Towards On-Line Intensity-Based Surface Recovery from Monocular Images

Oliver Ruepp¹ ruepp@in.tum.de Darius Burschka¹ burschka@cs.tum.edu Robert Bauernschmitt²

bauernschmitt@dhm.mhn.de

- ¹ Institut für Informatik VI Technische Universität München Boltzmannstraße 3 85748 Garching, Germany
- ² Deutsches Herzzentrum München Lazarettstr. 36 80636 München, Germany

Abstract

We present a novel method for vision-based recovery of three-dimensional structures through simultaneous model reconstruction and camera position tracking from monocular images. Our approach does not rely on robust feature detecting schemes (such as SIFT, Good Features to Track etc.), but works directly on intensity values in the captured images. Thus, it is well-suited for reconstruction of surfaces that exhibit only little texture due to partial homogeneity of the surfaces.

1 Introduction

Tracking and reconstruction of surfaces from video data is a problem that has been subject of extensive research work, and a number of methods exist for this problem. Many of the established methods, however, rely on presence of salient image features, such as SIFT [11] features, Good Features to Track [22], FAST corner detection [12] and so on. In some settings, however, the objects one is dealing with do not exhibit much structure, which makes it very hard to find robust, dense feature sets using traditional methods. In such situations, it pays off to use intensity-based methods, which is what we have investigated.

Originally, our idea was to generalize an approach developed by Ramey *et al.* [13] for efficient tracking of the disparity map in stereo video streams. Their method is quite general in that it can be used in conjunction with arbitrary parametric models of disparity maps, and it is especially efficient if the model is linear in parameters. In their test setups, they have used a B-Spline surface to represent the disparity map. We wanted to generalize their approach in the sense that the cameras do not need to be mounted on a stereo rig, but instead they are allowed to move independently from each other.

As an intermediate step towards this goal, we developed the method presented in this paper, which allows simultaneous model reconstruction and camera localization from monocular images in static scenes. In comparison to the two-camera scenario described above, this is equivalent to a situation where two cameras are present, but only one of them is moving, and the observed scene is static.



Figure 1: Schematic overview of the main idea of our algorithm.

Some examples of real-time dense reconstruction methods are described in papers by Palaanen *et al.* [13], Pan et al. [13], and in the recent work by Newcombe and Davison [13]. All of these methods have in common, however, that they rely on some kind of feature detecting scheme, which is what we want to avoid here.

A number of offline methods for model-based bundle-adjustment have been described with applications to face modeling $[\square, \square]$. Our method is different in that it tries to build the model during run-time, starting out with a very crude initial model (a plane) and refining the model in each step.

A part of the problem of surface reconstruction from image intensities is the surface modeling and reconstruction methodology itself. A thorough treatment of that problem has been done recently $[\square, \square]$, and results have been established using feature-based methods. Finally, Salzmann *et al.* have developed a method $[\square \square]$ for the closely related problem of monocular surface deformation recovery for weakly textured surfaces.

Ramey's tracking method [13] that inspired our development basically employs the Gauss-Newton minimization algorithm for tracking. The generalization that we have performed leads to an optimization problem that corresponds to intensity-based bundle-adjustment that is restricted to two frames. Thus, our solution shares some characteristics with typical bundle-adjustment algorithms. An in-depth survey of the original bundle-adjustment method is given in the book by Hartley and Zisserman [8]. The paper by Triggs *et al.* [23] provides a good overview of bundle adjustment variants and related methods. There is also a more recent paper evaluating the status of real-time bundle adjustment methods [9].

We are interested in recovering the surface of a 3D object on-line from a stream of monocular camera images. The surface we want to reconstruct must be static. Furthermore, since we are also tracking the object of interest, it is required that during the video sequence, sight of the object is not lost. Occlusions or self-occlusions are, until now, not accounted for. However, such problems have already been examined by other researchers, *e.g.* [**D**, **I**], and we expect it to be possible to incorporate similar techniques into our solution.

The basic concept of the algorithm is visualized in Figure 1. It can be summarized as follows: In traditional bundle adjustment, coordinates of 3D points that are associated with feature points are recovered from a set of 2D feature position measurements. This approach



Figure 2: Left, middle: Surface under two different camera positions. Right: Warping of surface coordinates from left to right image.

will obviously work only if a feature detecting scheme can be used at all. It has the advantage that the images can be taken from very different camera positions. In our case, we do not assume that robust feature extraction is possible, and thus we do not work with 2D positions, but with image intensities. This is only feasible if the camera positions of subsequent images are not too far away from each other.

In the next section, we give a detailed explanation of the method we have developed. Results have been obtained from real world data sets as well as synthetic data sets, and are presented in Section 3.

2 Method

2.1 Overview

There are many possibilities for representing a model of a scene, with the most straightforward one being a point cloud. This is a very general representation that is actually used in the traditional bundle adjustment algorithm, where it works well under the assumption that points can be reliably identified. Unfortunately, this assumption can not be used for intensity-based methods, since identifying a point based on its intensity is obviously bound to fail. This disqualifies the point cloud model for our purposes.

The usual approach taken to address this problem is the introduction of additional constraints in form of a parametric surface model of type $S : \mathbb{R}^k \times \mathbb{R}^2 \to \mathbb{R}^3$, on which the points lie. Mathematically speaking, *S* maps a set of *k* parameters together with surface coordinates *u*, *v* to three-dimensional spatial coordinates. Such a model is especially suitable for representation of scenarios that can be described with a small parameter set. Compared to the point cloud representation, it constitutes a loss of generality, but this is a compromise that seems to be necessary to make.

Inspired by the method of Ramey *et al.* $[\[La]\]$, we do not directly model the scene as a 3D surface. Instead, we choose the model to be a depth map of some object of interest for some reference image of the video stream. A 3D surface model can easily be retrieved from that representation, as will be shown later.

Observing a static, three-dimensional smooth surface *S* under two different camera positions will essentially yield two images that are related to each other via a "warping" function. If, for two snapshots of a scene, we exactly know the corresponding extrinsic camera param-

eters and we have a perfect mathematical description of the surface that we are observing, we can, for each surface pixel in one image, determine the position of that pixel in the other image. In other words, we can formulate a function of type $\mathbb{R}^2 \to \mathbb{R}^2$ that transforms pixel coordinates from one image to another, and we would expect the corresponding image values to be equal. Figure 2 shows an example for the warping function.

Because we require the surface to be completely visible at all times, it would not make sense to try to establish a depth map for the whole image, points could very easily be lost immediately after initialization as a result of minor camera movements. Instead, if we focus on only a certain region of interest within the image, it is easier for the user to assure that that region is always visible. Thus, before starting the actual reconstruction process, we have the user choose such an area within a reference image.

2.2 Mathematical Model

We do not take into account all pixels in the region of interest because the optimization process is quite costly. Instead, we only focus on a number of reference pixels that are selected according to a weak criterium that will be described later. These pixels are picked from a user-defined region of interest in a reference image and tracked through the entire image sequence.

As we have mentioned earlier, we are modeling the depth map of the region of interest that has been chosen by the user. That depth map is then a function $S_d(u,v)$ mapping a *k*-dimensional parameter vector **d** together with image coordinates $(u, v) \in \mathbb{R}^2$ to a depth value $\lambda \in \mathbb{R}$ at the specified coordinate. Given intrinsic camera parameters, this depth map can actually be interpreted as a 3D surface. Before we start to derive the image warping function, we want to give an overview of definitions and notations. In the following, images are numbered consecutively, and the numbering starts with n = 0. Then, let

- **d**_n denote the *k*-dimensional vector of parameters of the model describing the depth map.
- S_d(u,v) denote a function of type ℝ^k × ℝ² → ℝ that maps model parameters together with image pixel coordinates to 1D pixel depth values.
- **p**_n = (**t**_n, **q**_n) denote the extrinsic camera parameters corresponding to image *n*, consisting of translation vector **t**_n ∈ ℝ³ and rotation quaternion **q**_n ∈ ℝ⁴.
- *T*(**t**, **q**, **p**) : ℝ³ × ℝ⁴ × ℝ³ → ℝ³ is a transformation mapping 3D spatial coordinates **p** to 3D coordinates in the camera frame described by a translation vector **t** and a rotation quaternion **q**.
- π(p) be the projection of a 3D point p to 2D image coordinates, according to the internal camera calibration parameters of the camera used.
- $I_n(x,y)$ be the image function of image *n*, containing all pixel values. I_0 is hence the reference image function.
- $(u_1, v_1), \dots, (u_m, v_m)$ denote the pixel coordinates of the *m* reference pixels, chosen from the ROI in the reference image.

For the monocular camera, we assume a pinhole model with projection function

$$\boldsymbol{\pi}(\mathbf{p}) = \left(\frac{\mathbf{p}_1 f_x}{\mathbf{p}_3} + c_x, \frac{\mathbf{p}_2 f_y}{\mathbf{p}_3} + c_y\right)^T \tag{1}$$

where f_x, f_y are focal lengths in terms of pixel dimensions, c_x, c_y describe the location of the camera center, and $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)^T$ is a vector of Cartesian point coordinates. In case of significant radial distortions, the images can be rectified before usage.

If we associate the camera frame in image 0 with the reference frame, each pixel of the region of interest corresponds to a ray originating from the camera position (which coincides with the origin) that intersects the object surface at a certain depth. The pixel color then corresponds (ignoring possible specularities) to the color of the surface texture at that position. The ray corresponding to pixel coordinates (u, v) can then be parameterized by depth λ , yielding a function $r_{u,v}(\lambda)$:

$$r_{u,v}(\lambda) = \lambda \cdot \left(\frac{u - c_x}{f_x}, \frac{v - c_y}{f_y}, 1\right)^T.$$
(2)

It is obvious then that the composite function $r_{u,v}(S_d(u,v))$ is a description of the threedimensional model shape. If that model is observed from a different camera position \mathbf{p}_n , yielding a different image with index *n*, we need to rotate and translate the 3D coordinates produced by above function. This can be achieved by using the formula $T(\mathbf{p}_n, r_{u,v}(S_d(u,v)))$.

If we knew the perfect model parameters **d** and exact camera parameters \mathbf{p}_n for image *n*, we would expect the relationship $I_n(\pi(T(\mathbf{p}_n, r_{u,v}(S_{\mathbf{d}}(u, v)))) = I_0(u, v))$ to hold for all model surface coordinates (u, v).

Thus, we assume that the correct camera position and the correct model parameters together minimize some difference measure c (*e.g.* least squares) on intensity values, which can be formulated as

$$c(I_n(\pi(T(\mathbf{p}_n, r_{u,v}(S_{\mathbf{d}}(u, v))))) - I_0(u, v)).$$
(3)

As has been mentioned before, the optimization process necessary for determining camera and model parameters is quite computationally intensive. Thus, we will not include all possible pixel (u, v) coordinates in the optimization process, but only the coordinates of *m* chosen reference points. The corresponding objective function $o(\mathbf{d}, \mathbf{p}_n)$ can then be defined as

$$o(\mathbf{d}, \mathbf{p}_n) = \sum_{i=1}^m \left(c(I_n(\pi(T(\mathbf{p}_i, r_{u_i, v_i}(S_{\mathbf{d}}(u_i, v_i))) - I_0(u_i, v_i)))^2 \right)$$
(4)

Our problem of finding a warping function from the template image I_0 to the current image I_n could then be stated as the problem of minimizing the error function with respect to camera and depth map parameters.

There are two minor issues that we should also address: Because quaternions are used to represent the rotation of the camera frame, we need to constrain the corresponding parameters \mathbf{q}_n to represent a unit quaternion, and thus, a unit vector. This can trivially be formulated as a constraint $h_1(\mathbf{q}_n) = 0$ with $h_1(\mathbf{q}_n) = |\mathbf{q}_n|^2 - 1$. Furthermore, it is well-known that reconstruction from monocular images can only be done up to scale. However, it is desirable then at least to enforce a constant scale during the reconstruction process. This can be achieved with the formulation of a constraint $h_2(\mathbf{d}_n) = 0$ with $h_2(\mathbf{d}_n) = S_{\mathbf{d}_n}(u_1, v_1) - l$ for some constant *l*.

Since through optimizing above function, we implicitly try to track point positions through intensity values, our approach will have difficulties tracking points in areas with completely homogeneous intensity. Thus, wherever possible, the reference points are chosen from the ROI in such a way that they lie at pixel positions where the image derivative is non-zero.

Furthermore, reference points should be distributed in the region of interest such that the parameters determining the depth map are well constrained. For a B-Spline depth map model, one will, *e.g.*, need at least a number of reference points that is equal to the number of control points used.

2.3 Optimization Method

It is clear that, to actually recover the model parameters from the scene, we need some method to minimize the cost function described above. Since we are dealing with a constrained problem, an adequate method for optimization is Sequential Quadratic Programming (SQP). For a more detailed description of the method, the reader is referred to [12].

The basic idea is as follows: Let $f : \mathbb{R}^k \to \mathbb{R}$ be the scalar function to be minimized, and let $h : \mathbb{R}^k \to \mathbb{R}^l$ be a function that describes a constraint of the form h(x) = 0 on solutions. It is well-known that for such problems, the so-called Karush-Kuhn-Tucker (KKT) conditions must hold for any value x^* that is a minimum. These conditions can be formulated in equation form as:

$$\begin{pmatrix} \nabla \mathscr{L}(x,\lambda) \\ h(x) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \text{with} \quad \mathscr{L}(x,\lambda) = f(x) + \lambda^T h(x). \tag{5}$$

The term $\lambda \in \mathbb{R}^l$ is the Lagrange multiplier associated with the minimum. This is, in general, a nonlinear system of equations. The Lagrange-Newton-Method can be applied to these equations, and we can compute an update Δx to x and a new Lagrange multiplier λ^+ by solving the equation system

$$\begin{pmatrix} \nabla_{xx}^2 \mathscr{L}(x,\lambda) & \nabla_x h(x) \\ \nabla_x h(x)^T & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \lambda^+ \end{pmatrix} = - \begin{pmatrix} \nabla_x f(x) \\ h(x) \end{pmatrix}.$$
 (6)

Ultimately, we need to compute the Hessians $\nabla_{xx}^2 \mathscr{L}$ as well as the transposed Jacobian $\nabla_x h$ of *h*. Since *f* is, in our case, a quite complex composition of multi-dimensional functions, it is not straightforward to compute the full precision Hessian. Instead, it is common practice to use the Gauss-Newton approximation of the Hessian, as detailed below.

In our case, the objective function f is the composition $c \circ g$ of a scalar cost function c with some multi-dimensional comparison function g. The cost function could, *e.g.*, be the least-squares cost $g(x) = x^T x$, but since we also want our optimization to be robust against outliers, we will use something more robust, like the Pseudo-Huber [**B**, p. 619] cost function. In any case, the Hessian approximation that we are going to use is:

$$\nabla^2_{xx}(c \circ g)(x) \approx (\nabla_x g)(x) \cdot (\nabla^2_{xx} c)(f(x)) \cdot (\nabla_x g)^T(x)$$

A technique introduced for the popular method of Levenberg-Marquardt optimization $[\Box, \Box]$ is addition of a damping term λI to the Hessian. This allows the method to interpolate between Gauss-Newton and gradient descent steps, and greatly enhances the robustness of the method. This idea has been applied with success to the SQP method, *e.g.*, in the work by Gong and Xu $[\Box]$.

The Jacobians of f and h are computed using mainly Automatic Differentiation [**D**]. The only exception for this is the image function, which is interpolated and derivatives are



Figure 3: Left: Sample from artificial sequence, Right: Comparison by Normalized Cross-Correlation of surface parameters.

computed by hand. All of the involved matrices exhibit a high degree of sparsity. After computation of the Jacobian is finished, the approximate Hessian can be evaluated and the QP system is solved repeatedly using a sparse LDL^T Cholesky transformation on the whole system. Our SQP algorithm has been implemented using the efficient Eigen ¹ linear algebra library. As has been indicated above, we have used the Pseudo-Huber cost function in all of our experiments.

2.4 Dealing with Large Displacements

After we had implemented the optimization process as described above, it was evaluated on some image sequences. We found out that it works well on image sequences where camera movement is sufficiently smooth and no large pixel displacements occur between subsequent frames. However, problems occured when that was not the case. This was to be expected, since the algorithm operates on intensity values and will have trouble aligning with the correct values again if they are too far away.

The typical way to deal with this would be a pyramidal approach: One could start with the optimization on a coarse scale, and then move up to finer scales. This idea could probably be incorporated into our optimization approach. However, the idea has also been used by Lucas and Kanade [1] for their optical flow algorithm, which is well-established and implementations of which are readily available.

Thus, instead of incorporating the pyramidal approach directly into our method, we for now chose to implement a two-step technique: The first step when optimizing the model and aligning it to a new image would be to compute the optical flow between the previous image and the current image and perform optimization based solely on the 2D pixel coordinates of the reference points. The point position estimates derived from the optical flow algorithm shall in the following be denoted by (u'_i, v'_i) . The objective function that we use for that optimization is just a simplified version of the cost function for the intensity based optimization, namely

$$\sum_{i=1}^{m} c'((u'_i, v'_i) - \pi(T(\mathbf{p}_n, r_{u_i, v_i}(S_{\mathbf{d}}(u_i, v_i)))))$$
(7)

This is basically the original objective function, with the mapping from 2D coordinates to image intensities removed and with a different cost function c' instead of c. The cost function we used was the robust Pseudo-Huber cost function [**B**, p. 619]. The optimization is performed only with respect to camera parameters, since outliers in optical flow are quite common and tend to significantly disturb surface parameters in a full optimization step.

In the next step, we apply the original intensity based optimization process to realign the points to the reference intensity values and further optimize the surface parameters. This essentially prevents drifting away from the original point intensity values, which could easily occur over time if only optical-flow based optimization was used.

2.5 Algorithm Summary

We conclude the presentation of our simultaneous reconstruction and localization method with a step-by-step description of our algorithm:

- 1. Have the user choose a ROI from the first image of a video stream.
- Initialize the system with a run of coordinate-based optimization of camera and surface parameters based on optical flow data from the first two frames, and refine the fit by employing intensity-based optimization.
- 3. Acquire a new frame.
- 4. Perform one step of optical-flow based optimization to find the camera parameters for the new frame.
- 5. Refine fit using intensity-based optimization.
- 6. Go back to step 3.

3 Results

We have tested our algorithm on a set of artificial rendered image sequences, as well as on sequences of real scenes. The artificial data set was useful for generating images with known ground truth, while the sequences of real images have been used to show that the approach also works in the "real world." As depth map model, we have used B-Spline surfaces of varying order and complexity.

Our first tests were on artificial images generated by a renderer. Here, we show results for one of the used sequences. Figure 3 shows an example image from the sequence composed of renderings of a surface with a very difficult to track texture. Because we wanted to get a rough idea of how well traditional approaches would work on that sequence, we ran a SIFT feature detector on some of the images. The feature detection process resulted in about 20 features, depending on the actual image. Even when assuming that all features can be reliably identified through the whole sequence, and that no false feature matchings occur, this is by far not enough to fully describe the complexity of the actual surface. The surface is a quadratic spline surface determined by 25 control points (5 in each direction).

The right image of Figure 3 visualizes the reconstruction quality achieved by our algorithm as compared to the ground truth of the artificial sequence. It shows the difference



Figure 4: Reconstruction with real camera data. Left: Template image, ROI indicated by green rectangle. Right: Reconstruction result.

(measured by normalized cross correlation, since the reconstruction is only up to scale) between the surface parameters determined by our algorithm and the ground truth used by the renderer. The reconstruction can be seen to be pretty accurate, maintaining a NCC factor of over 0.96 during the entire reconstruction period.

The artificial sequences have been used because it is rather difficult in a real-world scenario to determine the ground truth. Still, it is important to show that our approach also works on actual data generated from a camera. Hence, we have tested our method an scene that was showing a piece of white cloth draped over a cup. You can see one image of the recorded sequence in Figure 4 as well as a rendering of the resulting 3D model.

Due to the piece of cloth being quite wrinkled, we were actually expecting more difficulties in reconstructing the real-world scene. However, we have seen that a spline surface with only 8×8 control points was already enough to model the scene.

As for running times: Our algorithm has been tried on a system with a 1.86 GHz dual core CPU. Using only one of the two CPU cores, framerates of about 10 frames per second were achieved. The performance is promising, and we expect it to be possible to further improve performance, *e.g.*, by utilizing GPU hardware.

4 Conclusion

The basis for further research has been established with our monocular model recovery and validation algorithm. There are many possible extensions and improvements to this technique.

First of all, while the reference-point based reconstruction works surprisingly well, it would probably constitute a major improvement if we were able to capture, in addition to point intensity values, some characteristics of the surface texture surrounding a reference point, thus introducing a patch-based correlation function. We would expect this to improve the stability and convergence speed of the optimization method considerably.

Furthermore, we did not address the issue of changing illumination conditions. We would like to be able to deal with changes in brightness, but also with specularities, which would, in the current approach, both cause severe problems. However, some techniques for dealing with problems of that kind have already been developed, *e.g.*, normalized cross-correlation matching for brightness-invariant matching. It should be possible to integrate them into our

method.

We would also like to extend the approach such that deformable surfaces can be reconstructed and tracked. For tackling this problem, we intend to use a setup of two independently moving cameras. Based on such an idea, we would like to introduce a method for determining deformation parameters, allowing us also to predict and simulate deformations. We see applications for such a technique mainly in medical imaging.

References

- Adrien Bartoli, Mathieu Perriollat, and Sylvie Chambon. Generalized thin-plate spline warps. *Int. J. Comput. Vision*, 88(1):85–110, 2010. ISSN 0920-5691. doi: http://dx. doi.org/10.1007/s11263-009-0303-4.
- [2] F. Brunet, A. Bartoli, N. Navab, and R. Malgouyres. Nurbs warps. In *British Machine Vision Conference (BMVC)*, London, September 2009.
- [3] C. Engels, H. Stewénius, and D. Nistér. Bundle adjustment rules. In *Photogrammetric Computer Vision (PCV)*. ISPRS, September 2006.
- [4] P. Fua. Using model-driven bundle-adjustment to model heads from raw video sequences. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 46–53 vol.1, 1999. doi: 10.1109/ICCV.1999.791196.
- [5] Vincent Gay-Bellile, Adrien Bartoli, and Patrick Sayd. Direct estimation of nonrigid registrations with image-based self-occlusion reasoning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(1):87–104, 2010.
- [6] Rubin Gong and Gang Xu. Quadratic surface reconstruction from multiple views using sqp. pages 197–217, 2004.
- [7] Andreas Griewank and Andrea Walther. Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. Number 105 in Other Titles in Applied Mathematics. SIAM, Philadelphia, PA, 2nd edition, 2008. ISBN 978–0–898716–59–7.
- [8] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
- [9] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly Journal of Applied Mathmatics*, II(2):164–168, 1944.
- [10] David G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157, 1999.
- [11] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision (darpa). In *Proceedings of the 1981 DARPA Image Understanding Workshop*, pages 121–130, April 1981.
- [12] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. SIAM Journal on Applied Mathematics, 11(2):431–441, 1963. doi: 10.1137/ 0111030. URL http://link.aip.org/link/?SMM/11/431/1.

- [13] Richard Newcombe and Andrew Davison. Live dense reconstruction with a single moving camera. In 2010 IEEE Conference on Computer Vision and Pattern Recognition CVPR'10, 2010.
- [14] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, August 2000. ISBN 0387987932.
- [15] Pekka Paalanen, Ville Kyrki, and Joni-Kristian Kamarainen. Towards Monocular On-Line 3D Reconstruction. In Workshop on Vision in Action: Efficient strategies for cognitive agents in complex environments, Marseille France, 2008. Markus Vincze and Danica Kragic and Darius Burschka and Antonis Argyros. URL http: //hal.inria.fr/inria-00325795/en/.
- [16] Q. Pan, G. Reitmayr, and T. Drummond. ProFORMA: Probabilistic Feature-based On-line Rapid Model Acquisition. In Proc. 20th British Machine Vision Conference (BMVC), London, September 2009.
- [17] D. Pizarro and A. Bartoli. Feature-based non-rigid surface detection with self-occlusion reasoning. In *Fifth International Symposium on 3D Data Processing, Visualization and Transmission*, May 2010.
- [18] Nicholas A. Ramey, Jason J. Corso, William W. Lau, Darius Burschka, and Gregory D. Hager. Real Time 3D Surface Tracking and Its Applications. In *Proceedings of Workshop on Real-time 3D Sensors and Their Use (at CVPR 2004)*, 2004.
- [19] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006. doi: 10.1007/11744023_34.
- [20] Mathieu Salzmann, Raquel Urtasun, and Pascal Fua. Local deformation models for monocular 3d shape recovery. In CVPR. IEEE Computer Society, 2008.
- [21] Ying Shan, Zicheng Liu, and Zhengyou Zhang. Model-based bundle adjustment with application to face modeling. In *Computer Vision*, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on, volume 2, pages 644–651 vol.2, 2001. doi: 10.1109/ICCV.2001.937687.
- [22] Jianbo Shi and Carlo Tomasi. Good features to track. In 1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94), pages 593 – 600, 1994.
- [23] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *ICCV '99: Proceedings of the International Workshop on Vision Algorithms*, pages 298–372, London, UK, 2000. Springer-Verlag. ISBN 3-540-67973-1.