Accuracy of Trajectories Estimation in a Driver-Assistance Context

Waqar Khan and Reinhard Klette

Computer Science Department, Tamaki Innovation Campus,
The University of Auckland, New Zealand
wkha011@aucklanduni.ac.nz

Abstract. Feature-point tracking for the purpose of object tracking in a driver-assistance context is not an easy task. First, to track rigid objects, feature points have to be matched frame-by-frame and then, by using disparity maps, their real-world position can be derived, from which the object velocity is estimated.

Unfortunately, a feature-point matcher cannot find (reliable) matches in all frames. In fact, the performance of a matcher varies with the type of feature-point detector and descriptor used. Our comparison of different feature-point matchers gives a general impression of how descriptor performance degrades as a rigid object approaches the ego-vehicle in a collision-scenario video sequence. To handle the mismatches, we use a Kalman-filter-based tracker for each tracked feature point. The tracker with the maximum number of matches and with a most recent match is chosen as the *optimal tracker*. The role of the optimal tracker is to assist in updating the tracker of a feature point which had no match. The optimal tracker is also used in estimating the object velocity.

To understand the behaviour of the safety system, we used the DoG detector in combination with SURF, BRIEF, and FREAK descriptors, while linBP and iSGM are used as stereo matchers. The novelty in our work is the performance evaluation of a stereo-based collision avoidance system (avoidance by brake warning) in a real collision scenario.

1 Introduction

To estimate an opposing vehicle or object trajectory, which may be a hazard for the *ego-vehicle* (i.e. the vehicle the vision system is operating in), the object has to be tracked as it moves around in a scene. Like stereo matching between reference and match cameras, tracking also involves searching for correspondences between images captured over time on the same camera. Generally, object tracking is considered to be a challenging task. Difficulties in tracking arise due to the following factors: object motion, camera motion, change in object pose, changes in the scene, non-rigid objects, and object occlusions.

The approach to tracking may vary based on the application. Few assumptions are used based on the application of tracking. For example, we assume that the object is rigid. We will also assume that tracked *feature points* (FPs) on object

F. Huang and A. Sugimoto (Eds.): PSIVT 2013 Workshops, LNCS 8334, pp. 47–58, 2014.

surfaces are always binocularly visible to stereo cameras. The object is assumed to be pre-detected in the left camera image, with a bounding box around the object.

Before tracking the object, it has to be represented first. If multiple FPs are used, then it becomes a challenge to distinguish which set of FPs belong to the same object over the course of time. Often, for a rigid object, a common *motion constraint* is applied where neighbouring features that are seen moving together are grouped to represent the same object [6].

We recall that a tracker is a module of a collision-avoidance system. With the object assumed to be pre-detected, FPs are first identified on the detected object at *observation time* $k = 0$. To track an object over time, these FPs are tracked in the following observed stereo frames over time. In order to match the FPs, each *feature point* (FP) is represented by a *descriptor*. A descriptor can describe salient features also known as *attributes*. So, at $k = 0$ the initially identified FPs are represented by their descriptors.

For discussing a tracking situation, consider the subsequent frame at $k = 1$. The object's position may change, leading to a change in positions of previously identified FPs. For an object detected in the first frame, the FPs detected over its region are the *query FPs*, while the FPs detected in the following frame are the *train FPs*. [1]

In a stereo-vision system, we have two cameras. The left camera provides *reference images*, while the right camera gives the *match images*. We assume that images are geometrically rectified. We also assume that the detection of FPs is performed only on reference image $I_L(u, v)$. Hence, matching between detected FPs over subsequent observations leads to 2-dimensional $I_L(u, v)$ image space tracking, where a matched train FP represents the updated position of a query FP.

The real-world position of each query or train FP is computed through *disparity* d at $I_L(u, v)$. The disparity is computed from reference to match images.

However, mismatches in FP matching is the limitation. Thus, an additional step of outlier-removal is necessary to select only the correct matches. Due to this, at each following frame, not all the query FPs have a correct match. Hence, to track each query FP in real-world 3D space, a *Kalman filter* (KF) [8] is used, which in the absence of a correct match can predict the real-world position instead of the corresponding FP.

By using the KF, this tracker estimates the future position and velocity of each FP. The estimated velocity is later used to determine whether the ego-vehicle is on a collision course. If it is then the system issues a braking warning to the driver, who can later apply brakes to avoid the collision. Figure 1 shows our experimental set-up for the evaluation of a driver assistance system.

The objective of this study is to practically test the findings from purely theoretical models described in [9,10] for a collision scenario. In the collision experiment, we kept a safe braking distance from the opposing vehicle. The

[1] The naming convention is consistent with the one used in *Open Source Computer Vision* (OpenCV) library for matching the FPs.

marked position is used by the driver of the ego-vehicle to apply brakes after the ego-vehicle crosses it to safely avoid collision (without a warning system). Similarly, the safety system must recognize the collision scenario and issue a warning before the ego-vehicle crosses this mark on the road, hence issuing a timely warning. To validate this, an observer is used, who raises a flag after the ego-vehicle has crossed the marker. This observer is also visible to reference and match cameras in the ego-vehicle.

So, if the safety system can issue a warning (based on estimated trajectories) before the observer raises the flag, then the system is proven to be issuing a timely warning for that scenario. Figure 1 illustrates the choreography.

2 Feature-Point Detector, Descriptor, and Matcher

Given the reference rectified image as input, a FP detector looks for regions of interest within the image. The FP orientation is computed with respect to the direction of strong image gradient in a region. So, even if the image is rotated, due to the strong image gradient, the orientation of detected FP can be changed. Furthermore, due to rotated orientation of the detected FP, the descriptor can be computed, independent of the image rotation. Such FP based algorithms are called *rotation invariant*.

Furthermore, if the FP based algorithm computes only fixed sized features instead of independently computing the optimal size for every FP then that algorithm can detect same features, even if the image is scaled. Such algorithms are called *scale invariant*.

2.1 Difference of Gaussian Detector

We briefly recall the detector part of the *Scale Invariant Feature Transform* (SIFT) [11]. To detect the scale invariant FPs, a scale space is constructed by convolving the image with Gaussian filters at various scales. Then, the *Difference*

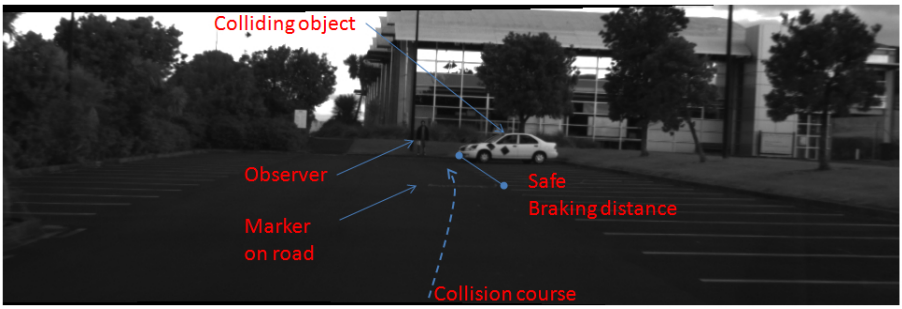

Fig. 1. Choreographed sequence. The *observer* is a person holding a flag. The flag is raised after the ego-vehicle, which is on a *collision course*, crosses the *marker on the road*. The marker on the road is at a safe *braking distance* from the *colliding object*.

of Gaussian (DoG) images are computed from the scaled images. Candidate FPs are chosen based on minima and maxima of DoG images at various scales. Candidate FPs location are further refined by interpolating neighbouring image intensities. Candidate FPs that are with low contrast or are at the edge are excluded, while the remaining form the set of detected scale invariant FPs.

2.2 Feature Point Descriptors

Speeded Up Robust Features Descriptor. The SURF descriptor describes the intensity distribution in the neighbourhood of the detected FP. SURF descriptor uses integral images along with Haar wavelets. To be rotation invariant, the gradient values in u and v direction are computed from Haar wavelets in a circular neighbourhood. The radius of this neighbourhood is derived from the scale at which the FP was detected [3].

Binary Robust Independent Elementary Feature Descriptor. Calonder et al. proposed *Binary Robust Independent Elementary Feature* (BRIEF) descriptor [5]. To reduce sensitivity to noise a Gaussian smoothing is applied by a 9pixels $\times$ 9pixels averaging filter centred on the FP. Later, using the Gaussian distribution around the detected FP, random pixels are chosen for comparison. The proposed bitwise descriptor vector is obtained by comparing the intensity of 512 pairs of pixels in a 48pixels $\times$ 48pixels region.

One of the advantages of a binary descriptor is that the matching of descriptors is efficient. However, the disadvantage is that the descriptor is neither invariant to orientation nor to scale changes.

Fast Retina Keypoint Descriptor. Alahi et al. originally proposed *Fast Retina Keypoint* (FREAK) descriptor that is an extension of BRIEF descriptor [2]. Inspired by human eye retinal pattern, the sampling of chosen points follows a specific pattern with more chosen points closer to the detected FP and as the distance increases the number of chosen points reduce exponentially. Due to the specific sampling pattern approach the feature descriptor allows for the 'coarse to fine' approach. The descriptor is a binary vector consisting of sum of estimated local gradients over selected point pairs.

2.3 Feature Point Matching and Outlier Removal

Correspondence between FPs in different frames can be done by matching FP descriptors. For doing so, we used an exhaustive *brute-force* (BF) matching approach. To find a correspondence to a query FP descriptor, all train FP descriptors are tested and only f nearest neighbours are chosen as matching descriptors.

FP matching often results in mismatching. Most common type of mismatch occurs when FPs correspondence is incorrect. To remove such outliers we followed a series of steps. Firstly, we used $f = 2$ to find two nearest descriptors for each query FP. We used the known approach of ratio test to remove initial outliers. After computing these distances of two nearest neighbours from the query FP,

we computed the ratio of the distances. The nearest point is chosen, only if the ratio is greater than 1.5. The ratio test removed most of the outliers. To remove the remaining outliers, we applied the RANSAC approach [7,4].

3 Tracker

A common limitation, that the FP tracker has to overcome, is that the matcher does not confirm that the matched points in one frame would also be matched in the following frames. Furthermore, mismatches in positions can also occur, if the stereo correspondence algorithm fails to determine the correct disparity for the matching train FP.

Hence, each FP has a KF tracker affiliated with it. So that, when there is a match for a FP, then its tracker has to be updated with the new observation (real-world position). And, when there is no match found, then the tracker has to predict FP's real-world position. Due to the rigid object assumption, all trackers should portray the similar real-world velocity estimate. So, during the prediction phase the neighbouring FP trackers can assist as well.

3.1 Feature Point Tracking by Kalman Filter

A KF is usually defined in three steps (see Algorithm 1 for the complete object tracking algorithm, and Table 1 for the detailed list system inputs).

Initialization of KF is the first step. Given the detected object in the first frame on reference image of size $w \times h$ pixels with $(u, v) = (0, 0)$ at the top-left corner of the image. Let n be the number of query FPs detected in this frame. Then, for $j = \{1, 2, \ldots, n\}$, initialize $\mathrm{KF_j}$ with the real-world location $\overrightarrow{\widehat{O}_j^r}$ of query FP j at $I_L(u, v, d)$:

$$\overrightarrow{\widehat{O}_j^r} = \begin{bmatrix} \widehat{X}_j^r \\ \widehat{Y}_j^r \\ \widehat{Z}_j^r \end{bmatrix} = \frac{b}{d} \begin{bmatrix} u - w/2 \\ -(v - h/2) \\ f/\tau \end{bmatrix} - \begin{bmatrix} \frac{b}{2} \\ 0 \\ 0 \end{bmatrix} \tag{1}$$

where d denotes the disparity, b is the baseline length, f is the focal length, and τ is the pixel size. Superscript r denotes that the measurements are in the ego-vehicle frame of reference.

Algorithm 2 describes the initialization KF for each query FP.

Prediction by KF is the second step. In the following frame, FPs are matched and outliers were removed. Each $\mathrm{KF_j}$ was asked to predict the new position $\overrightarrow{wO_j^r}$, which after first observations was still constant $\overrightarrow{\widehat{O}_j^r}$, as there was only one observation. However, with more observations, it was based on the $\mathrm{KF_j}$ estimated velocity $\overrightarrow{wV_j^r}$.

On real-world data, it is not possible for the matcher to guarantee the match of every query FP throughout the video sequence. Therefore, we assigned *weight*

w_j to a FP j as a counter of its matches, i.e. every time there is a match, w_j was incremented.

In a case, when the matcher failed to find the match, then *optimal tracker* m was chosen as the one with maximum w_j. In a case, when two or more FPs had the same w_j count, then priority was given to the point with the most recent successful match. Hence, the optimal Tracker m would have maximum $\frac{w_j}{(k-k_j)\delta s}$, where k_j denoted the observation number for FP j with recent successful match.

Previously in our models [9,10], we assumed that the ego-motion is known, while object trajectory was estimated. However, in our experiment, the object is already static, while the ego-vehicle is moving. So, instead the system used

Algorithm 1. FP tracker

collisionDecisionSystem$(f, b, d_{max}, \tau, \delta s, w, h, t_d, \overrightarrow{V^i_{crit}}, \mu, g, r_{exc})$**returns** state S
for each Observation k in $\{0, 1, \ldots\}$ **do**
  if $k == 0$ **then**
    Input detected object;
    Detect n query FPs in this frame;
    for each query FP j in $\{1, 2, \ldots, n\}$ **do**
      Initialize $(\mathrm{KF_j}, \overrightarrow{\widehat{O}^r_j}, \mathrm{w_j}, \mathrm{k_j}) = $ **initializeTracker**$(\mathrm{I_L(u, v, d)})$ using Algorithm 2
    end for
    System state $S \leftarrow$ **S0**;
  else
    for each query FP j in $\{1, 2, \ldots, n\}$ **do**
      $\mathrm{KF_j}$ prediction of position $\overrightarrow{wO^r_j}$ and velocity $\overrightarrow{wV^r_j}$;
    end for
    m = FIND optimal tracker j with maximum confidence criteria: $w_j / ((k - k_j)\delta s)$;
    l = FIND j with smallest predicted distance: $||\overrightarrow{wO^r_j}||$;
    $S \leftarrow$ **canWait2**$(\overrightarrow{wO^r_l}, \overrightarrow{wV^r_m}, t_d, \overrightarrow{V^i_{crit}}, \delta s, r_{exc}, \mu, g)$ using Algorithm 3;
    if $S = S3$ **then**
      Possible collision: issue precautionary warning as not safe to make further observations;
      return
    end if
    if $S = S4$ **then**
      Definite collision: issue necessary warning;
      return
    end if
    for each query FP j in $\{1, 2, \ldots, n\}$ **do**
      $(\mathrm{KF_j}, \overrightarrow{\widehat{O}^r_j}, \mathrm{w_j}, \mathrm{k_j}) = $ **updateTracker**$(k, w_j, k_j, \mathrm{KF_j}, \overrightarrow{\widehat{O}^r_j}, \overrightarrow{wV^r_m})$ using Algorithm 4;
    end for
  end if
end for

Algorithm 2. Initialize FP tracker

initializeTracker($I_L(u, v, d)$) **returns** ($\mathrm{KF_j}, \overrightarrow{\widehat{O}_j^r}, \mathrm{w_j}, \mathrm{k_j}$)
For each FP j, compute $\overrightarrow{\widehat{O}_j^r}$ using Equation 1;
Initialize $\mathrm{KF_j}$ with $\overrightarrow{\widehat{O}_j^r}$;
Let w_j be the observation frequency and k_j be the last frame with successful match for j. Initialize $w_j = 0$ and $k_j = k$;

Algorithm 3. Determine if braking warning is due

canWait2($\overrightarrow{wO_l^r}, \overrightarrow{wV_m^r}, t_d, \overrightarrow{V_{crit}^i}, \delta s, r_{exc}, \mu, g$) **returns** state
Object distance: $D_{cc} = ||\overrightarrow{wO_l^r}||$;
if $D_{cc} \leq r_{exc}$ **then**
  Definite collision;
  return S4;
end if
Object worst case position: $\overrightarrow{cO^r} = \overrightarrow{wO_l^r}$;
Estimated vehicle velocity: $\overrightarrow{V^i} = -\overrightarrow{wV_m^r}$;
Vehicle braking displacement after t_d: $\overrightarrow{D_b} = \overrightarrow{V_i}(t_d + \delta s) + \left((\overrightarrow{V^i})^2 - (\overrightarrow{V_{crit}^i})^2\right) / (2\mu g)$;
Maximum vehicle displacement to reach $\overrightarrow{V_{crit}^i}$: $\overrightarrow{O_{safe}^i} = \overrightarrow{V^i} \cdot \delta s + \overrightarrow{D_b} + [r_{exc}, 0, r_{exc}]^T$;
if $||\overrightarrow{cO^r}|| \leq ||\overrightarrow{O_{safe}^i}||$ **then**
  Might collide, and not safe to consider additional observation;
  return S3
else
  Might collide, but safe to consider additional observation;
  return S2
end if

the measurements from the optimal tracker $\mathrm{KF_m}$ to determine the ego-vehicle velocity $\overrightarrow{wV_m^r}$.

Warning decision is the intermediate step which does not affect the KF, however does affect the output of the system. The system uses $\overrightarrow{wV_m^r}$ to compute the safe *braking displacement* $\overrightarrow{D_b}$ in XZ directions with $\overrightarrow{V_{crit}^i} = [0, 0, 0]^T ms^{-1}$(see Algorithm 3).

Then, the system computes the FP l at the nearest predicted position $\overrightarrow{wO_l^r}$. The system uses Algorithm 3 to determine whether it has to issue a necessary warning at state **S4**, or a precautionary warning at state **S3**, or it can safely wait for an additional observation at state **S2**.

Update of KF is the final step. Each $\mathrm{KF_j}$ is updated based on the new matched real-world position. In a case, when the matcher fails in finding the correct match or the disparity is zero, then the optimal tracker m is used on the last observation $\overrightarrow{\widehat{O}_j^r}$ to find the new predicted position. $\mathrm{KF_j}$ is updated using this predicted position (see Algorithm 4).

Algorithm 4. Update FP tracker

updateTracker$(k, w_j, k_j, \mathrm{KF_j}, \overrightarrow{\widehat{O}_j^r}, \overrightarrow{wV_m^r})$ **returns** $(\mathrm{KF_j}, \overrightarrow{\widehat{O}_j^r}, \mathrm{w_j}, \mathrm{k_j})$
if match found in train FPs after outlier removal **then**
  $w_j = w_j + 1$ and $k_j = k$;
  Use matched train FP to compute $\overrightarrow{\widehat{O}_j^r}$ using Eq. 1 and update $\mathrm{KF_j}$;
else
  Update $\mathrm{KF_j}$ with $\overrightarrow{\widehat{O}_j^r} + \overrightarrow{wV_m^r} \cdot (k - k_j)\delta s$;
end if

4 System Parameters, Results, and Discussion

We use a collision experiment to evaluate the driver assistance system. Table 1, summarizes the parameters used in this experiment. We use iterative semiglobal matching (iSGM) stereo and linear belief propagation (linBP) stereo for this sequence for our evaluations [10]. In each experiment, the detector is common, so the detected points would also be common, however the observed points may vary depending on the matching descriptors and disparities.

Table 1. System parameters

Symbol	Description	Typical value
f	Focal length	$8.9mm$
τ	Pixel size	$5.01\mu m$
$w \times h$	Sensor pixel resolution	960×320 pixels
b	Baseline length	$395.8mm$
d_{max}	Maximum disparity	60
ϕ	Vergence angle	0°
δs	Sampling interval	$0.04\ s$
r_{exc}	Radius of vehicle exclusion zone	$3.6m$
V^i	Maximum Ego-vehicle speed	$11.1\ ms^{-1}(40kmh)$
$\overrightarrow{V_{crit}^i}$	Maximum safe collision velocity	$\overrightarrow{0}ms^{-1}(0kmh)$
V_{limit}	Maximum speed limit	$17ms^{-1}(60kmh)$
s	Speeding factor	1.5
t_d	Driver response time	$1.5\ s$ [1]
μ	Coefficient of friction	0.45
g	Gravitational constant	$9.8\ ms^{-2}$
n	Number of FPs	Detector dependant
t_b	Vehicle braking time	To be estimated
$\overrightarrow{D_b}$	Maximum safe braking distance	To be estimated

Although the system is tracking all query FPs on object's surface throughout the sequence, however, it will be much easier to explain the track of a single FP. Therefore, we choose the *nearest observed query FP N* in the first frame, and visually track it throughout the sequence. The representation of tracked position

is in the ego-vehicle frame of reference. Note that, even for a single track, the system will be tracking all query FPs at the back-end. And, in case there is a mismatch for point N then one of these neighbouring trackers will become an optimal tracker for N at that instance.

The tracking starts at $k = 50$ where the object is first detected and continues until either a warning is issued by the system or the vehicle has the crossed the safe brake distance marker at $k = 110$. So, basically, the system has to issue a braking warning within 60 observations to be deemed as timely.

The output is in the form of a track plot in the XZ grid. Both dimensions are measured in metres with reference to the ego-vehicle. The exclusion zone for the ego-vehicle is represented by a red circle of radius r_{exc}. In the plots, the r_{exc} is scaled based on X-axis only.

The observed or predicted position of point N is represented by a marker. Which in general is a combination of a circle and a horizontal line. The centre of circle presents the marked position of N in real-world from the ego-vehicle.

A green marker on the track, highlights an observation after correct FP match and with disparity greater than 0 at that instance.

A blue marker on the track highlights, that either the matcher did not find a match at that instance or the disparity of the corresponding train FP was zero. Thus, optimal tracker prediction is used to mark at this position.

A red marker on the track highlights, that either the matcher did not find a match at that instance or the disparity was zero. Also, the considered number of observations for all trackers are less than 7 (thus optimal tracker may have error in its estimations due to unsettled KF).

Due to integral disparities, the measured location might not change, leading to a constant predicted position, hence multiple observation markers can be drawn on top of each other.

If the system issues a braking warning before the safe braking distance i.e. 60 observations, then, the system labels the considered number of observations next to the last marker in the plot. The track plots have iSGM based output on the left while linBP based output on the right.

DoG Detector and SURF Descriptor. Figure. 2 illustrates that even with the similarity in matched FPs, there was a difference in types of markers and even position of markers. Firstly, the position of markers was different because the disparities computed for the matched points were different for iSGM and linBP. Secondly, the presence of red markers only in iSGM's plot suggested that, iSGM had generated zero disparities for the correct matches of N while for the same points linBP had green markers, hence correct disparities. There was no difference in the timing of the warning, as both iSGM and linBP based systems issue timely warnings after 52 observations. Even though there were zero disparities for iSGM, in the course of tracking, still the KF could accommodate them. This showed that iSGM with a KF tracker is a robust option.

DoG Detector and FREAK Descriptor. Figure 3 illustrates that while the ego-vehicle approached the object, the green markers in relative frame of reference are more evenly distributed for iSGM than for linBP. This highlights

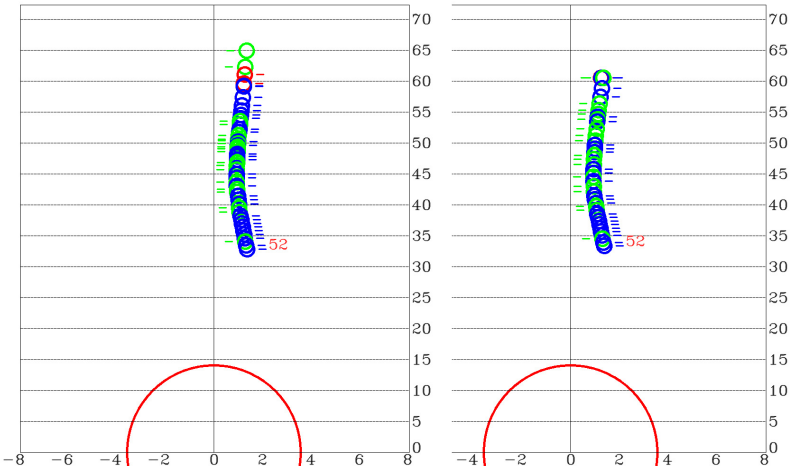

Fig. 2. DoG detector and SURF descriptor. Left: iSGM, Right: linBP.

that initially while N had correct FP matches, iSGM also had better sub-pixel disparities than that of integral linBP disparities.

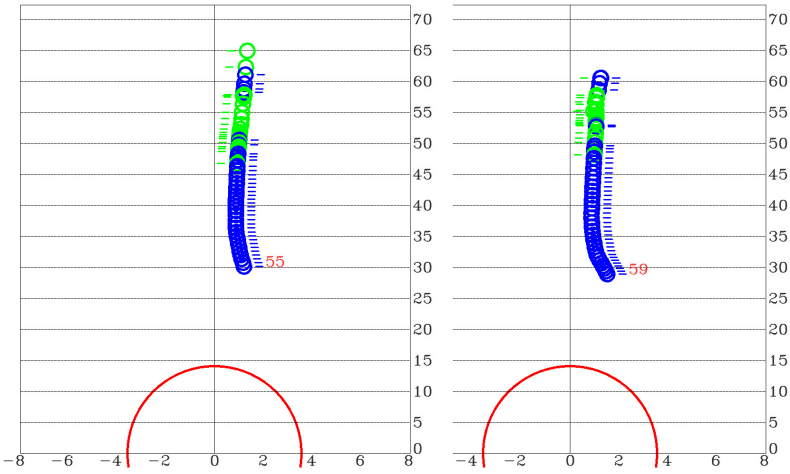

Fig. 3. DoG detector and FREAK descriptor. Left: iSGM, Right: linBP.

FREAK descriptor failed to identify any matches for point N after marked Z-distance $Z < 45m$ from ego-vehicle. Still, in both system (see Fig. 3), the tracked path of N rightly changed. This change in predicted trajectory was also consistent with real ego-vehicle trajectory (see Fig. 1).

Thus, in the absence of matches, the other FPs had correct matches and disparities, leading to correct feedback from the optimal tracker to the point N.

This also showed that, while the object trajectory changed, it was important to have matches either *ideally* for an optimal tracker or for any FP.

Due to the correct disparities of iSGM, its system classified the collision scenario earlier after 55 observations, while linBP based system took longer (59 observations), but both were timely warnings.

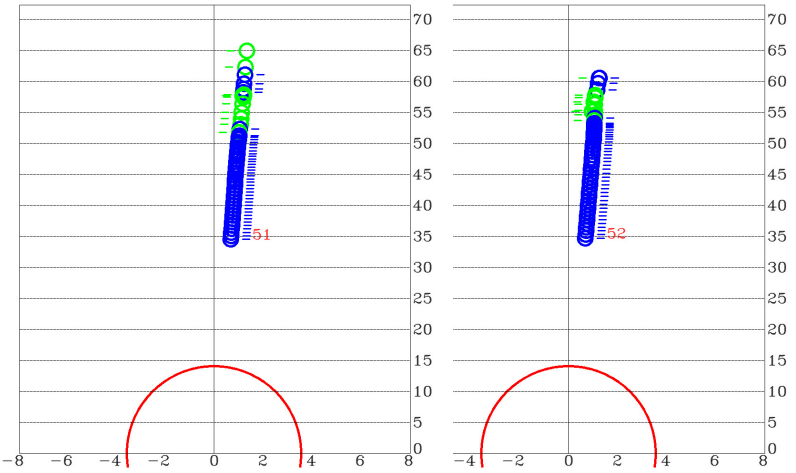

Fig. 4. DoG detector and BRIEF descriptor. Left: iSGM, Right: linBP.

DoG Detector and BRIEF Descriptor. Figure. 4 shows the system's performance for linBP and iSGM using BRIEF descriptor. It became clear that the BRIEF descriptor was able to match in the initial stages. However, as the scale of object increased while the object came closer to the ego-vehicle. Then, there were no FP matches, neither for the point N, nor for any optimal tracker. Hence, the track estimated by the systems did not include the change in trajectory of ego-vehicle as it approached the safe braking distance.

The system was still able to issue timely warnings, but this would be a less likely case if the object was approaching the ego-vehicle from X-distance $> r_{exc}$.

5 Conclusions

The number of feature points matched in each frame, cannot be used to identify the best feature point matcher for the purpose of feature point based object tracking. To have a good estimate, the same feature point has to be correctly matched frame-by-frame. Most matchers fail to do so. Instead, multiple feature points can be used for tracking through a Kalman filter. So, when there is a mismatch, an optimal tracker for any neighbouring feature point with maximum number of matches can assist in determining the next real position.

If the observed object trajectory is changing, then it is important that there is at least one up-to-date optimal tracker. So that even in case of a mismatch, the changing trajectory is correctly estimated by the tracker. Similarly, for the evaluation of a tracker in general, it is essential to evaluate it on a variable trajectory dataset.

In our experiment, we found out that iSGM based estimations were far more accurate, compared to linBP. Similarly, changing the feature point descriptor for frame-to-frame feature point matching would also change the system's performance. SURF descriptor was found to be much better than FREAK and BRIEF descriptors for the DoG detector.

We designed and tested a stereo-based safety system that can issue timely warning to avoid a possible collision scenario. It would also be very interesting to validate the system performance with a laterally moving object, crossing in front of the path of the ego-vehicle. The key limitation for recording such a sequence is the synchronization of ego-vehicle and colliding object. Nevertheless, a safe braking distance away from the point of collision can play an important role in choreographing such a sequence, and eventually evaluating any driver assistance system.

References

1. Abe, G., Richardson, J.: The influence of alarm timing on driver response to collision warning systems following system failure. J. Behaviour & Information Technology 25(5), 443–452 (2006)
2. Alahi, A., Ortiz, R., Vandergheynst, P.: Freak: Fast retina keypoint. In: Proc. IEEE Int. Conf. Computer Vision Pattern Recognition, pp. 510–517 (2012)
3. Bay, H., Tuytelaars, T., Gool, L.V.: Surf: Speeded up robust features. In: Proc. European Conf. Computer Vision, pp. 408–417 (2006)
4. Botterill, T., Mills, S., Green, R.: Fast RANSAC hypothesis generation for essential matrix estimation. In: Proc. Int. Conf. Digital Image Computing Techniques Applications, pp. 561–566 (2011)
5. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: Brief: Binary robust independent elementary features. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 778–792. Springer, Heidelberg (2010)
6. Coifman, B., Beymer, D., McLauchlan, P., Malik, J.: A real-time computer vision system for vehicle tracking and traffic surveillance. Transportation Research Part C: Emerging Technologies 6(4), 271–288 (1998)
7. Fischler, M.A., Bolles, C.R.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Comm. ACM 24(6), 381–395 (1981)
8. Kalman, R.E.: A new approach to linear filtering and prediction problems. J. Basic Engineering 82(1), 35–45 (1960)
9. Khan, W., Morris, J.: Safety of stereo driver assistance systems. In: Proc. IEEE Symp. Intell. Vehicles (IV), pp. 469–475 (2012)
10. Khan, W., Klette, R.: Stereo accuracy for collision avoidance for varying collision trajectories. In: Proc. IEEE Symp. Intell. Vehicles (IV) (2013)
11. Lowe, D.G.: Object recognition from local scale-invariant features. In: Proc. IEEE Int. Conf. Computer Vision, pp. 1150–1157 (1999)