boyer, and Slobdodan Ilic. Iterative mesh deformation for dense surface tracking. In *12th International Conference On Computer Vision Workshops*, 2009.



- [CBK05] Kong Man Cheung, Simon Baker, and Takeo Kanade. Shape-from-silhouette across time part i: Theory and algorithms. *International Journal of Computer Vision*, 62(3):221 – 247, May 2005.
- [CC91] Laurent D. Cohen and Isaac Cohen. Finite element methods for active contour models and balloons for 2d and 3d images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:1131–1147, 1991.
- [CGZZ10] Qin Cai, David Gallup, Cha Zhang, and Zhengyou Zhang. 3d deformable face tracking with a commodity depth camera. *Camera*, 6313(2):229–242, 2010.
- [CK94] Joao Costeira and Takeo Kanade. A multi-body factorization method for motion analysis. Technical Report CMU-CS-TR-94-220, Computer Science Department, Pittsburgh, PA, September 1994.
- [CS09] Boguslaw Cyganek and J. Paul Siebert. *Image Warping Procedures*, pages 409–428. John Wiley and Sons, Ltd, 2009.
- [dAST<sup>+</sup>08] Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. Performance capture from sparse multi-view video. *ACM Trans. Graph.*, 27(3):98:1–98:10, August 2008.
- [dATSS07] Edilson de Aguiar, Christian Theobalt, Carsten Stoll, and Hans-Peter Seidel. Marker-less deformable mesh tracking for human shape and motion capture. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Minneapolis, USA, June 2007. IEEE.
- [DBA06] Alessio Del Bue and Lourdes Agapito. Non-rigid stereo factorization. *International Journal of Computer Vision*, 66:193–207, February 2006.
- [DHI91] Herve Delingette, Martial Hebert, and Katsushi Ikeuchi. Deformable surfaces: A free-form shape representation. In *Proc. SPIE Vol 1570: Geometric Methods in Computer Vision*, pages 21–30, 1991.
- [DSA07] Alessio Del Bue, Fabrizio Smeraldi, and Lourdes Agapito. Non-rigid structure from motion using ranklet-based tracking and

## Bibliography

- non-linear optimization. *Image and Vision Computing*, 25(3):297–310, March 2007.
- [FAB10] João Fayad, Lourdes Agapito, and Alessio Del Bue. Piecewise quadratic reconstruction of non-rigid surfaces from monocular sequences. In *Proceedings of the 11th European conference on Computer vision: Part IV, ECCV'10*, pages 297–310, Berlin, Heidelberg, 2010. Springer-Verlag.
- [Fal07] Dragos Falie. Noise characteristics of 3d time-of-flight cameras. *International Symposium on Signals Circuits and Systems (ISSCS 2007)*, 1:1–4, 2007.
- [FDAA09] Joao Fayad, Alessio Del Bue, Lourdes Agapito, and Pedro .M.Q. Aguiar. Non-rigid structure from motion using quadratic deformation models. In *British Machine Vision Conference (BMVC), London, UK*, 2009.
- [FJP<sup>+</sup>12] Andreas Rune Fugl, Andreas Jordt, Henrik Gordon Petersen, Morten Willatzen, and Reinhard Koch. Simultaneous estimation of material properties and pose for deformable objects from depth and color images. In *Pattern Recognition - Joint 34th DAGM and 36th OAGM Symposium*, volume 7476 of *Lecture Notes in Computer Science*, pages 165–174. Springer, 2012.
- [FLP04] Olivier Faugeras, Quang-Tuan Luong, and Théodore. Papadopoulo. *The Geometry of Multiple Images: The Laws That Govern the Formation of Multiple Images of a Scene and Some of Their Applications*. Mit Press, 2004.
- [GAL07] Sigurjon Ami GuOmundsson, Henrik Aanaes, and Rasmus Larsen. Environmental effects on measurement uncertainties of time-of-flight cameras. In *Signals, Circuits and Systems, 2007. ISSCS 2007. International Symposium on*, volume 1, pages 1–4, 2007.
- [GCD11] John P. Godbaz, Michael J. Cree, and Adrian A. Dorrington. Understanding and ameliorating non-linear phase and amplitude responses in amcw lidar. *Remote Sensing*, 4(1):21–42, 2011.
- [GHL03] Chun Guan, Laurence G. Hassebrook, and Daniel Lau. Composite structured light pattern for three-dimensional video. *Opt. Express*, 11(5):406–417, Mar 2003.

- [Gor12] Steven J. Gortler. *Foundations of 3D Computer Graphics*. The MIT Press, 2012.
- [GSA<sup>+</sup>09] Jürgen Gall, Carsten Stoll, Edison De Aguiar, Christian Theobalt, Bodo Rosenhahn, and Hans peter Seidel. Motion capture using joint skeleton tracking and surface estimation. In *in IEEE Conference on computer vision and pattern recognition*. IEEE, 2009.
- [Han98] Nikolaus Hansen. *Verallgemeinerte individuelle Schrittweitenregelung in der Evolutionsstrategie*. Mensch und Buch Verlag, 1998.
- [Han06] Nikolaus Hansen. The CMA evolution strategy: a comparing review. In J.A. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, editors, *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer, 2006.
- [HBW99] Seon M. Han, Heym Benaroya, and Timothy Wei. Dynamics of transversely vibrating beams using four engineering theories. *Journal of Sound and Vibration*, 225:935–988, 1999.
- [HE09] Anna Hilsmann and Peter Eisert. Realistic cloth augmentation in single view video. In *Vision, Modeling, and Visualization Workshop 2009*, Braunschweig, Germany, November 2009.
- [HH91] Berthold K. P. Horn and John G. Harris. Rigid body motion from range image sequences. *CVGIP: Image Understanding*, 53:1–13, January 1991.
- [HMP05] Keigo Hirakawa, Student Member, and Thomas W. Parks. Adaptive homogeneity-directed demosaicing algorithm. *IEEE Trans. Image Processing*, 14:360–369, 2005.
- [HO01] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, June 2001.
- [HR10] Nikolaus Hansen and Raymond Ros. Black-box optimization benchmarking of newuoa compared to bipop-cma-es: on the bbob noiseless testbed. In *Genetic and Evolutionary Computation Conference, GECCO 2010, Proceedings, Portland, Oregon, USA*,

## Bibliography

- July 7-11, 2010, Companion Material*, pages 1519–1526. ACM, 2010.
- [HS95] Richard I. Hartley and Peter F. Sturm. Triangulation. In *Proceedings of the 6th International Conference on Computer Analysis of Images and Patterns, CAIP '95*, pages 190–197, London, UK, UK, 1995. Springer-Verlag.
- [HZ00] Richard I. Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.
- [IB98] Michael Isard and Andrew Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28, 1998.
- [IY01] Gavriel J. Iddan and Giora Yahav. Three-dimensional imaging in the studio and elsewhere. *Proc. SPIE*, 4298:48–55, 2001.
- [JFB<sup>+</sup>11] Andreas Jordt, Andreas R. Fugl, Leon Bodenhausen, Morten Willatzen, Reinhard Koch, Henrik G. Petersen, Knud A. Andersen, Martin M. Olsen, and Norbert Krueger. An Outline for an intelligent System performing Peg-in-Hole Actions with flexible Objects. In *International Conference on Intelligent Robotics and Applications (ICIRA 2011)*, 2011.
- [JJU04] Simon J. Julier, Jeffrey, and K. Uhlmann. Unscented filtering and nonlinear estimation. In *Proceedings of the IEEE*, volume 92, pages 401–422, 2004.
- [JK11] Andreas Jordt and Reinhard Koch. Fast tracking of deformable objects in depth and colour video. In *Proceedings of the British Machine Vision Conference, BMVC 2011*. British Machine Vision Association, 2011.
- [JK12] Andreas Jordt and Reinhard Koch. Direct model-based tracking of 3d object deformations in depth and color video. *International Journal of Computer Vision*, pages 1–17, 2012. 10.1007/s11263-012-0572-1.
- [JK13] Andreas Jordt and Reinhard Koch. Reconstruction of deformation from depth and color video with explicit noise models. In

- Marcin Grzegorzek, Christian Theobalt, Reinhard Koch, and Andreas Kolb, editors, *Time-of-Flight and Depth Imaging: Sensors, Algorithms, and Applications*, volume 8200 of *Lecture Notes in Computer Science*, pages 128–146. Springer Berlin Heidelberg, 2013.
- [JLS00] Ales Jaklic, Ales Leonardis, and Franc Solina. *Segmentation and Recovery of Superquadrics*, volume 20 of *Computational imaging and vision*. Kluwer, Dordrecht, 2000. ISBN 0-7923-6601-8.
- [JPMP14] David Jiménez, Daniel Pizarro, Manuel Mazo, and Sira E. Palazuelos. Modeling and correction of multipath interference in time of flight cameras. *Image Vision Comput.*, 32(1):1–13, 2014.
- [JSBK10] Andreas Jordt, Ingo Schiller, Johannes Bruenger, and Reinhard Koch. High-resolution object deformation reconstruction with active range camera. In Michael Goesele, Stefan Roth, Arjan Kuijper, Bernt Schiele, and Konrad Schindler, editors, *Pattern Recognition*, Lecture Notes in Computer Science, pages 543–552. Springer Berlin / Heidelberg, 2010.
- [JSK13] Andreas Jordt, Jürgen Steimle, and Reinhard Koch. Verfahren zur echtzeitfähigen materialschätzung und zur materialbasierten bildsegmentierung in elektronischen bildsequenzen, 2013.
- [KE95] James Kennedy and Russell C. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.
- [Koc93] Reinhard Koch. Dynamic 3-d scene analysis through synthesis feedback control. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(6):556–568, June 1993.
- [LNL<sup>+</sup>13] Damien Lefloch, Rahul Nair, Frank Lenzen, Henrik Schaefer, Lee Streeter, Michael J. Cree, Reinhard Koch, and Andreas Kolb. *Time-of-Flight and Depth Imaging, LNCS 8200*, volume 8200, chapter Technical Foundation and Calibration Methods for Time-of-Flight Cameras, pages 3–24. Springer, September 2013.
- [LSKK10] Marvin Lindner, Ingo Schiller, Andreas Kolb, and Reinhard Koch. Time-of-flight sensor calibration for accurate range sensing. *Computer Vision and Image Understanding*, 114(12):1318–1328, December 2010.

## Bibliography

- [MAC07] Daniele Menon, Stefano Andriani, and Giancarlo Calvagno. Demosaicing with directional filtering and a posteriory decision. *IEEE Trans Image Process* 16, 16:132–141, 2007.
- [Mar63] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.
- [MBB09] Enrique Muñoz, José Miguel Buenaposada, and Luis Baumela. A direct approach for efficiently tracking with 3d morphable models. In *ICCV*, pages 1615–1622. IEEE, 2009.
- [MSZ94] Richard M. Murray, S. Shankar Sastry, and Li Zexiang. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1994.
- [MT93] Tim Mcinerney and Demetri Terzopoulos. A finite element model for 3d shape reconstruction and nonrigid motion tracking. In *4th International Conference on In Computer Vision, ICCV*, pages 518–523, 1993.
- [NJK09] Miloslav Nic, Jiri Jirat, and Bedrich Kosata. Iupac compendium of chemical terminology – the gold book, 2009.
- [NS85] Arun N. Netravali and Jack Salz. Algorithms for estimation of three-dimensional motion. *AT&T Technical Journal*, 64(2), 1985.
- [OS88] Stanley Osher and James A. Sethian. Fronts propagating with curvature dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computations Physics*, 79(1):12–49, 1988.
- [Pho75] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, June 1975.
- [PLF08] Julien Pilet, Vincent Lepetit, and Pascal Fua. Fast non-rigid surface detection, registration and realistic augmentation. *Interation Journal of Computer Vision*, 76:109–122, February 2008.
- [PT97a] Les Piegl and Wayne Tiller. *The NURBS Book*. Springer, Berlin, 2nd edition, 1997.
- [PT97b] Les Piegl and Wayne Tiller. *The NURBS book (2nd ed.)*. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

- [PW82] Edward Pervin and Jon A. Webb. Quaternions in computer vision and robotics. Technical Report CMU-CS-82-150, Carnegie Mellon University, 1982.
- [RDP<sup>+</sup>11] Malcolm Reynolds, Jozef Doboš, Leto Peel, Tim Weyrich, and Gabriel J. Brostow. Capturing time-of-flight data with confidence. In *CVPR*, 2011.
- [RFA11] Chris Russell, Joao Fayad, and Lourdes Agapito. Energy based multiple model fitting for non-rigid structure from motion. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3009–3016, 2011.
- [Riv48] Ronald Samuel Rivlin. Large elastic deformations of isotropic materials. i. fundamental concepts. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 240(822):pp. 459–490, 1948.
- [RKP<sup>+</sup>07] Bodo Rosenhahn, Uwe Kersting, Katie Powell, Reinhard Klette, Gisela Klette, and Hans-Peter Seidel. A system for articulated tracking incorporating a clothing model. *Machine Vision and Applications*, 18:25–40, January 2007.
- [SBH11] Michael Stürmer, Guido Becker, and Joachim Hornegger. Assessment of time-of-flight cameras for short camera-object distances. In *Proceedings of the Vision, Modeling, and Visualization Workshop 2011*, pages 231–238, 2011.
- [SBK08] Ingo Schiller, Christian Beder, and Reinhard Koch. Calibration of a pmd camera using a planar calibration object together with a multi-camera setup. In *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume Vol. XXXVII. Part B3a, pages 297–302, Beijing, China, 2008. XXI. ISPRS Congress.
- [SFC<sup>+</sup>11] Jamie Shotton, Andrew W. Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, pages 1297–1304. IEEE, 2011.
- [SHB03] Lorenzo Torresani Stanford, Aaron Hertzmann, and Christoph Bregler. Learning non-rigid 3d shape from 2d motion. In *Pro-*

## Bibliography

- ceedings of the 17th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1555–1562. MIT Press, 2003.
- [SHF07] Mathieu Salzmann, Richard Hartley, and Pascal Fua. Convex optimization for deformable surface 3-d tracking. In *ICCV'07*, pages 1–8, 2007.
- [SJM13] Jürgen Steimle, Andreas Jordt, and Pattie Maes. Flexpad: Highly flexible bending interactions for projected handheld displays. In Stephen Brewster, Susanne Bødker, Patrick Baudisch, and Michel Beaudouin-Lafon, editors, *Proceedings of the 2013 Annual Conference on Human Factors in Computing Systems (CHI 2013)*, Paris, France, April 2013. ACM, ACM.
- [SLF07] Mathieu Salzmann, Vincent Lepetit, and Pascal Fua. Deformable surface tracking ambiguities. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.
- [SMSL10] Shuhan Shen, Wenjuan Ma, Wenhuan Shi, and Yuncai Liu. Convex optimization for nonrigid stereo reconstruction. *IEEE Transactions on Image Processing*, 19:782–794, March 2010.
- [SPB04] Joaquim Salvi, Jordi Pagés, and Joan Batlle. Pattern codification strategies in structured light systems. *PATTERN RECOGNITION*, 37:827–849, 2004.
- [Suf07] Kevin Suffern. *Ray Tracing from the Ground Up*. A. K. Peters, Ltd., Natick, MA, USA, 2007.
- [SZL08] Shuhan Shen, Yinqiang Zheng, and Yuncai Liu. Deformable surface stereo tracking-by-detection using second order cone programming. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–4. IEEE, 2008.
- [TJK10] Jonathan Taylor, Allan D Jepson, and Kiriakos N. Kutulakos. Non-rigid structure from locally-rigid motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2761 – 2768, 2010.
- [TK92] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9:137–154, November 1992.



- [Tsa87] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *Robotics and Automation, IEEE Journal of*, 3(4):323–344, August 1987.
- [TYAB01] Lorenzo Torresani, Danny B. Yang, Eugene J. Alexander, and Christoph Bregler. Tracking and modeling non-rigid objects with rank constraints. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 493–500, 2001.
- [VBCK99] Sundar Vedula, Simon Baker, Robert Collins, and Takeo Kanada. Three-dimensional scene flow. In *Proceedings of the 7th International Conference on Computer Vision, ICCV*, pages 722–726. IEEE, 1999.
- [WW91] Alan Watt and Mark Watt. *Advanced Animation and Rendering Techniques*. ACM, New York, NY, USA, 1991.
- [XPH13] Zenglin Xu, Travis Perry, and Gage Hills. Method and system for multi-phase dynamic calibration of three-dimensional (3d) sensors in a time-of-flight system, November 19 2013. US Patent 8,587,771.
- [YBBR93] Masanobu Yamamoto, Pierre Boulanger, Jean-Angelo Beraldin, and Marc Rioux. Direct estimation of range flow on deformable shape from a video rate range camera. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(1):82–89, January 1993.
- [YIM07] Giora Yahav, Gavriel J. Iddan, and David Mandelbroum. 3d imaging camera for gaming application. In *Consumer Electronics, 2007. ICCE 2007. Digest of Technical Papers. International Conference on*, pages 1–2, Jan 2007.
- [ZHXL08] Jianke Zhu, Steven C. Hoi, Zenglin Xu, and Michael R. Lyu. An effective approach to 3d deformable surface tracking. In *Proceedings of the 10th European Conference on Computer Vision: Part III, ECCV '08*, pages 766–779, Berlin, Heidelberg, 2008. Springer-Verlag.