Optical Speed Measurement via Inverse Perspective Mapping

Firas Kastantin, Tamás Dancs, Bandó Kovács, Tekla Tóth, and Levente Hajder

Faculty of Informatics, Eötvös Loránd University
{qk4hgr,w5u6vq,kovbando,teklatoth,hajder}@inf.elte.hu

Abstract. This paper deals with speed measurement from images of a vehicle-mounted camera. The solution is obtained via transformed images by Inverse Perspective Mapping (IPM). In order to generate IPM images, the non-perspective distortion of the camera should be rectified. The calibration method of Scaramuzza et al. is applied here for this purpose. From the IPM video, 2D velocity vectors are obtained by Lucas-Kanade feature tracking and Gunnar-Farneback optical flow. The final vehicle speed is robustly estimated from the speed vectors. The metric speed vector of the vehicle can also be calculated due to the applied markers in the calibration step.

1 Introduction

Recently, research on algorithms for autonomous systems has become one of the most popular topics in computer vision. This paper deals with the processing of videos coming from a vehicle-mounted cameras.

The aim of this paper is to measure the speed of the vehicle. This task is called Visual Odometry (VO) in the literature [12]. Thought there are more effective sensors, that can be e.g. tachometer or GPS, to measure the speed, video-based measurement is also interesting as it can be applied as part of a visual system, thus, it is valuable for both research and educational purposes.

The proposed approach here is based on Inverse Perspective Mapping (IPM) [6]. IPM is a perspective transformation between the original and a hypothetical image. IPM processes the frontal camera view as input, it applies the suitable transformation, i.e. a homography, and produces a top-down view of the scene by mapping the pixels to a different 2D-coordinate frame, which is also known as Bird's-Eye View (BEV).

IPM images/videos can be exploited very effectively within tasks such as lane detection [11] road marking detection [10], road topology retrieve [1], object detection and tracking [14, 3], as well as intersection prediction [9] or path planning.

The main contribution of this paper is to develop a novel system in order to implement visual odometry-based speed measurement. An SJCAM SJ4000 digital camera with high Field of View (FoV) is applied for the tests, its radiometric calibration is also addressed here.

The structure of the paper is as follows. The calibration of high FoV cameras is discussed in Section 2. The theoretical and practical issues of IPM image generation is overviewed in Section 3. Then our novel optical flow based speed measurement is proposed in Section 4. The novel approach is tested on real images sequences, it is discussed in Section 5. Finally, the work is concluded and some possible research directions are proposed in Section 6

2 Omnidirectional Camera Calibration

Images captured with cameras that have a large FoV always have a significant non-perspective distortion. Such images can cause bad visual feedback for human eyes and most importantly, it is very hard to perform computer vision tasks – such as feature matching, or any kind of object detection – in these images. However, there are ways to remove these distortions. In this section, the calibration method of Scaramuzza et al. [13] is overviewed in brief.

2.1 Camera model

For this purpose, first, we have to define the camera model. The perspective camera model can be easily used for a standard camera, but when we are considering a camera with e.g. fisheye lenses, there may be inaccuracies. This is the reason why we are using a different camera model. Scaramuzza et al. [13] introduced a general omnidirectional camera model that can be used for both dioptric and catadioptric camera systems. In this model, we have two different planes: the camera image plane, and the sensor plane. A point on these planes denoted with $\mathbf{u}' = [u' \quad v']^T$ and $\mathbf{u}'' = [u'' \quad v'']^T$ respectively.

In Figure 1, we can see the catadioptric case of the model. In the dioptric case the sign of is reversed due to the lack of reflective surface. We can easily observe that they are related by an affine transformation. We also introduce an image projection function, which shows the relationship between the sensor plane and rays emanating from the viewpoint. The complete model for general omnidirectional cameras is as follows:

$$\lambda \mathbf{p} = \lambda \mathbf{g}(\mathbf{u}'') = \lambda \mathbf{g}(\mathbf{R}\mathbf{u}' + \mathbf{t}) = \mathbf{P}\mathbf{X}, \quad \lambda > 0, \tag{1}$$

where $\mathbf{X}$ is a homogeneous coordinate of a scene point and $\mathbf{P}$ is the projection matrix, containing extrinsic camera perameters, that are represented by a rotation $\mathbf{R}$ and a translation $\mathbf{t}$ as $\mathbf{P} = [\mathbf{R}|\mathbf{t}]$. By the calibration of the model we mean the estimation of the affine parameters, that are stacked in matrix $\mathbf{R}$ and vector $\mathbf{t}$. $\mathbf{g}$ is the non-linear function that satisfy the projection equation. For the latter one, we assume the following:

$$\mathbf{g}(u'', v'') = (u'', v'', f(u'', v''))^T \tag{2}$$

where $f(u'', v'') = a_0 + a_1\rho + a_2\rho^2 + \ldots + a_n\rho^n$. In the calibration, we are estimating the coefficients denoted by ρ_i, $i \in \{1, \ldots, n\}$ and the polynomial degree by n. $\rho > 0$ is the metric distance of the pixel w.r.t. the sensor plane.

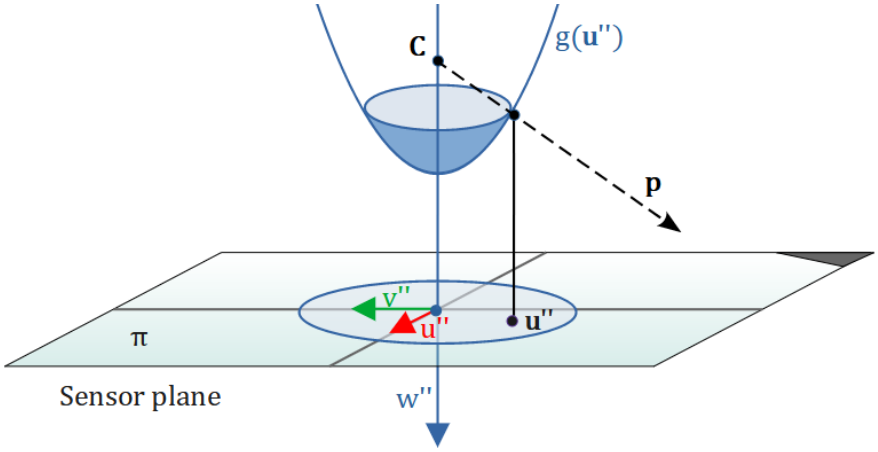


Fig. 1: The catadioptric coordinate system. The camera model describes the connection between the **p** direction of a 3D scene point and the corresponding image point $\mathbf{u}''$. Note, the function g is rotational symmetric around $\mathbf{w}''$ and limited along $\mathbf{u}''$ and $\mathbf{v}''$. Hence, the camera image is a circle in the sensor plane.

2.2 Camera Calibration

In the first step of the calibration, the affine parameters are computed up to a scale factor α to reduce the number of unknown parameters to be estimated. After performing the affine transformation, the relationship between $\mathbf{u}'$ and $\mathbf{u}''$ can be written as $\mathbf{u}'' = \alpha\mathbf{u}'$. Thus, the projection equation can be rewritten as:

$$\lambda\mathbf{g}(\alpha\mathbf{u}') = \lambda \begin{bmatrix} \alpha u' \\ \alpha v' \\ f(\alpha\rho') \end{bmatrix} = \lambda\alpha \begin{bmatrix} u' \\ v' \\ \alpha_0 + \alpha_1\rho + \cdots + a_n\rho^n \end{bmatrix} = \mathbf{PX}. \tag{3}$$

where u' and v' are pixel coordinates of a point on the image with respect to the circle center, and ρ is the Euclidean distance from the circle center. The factor α can be integrated into the depth factor λ, thus the parameters to be determined are: $(a_0, a_1, \ldots, a_n)$.

We use planar patterns in different positions for calibration, similarly to the well-known calibration method of Zhang [15], so the extrinsic parameters between the sensor plane and the planar object can be extracted. Let $\mathbf{M}_{ij} = [X_{ij} \quad Y_{ij} \quad Z_{ij}]^T$ and $\mathbf{m}_{ij} = [u_{ij} \quad v_{ij}]^T$ be a 3D point in the pattern coordinate system and its corresponding point on the image respectively. We can assume that $Z_{ij} = 0$ as we are using a planar pattern. In order to solve the camera calibration, we have to determine the extrinsic parameters for each image of the

calibration pattern:

$$\lambda_{ij}\mathbf{p}_{ij} = \lambda\alpha \begin{bmatrix} u_{ij} \\ v_{ij} \\ \alpha_0 + \alpha_1\rho + \cdots + a_n\rho^n \end{bmatrix} = \mathbf{P}_i\mathbf{X} = \begin{bmatrix} \mathbf{r}_1^i \\ \mathbf{r}_2^i \\ \mathbf{r}_3^i \\ \mathbf{t}^i \end{bmatrix}^T \begin{bmatrix} X_{ij} \\ Y_{ij} \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1^i \\ \mathbf{r}_2^i \\ \mathbf{t}^i \end{bmatrix}^T \begin{bmatrix} X_{ij} \\ Y_{ij} \\ 1 \end{bmatrix}, \tag{4}$$

where $\mathbf{r}_j^i = \begin{bmatrix} r_{1j}^i & r_{2j}^i & r_{3j}^i \end{bmatrix}^T$ is the j-th column vector of rotation matrix $\mathbf{R}^i$. Matrix $\mathbf{R}^i$ and vector $\mathbf{t}^i$ are the extrinsic parameters for the i-th frame. In this equation, we can eliminate the dependence from the factor λ_{ij} by multiplying both sides by cross product with $\mathbf{p}_{ij}$. After the elimination, each $\mathbf{p}_i$ point on the pattern will produce three homogeneous equations:

$$v_j\left(r_{31}X_j + r_{32}Y_j + t_3\right) - f(\rho)\left(r_{21}X_j + r_{22}Y_j + t_2\right) = 0, \tag{5}$$

$$u_j\left(r_{31}X_j + r_{32}Y_j + t_3\right) - f(\rho)\left(r_{11}X_j + r_{12}Y_j + t_1\right) = 0, \tag{6}$$

$$v_j\left(r_{31}X_j + r_{32}Y_j + t_3\right) - u_j\left(r_{21}X_j + r_{22}Y_j + t_2\right) = 0. \tag{7}$$

If we reformulate the third equation into the form $\mathbf{NH} = 0$, where $\mathbf{H} = [r_{11}, r_{12}, r_{21}, r_{22}, t_1, t_2]^T$ and

$$\mathbf{N} = \begin{bmatrix} -v_1X_1 & -v_1Y_1 & u_1X_1 & u_1Y_1 & -v_1 & u_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -v_lX_l & -v_lY_l & u_lX_l & u_lY_l & -v_l & u_l \end{bmatrix}, \tag{8}$$

then a homogeneous linear system of equations is obtained that can be optimally, in the least squares (LS) sense, solved via an eigenvalue-eigenvector computation [6].

The estimation of h can be done by minimizing the expression with respect to the criterion $||\mathbf{H}||^2 = 1$. Because the columns of a rotation matrix are orthonormal, the solution can be uniquely determined along with the unknown r_{31} and r_{32} parameters. Doing this with every calibration image we can estimate all of the extrinsic parameters except t_3, which can be determined along with the intrinsic parameters.

To estimate the intrinsic parameters, we are exploiting the first and second equations shown above to estimate the coefficients $(a_0, a_1, \cdots, a_n)$ and t_3 for each calibration image. After reformulating the equations, the problem can be reformulated as follows:

$$\begin{bmatrix} A_1 & A_1\rho_1 & \cdots & A_1\rho_1^n & -u_1 & 0 & \cdots & 0 \\ C_1 & C_1\rho_1 & \cdots & C_1\rho_1^n & -v_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ A_k & A_k\rho_k & \cdots & A_k\rho_k^n & 0 & 0 & \cdots & -u_k \\ C_k & C_k\rho_k & \cdots & C_k\rho_k^n & 0 & 0 & \cdots & -v_k \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \\ t_3^1 \\ t_3^2 \\ \vdots \\ t_3^k \end{bmatrix} = \begin{bmatrix} B_1 \\ D_1 \\ \vdots \\ B_k \\ D_k \end{bmatrix}, \tag{9}$$

$$\text{where} \quad \begin{aligned} A_i &= r_{21}^i X_i + r_{22}^i Y_i + t_2^i, & B_i &= v_i \left(r_{31}^i X_i + r_{32}^i Y_i \right), \\ C_i &= r_{11}^i X_i + r_{12}^i Y_i + t_1^i, & D_i &= u_i \left(r_{31}^i X_i + r_{32}^i Y_i \right). \end{aligned} \tag{10}$$

The LS solution can be obtained by using the pseudoinverse. This way the intrinsic parameters $a_1, a_2 \ldots, a_n$ will be estimated. To compute the best polynomial degree n, we start from $n = 2$, then we increase until the reprojection error of the calibration points will be minimal.

(a) Original image. (b) Rectified image.

Fig. 2: Image rectification by calibrating the non-perspective camera parameters.

If the parameters are calibrated, the rectification can be carried out for the images of the processed video sequence. An example is pictured in Figure 2. The calibration algorithm works very accurately as it is clearly scene. The only drawback of the calibration is that the FOV of the rectified images are significantly smaller, therefore, information close to the original border are lost.

3 Inverse Perspective Mapping (IPM)

IPM is the method of omitting the **perspective effect**. Camera works like human eye works, the image is formed by the intersections of the optical rays through the focal point with the so-called image or picture plane, and this projection is called **perspective projection**.

The perspective projection is given using the formula as follows:

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}$$

, where u_i, v_i are the coordinates of the point i in the image plane, x_i, y_i, z_i are the spatial coordinates of the point i. $\mathbf{P}$ is the so-called projection matrix.

We have two problems in the perspective projection model (perspective effect):

- The actual parallel lines in nature are non-parallel in the image plane (they intersect at the point of infinity).
- The distant objects appear smaller than the close ones to the focal point, in other word the relation between the distances in the image plane and the actual distances are nonlinear.

Homography-Based IPM is one of the techniques to omit the perspective effect, it works by transforming one of the spatial plane to the image plane, it eliminates the perspective effect of the transformed plane.
Since we perform plane-to-plane transformation, we can assume that one of the spatial coordinate is always constant (e.g. $y_i = 0$) without loss of generality, then we can write the projection formula as follow:

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11}\ p_{13}\ p_{14} \\ p_{21}\ p_{23}\ p_{24} \\ p_{31}\ p_{33}\ p_{34} \end{bmatrix} \begin{bmatrix} x_i \\ z_i \\ 1 \end{bmatrix},$$
$$\text{where } \begin{bmatrix} p_{11}\ p_{13}\ p_{14} \\ p_{21}\ p_{23}\ p_{24} \\ p_{31}\ p_{33}\ p_{34} \end{bmatrix} = \begin{bmatrix} h_{11}\ h_{12}\ h_{13} \\ h_{21}\ h_{22}\ h_{23} \\ h_{31}\ h_{32}\ p_{33} \end{bmatrix} = \mathbf{H},$$

and **H** is the so-called **Homography Matrix**.

Since we are interested in calculating the speed of the vehicle and it travels on the roadway, we choose the **roadway surface** as our spatial plane. The homography-based IPM works in three steps: (i) Calculating the homography matrix **H**. (ii) Build the pixel-to-pixel mapping matrix. (ii) Finally, perform plane-to-plane transformation.

3.1 Homography Matrix

The homography matrix **H** has 8 DOF, we need **four point correspondences** between the spatial plane (roadway plane) and the perspective image. It is shown here that homography estimation can be solved as a homogeneous linear system of equations. The hompgraphy projects the coordinates as follows:

$$u_i = \frac{h_{11}x_i + h_{12}z_i + h_{13}}{h_{31}x_i + h_{32}z_i + h_{33}}, \quad v_i = \frac{h_{21}x_i + h_{22}z_i + h_{23}}{h_{31}x_i + h_{32}z_i + h_{33}}.$$

After multiplication by the common denominator, it can be written that

$$h_{31}x_iu_i + h_{32}z_iu_i + h_{33}u_i - h_{11}x_i - h_{12}z_i - h_{13} = 0,$$
$$h_{31}x_iv_i + h_{32}z_iv_i + h_{33}v_i - h_{21}x_i - h_{22}z_i - h_{23} = 0.$$

Fig. 3: Four yellow markers are placed in order to estimate the homography. Best viewed in color.

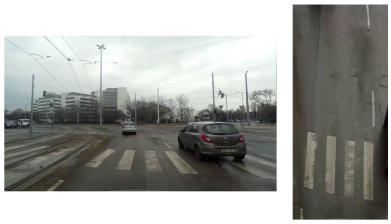
Therefore, the final linear system is as follows:

$$\begin{bmatrix} -x_i & -z_i & -1 & 0 & 0 & 0 & u_i x_i & u_i z_i & u_i \\ 0 & 0 & 0 & -x_i & -z_i & -1 & v_i x_i & v_i z_i & v_i \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0.$$

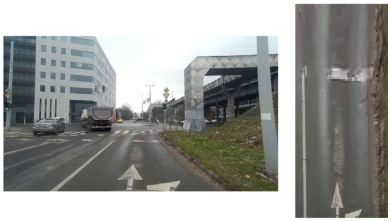
We used four markers, see Figure 3, which form a rectangle with 12×2.5 m, the spatial coordinates are $s_1 = [0 \quad 0]^T$, $s_2 = [2.5 \quad 0]^T$, $s_3 = [0 \quad 12]^T$, and $s_4 = [2.5 \quad 12]^T$. For each marker corresponding image coordinates in the perspective image are measured $p_i = [u_i \quad v_i]^T$.

$$\begin{bmatrix} -s_{1,x} & -s_{1,y} & -1 & 0 & 0 & 0 & p_{1,x}s_{1,x} & p_{1,x}s_{1,y} & p_{1,x} \\ 0 & 0 & 0 & -s_{1,x} & -s_{1,y} & -1 & p_{1,y}s_{1,x} & p_{1,y}s_{1,y} & p_{1,y} \\ -s_{2,x} & -s_{2,y} & -1 & 0 & 0 & 0 & p_{2,x}s_{2,x} & p_{2,x}s_{2,y} & p_{2,x} \\ 0 & 0 & 0 & -s_{2,x} & -s_{2,y} & -1 & p_{2,y}s_{2,x} & p_{2,y}s_{2,y} & p_{2,y} \\ -s_{3,x} & -s_{3,y} & -1 & 0 & 0 & 0 & p_{3,x}s_{3,x} & p_{3,x}s_{3,y} & p_{3,x} \\ 0 & 0 & 0 & -s_{3,x} & -s_{3,y} & -1 & p_{3,y}s_{3,x} & p_{3,y}s_{3,y} & p_{3,y} \\ -s_{4,x} & -s_{4,y} & -1 & 0 & 0 & 0 & p_{4,x}s_{4,x} & p_{4,x}s_{4,y} & p_{4,x} \\ 0 & 0 & 0 & -s_{4,x} & -s_{4,y} & -1 & p_{4,y}s_{4,x} & p_{4,y}s_{4,y} & p_{4,y} \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0, \qquad (11)$$

where indices $(*, x)$ and $(*, y)$ denote first and second 2D coordinates, respectively.



(a) Before a pedestrian crossing.
Karinthy Frigyes Str., Budapest



(b) At crossroads.
Dombóvári and Budafoki Str., Budapest

Fig. 4: **Left.** Camera images, **Right.** Bird Eye's View, obtained by estimated homography.

By solving these eight linear equations using LS method, we get the values of the homography matrix **H**. If the left coefficient matrix in Eq. 11 is denoted by **C**, the LS solution is the eigenvector of $\mathbf{C}^T\mathbf{C}$ corresponding to the least eigenvalue [6].

3.2 Homography transformation

Then the estimated homography **H** can be applied in order to transform the original pixels into their new positions. The computational cost for a single frame transformation is $\mathcal{O}(wh)$, where w and h denote the width and height of the image. The generated image is called **Bird's Eye View (BEV)** or **top-down view**, see Figure 4 for two examples.

When we performed IPM, we assumed that the roadway is a plane but in the case when the roadway has elevations or some object is placed on it, IPM generates distorted BEV but however, we accept the transformation as an estimation.

4 Optical flow based speed measurement

In our approach, we propose a method to find the vehicle's velocity vector based on optical flow of the BEV. For each consecutive BEVs (BEV_i, BEV_{i+1}), the translation vector is calculated then velocity vector is calculated based on the change of the vehicle's position. Two different optical-flow-based methods are tested:

- Feature-based optical-flow.
- Dense optical-flow.

4.1 Feature-based optical-flow

The visual features are extracted from BEV_i using Shi-Tomasi corners detector [7], the best 1000 corners are only considered.

$$(f_1^i, f_2^i, .., f_n^i)$$

f_j^i is the position of the feature j in BEV_i.

Optical flow Lucas-Kanade feature tracker [2] is used to detect the new position of BEV_i detected features in BEV_{i+1}.

$$(f_1^{i+1}, f_2^{i+1}, ..., f_n^{i+1})$$

Translation vector of feature j in BEV_i is given by:

$$\mathbf{t}_j^i = f_j^{i+1} - f_j^i = [f_{j,x}^{i+1} - f_{j,x}^i \quad f_{j,y}^{i+1} - f_{j,y}^i]^T$$

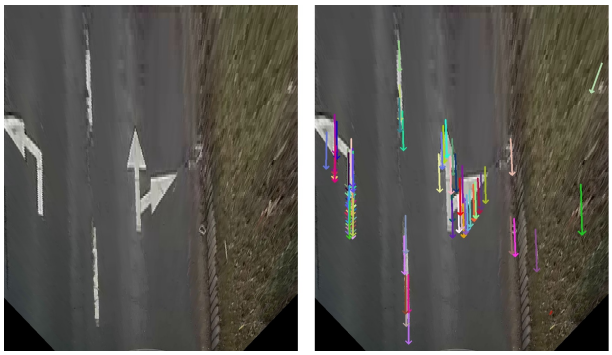


Fig. 5: **Left.** BEV image. **Right.** BEV with feature translation vectors, estimated by [2].

Two issues emerge:

- Many translation vectors are detected but we are interested to detect only the translation vectors related to the vehicle motion.
- When detecting the features in BEV_i and BEV_{i+1}, we assume that the whole scene is fixed and only the vehicle travels but there might be moving objects indeed.

We solve the issues using RANSAC (Random sample consensus) outliers rejection technique [5]. In each RANSAC iteration, a random translation vector t_k^i is selected and examined, we keep the translation vector fits best the features translation in BEV_i to the new positions in BEV_{i+1} which has the maximum number of inliers with threshold σ. t^i is the best examined translation vector which has the maximum number of inliers, and $-t^i$ is the vehicle translation vector due to the fact that the vehicle moves in the opposite direction of the features movement. In our test, we use 20 cm, 1000, and 0.95 for the threshold σ, max number of iterations, and confidence as RANSAC parameters, respectively.

The velocity vector is given by:

$$\mathbf{v}^i = \frac{-\gamma \mathbf{t}^i}{\Delta time}$$

Fig. 6: Estimated speed (bottom left) and vehicle velocity vector (right) are visualized.

where $-\mathbf{t}^i$, γ, and $\Delta time$ are the vehicle translation vector, the pixel size, and the time interval between BEV_{i+1} and BEV_i, respectively. We visualized the estimated values in every frame of the video as it can be seen in the Figure 6.

4.2 Dense optical-flow

Dense optical-flow method estimates the motion of the entire IPM image (all pixels) unlike the feature-based method which estimates the motion for some interesting corners and edges. We use Gunner Farneback's algorithm [4] which takes two consecutive frames BEV_i, BEV_{i+1} and for each pixel in BEV_i it estimates the new location in BEV_{i+1}.

In the same way we estimated the translation vector in feature-based optical-flow, we use RANSAC [5] to find the vehicle translation vector and we define two criteria:

- **Magnitude threshold**: a pixel should have a translation vector with at least c_1 magnitude to be considered in the robustification step. We add this criterion to prevent no texture pixels from dominating over other pixels.
- **Minimum number of inliers**: a translation vector should have at least c_2 inliers to be considered. We add this criterion to prevent small object movement from affecting the vehicle motion estimation.

An additional error reduction technique is applied by using **Kalman Filter** [8]. Kalman Filter works in two steps, the translation vector is corrected in the correction step when a movement is detected using the aforementioned technique,

Fig. 7: **Left.** Perspective image with estimated speed (bottom left) and vehicle velocity vector (right) are visualized. **Mid.** Top-down (BEV) image. **Right.** Top-down image motion with heat map.

otherwise the translation vector is predicted in the prediction step using the translation vector of the previous frame and if no movement is detected for more than 10 consecutive frames the vehicle is considered not moving.

5 Experiments

Our proposed technique estimates the velocity vector with good precision and close to the real velocity vector when optical-flow correctly estimates the motion, and that is due to our robustification technique using RANSAC [5] outlier rejection strategy.

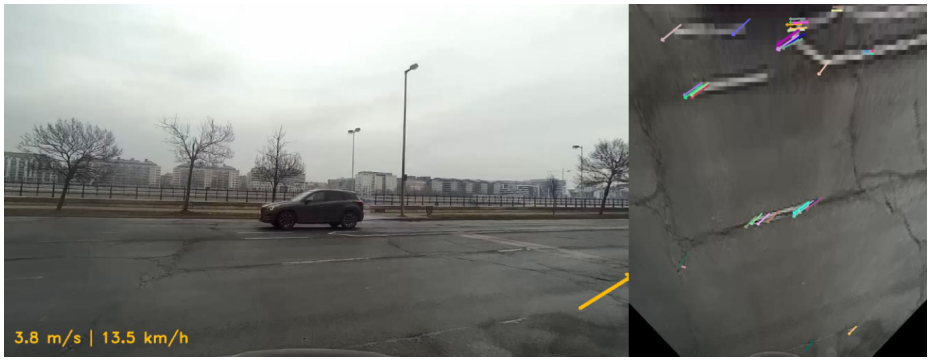
The results are visualized in two videos, they are available at our website[1]. The main difference between the videos is the applied method to estimate the optical flow. The results are visualized for both the feature-based method and the one that uses dense optical flow.

The quality of the speed measurement can only be evaluated qualitatively as ground truth data are not available. We basically think that the quality of optical flow based speed measurement is satisfactory, however, there are short periods within the videos when the speed is not realistic. For feature-based flow, it is trivial that the speed cannot be accurately estimated if there are no trackable features in the video. It is also a challenging problem to filter out the motion of another moving vehicle. This cases are discussed below in short.

The Gunnar-Farneback dense optical flow seems to be a better choice as it generates more flow. Principally, features can be detected and tracked around road paintings, e.g. lane separators or zebra-crossing. In this cases, the speed can be accurately reconstructed. This statement is true for both feature-based and dense optical flow.

The whole trajectory and the computed speed charts are pictured in Figure 10. In the charts, the magnitude and the angle of the estimated metric

[1] Resulting videos are available at `http://cg.elte.hu/~hajder/ELTECar/`



(a) The velocity vector is estimated while the vehicle is turning right.

(b) The velocity vector is estimated while the car travels with high speed.

Fig. 8: Velocity vector estimation in different traffic scenarios. **Left.** Original images with displayed speed in the bottom-left corner and visualized velocity vector on the right. **Right.** IPM image with tracked features. Best viewed in color.

velocity vector are visualized. The stops and turnings are highlighted as they can be straightforwardly detected in the magnitude and angle chart, respectively. The spread of the magnitude and speed chart is very high, thus, the precision of the speed estimation is low. Therefore, it is proposed to use other techniques to improve the quality of speed reconstruction. We plan to use 3D vision methods [6] for this purpose, this is a possible future work.

A few examples are also visualized, see those in Figure 8. The original and IPM images are on the left and right, respectively. Both the magnitude and the direction of the velocity vector is visualized in the bottom-left and bottom-right of the original images. The tracked feature points are pictured in the IPM images.

Our technique still has some limitations. Some examples are exhibited in the Figure 9. Estimating the velocity vector is difficult when the detected corners

are located on a moving object, see in the example 9(a). Another problematic case is when not enough good features are detected by feature-based optical flow. One situation for the latter case is in the picture 9(b). Another problem appears when the roadway is not a plane which causes the car to go up and down, in turn it causes wrong estimation of the translation vectors as it is seen in the subfigure 9(c).

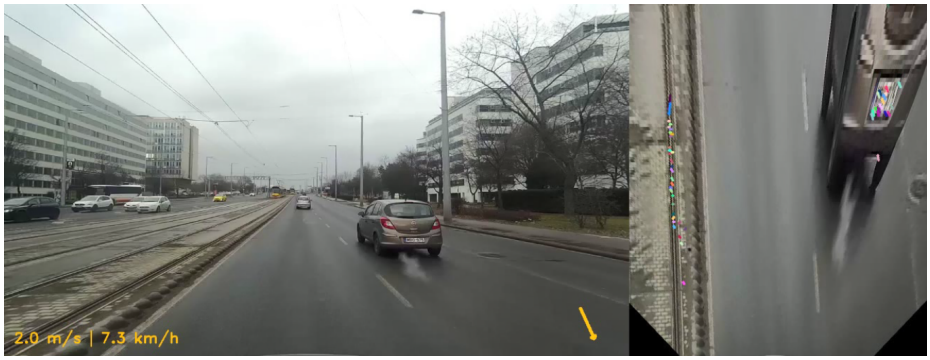
6 Conclusion

This paper addresses the problem of speed measurement from videos of a vehicle-mounted camera. IPM images are computed for this purpose. The proposed method works for both pin-hole and omnidirectional cameras. However, images from latter ones should be rectified first, we propose the method of Scaramuzza et al. [13] in order to rectify non-perspective distortions. From the IPM video, velocity vectors in 2D are estimated by both the Lucas-Kanade feature tracking and Gunnar-Farneback optical flow. The metric speed vectors can be calculated as the metric distances are known in the IPM images due to the marker-based calibration process. The proposed approach is tested on real-world images taken by our SJCAM SJ4000 digital event camera with high field of view.

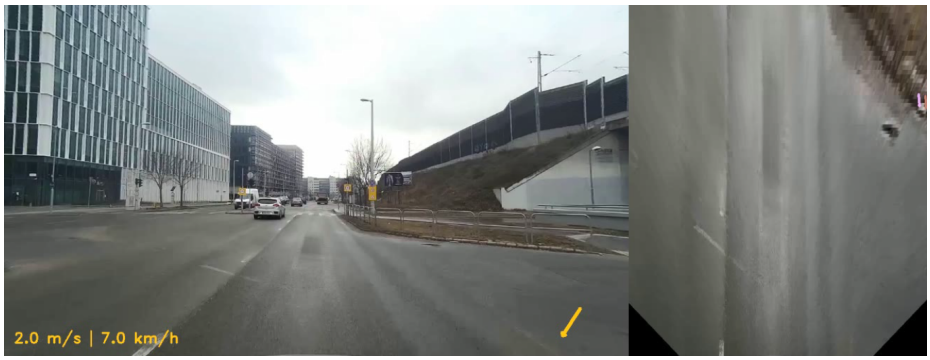
Acknowledgements

The authors have been financed by the project EFOP-3.6.3-VEKOP-16-2017-00001: Talent Management in Autonomous Vehicle Control Technologies, by the Hungarian Government and co-financed by the European Social Fund. Their work was also supported by Thematic Excellence Programme, Project no. ED 18-1-2019-0030, Industry and Digitization Subprogramme, NRDI Office.

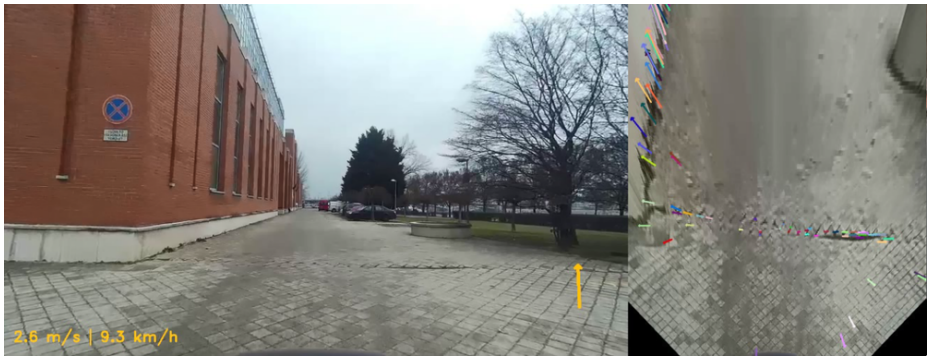
L. Hajder has been supported by the Application Domain Specific Highly Reliable IT Solutions project. It has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the Thematic Excellence Programme TKP2020-NKA-06 (National Challenges Subprogramme) funding scheme.



(a) Wrong velocity vectors is estimated when the feature detector recognizes corners which located on another moving vehicle.



(b) Wrong velocity vectors is estimated when not enough corners are detected due to lack of street lines.



(c) Wrong feature translation vectors are calculated due to road elevation.

Fig. 9: Limitation examples of the proposed method. **Left.** Original images with displayed speed in the bottom-left corner and visualized velocity vector on the right. **Right.** IPM image with tracked features. Best viewed in color.

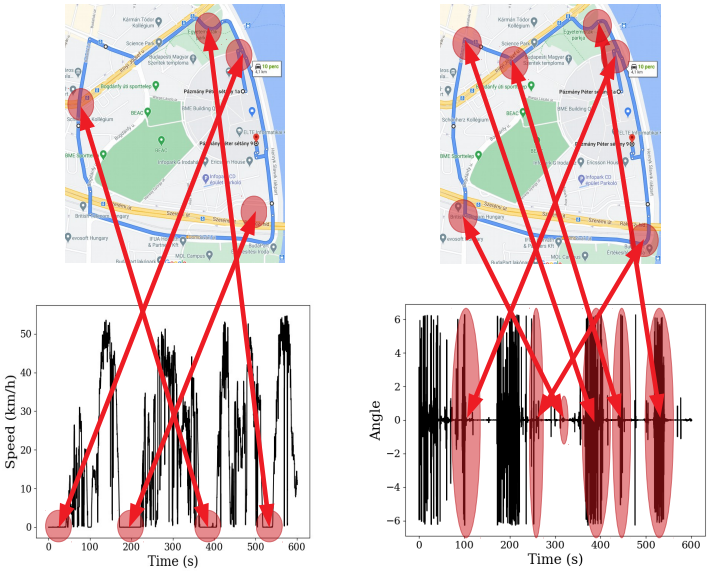


Fig. 10: **Top.** Trajectory of the vehicle during the video. **Bottom-Left.** Magnitude of the speed w.r.t. time. **Bottom-Right.** Angle of the speed w.r.t. time. Interesting results are highlighted by red bubbles: the stops (left) and turnings (right) can be easily detected on the speed and angle charts, respectively.

Bibliography

[1] Augusto Luis Ballardini, Daniele Cattaneo, Simone Fontana, and Domenico Giorgio Sorrenti. An online probabilistic road intersection detector. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, pages 239–246, 2017.

[2] J.-Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker. 1999.

[3] Julie Dequaire, Peter Ondruska, Dushyant Rao, Dominic Zeng Wang, and Ingmar Posner. Deep tracking in the wild: End-to-end tracking using recurrent neural networks. *Int. J. Robotics Res.*, 37(4-5):492–512, 2018.

[4] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In Josef Bigun and Tomas Gustavsson, editors, *Image Analysis*, pages 363–370, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[5] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. 24(6):381–395, 1981.

[6] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[7] Jianbo Shi and Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[8] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[9] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B. Choy, Philip H. S. Torr, and Manmohan Chandraker. DESIRE: distant future prediction in dynamic scenes with interacting agents. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2165–2174, 2017.

[10] Bonolo Mathibela, Paul Newman, and Ingmar Posner. Reading the road: Road marking classification and interpretation. *IEEE Trans. Intell. Transp. Syst.*, 16(4):2072–2081, 2015.

[11] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: an instance segmentation approach. In *2018 IEEE Intelligent Vehicles Symposium, IV 2018, Changshu, Suzhou, China, June 26-30, 2018*, pages 286–291, 2018.

[12] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE Robotics Autom. Mag.*, 18(4):80–92, 2011.

[13] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. A flexible technique for accurate omnidirectional camera calibration and structure from motion. In *2006 IEEE International Conference on Computer Vision Systems*, page 45, 2006.

[14] Nicolas Simond and Michel Parent. Obstacle detection from IPM and superhomography. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 29 - November 2, 2007, Sheraton Hotel and Marina, San Diego, California, USA*, pages 4283–4288, 2007.

[15] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.