Tracking Ground Moving Extended Objects using RGBD Data

Marcus Baum, Florian Faion, and Uwe D. Hanebeck

***Abstract*— This paper is about an experimental set-up for tracking a ground moving mobile object from a bird's eye view. In this experiment, an RGB and depth camera is used for detecting moving points. The detected points serve as input for a probabilistic extended object tracking algorithm that simultaneously estimates the kinematic parameters and the shape parameters of the object. By this means, it is easy to discriminate moving objects from the background and the probabilistic tracking algorithm ensures a robust and smooth shape estimate. We provide an experimental evaluation of a recent Bayesian extended object tracking algorithm based on a so-called *Random Hypersurface Model* and give a comparison with active contour models.**

I. Introduction

Object tracking [1] treats the successive localization of a moving object over time. Object tracking is a fundamental task in many technical areas such as robotics or surveillance.

This work considers the marker-less tracking of a moving object on the ground with an RGBD camera observing the scene from a bird's eye view. We set up a specific miniature scenario in which a toy train moves on a table as shown in Fig. 1. On top of the table, a Microsoft® Kinect™ sensor is mounted that provides RGB and depth images from the table. In order to track the train, we detect moving points in the RGB and depth images (see Fig. 2). As a result, a set of noisy measurements from the train is obtained for each frame. These measurements typically do not cover the entire surface of the object and also false measurements not stemming from the train may be obtained. Hence, we have to deal with a so-called *extended object tracking problem*, i.e., the successive estimation of both the shape and kinematic parameters of an object based on noisy measurements from the surface. In particular, we will use a recently developed approach called *Random Hypersurface Model* (*RHM*) [2] that estimates a star-convex shape approximation of the object.

We believe that the presented experimental setting can serve as a benchmark and evaluation scenario for extended object tracking methods as its behavior is similar to larger sensors such as *Ground Moving Target Indicator (GMTI)* sensors [3]. But, in contrast to a radar device, the presented scenario is easy to set up and affordable. Nevertheless, there are many further applications of the suggested tracking system, e.g., people tracking or traffic surveillance.

An obvious advantage of the proposed tracking system is that non-moving objects are invisible in the sense that they do not cause measurements. The combination of RGB and depth data is in particular suitable for detecting moving points as moving objects always differ from the background in depth and usually also in color. Hence, typical problems of vision-only algorithms are avoided, i.e., the proposed method is more robust against light changes.

Marcus Baum, Florian Faion, and Uwe D. Hanebeck are with the Intelligent Sensor-Actuator-Systems Laboratory (ISAS), Institute for Anthropomatics, Karlsruhe Institute of Technology (KIT), Germany. marcus.baum@kit.edu, florian.faion@kit.edu,uwe.hanebeck@ieee.org

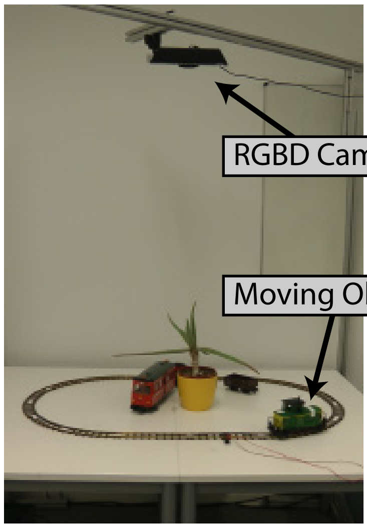


Fig. 1: Setup: Moving object, i.e., a toy train, observed with an RGBD camera from a bird's eye view.

The method for extracting the moving points employs rather basic image processing techniques as we follow the philosophy of performing as less as possible preprocessing as possible. The probabilistic tracking algorithm we suggest is capable of dealing with the extracted data in a systematic and sound manner based on a probabilistic model for the generation of a *single* point measurement. No prior information about the object is required as its shape is estimated from scratch, and shape changes are followed based on a probabilistic model for the temporal evaluation of the shape. The presented approach is real-time capable as the detection of moving points can be efficiently performed, and the tracking algorithm is based on a recursive closed-form measurement update.

A. *Related Work*

For a detailed overview of visual object tracking, we refer to [4], [5]. For example, popular methods are active contour models that determine an enclosing contour of the object by minimizing an energy function [6]. Furthermore, kernel methods such as [7] employ mean-shift algorithm

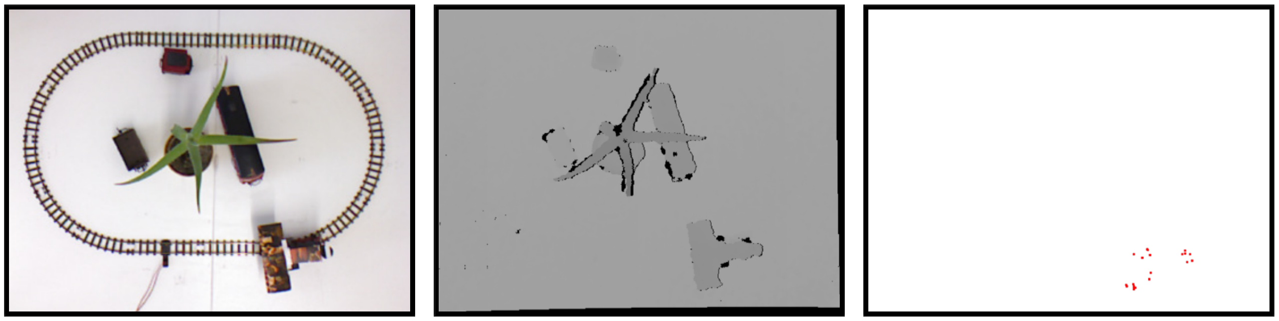

(a) RGB image. (b) Depth image. (c) Extracted measurements.

Fig. 2: RGB and depth images supplied by the Kinect Sensor and the extracted measurements. Only measurements from the train are received as it is the only moving object in the scene.

to the spatial and color space. There exist several visual tracking methods based on moving object detection in RGB images (see [5], [8]). The easiest way to detect moving points is to consider the difference between two consecutive frames. More advanced techniques such as [9] learn the color histogram of the background.

Apart from computer vision, moving object detection is also known in surveillance. For example, *Moving Target Indication (MTI)* [3] is a special mode of a radar that discriminates an object from the background by exploiting the Doppler effect. Special MTI sensors can be distinguished by the environment in which they operate: For example, a *Ground MTI (GMTI)* [3] aims at moving objects on the ground, while an *Airborne MTI (AMTI)* targets flying objects. In this context, there is also a concept called *Visual MTI (VMTI)* [10], which is used for moving objects based on visual data.

Some recent work on detection and tracking using combined RGB and depth cameras can be found for example in [11], [12], [13]. To the best of our knowledge, combined RGB and depth data has not yet been used for tracking ground moving objects based on moving object detection.

B. Overview

The remainder is structured as follows. In the next section, we explain the signal processing algorithms used for moving point extraction. Section III shows how the extracted measurements are used for tracking the object's location and shape based on a probabilistic shape tracking algorithm. In Section V, we present a detailed evaluation of the proposed method including a comparison with active contour models.

II. Signal Processing

In this section, we present the signal processing algorithm for extracting moving points from the RGBD data stream produced by the sensor. More precisely, the sensor provides a series of RGB images $I_k(x,y) \in \mathbb{R}^3$ and depth images $D_k(x,y) \in \mathbb{R}$, where k is the time index and $[x,y]^T \in \{1,2,\dots,u\} \times \{1,2,\dots,v\}$ denotes the pixel coordinates in an image with resolution $u \times v$. The purpose of the signal processing algorithm is to detect moving points, i.e., a set of pixel coordinates

$$\mathcal{Y}_k = \{\underline{\hat{y}}_{k,1}, \dots, \underline{\hat{y}}_{k,n_k}\} \ ,$$

where $\underline{\hat{y}}_{k,i} \in \{1,2,\dots,u\} \times \{1,2,\dots,v\}$ for all $i \in \{1,\dots,n_k\}$. These points serve as measurements for the tracking algorithm (see next section).

The signal processing algorithm consists of three major parts, i.e., *remove ground*, *detect moving points*, and *dismiss clutter*, which are discussed in the following.

a) Remove Ground: As the height of the table and the sensor is known, it is obvious to exploit this knowledge to remove pixels, which belong to the ground. This is done by applying a (user-defined) threshold on the depth images, i.e., each pixel, which exceeds a specific distances from the sensor is marked as background. It is neglected in the further processing as the ground does not move. Hence, the RGB and depth images with subtracted background are

$$D_k^t(x,y) := \begin{cases} D_k(x,y) & \text{if } D_k(x,y) < t_{\text{depth}} \\ -1 & \text{otherwise} \end{cases} \ ,$$

and

$$I_k^t(x,y) := \begin{cases} I_k(x,y) & \text{if } D_k(x,y) < t_{\text{depth}} \\ -1 & \text{otherwise} \end{cases} \ ,$$

where -1 marks a pixel as "ground".

b) Detect Moving Points: This is a central processing step in which moving points are determined by applying an optical flow algorithm, i.e., the Horn-Schunck method [14], to the RGB images $I_k^t(x,y)$ and depth images $D_k^t(x,y)$ separately. As a result, a velocity vector is obtained for each pixel in the RGB and depth image. The velocity fields are denoted with $I_k^v(x,y) \in \mathbb{R}^2$ for the RGB images and $D_k^v(x,y) \in \mathbb{R}^2$ for the depth images. Based on these velocity vectors, moving pixels are determined by thresholding. A pixel is considered as moving if the magnitude of its velocity exceeds a use-specified threshold in its RGB *or* depth image, i.e.,

$$\mathcal{Y}_k^{temp} := \left\{ \left[x, y\right]^T \,\middle|\, ||I_k^v(x,y)|| > t_{\text{vel}} \right. \\ \left. \text{or } ||D_k^v(x,y)|| > t_{\text{vel}} \right\} \quad (1)$$

Note that it is also reasonable consider a point as moving when both RGB *and* depth image exceed a threshold. It is even possible to use only RGB or depth information for moving point detection.

c) Dismiss Clutter: For small objects as considered in our setting, the signal-to-noise ratio of the Kinect sensor is rather low, i.e., the boundary contours of objects in the depth image are very noisy. As a consequence, boundary points of non-moving object often get classified as moving. In order to solve this issue, an edge detection algorithm is used to dismiss all points from the edges. For this purpose, a moving point is dismissed if a specific number n_{edge} of neighbor pixel is also moving. A neighbor pixel is a pixel whose distance to the considered pixel is less than r_{edge}. As a result, only moving points from the inner part of the object are obtained, i.e.,

$$\mathcal{Y}_k := \left\{ \left[x, y\right]^T \,\middle|\, \left[x, y\right]^T \in \mathcal{Y}_k^{temp} \text{ and} \right. \\ \left. |\mathcal{Y}_k^{temp} \cap K\left(\left[x, y\right]^T, r_{\text{edge}}\right)| \leq n_{\text{edge}} \right\} \ , \quad (2)$$

where $K(\left[x, y\right]^T, r_{\text{edge}})$ denotes a circular disc with center $\left[x, y\right]^T$ and radius r_{edge}. The extracted set of pixels $\mathcal{Y}_k$ now serves as input for the tracking algorithm as described in the following section. An example frame with the extracted measurements $\mathcal{Y}_k$ is shown in Fig. 2c.

A. Adjusting the Parameters

The proposed signal processing algorithm involves some parameters to adjust. Obviously, the threshold for subtracting the table ground t_{depth} is very easy to find. The thresholds for removing the edges n_{edge} and r_{edge} can be adjusted by visual inspection. It has to be determined such that the edges of non-moving objects disappear. The threshold on the velocity t_{vel} has to be determined such that a non-moving object does not produce measurements, however, a moving object does.

Note that each radar devices involves a similar parameter, the so-called detection threshold, which determines if a received echo stems from a target. Finally, it is worth mentioning that automatic methods for determining the above parameters are currently under development.

III. Extended Object Tracking Algorithm

According to the previous section, a varying number of noisy measurements $\mathcal{Y}_k$ from the target surface is obtained for each frame (see Fig. 2c). Typically, only a small number of measurements is extracted, and the measurements do not cover the entire surface at object. This is a consequence of the sensor noise, which is rather large compared to the extent of the object, and the limited resolution capabilities.

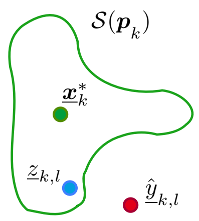


Fig. 3: Modeling the shape: The parameter vector for the shape is given by $\underline{p}_k$, and $\underline{x}_k^*$ subsumes parameters for the kinematic, i.e., position and velocity. The measurement model specifies the generation of a measurement source $\underline{z}_{k,l}$ on the shape. The measurement itself, i.e., the detected moving point, $\hat{\underline{y}}_{k,l}$, is a noisy observation of this measurement source. Note that the measurement may lie outside of the shape due to noise.

The goal is to estimate the shape and kinematic parameters of the object based on these measurements The problem is called *extended object tracking* and can be frequently found in surveillance. For example, when a radar device is close to a target, a varying number of reflections from different reflection centers may be obtained. In this context, a variety of *extended object tracking* algorithms were developed in the recent years (see for example [2]). These algorithms are capable of estimating the shape and kinematic parameters based on successively arriving point measurements.

In this work, we use an extended object tracking approach, which is able to estimate a star-convex shape approximation of the target as presented in [2] called *Random Hypersurface Model* (*RHM*).

In the following, we will outline the underlying probabilistic models for *RHMs* and briefly discuss the inference mechanism. We also present some simple extensions for track management on top of the algorithm proposed in [2].

A. Object Model

The object state is represented with a random vector $\underline{x}_k = \left[\underline{p}_k^T, (\underline{x}_k^*)^T\right]^T$, where $\underline{p}_k^T$ are parameters for the shape and $\underline{x}_k^*$ are the kinematic parameters (e.g., position, velocity).

As in [2], the shape is represented with a radius function $r_k(\phi)$, which gives the distance from the object center to a contour point depending on an angle ϕ. The shape parameters $\underline{p}_k$ consist of the first Fourier coefficients of the Fourier expansion of the radius function $r_k(\phi)$. We denote the shape (including its interior) as $\mathcal{S}(\underline{p}_k)$.

B. Dynamic Model

The dynamic model specifies the temporal evolution of the object, i.e., for a given state $\underline{x}_k$ it says how to the state $\underline{x}_{k+1}$ will be. In a Bayesian setting, the dynamic model is specified by a conditional probability density. Here in this work, we use a (nearly) constant velocity model for the object center [15]. The shape parameters are assumed to be constant over time, however, a noise term is added in order to capture shape changes, e.g., rotation of the object.

C. Measurement Model

A special property of extended object tracking methods is that the measurement model, models the generation of a *single* measurement. If several measurements are received for a frame, their generation is assumed to be independent. A major benefit of this model is that the measurements do not have to cover the entire surface at a particular frame.

The underlying measurement model relates the state vector $\underline{\boldsymbol{x}}_k$ to a single measurement $\hat{\underline{y}}_{k,l}$. It is composed of two parts called *target extent model* and the *sensor model*. The target extent model specifies a measurement source on the target surface, i.e., $\underline{z}_{k,l} \in \mathcal{S}(\underline{\boldsymbol{p}}_k)$, where $\underline{z}_{k,l}$ denotes the l-th measurements source in frame k. In this work, the target extent model is given by an *RHM*, which assumes that each measurement sources lies on a scaled version of the shape boundaries. Details can be found in [2].

Given a measurement source $\underline{z}_{k,l}$ obtained from the target extent model, the sensor model gives the measurement $\hat{\underline{y}}_{k,l}$. In this setting, it is suitable to treat the measurement as noisy observation of a measurement source that is corrupted with additive Gaussian noise, i.e.,

$$\hat{\underline{y}}_{k,l} = \underline{z}_{k,l} + \underline{\boldsymbol{v}}_{k,l} \ , \tag{3}$$

where $\underline{\boldsymbol{v}}_{k,l}$ denotes zero-mean white Gaussian noise.

D. Bayesian Inference

Based on the above model, a nonlinear Bayesian state estimator can be used for recursively updating a probability density the object state $\underline{\boldsymbol{x}}_k$ based on the measurements. For this purpose, we use the Unscented Kalman filter (UKF) [2], which represents the uncertainty of the state $\underline{\boldsymbol{x}}_k$ with a Gaussian distribution (see [2]).

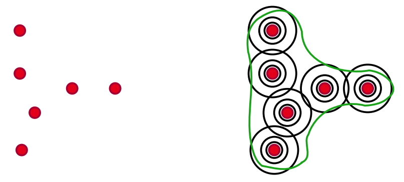
E. Track Management

On top of the shape tracking algorithm, we implemented basic track management algorithms as discussed in the following.

1) Gating: Even though the signal processing algorithm aims at detecting only points on the object, false measurements, which do not stem from the object, may arise due to noise. A simple gating procedure based on the current shape estimate is performed in order to dismiss these clutter measurements. For this purpose, each measurement that is an element of a scaled version of the star-convex contour is considered as stemming from the object, i.e., we blow up the shape for finding potential measurements.

2) Track Initialization: In this work, we only consider the tracking of a single object, hence, a new track is initialized if the number of measurements exceeds a specific user-defined threshold. The prior location of the object estimate is set to the mean of the measurements and the prior shape is given by a circle.

3) Track Termination: A track is terminated if the number of measurements in the validation gate falls below a user-defined threshold for a specific number of frames.



(a) Example measurements. (b) Active contour for smoothed measurements.

Fig. 4: In order to apply active contour models to point sets, we render a point set to a continuous intensity image by placing a Gaussian at each measurement point.

IV. Implementation Details

In this section, we briefly describe some implementation details. As already mentioned earlier, we use the Microsoft Kinect sensor. which provides RGB images with a resolution of 640x480 (8-bit color depth) and depth images with a resolution of 640x480 (11-bit depth). The frame rate of the Kinect sensor is 30 frames per second.

Both the signal processing and tracking algorithm were implemented on a basic desktop computer equipped with an Intel Core 2 Quad-Core Q660 processor with 2.4 GHz, 8192 MB DDR2 ram, and an AMD Radeon HD 5750 graphic card. The signal processing algorithm was implemented in OpenCV [16], OpenCL [17] (for the optical flow algorithm) and OpenGL [18]. The tracking algorithm was implemented in C++ using the Eigen library [19].

With the above system and implementation, we reached the achievable frame rate of 30 frames per second for tracking the object (in the considered scenarios, see next section). Of course, the frame rate depends on the number of Fourier coefficients, i.e., dimension of state, and number of measurements per time step. All told, the presented tracking algorithm is real-time capable even on a standard desktop computer. Note that there is a lot of room left for further run-time optimizations, e.g., a simultaneous update with several measurements could be performed.

V. Evaluation

In this section, we provide a detailed evaluation of the proposed tracking system. For this purpose, we compare the new method with active contour models [20], [21], which are widely spread object tracking methods in computer vision. Of course, in the considered scenario, the tracking algorithm does not know that a train is to be tracked. And, most important, it will not exploit any knowledge about the tracks or the adjusted velocity.

A. Random Hypersurface Model

Fig. 5 depicts the tracking result using the *RHM* as described in the previous section. The results show that the *RHM* gives precise and smooth shape estimates.

B. Active Contour Model on Point Measurements

In the following, we demonstrate the need for an extended object tracking method such as an *RHM* for tracking the object in case of a few measurements per frame. We compare the *RHM* (see Fig. 5) with an active contour model [21]. The active contour model employs the same point measurements $\mathcal{Y}_k$ as the *RHM*. Active contour models are not directly applicable when a set of point measurements is given as they are usually defined for intensity images. However, we can interpret the measurements $\mathcal{Y}_k$ as an intensity image by placing a Gaussian kernel at each measurement as illustrated in Fig. 4. The kernel width is tuned to get the best results. Note that this kernel width in fact depends on the (unknown) size of the object.

The shape estimates provided by the active contour model are depicted in Fig. 6. Active contour models will definitely work worse than an extended object tracking method in case of a few measurements per frame. For example, if only one measurement per frame, would be available, active contour models are bound to fail. This behavior can be seen when comparing both methods in Fig. 5 and Fig. 6. *RHMs* provide a smooth and precise shape estimates, while active contour models suffer from imprecise and too small shape estimates (due to the less measurements). Of course, if a large amount of measurements that covers the entire surface of the object would be available, active contour models would give much better results. Then there is actually no need for a methods like *RHMs*, although *RHMs* would also give precise results in this case.

C. Active Contours on RGB image

A reasonable method for tracking an object would be to use active contour models only for the RGB images. However, this method is bound to fail as the tracks prevent the active contour models to work properly due to the similar optical appearance. In general, active contour models will only work well if the object to be tracked differs in color from the background.

D. Active Contours on Depth image

An alternative is the application of an active contour model to the depth images. This approach works well in the considered scenario, if the object significantly differs from the background in height. A situation in which this scenario fails is depicted in Fig. 7, when the train passes a non-moving object that is located next to the tracks.

VI. Conclusion

In this work, we described an experimental setup for tracking ground moving objects using RGBD data. For this purpose, RGB and depth information has been used for detecting moving points. As a result, an extended object tracking problem is obtained, where the detected moving points serve as measurements from the object surface. In particular, we used a recently developed extended object tracking method based on *RHMs* in order to estimate a star-convex shape approximation of the object. The benefits of the tracking method have been elucidated with respect to active contour models. Prospective work will be focused on an automatic procedure for parameter adaption and an extension to a three-dimensional scenario is planned.

Acknowledgment

We would like to thank Kai Bouché for implementing the tracking system.

References

[1] S. Chen, "Kalman Filter for Robot Vision: A Survey," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4409 –4420, nov. 2012.

[2] M. Baum and U. D. Hanebeck, "Shape Tracking of Extended Objects and Group Targets with Star-Convex RHMs," in *Proceedings of the 14th International Conference on Information Fusion (Fusion 2011)*, Chicago, Illinois, USA, Jul. 2011.

[3] W. Koch, J. Koller, and M. Ulmke, "Ground Target Tracking and Road Map Extraction," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 61, no. 34, pp. 197 – 208, 2006.

[4] H. Yang, L. Shao, F. Zheng, L. Wang, and Z. Song, "Recent Advances and Trends in Visual Tracking: A Review," *Neurocomputing*, vol. 74, no. 18, pp. 3823–3831, November 2011.

[5] A. Yilmaz, O. Javed, and M. Shah, "Object Tracking: A Survey," *ACM Computing Surveys*, vol. 38, no. 4, Dec. 2006.

[6] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *International Journal of Computer Vision*, vol. 22, pp. 61–79, 1995.

[7] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 564–577, 2003.

[8] M. Yokoyama and T. Poggio, "A Contour-Based Moving Object Detection and Tracking," in *Proceedings of the 14th International Conference on Computer Communications and Networks*, ser. ICCCN '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 271–276.

[9] X. Gao, T. Boult, F. Coetzee, and V. Ramesh, "Error Analysis of Background Adaption," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2000, pp. 503–510.

[10] R. Jones, D. M. Booth, and N. J. Redding, "Video Moving Target Indication in the Analysts' Detection Support System," DSTO, Edinburgh, S. Aust., Tech. Rep. DSTO-RR-0306, 2006.

[11] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, june 2011, pp. 1297 –1304.

[12] C. Plagemann, V. Ganapathi, D. Koller, and S. Thrun, "Real-time identification and localization of body parts from depth images," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 3108 –3113.

[13] M. Luber, L. Spinello, and K. O. Arras, "People Tracking in RGB-D Data With On-line Boosted Target Models," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, USA, 2011.

[14] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 13, pp. 185 – 203, 1981.

[15] Y. Bar-Shalom, T. Kirubarajan, and X.-R. Li, *Estimation with Applications to Tracking and Navigation*. New York, NY, USA: John Wiley & Sons, Inc., 2002.

[16] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[17] Khronos OpenCL Working Group, *The OpenCL Specification, version 1.0.29*, 8 December 2008.

[18] Opengl, D. Shreiner, M. Woo, J. Neider, and T. Davis, *OpenGL(R) Programming Guide : The Official Guide to Learning OpenGL(R), Version 2 (5th Edition)*. Addison-Wesley Professional, August 2005.

[19] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.

[20] A. Blake and M. Isard, *Active Contours*, 1st ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1998.

[21] M. Jacob, T. Blu, and M. Unser, "Efficient energies and algorithms for parametric snakes," *IEEE Transactions on Image Processing*, pp. 1231–1244, 2004.

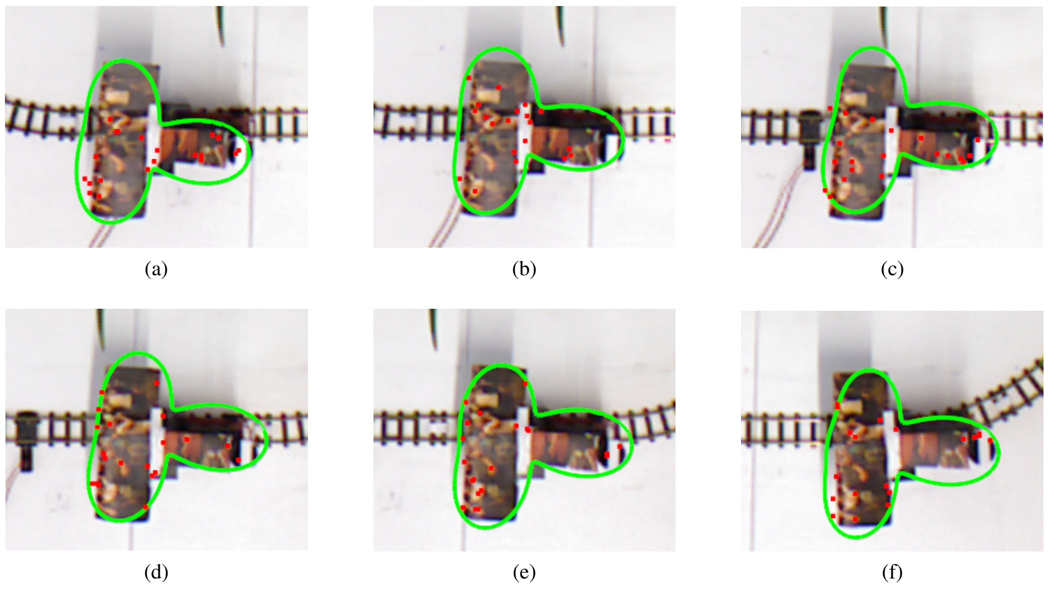


Fig. 5: Tracking results using a *Random Hypersurface Model*.

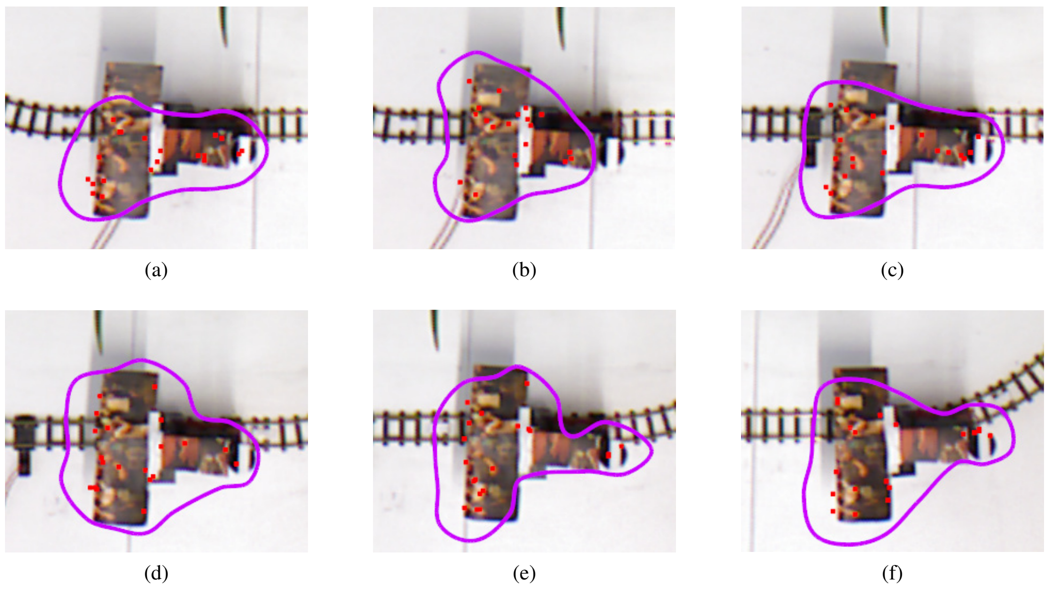


Fig. 6: Tracking results using an active contour model.

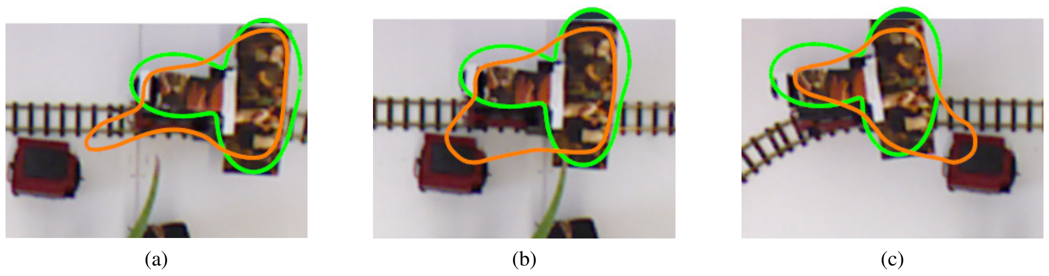


Fig. 7: Comparison: *RHM* applied to point measurements vs. active contour model applied to depth image.