Robust Pedestrian Detection and Tracking From A Moving Vehicle

Nguyen Xuan Tuong[a], Thomas Müller[b] and Alois Knoll[b]

[a]Department of Computer Engineering, Nanyang Technological University, Singapore
[b]Robotics and Embedded Systems, Technische Universität München, Germany

ABSTRACT

In this paper, we address the problem of multi-person detection, tracking and distance estimation in a complex scenario using multi-cameras. Specifically, we are interested in a vision system for supporting the driver in avoiding any unwanted collision with the pedestrian.

We propose an approach using *Histograms of Oriented Gradients* (HOG) to detect pedestrians on static images and a particle filter as a robust tracking technique to follow targets from frame to frame. Because the depth map requires expensive computation, we extract depth information of targets using *Direct Linear Transformation* (DLT) to reconstruct 3D-coordinates of correspondent points found by running *Speeded Up Robust Features* (SURF) on two input images. Using the particle filter the proposed tracker can efficiently handle target occlusions in a simple background environment. However, to achieve reliable performance in complex scenarios with frequent target occlusions and complex cluttered background, results from the detection module are integrated to create feedback and recover the tracker from tracking failures due to the complexity of the environment and target appearance model variability.

The proposed approach is evaluated on different data sets both in a simple background scenario and a cluttered background environment. The result shows that, by integrating detector and tracker, a reliable and stable performance is possible even if occlusion occurs frequently in highly complex environment. A vision-based collision avoidance system for an intelligent car, as a result, can be achieved.

Keywords: Histograms of Oriented Gradients, Particle Filter, Direct Linear Transformation, Speeded Up Robust Features

1. INTRODUCTION

In the past few years, there has been an increasing number of accidents on the street. Pedestrians are one of the most vulnerable traffic participants. Although a passive pedestrian safety system is able to reduce injury level upon impact, a more intelligent car needs a safety feature that is able to detect and track the pedestrian to avoid collision ahead of time. While a lot of research is made to both tracking and detection and a remarkable success has been achieved in the case of static camera and simple background environment, it is still highly challenging to deal with the problem in complex, cluttered background environments and with ego motion taken into account.

For the safety feature of a smart car, it is important to extract the depth information from targets to the car to start the brake system immediately if necessary. For this reason, detection and tracking systems that use single camera might not be able to obtain robust depth estimation. Therefore a multi-camera approach is proposed and a stereo vision algorithm is used to extract depth information. Although depth information can be achieved with great accuracy by constructing a disparity map[1],[2] this approach requires a lot of computational power and therefore might not be a good choice in a real-time systems. To cope with this issue, in this paper, we propose a simple way to extract the depth estimation based on a 3D point triangulation technique that is able to extract depth information with a

Send correspondence to N.X. Tuong (nguy0066@e.ntu.edu.sg) or T. Müller (muelleth@cs.tum.edu)

reasonable accuracy for a braking system. The triangulation technique computes correspondent points found by the SURF[3] algorithm that is able to operate in real-time with GPU support.[4]

Although our main interest is to estimate the distance between a pedestrian and the car for the braking system, it can only be achieved if we can detect the location of a pedestrian from cluttered background[5].[6] Because of this, pedestrian detection is the first step we need to consider in building a safety system. After that, we need a robust tracking algorithm that is able to follow a target from frame to frame because even the best detector is not able to detect the same target with one hundred percent accuracy. Furthermore, a detector is generally more expensive to execute and it might ignore some important targets compared to others. For all of those reasons, a safety system for a smart car in generally must deal with three important requirements:

1. The system must be able to detect interested targets within a certain safe threshold distance.
2. The system must be able to track the detected targets with robust and stable performance.
3. The system must be able to estimate the distance to interesting targets.

While the target detection process using a good detector such as *Histograms of Oriented Gradients* (HOG)[7] seems to be robust enough to detect pedestrian within a near distance, the tracking problem is largely unsolved. A

visual tracking system in general can be classified into one of two main categories: *Target Representation Localization* tracking or *Filtering and Data Association* tracking. The target representation and localization trackers such as contour-based trackers can be very useful for rigid object tracking. However in the highly nonlinear noise system of a moving camera and big target appearance model variability, we decided to use a filtering and data association tracker such as particle filter. This type of tracker can deal with complex objects along with multi-object interactions.

According to our observation, in general, almost any tracker will fail or perform badly when a big changes in the object appearance model happens. This is because the tracker uses initial data from the detection step to compare it with the observed data from the predicted hypothesis to track the target. Therefore, whenever the target appearance model changes, the observed data varies much with the initial data held within the tracker at the beginning. As a result, the tracker is likely to fail. In order to cope with this, we propose a solution that uses a detector to update the target appearance model whenever the likelihood between the observed data and initial data exceeds a certain threshold. It will keep the tracker up to date and perform more stable in highly cluttered environment.

The remainder of the paper is structured as follows. After discussing related work in the next section, we will describe how to detect a pedestrian on a single image and map it to another image to extract depth information. Then we propose a particle filter solution for the tracking problem. Finally, experimental results and conclusion are shown.

2. RELATED WORK

Object detection. There certainly is a lot of research done in the field of object detection to make it applicable for a variety of practical real-world applications. However, only few of them can cope with the challenging task of detecting pedestrians in complex and highly cluttered environments. After the success of using *Haar*-like features to build a cascade of boosted classifiers to detect pedestrian on static images,[8] a great success was made when in 2005, Navneet Dalal introduces *Histograms of Oriented Gradients* that were able to detect people with great accuracy. In this paper, we will use HOG as main method to detect the location of pedestrians on static images. Note that for the detector using HOGs, a real-time GPU implementation exists.[9] Therefore by using HOG detector, we aim to achieve real-time performance for detection,tracking and distance estimation.

Multi-target Tracking. There is a lot of research in the field of object tracking. For rigid object tracking, a remarkable success has been achieved by using some traditional tracking algorithms such as

contour-based tracking, blob tracking or a tracking technique that uses PnP[10] algorithm to estimate pose of object from frame to frame. However, there is only very limited success in the case of pedestrian tracking in highly cluttered environment and concerning moving cameras. To cope with this challenging requirement, a particle filter-based tracking system is proposed. It can handle multiple occlusions in simple background environments, however, for complex and cluttered environment with appearance model variability, we need to reuse the detector to update the tracking object model frequently. Our particle filter tracking is based on previous work[11] with OpenTL*. Here, we use a single modality likelihood $P(z|s)$ as an implicit measurement model.

3. PEDESTRIAN DETECTION AND DEPTH ESTIMATION

Because the depth information can only be estimated robustly with multi-view geometry, our depth estimation algorithm is tested on a data set †[12] taken from two calibrated cameras. Different from other algorithms that use disparity maps to extract depth information of targets, our algorithm aims to use a simple triangulation method such as the *Direct Liner Transformation*(DLT) algorithm to reconstruct 3D coordinates of some pairs of correspondent points. As a result, we propose an algorithm that is able to find correspondent points from two image patches with high accuracy. To achieve this, the usage of *Speeded Up Robust Feature* (SURF) is proposed. However, the SURF algorithm needs two image patches of the same target on both left and right images to find correspondent points. Unfortunately, running the HOG detector on a single static image can only find a target location on a single image. Therefore, we need an algorithm that is able to map each detected pedestrian location (represented as a rectangle) found by HOG on the left image to the other. An exhaustive search for template matching on 2D image is possible, but it is highly expensive. For this reason, we propose a template matching algorithm that is performed on the rectified image pair instead. It can perform very efficiently with only very little computational effort. Figure 1 below shows the target mapping algorithm between the two views of the stereo system.

3.1 Static map matrices for target mapping between two views

The data set used does have some calibration files. However it does not provide static map matrices necessary for rectified-image interpolation. Therefore, in our application, we build an algorithm constructs static map matrices for each camera from provided intrinsic and extrinsic camera parameters. The static map matrix then is used to map each pixel from the rectified image to the original. We compute the static map matrix using three following steps:

1. We compute two 3×3 row-aligned rectification rotation matrices R_l,R_r using provided camera intrinsic matrix K and rotation, translation matrices R, T. R_l,R_r can put two cameras into coplanar and row alignment.

2. We use the row-aligned rectification rotation matrices to compute corresponding pixel locations from the rectified image to the distorted image. Because each camera has distortion parameter, we need a third step to correct the distortion error.

3. We correct the corresponding pixel location from the second step using provided distortion coefficients. In our dataset, there are only two distortion parameters. These are the radial distortion coefficients k_1, k_2.

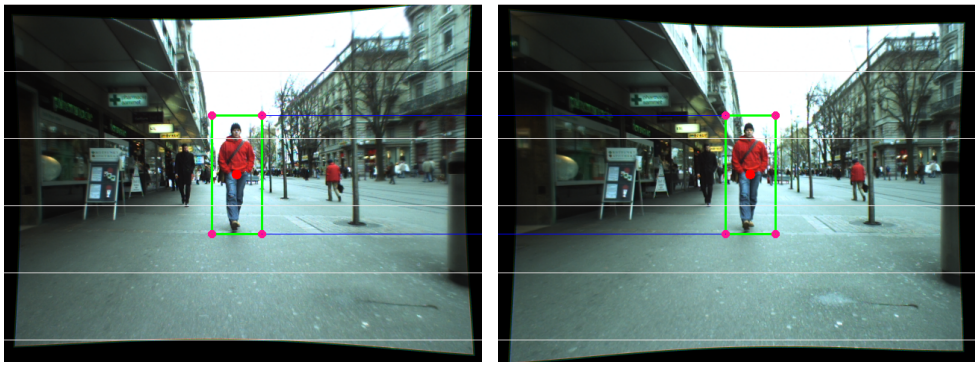
Afterwards, we construct a final static map matrix that can give us a very fast way to compute the corresponding pixel from rectified image to original image. Given this static map matrix, it is easy to construct a back-mapping matrix that, on other hands, is able to map a given pixel location from original image to rectified image. Then our target mapping algorithm can be performed very efficient as shown in Figure 1.

*`http://www.opentl.org`

†`http://www.vision.ee.ethz.ch/~aess/dataset/`

(a) A typical pedestrian rectangle location found by the HOG detector.

(b) Four pink corners on rectified image (d) are mapped to the navy-colored corners in (b) using the static mapping matrix of the right camera.



(c) Four navy-color corners from (a) are mapped to four pink-color corners of the same target on (c).

(d) A simple template matching algorithm to map the target location from (c) to (d).

Figure 1. The target mapping algorithm between two views.

3.2 Distance Estimation

To estimate the distance between the pedestrian and the car using point triangulation, we need to find at least one pair of correspondening points. As shown in figure 1, we choose the centres of two rectangle patches on the left and right rectified image (red color) and map them back to left and right original images to obtain a first pair of correspondent points (blue color). Because there is an error in our estimation, we need an algorithm to reduce the error so that the triangulation yields results with reasonable accuracy. To achieve this, we use the optimal triangulation method in[13] to minimize the geometric error subject to the epipolar constraint of this pair of corresponding points and then triangulate this pair using DLT to obtain a rough target depth estimation.

After that, we use SURF to find correspondent points from two mapped pedestrian rectangle locations on left and right images. The SURF algorithm returns many pairs of correspondent points as shown in the figure 2. However it also returns some false matches. As a result, we need an algorithm to filter the matching points so that only good matching points are kept for depth triangulation.

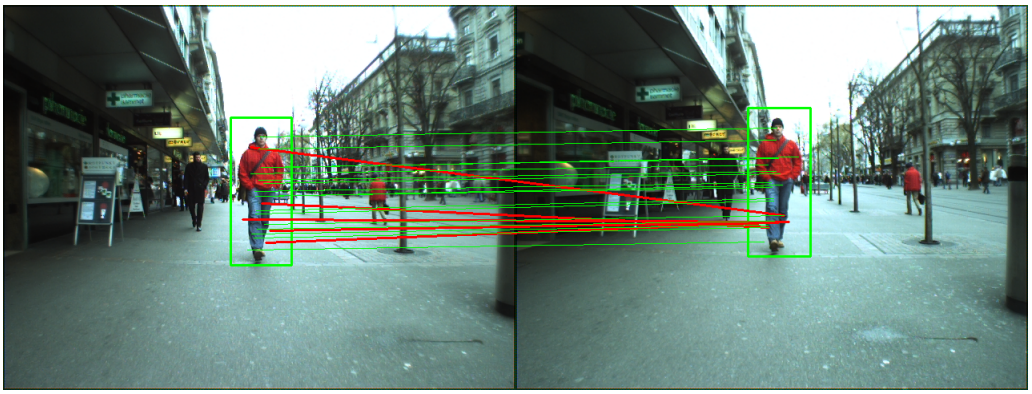


Figure 2. Pairs of correspondent points obtained with SURF. Red color is used to denote wrong matches or non-interesting correspondent points from background.

We then define good matching points for depth estimation based on two criteria:

1. Two matching points must follow epipolar geometry. We compute the fundamental matrix F using the provided intrinsic and extrinsic camera parameters. Then we use this matrix to eliminate wrong matching points.

2. We have the rough depth estimation from 3D triangulation of the first pair of two correspondent points as described in the previous section. All pairs of SURF matching points are triangulated to get a list of distances from each corresponding 3D point to the camera. The distances from the list are compared to the rough depth estimation to eliminate all points that belong to the background (see Figure 3). The purpose is to only keep good matching points that belong to the target of interest.

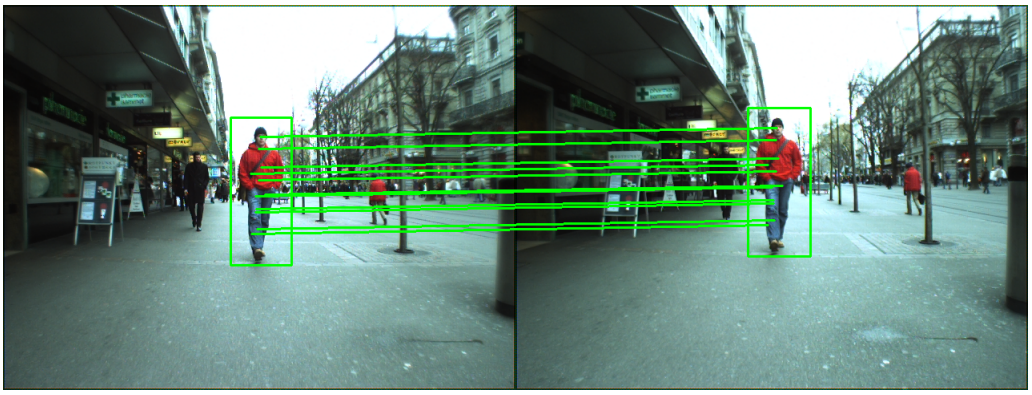


Figure 3. Good pairs of corresponding points after using the filter algorithm. Each pair is then corrected in order to closely follow epipolar geometry.

After having a good list of correspondent points, we correct them so that each pair of correspondent point follows exactly epipolar geometry using the same algorithm described in.[13] Afterwards, we use DLT to triangulate those points to get an average depth estimation for the target.

4. MULTI-TARGET TRACKING

4.1 Object Pose Representation

A target location can be represented by a rectangle in an image. Therefore we can represent the object pose by using a simple 2D pose model that consists of three degrees of freedom: translations t_x, t_y and scale parameter s. In our model, t_x, t_y are coordinates subject to the upper-left corner of rectangle in the image coordinate system and the scale parameter s can be used to compute sizes of the target locations under a given pose. The transformation matrix that is used to project targets under the given pose to the image can then be represented as:

$$T = \begin{pmatrix} s & 0 & 0 & t_x \\ 0 & s & 0 & t_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1}$$

Of course, one could use four degrees of freedom to represent the object pose with different scale parameters s_x, s_y for different x, y directions. However, more degrees of freedom increase computational complexity. Therefore in our application, we restrict ourselves to three degrees of freedom.

4.2 State-Space Model

In general, a state-space representation model of a discrete-time linear dynamic system with additive white Gaussian noise can be written as follows:

$$x(k+1) = A(k)x(k) + B(k)u(k) + w(k) \tag{2}$$

where x(k) is the state vector of dimension n_x, $u(k)$ is the input (control) vector of dimension n_u and w_k is a zero-mean white Gaussian process noise vector of dimension n_x. $A(k)$ is a $n_x \times n_x$ system matrix, $B(k)$ is the input gain $n_x \times n_u$ matrix. The process noise covariance matrix can be denoted as $Q(k) = E[w(k)w(k)^T]$. This matrix can be adapted from time to time for 2D tracking to update the state of the particles.

Using the given pose representation, we can use three degrees of freedom t_x, t_y and scale parameter s to model a state vector. Because of random pedestrian motion, unknown car velocity, unknown car and pedestrian direction of motion, we can only model the state transition as a simple Brownian model without any information about velocity or acceleration for each incremental parameter of the target state. Therefore in our application we use a simple state transition model as follows:

$$S(k+1) = S(k) + w(k) \tag{3}$$

where $S(k)$ is the state vector at time k and $w(k)$ is the noise vector generated from the adaptive noise covariance matrix $Q(k)$.

4.3 Measurement Model

In our application, we use an implicit form of the measurement model to compare the initial object model with the target region hypothesis on the current image under a given pose. We use a likelihood function $P(z|s)$ to express the probability of the observed data if the hypothesis about target state is correct. This likelihood function helps us dealing with the uncertainty arising from nonlinear noise. The computed likelihood then is used to set the weight of each particle according to the probability of the target under a certain state hypothesis.

Many different likelihood models have been developed in computer vision. In our application, we use a single visual modality, a color-based likelihood. This likelihood function uses the Bhattacharyya distance[14] as a metric to compare the Hue-Saturation color histogram data with the corresponding color

histogram of the initial object model. Given a covariance of the residual noise R, a simple Gaussian likelihood function can be obtained by the following formula:

$$P(z|s) \propto exp\left(\frac{-B^2}{2R^2}\right) \tag{4}$$

where B is Bhattacharyya distance between two color histograms of the target region under a given hypothesis and a corresponding reference object model.

4.4 Particle Filter Tracking

There are three main steps in the tracking module:

- **Initialization step:** we initialize the set of particles for each target with a prior uniform distribution using a specific uniform noise range vector.
- **Prediction step:** from the previous state posterior, we use the state transition equation described in the previous section to generate the prior distribution. The prediction step is used to predict the states of sets of particles of every target. Note that different targets have different noise covariance matrices.
- **Correction step:** In the correction step, we use the measurement model described above to measure the likelihood of observed data under given state hypothesis. The result of the likelihood function is used to set the weight for each particle. In case of a highly nonlinear measurement model, Sample importance resampling is suitable to avoid the problem of degeneracy and to update the weight of low-weight particles.

Afterwards, we use the weighted average of the particle set to output the current pose hypothesis for each target.

5. RESULTS

We want to test the robustness of our tracker in a large number of frames. Because the output of the tracker is a rectangle, we refrain from using the distance-error in pixels as criteria for tracking evaluation. Instead, we use the same tracking evaluation methodology as in[15] to evaluate our tracker performance. The methodology tests the robustness of a tracker based on the overlap areas between tracker output and ground truth annotation data. An overlap that is greater than 0.33 is considered as a hit. Specifically, the proportion of overlap is computed as follows:

$$OverlapProportion = \frac{OverlapArea}{GroundTruthArea + TrackingArea - OverlapArea} \tag{5}$$

We use different data sets for testing. The purpose is to evaluate our system performance in different situations. Because our tracker is a feature based tracker, we want to evaluate its performance in the case of small target model variability. This means there is no big variation in the target appearance model. Then, we test our tracker with more challenging data sets that evaluate large target model variability. The purpose is to show the importance of updating the object appearance model during the tracking process because most traditional trackers are likely to fail when the object appearance model changes too frequently. Figure 4 shows our tracker tested on BoBot data sets[‡]. It consists of a 320×240 video sequence with more than 1000 frames taken from a moving camera at 25fps and encoded with MPEG2. The data set contains multiple temporary occlusions.

[‡]`http://www.iai.uni-bonn.de/~kleind/tracking/`

(a) Frame #141 (b) Frame #147 (c) Frame #171 (d) Frame #211

(e) Frame #234 (f) Frame #324 (g) Frame #557 (h) Frame #982

Figure 4. Tracker quality performance on BoBot data set. The ground-truth data is shown in green, while the tracker result is shown in blue.

The BoBot data set is provided together with a Java evaluation tool for result evaluation. After slight adaptation of the evaluation code, we process the data from random runs and compare it to the ground truth file to get a statistic value. The mean and standard deviation of the first run are shown in the Figure 5. In the figure is very little difference between the three analyzed trials. The variation between three tests is 0.0008401186. This shows that our tracker result is stable, although it uses a probabilistic approach. The total mean of overlap area of the first trial test is more than 0.6. This shows the effectiveness and robustness of our tracker because an overlap that is above 0.33 is considered as a hit.

After that we test the perfomance of our tracker on more challenging data set. Figure 6 shows the result of our tracker tested on ETHZ's image sequence[§] with moving camera in highly cluttered environment.

Our tracking framework currently supports unlimited numbers of targets . However, because we want to analyze data in a large number of frames, for evaluation only the two targets shown in Figure 6 were chosen to track and analyze results:

- Those two targets are visible in the whole image sequence of 354 frames.
- They are targets that are at the right most on the image.

ETHZ provides annotation data for the data set. But every target in the scene is annotated, numbered randomly. Therefore by choosing targets that are at the right most on the images, we can write a simple annotation filter to filter the annotation files of the two interesting targets only. The data annotation is used to compare with tracker result. The statistics of tracker performance of two targets then are shown in the Figure 7(a).

As shown in Figure 7(a), while the tracking result is quite stable for the first target (yellow) on 354 frames of the whole ETHZ image sequence, the second target is declared lost during frame #140 to

[§]`http://www.vision.ee.ethz.ch/~aess/dataset/`

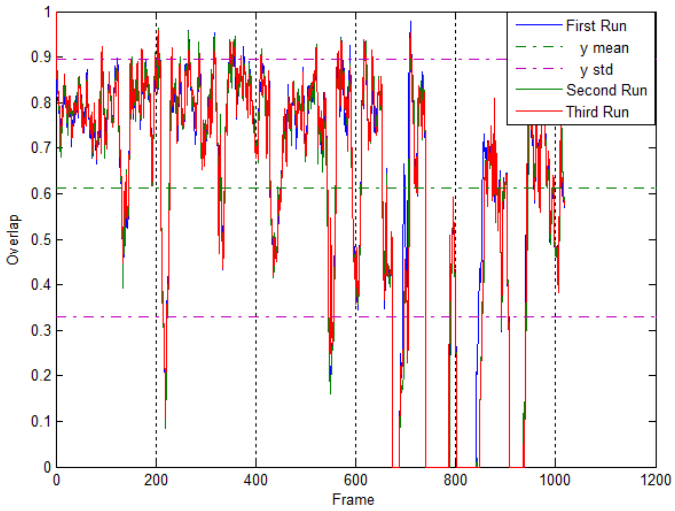
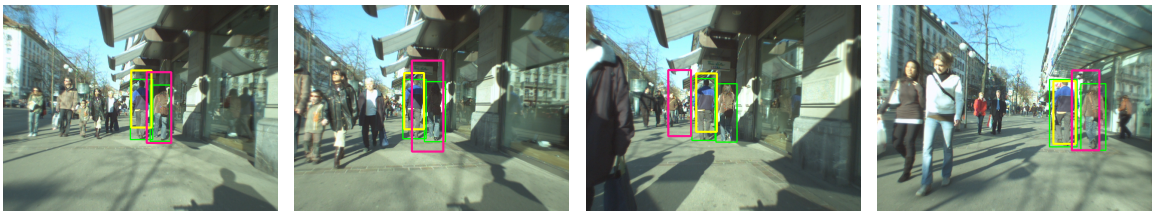


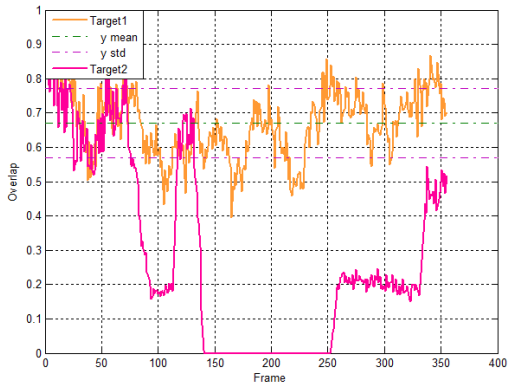
Figure 5. Descriptive statistics of three runs show the effectiveness and robustness of the tracker.

(a) Frame #9 (b) Frame #29 (c) Frame #50 (d) Frame #84

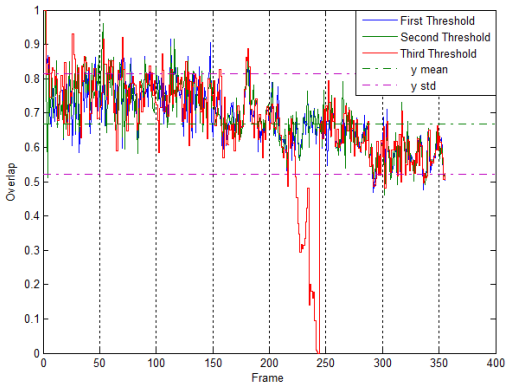


(e) Frame #118 (f) Frame #136 (g) Frame #155 (h) Frame #349

Figure 6. Tracker quality performance within a highly cluttered scenario. The ground-truth datas are shown in green, while the tracker results are shown in yellow and pink. No detector feedback is used in the tracking process.



(a) Only tracker is used without detector feedback on both targets.



(b) Detector feedback is used to correct tracking error on the second target. Threshold values increase from thrid to first.

Figure 7. Results for multi-target tracking on the ETHZ image sequence with and without detector feedback.

frame #250. From our observation, the tracker loss second target when it goes below the shadow of the building. The first target, on other hand, doesn't change its appearance model much during the whole image sequence. Therefore, the tracker can handle the first target successfully but fails when the appearance model of the second target varies much compared to its initial object model. Because the tracker successfully handles target 1 (yellow), we focus the discussion on how to improve the performance of tracking the second target (pink). At the right of Figure 8 is the tracking result when integrating detector and tracker. The proposed system uses a predefined likelihood threshold. Our scheme is to update the appearance model and reference histogram using the HOG detector whenever the likelihood for observed data and initial data drops below this threshold. As shown in Figure 7(b), there is a big improvement for tracking second target. Figure 7(b) shows the tracker performance tested with three different predefined likelihood thresholds that we use as thresholds to update target appearance model. As shown in Figure 7(b), tracker performance with the first and second likelihood thresholds seems to be better compared to tracker performance with the third one. This can be explained as follow:

- Tracker with first or second likelihood threshold is a typical type of a tracker based on detector. For this kind of tracker, information from the detector is used extensively. This type of tracker is generally more computationally expensive.
- Tracker with third likelihood threshold is a normal tracker with detector feedback. For this kind of tracker, information from detector is used only in a few occasions. This type of tracker is in general more efficient considering computational resources.

For both kinds of above trackers, when the likelihood between the observed data and initial data drops below the predefined threshold, the information from detector is used to reset states of every particles for every targets. The adaptive motion noise covariance matrix $Q(k)$ is also reset. The reference color histogram is re-computed under a given pose. As shown in Figure 7(b), even with the third threshold, the tracker performance for the second target has been improved a lot compared to the tracker without detector feedback.

(a) Frame #9 (b) Frame #29 (c) Frame #50 (d) Frame #84

(e) Frame #118 (f) Frame #136 (g) Frame #155 (h) Frame #349

Figure 8. Tracker quality performance on a challenging ETHZ image sequence. The ground-truth data is shown in green, while the tracker results are shown in yellow and pink. Detector feedback is used in the tracking process.

6. CONCLUSION

In this paper, we propose a novel method for the pedestrian detection and tracking problem. Our vision system has 3 main modules. The tracking module uses *Histograms of Oriented Gradients* to detect pedestrians on a single static image. After that, we extract the depth information by using *Direct Linear Transformation* to triangulate correspondent points found by *Speeded Up Robust Features*. A particle filter approach then is proposed as a robust solution for the tracking problem. It can track targets reliably on a simple background case. However, to achieve stable performance on a real world scenario with complex background and cluttered environment, we use the detector to provide feedback to the tracking process. By integrating detector and tracker, we show that the stablility of tracking performance increases significantly, even in highly cluttered environment.

In future work, we plan to use depth information more extensively. The depth cue and other cues can be fused to provide an even more reliable measurement model. We also plan to optimize the system by offloading expensive computation to the GPU. As a result, real-time performance can be achieved.

ACKNOWLEDGMENTS

This work is supported by the German Research Foundation (DFG) within the Collaborative Research Center SFB 453 on *High-Fidelity Telepresence and Teleaction*[¶] and the Cluster of Excellence CoTeSys *Cognition for Technical Systems*[‖].

REFERENCES

[1] Kolmogorov, V. and Zabih, R., "Computing visual correspondence with occlusions via graph cuts," in [*In International Conference on Computer Vision*], 508–515 (2001).

[¶] `http://www.sfb453.de`
[‖] `http://www.cotesys.org`

[2] Kolmogorov, V. and Zabih, R., "Multi-camera scene reconstruction via graph cuts," in [*in European Conference on Computer Vision*], 82–96 (2002).

[3] Bay, H., Tuytelaars, T., and Gool, L. V., "Surf: Speeded up robust features," in [*In ECCV*], 404–417 (2006).

[4] Terriberry, T. B., French, L. M., and Helmsen, J., "Gpu accelerating speeded-up robust features."

[5] Leibe, B., Seemann, E., and Schiele, B., "Pedestrian detection in crowded scenes," in [*In CVPR*], 878–885 (2005).

[6] Porikli, F., Meer, P., Tuzel, O., and Tuzel, O., "P.: Human detection via classification on riemannian manifolds," in [*In Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*], 1–8 (2007).

[7] Dalal, N. and Triggs, B., "Histograms of oriented gradients for human detection," in [*In CVPR*], 886–893 (2005).

[8] Viola, P. and Jones, M., "Rapid object detection using a boosted cascade of simple features," 511–518 (2001).

[9] Prisacariu, V. A. and Reid, I., "fasthog- a real-time gpu implementation of hog technical report no. 2310/09," (2009).

[10] Schweighofer, G. and Pinz, A., "Globally optimal o(n) solution to the pnp problem for general camera models."

[11] Panin, G., Lenz, C., Nair, S., Roth, E., Wojtczyk, M., Friedlhuber, T., and Knoll, A., "A unifying software architecture for model-based visual tracking," in [*IS&T/SPIE 20th Annual Symposium of Electronic Imaging*], (2008).

[12] Ess, A., Leibe, B., Schindler, K., , and van Gool, L., "A mobile vision system for robust multi-person tracking," in [*IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*], IEEE Press (June 2008).

[13] Hartley, R. I. and Zisserman, A., [*Multiple View Geometry in Computer Vision*], Cambridge University Press, ISBN: 0521623049 (2000).

[14] Bhattacharyya, A., "On a measure of divergence between two statistical populations defined by their probability distributions," *Bull. Calcutta Math. Soc.* **35**, 99–109 (1943).

[15] Klein, D. A., Schulz, D., Frintrop, S., and Cremers, A. B., "Adaptive real-time video-tracking for arbitrary objects," in [*IROS*], to appear (2010).