Immediate Vehicle Movement Estimation and 3D Reconstruction for Mono Cameras by Utilizing Epipolar Geometry and Direction Prior

Zoltan Rozsa, *Member, IEEE*, Marcell Golarits, and Tamas Sziranyi, *Senior Member, IEEE*

***Abstract*— Motion estimation of surrounding objects is indispensable to any mobile machinery. The paper proposes a method to solve the estimation and reconstruction problem of dynamic objects with a mono camera. Using the relative camera motion and detected rigidly moving objects on the image, we estimate their movement up to a scale factor. Utilization priors about their moving direction are used to estimate the transformation, which maps the 3D object from the previous frame to the actual one. Our two-frame method works twice the speed or more as other methods using three frames or more for the estimation, and we do this without any constraints. We evaluate our method on various traffic scenarios of different autonomous driving datasets.**

***Index Terms*— Vehicle trajectory, 3D reconstruction, mono camera, intelligent transportation.**

I. Introduction

ESTIMATING relative camera motion is a high research interest in many machine vision fields. UAVs, mobile robots, automated guided vehicles, autonomous cars, etc., require the knowledge of vehicle ego-motion to navigate. Methodologies like Simultaneous Localization and Mapping (SLAM) or Structure from Motion (SfM) offer the possibility of reconstructing the static environment besides determining the camera motion. With other simple sensors on-board (e.g., incremental encoder) or based on simple assumptions (identifying objects, knowing camera height), the vehicle displacement's absolute scale can be determined. However, moving objects are generally ignored; the reconstructions are focused on the static scene; the image points of the dynamic objects are treated as outliers. Although other dynamic participants of the transportation system can pose a higher threat than static objects. Pedestrians are vulnerable [1], and other vehicles can move at high speed. When these objects move toward us, a quick reaction is needed for safety [2]. Techniques are needed that are capable of operating at the moment the dynamic object appears [3]. While other methods require a series of frames to establish a decision, our method can already give a reliable estimation of the object motion and shape from only two frames (so two times faster than estimating from three; also, calculation time is negligible), which can be critical in braking distance. Our method can also be used as an initial estimation, supporting future trajectory estimation methods. There are end-to-end deep learning-based methods for motion prediction of multiple moving objects like [4] and [5] which are capable of predicting that. However, they model the vehicles as 2D objects. Considering real extensions of vehicles (instead of simplified models) is vital for passenger safety; that is why we reconstruct not just the trajectories but the objects in 3D as well.

Movement of dynamic objects (so-called eoru-motions: motions of multiple moving objects not rigid with the scene [7]) can also be reconstructed up to a scale. The scale ratio between this and the reconstruction of the environment is unknown, thus the object's distance and velocity. In the paper, we propose a method to calculate the relative scale ratio and the displacement of moving objects at the environment reconstruction scale. In real-life scenarios, we utilize GPS data to estimate the metric scale. We perform this by estimating the essential matrix of a virtual camera pair corresponding to the target object's movement, reconstructing the motion at a selected scale, and estimating the movement's homogeneous transformation as a matrix maps the object center translation parallel to a given direction. Our assumption is true for most vehicle movements (because of straight road segments), and our tests show that it can result in a satisfactory solution in other cases as well as we only need two applicable frames to calculate the proper scale for the whole trajectory. This assumption makes it possible to determine the transformation matrix between frames on the absolute scale and achieve state-of-the-art accuracy in mono camera-based trajectory estimation and 3D reconstruction. Fig. 1 shows that the accuracy of our 3D reconstruction using only two camera frames is comparable to LIDAR measurements.

A. Contributions

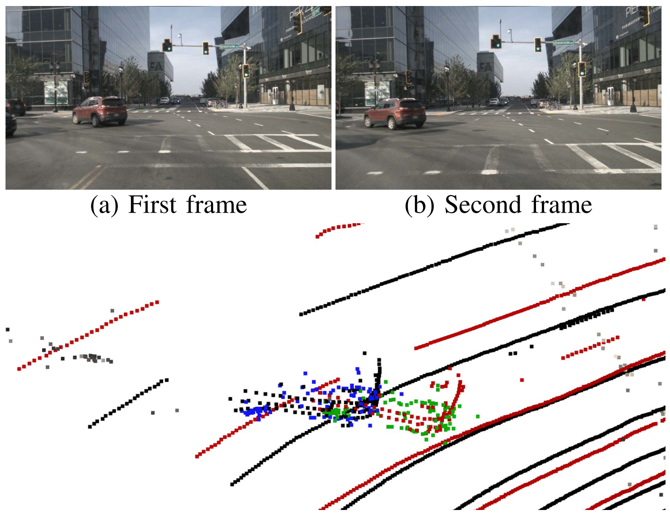
The paper contributes to the following:

- New methodology is proposed to estimate the transformation matrix of the motion of moving objects not rigid with the scene (based on an image pair of a mono camera) at the scale of background reconstruction.

Manuscript received 11 July 2021; revised 27 January 2022 and 13 June 2022; accepted 20 July 2022. Date of publication 26 August 2022; date of current version 5 December 2022. This work was supported in part by the European Union within the framework of the National Laboratory for Autonomous Systems under Grant RRF-2.3.1-21-2022-00002 and in part by the Hungarian Scientific Research Fund (NKFIH OTKA) under Grant K 139485. The Associate Editor for this article was S. E. Li. *(Corresponding author: Zoltan Rozsa.)*

The authors are with the Machine Perception Research Laboratory, Institute for Computer Science and Control (SZTAKI), Eötvös Loránd Research Network (ELKH), 1111 Budapest, Hungary, and also with the Faculty of Transportation Engineering and Vehicle Engineering, Budapest University of Technology and Economics (BME KJK), 1111 Budapest, Hungary (e-mail: rozsazoltan@sztaki.hu; sziranyi.tamas@sztaki.hu).

Digital Object Identifier 10.1109/TITS.2022.3199046



(c) Estimated positions and positions in ground-truth LIDAR cloud

Fig. 1. Example of reconstructed moving object position based on two frames (with moving camera). 1a and 1b show two consecutive frames of Nuscenes dataset [6]. 1c shows our reconstructions together with LIDAR clouds. Color: Red - first LIDAR frame, Black - second LIDAR frame, Green - first frame's reconstruction, Blue - second frame's reconstruction.

- It extends the state-of-the-art because this method can be applied in general 3D scenarios where earlier estimation algorithms (like [8] [9]) cannot.
- Utilizing only two frames for the solution reduces the initial estimation time significantly
- Efficient solution is proposed for the degenerate cases.

B. Outline of the Paper

The paper is organized as follows: Section II surveys the literature about the related topics. Section III describes the proposed method and the concept in detail. Sections IV and V show our test results. In Section VI we propose a complement to our method to deal with its limitations. Section VII continues with discussion. Finally, Section VIII draws some conclusions.

II. Related Works

3D environment reconstruction is achievable from camera images by using SfM or SLAM [10] models, but it is more challenging for dynamic scenes. Reference [11] supports us with a great survey about the topic. There are a few solutions for multi-body non-rigid SfM [12], and some more for rigid multi-body SfM. However, multi-body SfM causes problems in practice [13], like regular bundle adjustment. In most cases, it is solved by factorization [7]. Trajectory triangulation of points moving along a line was first solved in [14] by using Plucker representation. Later [15] proposed a solution to reconstruct the trajectory of points with more general movement by utilizing a linear combination of trajectory basis vectors.

Estimation of relative scale to the background is one of the main issues in reconstructing trajectories of objects not rigid with the scene. As in general, these trajectories can be determined only up to the one-parameter family. Additional constraints are required to obtain the correct scale. Constraints can arise from the non-accidentalness principle, as in the case of [16] or deep learning for depth estimation, using object shapes template can be applied as well. [9] Motion models [8] can be applied as a constraint like a constant velocity model like in the case of [9]. The most common assumption is the planar motion of objects, relating the movement on the ground by being perpendicular to the normal vector of the ground plane [17] and the distance of object points to the ground plane being constant [18]. However, this calculation cannot be executed with a consecutive frame pair of a mono camera in most cases, even in case of constraints. The methods above require multiple frames for the estimation, the knowledge of the ground plane's orientation, and special camera or object motion. Real-time ground plane estimation from a mono camera is hardly applicable (because of slopes, it requires computationally expensive dense reconstruction). Figure 1c shows how few road points can be reconstructed (gray points). Also, methods relying on that require specific camera motion, making it inapplicable to driving scenarios. Compared to the above, the present paper brings the following advantages:

- Only two frames are required for the displacement estimation;
- the initial estimation can be refined with each frame;
- no specific camera motion is assumed;
- 2D motion of moving objects is not a requirement; objects can move through 3D space (e.g., uphill);
- only visual odometry is required, reconstruction is not necessary;
- does not require learning or object templates;
- orientation of the ground plane is not needed.

III. The Proposed Method

Here, we describe the method in detail. We divide our pipeline of scale and transformation estimation into four steps.

1) Preliminary data generation
2) Estimation of virtual camera motion and reconstruction without unknown scale
3) Estimate movement direction
4) Compute the relative scale

These steps are described in the following subsections.

In Figure 2, we illustrate the transformation matrices and coordinate system we use. (Symbols not indicated in the figure caption are defined in the following subsections.) Three coordinate systems are illustrated in the figure. Two of them are the camera coordinate system at different positions, with C_1 and C_2 camera centers. The third (global) one for this illustration is fixed to the traffic pole.

A. Preliminary Data Generation

To generate the input data to our algorithm, we propose to execute three preprocessing sub-tasks. First, the cameras' intrinsic parameters should be determined [19]. The intrinsic

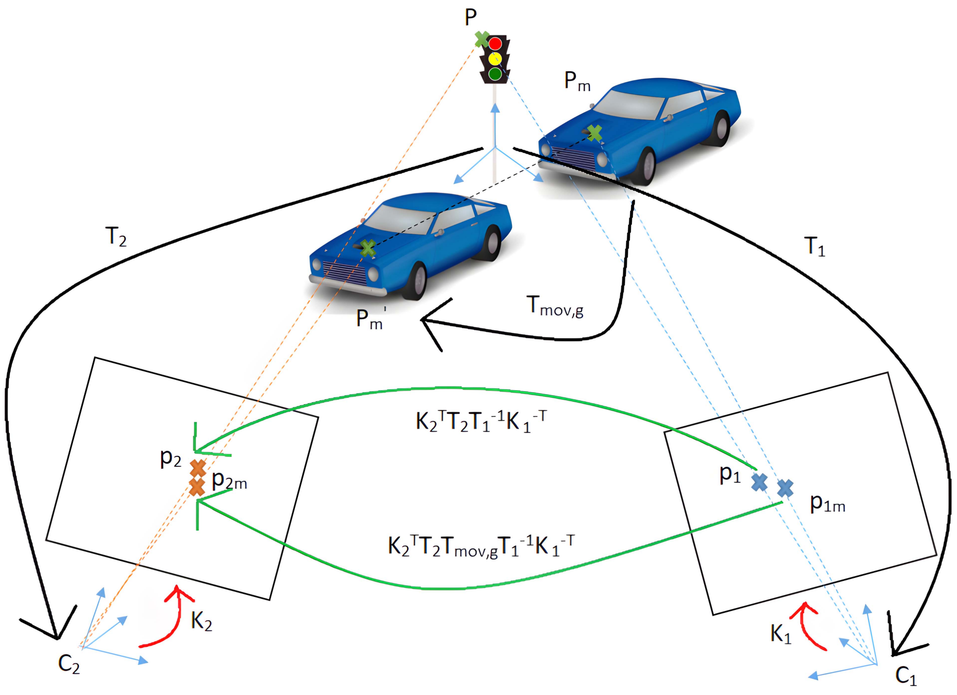


Fig. 2. Illustration of the transformations in our case. The blue car represents the moving object (the traffic pole is a static one) inspected from two camera positions. P is a static point, while P_m, and $P_m^{|}$ corresponds to the same moving 3D point defined in the global coordinate system. $T_{mov,g}$ describes the transformation of the moving car (and P_m on it).

matrix of a camera is constructed from these parameters and maps the 3D point (P) coordinates (in the camera coordinate system) to its 2D projections on the image plane (p):

$$p = K^T P \tag{1}$$

where K is the intrinsic camera matrix.

Then, the relative camera poses should be estimated. In our real-life experiments we used GPS-based poses, but without other sensors, it can also be estimated utilizing only the mono camera with an SfM software like COLMAP [20] for accuracy or SLAM method like ORB [21] for the real-time run. (We used these in the case of virtual camera pose estimation.) In the latter case, the vehicle trajectory is determined on the scale of ego-motion trajectory. Finally, the moving object must be detected and matched through consecutive frames. In our case Yolo_v2 [22] is used for detection and Hungarian algorithm for [23] tracking. The methods of the current subsection are not an essential part of our proposed method (e.g., [24] provides an excellent survey for detection alternatives). However, real-life experiments showed that the above methods provide good performance to provide input data to our estimation.

B. Estimation of Virtual Camera Motion and up to Scale Reconstruction

We can estimate a virtual camera motion between consecutive frames corresponding to the tracked objects, assuming that the vehicle is static and only the camera (on the ego vehicle) is moving. Later on, knowing the ego-motion, the motion of the tracked object can be calculated from this virtual camera motion.

By decomposing the estimated essential matrix [25] corresponding to the moving object, we can define the projection matrices of virtual camera poses (denoted with υ index):

$$P_{\upsilon,1} = K_1^T T_{\upsilon,1} = K_1^T [R_{\upsilon,1}|\lambda_1 t_{\upsilon,1}] \tag{2}$$
$$P_{\upsilon,2} = K_2^T T_{\upsilon,2} = K_2^T [R_{\upsilon,2}|\lambda_1 t_{\upsilon,2}] \tag{3}$$

where $T = [R|t]$ are homogenous transformation matrices with R rotation matrices and t translation vectors. υ index means virtual camera poses, and numbering indicates the given frame. We determine the reconstruction up to a scale factor by defining λ_1 as the unknown scale. In our tests, we used COLMAP [26] to reconstruct the dynamic objects by estimating the motion of a virtual camera pair. Different image features [27] and robust estimators like [28] can be used for this estimation (customized to the needs), just in the case of the real camera motion estimation.

C. Estimate Movement Direction

1) Coordinate Transformation: The virtual camera motion determines the object motion and shape at a given scale (e.g., $\lambda_1 = 1$) and the 3D point cloud shape. However, we would like to determine the trajectory and object shape on an absolute scale. We will utilize the fact that the coordinate systems of the virtual cameras and the real ones of the static scene's reconstruction are the same.

The previously triangulated point cloud can be transformed to the coordinate systems of the virtual cameras by the previously determined $T_{\upsilon,i}$ transformations. We apply a statistical outlier removal algorithm to denoise the triangulated object cloud. This method is thresholding based on the average

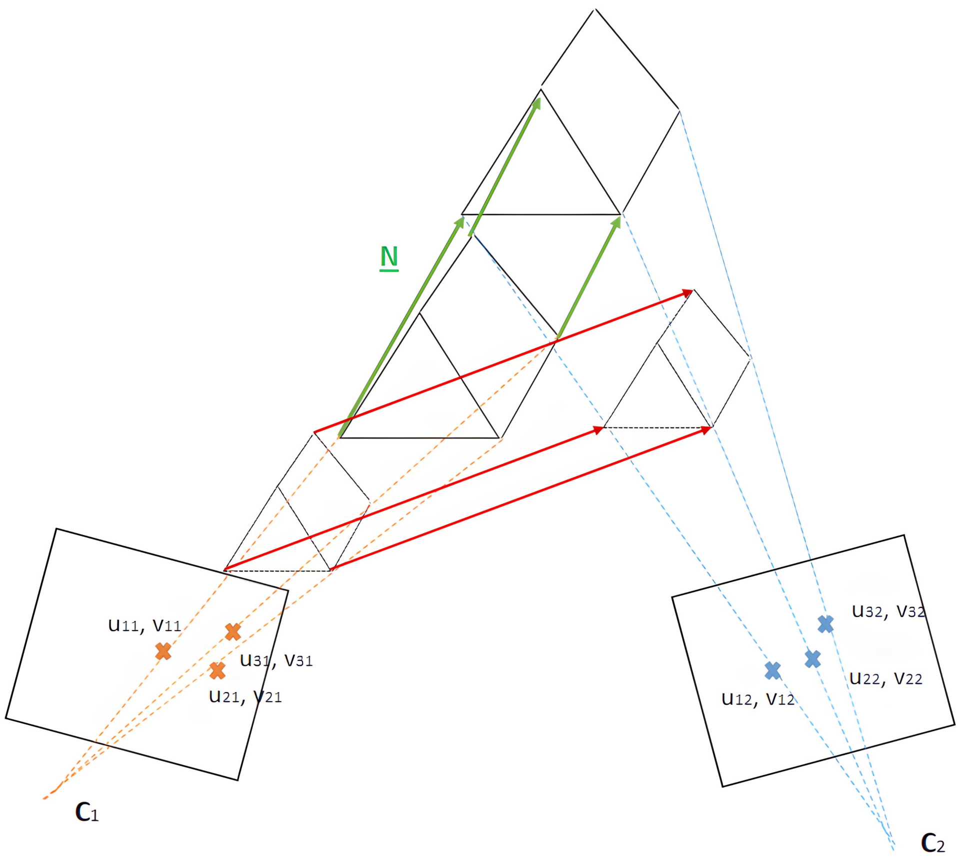


Fig. 3. Illustration of the object motion estimation with known point shape and relative motion direction. Dashed lines represent the projection directions. Crosses indicate the projected points with u and v coordinates (the first indices refer to point numbering, and the second ones indicate the image number).

distances in the neighboring points [29]. After that, by using T_i transformations which maps the global coordinate system object points before (P_m) and after the movement (P'_m) to the i^{th} camera coordinate system can be described as a function of λ_1:

$$P_m = T_1^{-1}[\lambda_1 X_1 \quad \lambda_1 Y_1 \quad \lambda_1 Z_1 \quad 1]^T \tag{4}$$

$$P'_m = T_2^{-1}[\lambda_1 X_2 \quad \lambda_1 Y_2 \quad \lambda_1 Z_2 \quad 1]^T \tag{5}$$

where X_i, Y_i and Z_i are point coordinates of the triangulated 3D points in the coordinate systems of virtual cameras, and $i = 1:2$ are the camera indices.

2) Principal Component Analysis (PCA): On the triangulated and transformed point cloud, we apply PCA [30] to determine its oriented bounding box, and we will use the direction in which the point cloud is the most scattered, indicated as $\underline{N}$ as our estimation of the object's center movement direction. The covariance matrix of the point cloud coordinates:

$$\mathbf{C} = \frac{1}{n-1} P_\mu^T P_\mu \tag{6}$$

where n is the number of points of the point cloud P_μ of which P_c center point is transformed to the origin. By applying singular value decomposition on $\mathbf{C}$ we can get the eigenvalues and eigenvectors of the covariance matrix. The direction corresponding to the smallest eigenvalue will be our estimation of $\underline{N}$.

Illustration of how movement direction specifies the correct scale is shown in Figure 3. The moving object in the illustration is a triangular prism. The estimated essential matrix gives the shape, knowing the object only translates (does not scale between views, and here, in this illustration, does not rotate either). Red arrows indicate a possible translation direction. Green arrows indicate the real translation of points in the known direction $\underline{N}$. The movement direction (here, parallel to its longitudinal axis) determines the correct scale. The pair of smaller triangular prisms is a valid solution without knowing the direction, but the red arrows are not parallel to what we are searching. Only choosing the scale of the bigger pair of triangular prisms results in the searched direction $\underline{N}$.

D. Compute the Relative Scale

Here, we use the assumption that the object (or at least its centroid) goes through a transformation that can be approximated as a translation parallel to its longitudinal axis. Thus the difference vector of the two center points (in the global coordinate system) is equal to the unit vector of the movement direction multiplied with an unknown scale λ_2 of the displacement:

$$P_c^| - P_c = \lambda_2 \underline{N} \tag{7}$$

utilizing the known transformations (from the camera coordinate systems to the global one) and the estimated points:

$$T_2^{-1}\lambda_1 P_{c,2} - T_1^{-1}\lambda_1 P_{c,1} = \lambda_2 \underline{N} \tag{8}$$

TABLE I
COMPARISON OF OUR RESULTS ON VEHICLE TRAJECTORY DATASET TO EARLIER METHODS

	Average Scale Ratio Deviation					Average Trajectory error [m]				
Cars	Lancer	Lincoln	Smart	Golf	Van	Lancer	Lincoln	Smart	Golf	Van
Baseline of [18]	0.05	0.07	**0.01**	0.08	0.13	0.42	0.53	**0.25**	0.95	1.68
Proposed by [18]	0.04	**0.04**	0.04	0.06	0.08	0.20	0.23	0.33	0.33	0.47
Our Proposed (geomean)	**0.03**	**0.04**	0.08	0.06	0.05	**0.15**	**0.22**	0.63	0.23	**0.35**
Our Proposed (eq. sys)	0.04	**0.04**	0.09	**0.04**	**0.04**	0.21	0.24	0.71	**0.20**	**0.35**

where $P_{c,1}$ and $P_{c,2}$ are the centroid of the point cloud in the first and second camera coordinate system.

Rearranging the equation, and writing separately the rotation and translation parts of the transformations gives:

$$\left(R_2^{-1}P_{c,2} - R_1^{-1}P_{c,1} \quad \underline{N}\right)\begin{pmatrix}\lambda_1\\ \lambda_2\end{pmatrix} = C_1 - C_2 \tag{9}$$

where C_1 and C_2 are camera centers in the global static coordinate system and their difference is a vector describing the ego-motion.

Let us denote $\lambda = \begin{pmatrix}\lambda_1\\ \lambda_2\end{pmatrix}$, $b = C_1 - C_2$, and $A = (R_2^{-1}P_{c,2} - R_1^{-1}P_{c,1} \quad \underline{N})$ the coefficient matrix. The equation systems above (two unknowns with three equations) can be solved in a least-squares sense:

$$\lambda = (A^T A)^{-1} A^T b \tag{10}$$

By substituting λ_1, the searched scale to $T_{v,1}$ and $T_{v,2}$ we get the transformation matrix of the moving object in the global coordinate system:

$$T_{mov,g} = T_2^{-1} T_{v,2} T_{v,1}^{-1} T_1 \tag{11}$$

Finally, we got the object motion's transformation matrix. Moreover, we also get the 3D reconstruction of the moving object on the correct scale.

IV. VALIDATION

We evaluated the proposed method in the Vehicle Trajectory dataset [18], which is the only demanding dataset currently available for vehicle trajectory estimation and object reconstruction with a moving mono camera to the best of our knowledge.

The Vehicle Trajectory dataset is a virtual dataset containing seven scenes (each with one trajectory) and five cars. Ground truth object masks are published with the dataset, so trajectory estimation can be measured without preprocessing influencing the results. The dataset is designed for vehicle trajectory estimation for mono and stereo cameras. As we use mono cameras, left images are used in our case. The seventh scene (Bumpy road) is an exception as it operates just with object motion parallel to the camera (degenerate case to our algorithm). In this case, we used left and right camera images alternately (in different time steps) to evaluate the whole dataset (note: only two frames without degeneracy would be enough to estimate the relative scale). We used scale ratios (calculated from frame pairs) with degeneracy degree smaller than 0.75 (judged as trustworthy, see in Fig. 10) for the one global scale ratio estimation. Degeneracy degree is defined absolute value of the scalar product of unit vectors in the direction of the estimated object and camera motion (see more in Section VII-A.1). However, in the case of some trajectories, there were just degenerate consecutive frame pairs based on the defined threshold. In this case, we used the available frame pairs and estimations. We estimated a global scale in two different ways for the whole trajectory. In the first case, we used the geometric mean of the calculated scale ratios (from consecutive frame pairs), and in the second case, we constructed a linear equation system from the equations of the frame pairs (Eq. 9):

$$\begin{pmatrix} N_{g,i} & \underline{N_i} & 0 & \cdots & 0 \\ N_{g,i+1} & 0 & \underline{N_{i+1}} & \cdots & 0 \\ & & \vdots & & \\ N_{g,n} & 0 & 0 & \cdots & \underline{N_n} \end{pmatrix}\begin{pmatrix}\lambda_1\\ \lambda_i\\ \lambda_{i+1}\\ \vdots\\ \lambda_n\end{pmatrix} = \begin{pmatrix} C_{i-1} - C_i\\ C_i - C_{i+1}\\ \vdots\\ C_{n-1} - C_n\end{pmatrix} \tag{12}$$

where $i = 2..n$, n is the number of frames and $R_i^{-1}P_{c,i} - R_{i-1}^{-1}P_{c,i-1}$ denoted as $N_{g,i}$.

We solve the equation system in the least-squares sense (like in Eq. 10) and we used the estimated λ_1 scale as the global scale relating the object trajectory to the background reconstruction.

Table I shows the average performance measure values are compared to [18]. Reference [18] used the whole image sequences to select view pairs of specific conditions to estimate the scale ratio. We used only consecutive view pairs for the estimation. For the global scale estimation, we used a geometric mean of the estimated ratios (geomean) or estimated only one global scale ratio by gathering the equations of consecutive frame pairs (eq. sys). We estimated the average scale ratio at least as accurately as the baseline for four out of the five cars. The trajectory errors calculated by the proposed estimation are shown in Figure 4 for all the 35 trajectories. Comparing Figure 4 to a similar figure published in [18] shows that we outperformed the baseline in most of the cases. For both performance measures, we outperformed [18]'s proposal except only the case of the Smart car, which the fact can explain, the car has a small length/width ratio, so the longitudinal axis of the car is harder to estimate. Example reconstructed trajectory is illustrated in Fig 5. (The trajectory consists of 60 frames, every 10^{th} and the 1^{st} frame is illustrated.)

V. REAL-LIFE TESTS

For the real-life testing of the proposed method, we chose the Argoverse dataset [31]. From the dataset, we selected

TABLE II
DESCRIPTION OF TEST TRAJECTORIES FROM ARGOVERSE DATASET [31]

Sequence id	Sequence id (our)	Object id (our)	traffic scenario	Length of sequence [s]	Mean distance of vehicles [m]
10f92308-e06e-3725-a302-4b09e6e790ad	0	70	target vehicle crosses the ego vehicle lane from left to right	1.3	41.36
84c35ea7-1a99-3a0c-a3ea-c5915d68acbc	1	13	target vehicle crosses from left to right while ego vehicle turns right	4.3	12.84
99c45b6e-6fc7-39b8-80d7-727c485fb561	2	10	target vehicle turns left from left	1.6	36.11
a073e840-6319-3f0b-843e-f6dccdcc7b77	3	8	target vehicle turns right from the oncoming lane	4.0	34.78
ba067318-0d89-34b5-b577-b171b1a4212b	4	1	target vehicle goes parallel lane in successive bends	5.0	15.92
e8ce69b2-36ab-38e8-87a4-b9e20fee7fd2	5	0	target vehicle turns right from oncoming lane while ego vehicle turns left	4.3	14.25
f0826a9f-f46e-3c27-97af-87a77f7899cd	6	15	target vehicle turns right from the ego vehicle lane	1.8	23.71
fa0b626f-03df-35a0-8447-021088814b8b	7	33	target vehicle joins the ego vehicle lane	2.4	24.2

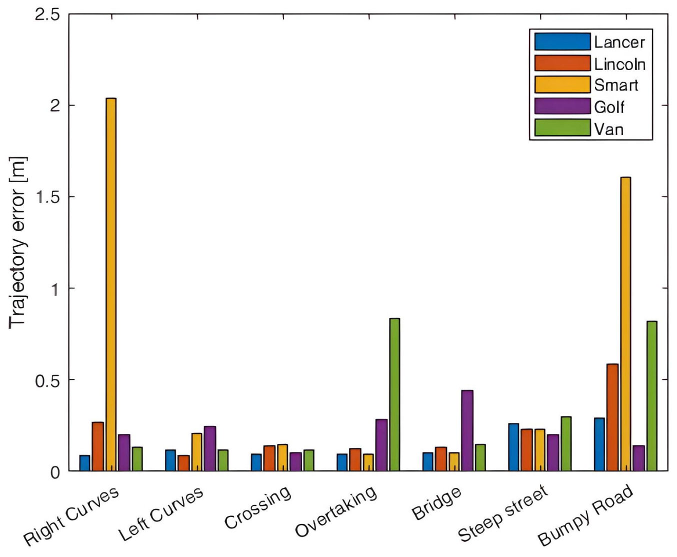


Fig. 4. Trajectory error of different vehicles and scenes of the Vehicle Trajectory dataset [18] with our proposed method. On the x-axis, different scenes are listed, while different colors indicate different cars in these scenes.

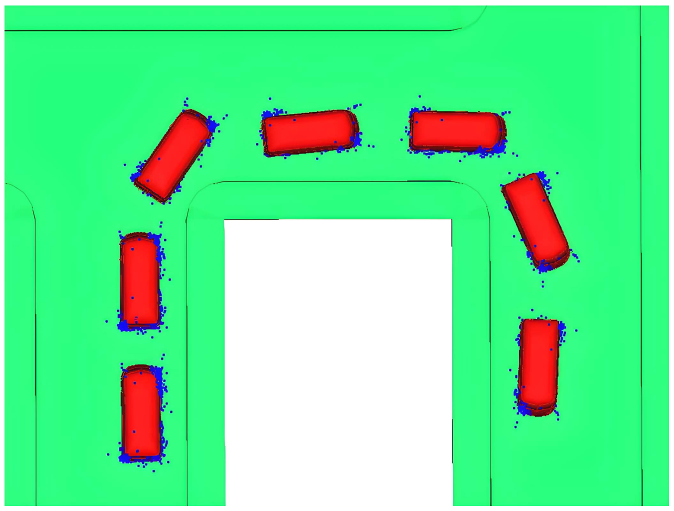


Fig. 5. Reconstructed trajectory and point cloud of VW van of Vehicle Trajectory dataset [18] in case of Right Curves scene. Colormap: Green - ground truth ground, Red - ground truth object poses, Blue - Reconstructed vehicle points of the given position.

8 vehicle trajectories corresponding to different traffic situations and vehicle movements. These data are summarized in Table II with some statistics. For the simpler referencing, we assigned an integer number to identify the sequence and objects instead of the original Argoverse id, and this is indicated as 'our' in the table.

Results of the trajectory estimation and reconstruction can be found in Table III. **Note**: utilizing camera frames with high frequency can make the estimation even more accurate, but in these test scenarios, only 10 Hz sampling rate was used because of the LIDAR ground truths and the analysis provided in Section VII-C.

Our results were compared to the 'lift and splat' part of [32] as it is a deep-learning-based method designed and utilized by state-of-the-art methods like [5] for trajectory prediction. We used [32] for comparison because the part of the network can be considered to have a similar purpose (localizing surrounding vehicles) to ours. However, some significant differences should be discussed. Although the lift part of the network estimates 3D from the images (in very low resolution), at the end of the splat part, only 2D information (in bird's-eye view) is available in the form of a probability map about the surrounding vehicles with a simplified model. We provide 3D information about the vehicles with the actual shape, given positions, and data points.

The depth accuracy of our two frames-based estimations outperformed [32] on average and in most of the sequences, too; utilizing more than two frames resulted in an even more significant performance increase. Besides, the segmentation result of [32] is often a box estimation parallel or perpendicular to the ego-vehicle. Inbetween angles, it often cannot estimate the correct pose of the vehicles, unlike our geometry-based estimation, as examples in Fig. 6 demonstrate to us.

TABLE III
RESULTS OF TEST TRAJECTORIES FROM ARGOVERSE DATASET [31]

Sequence id (our)	0	1	2	3	4	5	6	7	Average
Depth error of lift and splat of [32] [%]	33.21	19.59	**6.07**	6.63	8.80	22.25	16.61	27.42	17.57
Depth error proposed (first frame pair) [%]	7.00	4.85	18.21	20.59	16.83	19.36	16.08	26.51	16.18
Depth error proposed (geomean) [%]	1.29	8.39	18.21	12.90	11.21	**14.82**	14.90	**7.68**	11.18
Depth error proposed (eq. sys) [%]	**0.91**	**2.95**	18.21	**1.53**	**1.36**	17.42	**11.19**	10.12	**7.96**

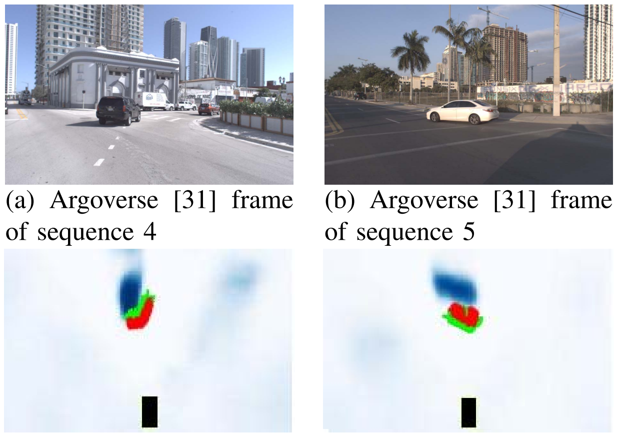

(a) Argoverse [31] frame of sequence 4
(b) Argoverse [31] frame of sequence 5
(c) Estimation of [32] and our proposed method for the frame above
(d) Estimation of [32] and our proposed method for the frame above

Fig. 6. Example of vehicle estimations on Argoverse dataset [6]. 6a and 6b show the original images. 6c and 6d are the estimation results. Color: Black - Ego vehicle, Green - (LIDAR based) ground truth position, Blue - [32]'s segmentation result, Red - our position estimation.

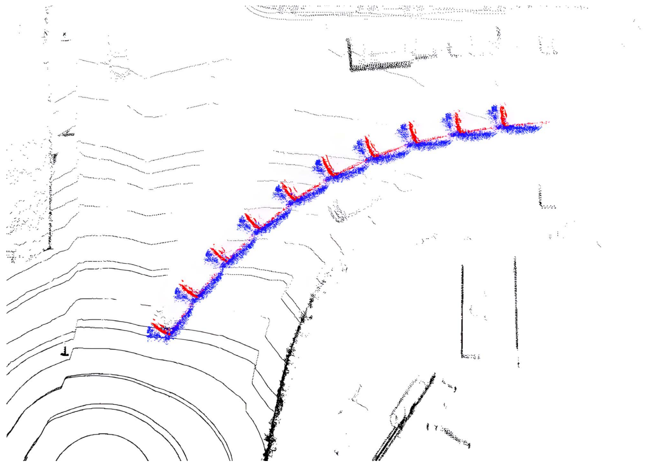


Fig. 7. Reconstructed trajectory of Argoverse dataset [31] in case of sequence 4. Colormap: Black - Lidar frame (just for reference), Red - (LIDAR based) ground truth object positions (just for reference), Blue - Reconstructed vehicle points from the camera of the given position.

The average depth error of our method in different challenging sequences is **7.96** %. This is the same level of accuracy as reported results by [34] and somewhat below than reported by [35] and [9] in similar real-life data, but with less challenging scenarios (only straight movement). Also, not relying upon constant motion assumption as [34] our method can solve more generic and challenging trajectories (e.g., the Vehicle trajectory dataset of the previous section). See more comparisons to these methods in section VI. A reconstructed trajectory of the dataset is illustrated in Fig. 7. (Every 15^{th} frame is illustrated.)

VI. COMPLEMENTARY METHOD

As previously mentioned, parallel camera and object movement results in degenerate cases for our method (see more in section VII-A.1). On the one hand, we find that it is a very frequent movement type, but also the least dangerous one (e.g., ADAS systems like adaptive cruise control are already handling parallel moving vehicles in front of us); most hazardous scenarios happen at vehicle path intersections. On the other hand, only one frame-pair with non-degenerate movement is sufficient for estimating the whole trajectory; also, our method is robust against cases close to degeneracy (Section VII-A.2). Finally, these parallel movement driving scenarios are not even challenging in terms of camera-based depth estimation. There are some methods like [34] or [9], which demonstrate that they are capable of vehicle trajectory estimation in this simple case of approximately parallel movement of cars (in the case of 9 trajectories of the KITTI dataset), so these can be used. Still, we propose a straightforward solution to deal with these degenerate cases, using mostly the same pipeline as we have used so far. We suggest altering it after $\underline{N}$ is computed in Section III-C (and so degeneracy degree is known). From there, we can see that the target vehicle is moving approximately parallel to the ego vehicle. In this case, we can apply a simplified scale estimation because the target vehicle is parallel to our vehicle (and so their rear or front part is approximated as a plane). Based on the knowledge of camera installation position (height in absolute scale), we can use homography to estimate the vehicle distance (depth) on the ground. Steps of the scale estimation in these cases:

- Estimate the distance (D_{plane}) of the car's (rear or front) plane perpendicular to our moving direction from the reconstructed point cloud. We can use the prior of its normal vector orientation (in the case of the KITTI dataset with roll and pitch angles being approximately 0, this orientation is approximately [0 0 1] in the camera coordinate system). We used RANSAC to estimate this plane of the target object [28].
- By projecting the reconstructed object cloud to a given image, we select the points with the highest v coordinate (closest to the ground plane) inside the previously determined bounding box of the object.
- We assume that the previously determined points are on the ground plane, and we use projective transformation determined from the known camera installation position to estimate the distance of these ground points (D_{point}).

TABLE IV
RESULTS OF TEST TRAJECTORIES FROM KITTI RAW DATASET [33] FOR DEGENERATE CASES

Sequence id		04				47				56	Average
Object id		1	2	3	6	0	4	9	12	0	
Depth error[%]	[34]	4.1	6.8	5.3	7.3	9.6	11.4	7.1	10.5	5.5	7.5
	[35]	6.0	**5.6**	4.9	5.9	**5.9**	12.5	7.0	8.2	6.0	6.9
	[9]	5.1	11.1	**1.8**	**2.5**	9.3	3.8	6.6	**2.8**	**0.8**	**4.9**
	proposed	**3.0**	10.9	2.3	8.2	14.0	**2.9**	**2.7**	10.1	2.5	6.3

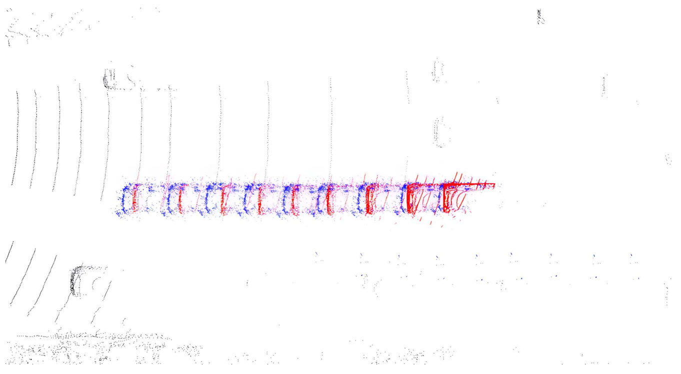


Fig. 8. Reconstructed trajectory of KITTI raw dataset [31] in case of sequence id 47 and object id 6. Colormap: Black - Lidar frame (just for reference), Red - (LIDAR based) ground truth object positions (just for reference), Blue - Reconstructed vehicle points from camera of the given position.

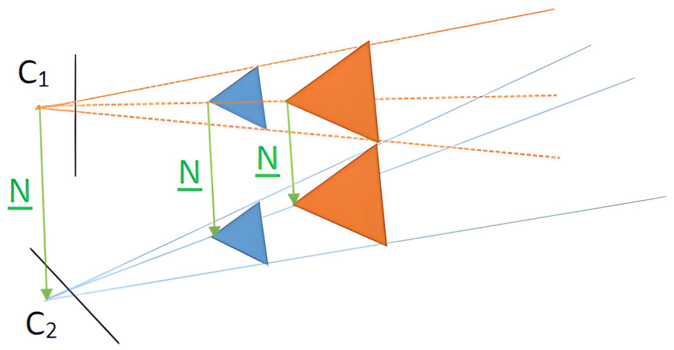


Fig. 9. 2D Illustration of the degenerate case of object motion estimation with known relative motion direction.

- The relative scale which relates the target vehicle to the background can be computed as: $\lambda_1 = \frac{D_{point}}{D_{plane}}$

Results of this estimation can be seen in Table IV and a reconstructed trajectory and object shape are illustrated in Fig. 8. (Every 10th frame is illustrated.)

One can see that using this assumption, we achieved state-of-the-art depth error estimation for half of the benchmark. However, our goal was to show that this benchmark which was frequently used to evaluate trajectory estimation is not challenging. It uses only one type of vehicle movement (and methods evaluated on this cannot be compared to our solution working in general camera and target motion cases). That is why we created a new benchmark for vehicle trajectory estimation by selecting diverse traffic scenarios from the Argoverse dataset (described in Table II and Section V).

VII. DISCUSSION

In this section, the proposed method's well-known limitations and comparison are discussed, and finally, a running time analysis is provided.

A. Degeneracy

In this subsection, first, degenerate cases are derived mathematically, then robustness against degeneracy is investigated.

1) Degenerate Case: In case of translation of the camera has the same direction as the translation of the moving object, scale of the moving object translation cannot be estimated. Substituting this condition, $\frac{C_2-C_1}{|C_2-C_1|} = N$ to Equation 9, and rearranging it, gives:

$$\lambda_1 N_g = -(\lambda_2 + |C_2 - C_1|)N \tag{13}$$

so N_g must be a multiplication of N as well. This results in a coefficient matrix of rank equal to one, or a scalar equation:

$$\lambda_1 = -\lambda_2 - |C_2 - C_1| \tag{14}$$

which naturally cannot be solved for the two unknowns. Figure 9 illustrates this degenerate case. If the motion $\underline{N}$, is parallel to the camera motion, both triangles (blue and orange) can be the solution. The scale remains ambiguous.

2) Degeneracy in the Validation Data (Robustness Against Degeneracy): In the validation dataset, as we have said in the Bumpy Road scene, the camera and object trajectory (from our method's point of view) was completely degenerate. However, other parts of the dataset contained degenerate or nearly degenerate cases too. Figure 10 illustrates how the proposed method behaves as a function of a degeneracy. We measure the degree of degeneracy as the absolute value of the scalar product of unit vectors in the direction $C_2 - C_1$ and N (0 means they are perpendicular, and 1 indicates parallel). In the figure, the ratio of estimated and ground truth scales ($\frac{\lambda_{1es}}{\lambda_{1gt}}$) in logarithmic scale is visualized as a function of degeneracy.

The illustration corresponds to one car (Golf) through the different trajectories, but all the cars have a very similar data distribution. One can see that there were many consecutive frame pairs in the dataset close to the degeneracy degree 1, and they resulted in erroneous estimation. However, they can be easily filtered out (as we can calculate this factor, which we call the degree of degeneracy). We get reasonable estimations a bit farther from the value 1 (we used 0.75, as mentioned earlier). On the other hand, as in the logarithmic scale, the distribution seems symmetric on the x axis. Calculating a geometric mean (or formulating a global equation system) could result in a correct scale approximation close to degenerate motions (if we have enough measurements)

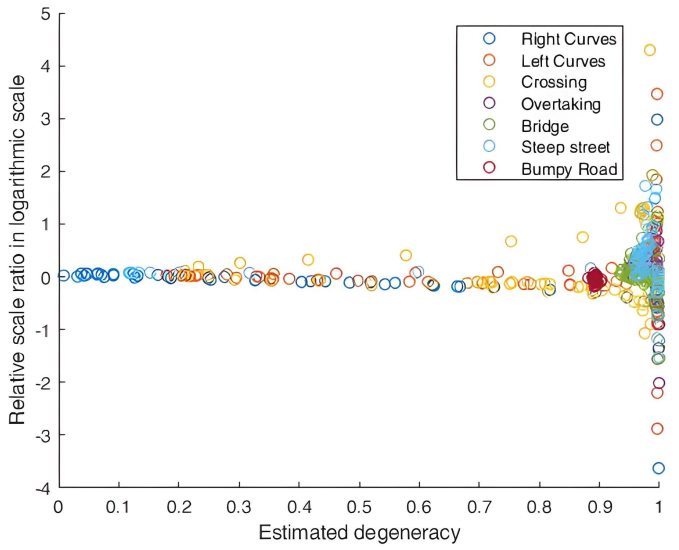


Fig. 10. Distribution of consecutive frame pairs degeneracy (in terms of our estimation method) and accuracy of corresponding estimations in Vehicle Trajectory dataset. Different color circles indicate consecutive frame pairs in different scenes. On the x-axis, as we get closer to the right, the higher the degeneracy gets (related object and camera movement). On the y-axis the closer to 0 means more accurate position estimation.

as well. Considering the dataset's overall degeneracy degree, our method proved to be robust.

Note: The parallel camera motion case cannot be solved by the method proposed by [18] either, but in fact, it has a lot more theoretically degenerate cases, discussed in the following subsection.

B. Earlier Attempt in Degenerate Case

The method of [18] is the only available in the literature, which is capable of achieving similar accuracy as our proposed one (Table I) for general vehicle trajectories (including general camera motion and turning vehicles). However, besides its earlier mentioned limitation (coming from ground plane estimation) its applicability to general driving scenarios (camera and target object are moving in approximately parallel planes) is questionable. They provide no quantitative result on Cityscapes dataset [36] which they use to illustrate their method in these cases. Reference [18] approximate the ground as a plane, and based on plane motion assumption; they aim to solve the relative scale estimation. They utilize the fact that object points should have the same distance to the estimated ground on each frame. Formulating this criterion to any point of the object (here, we use object center) in the case of two frames, we can write the following equation (using the notations of this paper):

$$\underline{n}\left(N_g \quad C_2 - C_1\right)\begin{pmatrix}\lambda_1\\1\end{pmatrix} = 0 \tag{15}$$

where $\underline{\mathrm{n}}$ is the normal vector of the plane. Let us suppose a coordinate frame fixed to one point in the ground plane, oriented so that direction of z axis is the same as its normal vector, pointing upward. In this coordinate frame the equation of the ground plane is $0 = Ax+By+Cz-D = 0x+0y+1z-0$ (A, B and C are normal vector components and D is the offset of the plane) and distance of a point to the ground plane can be calculated simply by the scalar product of its normal vector $\underline{n} = [0, 0, 1]$. In this case only one equation remains for the z coordinate of vector N_g:

$$\lambda_1 N_{g,z} = C_{1,z} - C_{2,z} \tag{16}$$

where z index means the z coordinate of a vector. As it can be seen from Eq. 16 if there is no difference between the z coordinates of the camera position (moving parallel to the ground), the relative scale λ_1 cannot be determined. As the height difference of a camera (installed on a car) between consecutive frames is minimal (approximately 0), any general driving scenario could mean (close to) degenerate case to [18]. Numerous frames can be required for a proper estimation. They provide only qualitative result on two trajectories of this type of data (Cityscapes [36]), which can be reproduced uncertainly based on their description. In this way, we cannot compare their performance to ours in the case of driving scenarios. However, on their dataset, we outperformed them in most of the cases (Table I).

TABLE V

RUNNING TIME OF PIPELINE COMPONENTS

Average running Time [ms]	Measured in Matlab development config.	Estimation for latest real-time hardware solution
Yolo_v2 detector	96	10
Tracker	21	21
ORB object SLAM	91	25
Proposed scale estimation	2	2
Total	210	58

C. Running Time Analysis

The theoretical speedup is coming from the two frames' sampling instead of three or more, which is an important issue in risky traffic conditions. The present development machine configuration carried out the running time analysis: RAM: 31GB, CPU: Intel® Core i7-7820X CPU @ 3.60GHz × 16, GPU: GeForce GTX 1080 Ti 12GB, Operating system: Ubuntu 18 in Matlab environment. Measured running time values are shown in Table V. The computation depends on the dataset image size. The indicated values correspond to Argoverse dataset with an image size of 1920 × 1200.

The ORB feature based reconstruction runs with bundle adjustment with an average running time of 569 ms. However, it is a background feature not necessarily counted in real-time frequency. In this configuration (with high-resolution images), the system can run about 5 FPS in a not-optimized Matlab environment.

Yolo_v2 detector is reported in [22] to run about 91 FPS (about 10 ms/frame) with 69.0 mAP and ORB-SLAM about 25 ms/frame in [37]. Substituting these values to the running time calculation of the pipeline (and not assuming any speed up in the tracker), would result in about 58 ms total run time (17 FPS), close to the real-time speed of 20 FPS. Summing up in Table V, it is safe to assume that real-time run is provided in a usual on-board embedded system.

VIII. Conclusion

This paper has presented an approach to reconstruct vehicle trajectories and vehicles in 3D with a mono camera, based on direction prior and epipolar geometry. We provide reliable results for eoru-motions by utilizing only two frames. For this, we estimate the relative scale that relates the object and the background. Earlier methods relied on priors like ground estimates, object shape models, and learning while working in the case of specific camera motion. We do not require these or utilize any assumption besides direction prior. Propagation of the scale ratio is possible, as the evaluation shows. We have discussed the limitation of the method with the degenerate case. We can detect these cases and propose a complementary to our method. With this proposal, we have reached state-of-the-art performance results in the problematic cases too. Our method uses only two frames to estimate the pose, speed, and 3D shape of moving objects two times faster (much earlier) than a method using three frames. It is advantageous in an autonomous driving application where prediction speed (especially of a hazardous moving object) is essential. The method's quantitative evaluation supports the theoretical advantages. We estimated trajectories of such general traffic scenarios that have not been aimed before in the literature. It can be used in any general vehicle trajectory estimation scenarios, too, like estimation from a UAV (Vehicle Trajectory dataset [18]). In these cases, we have outperformed the state-of-the-art. Our method has potential extensions as many natural and artificial objects are elongated parallel to their travel direction. Object recognition can extend to other predetermined directions if not the case. We plan to extend the method to estimate other object types' movements in the future.

References

[1] X. Song *et al.*, "Pedestrian trajectory prediction based on deep convolutional LSTM network," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3285–3302, Jun. 2021.

[2] H. Szűcs and J. Hézer, "Road safety analysis of autonomous vehicles: An overview," *Periodica Polytechnica Transp. Eng.*, vol. 50, no. 4, pp. 426–434, Aug. 2022. [Online]. Available: https://pp.bme.hu/tr/article/view/19605

[3] Z. Rozsa and T. Sziranyi, "Object detection from a few LiDAR scanning planes," *IEEE Trans. Intell. Vehicles*, vol. 4, no. 4, pp. 548–560, Dec. 2019.

[4] L. Fang, Q. Jiang, J. Shi, and B. Zhou, "TPNet: Trajectory proposal network for motion prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 6797–6806.

[5] A. Hu *et al.*, "FIERY: Future instance prediction in bird's-eye view from surround monocular cameras," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 15273–15282.

[6] H. Caesar *et al.*, "nuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11621–11631.

[7] R. Sabzevari and D. Scaramuzza, "Multi-body motion estimation from monocular vehicle-mounted camerass," *IEEE Trans. Robot.*, vol. 32, no. 3, pp. 638–651, Jun. 2016.

[8] J. Huang, S. Yang, T.-J. Mu, and S.-M. Hu, "ClusterVO: Clustering moving instances and estimating visual odometry for self and surroundings," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2168–2177.

[9] S. Yang and S. Scherer, "CubeSLAM: Monocular 3-D object SLAM," *IEEE Trans. Robot.*, vol. 35, no. 4, pp. 925–938, Aug. 2019.

[10] Z. Rozsa, M. Golarits, and T. Sziranyi, "Localization of map changes by exploiting SLAM residuals," in *Advanced Concepts for Intelligent Vision Systems*, J. Blanc-Talon, P. Delmas, W. Philips, D. Popescu, and P. Scheunders, Eds. Cham, Switzerland: Springer, 2020, pp. 312–324.

[11] M. R. U. Saputra, A. Markham, and N. Trigoni, "Visual SLAM and structure from motion in dynamic environments: A survey," *ACM Comput. Surv.*, vol. 51, no. 2, pp. 1–36, Mar. 2019, doi: 10.1145/3177853.

[12] S. Kumar, Y. Dai, and H. Li, "Multi-body non-rigid structure-from-motion," in *Proc. 4th Int. Conf. 3D Vis. (3DV)*, Oct. 2016, pp. 148–156.

[13] K. E. Ozden, K. Schindler, and L. Van Gool, "Multibody structure-from-motion in practice," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 6, pp. 1134–1141, Jun. 2010.

[14] S. Avidan and A. Shashua, "Trajectory triangulation of lines: Reconstruction of a 3D point moving along a line from a monocular image sequence," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2, Jun. 1999, pp. 62–66.

[15] H. S. Park, T. Shiratori, I. Matthews, and Y. Sheikh, "3D trajectory reconstruction under perspective projection," *Int. J. Comput. Vis.*, vol. 115, no. 2, pp. 115–135, Nov. 2015.

[16] K. E. Ozden, K. Cornelis, L. Van Eycken, and L. Van Gool, "Reconstructing 3D trajectories of independently moving objects using generic constraints," *Comput. Vis. Image Understand.*, vol. 96, no. 3, pp. 453–471, Dec. 2004.

[17] C. Yuan and G. Medioni, "3D reconstruction of background and objects moving on ground plane viewed from a moving camera," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2, Jun. 2006, pp. 2261–2268.

[18] S. Bullinger, C. Bodensteiner, M. Arens, and R. Stiefelhagen, "3D vehicle trajectory reconstruction in monocular video data using environment structure constraints," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 35–50.

[19] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000.

[20] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4104–4113.

[21] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[22] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6517–6525.

[23] M. L. Miller, H. S. Stone, and I. J. Cox, "Optimizing Murty's ranked assignment method," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 33, no. 3, pp. 851–862, Jul. 1997.

[24] L. Chen, S. Lin, X. Lu, D. Cao, and F. Y. Wang, "Deep neural network based vehicle and pedestrian detection for autonomous driving: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3234–3246, Jun. 2021.

[25] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[26] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 501–518.

[27] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 2564–2571.

[28] P. H. S. Torr and A. Zisserman, "MLESAC: A new robust estimator with application to estimating image geometry," *Comput. Vis. Image Understand.*, vol. 78, no. 1, pp. 138–156, 2000.

[29] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3D point cloud based object maps for household environments," *Robot. Auton. Syst.*, vol. 56, no. 11, pp. 927–941, Nov. 2008, doi: 10.1016/j.robot.2008.08.005.

[30] A. Ilin and T. Raiko, "Practical approaches to principal component analysis in the presence of missing values," *J. Mach. Learn. Res.*, vol. 11, no. 1, pp. 1957–2000, 2010. [Online]. Available: http://jmlr.org/papers/v11/ilin10a.html

[31] M.-F. Chang *et al.*, "Argoverse: 3D tracking and forecasting with rich maps," 2019, *arXiv:1911.02620*.

[32] J. Philion and S. Fidler, "Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3D," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 194–210.

[33] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013.

[34] S. Song and M. Chandraker, "Robust scale estimation in real-time monocular SFM for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1566–1573.

[35] S. Song and M. Chandraker, "Joint SFM and detection cues for monocular 3D localization in road scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3734–3742.

[36] M. Cordts *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3213–3223.

[37] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Jun. 2017.

Zoltan Rozsa (Member, IEEE) received the Ph.D. degree in vehicle and transportation engineering from the Budapest University of Technology and Economics in 2020. He is currently with the Faculty of Transportation Engineering and Vehicle Engineering, Budapest University of Technology and Economics, and a Research Fellow at the Machine Perception Research Laboratory, Institute for Computer Science and Control. His research interests include automated guided vehicles, machine vision, 3D recognition, and reconstruction.

Marcell Golarits received the M.Sc. degree in mechatronics engineering from the Budapest University of Technology and Economics in 2021. He is currently pursuing the Ph.D. degree with the Department of Material Handling and Logistic Systems. He is also a Software Developer at the Machine Perception Research Laboratory, Institute for Computer Science and Control (SZTAKI). His research interests include machine vision, machine learning, and 3D reconstruction.

Tamas Sziranyi (Senior Member, IEEE) received the Ph.D. and D.Sci. degrees from the Hungarian Academy of Sciences, Budapest, in 1991 and 2001, respectively.

He was appointed as a Full Professor in 2001 at Pannon University, Veszprém, Hungary, and in 2004, at Pázmány Péter Catholic University, Budapest. He has been a Research Scientist at the Institute for Computer Science and Control (SZTAKI) since 1992, where he has been leading the Machine Perception Research Laboratory since 2006. He is currently a Full Professor at the Budapest University of Technology and Economics. He has participated in several prestigious international (ESA, EDA, FP6, FP7, and OTKA) projects with his research laboratory. He has more than 310 publications, including 60 in major scientific journals and several international patents. His research activities include machine perception, pattern recognition, texture and motion segmentation, Markov random fields and stochastic optimization, remote sensing, surveillance, intelligent networked sensor systems, graph-based clustering, and digital film restoration. He was the Founder and the Past President (1997–2002) of the Hungarian Image Processing and Pattern Recognition Society. Since 2008, he has been a fellow of the International Association for Pattern Recognition (IAPR) and the Hungarian Academy of Engineering. He has been a member of Hungarian Academy of Sciences since 2022. He was honored by the Master Professor Award in 2001, the ProScientia (Veszprem) Award in 2011, and the Officers Cross by the President of Hungary in 2018. He was an Associate Editor of IEEE TRANSACTIONS ON IMAGE PROCESSING (2003–2009). He has been an Assistant Editor and now an Associate Editor of *Digital Signal Processing* (Elsevier) since 2012.