Stixels Motion Estimation without Optical Flow Computation

Bertan Günyel[1,2], Rodrigo Benenson[1], Radu Timofte[1], and Luc Van Gool[1]

[1] ESAT-PSI-VISICS/IBBT, Katholieke Universiteit Leuven, Belgium
firstname.lastname@esat.kuleuven.be
[2] 3cap Technologies Gmb, Oberschleißheim, Germany
bertan.guenyel@3cap.de

Abstract. This paper presents a new approach to estimate the motion of objects seen from a stereo rig mounted on a ground mobile robot. We exploit the prior knowledge on ground plane presence and rough shape of objects, to extract a simplified world model, named *stixel world*. The contribution of this paper is to show that stixels motion can be estimated directly solving a single dynamic programming problem instead of an image wide optical flow computation. We compare this new method with baseline methods, show competitive results quality-wise, and a significant gain speed-wise.

Fig. 1. Pipeline of our motion estimation method. We focus on step 2.

1 Introduction

For safe and robust navigation mobile robots require the prediction of traversable space in time. In order to do such prediction we need to detect surrounding obstacles, classify them as mobile or static, and estimate their state vector (e.g. position, velocity, orientation).

In this paper we focus on the estimation of objects motion between two consecutive frames. This information can be used to feed an object tracker or as an additional cue for object classification.

Traditionally, motion estimation between two frames is done using optical flow methods, which are computationally expensive. The aim of this paper is

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part VI, LNCS 7577, pp. 528–539, 2012.

to propose a lighter method for objects motion estimation in the context of a ground mobile robot.

We base our approach on the stixel world model [1]. Given an input stereo pair, the stixel world model will estimate the ground plane and the distance to the main objects in the scene (represented by "sticks" raising from the ground, see figure 2). The key attribute of such model is that it only focuses on the dominant objects of the scene. By avoiding the computation of a pixel-wise depth map, such model can be estimated much faster than traditional stereo matching methods [2].

(a) The stixel world

(b) Estimated stixels, yellow line indicates the objects bottom

Fig. 2. The stixel world is composed by the ground plane and vertical sticks describing the obstacles. Illustration from [2].

In this paper we show that the motion of stixels can be computed without the need of computing a full pixel-wise optical flow. Instead, we can formulate the problem as a simple dynamic programming problem; by doing so we significantly reduce the computational load, while keeping good quality in the motion estimation (see figure 1).

The key insight of this work is realizing that in the stixel world, objects motion estimation can be reduced to a 1D problem (solved via 2D dynamic programming).

Our method will only compute the motion in the areas covered by the stixels. It has been previously shown that stixels represent the dominant objects of the scene adequately [2]. For applications involving moving objects detection and object tracking, we argue that stixels motion is the sufficient level of detail.

Assumptions. We assume calibrated stereo input, and that the stixel world model holds (as verified by [3] for urban scenes); i.e. the ground is locally planar and objects of interest are mainly vertical. We also assume that the cameras are roughly parallel to the horizon, that the objects of interest have a height in a known limited range (e.g. 0.5 to 3 m) and that the motion of objects is bounded by a known maximum speed.

In what follows of this section we discuss how our work contrasts to previous ones. In section 2, our stixels motion estimation algorithm is detailed. Section 3 explains our evaluation protocol and presents experimental results. We conclude in section 4.

1.1 Related Work

The stixel world model was introduced not long ago [1], so little literature exists yet on the subject.

Typically motion estimation is done either using optical flow [4] and/or some kind of tracking-by-detection framework [5,6,7]. Recent progress has enabled running optical flow in real-time [8,9]; however, it still requires specialized hardware (GPU) and significant computational resources. Our approach is, by design, much simpler and can run in real-time using CPU only. Tracking-by-detection approaches intrinsically require some kind of class specific knowledge. Our proposed method is class agnostic and attempts to track any dominant object (captured by the stixels) using only appearance and depth cues.

The notion of *dynamic stixels* was introduced by Pfeiffer and Franke [3]. There the velocity vector of each stixel is estimated using optical flow as input and per-stixel Kalman filters for smoothness. However, the presented results lack any quantitative evaluation of the estimated motion. In this paper we propose to obtain similar results, but without requiring the computation of optical flow.

An interesting related work was recently proposed by Mitzel et al. [10]. They focus on improving detection speed by exploiting the information contained in depth maps. Given a detected object, they propose to crop the covered depth map and track directly the point cloud across frames. With a successful tracking they avoid the need of re-detecting the objects on every frame. Although not directly related with generic objects motion (due to the classification step), we will use this approach for our evaluation in section 3. In our approach, pixel-wise depth maps are not computed.

1.2 Contribution

The contribution of this paper is two-fold:

1. We propose the first algorithm for stixels motion estimation without requiring the computation of optical flow. This enables much faster computation while keeping good quality.
2. We present the first evaluation of the stixels motion quality. Previous work on the topic simply skipped such evaluation [3]. We compare our method against two baselines.

In the next section we describe our algorithm in detail.

2 Stixel Motion Estimation

Stixels are estimated as an intermediate representation of obstacles in the scene for mobile robot navigation. They can be estimated by initially computing a depth map and using dynamic programming [1] or without the computationally expensive depth map [2]. The details of how stixels are estimated are beyond the scope of this paper.

Having estimated the stixels for the *current frame* at time t_1 and having kept the stixel estimations from the *previous frame* at time t_0, the stixel motion estimation process can be viewed as a matching problem between stixels at time t_1 and stixels at time t_0. For the sake of simplicity, and without loss of generality, we will assume that each stixel covers a segment of one column u in the image. The stixel motion estimation corresponds to a search along the u direction.

The proposed method for stixel motion estimation includes two main steps. Having estimated the stixels at time t_1 and t_0, a matching cost matrix is computed (section 2.1) which is then used by the dynamic programming module to find the optimal motion assignment for each stixel (section 2.2).

2.1 Matching Cost Matrix Computation

Assuming properly estimated stixels and a small time difference between two consecutive frames, the appearance of a stixel estimated at the current frame should be similar to the corresponding stixel estimated at the previous frame. They also should have the same height, in meters. We have no particular assumption on the motion of the obstacles in the scene, other than a known maximum speed v_{max} (set to 2.5 meters per seconds).

Without loss of generality, we assume that stixels have width of one pixel. We use u to indicate a stixel at column u, and u_1, u_0 to indicate stixels covering the same object at $t_1$and t_0 respectively.

Given the set of stixels estimated at each column of the current frame t_1 and the set of stixels estimated at each column of the previous frame t_0, the stixel motion estimation can be reformulated as a procedure for the assignment of motion $m^*(u_1)$ (in image columns), to the stixel at u_1, such that $u_0 = u_1 + m^*(u_1)$.

The motion cost $c_m(u_1, m)$ is computed as,

$$c_m(u_1, m) = \begin{cases} \alpha \cdot SAD(u_1,\ m) + (1-\alpha) \cdot |h_1(u_1) - h_0(u_1 + m)| & \text{if } m \in M(u_1) \\ c_{null} & \text{otherwise} \end{cases} \tag{1}$$

where, $h_i(u)$ is the height, in meters, of the stixel at column u on frame t_i. α is a scaling parameter set to 0.5 in our experiments. $M(u)$ is the set of possible motions for the stixel u (with respect to the previous frame), it depends on the depth $z_1(u)$ of the stixel and the frame rate at which the images are captured. The cost value for non-valid motions is set to c_{null}, set to 0.6 times the maximum possible cost value.

$SAD\,(u_1,\ m)$ is the pixel-wise sum of absolute differences over the RGB colour channels between stixels u_1 and $u_0 = u_1 + m$. A fixed (sampled) pixel-wise height (e.g. 30 pixels) is used for comparison since the stixel heights, in pixels, can be different.

The stixels are estimated from stereo data. Thus, some of them correspond to occluded areas for which no reliable information is available [2]. Accordingly, the entries of the cost matrix $c_m\,(u_1,m)$ are directly set to the maximum cost value if one or both of the stixels u_1 and u_0 correspond to occluded areas.

Stixels do not always have a valid match on the other frame, especially if they correspond to appearing or disappearing objects. For instance, a stixel seen for the first time at the current frame does not have a valid match on the previous frame and it cannot have a meaningful motion estimate. In order to handle this, we introduce the label *null motion* $m_{null} \notin M(u)$ that is assigned to any stixel u if its assigned motion (by the dynamic programming) has a cost value c_m higher than c_{null}, i.e. it is a mismatch.

An example motion cost matrix c_m is illustrated in figure 3 with the estimated motion for each stixel drawn in pink. The column index of the cost matrix is the stixel horizontal coordinate, u, whereas the row coordinate represents the 1D motion values, m, of the stixel, which can be positive, zero, or negative. The size of the cost matrix is $(2 \times maximum\,1D\,motion + 1 + 1) \times number\,of\,stixels$, representing negative, zero, and positive motions in addition to the *null motion*.

2.2 Dynamic Programming for Stixel Motion Estimation

The optimal displacement in pixels $m^*\,(u_1)$ is estimated via a standard 2D dynamic programming minimization, similar to the one presented in [2]. The minimization problem to solve is,

$$m^*\,(u_1) = \underset{m(u_1)}{\operatorname{argmin}} \sum_{u_1 \in U_1} c_m\,(u_1, m\,(u_1)) + \sum_{u_1\ \in\ U_1} n_m\,(m\,(u_1)\,, m\,(u_1 + 1)) \quad (2)$$

where U_1 is the set of all columns on the input frame and $n_m\,(m_a, m_b)$ is a smoothing constraint applied over neighbour stixels.

Neighbouring stixels can either correspond to the same object or to separate neighbouring objects. If the stixels correspond to the same object, they should have very similar motion. Hence, the degree of neighbouring constraint between adjacent stixels should be proportional with the likeliness of the stixels to belong to the same object.

Objects in a scene can have different orientations with respect to the camera. Depending on the object orientation, different parts of the same object might have different depth values. However, there is a spatial continuity in depth in small neighbourhoods of the objects. Thus, neighbour stixels corresponding to the same object have similar depth values. In addition, spatially neighbouring stixels of the same object are generally observed to have similar height values.

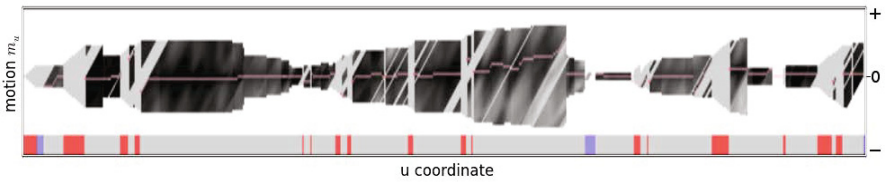


Fig. 3. Example motion cost matrix (frame 8). Estimated m_u^* shown in pink. Red stixels in the lower part correspond to occluded areas in either of the stereo images, blue stixels are the ones assigned to *null motion*, grey stixels are estimated normally.

The smoothing term $n_m\left(m_{u_1}, m_{u_1+1}\right)$, defined in equation (3), reflects these weak cues.

$$n_m\left(m\left(u_a\right), m\left(u_b\right)\right) = \left|m_a - m_b\right| \cdot \left(\beta_z + \beta_h\right) \tag{3}$$

$$\beta_z = k_z \cdot \max\left(\alpha_z, 1 - \frac{\left|z_1\left(u_a\right) - z_1\left(u_b\right)\right|}{\Delta z}\right) \tag{4}$$

$$\beta_h = k_h \cdot \max\left(\alpha_h, 1 - \frac{\left|h_1\left(u_a\right) - h_1\left(u_b\right)\right|}{\Delta h}\right) \tag{5}$$

In (4) and (5), $z_1\left(u\right)$ and $h_1\left(u\right)$ represent respectively the depth and height (both in meters) of stixel u at time t_1. Δz, Δh, k_z, k_h, α_z, α_h are parameters controlling the extent of stixel depth and stixel height on the smoothing constraint. In our experiments we use $\Delta z = 3$, $\Delta h = 1$, $\alpha_z = 0.1$, $\alpha_h = 0.5$, $k_z = p_{max}$, $k_h = 0.2 \cdot p_{max}$ where p_{max} is the maximum possible pixel value in the input images.

The dynamic programming computations are performed in two passes. The dynamic programming matrix D_m is initialized with the values of the matrix c_m. In the first pass, $D_m\left(u, m\right)$ is computed recursively (from right to left) as given in (6).

$$D_m\left(u_1, m\right) = \begin{cases} c_m\left(u_1, m\right) & \text{if } u_1 = u_{max} \\ \min\limits_{e \in M(u_1+1)} \left(D_m\left(u_1+1, e\right) + n_m\left(m, e\right)\right) & \text{otherwise} \end{cases} \tag{6}$$

The stixel motion estimations are obtained with backtracking in the second pass (from left to right) of dynamic programming as shown in (7).

$$m_{u_1}^* = \begin{cases} \operatorname{argmin}\limits_m D_m\left(1, m\right) & \text{if } u_1 = 1 \\ \operatorname{argmin}\limits_m \left(D_m\left(u_1, m\right) + n_m\left(m_{u_1-1}^*, m\right)\right) & \text{otherwise} \end{cases} \tag{7}$$

3 Evaluation

3.1 Methodology

The only previous work on stixels motion estimation provided no quantitative evaluation of their results [3]. Our paper is the first to provide quantitative evaluation of the stixels motion. Doing such an evaluation is difficult since no ground truth is available for the 3D world; optical flow evaluations have the same issues. Using synthetic video sequences for evaluation is possible, but this provides no confidence on the expected real-world performance.

Similar to [2], we propose to use annotated pedestrian bounding boxes as a proxy for our evaluation. Starting from ground truth annotations at frame n, each evaluated algorithm is used to predict the bounding box positions up to Δ frames in the future. For each predicted frame the recall is evaluated using the standard intersection-over-union (superior to 0.5). By running this evaluation starting from every frame in a video sequence we obtain a "recall versus Δ frames" curve that can be used to compare algorithms. To the best of our knowledge this is the first time such an evaluation is done.

Since we are interested in objects tracking, we prefer to use an object level measure rather than standard sub-pixel optical flow accuracy measures. By evaluating over multiple frames we have (indirectly) access to the overall accuracy of the method.

We evaluated the following algorithms:

`fixed.` As a baseline for bad performing algorithms, we use the simplest prediction method possible: we assume zero motion. For each predicted frame the detection bounding boxes are kept in the same position as the initial one. No algorithm should be worse than this method.

`greedy tracker.` This method serves as a baseline for good performing algorithms. For each prediction frame, we estimate the bounding box motions by doing a greedy matching between the current estimate and the ground truth annotations for that frame. In this method the recall still falls as Δ increases since pedestrians not present in the initial frame enter the scene in following frames.

Although it is not a strict upper bound, since this algorithm accesses the ground truth annotations to do the motion prediction, it is expected to be better than any other method.

`ICP tracker.` For each frame, we compute (offline) a dense depth map [11] (same as the one used in [10], see figure 4b). For each initial frame, we use the ground truth annotations to extract a mask corresponding to the upper half of the body (one mask for each annotation, see figure 4d). We then sample 100 points randomly inside the mask and extract their 3D positions. Following the work of Mitzel et al. [10], we track these point clouds using iterative closest point (ICP) matching (10 iterations per frame).

After estimating the 3D motion between two frames, we use an assumed person height (1.8 meters), and the ground plane from the stixel world model [2] to estimate bounding boxes on the new frame. The point cloud is kept

fixed, and the ICP matching + bounding box estimation is repeated for each predicted frame.

In their original work Mitzel et al. [10] evaluated their algorithm integrated with a high level tracker, showing state-of-the-art results. However, it is hard to know if such good results are obtained due to the object detector quality, the high level tracker, or the low level ICP tracker. Our evaluation is more focused and allows a better comparison.

`optical flow.` We compute (offline) a high quality optical flow (using [4], see figure 4c) for each frame. At each frame we use the estimated stixels and optical flow to propagate the detection windows across time. Given the high quality of the optical flow, simply sampling the middle point of the torso provided the best results (better than when using a mask based on the depth map). This baseline aims at emulating the results of [3]. In fact, [3] includes more than optical flow and it should perform better than sole optical flow as long as the motion fits in the assumed (class specific) motion model. However, in our work, we do not assume such model to make the comparison fair.

Since this method uses a full-fledged optical flow we expect it to provide competitive results.

`stixels motion.` We compute the stixel world [2] and the stixels motion for each frame. Given this data we can predict the 3D motion between each frame (since stixels include depth information), and then use the same height and ground planes as the previous methods to estimate the bounding boxes on each predicted frame.

3.2 Quantitative Results

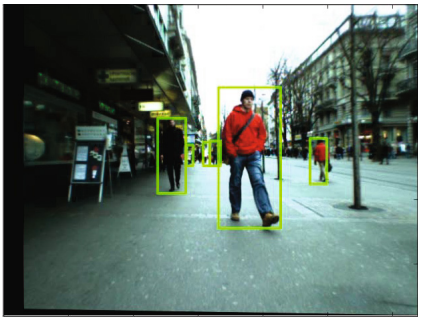
The proposed algorithm is evaluated on the "Bahnhof" sequence [12], which is a challenging sequence, already used to test object detection and tracking. This stereo sequence was captured from a child stroller on a side walk. It provides ~ 7400 annotations of pedestrians with height ≥ 40 pixels on 999 frames, with an image resolution of 640×480 pixels and a frame rate of ~ 15 fps.

The recall vs Δ frames curves are presented in figure 5. This result is obtained over the full Bahnhof sequence.

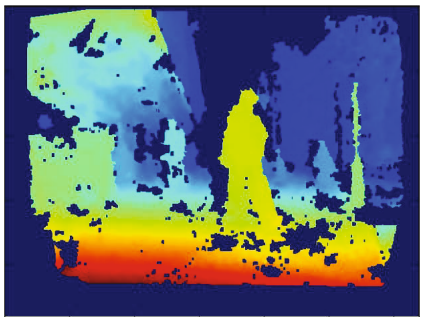
We observe that `optical flow` provides the best results, following closely the (ground truth based) results presented by `greedy tracker`. As expected no method is worse than `fixed.` The `ICP tracker` shows a low recall in the initial frames, this indicates that our implementation fails to initialize valid depth masks (due to noise or lack of depth information).

Our `stixels motion` method provides results that lie in between the `ICP tracker` and the high quality `optical flow` results. We obtain comparable quality, yet our algorithm runs significantly faster.

Speed. Both depth map computation and optical flow computation require expensive optimizations over every pixel in the image. In comparison, our method only requires solving a single 2D dynamic programming problem, that accesses

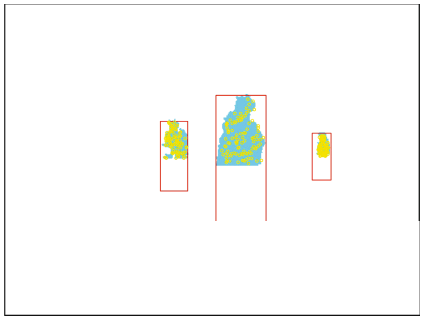


(a) Input frame 8, with ground truth annotations



(b) Corresponding depth map

(c) Corresponding optical flow



(d) Extracted masks and sampled points for ICP

Fig. 4. Example data used in the different algorithms

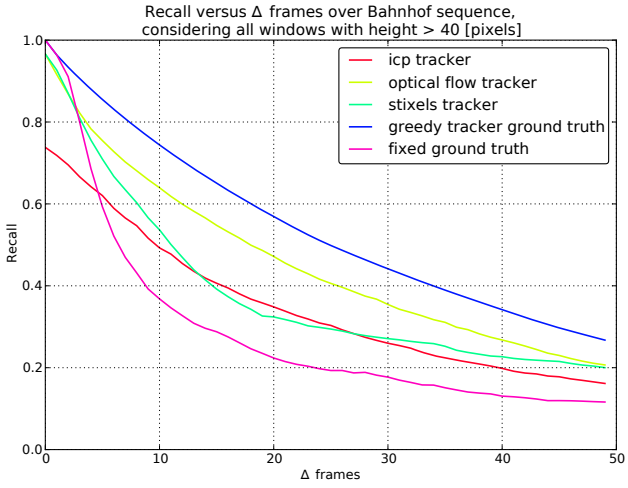


Fig. 5. Tracking capabilities over different Δ frames

a few pixels of the image for the data term. When properly implemented, this approach can comfortably run at more than 30 Hz in a modern CPU. Our current naive C++ implementation (single thread, memory inefficient, no SIMD), runs at 11 Hz. This number can be easily improved ($3\times \sim 10\times$); as a reference the dynamic programming used to estimate the stixels does very similar computations, and runs at 100 Hz on CPU [2].

3.3 Qualitative Results

To complement the quantitative results of section 3.2, we present typical results of our algorithm in figures 6 and 7.

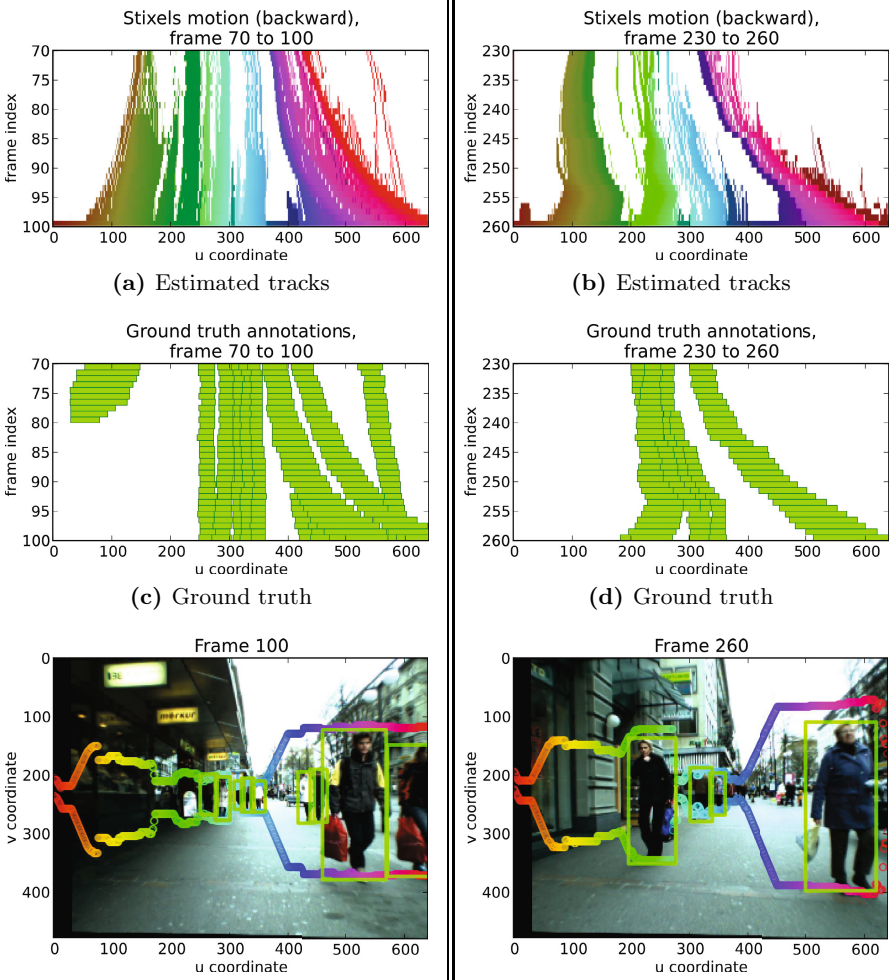
In figure 6 we colour coded the stixels to indicate the corresponding image motion vector. It can be seen that different objects are indeed associated to different type of displacements. These results are computed using only two consecutive stereo frames.

Fig. 6. Example results of our stixel motion estimation algorithm. Motion vector orientation is colour coded (see image border). Left image shows frame 8.

In figure 7 we show the results of linking multiple consecutive motion estimates. In figures 7a and 7b we present motion *tracks* over 30 frames of the Bahnhof sequence (from frame 70 to 100, and frames 230 to 260, respectively). The bottom rainbow identifies the stixels in the last frame. Stixels corresponding to ground truth pedestrian annotations in the last frame are marked with a lighter colour (see figures 7e and 7f). For each previous frame we can see the estimated stixel position, based on repeatedly propagating (backwards) the frame-wise stixel motion estimation.

As a reference point for the computed tracks we show the ground truth annotations per frame (horizontal u dimension only). The tracks in figures 7e and 7f should match the ground truth tracks on in figures 7c and 7d, respectively.

It can be seen in these illustrative examples that our method can track visible pedestrians along 30 frames.



(a) Estimated tracks **(b)** Estimated tracks

(c) Ground truth **(d)** Ground truth

(e) Stixels at frame 100 **(f)** Stixels at frame 260

Fig. 7. Example of tracks obtained by our algorithm. Sub-figures (a) and (b) show the tracks obtained by backward propagation of the stixels of the latest frame (shown in (e) and (f) respectively). Lighter colours indicate stixels marked as "pedestrian" in the ground truth annotation of the latest frame. Darker colours are other tracked objects (the wall).

4 Conclusion and Future Work

In this paper, it is shown that stixels motion estimation can solve the short term data association problem. The described stixels motion estimation algorithm obtains comparable performance with competing methods without the need of depth map or optical flow computation. While having comparable results quality wise, our method is significantly faster than depth map or optical flow computation. Both the proposed algorithm and the quantitative evaluation of stixel motion methods are novel results.

Stixels motion estimation can be used as a low level tracker that feeds information to a higher level tracker. We are currently working on such an integration.

Acknowledgement. This work has been partly supported by the Toyota Motor Corporation and the ERC grant COGNIMUND.

References

1. Badino, H., Franke, U., Pfeiffer, D.: The Stixel World - A Compact Medium Level Representation of the 3D-World. In: Denzler, J., Notni, G., Süße, H. (eds.) DAGM 2009. LNCS, vol. 5748, pp. 51–60. Springer, Heidelberg (2009)
2. Benenson, R., Timofte, R., Van Gool, L.: Stixels estimation without depthmap computation. In: ICCV, CVVT Workshop (2011)
3. Pfeiffer, D., Franke, U.: Efficient representation of traffic scenes by means of dynamic stixels. In: IVS (2010)
4. Sun, D., Roth, S., Black, M.: Secrets of optical flow estimation and their principles. In: CVPR (2010)
5. Leibe, B., Schindler, K., Van Gool, L.: Coupled detection and trajectory estimation for multi-object tracking. In: ICCV (2007)
6. Andriluka, M., Roth, S., Schiele, B.: People-tracking-by-detection and people-detection-by-tracking. In: CVPR (2008)
7. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. PAMI (2011)
8. Zach, C., Pock, T., Bischof, H.: A Duality Based Approach for Realtime TV-L^1 Optical Flow. In: Hamprecht, F.A., Schnörr, C., Jähne, B. (eds.) DAGM 2007. LNCS, vol. 4713, pp. 214–223. Springer, Heidelberg (2007)
9. Sundaram, N., Brox, T., Keutzer, K.: Dense Point Trajectories by GPU-Accelerated Large Displacement Optical Flow. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part I. LNCS, vol. 6311, pp. 438–451. Springer, Heidelberg (2010)
10. Miztel, D., Leibe, B.: Real-time multi-person tracking with detector assisted structure propagation. In: Proc. of the Int. Conf. on Computer Vision (ICCV), Workshop on Challenges and Opportunities in Robot Perception, CORP (2011)
11. Geiger, A., Roser, M., Urtasun, R.: Efficient Large-Scale Stereo Matching. In: Kimmel, R., Klette, R., Sugimoto, A. (eds.) ACCV 2010, Part I. LNCS, vol. 6492, pp. 25–38. Springer, Heidelberg (2011)
12. Ess, A., Leibe, B., Schindler, K., Van Gool, L.: Robust multi-person tracking from a mobile platform. PAMI (2009)