A Monocular Collision Warning System

Felix Woelk[1] Stefan Gehrig[2] Reinhard Koch[1]

[1] *Institute for Computer Science*
Christian-Albrechts-Universität zu Kiel
24118 Kiel, Germany
{woelk, rk}@mip.informatik.uni-kiel.de

[2] *DaimlerChrysler AG*
HPC T728
70546 Stuttgart, Germany
stefan.gehrig@daimlerchrysler.com

Abstract

A system for the detection of independently moving objects by a moving observer by means of investigating optical flow fields is presented. The usability of the algorithm is shown by a collision detection application. Since the measurement of optical flow is a computationally expensive operation, it is necessary to restrict the number of flow measurements. The first part of the paper describes the usage of a particle filter for the determination of positions where optical flow is calculated. This approach results in a fixed number of optical flow calculations leading to a robust real time detection of independently moving objects on standard consumer PCs. The detection method for independent motion relies on knowledge about the camera motion. Even though inertial sensors provide information about the camera motion, the sensor data does not always satisfy the requirements of the proposed detection method. The second part of this paper therefore deals with the enhancement of the camera motion using image information. The third part of this work specifies the final decision module of the algorithm. It derives a decision (whether to issue a warning or not) from the sparse detection information.

1. Introduction

Moving objects exhibit potential threats in traffic situations and hence this work focuses on the detection of such objects. Using the human head as an inspiration, a lightweight monocular camera mounted on a pan-tilt-unit (PTU) is chosen to investigate the environment in this application. The analysis of optical flow fields gathered from this camera system is a cheap and straight forward approach avoiding heavy and sensitive stereo rigs. A system using the detection of moving objects for the task of collision warning is presented. The system is easily adaptible to mobile robot applications.

After a short system overview, the basic concept for the detection of independent motion and the particle filter used for the detection is described (section 2). In section 3 the estimation of the camera motion from a fusion of car inertial sensor data and image information is characterized. The exact knowledge of camera motion is necessary for the detection method. Afterwards the decision algorithm which is responsible for issuing a warning if any of the moving objects are on a collision trajectory is presented (section 4).

Figure 1. The UTA demonstrator from the Daimler Chrysler AG. Only one of the two PTU mounted cameras is used in this work.

System Overview: The demonstrator from the Daimler Chrysler AG, which is used in this work, is called Urban Traffic Assistant (UTA) (fig. 1). The setup of UTA [11] includes a digital camera mounted on a pan-tilt-unit (PTU), GPS, map data, internal velocity and yawrate sensors, etc. The fusion of GPS and map data could be used to announce the geometry of an approaching intersection to the vision system. The camera then focuses on the intersection. Using

the known egomotion of the camera, independently moving objects are detected and the driver's attention can be directed towards them.

2. Detection

The basic concepts used for the detection of independently moving objects by a moving observer through investigation of the optical flow are introduced in this section.

Computation of Optical Flow: A large number of algorithms for the computation of optical flow exist [6]. Any of these algorithms calculating the full 2D optical flow can be used for the proposed algorithm. Algorithms calculating the normal flow only (i.e. the flow component parallel to the image gradient) are, however, inappropriate. The optical flow in this work is calculated using an iterative gradient descend algorithm [2], applied to subsequent levels of an image pyramid.

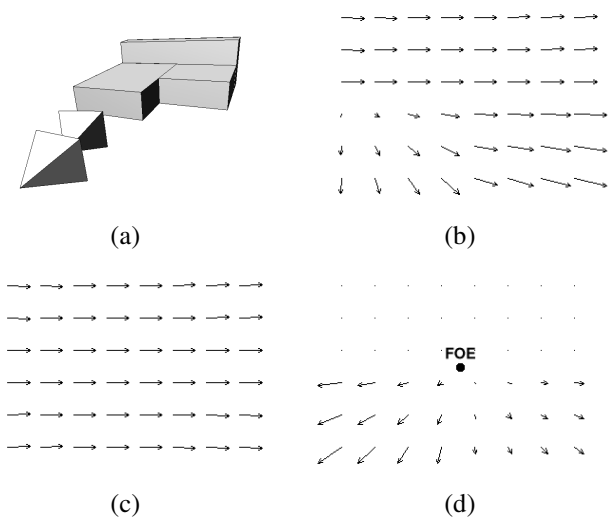


Figure 2. Theoretical flow field for a simple scene. The 3D scene is shown at (a). The scene consists of 3 blocks. The camera, displayed as small pyramids, translates towards the blocks while rotating around the y axis. The flow field F as induced by this movement is shown in (b). Its rotational component F_R (c) and translational component F_T (d) with the Focus of Expansion (FOE) are shown on the right.

Detection of Independent Motion: Consider the optical flow induced by the translational velocity v_c of a camera observing a rigid static scene. Changing the frame of reference using a Galilei transformation [28] does not alter the imaging process and hence the same flow field may arise from a camera moving with $v'_c = v_c + v_a$ viewing the same rigid scene moving with $v'_s = v_a$. Only the relative velocity $v_r = v'_c - v'_s = v_c$ between camera and object determines the optical flow field. Similar considerations lead to the fact that only the relative rotation between camera and object is relevant for the optical flow field. In the following all considered motions and rotations are relative.

An arbitrary rigid motion can always be described by a rotation R followed by a translation T. Hence the optical flow field F induced by the motion is the superposition of the optical flow field F_R resulting from the rotation R and the optical flow field F_T resulting from the translation T (Fig. 2). Optical flow resulting from a rotation is independent of the scene geometry and can be computed directly if the rotation is known. A flow field resulting from a translation has simple geometric properties: All flow vector point radially away from the focus of expansion (FOE) and radially towards the focus of contraction (FOC). The locations of the FOE and FOC are determined by the camera translation and do not necessarily lie in the image. The length of each flow vector depends on the distance of the projected object from the camera center and is hence scene dependent.

Now consider again a moving camera viewing a static rigid scene. Given a known camera motion T and R and an optical flow field F, the translational part of the flow field F_T can be calculated by subtracting F_R (computed from R) from F (measured in the images). The direction of the translational part of the flow field can also be calculated using the known camera motion. Comparing the directions of these two translational flow fields (i.e. by using the angle between the vectors) yields points where the underlying model is violated. Neglecting measurement errors (both for the camera motion and the optical flow), this results in the fact that the underlying (relative) motion was inconsistent with the camera motion. Hence an object which moved with respect to the static scene has been projected to the points where the direction of the two flow fields do not coincide [5].

In our approach, the camera motion (relative to the static scene) can be derived from rotation sensor and speed sensor data of the car, or it can alternatively be measured directly from the images if only the static scene is pictured [9]. Since the car inertial sensor do not capture certain movements (e.g. pitching) and are corrupted by measurement noise, a robust approach for the refinement of the necessary camera motion parameters from image information is necessary. It is described in section 3.

Fast Detection Of Independent Motion: Since determining highly accurate optical flow (OF) with subpixel preci-

sion is a computationally expensive operation, the number of OF measurements have to be restricted in real time environments. However, when computing OF at sparse locations, one would like to capture as much flow on independently moving objects as possible. An adapted particle filter is chosen for this task. The approach chosen in this work is inspired by [19] and determines the sample positions (i.e. points where optical flow will be calculated) partly by using a vector of random variables, which are distributed according to an initialization distribution function (IDF), and partly by propagating samples from the last time step using a particle filter approach.

Particle Filter: The idea of the particle filter has been first published in 1949 by Metropolis and Ulam. Since than and it has been only sporadically mentioned in the literature up to the rediscovery in 1993 by Gordon et. al [10]. A large variety of papers on particle filters has been written since than. A good introduction can be found in [3]. In 1998 Isard introduced this technique into the field of computer vision for tracking tasks [19, 20]. Lately a lot of effort went into improvements of particle filters to overcome certain limitations and problems [4, 13, 21, 7]. However, only little work has been done in the field of using such probabilistic techniques for the investigation and interpretation of optical flow fields: In [22] motion discontinuities are tracked using optical flow and the CONDENSATION algorithm and in 2002 Zelek [29] used a particle filter to predict and therefore speedup a correlation based optical flow algorithm.

Let θ_t denote the unobserved state of the system at the discrete time $t \in \mathbb{N}$ (in our case the system state consists of the position of the point resulting from a projection of an independent motion). Modeling the system as a Markov process results in the conditional transition probability $p(\theta_{t+1}|\theta_0, \theta_1, \ldots, \theta_t) = p(\theta_{t+1}|\theta_t)$. In other words the state of the system in the next time step does only depend on the current state of the system and not on the history (here the transition probability of a point is given by the measured flow vector at this point plus zero mean normal distributed noise). Let further denote y_t the observation of our system at time t. If we assume a likelihood function $p(y_t|\theta_t)$, modeling the observation process (the observation process in this application modeled by the likelihood function is described in paragraph 2), the posteriori distribution is given by Bayes' law

$$p(\theta_t|y_t) = \frac{p(y_t|\theta_t)p(\theta_t)}{\int p(y_t|\theta_t)p(\theta_t)\ d\theta} \tag{1}$$

where the marginalization $p(y_t) = \int p(y_t|\theta_t)p(\theta_t)\ d\theta$ can be seen as a normalization factor.

Since $p(\theta_t)$ is at no time known exactly (it is unobserved), the best estimate for $p(\theta_t)$ can be calculated by using the estimate from the last time step $p(\theta_{t-1}|y_{t-1})$ and the transition probability $p(\theta_t|\theta_{t-1})$ [3] resulting in the Chapman-Kolmogorov prediction equation

$$p(\theta_t|y_{t-1}) = \int p(\theta_t|\theta_{t-1})p(\theta_{t-1}|y_{t-1})\ d\theta \tag{2}$$

Plugging this estimate into Bayes' law (eq. 1) results in the update equation

$$p(\theta_t|y_t) = \frac{p(y_t|\theta_t)p(\theta_t|y_{t-1})}{\int p(y_t|\theta_t)p(\theta_t|y_{t-1})\ d\theta} \tag{3}$$

Generally the prediction and update equations cannot be computed in closed form since they require the evaluation of complex and possibly multidimensional integrals. Applying Monte Carlo techniques leads to the particle filter algorithm (given in fig. 3). In the particle filter algorithm, the distributions are approximated by a set of N particles $\theta^{(i)}$ and their weights $w^{(i)}$. To reduce the degeneracy problem and to allow reinitialization if the object is lost, step 4 in the iteration process is modified so that a fraction of the samples are chosen by sampling from an importance distribution $g(\theta)$ [3]. The weights of these importance samples $w^{(i)}_{\text{imp},t}$ must then be corrected to achieve a consistent representation of the posteriori density [20]

$$w^{(i)}_{\text{imp},t+1} = \frac{p(y_{t+1}|\theta^{(i)}_{t+1})}{g(\theta)} \tag{4}$$

Fig. 4 shows a graphical representation of one iteration step in the particle filter with the modified resampling step. For a thorough discussion of particle filters see [3].

Likelihood: The likelihood function models the observation process. For simplicity the inverse measurement model, giving the probability for a pixel belonging to the static scene, is derived.

- Initialize (t=0):
  Generate N independent identical distributed (iid) samples $\theta^{(i)}_0, i \in \{1, \ldots, N\}$ from the user given initial distribution $p(\theta_0)$.
- Iterate:
  1. Predict $\theta^{(i)}_{t+1}$ by sampling from $p(\theta_{t+1}|\theta^{(i)}_t)$
  2. Evaluate weights[0] $w^{(i)}_{t+1} = p(y_{t+1}|\theta^{(i)}_{t+1})$
  3. Normalize the weights.
  4. Resample N times with replacement from the samples $\theta^{(i)}_{t+1}$ according to the weight $w^{(i)}_{t+1}$.
  5. Set $t = t + 1$ and repeat iteration (goto 1).

Figure 3. Description of the particle filter algorithm [3].

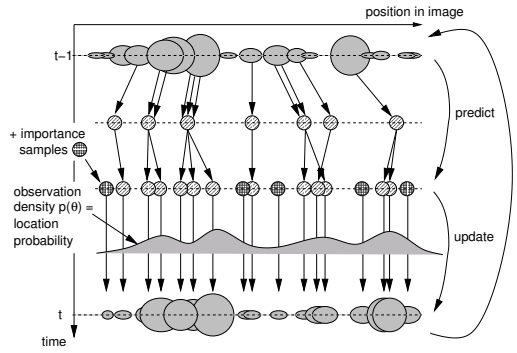


Figure 4. Graphical representation of a particle filter cycle with importance sampling. The state space dimension is given on the horizontal axis. The vertical axis represents the time. The new samples are selected from the samples of the last timestep (top) according to their weight (size of blobs). After applying the motion model, diffusion is added. These two steps represent the prediction step. The importance samples are added and weights are calculated by evaluating the observation density at the sample positions (update). The weights of the importance samples are corrected according to eq. 4 in this step. The resulting samples are used as input to the next time step. (Figure similar to [3]). The "clustering effect" of the particle filter can be observed.

The measurement is assumed to be corrupted by zero mean normal distributed noise of varying variance σ_l^2. The variance depends on the distance of the point to the FOE d_e and on the length of the flow vector d_l:

$$\sigma_l^2 = (\arcsin(\frac{\sigma_e}{d_e}) + \arcsin(\frac{\sigma_p}{d_l}))^2 \tag{5}$$

where σ_e is a user given standard deviation for the position of the FOE and σ_p is a user given standard deviation for the accuracy of the tracking process. Equation 5 can be made plausible by simple geometric considerations (Fig. 5): If the error in the FOE position is σ_e, the maximum induced angular error is given by $\arctan(\frac{\sigma_e}{d_e})$. Similar considerations explain the second therm. The standard deviation σ_l can be interpreted as a "soft threshold" for the angle between the two flow fields. The dependency on the flow length makes sense. For a given error in the determination of the correspondence, the angular error gets smaller with longer flow vectors. Similar considerations apply to the distance from the FOE.

Since we are interested in the probability of a pixel belonging to an independently moving object, the likelihood

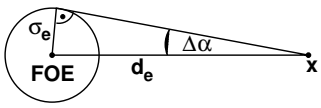


Figure 5. Geometric derivation of the likelihood function: The angular error $\Delta\alpha$ is induced by the error in the FOE position σ_e in dependence of the distance of point x from the FOE d_e.

function is modeled as

$$p(y|\theta) = 1.0 - \frac{1}{\sqrt{2\pi}\sigma_l} \exp(-\frac{\alpha^2}{2\sigma_l^2}) \tag{6}$$

where α is the angle between the two flow fields.

Summary Fast Detection: The usage of a particle filter results in a Bayesian approximation of the posterior probability function for independent motion. The posterior distribution is approximated by a set of particles. Every particle has a weight according to the probability for independent motion at its position. One outcome of the particle filter is, that the particle positions are chosen according to the probability distribution for independent motion. This leads to a "clustering" of particles on moving objects. Optical flow is measured at every particle position and hence many flow measurements are captured on moving objects. See [14] for a detailed description.

3. Refinement

The detection method for independent motion relies on precise knowledge about the camera motion. Car motion can also be measured using image information. In order to derive camera motion parameter from image information, image correspondences for static scene points are needed. Therefore the correspondences used in the refinement are not chosen from the particle filter, but come from a second pool of flow measurements.

Since firstly the car inertial sensors are subject to measurement errors and secondly not every possible motion is captured by sensors (e.g. there is no rotation sensor measuring the tilt axis), the image information is fused with car inertial sensor data to obtain a more precise camera motion.

Essential Matrix: The essential matrix E describes the relationship between two views of a static scene. Any point x in the first view is constrained to lie on the epipolar line l in the second view. The essential matrix E relates x to l via:

$$l = Ex \tag{7}$$

This constraint can also be expressed for an image point correspondence x_1 and x_2 through the so called "fundamental

constraint":

$$x_2^T E_{12} x_1 = 0. \quad (8)$$

Where E_{12} is the essential matrix between the two views 1 and 2.

An important property of all epipolar lines l is, that they intersect in the epipole e itself. The epipole e is the projection of the center of the first camera into the second camera. With pure translational camera movement the epipole coincides with the focus of expansion (FOE) or focus of contraction (FOC), depending on the direction of the camera motion. For a thorough discussion on multiple view geometry see [26].

MAP Essential Matrix Estimation: Extensive studies about essential matrix estimation [26, 16] and the closely related problem of fundamental matrix computation [25, 26, 8, 17] exist. Further on there is a vast variety of literature about ego-motion estimation [27, 12, 1].

Since an approximation to the essential matrix can be inferred from the car inertial sensors, it seems an obvious improvement to use this prior knowledge in the estimation process. The Bayesian maximum a posteriori (MAP) estimation is well adapted to solve these kind of estimation problems. Input to the MAP estimation are a likelihood function $p(y|\theta)$ describing the measurement process and a prior density $p(\theta)$ for the system state $\theta = (e_x, e_y, r_x, r_y, r_z)^T$. The system state θ consists of the epipole position $e = (e_x, e_y)^T$ and the Euler rotation angles $r = (r_x, r_y, r_z)^T$. A multidimensional normal distribution is assumed for the prior $p(\theta)$

$$p(\theta) = \frac{1}{\sqrt{(2\pi)^5 \det(\Sigma)}} \exp(-\frac{1}{2}(\theta - \overline{\theta})^T \Sigma^{-1} (\theta - \overline{\theta})) \quad (9)$$

where the mean $\overline{\theta}$ is calculated using the inertial sensors and the covariance matrix Σ is a user given diagonal matrix with entries $\sigma^2_{e_x}, \sigma^2_{e_y}, \sigma^2_{r_x}, \sigma^2_{r_y}$ and $\sigma^2_{r_z}$ representing the confidence in the sensor data. The likelihood function $p(y|\theta)$ of the measurements y given the system state θ is chosen to be represented by a normal distribution

$$p(y|\theta) = \frac{1}{\sqrt{2\pi\sigma_l^2}} \prod_i \exp(-\frac{d_i^2}{2\sigma_l^2}) \quad (10)$$

where a measurement y_i consists of a point correspondence between x_i and x_i', σ_l describes the second moment of the distribution and d_i is the distance of the point $x_i = (x_{i,x}, x_{i,y}, x_{i,w})^T$ from the epipolar line $l_i = (l_{i,1}, l_{i,2}, l_{i,3})^T$ corresponding to x_i'

$$d_i^2(x_i, x_i', e, r) = \frac{x_i \cdot l_i}{x_{i,w}^2 (l_{i,1}^2 + l_{i,2}^2)} \quad (11)$$

The epipolar line l_i corresponding to x_i' can be calculated by

$$l_i = [e]_x R x_i' \quad (12)$$

where $[e]_x$ denotes a skew symmetric 3 by 3 matrix

$$[e]_x = \begin{pmatrix} 0 & -1 & e_y \\ 1 & 0 & -e_x \\ -e_y & e_x & 0 \end{pmatrix} \quad (13)$$

and R is the rotation matrix constructed from the Euler angles r.

The posteriori distribution is proportional to the product between the prior and the likelihood function

$$p(\theta|y) \propto p(y|\theta) p(\theta). \quad (14)$$

Estimating the maximum of the posterior can be simplified by taking the logarithm of equation 14

$$\ln p(\theta|y) \propto -(\theta - \overline{\theta})^T \Sigma^{-1} (\theta - \overline{\theta}) - \sum_i \frac{d_i^2}{2\sigma_l^2} \quad (15)$$

Obviously equation 15 is less or equal to zero

$$0 \geq -(\theta - \overline{\theta})^T \Sigma^{-1} (\theta - \overline{\theta}) - \sum_i \frac{d_i^2}{2\sigma_l^2} \quad (16)$$

and finding the maximum of equation 15 is identical to finding the minimum of its negative

$$\arg\max_\theta \ln p(\theta|y) = \arg\min_\theta - \ln p(\theta|y) \quad (17)$$

The MAP estimate of the camera pose (epipole and rotation) is hence given by the minimum of the functional $L(e, r)$

$$L(e, r) = (\theta - \overline{\theta})^T \Sigma^{-1} (\theta - \overline{\theta}) + \sum_i \frac{d_i^2}{2\sigma_l^2} \quad (18)$$

The Levenberg-Marquard algorithm is used to find the minimum of this functional. Only the translation direction can be estimated from image point correspondences. The distance cannot be calculated without further knowledge. A parameterization of the epipole (representing the translation direction) in polar coordinates is chosen. Since only the translation direction can be estimated, the radius is dropped from the polar coordinate representation.

Preemptive MSAC: A single wrong image correspondence (f.e. resulting from a measurement error) can lead to a significantly wrong result of the MAP estimation. Hence a robust estimation method is needed. The M-estimator SAmple Consensus (MSAC) [24] is used to deal with this problem. The basic idea behind the MSAC algorithm is described in figure 6. In difference to the MSAC algorithm, the preemptive MSAC chooses the samples according to the known posterior probability distribution from the last time step. This generally results in a faster solution. For a thorough investigation of the advantages of the preemptive MSAC refer to [23].

4. Decision

In the following section the algorithm deriving a warning decision from the sparse probability measurements is given in detail.

Spatio-Temporal Filtering: At each sample position the probability for an independently moving object is calculated (the sample weight). These probabilities are inserted in an otherwise empty image and Gauss filtered with a user selected window size. The low pass filtering can be regarded as an interpolation step to get a denser estimate of the probability. Because short interruptions (f.e. by the windscreen wiper) may occur, the resulting sequence is temporally filtered. The probability at a certain pixel position is thereby regarded as a one dimensional signal, which is passed through a digital low pass filter. This is done for every pixel position.

Clustering: In order to get from a pixel wise probability representation to few moving objects, a clustering algorithm is used. Thresholding the probability image leads to a binary image. After running a dilation algorithm for connecting regions close to each other, a labeling algorithm is run. The labeling algorithm identifies and labels connected regions by processing the image pixel per pixel and looking at the neighbors of the current pixel. The bounding boxes are extracted for each of the regions. The relevant characterization is extracted for every region, namely position, size, number of correspondences in it, mean probability and weighted mean motion. Position, size and number of correspondences N in every region can be extracted straight forward from the bounding boxes. The mean probability and the mean motion are calculated using only the correspondences in the region. Let $f_i = x_{i,t} - x_{i,t-1}$ denote the motion given by a point correspondence between $x_{i,t-1}$ and $x_{i,t}$ and s_i denote the corresponding entry in the probability image from time t at position $x_{i,t}$. The weighted mean motion $\overline{f}$ is calculated by

$$\overline{f} = \frac{\sum_{i=1}^{N} s_i f_i}{\sum_{i=1}^{N} s_i} \tag{19}$$

and the mean probability $\overline{q}$ is calculated by

$$\overline{q} = \frac{\sum_{i=1}^{N} s_i}{N} \tag{20}$$

- Iterate maximal n times:
  1. Select a minimum number (5) of correspondences randomly. A correspondence at position x is chosen with probability $1.0 - p(x)$, where $p(x)$ indicates the probability that at position x has been a moving objects at the last time step. The spatio-temporal filtered posterior distribution (section 4) is used as $p(x)$.
  2. Calculate the MAP estimate from these correspondences.
  3. Calculate the error δ_i for every correspondence.
  4. Calculate the loss function $L = \sum_i \rho(\delta_i^2)$ with $\rho(\delta^2) = \begin{cases} \delta^2 & \text{if } \delta^2 < T^2 \\ T^2 & \text{otherwise} \end{cases}$
  5. If L is bigger than a threshold repeat iteration (goto 1), else choose this solution and exit.
- Select the solution with the minimal loss function L

Figure 6. Description of the MSAC algorithm [24].

Kalman Tracking: The idea behind this algorithm is that a separate Kalman filter is assigned to every moving object. The 2D position and velocity of the moving object in the image is used as system state in the Kalman filters. The list of regions from the clustering algorithm is used as measurements. Since multiple measurements and Kalman filters might be involved, each region must be assigned to a specific Kalman filter instance from the list of all Kalman filters. This list is maintained in the following manner in every time step:

1. The prediction step is performed for every Kalman filter in the list.
2. For every Kalman filter the closest region to the current position of the Kalman filter is searched. If this region is closer than either three times the Kalman filter variance or closer than 7 pixel, it is used as measurement for the update step. If the region is further away, no update step is performed.
3. Kalman filters which have not been updated in the first three subsequent time steps of their lifetime are removed. Kalman filters which have not been updated for the last 3 cycles are also deleted.
4. A new Kalman filter is added to the list for every region which has not been assigned to an already existing Kalman filter.
5. When two Kalman filter adopt positions closer than 7 pixels to each other, the younger Kalman filter is deleted.

The details of the Kalman filter implementation are presented in the subsequent section.

Kalman Filter Implementation The Kalman filter [18] is a set of mathematical equations that provides an efficient solution to the discrete-data linear filtering problem. For the algorithm presented here, we follow the notation convention of [15] where the basic filter equations can also be found.

System Description: Since we only have image coordinates (no 3D information) for our independently moving clusters, we estimate the cluster centers in image coordinates. Approximating the system by a simple point mass model (as often used in physics) the x and y-motion are independent, yielding two independent Kalman filters per cluster. The main advantage of two independent Kalman filters per cluster is the computational speed. We estimate the following states: Notation:

$$\vec{x}_x = \begin{bmatrix} x & v_x & a_x \end{bmatrix}^T, \qquad \vec{x}_y = \begin{bmatrix} y & v_y & a_y \end{bmatrix}^T \tag{21}$$

where x, v_x, and a_x are position, velocity, and acceleration of the cluster (analogous for y).

Measurements are the cluster positions and the average optical flow of the contributing flow vectors as cluster velocity.

$$\vec{z}_x = \begin{bmatrix} x_{\text{cluster}} \\ v_{x,\text{cluster}} \end{bmatrix}, \qquad \vec{z}_y = \begin{bmatrix} y_{\text{cluster}} \\ v_{y,\text{cluster}} \end{bmatrix} \tag{22}$$

For our simple filter, the control vector describing external system input is 0.

Process description: The connection between measurements and system variables is straight forward:

$$\dot{x} = v_x, \qquad \dot{v_x} = a_x, \qquad \dot{a_x} = 0, \tag{23}$$

The same equations apply for the y direction.

Measurement Description: The measurement update step incorporates the new measurements. x_{cluster} and $v_{x,\text{cluster}}$ as measurements can be easily expressed in terms of state variables:

$$h(\vec{x})_x = \begin{bmatrix} x_{\text{cluster}} \\ v_{x,\text{cluster}} \end{bmatrix} = \begin{bmatrix} x \\ v_x \end{bmatrix}. \tag{24}$$

Again, the same equations apply for the y direction. These equations and their derivatives w.r.t. the state vector are the input to the Kalman filter. Time update is performed at each time step. Measurement update steps are performed whenever new measurements are available.

Measurement variances are estimated to be 7 pixels for the cluster position due to the uncertainties in detecting different parts of the object in different frames. The uncertainty of the cluster velocity is robustly obtained from optical flow measurements with an conservatively estimated uncertainty of 2 pixels. The system uncertainties are estimated with 2 pixels, and the initial covariances are conservatively set to 7 pixels and 7 pixels/s, respectively. The acceleration state is not considered at this point, but could be estimated if needed.

5. Experiments

Figs. 7, 8 and 9 show results on a real world image sequences acquired with UTA.

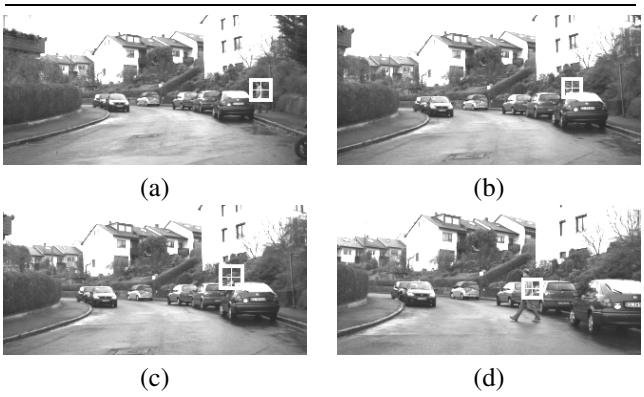


Figure 7. Some images from a pedestrian crossing sequence. The pedestrian is detected early and tracked, even though it is temporarily partially occluded by a parked car. A warning is issued in image (c).

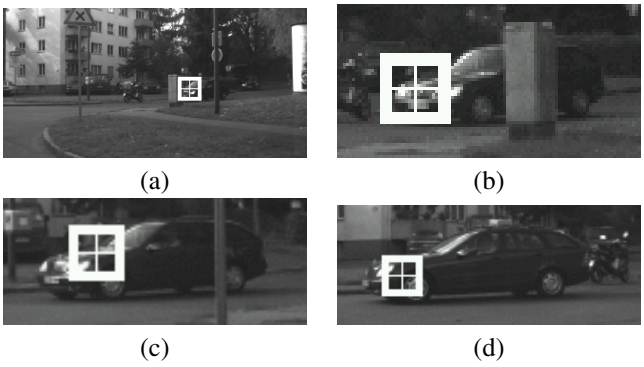
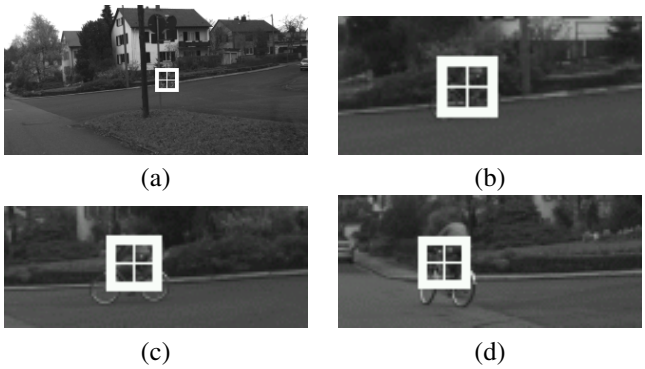


Figure 8. Some images from an intersection sequence. The car is detected early and a warning is issued in image (b). Images (b) to (d) are magnified sections of the original images.

Timing: The mean computation time is 80 ± 13 ms for the real world sequence with the intersecting car (Fig. 8). These timings were measured on a standard 3.0 GHz Pentium IV PC with an overall number of 500 samples. The optical flow used a pyramid of size 2.

6. Conclusions

A fast and robust Bayesian system for the detection of independently moving objects by a moving observer has been presented. The usability of the system has been demonstrated by a collision warning application. The main improvements with respect to earlier versions of this algorithm



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

Figure 9. Some images from a cyclist intersection sequence. The cyclist is detected early and the driver is warned in image (b). Images (b) to (d) are magnified sections of the original images.

are:

- Fusion of car inertial sensor information with image information by usage of MAP estimation for camera pose.
- Incorporation of a final Kalman filter based decision module.
- Improvement of the likelihood function.

Acknowledgments: This work was supported by BMBF Grant No. 1959156C.

References

[1] H. Badino. A robust approach for ego-motion estimation using a mobile stereo platform. In *Proc. IWCM*, 2004.

[2] T. K. B.D. Lucas. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981.

[3] A. D. et al. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[4] C. H. et al. Tracking multiple objects with particle filtering. *Trans. AES*, 38(3):791–812, 2002.

[5] F. W. et al. A monocular image based intersection assistant. In *Proc. IV*, 2004.

[6] J. B. et al. Performance of optical flow techniques. Technical report, Univ. Western Ontario, London, Ontario, N6A 5B7, 1994.

[7] J. V. et al. Maintaining multi-modality through mixture tracking. In *ICCV*, 2003.

[8] K. K. et al. Fundamental matrix from optical flow: Optimal computation and reliability evaluation. *JEI*, 2000.

[9] M. P. et. al. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. *IJCV*, 32(1):7–25, 1999.

[10] N. G. et al. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE-Proceedings-F*, pages 107–113, 1993.

[11] S. G. et. al. System architecture for an intersection assistant fusing image, map and gps information. In *IV*, 2003.

[12] T. T. et al. Comparison of robust approaches to ego-motion computation. In *CVPR*, 1996.

[13] Z. et al. An mcmc-based particle filter for tracking multiple interacting targets. In *ECCV*, 2004.

[14] R. K. F. Woelk. Fast monocular bayesian detection of independently moving objects by a moving observer. In *Proc. DAGM*, 2004.

[15] G. B. G. Welch. An introduction to the kalman filter. Technical report, Univ. of N. Carolina Chapel Hill, 1995.

[16] R. Hartley. Computation of the essential matrix from 6 points. http://rsise.anu.edu.au/ hartley/Papers/6pt/6ptnew.pdf.

[17] R. Hartley. In defence of the eight-point algorithm. *TPAMI*, 1997.

[18] R. Kalman. A new approach to linear filtering and prediction problems. *Transactions ASME J. of Basic Engineering, Series D*, 82:35–45, March 1960.

[19] A. B. M. Isard. Condensation – conditional density propagation for visual tracking. *IJCV*, 29(1):5–28, 1998.

[20] A. B. M. Isard. Icondensation: Unifying low-level and high-level tracking in a stochastic framework. In *ECCV*, volume 1, pages 893–908, 1998.

[21] J. M. M. Isard. Bramble: A bayesian multiple-blob tracker. In *ICCV*, 2001.

[22] D. F. M.J. Black. Probabilistic detection and tracking of motion discontinuities. In *ICCV*, 1999.

[23] D. Nister. Preemptive ransac for live structure and motion estimation. In *Proc. ICCV*, pages 199–206, 2003.

[24] A. Z. P. Torr. MLESAC: A new robust estimator with application to estimating image geometry. *CVIU*, 1996.

[25] D. M. P. Torr. Development and comparison of robust methods for estimating the fundamental matrix. *IJCV*, 1996.

[26] A. Z. R. Hartley. *Multiple View Geometry in Computer Vision*. Cambridge university press, 2000.

[27] T. K. T. Suzuki. Measurement of vehicle motion and orientation using optical flow. In *ITS*, 1999.

[28] P. Tippler. *Physik*. Spektrum, 1994.

[29] J. Zelek. Bayesian real-time optical flow. In *Proc. VI*, 2002.