Fast Tracking of Deformable Objects in Depth and Colour Video

Andreas Jordt
http://www.mip.informatik.uni-kiel.de
Reinhard Koch
http://www.mip.informatik.uni-kiel.de

Multimedia Information Processing
Department of Computer Science
Christian-Albrechts-University
Kiel, Germany

Abstract

One challenge in computer vision is the joint reconstruction of deforming objects from colour and depth videos. So far, a lot of research has focused on deformation reconstruction based on colour images only, but as range cameras like the recently released Kinect become more and more common, the incorporation of depth information becomes feasible.
In this article a new method is introduced to track object deformation in depth and colour image data. A NURBS based deformation function allows to decouple the geometrical object complexity from the complexity of the deformation itself, providing a low dimensional space to describe arbitrary 'realistic' deformations. While modelling the tracking objective as an analysis by synthesis problem, which is robust but usually computationally expensive, a set of optimisations is introduced, allowing a very fast calculation of the resulting error function. With a fast semi-global search a system is established that is capable of tracking complex deformations of large objects (6000 triangles and more) with more than 6Hz on a common desktop machine. The algorithm is evaluated using simulated and real data, showing the robustness and performance of the approach.

1 Introduction

Fast deformable object tracking is a prerequisite to the handling of flexible objects in robotics. Most of the existing deformation tracking methods apply 2D contour deformation tracking, optical flow-based, or feature-based tracking in colour images, but all of these algorithms require visible contours or texture information, i.e. rely on intrinsically two- or one dimensional image patches, introducing a clear limitation to their applicability.
Active real-time range sensors like Time-of-Flight cameras and the recently released Kinect camera allow to measure dense depth maps at high frame rates without the need for texture or contour information while allowing to additionally use colour information. However, even with dense depth maps at hand, the 3D motion in the observed scene remains unknown to the observer if no texture or trackable surface marks provide information about the motion along the 2D image plane.
So far, there is no algorithm able to track deformable objects in depth and colour information if no explicit clue on the 2D movement is provided by the colour image via contours, optical

BMVC 2011 http://dx.doi.org/10.5244/C.25.114

flow, or trackable features in real-time. Sequences like the one depicted in fig. 4 are nearly impossible to handle for such methods.

1.1 Contribution

In this article, an algorithm is introduced that is receiving a stream of colour and depth images as input data. It is capable of tracking deformable objects without the need to explicitly detect optical flow, extract contours, or track depth map textures. Instead, it searches for the deformation of the tracked object which fits most likely to the provided depth- and colour data. This way, even complex deformations on very flexible objects can be tracked when only little or no colour information provide a clue on 2D motion. A system is introduced, similar to [12], to decouple the deformation complexity from the complexity of the object surface, not only allowing to have a low dimensional deformation space for complex objects but also to apply the same deformation to different objects. A fast semi global search is providing results in less then 160ms, making the algorithm capable of real-time tracking.
The remainder of this article is organised as follows: Section 1.2 gives an overview on deformation tracking algorithms. In section 2, our algorithm is introduced in detail. Experimental results based on synthetic and real input data can be found in section 3. Section 4 draws a conclusion and provides an outlook on further improvements.

1.2 Related Work

There are only few deformation tracking algorithms that incorporate depth data from active range sensors [12]. Mitra *et al.* [13] introduced a system for dynamic geometry registration under small deformations using 4D space-time surfaces with no application of colour data. Like most of the algorithms introduced in this section it is designed as an offline method, optimising over a complete sequence of input images.
Multi-camera setups for deformation tracking are most often used for motion capturing and people tracking. Aguiar *et al.* [6],[7] use analysis by synthesis to deform a low resolution triangle mesh of the tracked object using SIFT feature tracking to deliver first estimates for local optimisation. Cagniar *et al.* [4] introduced an iterative ICP like approach to track surface patches throughout the image sequence. This approach, too, relies on textured surfaces on the tracked object to create the underlying 3D measurements. Rosenhahn *et al.* [15] rely only on contours in their tracking approach but use an explicit human body model as prior knowledge.
Most of the stereo based algorithms for deformation tracking propagate the stereo correspondences over time. This is a reasonable approach since it directly yields sparse 3D scene flow information. Shen *et al.* [18], [19] formulate the triangle vertex fitting to a 3D movement as an second order cone problem and solve it using convex optimisation techniques. Their work is based on the methods introduced by Salzmann *et al.* [16] for monocular image data improved by Zhu *et al.* [21] with a fast converging problem formulation. Hilsmann and Eisert [11] introduced an algorithm for real time cloth tracking based on optical flow.
A different approach to deformation tracking is non-rigid structure from motion (NRSfM) [8]. NRSfM is based on the Structure from motion approach by Tomasi and Kanade [20], which is assuming a rigid scene to extract 3D data and camera poses from 2D correspondences. NRSfM, as first introduced by Bregler *et al.* [3] for monocular camera data adds linear deformation parameters to the search space allowing the scene to change. Del Bue [9] extended this approach to stereo data.

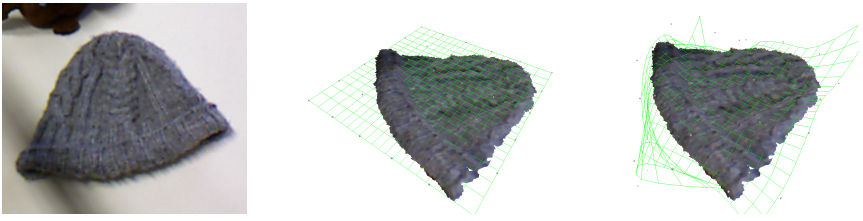

Figure 1: Left: The colour input from the Kinect data; Centre: The selected object with a bounding box initialisation of the nurbs surface (green); Right: The selected object with an approximation initialisation of the NURBS surface (green)

A mutual property of all these algorithms is the dependency on detectable 2D motion in the colour images. The example applications of these algorithms often use smooth surfaces with high-frequency colour changes in the texture, to provide the possibility of stable 2D motion detection.

2 Deformation Tracking

The tracking method we propose in this paper is performing analysis by synthesis on a stream of depth and colour images received from a Kinect camera. The basic idea is to define a 3D object mesh from the initial frame and deform it subsequently in the following images, such that the deformed mesh matches the depth and colour data best.
Unlike most other approaches, the deformation is not described by a displacement for every vertex in the mesh but a deformation function that allows arbitrary 'realistic' deformation.

The foundation of this deformation functions is a NURBS function [14] (Non-uniform rational b-spline), similar to [12]. NURBS surfaces are very common in computer graphics to describe continuous surfaces because of their intuitive handling and their ability to approximate every kind of surface. Due to the good convergence behaviour when fit to a point cloud, NURBS surfaces are also used in computer vision.
A NURBS surface function $\mathcal{N}$ in 3D is defined by a set of m^2 control points $c_{i,j} \in C$:

$$\mathcal{N}_C : [0,1]^2 \to \mathbb{R}^3, (u,w) \mapsto \sum_i \sum_j R_{i,j}(u,w) c_{i,j} \tag{1}$$

where

$$R_{i,j}(u,w) = \frac{n_{i,d}(u) n_{j,d}(w)}{\sum_{k=0}^{m} \sum_{l=0}^{m} n_{k,d}(u) n_{l,d}(w)}. \tag{2}$$

The functions $n_{i,d}(\cdot)$ provide the recursively defined base functions of degree d for a knot i. Please refer to [14] for further details on NURBS functions.

For the proposed method, we use a NURBS function as a rough approximation of the initial surface shape so that the actual high resolution mesh provided by the Kinect device can be registered to this surface. Once the mesh is registered to the NURBS surface function, the mesh is deformed by manipulating the control points of the NURBS function. The tracking is done by finding the set of NURBS control points, which is deforming the mesh in the way that fits the observations best.

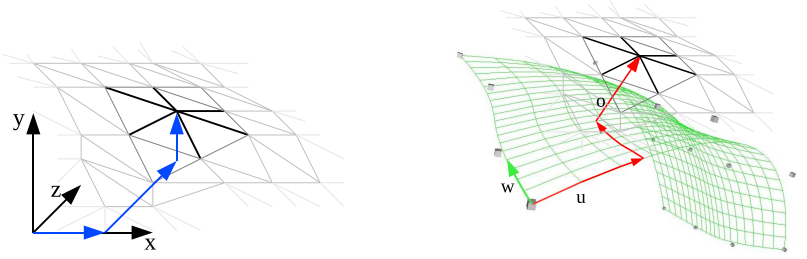

Figure 2: Left: A mesh vertex described by its native (x,y,z) coordinates. Right: A vertex mesh described by the parameters (u,w) and distance o.

The colour and depth sensors are calibrated according to [17] such that the intrinsic and relative extrinsic parameters of the setup are known.

2.1 Initial Approximation

The object of interest needs to be defined in the first frame, which can be done via user interaction, colour segmentation, depth segmentation, etc.. Figure 1 e.g. shows a cap extracted by colour segmentation. We assume to have such an initial object selection given. The initial depth and colour images can now be transformed into a coloured mesh: The depth image is used to generate a 3D triangle mesh by unprojecting all pixels in the image to 3D vertices according to the given calibration information and connecting these vertices according to their adjacencies in the depth image. Afterwards, every vertex is projected into the colour image to look up a vertex colour. If segmentation information for the initial object is available in the colour or depth image this information can be propagated to the 3D mesh to extract the initial 3D model of the tracked object.

The first step of the deformation tracking is to create a NURBS surface function and to make a rough approximation of the object surface with it (fig. 1 right). This is done by minimising the distances to the NURBS surface of all mesh vertices, a standard procedure that is described in detail in e.g. [5]. Another possible initial approximation of the object is to equally distribute the control points in a bounding box fashion around the object (fig. 1 centre). Different initialisations will result in different stretching behavoirs (see section 2.3).

2.2 Mesh Registration

After the NURBS surface is roughly fit to the 3D object model, the vertices are registered to the NURBS surface function. Let $\mathcal{N}_{C_0}$ be the NURBS surface function for the initial set of control points C_0 and let V be the set of mesh vertices $v_i \in \mathbb{R}^3, i = 0,\ldots,|V|$ of the object mesh.
In a first step, we find the parameters $(u,w) \in [0,1]^2$ such that

$$(u_i,w_i) = \arg\min_{u,v} \left(\|\mathcal{N}_{C_0}(u,w) - v_i\|_2 \right) \tag{3}$$

for every vertex $v_i \in V$, i.e. the surface point closest to each vertex. Let $\hat{\mathcal{N}}$ be the surface normal function for a given NURBS surface $\mathcal{N}$. Since the initial surface function $\mathcal{N}_{C_0}$ is

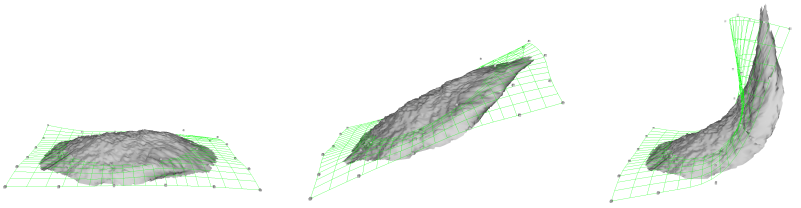

Figure 3: A triangle mesh registered to a NURBS function is deformed according to the control points describing the NURBS surface.

chosen "large enough", these minimal arguments do not lie directly at the border of the parameter space and, hence, the surface normal $\hat{\mathcal{N}}_{C_0}(u_i, w_i)$ of the corresponding surface point $\mathcal{N}_{C_0}(u_i, w_i)$ is pointing directly at the vertex v_i. Let o_i be the signed distance of the vertex v_i to $\mathcal{N}_{C_0}(u_i, w_i)$, i.e. the offset in respect to the normal vector. Then:

$$v_i = (x_i, y_i, z_i) = \mathcal{N}_{C_0}(u_i, w_i) + \hat{\mathcal{N}}_{C_0}(u_i, w_i) \cdot o_i. \tag{4}$$

Figure 2 depicts a vertex described by (x, y, z) and by (u, w, o). A change in the NURBS surface by moving the control points can now be transferred to every vertex of the triangle mesh. The fundamental achievement of this representation is the complete decoupling of the complexity of the mesh (number of vertices) and the complexity of the deformation (number of NURBS control points).

2.3 Error Function

The registration of the mesh to the NURBS function allows to synthesise arbitrary mesh deformation, limited in complexity by the number of control points the NURBS surface is defined by. The fit of a deformation to a pair of depth and colour images is defined by an error function, describing the difference between the deformed mesh and the measurements. This function is a sum of three values: depth error, colour error and stretch penalty.

The depth error describes the negative fit of the deformed mesh to the current depth measurement. For the following equations, Let $p_{j,i}$ be the deformed vertex v_i at frame j, such that $p_{j,i} = \mathcal{N}_{C_j}(u_i, w_i) + \hat{\mathcal{N}}_{C_j}(u_i, w_i) \cdot o_i$. Let P_D be the projection of the depth sensor with the centre of projection at c_D and let I_j be the depth image, then the depth error e_d is defined by

$$e_d = \frac{\sum_i \left(I_j\left(P_D\left(p_{j,i}\right)\right) - \|c_D - p_i\|_2\right)^2}{|V|}, \tag{5}$$

i.e. the mean squared difference between the vertex-camera centre distance and the corresponding depth value in the depth image.

The colour error e_c is calculated similarly by projecting each vertex into the colour image and comparing this colour value to the initial colour value of this vertex. Let P_C be the colour camera projection function and $\hat{I}_j$ be the colour image of a frame j, then

$$e_c = \frac{\sum_i \|\hat{I}_j(P_C(p_{j,i})) - \hat{I}_0(P_C(p_{0,i}))\|_2^2}{|V|}. \tag{6}$$

The distance between two colour values is calculated in the HSL colour space while neglecting the L-component (Lightness). This way, we regard varying illumination caused by changing surface normal directions during the deformation.
To ensure that the generated deformation is 'realistic' a third error value, the stretch penalty, is introduced. This stretch penalty increases the more the NURBS surface is stretching. This way, small stretching / shrinking deformations are allowed, as long as they go together with a better fit of the deformation to the colour or depth data. To calculate such changes on the surface, a set of points in the parameter domain is distributed on a grid across the parameter domain. Let G be the set of grid points $g \in [0,1]^2$ and V_g the set of direct and diagonal grid neighbours of g. Then the stretch penalty is defined as the average squared distance change:

$$e_s = \frac{\sum_{g \in G} \sum_{g' \in V_g} \left(\|\mathcal{N}_{C_j}(g) - \mathcal{N}_{C_j}(g')\|_2 - \|\mathcal{N}_{C_0}(g) - \mathcal{N}_{C_0}(g')\|_2 \right)^2}{\sum_{g \in G} |V_g|} \tag{7}$$

The value sums up the squared changes in the distances between the grid points from the initial frame to the current frame. At this point the choice of the NURBS initialisation (2.1) is affecting the tracking. The placement of the initial NURBS surface defines the subspace stretching the least.
The complete error function result is now defined as the weighted sum of e_d, e_c and e_s:

$$e = e_d + \lambda_c e_c + \lambda_s e_s \tag{8}$$

In our tests, λ_c was fixed to 0.1, but similar results were achived with values between 0.02 and 0.3. An increased λ_c shifts the focus of the optimisation towards a good colour fit and can be adjusted in case of large depth error values of objects at large distances. λ_s was set to 5, this way, a strong penalty on stretching forces the optimisation to only allow small stretching.

2.4 Minimising the Error Function

The problem formulation for the deformation estimation is very generic and versatile, which allows to directly apply it to many deformation tracking tasks. Calculating the error values directly in the measurement domain guarantees the correct handling of measurement noise without the need to propagate it from the measurement domain into the optimisation domain.
The downside of such a formulation becomes obvious when trying to minimise the error value as a function depending on the set of control points C. The problem is high dimensional and not separable, it is not possible to calculate a derivation analytically, nor is the error function convex. Computing a hessian matrix numerically for a given parameter in an optimisation problem with e.g. 4×4 control points requires $(4 \cdot 4 \cdot 3)^2 = 2304$ error function evaluations, which is far too much if several iterations are necessary to find the required deformation for one frame.

When comparing optimisation methods for such a task [1], i.e. high dimensional, non-separable, non-convex, derivation-free fitness function optimisation, the CMA-ES (Covariance Matrix Adaptation - Evolution Strategy) [10] algorithm proves to be a fast and stable choice.
CMA-ES analyses the search space by sampling the fitness function for a set of parameters distributed with respect to a multivariate Gauss distribution, given by a covariance matrix.

The resulting fitness values of this samples are used to adapt the covariance matrix of the distribution such that the likelihood for higher fitness values of a corresponding set of random samples is maximised. The covariance can be seen as an approximation of the inverse hessian matrix, in the way Gauss-Newton methods apply them, just with a number of function evaluations far less and an evolutionary strategy to place and propagate the sample points.

CMA-ES recommends using $4+\lfloor 3\cdot\log(N)\rfloor$ individuals (samples) as a rule of thumb, N being the number of dimensions. We experienced a faster convergence using $4+\lfloor 5\cdot\log(N)\rfloor$ for the error function as formulated in section 2.3.
Evolutionary algorithms have the reputation of converging slowly. Nevertheless, compared to e.g. a Gauss-Newton method, CMA-ES can make close to 200 iterations with the same number of function evaluations the Gauss newton method requires for one iteration, for a small set of 4×4 control points.
The time consumption for the adaptation of the covariance and the calculation of a new set of samples for an iteration is negligibly small, i.e. the optimisation time is proportional to the evaluation time of the error function.

2.5 Efficiency Improvements

Though the usage of the CMA-ES optimiser minimises the number of function evaluations per frame, there are still thousands of error function results to be calculated for one pair of images. A straight forward implementation of the error function as it is described in section 2.3 yielded an average frame rate of 1-2 minutes per frame, including the calculation of lens distortion while looking up image coordinates, etc.. To speed up the evaluation, a set of special improvements was applied, in addition to the obvious ones (undistort images and convert images to HSL before optimising, etc.):

- The calculation of two sets of image coordinates from 3D points, i.e. the transformation into the camera coordinate systems and the projections for the colour and depth image can be reduced to only one calculation of image coordinates, if the colour image is warped into the depth image before the optimisation starts. Colour and depth image can then be addressed via the same image coordinates. See [2] for detailed information on the applied warping algorithm.

- Since there is only one camera coordinate system to calculate the image coordinates from, the optimisation can directly take place in this coordinate system. The NURBS surface function is easily transformed into the system by applying the transformation to the control points, as each point is a linear combination of these control points (see equation (1)).

- Though the NURBS surface function is a polynomial surface, the relationship of a surface point to the control points is linear for a constant parameter, as can be seen in eq. (1). This fact can be exploited in the error function: Since every vertex is rigidly bound to a constant parameter, there is no need to evaluate the computationally expensive recursive NURBS function (see section 2) for every evaluation. Before the optimisation starts, a NURBS-cache is generated containing the references to the non-zero weighted control points $c_{i,j}$ along with the according weights $R_{i,j}(u,v)$ for every

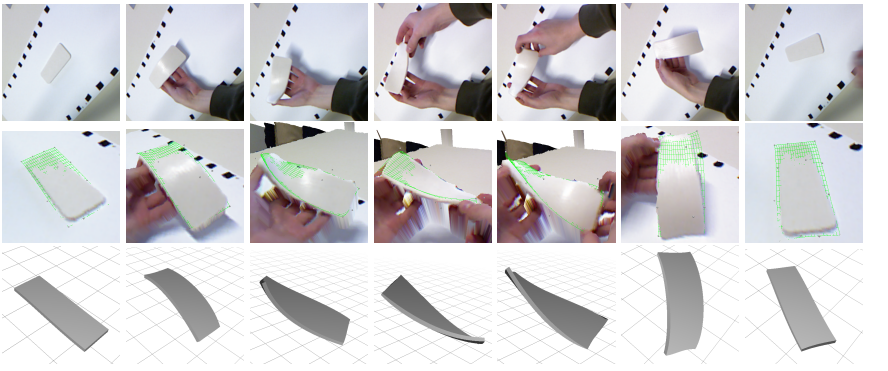

Figure 4: First row: Colour images from a deformation sequence recorded with a Kinect camera. Second row: The resulting coloured 3D mesh and the NURBS surface approximating the deformation of the tracked object. Third row: The deformation applied to a cube with a shape similar to the tracked object

vertex (see eq. (2)). A similar cache is generated for the surface normal vectors and the grid points for the stretch penalty.

In our experiments we experienced a speed up factor larger than 150 after introducing this improvements together with code, optimised for fast execution.

3 Experiments

To analyse the performance of the proposed algorithm, tests on synthetic and real data have been performed. Accuracy driven tests on synthetic and real data, with a large number of optimisation iterations, evaluate the method as an off-line tracking algorithm and speed driven tests on real data demonstrate the real-time capabilities. All image data has been acquired using a Microsoft Kinect camera, providing simultaneously colour and depth images with a resolution of 640×480 at 30Hz.

3.1 Synthetic Data Tests

To create most realistic synthetic input data (depth and colour images), a real tracking sequence (pick up and bending, see fig. 3) was reconstructed. The resulting deformation function was applied to an object mesh extracted from the initial frame. The deforming triangle mesh was put close to a background mesh, which was also reconstructed from real data. The resulting sequence was rendered into 640×480 depth and colour images.
Figure 5 (left) shows the average depth error of a sequence (RMS in mm) at different noise levels, i.e. different deviations for Gaussian distributed noise. Fig. 5 (centre) shows the actual RMS difference to the ground truth data respectively, being far beneath the noise levels and the depth reprojection error. The results were produced off-line, using 1000 CMA-ES iterations per frame, taking 4s to calculate one frame.

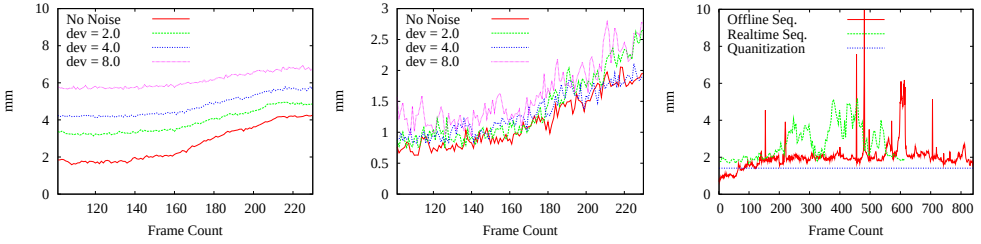

Figure 5: Left: The depth error of a simulated pick and bend sequence (RMS in mm). Centre: The actual difference between reconstructed mesh and ground truth of the same sequence (RMS in mm). Right: Depth error of a tracking sequence based on real data (fig. 4).

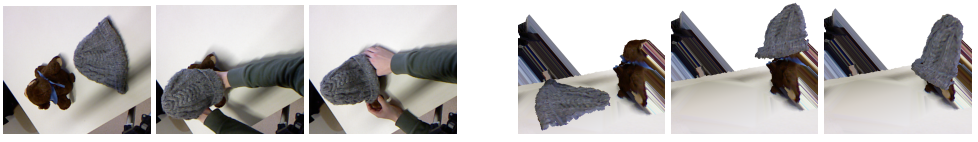

Figure 6: Left: Colour images from a Kinect recording. Right: Reconstructed deformation applied to the object mesh of the first frame. For illustration, the environment has been added as a static mesh.

3.2 Real Data Tests

To test the proposed method on real input data, several test sequences were recorded. Figure 4 shows one of the sequences used for off-line tracking, in which a white rubber object is deformed. Figure 5 (right, red) displays the RMS depth reprojection error of the object vertices over the complete sequence. The depth error is close to the quantisation step size of the depth values provided by the Kinect at that distance (depicted in fig. 5 (right, blue)), indicating a perfect fit. Figure 4 displays the semantic correctness of the fit. Figure 6 shows another reconstructed deformation from a wool cap put over a teddy bear.

For the real-time tracking, we reduced the number of iterations to 40 to achieve a high frame rate while still being able to keep track of the object. It shows, that capturing, preprocessing and the tracking process can be accomplished with more than 6Hz (160ms per frame) on a common desktop machine (8 cores, 2.4 GHz). Figure 5 (right, green) shows the depth error of a 20 seconds tracking sequence similar to fig. 3, less complex than the sequence in fig. 4. Though using far less iterations and a larger search space (caused by more movement per frame due to the dropped frames), the error of the real-time application is less than twice the error of the offline optimisation.

An additional feature resulting from the decoupling from deformation and geometry is the ability to transfer a deformation function to another geometry. Figure 4 (bottom) shows a recorded deformation applied to a simple box model.

4 Conclusion

In this article, a method is proposed to track pose and deformation of flexible objects in depth and colour video. It is able to reconstruct complex deformations without the need for

trackable features. It is able to provide tracking results at more than 6Hz on a desktop computer in real-time and high accuracy as an offline tracking system. The tracked deformation is decoupled from the actual object geometry and resolution, so it allows to apply recorded deformation to other objects.
For further improvement, we will investigate occlusion handling as well as inter-frame covariance propagation of the CMA-ES optimisation to further speed up the convergence rate.

References

[1] A. Auger, D. Brockhoff, and N. Hansen. Benchmarking the (1,4)-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.

[2] Bogumil Bartczak, Ingo Schiller, Christian Beder, and Reinhard Koch. Integration of a time-of-flight camera into a mixed reality system for handling dynamic scenes, moving viewpoints and occlusions in real-time. In *Proceedings of the 3DPVT Workshop*, Atlanta, GA, USA, June 2008.

[3] Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3d shape from image streams. In *CVPR*, pages 2690–2696. IEEE Computer Society, 2000. ISBN 0-7695-0662-3.

[4] Cedric Cagniar, Edmond Boyer, and Slobdodan Ilic. Iterative mesh deformation for dense surface tracking. In *12th International Conference On Computer Vision Workshops*, 2009.

[5] Fernand S. Cohen, Walid Ibrahim, and Chuchart Pintavirooj. Ordering and parameterizing scattered 3d data for b-spline surface approximation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(6):642–648, 2000. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/34.862203.

[6] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun. Performance capture from sparse multi-view video. In *ACM Transactions on Graphics,Proc. of ACM SIGGRAPH*, volume 27, 2008.

[7] Edilson de Aguiar, Christian Theobalt, Carsten Stoll, and Hans-Peter Seidel. Markerless deformable mesh tracking for human shape and motion capture. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages XX–XX, Minneapolis, USA, June 2007. IEEE, IEEE.

[8] A. Del Bue, F. Smeraldi, and L. Agapito. Non-rigid structure from motion using ranklet–based tracking and non-linear optimization. *Image and Vision Computing*, 25 (3):297–310, March 2007.

[9] Alessio Del Bue and Lourdes Agapito. Non-rigid stereo factorization. *Int. J. Comput. Vision*, 66:193–207, February 2006. ISSN 0920-5691. doi: 10.1007/s11263-005-3958-5.

[10] N. 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.

[11] A. Hilsmann and P. Eisert. Realistic cloth augmentation in single view video. In *Vision, Modeling, and Visualization Workshop 2009*, Braunschweig, Germany, November 2009.

[12] 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*, volume 6376 of *Lecture Notes in Computer Science*, pages 543–552. 2010.

[13] Niloy J. Mitra, Simon Flory, Maks Ovsjanikov, Natasha Gelfand, Leonidas Guibas, and Helmut Pottmann. Dynamic geometry registration. In *Symposium on Geometry Processing*, pages 173–182, 2007.

[14] Les Piegl and Wayne Tiller. *The NURBS book (2nd ed.)*. Springer-Verlag New York, Inc., New York, NY, USA, 1997. ISBN 3-540-61545-8.

[15] Bodo Rosenhahn, Uwe Kersting, Katie Powell, Reinhard Klette, Gisela Klette, and Hans-Peter Seidel. A system for articulated tracking incorporating a clothing model. *Mach. Vision Appl.*, 18:25–40, January 2007. ISSN 0932-8092. doi: 10.1007/s00138-006-0046-y.

[16] Mathieu Salzmann, Richard Hartley, and Pascal Fua. Convex optimization for deformable surface 3-d tracking. In *ICCV'07*, pages 1–8, 2007.

[17] 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.

[18] Shuhan Shen, Yinqiang Zheng, and Yuncai Liu. Deformable surface stereo tracking-by-detection using second order cone programming. In *ICPR*, pages 1–4. IEEE, 2008. ISBN 978-1-4244-2175-6.

[19] Shuhan Shen, Wenjuan Ma, Wenhuan Shi, and Yuncai Liu. Convex optimization for nonrigid stereo reconstruction. *Trans. Img. Proc.*, 19:782–794, March 2010. ISSN 1057-7149.

[20] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *Int. J. Comput. Vision*, 9:137–154, November 1992. ISSN 0920-5691. doi: 10.1007/BF00129684.

[21] 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. ISBN 978-3-540-88689-1.