Abstract—We present and evaluate a novel scene descriptor for classifying urban traffic by object motion. Atomic 3D flow vectors are extracted and compensated for the vehicle’s egomotion, using stereo video sequences. Votes cast by each flow vector are accumulated in a bird’s eye view histogram grid. Since we are directly using low-level object flow, no prior object detection or tracking is needed. We demonstrate the effectiveness of the proposed descriptor by comparing it to two simpler baselines on the task of classifying more than 100 challenging video sequences into intersection and non-intersection scenarios. Our experiments reveal good classification performance in busy traffic situations, making our method a valuable complement to traditional approaches based on lane markings.

I. INTRODUCTION

Vision-based intersection detection and recognition [10], [26], [23], [20], [9] is an important task for advanced driver assistance systems and autonomous driving. However, it is also highly difficult for several reasons.

First, nature’s diversity in appearance is hard to capture using a computer algorithm: While large and small intersections exist, they may be marked or unmarked. Traffic lights and signs may or may not be present. Also, the surrounding scenery can be either natural (e.g., trees and bushes) or man-made (e.g., buildings and bridges).

Second, the tilted installation angle of wide angle car-mounted cameras offers rich details in the cars vicinity, but only little details at the region of interest (at distances ≥ 15 meters).

Third, other vehicles, which show up in a variety of different colors and shapes, often occlude major information sources of the scene such as lane markings. Hence, approaches based solely on lane markings are not applicable to busy traffic situations, which make up for most of the inner-city scenarios. A typical ‘intersection’ and ‘non-intersection’ image frame is depicted in figure 1, demonstrating the aforementioned difficulties.

In recent years, much research has been conducted in vision based road geometry estimation and tracking [3], [29], [18], [5], [28], [6], [2], [8], [27], [22], [24]. However, most of the approaches use features based on lane markings and assume an unobstructed view onto the road, which is often not the case in inner-city scenarios. Furthermore, a simple road model based on splines or clothoids is used, which does not naturally allow for representing more complex multi-lane scenarios like intersections.

In contrast to our approach, [16], [17], [13] exploit digital road maps to generate road models which are mapped into the scene and matched against road features in the images, also assuming an unobstructed view onto the road. In [25] feature maps based on aerial images are generated for the same purpose. Deductive inference is used in [19], in combination with description logic to narrow down the space of plausible intersection hypotheses.

In [30] a Kalman-filter based tracking system is presented, which tracks vehicles and pedestrians at intersections. However, in their scenario a static camera on top of a building was used, making position estimation less noisy than in our setting, where depth estimates have to be deduced from disparities and the observer is moving.

In [23] road border discontinuities and lane markings are estimated to detect intersections from imagery. In [9] simple intersections without traffic are recognized by detecting road clothoids and their intersecting lines. An active camera setup is proposed in [20] for the same task. First classification approaches were conducted in [26], where color appearance has been used to segment the shape of the road in a rectified view into several intersection types using a binary support
vector machine. [10] employs super-pixel segmentation to construct feature sets, which are fed into a boosting-based classifier to distinguish between different road types and detect cars, pedestrians and crossings.

In this paper, we present a descriptor for traffic classification which is easy to compute and complementary to existing lane-marking based approaches. Instead of focusing on lane-markings and curbstones, we robustly compute the flow of scene objects like bicycles, pedestrians and cars. This 'object flow' is accumulated in a local 2D map from which histogram grid features are computed, based on a voting scheme. Unlike the map-mosaicing approach proposed in [12], this allows for a memory-efficient implementation, since only a sparse set of vectors has to be stored temporarily. Using a large margin classifier, we find optimal boundaries for classifying these features into the two classes 'intersection' and 'non-intersection' without prior streetmap knowledge.

II. SYSTEM OVERVIEW

Figure 2 gives an overview over our system: First, sparse image features are extracted and matched sub-pixel accurately against spatially and temporally neighboring frames. A robust egomotion approach (section III) yields the extrinsic motion parameters between consecutive frames, which are used for extracting motion compensated 3D object flow vectors from the feature matches. We accumulate these vectors into a local map, containing approximately the last 100 frames. Voting, based on the object flow position and orientation finally gives a global histogram grid descriptor which is fed into a Support Vector Machine for classification (section IV). Our approach is evaluated on more than 100 video sequences (section V). We conclude this paper with an outlook on future work.

III. VISUAL ODOMETRY

Due to our assumption of a moving observer, static parts of the scene have to be compensated for the egovehicle’s movement. The computation of egomotion further allows for registering object flow vectors into a local map. In our experiments we use a time span of 10 seconds, corresponding to ≈ 100 image frames.

Figure 3 shows the configuration of our moving stereo-camera-rig at two consecutive time-steps. Without loss of generality, the world reference frame coincides with the camera coordinate frame of the previous right camera. Using this assumption, the pose of the left camera with respect to the right camera is given by the extrinsic calibration of the stereo-camera-rig \( \{R_C, t_C\} \) which is assumed to be fixed and known. Note that self-calibration methods like [7] could be used to relax this assumption. The poses of the current cameras with respect to the previous right camera \( \{R_{R,k}, t_{R,k}\} \) and \( \{R_{L,k}, t_{L,k}\} \) are defined by the unknown egomotion and the extrinsic calibration parameters.

To parameterize the camera motion, i.e. the orientation of the right camera coordinate frame with respect to the world reference frame, we use a translation vector \( t = (t_X, t_Y, t_Z)^T \) and a rotation matrix \( R(\Theta, \Phi, \Psi) \) which is a
concatenation of rotations around the coordinate axis of the world reference frame. With the knowledge of the egomotion $(V_X, V_Y, V_Z, \omega_X, \omega_Y, \omega_Z)^T$ and the time difference $\Delta T$ this transformation is readily given.

A. Trifocal Constraints for Egomotion Estimation

As can be seen from figure 3, the projection matrices of all cameras $\{P_{r,k-1}, P_{L,k-1}, P_{R,k}, P_{L,k}\}$ can be computed based on the calibration and the egomotion of the stereo rig. The geometric relationship between three of these cameras, i.e. both cameras at frame $k-1$ and one of the cameras at frame $k$ respectively, can be expressed by the trifocal tensor $T$ [31]. This tensor encapsulates the projective relations between three views of a scene. Its entries are determined by

$$T^a_l = (-1)^{i+1} \cdot \text{det} \left( \begin{array}{c} a_i^r \\ b^q \\ c^r \end{array} \right),$$

where $a_i^r$ represents matrix $P_A$ without row $i$ and $b^q$ and $c^r$ are the $q$-th row of $P_B$ and the $r$-th row of $P_C$ respectively. Given the trifocal tensor and matching image points $x_A \leftrightarrow x_B$ in two images, the corresponding image point $x_C$ in the third image can be determined using the point-line-point transfer [31]. This transfer can be expressed by a non-linear mapping via $x_{j,k} = h_j(T, x_{R,k-1}, x_{L,k-1})$ with $j \in \{R, L\}$ defining the current camera.

Thus, the relationship between the trifocal tensor and point correspondences in three images can be used to determine the egomotion of the stereo rig.

In a first step, we use a Laplacian-of-Gaussian (LoG) approximating interest point detector (e.g., CenSurE [1]) to quickly detect blob-like features in the current and previous stereo pair $\{I_{R,k-1}, I_{L,k-1}, I_{R,k}, I_{L,k}\}$.

Second, we extract MuSURF (modified upright SURF) descriptors [1] and match them using the $l_1$-norm for robustness and efficiency. Only reliable correspondences which match in a loop $(x_{L,k-1} \leftrightarrow x_{R,k-1} \leftrightarrow x_{R,k} \leftrightarrow x_{L,k} \leftrightarrow x_{L,k-1})$ are kept. Note that we are using rotation-variant features since exaggerated roll motion between consecutive frames is unlikely.

Third, a bucketing [31] technique is used to reduce the number of features. This results in several advantages: On one hand, the number of correspondences is reduced, hence also the computational complexity of the filtering algorithm. On the other hand, all image features are approximately uniformly distributed over the image plane. This is important, since far features are crucial for reliably estimating the angular velocity and near features guarantee an accurate estimation of the longitudinal vehicle velocity. Furthermore, bucketing guarantees that no biasing towards features on independently moving objects is present, which results in a more stable motion estimation even in highly dynamic environments. The remaining feature points located on independently moving objects as well as false feature correspondences are rejected in an iterative manner using an iterated Extended Kalman-Filter (IEKF) described in the following section.

B. ISPKF based Motion Estimation

To reduce the impact of noisy measurements and to include knowledge about the dynamic behaviour of the vehicle we use a Kalman Filter based state estimation. Because of the non-linearities in the mappings induced by the trifocal tensor we use an Iterated Sigma-Point Kalman-Filter (ISPKF), which linearizes the observation model. For discrete-time systems [14], the state and observation model are given by

$$y_{k+1} = f(y_k) + w_k$$

$$z_{k+1} = h(y_{k+1}) + v_{k+1}.$$ 

Here, $h(\cdot)$ is the non-linear measurement equation as described in section III-A. $w$ and $v$ are the system noise and measurement noise, respectively. The state $y = (V_X, V_Y, V_Z, \omega_X, \omega_Y, \omega_Z)^T$ is defined by the egomotion of the vehicle and the measurements $z$ are the locations of the matched feature points in both current images. Since we are making a constant velocity assumption we have $f(y) = y$.

Compared to non-iterative filtering, the application of an ISPKF yields two central benefits: First, the linearization error can be reduced which results in more accurate motion estimates. Second, we eliminate feature correspondences not consistent with the current estimate by applying RANSAC on a random subset of the features. In a final step, all inliers are used. More details can be found in [21].

IV. OBJECT FLOW

This section describes how we compute the object flow by extracting 3D flow vectors. Note that our method is related to the approach proposed in [11], however we do not employ Kalman filters to track single features over time, but rather project atomic flow vectors onto the ground plane (i.e., bird’s eye perspective). The flow is further processed in a voting-based procedure to give the final histogram descriptor which is used for classification.

A. Local Flow Accumulation and Filtering

Since we are using a stereo system with disparities estimated at sub-pixel accuracy, we can use triangulation to map the already extracted image features $x_{L,k-1}, x_{R,k-1}, x_{R,k}, x_{L,k}$ to 3D points $X_{k-1}, X_k$ with $X \in \mathbb{R}^3$. Without loss of generality we define those points in the coordinate system of the right camera. To compensate for the observer’s motion and to separate the static scene part from dynamic objects, 3D points from frame $k-1$ are mapped into the coordinate system of frame $k$ according to

$$X_{k-1}^+ = (R_{k-1 \rightarrow k} \ t_{k-1 \rightarrow k}) \begin{pmatrix} X_{k-1} \\ 1 \end{pmatrix}$$

where $\{R_{k-1 \rightarrow k}, t_{k-1 \rightarrow k}\}$ denotes the vehicle’s egomotion between frames $k-1$ and $k$. The 3D flow vector at frame $k$
is finally given by
\[
\xi_{k}^{3D} = \left(\frac{X_k - X_k^+}{\Delta T_{k-1 \rightarrow k}}\right) \in \mathbb{R}^6 \tag{5}
\]
with time difference \(\Delta T_{k-1 \rightarrow k}\). The first 3 elements of \(\xi_{k}^{3D}\) represent the flow vector’s location (in meters) in frame \(k\)’s coordinate system and the last 3 elements are its velocity (in meters per second). To account for outliers and static objects in the scene we apply three additional steps:

First, we reject all flow vectors for which the compensated optical flow (reprojection into the image plane) does not exceed 8 pixels. We further keep only flow vectors which are positioned at plausible distances (5-50 meters). Last, we limit the flow vector speed to values between 0.3 \(m/s\) and 30 \(m/s\). This filtering step is illustrated in figure 2 (left).

Since disparity depth estimates are usually highly sensitive to calibration errors and noisy at large distances, a 3D median filter is adopted at each frame with a kernel size of 3 meters. Finally, temporal integration is achieved by building anisotropic voting weights according to the motion direction \(\xi_{k}^{3D}\) at position \(\mathbf{p}_i = (p_{x,i}, p_{y,i})^T\) and with velocity \(\mathbf{v}_i = (v_{x,i}, v_{y,i})^T\). Figure 2 (middle) illustrates the object flow vectors extracted from a typical intersection scenario.

B. Voting-based Flow Description

Since our final goal is classification, a fixed-sized descriptor must be extracted from the variable-sized object flow vectors described in the previous section. We tackle this problem by employing a voting scheme, in which each flow vector casts votes for its own direction on a fixed 2D grid of size \(M_x \times M_z\). Flow vectors are sparse, but can be extrapolated to a certain extend. Thus we decided for anisotropic voting weights according to the motion direction (see figure 4). In the following, we denote the grid locations by \(\{q_j = (q_{x,j}, q_{z,j})^T\}_{j=1}^{M_x \times M_z}\). In order to accumulate all votes, each grid point \(j\) is represented by a histogram \(\mathbf{h}_j \in \mathbb{R}^{M_{\theta}}\), quantizing the object flow into \(M_{\theta}\) orientations. Thus, the proposed descriptor becomes a vector of size \(M_x \times M_z \times M_{\theta}\).

Every object flow instance \(\xi_{i}^{2D} = (p_{x,i}, p_{z,i}, v_{x,i}, v_{z,i})^T\) at position \(\mathbf{p}_i\) casts one vote for its orientation \(\mathbf{v}_i\) to each histogram \(\mathbf{h}_j\) on the grid, located at \(q_j = (q_{x,j}, q_{z,j})^T\).

The weight of each vote \(w(q, \xi_{i}^{2D})\) depends on the grid point \(q\) and the flow \(\xi_{i}^{2D}\) itself via
\[
w(q, \xi_{i}^{2D}) = \exp \left\{- (q - \mathbf{p})^T \mathbf{W} (q - \mathbf{p}) \right\} \in [0,1]. \tag{6}
\]
Here \(\mathbf{W} \in \mathbb{R}^{2 \times 2}\) is defined by its eigenvalue decomposition
\[
\mathbf{W} = \mathbf{R} \Lambda \mathbf{R}^T \tag{7}
\]
with
\[
\mathbf{R} = \begin{pmatrix}
1 & 1 \\
1 & -1
\end{pmatrix}, \quad \Lambda = \begin{pmatrix}
1/\lambda_1^2 & 0 \\
0 & 1/\lambda_2^2
\end{pmatrix}. \tag{8}
\]
\(\Lambda\) and \(\mathbf{W}\) are mathematically similar and \(\mathbf{R}\) is an orthonormal change-of-basis matrix, defined by the orientation of the respective object flow \(\xi_{i}^{2D}\).

\[
r_1 = \frac{v_x}{\sqrt{v_x^2 + v_z^2}} \quad r_2 = \frac{v_z}{\sqrt{v_x^2 + v_z^2}} \tag{9}
\]
Here, the parameters \(\lambda_1\) and \(\lambda_2\) control the longitudinal and lateral influence of each flow vector.

The weight calculation step is illustrated in figure 4 for one single flow vector and a grid of size \(16 \times 16\). After accumulating the votes from all object flow vectors, each of the \(M_x \times M_z\) histograms \(\mathbf{h}_j\) is individually normalized (i.e., divided) by \(\max(10, \max(\mathbf{h}_j))\). Concatenating all histograms finally yields the object flow descriptor
\[
\mathbf{d} = (\mathbf{h}_1, \ldots, \mathbf{h}_{M_x \times M_z}), \tag{10}
\]
a vector of size \(M_x \times M_z \times M_{\theta}\). Figure 2 (right) illustrates an object flow descriptor with \(M_x = M_z = 16\), where the arrow directions represent \(M_{\theta} = 8\) canonical histogram orientations and the arrow lengths depict the relative number of votes for each orientation.

V. EXPERIMENTS

A qualitative assessment of the proposed method is given in figures 5,7. Figure 7 depicts in each row two frames of the sequence, the extracted object flow and the computed
In Fig. 5, we see the object flow of a lane change maneuver. This figure depicts the object flow field of a preceding car. While the first car changes lanes from the left to the right, the observer changes lanes in the opposite direction.

To evaluate the proposed descriptor quantitatively, we recorded 113 grayscale video sequences using our car-mounted stereo rig at a framerate of 10 fps, with $1382 \times 512$ pixels and an opening angle of $90^\circ$. Each sequence contained approximately 10 seconds, which was empirically found to be well suited for our method. We manually classified the sequences into 65 positive (intersections) and 48 negative (non-intersections) examples.

For comparison, two baselines were created: The first baseline, called ImageFlow 2d accumulates sparse optical flow vectors of the left camera in a single histogram with eight bins corresponding to eight canonical directions in the image. The second baseline, called ObjectFlow 2d is essentially the same, except that the optical flow vectors in the image were compensated for the vehicles' egomotion prior to histogram voting, using the technique described in section III. The idea behind both baseline algorithms is to capture the main optical flow direction in the image: A high horizontal optical flow ratio should be expected at intersections while non-intersection scenarios are supposed to produce mainly vertical flow vectors in the image. However, both baselines make no use of depth information, except for compensating the egomotion (ObjectFlow 2d).

To account for the high dimensionality of the ObjectFlow descriptor proposed in section IV, we extended the training set by shifting all examples by $\pm 2$ meters in the $x/z$ plane. Sure enough, we did not include the translated instances of the test data into the training set of the respective run for fairness reasons.

For classification we employed a soft-margin support vector machine (SVM), making use of the libsvm library [4]. A linear kernel was chosen to avoid overfitting due to the high-dimensional nature of the feature space ($M_x \times M_z \times M_\theta$). Since the number of recorded scenarios (113) is relatively small compared to the high-dimensional feature space, leave-one-out cross-validation was employed.

Figure 6 shows the result of our experiments in terms of true positives, false positives, true negatives, false negatives, precision and recall. Here a ‘true positive’ denotes a correctly detected intersection. We compare the two baselines to our method using four sets of histogram grid parameters. The parameter $w$ adjusts the influence area of the exponential voting kernel from equation 8 via

\[
\lambda_1 = 3w \quad \lambda_2 = w
\]
Fig. 7. Results on 2 non-intersection and 3 intersection sequences. Each row depicts one scenario: The left column shows two image frames of the left camera, the middle column displays the extracted object flow from a bird’s eye perspective (unit: meter) and the right column depicts the histogram grid descriptor using a step size of $s = 5$ meters.
and $s$ is the histogram grid step size, e.g. $s = 2$ in figure 4 and $s = 5$ in figure 7.

While the ImageFlow 2d and ObjectFlow 2d baselines perform quite well already, adding depth information clearly helps in discriminating intersection scenarios versus non-intersection scenarios. For all parameter sets of the ObjectFlow descriptor no false positive has been reported. The false negative rate differs slightly among the parameter settings with the lowest rate for $w = 5$ and $s = 10$. We believe that this ‘averaging’ parameter set is mainly preferred as a consequence of the relatively small amount of training examples. Still, all results clearly indicate the benefits of stereo-based measurements with a precision of 1.0 and a recall $\geq 0.89$. Our system currently runs on a single CPU core at about 2 frames per second. Since the most computationally expensive part is feature matching, real-time can be achieved with a GPU or FPGA-based feature matcher.

**VI. CONCLUSION AND FUTURE WORK**

In this paper a complementary novel descriptor for visually classifying traffic motion into intersection and non-intersection scenarios from within a moving vehicle has been proposed and evaluated against two baseline algorithms. Our findings indicate that including depth information into the classification process clearly outperforms classifiers with image based motion features. Though our results are promising, we intend to include more image sequences to the training data base in the future. Further, we plan to investigate other features for matching as well as combining the complementary object flow features with road based ones into local maps. Robustly fitting road topography models to this data promises valuable directions for future research towards advanced driver assistance systems and autonomous driving in difficult urban scenarios.

**VII. ACKNOWLEDGMENTS**

We would like to thank the Karlsruhe School of Optics and Photonics and the Deutsche Forschungsgemeinschaft for supporting this work and the reviewers for their comments.

**REFERENCES**


