6-DoF Model-based Tracking of Arbitrarily Shaped 3D Objects

Pedram Azad[1], David Münch[2], Tamim Asfour[1], Rüdiger Dillmann[1]

[1]*Institute for Anthropomatics, Karlsruhe Institute of Technology, Germany*
[2]*Fraunhofer IOSB, Ettlingen, Germany*

azad@kit.edu, *david.muench@iosb.fraunhofer.de*, *asfour@kit.edu*, *dillmann@kit.edu*

***Abstract*— Image-based 6-DoF pose estimation of arbitrarily shaped 3D objects based on their shape is a rarely studied problem. Most existing image-based methods for pose estimation either exploit textural information in form of local features or, if shape-based, rely on the extraction of straight line segments or other primitives. Straight-forward extensions of 2D approaches are potentially more general, but in practice assume a limited range of possible view angles. The general problem is that a 3D object can potentially produce completely different 2D projections depending on its relative pose to the observing camera. One way to reduce the solution space is to exploit temporal information, i.e. perform tracking. Again, existing model-based tracking approaches rely on relatively simple object geometries. In this paper, we propose a particle filter based tracking approach that can deal with arbitrary shapes and arbitrary or even no texture, i.e. it offers a general solution to the rigid object tracking problem. As our approach can deal with occlusions, it is in particular of interest in the context of goal-directed imitation learning involving the observation of object manipulations. Results of simulation experiments as well as real-world experiments with different object types prove the practical applicability of our approach.**

I. Introduction

Image-based pose estimation of 3D objects has so far been addressed mostly for specific object geometries. It is an accepted fact that 3D shapes can in general not be represented fully by a single 2D representation [1]. The reason is that a 3D object can potentially produce completely different 2D projections depending on its relative pose to the camera, as illustrated in Fig. 1. Thus, for 6-DoF pose estimation some kind of 3D model is required, i.e. for instance image features associated with 3D positions or a full 3D object model.

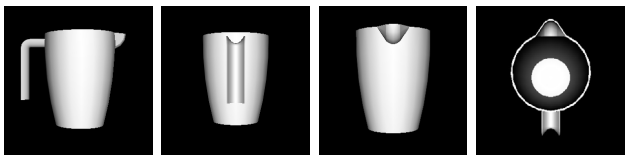

Fig. 1. Different views of a measuring cup.

Traditional edge-based recognition and in particular tracking methods rely on rather simple 3D models, as illustrated in Fig. 2. Initial work in this context has been performed by Lowe in 1987 [2] and later for the purpose of tracking in [3]. It is typical for this type of approaches to rely on straight line segments as primitives, as they allow simple and efficient 2D-image to 3D-model matching. Either the model edges are sampled and 2D-point-to-3D-line correspondences ([4]) or 2D-point-to-3D-point correspondences (e.g. [5], which requires POSIT [6] or a similar method for computing the pose estimate) are established, or straight line segments are extracted from the image in a pre-processing step (e.g. [7]). The determined matches are usually used as input to a least squares based optimization method.

Fig. 2. Typical 3D model for model-based tracking composed of straight line segments, as used in [8].

An alternative to applying an optimization method is to perform a search for the correct pose, which seeks to maximize a similarity function between the projected instance of the 3D model and the 2D image. Such a similarity function can be formulated on the basis of line-to-line correspondences, as done in [8], or the model edges can be sampled. The search itself is often performed within a particle filter as statistical framework, as done e.g. in [9].

When dealing with more complex shapes, however, such approaches become inapplicable. The reason is that in particular curved surfaces can be modeled accurately only by a multiplicity of polygons (or as freeform surfaces, e.g. NURBS), so that the real edges of the object cannot be expressed by straight line segments of the model (see Fig. 3). In other words, a priori knowledge of the real edges is not available: It depends on the pose of the object, which polygon edges become real edges.

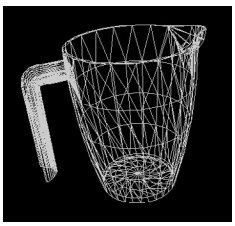

Fig. 3. 3D wireframe model of measuring cup.

2D tracking approaches model the appearance of an object by 2D contours, using active contours or active shape models. However, the 6-DoF pose cannot be derived accurately on the basis of deformable 2D contours directly.

Since 2000, local appearance-based methods using point features (e.g. [10], [11], [12]) or region-based features (e.g. [13]) have become popular. However, such approaches are only applicable for objects that exhibit such local features, either computed by blob detectors on the basis of a scale space analysis or by corner detectors. For single-colored objects (see e.g. Fig. 1) this is not the case.

Global appearance-based approaches model the object by a set of global views. So-called *canonical views* have been proposed for representing an object by a reduced number of views that are sufficient to cover all possible appearances of the object [14], [15]. A suitable data structure for storing and arranging such views is an aspect graph; a survey is given in [16]. However, such representations are mainly used for recognition; accurate 6-DoF pose estimation requires more information than matching of canonical views can provide.

In our previous work [17], we have developed an approach for shape-based object recognition and pose estimation, which is based on matching of global views, stereo triangulation, and online rendering. The object's appearances are represented by a dense view set consisting of several thousand views. While being able to compute the full pose given a single stereo image pair within less than 20 ms, global segmentation of the object of interest is required and therefore object occlusions cannot be handled.

We will motivate this work and provide a problem definition in Section II. Our proposed approach is presented in detail in Section III. Results of simulation experiments as well as real-world experiments are provided in Section IV, ending with a conclusion in Section V.

II. Motivation and Problem Definition

On our humanoid robot ARMAR-III [18], our object recognition and pose estimation approach proposed in [17] is successfully applied for single-colored objects in the context of manipulation tasks. As already mentioned, it requires global segmentation and cannot deal with occlusions. Therefore, the intention is to provide with this work a tracking approach that starts tracking after an initial pose has been computed with our approach from [17]. Once tracking has started, e.g. complex object manipulations (which potentially lead to occlusions) can be performed by a human and observed by the robot by using the approach presented in this work.

Therefore, in this work, we focussed on the problem of 6-DoF rigid object tracking, given an accurate 3D model of the object of interest. We do not make any assumptions about the object, neither on its texture, nor on its shape, nor on its color. The only requirement on the 3D model is that it can be rendered by a 3D engine.

Note that the objective is not a *relative* pose estimator, which is often implemented using the KLT tracker [19] for propagating 2D-3D feature correspondences. Such approaches have two drawbacks: they require the presence of local features and pose estimation errors accumulate. In contrast, the objective is an *absolute* model-based pose estimator that does not assume the presence of local features.

Throughout our experiments, we used object models from the object modeling system presented in [20][1]. In our implementation, the 3D models were rendered using OpenGL.

III. Proposed Approach

The core idea of our approach is to perform an appearance-based matching based on *online* rendering of a finegrained *local* view space using a 3D model of the object.

In [17], we have shown that not only the orientation of a 3D object influences its appearance in the projected image, but also its position does. Computing a complete view space thus must involve all six degrees of freedom, i.e. be six-dimensional. For a global finegrained view space, this would go far beyond reasonable memory consumption and computation time.

Therefore, we propose to render only those views that are in the vicinity of the current pose estimate, which has to be done online. This is done within an annealed particle filter [21], which decides which object poses to render for each frame. As probability function of the particle filter an edge-based similarity function is used.

In the following, the involved steps of our approach are presented:

1) Pose update
2) Probability function
3) Normalization
4) Annealed Particle Filter

A. Pose Update

The object position is represented as a 3D translation vector $\boldsymbol{t} \in \mathbb{R}^3$. The orientation is represented as a three-dimensional vector $\boldsymbol{r} \in \mathbb{R}^3$, its direction specifying the rotation axis and its length $\|\boldsymbol{r}\|_2$ specifying the rotation angle – essentially the same as a quaternion representation reduced to three dimensions. In contrast to Euler angles, such a representation does not suffer from singularities, which would lead to tracking "dead ends".

We denote a pose as $\boldsymbol{p} \in \mathbb{R}^6$ with:

$$\boldsymbol{p} = \begin{pmatrix} \boldsymbol{r} \\ \boldsymbol{t} \end{pmatrix} \tag{1}$$

For particle sampling, the pose update for a given pose $\boldsymbol{p}_t$ from the previous time step t is computed as follows:

$$\boldsymbol{p}_{t+1} = \boldsymbol{p}_t + B_n \cdot \boldsymbol{\omega} \tag{2}$$

where $\boldsymbol{\omega} \in \mathbb{R}^6$ is a vector that in each of its components contains independent Gaussian noise. $B_n \in \mathbb{R}^{6\times 6}$ is a diagonal matrix containing amplification weights. The index n in B_t indicates that adaptive weights are applied, depending on the current layer n of the particle filter (see Section III-D).

[1] http://i61p109.itec.uni-karlsruhe.de/ObjectModelsWebUI/

B. Probability Function

The evaluation function, on the basis of which the a posteriori probabilities are computed, builds the core of any particle filter. As we want to deal with arbitrary shapes, we cannot assume an object model in which polygon edges are directly related to real edges (see Section I). Our general solution is to render the object with the given pose and to treat the rendering result as an image. As the real edges will per definition be visible edges in the rendering result, we apply a gradient-based edge filter or edge detector to the rendering result to compute them. The filter result will be denoted as $I_{g,\boldsymbol{p}}$ (the index g indicating that it is a gradient image, the index $\boldsymbol{p}$ indicating that it is a rendered view for a particle pose). An example is shown in Fig. 4.

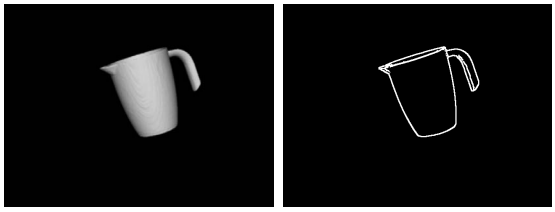

Fig. 4. Rendered and processed view for one particle. In this example, the Canny edge detector with a subsequent 3×3 morphological dilation operator was applied. Left: rendered view, given the pose of the particle and the 3D model. Right: edge image.

The current input image (see Fig. 5) is processed with the same edge filter that is used for processing the rendered per-particle object views, in order to achieve a similar filter response. The filter result will be denoted as I_g in the following.

Fig. 5. Input image and computed edge image. In this example, the Canny edge detector with a subsequent 3×3 morphological dilation operator was applied. Left: input image. Right: edge image.

In conventional model-based trackers using particle filters, a commonly used evaluation function for an edge-based cue is:

$$w'_g(I_g, P_{\boldsymbol{p}}) = 1 - \frac{1}{|P_{\boldsymbol{p}}|} \sum_{i=1}^{|P_{\boldsymbol{p}}|} I_g(\boldsymbol{p}_i) \tag{3}$$

where I_g denotes the edge image and $P_{\boldsymbol{p}}$ is a set of sampled and projected model points for a given pose $\boldsymbol{p}$ to be evaluated (see e.g. [22]). Here, we want to implement the same error measure, but now replacing the sampled point set $P_{\boldsymbol{p}}$ by the edge filtered rendered view $I_{g,\boldsymbol{p}}$. The reason for this replacement is that, as discussed earlier, with arbitrary shapes it is not possible to sample a priori known line segments for the visible edges. Instead, we propose to define an error measure based on the bitwise AND operation between the edge image I_g and the edge image of the rendered particle view $I_{g,\boldsymbol{p}}$:

$$w_g(I_g, I_{g,\boldsymbol{p}}) = 1 - \frac{\sum_{x,y} [I_g(x,y) \text{ AND } I_{g,\boldsymbol{p}}(x,y)]}{\sum_{x,y} I_{g,\boldsymbol{p}}(x,y)} \tag{4}$$

where the sums run over the complete image and $I_{g,\boldsymbol{p}}$ is a binary image with $I_{g,\boldsymbol{p}}(u,v) \in \{0, 255\}$ for all pixels (u,v). Note that this formulation of the error measure is equivalent to Eq. (3) in case the point set $P_{\boldsymbol{p}}$ is sampled with the discretization of 1 pixel.

In case of the Canny edge detector, both I_g and $I_{g,\boldsymbol{p}}$ are already binary. A subsequent morphological dilate operation is used to spread the edges so that the potentially overlapping area is increased. When using the Sobel filter instead, threshold binarization is performed to produce a binary edge image. An exemplary result of the bitwise AND operation is shown in Fig. 6.

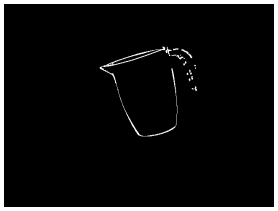

Fig. 6. Result of the bitwise AND operation (before summation) for the right images from the Figs. 4 and 5. As can be seen, the similarity of the particle pose and the real pose lead to a relatively large overlap.

The function computing the a-posteriori probabilities is

$$p_g(I_g \,|\, \boldsymbol{p}) \propto \left\{ -\frac{1}{2\sigma_g^2} w_g(I_g, I_{g,\boldsymbol{p}}) \right\} \tag{5}$$

where $\boldsymbol{p}$ denotes the pose of the particle to be evaluated and σ^2 denotes the system's variance. In practice, the term $\frac{1}{2\sigma_g^2}$ is interpreted as an either fixed or adaptive weighting factor, which affects the tracking behavior of the particle filter.

C. Normalization

The evaluation function from Eq. (4) normalizes the number of overlapping pixels with the total number of edge pixels of the rendered view. This is precisely the same normalization as in Eq. (3), where the number of overlapping pixels is normalized with the total number of sampled points. Depending on the object and its pose, this kind of normalization can potentially lead to ambiguities.

An exemplary situation of such an ambiguity is illustrated in Fig. 7. As can be seen the pose visualized in the bottom row is the correct pose. However, the wrong pose (top row) produces the slightly better rating when using Eq. (4). The reason is that both poses produce an almost complete overlap, but the filter response for the lower half ellipse for the

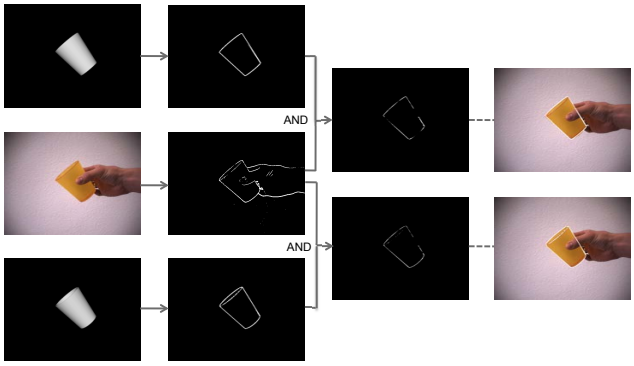

Fig. 7. Example of a situation that leads to an ambiguity when using Eq. (4) as evaluation function. The wrong pose (top row) produces a slightly better rating.

correct pose is relatively weak. This leads to a slightly higher matching percentage for the wrong pose, as in that case this half ellipse does not appear in the rendered view due to a self-occlusion.

If evaluating the total number of overlapping pixels instead of the percentage of the overlap, this problem does not occur. However, the maximum number of overlapping edge pixels can substantially vary, in particular depending on the distance of the object to the camera. This means that the input range of the exponential function would significantly vary depending on the pose, when applying constant weights. We therefore first collect all ratings, then linearly map these to the interval $[0, 1]$, and finally weight the mapped ratings and apply the exponential function:

$$\begin{aligned} w_i &:= -\sum_{x,y} [I_g(x,y) \text{ AND } I_{g,i}(x,y)] \\ w_{min} &:= \min_i w_i \\ w_{max} &:= \max_i w_i \\ \pi_i &:= \exp\left\{1 - \alpha \cdot \frac{w_i - w_{min}}{w_{max} - w_{min}}\right\} \end{aligned} \tag{6}$$

Throughout our experiments, $\alpha = 15$ was chosen as a constant weight.

D. Annealed Particle Filter

The essential idea of annealed particle filtering [21] is to apply several runs of a particle filter – so-called layers – to the same frame, but instead use less particles. In [21], it is claimed that the total number of evaluations can be reduced drastically, in particular by modifying the broadness of the probability function from broad to fine towards the last layer. In their human motion capture system with an articulated human model with 30 DoF, a conventional particle filter with 40,000 particles was replaced by an annealed particle filter with 10 layers and 200 particles, i.e. 2,000 evaluations instead of 40,000.

This drastic reduction must however be taken with care. What happens is that the coverage of the relevant state space is reduced considerably and the parameters of the layers are adjusted to achieve stronger convergence towards the last layer. By doing this, fewer hypotheses are generated in the early layers (which are refined in the later layers), as the number of particles is significantly less. Furthermore, from experience, applying annealed particle filtering leads to increased jitter in the estimated trajectories, in particular when there is a discrepancy between the number of particles and the dimensionality of the search space. One method for reducing jitter is to apply adaptive weights for particle sampling (matrix B_t in Eq. (2)), as proposed in [23]. There, the applied noise for a parameter was chosen to be proportional to the variance of that parameter.

As the variance of a parameter does not necessarily have to be related to an error of that parameter itself, we choose a different scheme. The degrees of freedom of the weighting matrix are weighted with a single scalar weight:

$$B_n = \frac{1}{n} B \tag{7}$$

where $n \geq 1$ is the number of the current layer and B is a diagonal matrix with constant weights. By doing this, the same number of particles is spread within a continuously decreasing space, which means that the granularity of the search becomes finer toward the final layer. The practical effect is that estimation noise is reduced. Throughout the performed experiments, three layers with 250 particles were used for the present 6-DoF tracking problem. The motivation for this layout is to use a reasonable amount of particles to cover the search space with two additional layers for pose refinement.

IV. Experimental Results

In the following we present the results of an experimental evaluation. The system was implemented with the Integrating Vision Toolkit (IVT)[2]. The processing times were measured for an optimized CPU and an optimized GPU implementation. The applied optimizations are described in Section IV-B.

A. Accuracy

We have performed experiments both with simulated and with real image data. As ground truth information for the real image sequences was not available, the accuracy was evaluated for the simulated images only. The snapshots from Fig. 9 show the success of our approach for simulated as well as real image data. The video attachment provides the complete image sequences.

The evaluation of the accuracy with simulated images was performed with three objects: a cup, a plate, and a measuring cup. The results are illustrated in Fig. 8. As can be seen the translational errors are below 5 mm for the x, y-coordinates and below 15 mm for the z-coordinate. The rotational errors are below 0.1 radians (5.7^o). The rotational part was compared with Euler angles to isolate the rotational symmetry axis of the cup and the plate. Note that this is independent from modeling the rotational part within the

[2] ivt.sourceforge.net

particle filter with an axis/angle representation, as explained in Section III-A.

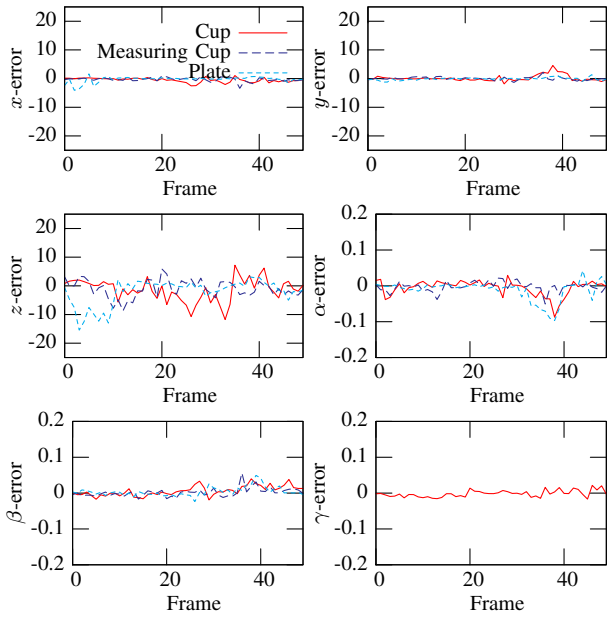

Fig. 8. Errors for all 6-DoF. The x, y, z-errors are given in mm, the α, β, γ-errors in radians. Note that the angle γ is not provided for the cup and the plate, as it is the angle around their rotational symmetry axis.

B. Runtime – CPU vs. GPU

We have optimized our approach both for the CPU and for the GPU and have performed comparative runtime experiments; the results are provided in the following.

To achieve maximum speedups on the CPU, we used the Keyetech Performance Primitives (KPP)[3] for the image processing routines. Although the KPP achieve a speedup for the image processing by a factor of 5, the total speedup is at most a factor of 2. The reason is that rendering and readback to main memory require approx 2.5 ms whereas the image processing with the KPP requires only approx. 0.5 ms.

Table I depicts the runtime for tracking the cup (see Fig. 9) for different numbers of evaluations. The processing times were measured on a standard PC with an Intel Core 2 Duo 3.0 GHz and an NVIDIA Tesla C1060. In the case of the CPU-version, the graphics card was used for rendering the views only.

Evaluations	CPU [s]	CPU optimized (KPP) [s]	GPU (CUDA) [s]
100	0.5	0.30	0.24
500	2.4	1.5	1.1
750	3.6	2.2	1.7
1000	5.7	2.9	2.3

TABLE I
RUNTIME OF CPU VS. GPU FOR DIFFERENT NUMBERS OF EVALUATIONS.

[3] www.keyetech.de

Fig. 9. Visualization of the tracking result of different tracked objects both with real-world experiments and experiments in simulation.

As reading back the framebuffer to the main memory is the bottleneck of the CPU implementation, the aim was to exploit NVIDIA's CUDA to avoid the transfer to main memory. All image processing routines were implemented with CUDA, as they are highly suitable for parallel processing. Since the rendering result is already stored in the GPU memory, the transfer from the framebuffer of the GPU to main memory of the host can be skipped. Instead, the rendered view is directly mapped into the memory space of CUDA on the GPU.

Figure 10 depicts the optimization process. For the image processing, the image is subdivided into tiles, and each part is computed by a group of threads called a thread block. To avoid concurrency errors, synchronization is required among all the thread blocks. Finally, the rating calculated for each pose is transferred back from the GPU to the CPU's main memory.

The image processing routines (edge detection, threshold binarization, bitwise AND operation, and pixel sum computation) perform slightly better than the CPU implementations offered by the KPP. However, our measurements have shown

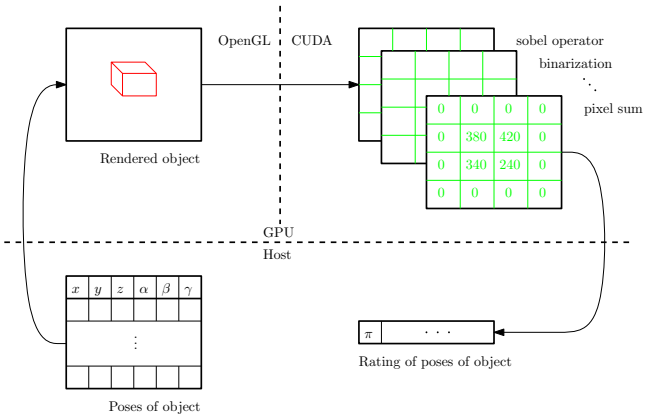

Fig. 10. From the pose of the object to its probability. Each view for a given object pose is rendered with OpenGL on the GPU. Then, the image of this rendered view is mapped into the memory space of CUDA. The image is divided into different blocks of the same size. On each block, several threads apply the Sobel operator, the binarization, the bitwise AND operation and the computation of the sum of the pixels. Finally, the calculated probability is transferred to the host.

that mapping the rendering result from the memory space of OpenGL into CUDA's memory space requires the same time as transferring it to the CPU's memory space. The reason is the framebuffer, which is the bottleneck due to its slow bandwidth. Therefore, our expectations for using CUDA to reduce the readback time significantly were not fulfilled.

V. Discussion and Outlook

We have presented an approach to 6-DoF tracking of arbitrarily shaped rigid objects. In contrast to conventional approaches, neither a simplified geometric model nor the presence of local point features (SIFT, SURF, etc.) is assumed. The only requirement is an accurate 3D polygon model that can be rendered by a graphics card. We have presented experimental results with simulated and real image data, with objects substantially differing in shape and texture. The approach is robust to occlusions and is therefore suitable for the observation of humans in realistic scenarios, such as imitation and interaction involving the manipulation of objects.

The universality of the approach comes at the price of a high computational effort. We have fully optimized our approach both for the CPU and GPU, and in both cases the bottleneck was the slow readback from the frame buffer. However, future graphics hardware development will at some point resolve this issue. We thus hope to make with this work an important step toward a universal 6-DoF rigid object tracker for real-time robotics applications. Future work will focus on the integration of additional cues to support application in the presence of complex cluttered backgrounds.

Acknowledgment

The work described in this paper was conducted within the German Humanoid Research project SFB588 funded by the German Research Foundation (DFG: Deutsche Forschungsgemeinschaft).

References

[1] D. G. Lowe, *Perceptual Organization and Visual Recogntion*. Kluwer Academic Publishers, 1985.

[2] ——, "Three-Dimensional Object Recognition from Single Two-Dimensional Images," *Artificial Intelligence*, vol. 31, no. 3, pp. 355–395, 1987.

[3] ——, "Robust Model-based Motion Tracking through the Integration of Search and Estimation," *International Journal of Computer Vision (IJCV)*, vol. 8, no. 2, pp. 113–122, 1992.

[4] C. G. Harris and C. Stennett, "3D object tracking at video rate – RAPiD," in *British Machine Vision Conference (BMVC)*, Oxford, UK, 1990, pp. 73–78.

[5] E. Marchand, P. Bouthemy, F. Chaumette, and V. Moreau, "Robust Real-Time Visual Tracking using a 2D-3D Model-based Approach," in *IEEE International Conference on Computer Vision (ICCV)*, Kerkyra, Greece, 1999, pp. 262–268.

[6] D. F. DeMenthon and L. S. Davis, "Model-Based Object Pose in 25 Lines of Code," *International Journal of Computer Vision (IJCV)*, vol. 15, no. 1-2, pp. 335–343, 1995.

[7] M. Armstrong and A. Zisserman, "Robust Object Tracking," in *Asian Conference on Computer Vision (ACCV)*, vol. 1, Singapore, 1995, pp. 58–61.

[8] Y. Yoon, A. Kosaka, J. B. Park, and A. C. Kak, "A New Approach to the Use of Edge Extremities for Model-based Object Tracking," in *IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain, 2005, pp. 1883–1889.

[9] G. Klein and D. Murray, "Full-3D Edge Tracking with a Particle Filter," in *British Machine Vision Conference (BMVC)*, vol. 3, Edinburgh, UK, 2006, pp. 1119–1128.

[10] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision (IJCV)*, vol. 60, no. 2, pp. 91–110, 2004.

[11] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded Up Robust Features," in *European Conference on Computer Vision (ECCV)*, Graz, Austria, 2006, pp. 404–417.

[12] P. Azad, T. Asfour, and R. Dillmann, "Combining Harris Interest Points and the SIFT Descriptor for Fast Scale-Invariant Object Recognition," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, USA, October 2009, pp. 4275–4280.

[13] S. Obdrzalek and J. Matas, "Object Recognition using Local Affine Frames on Distinguished Regions," in *British Machine Vision Conference (BMVC)*, vol. 1, Cardiff, UK, 2002, pp. 113–122.

[14] D. Weinshall and M. Werman, "On View Likelihood and Stability," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 19, no. 2, pp. 97–108, 1997.

[15] T. Denton, M. F. Demirci, J. Abrahamson, A. Shokoufandeh, and S. Dickinson, "Selecting Canonical Views for View-based 3-D Object Recognition," in *International Conference on Pattern Recognition (ICPR)*, Cambridge, UK, 2004, pp. 273–276.

[16] R. D. Schiffenbauer, "A Survey of Aspect Graphs," Polytechnic University, New York City, USA, Tech. Rep. TR-CIS-2001-01, 2001.

[17] P. Azad, T. Asfour, and R. Dillmann, "Accurate Shape-based 6-DoF Pose Estimation of Single-colored Objects," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, USA, October 2009, pp. 2690–2695.

[18] T. Asfour, K. Regenstein, P. Azad, J. Schröder, N. Vahrenkamp, and R. Dillmann, "ARMAR-III: An Integrated Humanoid Platform for Sensory-Motor Control," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, Genova, Italy, 2006, pp. 169–175.

[19] C. Tomasi and T. Kanade, "Detection and Tracking of Point Features," Carnegie Mellon University, Pittsburgh, USA, Tech. Rep. CMU-CS-91-132, 1991.

[20] R. Becher, P. Steinhaus, R. Zöllner, and R. Dillmann, "Design and Implementation of an Interactive Object Modelling System," in *Robotik/ISR*, Munich, Germany, 2006.

[21] J. Deutscher, A. Blake, and I. Reid, "Articulated Body Motion Capture by Annealed Particle Filtering," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Hilton Head, USA, 2000, pp. 2126–2133.

[22] P. Azad, *Visual Perception for Manipulation and Imitation in Humanoid Robots*. Berlin, Heidelberg, Germany: Springer, 2009.

[23] J. Deutscher, A. Davison, and I. Reid, "Automatic Partitioning of High Dimensional Search Spaces associated with Articulated Body Motion Capture," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Kauai, USA, 2001, pp. 669–676.