Tracking of industrial objects by using CAD models

Harald Wuest and Didier Stricker

Department of Virtual and Augmented Reality
Fraunhofer IGD
Darmstadt, Germany
phone: +49 (0)6151 155273
email: harald.wuest@igd.fraunhofer.de
www: www.igd.fhg.de/igd-a4/

Abstract

In this paper we present a model-based approach for real-time camera pose estimation in industrial scenarios. The line model which is used for tracking is generated by rendering a polygonal model and extracting contours out of the rendered scene. By un-projecting a point on the contour with the depth value stored in the z-buffer, the 3D coordinates of the contour can be calculated. For establishing 2D/3D correspondences the 3D control points on the contour are projected into the image and a perpendicular search for gradient maxima for every point on the contour is performed. Multiple hypotheses of 2D image points corresponding to a 3D control point make the pose estimation robust against ambiguous edges in the image.

Keywords: Augmented Reality, Tracking, CAD model, Line Tracking

First presented at the 3rd Workshop "Virtuelle und Erweiterte Realität der GI-Fachgruppe VR/AR", extended and revised for JVRB

1 Introduction

For an augmented reality application a tracking system for estimating the camera pose is indispensable. Several markerless tracking approaches exist which use either lines [HS90, CMPC06, WVS05] or point features [BPS05, MDR04] or a mixture of both [VLF04, RD05] to determine the camera pose. Tracking point features like the well-known KLT-Tracker [ST94] can be very promising, if the scene consists of many well-textured planar regions. In industrial scenarios objects which shall be tracked are unfortunately often poorly textured and consist of reflecting materials, whereas methods based on point features often produce insufficient results. Line features are more robust against illumination changes and reflections and are therefore very essential for the camera pose estimation. As 3D line models can be extracted out of construction data (i.e. VRML-models) lines can be a good connection between the geometric data of an object and its visual appearance in an image.

In this paper we present a model based approach which generates a 3D line model out of a surface model of an object and uses this line model to track the object in the image. For the line model generation a VRML-model of the object is rendered with a predicted camera pose and contours are extracted out of the z-buffer. The generated 3D line model is projected into the current image and the extrinsic camera parameters are estimated by minimizing the error between the projected lines and strong maxima of the image gradient. An application for this tracking approach is an augmented reality system for the maintenance of industrial facilities, which are poorly textured.

urn:nbn:de:0009-6-11595, ISSN 1860-2037

2 Tracking of CAD models

Whereas in [WVS05] the VRML model to be tracked must be made up of geometric 3D lines, in this method the 3D line model is generated out of a surface model stored in a VRML file. For the tracking it is important, that the line model is composed of lines, which are visible as lines in the image as well. In the ideal case only lines which are visible from a certain viewing direction shall be taken into account for tracking. One possibility is to generate a line model in object space by examining the angle of neighbouring triangles. To get only a set of visible lines for a certain viewing direction, a visibility test is necessary for a successful line model registration. We present a method which generates a line model for a certain view direction in image space by rendering the object and analysing the discontinuities in the z-buffer. The whole tracking algorithm consists of two stages. First a prediction of the camera pose is made to guarantee that the projected lines of the generated model are close enough to the image edges. Therefore the line features from the previous frame are used to make a prediction of the camera pose in the current image. This problem is addressed in section 2.3. The second stage consists of the line model generation, which is detailed in section 2.1, and another registration step where the generated line model is aligned onto the image. The registration step is described in section 2.2.

2.1 Line Model Generation

The image of an object can consist of lines which correspond to object silhouettes, edges of the object, or changes of the material of the object surfaces. As material properties of industrial objects as textures or colours are rarely modelled correctly or exist at all, they do not contain valuable information which can be used for the line model generation. In our approach we therefore only focus on geometric properties of a 3D model. The real-time silhouette generation of polygonal models has been an issue in non-photorealistic rendering. Isenberg et. al. [IFH+03] describe methods which create silhouettes of a model in both object space and image space. Object space methods analyse the normal vectors of triangles in relation to the camera viewing direction. Image space methods render the scene and analyse the output of the rendered images. Whereas object space methods can produce curves with much higher precision, they come along with higher computational costs. Image space methods are computationally less expensive and can be implemented using pixel shading hardware as presented in [MBC02]. Hybrid algorithms exist [NM00] which combine the advantages of both image space and object space methods. As real time performance is an important criterion in our application we use only an image space method as detailed in [ND03] to generate a 3D line model.

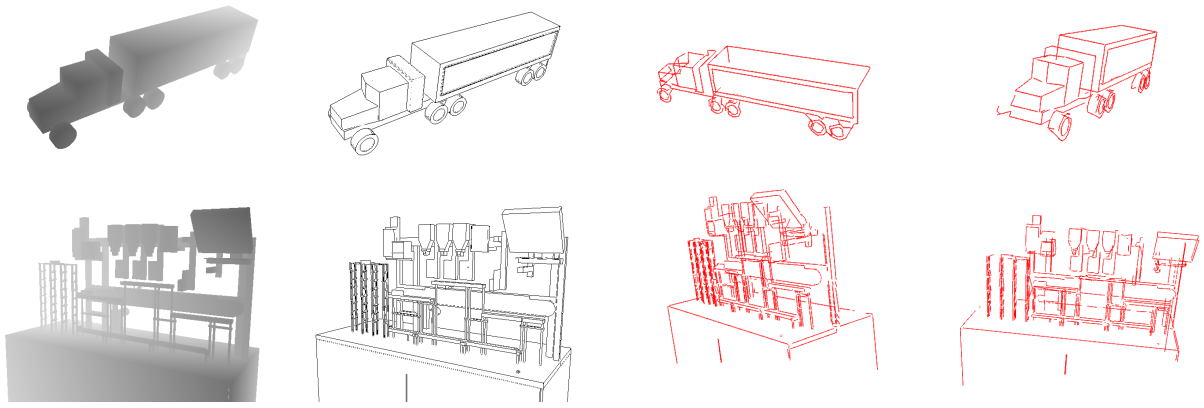
To create an edge map the scene is rendered with the predicted camera and the same resolution as the image, which is used as input for the tracking. To keep sampling artefacts as low as possible, the near and far clipping planes are adjusted to exactly match the bounding volume of the object. Discontinuities in the resulting z-buffer image are changes in depth and can be regarded as a point on an edge of an object according to the given camera viewing direction. A second order differential operator is applied on the z-buffer image to generate an edge image. In our implementation we use the following simple 2D Laplacian filter mask:

0	1	0
1	-4	1
0	1	0

As the values in the z-buffer are not linear in depth, a threshold is not applied directly to the edge map, but to the amount of difference in camera z-coordinates. This is necessary since an edge further in depth would have a smaller difference of z-buffer values than the same edge, if it would be closer to the camera. Furthermore the values of the z-buffer depend on the distance of the near and the far plane, which is not constant. Therefore taking directly the differences of z-buffer values would produce unsteady results.

The camera z-value can be computed by unprojecting a pixel in the edge image with the aid of the depth value in the z-buffer. To ensure that an edge consists of a only one pixel thick contour, a non-maxima suppression is applied on the edge map. For silhouette edges the Laplace operator returns a high absolute value both on the object border and on the neighbouring pixel in the background. Therefore at strong discontinuities we only regard positive values returned by the Laplace operator as edge pixels, in order to guarantee that no edges are generated on a background surface or on the far plane.

A Canny like edge extraction algorithm walks along the 2D edges of the edge map, but creates 3D contours. A recursive subdividing algorithm splits the 3D contours into straight line segments. However, the lines



(a) (b) (c) (d)

Figure 1: 3D Line extraction using discrepancies in the z-buffer. (a) shows a z-buffer image. (b) illustrates an edge image obtained by a Laplacian Filter. (c) and (d) show the generated 3D line model of different views.

are here only used for visualisation. As the tracking approach described in section 2.2 needs only control points and the direction of the contour on the control point, these control points can be generated directly by walking along the contour on the edge map. A 3D point in the world coordinate system can be computed by un-projecting a 2D image point and applying an inverse transformation of the current camera rotation and translation parameters. Figure 1 illustrates z-buffer images of some objects, the resulting edge maps and the generated line model.

2.2 Line model registration

The results of the line model generation described in the previous section can now be used as input for the tracking step. Our implementation is based on the approach of [CMPC06] and [VLF04]. The camera pose computation is based on the minimization of the distance of a projected 3D line and a 2D line in the image. A very efficient technique to create correspondences between 2D image points and 3D model points is to create control points on the projected line and to perform a one-dimensional search for gradient maxima along the normal of an edge. Figure 2 illustrates this process.

As it is difficult to decide which of the gradient maxima really corresponds to the control point on the projected line, more than one point is considered as a possible candidate. In [VLF04] it was shown that using multiple hypotheses prevents wrong gradient maxima from being assigned to control points on a projected line. When more than one gradient maximum exists for a control point, then during the minimisation always that hypothesis is used which has the closest distance to the projected control point. If p_i is the i^{th} control point and $q_{i,j}$ is the j^{th} hypothesis of the i^{th} control point, then the error to be minimised can be expressed as

$$err = \sum_i \min_j \Delta(p_i, q_{i,j}) \tag{1}$$

where Δ is the distance function between the projected sample point p_i and the line through the edge feature point q_i in the image. The line through q_i is parallel to the projected line of the line model. The distance function can be written as:

$$\Delta(p_i, q_i) = |(q_i - p_i) \cdot (n_i)| \tag{2}$$

where n_i indicates the normal of the projected line.

To make the pose estimation robust against outliers, an estimator function can be applied to the projection error. In out implementation we use the Tukey estimator, which is defined as follows:

$$\rho_{Tuk}(x) = \begin{cases} \frac{c^2}{6}[1 - (1 - (\frac{x}{c})^2)^3] & \text{if } |x| \leq c \\ \frac{c^2}{6} & \text{if } |x| > c \end{cases} \tag{3}$$

where c is a threshold depending on the standard deviation σ of the estimation error. Together with the estimator function we can write the error as follows:

$$err = \sum_i \rho_{Tuk}(\min_j \Delta(p_i, q_{i,j})) \tag{4}$$

In contrast to previous work [WVS05] the control points are not created on the projected line of the 3d

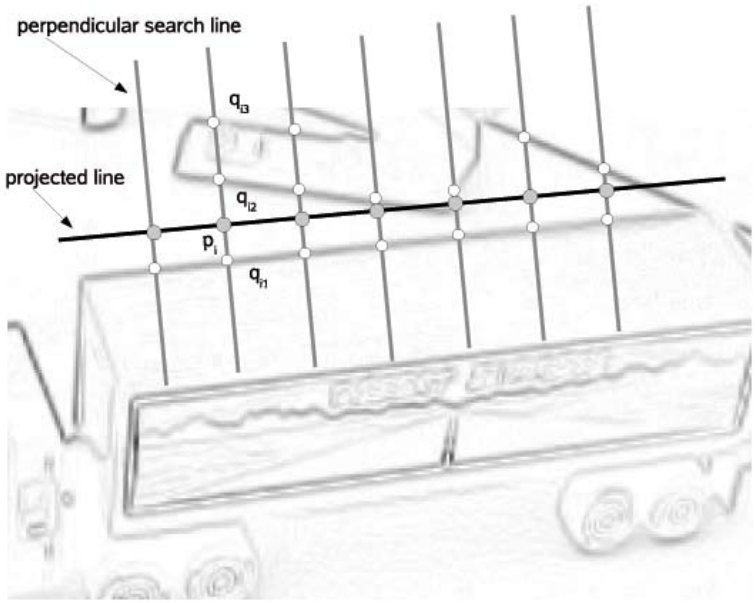


Figure 2: Perpendicular search for gradient maxima. p_i is a point on the projected 3d line. q_{ij} are hypotheses of the corresponding points in the image.

line model, but directly during the tracing of the contours in the edge map. Since only control points and their normal directions to the contour can be taken as input for the line tracking step, there is no need to generate straight 3D line segments.

To decide if a tracking step was successful the line model is projected into the image again with the computed camera pose. If the average aberration of the direction of the projected lines and the direction of the image gradient is smaller than a given threshold, the registration of the line model is regarded as successful.

2.3 Prediction of the camera pose

The line model generation described in the previous section is a local search method. The convergence towards local minima can be avoided if the initial camera pose used for the minimisation is a very close approximation to the real camera pose. Therefore a prediction of the camera pose from frame to frame is very beneficial. A learning-based approach as described in [WVS05] is not possible, since a new line model and new control points are generated in every frame. As correspondences between 3D contour points and 2D image points exist, if the previous frame was tracked successfully, these correspondences can be used again to make a prediction of the current frame. Therefore the control points of the 3D contour generated in the previous frame are projected into the current image with the camera pose of the previous frame. Again a one-dimensional perpendicular search is performed, but instead of looking for gradient maxima the point which is most similar to the 2D point in the last frame is regarded as the wanted 2D point. Similarity is measured by the cross correlation of a pixels row. So if the tracking in the previous frame was successful, a one-dimensional window on every control point perpendicular to the direction of the edge is extracted out of the previous image and correlated along the one-dimensional search line in the current image. For every 3D control point the point with the highest correlation result is regarded as the corresponding 2D image point. The calculation of the camera pose is done exactly as in section 2.2, except that only one 2D point for every 3D point exists.

3 Algorithm outline

The processing of one frame of the tracking method can be described as follows:

1. If the previous frame was tracked successfully,
   (a) create correspondences between the 3D control points of the last generated model, and the 2D points obtained by the correlation along the line perpendicular to the regarded edge,

(b) predict camera pose by minimising the distance of every projected 3D control point to the line through the corresponding 2D point parallel to the projected line.

2. Generate a new line model to the predicted camera pose.
3. Apply the line model registration.
4. If the registration was successful, extract a one-dimensional correlation window at every control point.

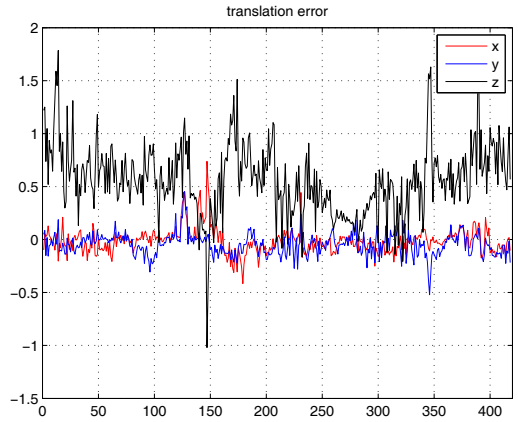
The first camera pose is defined by the user. It must be close to the real camera pose, so that the registration of the generated line model can be successful. The generation step produces a line model which is only used for visualisation and a set of 3D control points, which are used for tracking. If the registration step fails, no camera pose prediction is performed in the proximate frame. All images are undistorted with given radial distortion parameters, before they are processed by the algorithm.

4 Experimental results

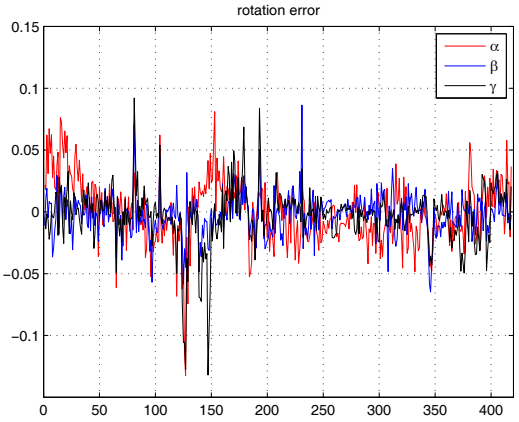
All the tests are done on a Pentium 4 with 2.8GHz and a ATI Radeon 9700Pro graphic card. To evaluate the robustness and the accuracy, the algorithm is tested on a synthetic image sequence first. A virtual model of a toy car is rendered with a predefined camera path, where the camera moves half around the model and back again. The images, which are rendered with a resolution of 512×512 pixels, and the very first camera pose are used as input for the tracking method. After every processed frame the 6 degrees of freedom of the estimated camera pose are stored and compared with the ground truth data. In figure 3 these values are plotted separately for every parameter. It can be observed that there is always a small error, but the method is capable of tracking the camera path throughout the whole synthetically generated sequence correctly.

parameter	mean error	standard deviation
x	-0.0148	0.1182
y	-0.0462	0.1068
z	0.5608	0.3335
α	0.0003	0.0259
β	-0.0022	0.0163
γ	-0.0031	0.0229

Table 1: Average error and standard deviation of the extrinsic camera parameters.



(a)



(b)

Figure 4: Error between the estimated and the real camera pose of a synthetic image sequence. In (a) the components of the camera translation error are plotted, (b) shows the rotation error as the difference of Euler angles.

The difference of the values between the real and the estimated camera pose are shown in figure 4. Euler angles in radians are used to represent the three parameters of the camera rotation.

In table 1 the mean error and the standard deviation of every component of the extrinsic camera parameters can be seen. As for the most parameters the error is alternating around 0, the mean of the z-component of the translation error is clearly above 0. This means that the estimated camera pose is always further away or that the tracked object in the image seems smaller than it really is. The reason of this fact is, that the extracted

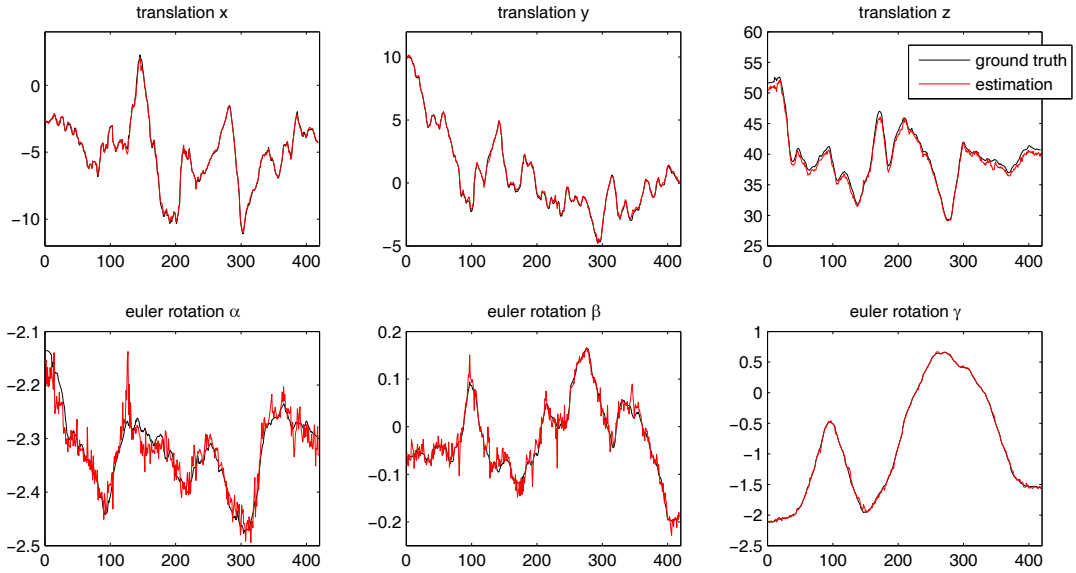


Figure 3: Comparison of the estimated pose and the ground truth of the camera path.

silhouette edges are always on the object and the gradient edges in the image have their peak between the object border and the background pixel. Therefore the extracted silhouette edges have an error of half a pixel which mostly affects the z-component of the camera translation. By analyzing the standard deviations it can be seen, that the uncertainty of the camera z-direction is significant larger than the other dimensions. The mean rotation error in radians is 0.0217, which is an average rotation error of 1.242 degree.

The computational costs of the individual steps is shown in table 2. Retrieving the depth buffer from the GL system requires a major part of processing time. Better performance might be able with newer hardware like PCI Express boards. The creation of correspondences in the prediction step is also very time consuming, which mostly can be attributed to the normalized cross correlation with sub-pixel accuracy. Together with the image acquisition and the visualization the system runs with a frame-rate at 20Hz.

prediction step	**time in ms**
create correspondences	10.46
predict pose	2.12
tracking step	
render model / read depth buffer	12.94
extract edges	6.84
create correspondences	8.10
estimate pose	2.42
total time	42.88

Table 2: Average processing time of the individual steps of the tracking approach.

resolution	std. dev. (trans/rot)	runtime in ms
384×384	0.2499 / 0.0267	31.74
512×512	0.1862 / 0.0217	42.88
768×768	0.1420 / 0.0241	81.28
1024×1024	0.0981 / 0.0116	120.41

Table 3: Comparison between image resolution, the average standard deviation of the error and the runtime.

Both the accuracy and the runtime of the tracking highly depends on the resolution of the rendered image, which is used to generate the 3D line model. A comparison of the image resolution and the runtime is shown in table 3. With an increasing image resolution a more detailed 3D line model can be extracted and therefore the result of the pose estimation gets more precise. As expected the runtime increases, since not only a bigger image has to be analyzed in the line model generation step, but also more control points on the contours are extracted and used in the tracking step. To reduce the processing time in the tracking step, the minimum distance between extracted control points can be increased, which would lead to a smaller number of correspondences between control points on

the extracted 3D contours and maxima of the image gradient. The length of the edge search and the termination criterion of the minimization also have an influence on the robustness and the runtime. Altogether the proper choice of the thresholds is a tradeoff between the performance and the accuracy and robustness.

The algorithm is tested on several real image sequences with different objects. The first sequence shows an industrial production line. The initial camera pose is set close to the real camera pose of the first frame. Throughout the whole sequence the correct camera pose is estimated. Occluded edges do not appear in the line model, since a new line model is generated in every frame. In figure 5 some frames of this sequence are shown. The same sequence is tested without the prediction step as well. When large movements, especially fast rotations of the camera occur, the registration step does not produce correct results. The parameters of the camera pose get stuck in local minima and the overall tracking fails. Therefore a rough estimation of the camera pose is really necessary to handle fast camera movements.

In another sequence a toy truck was used as an object to be tracked. Only a very coarse polygonal model without any details is used as an input for the algorithm. To demonstrate that it is possible to track an object not only from one point of view, the camera trail starts at one side of the object and moves around until the object is visible from the other side. The toy truck is tracked throughout the whole sequence, although the polygonal model is not very accurate. Some jitter can be recognized in some frames. Figure 6 illustrates same frames of this sequence.

5 Conclusion

We have presented a tracking method for augmented reality applications, which is able to track an object with a given polygonal model. The advantage of this model-based approach is that no drift can be accumulated during the tracking, since through the model a very significant connection between the virtual and the real world exists. No preprocessing step like the generation of a line model or the calibration of the scene is necessary. The virtual information can be created in the same coordinate system as the polygonal model, which is used for tracking. Especially for poorly textured scenes and objects with sharp edges or high-contrast silhouettes the method produces reasonably good results.

If not enough significant edges of the generated line model appear in the image, the tracking gets very unstable. Problems occur as well in highly detailed scenes, where the generated line model consists of so many edges that no unique registration is possible.

Future work will concentrate on a more precise and less time-consuming generation of line models. A promising approach would be to shift the edge map generation on the GPU by using pixel and vertex shaders.

Acknowledgements

This work was supported by a grant of the German Ministry of Education and Research (BMBF) entitled "ARTESAS: Advanced Augmented Reality Technologies for Industrial Service Applications" (www.artesas.de). The authors would like to thank the project partners within the consortium and the BMBF for their financial support.

References

[BPS05] Gabriele Bleser, Yulian Pastarmov, and Didier Stricker, *Real-time 3D Camera Tracking for Industrial Augmented Reality Applications*, WSCG, 2005, ISBN 8-0903-1007-9, pp. 47–54.

[CMPC06] Andrew I. Comport, Eric Marchand, Muriel Pressigout, and Francois Chaumette, *Real-time markerless tracking for augmented reality: the virtual visual servoing framework*, IEEE Trans. on Visualization and Computer Graphics **12** (2006), no. 4, 615–628, ISSN 1077-2626.

[HS90] Chris Harris and Carl Stennett, *RAPID - a video rate object tracker*, Proceedings of British Machine Vision Conference (BMVC), 1990, pp. 73–77.

Citation

Harald Wuest and Didier Stricker, *Tracking of industrial objects by using CAD models*, Journal of Virtual Reality and Broadcasting, 4(2007), no. 1, April 2007, urn:nbn:de:0009-6-11595, ISSN 1860-2037.

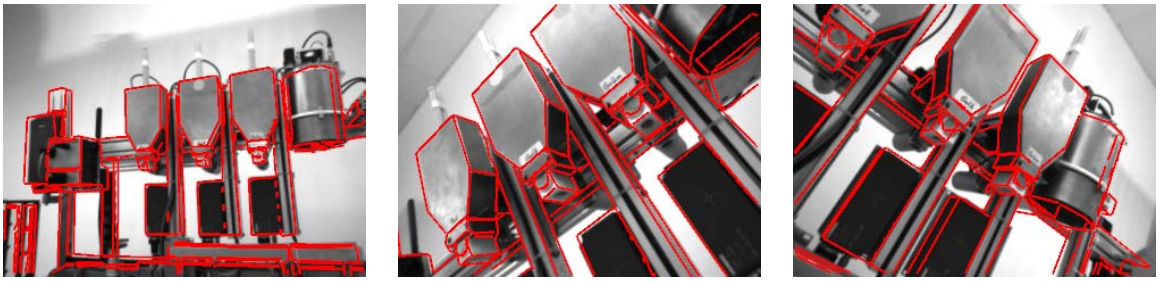


Figure 5: Tracking of an industrial production line.

Figure 6: Results of the tracking algorithm using a toy truck.

[IFH+03] Tobias Isenberg, Bert Freudenberg, Nick Halper, Stefan Schlechtweg, and Thomas Strothotte, *A Developer's Guide to Silhouette Algorithms for Polygonal Models*, IEEE Comput. Graph. Appl. **23** (2003), no. 4, 28–37, ISSN 0272-1716.

[MBC02] Jason L. Mitchell, Chris Brennan, and Drew Card, *Real-Time Image-Space Outlining for Non-Photorealistic Rendering*, ACM SIGGRAPH Conference Abstracts and Applications, ACM Press, 2002, ISBN 1-58113-525-4.

[MDR04] Nicholas D. Molton, Andrew J. Davison, and Ian D. Reid, *Locally Planar Patch Features for Real-Time Structure from Motion*, Proc. British Machine Vision Conference, BMVC, Sep 2004, ISBN 1-9017-2525-1.

[ND03] Marc Nienhaus and Jürgen Döllner, *Edge-Enhancement - An Algorithm for Real-Time Non-Photorealistic Rendering*, WSCG, 2003, ISSN 1213-6972.

[NM00] J. D. Northrup and Lee Markosian, *Artistic Silhouettes: A Hybrid Approach*, Proceedings of the First International Symposium on Non-Photorealistic Animation and Rendering (NPAR) for Art and Entertainment, June 2000, ISBN 1-5811-3277-8, pp. 31–37.

[RD05] Edward Rosten and Tom Drummond, *Fusing points and lines for high performance tracking*, IEEE International Conference on Computer Vision, vol. 2, October 2005, ISSN 1550-5499, pp. 1508–1515.

[ST94] Jianbo Shi and Carlo Tomasi, *Good Features to Track*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94), 1994, ISBN 0-8186-5825-8, pp. 593 – 600.

[VLF04] Luca Vacchetti, Vincent Lepetit, and Pascal Fua, *Combining Edge and Texture Information for Real-Time Accurate 3D Camera Tracking*, Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR), November 2004, ISBN 0-7695-2191-6, pp. 48–57.

[WVS05] Harald Wuest, Florent Vial, and Didier Stricker, *Adaptive line tracking with multiple hypotheses for augmented reality*, Fourth IEEE and ACM international symposium on mixed and augmented real-

ity (ISMAR), 2005, ISBN 0-7695-2459-1,
pp. 62– 69.