

# Extracting Driving Behavior: Global Metric Localization from Dashcam Videos in the Wild

Shao-Pin Chang<sup>1</sup>, Jui-Ting Chien<sup>2</sup>, Fu-En Wang<sup>1</sup>, Shang-Da Yang<sup>2</sup>,  
Hwann-Tzong Chen<sup>2</sup>, and Min Sun<sup>1</sup>(✉)

<sup>1</sup> Department of Electrical Engineering, National Tsing Hua University,  
Hsinchu, Taiwan  
s104061554@m104.nthu.edu.tw, s102061149@m102.nthu.edu.tw,  
sunmin@ee.nthu.edu.tw

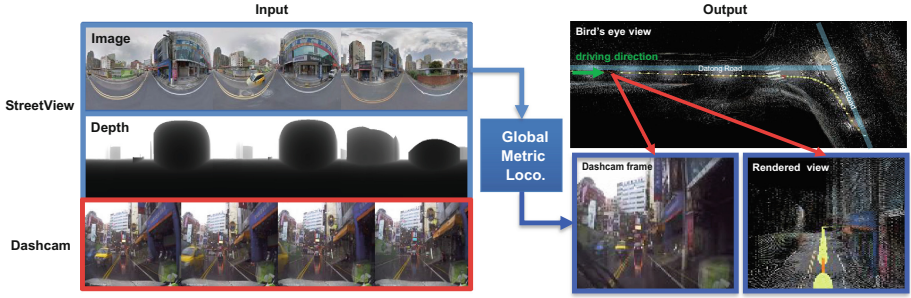
<sup>2</sup> Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan  
s104062535@m104.nthu.edu.tw, s102062322@m102.nthu.edu.tw,  
htchen@cs.nthu.edu.tw

**Abstract.** Given the advance of portable cameras, many vehicles are equipped with always-on cameras on their dashboards (referred to as dashcam). We aim to utilize these dashcam videos harvested in the wild to extract the driving behavior—global metric localization of 3D vehicle trajectories (Fig. 1). We propose a robust approach to (1) extract a relative vehicle 3D trajectory from a dashcam video, (2) create a global metric 3D map using geo-localized Google StreetView RGBD panoramic images, and (3) align the relative vehicle 3D trajectory to the 3D map to achieve global metric localization. We conduct an experiment on 50 dashcam videos captured in 11 cities under various traffic conditions. For each video, we uniformly sample at least 15 control frames per road segment to manually annotate the ground truth 3D locations of the vehicle. On control frames, the extracted 3D locations are compared with these manually labeled ground truths to calculate the distance in meters. Our proposed method achieves an average error of 2.05 m and 85.5% of them have error no more than 5 m. Our method significantly outperforms other vision-based baseline methods and is a more accurate alternative method than the most widely used consumer-level Global Positioning System (GPS).

**Keywords:** Camera localization · Structure from motion

## 1 Introduction

Recently, self-driving car is one of the hottest topic in computer vision and it has received a huge amount of industrial investment to solve this holy-grail problem. One very important topic for advancing self-driving car is to build a realistic simulation environment, in particular, realistic driving behavior of other AI agents in the environment. Collecting in-house driving behavior data from humans is time-consuming and not scalable to cover many corner cases. Hence,



**Fig. 1.** Global Metric Localization of 3D vehicle trajectory. Left-Panel: inputs including StreetView RGBD panorama images (Top) and dashcam frames (Bottom). Right-Panel: output—3D trajectory (yellow dots) in bird’s eye view (Top). A rendered view compared to a dashcam frame is shown at the Bottom. (Color figure online)

we propose to crowd-source driving behavior from many individual drivers using cheap portable cameras.

Thanks to the advance of portable camera, many vehicles are equipped with always-on cameras on their dashboards (referred to as dashcam). For instance, dashcams are equipped on almost all new cars in Taiwan, South Korea, and Russia. Its most common use case is to report special events such as traffic violations. Due to the popularity of these videos on video sharing website such as YouTube, many dashcam videos can be harvested on the web. In this work, we aim for extracting the driving behavior—“global metric localization” of 3D vehicle trajectories—in these dashcam videos. This is a challenging task, since dashcams typically are uncalibrated wide field-of-view cameras. Nevertheless, these driving behaviors can potentially be very valuable due to its diversity—it is captured by different drivers at different times and locations under different traffic conditions.

We define the task of “global metric localization” as localizing the 3D vehicle trajectory in the global metric 3D map. The 3D map is referred to as global and metric, since the whole world shares the same map and the unit in the 3D map can be converted into meters, respectively. The idea is that, with 3D vehicle trajectories in global metric 3D map, the extracted driving behavior can be analyzed to create realistic AI driving agents in the future. We propose a three-step approach to extract the driving behavior. First, we apply state-of-the-art structure from motion technique [1] to jointly estimate the camera intrinsic, the camera motion, and the 3D structure of the scene (referred to as Dashcam 3D). At this point, the 3D information of each video is in each coordinate system (Sect. 3.1). Hence, we cannot accumulate driving behavior in a single reference coordinate. Second, we use Google StreetView to create a 3D map in a single reference coordinate (Sect. 3.2). We modify the approach of Cavallo [2] and combine many geo-localized scans of Google StreetView RGBD panoramic images to create a simplified 3D map (referred to StreetView 3D).

Finally, we apply a cascade of image-level and feature-level matching methods to efficiently find image patch matches between the StreeView 3D and Dashcam 3D. We further extract the ground plans in both StreeView and Dashcam 3D and assume two ground plans are identical. Given the patch matches and this assumption, we can simplify a 3D transformation estimation problem into a 2D transformation and vertical scale estimation problem. This significantly makes the RANSAC model estimation more robust (Sect. 3.3).

As a first step toward this direction, we conduct an experiment on 50 dashcam videos captured in 11 cities under various traffic conditions. For each video, we uniformly sample at least 15 control frames per road segment to manually annotate the ground truth 3D coordinates of the vehicle. Each extracted 3D trajectory is compared with these manually labeled ground-truth 3D control frames to calculate the distance in meters. Our proposed method achieves an average distance of 2.04 m and 85.5% of them have distance no more than 5 m. Our method significantly outperforms other vision-based baseline methods and is a more accurate alternative method for vehicle localization than the most widely used consumer-level Global Positioning System (GPS).

## 2 Related Work

Extracting vehicle trajectory is related to three types of visual localization tasks: image-based landmark localization, landmark localization using 3D point clouds in a map, and vehicle localization.

**Image-Based Landmark Localization.** [3, 4] are early work showing the ability to match query images to a set of reference images. Schindler et al. [5] improve the performance to handle city-scale localization. Hays and Efros [6] further demonstrate that query images can be matched to a collection of 6 million GPS-tagged images dataset (within 200 km) at a global scale. Zamir and Shah [7] propose to use Street View images as reference images with GPS-tags, and match SIFT keypoints in a query image efficiently to SIFT keypoints in reference images by using a tree structure. A voting scheme is also introduced to jointly localize a set of nearby query images (within 300 m). As a result of voting, it is able to outperform [5]. Vaca-Castano et al. [8] propose to estimate the trajectory of a moving camera in the longitude and latitude coordinate using Bayesian filtering to incorporate the map topology information. Similar to [7, 8], we also use Street View images as reference images. However, unlike [7], we assume a video sequence is captured across an arbitrary distance (not restricted to within 300 m). Unlike [8], we use not only the topology of the map, but also the relative 3D position of the dashcam frames to improve the localization accuracy. Cao and Snavely [9] propose to match a query image to reference images with a graph-based structure to reliably retrieve a sub-group of images corresponding to a representative landmark. This method can be used to improve single image matching accuracy at the first stage of our method. Bettadapura et al. [10] also use Street View images as reference images to match the point-of-view images

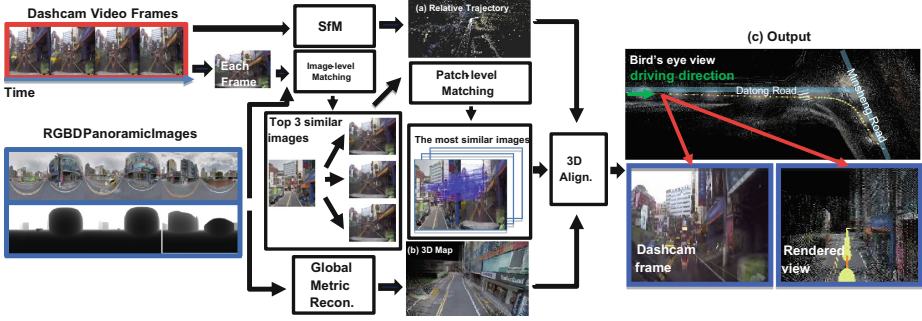
captured by a cellphone camera. Moreover, the method utilizes accelerometers, gyroscopes and compasses on the cellphone to improve the matching accuracy. The final estimated point-of-view is used as an approximation of the users’ attention in applications such as egocentric video tours at museums.

**Landmark Localization Using 3D Point Clouds in a Map.** Accurate image-based camera 6 Degrees of Freedom (DoF) pose estimation can also be achieved by utilizing 3D point clouds in a map [11–14]. In our approach, we use many geo-localized Google StreetView RGBD panoramic images to create a 3D map using a modified method of [2]. Yu et al. [15] treat each frame in a dashcam video as a separate image query to estimate the 6DoF camera pose in the StreetView 3D map by solving the Perspective-n-Point (PnP) problem [16]. Our method is similar to [15]. Except that we treat frames in the same video as a whole sequence. Hence, our method additionally integrates all frames in the relative 3D trajectory. This makes our proposed method more robust.

**Vehicle Localization.** For egocentric vehicle localization, many methods combining image-based sensors with other sensors or map information have been proposed. Given the vehicle speed at all time and the rough initial vehicle location, Badino et al. [17] use Bayesian filter and a per-frame-based visual feature to align a current frame to pre-recorded frames in the database while considering the candidate location of previous frames. Taneja et al. [18] propose a similar but lightweight method requiring images sparsely sampled in space (in average every 7 m). Lategahn et al. [19] combine frame-based visual matching and Inertial Measurement Unit (IMU) for localization. However, it also assumes the GPS information of the first frame is also given. Both [20, 21] utilize relative position information from visual odometry with map information from OpenStreetMaps to globally localize a vehicle. However, these methods require the vehicle trajectory to be complex enough to be uniquely identified on the map (i.e., many turns, etc.). Dashcam videos on YouTube do not come with additional speed or initial location information. Moreover, a video is typically less than 5 min with simple trajectories. In contrast, our method does not require an initial location of the first frame or any extra sensor, and it can even localize vehicles with simple trajectories using a global metric 3D map created from StreetView RGBD panoramic images. However, we do assume some weakly location information of the whole video sequence such as city, road name, etc. are known.

### 3 Our Method

In order to extract the driving behavior, we propose a three-step approach for “global metric localization” of 3D vehicle trajectories. The pipeline of the approach includes (1) extracting a relative 3D vehicle trajectory (Fig. 2(a)), (2) creating a global metric 3D map (Fig. 2(b)), and (3) aligning the 3D trajectory to the 3D map (Fig. 2(c)). In these steps, we utilize state-of-the-art Structure-from-Motion (SfM), Geographical Information Systems (GIS), and dense matching.

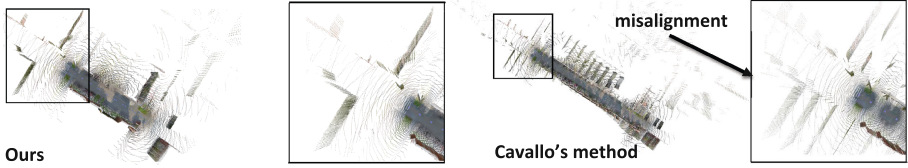


**Fig. 2.** System Pipeline. Panel (a): the obtained relative 3D trajectory given the dashcam video. Panel (b): the obtained global metric 3D map given StreetView RGBD panorama images. Panel (c): the output – global metric vehicle 3D trajectory.

Moreover, we propose a novel 3D alignment method utilizing a ground-plane prior information in StreetView scenes to achieve the best result.

### 3.1 Extracting 3D Vehicle Trajectory

A dashcam video can be interpreted as sequential images captured along with the movement of the vehicle. Hence, we treat the dashcam 3D camera trajectory as the proxy of the 3D vehicle trajectory. In order to obtain the 3D vehicle trajectory, we apply sequential Structure-from-Motion (SfM) techniques to the estimation of relative camera positions with respect to the first frame. Note that our problem is not a classical Visual Odometry (VO) problem, since we do not have the camera intrinsic information of the harvested dashcam video. Moreover, most of the dashcam videos are captured by a wide-angle or even a fisheye camera, which introduces an unknown camera distortion in raw video frames. Therefore, it is critical to estimate the camera intrinsic including camera distortion. We use a state-of-the-art sequential Structure-from-Motion (SfM) method [1] to jointly estimate the camera intrinsic and extrinsic, and the sparse 3D point cloud representation of the scene. We refer to the coordinate system of the reconstruction as “Dashcam 3D”. Note that the 3D vehicle trajectory in Dashcam 3D has two drawbacks. Firstly, the trajectory is only known up-to-scale (not metric). For instance, we can know the car is driving with constant speed, but we do not know the exact speed in meters. Secondly, Dashcam 3D is isolated from other meta information in GIS. For instance, we can know the car is making a turn, but we do not know the car is turning at a specific intersection (e.g., between market street and main street). Hence, we need a coordinate system which is metric and is globally consistent with meta information such as road intersections, traffic signs, etc.



**Fig. 3.** Our Global Metric 3D Map (Left Panel) vs. Cavallo’s method [2] (Right Panel). It shows that our method generates a much better aligned 3D Map.

### 3.2 Creating Global Metric 3D Map

Google StreetView is one of the largest and most information-rich street-level GIS with millions of RGBD panoramic images captured all over the world. Every panoramic image is geotagged with an accurate GPS position and covers a  $360^\circ$  horizontal and  $180^\circ$  vertical field-of-view (FoV). The spherical projection of the RGB and Depth images are shown in the bottom-left corner of Fig. 2. Note that the depth image is calculated from normal directions and distances of dominant surfaces in the scene. This allows to map building facades and roads while ignoring smaller entities such as vehicles or pedestrians. This also implies the depth image is an approximation of the true depth image of the scene. Generally, there is a panorama every 6 to 15 m with a nearly uniform street coverage. Hence, it is very likely that a dashcam observes many similar StreetView panoramas along the path of the vehicle. Our goal is to use Google StreetView to create a metric 3D Map which is consistent with other meta information such as roads, intersections, etc. at a global scale. We modify Cavallo’s method [2] to create a global metric 3D Map. In particular, Cavallo’s method places the 3D Map of each panorama onto a 2D plane according to the associated GPS position. This introduces noticeable approximation error when the 3D Map becomes large. Alternatively, we place the 3D Map of each panorama onto a 3D sphere as below,

$$r = E_r \cdot \cos(lat), \quad (1)$$

$$y = E_r \cdot \sin(lat), \quad (2)$$

$$x = r \cdot \sin(lon), \quad (3)$$

$$z = r \cdot \cos(lon), \quad (4)$$

where  $lat$  stands for latitude,  $lon$  stands for longitude,  $E_r$  denotes the radius of the Earth, and  $[x, y, z]$  denotes a location in the global metric 3D map coordinate. This mitigates the approximation error even when the 3D Map is large (Fig. 3).

### 3.3 Aligning 3D Trajectory to 3D Map

In order to obtain the global metric vehicle 3D trajectory, we propose to align the trajectory in Dashcam 3D to the global metric 3D Map obtained from Google StreetView. We introduce the following coarse-to-find steps for alignment.

**Weak Location Information.** Since most dashcams are equipped with a consumer-level GPS, we roughly know that the video is captured close to a few geo-locations. For instance, a set of road intersections. Then, we use Google StreetView API to retrieve all StreetView panoramic images within  $M$  meters of the known geo-locations.

**Image-Level Matching.** Given a set of retrieved panoramic images and a dashcam video, we want to find a set of matched image pairs—one panorama matched with one dashcam frame. Since dashcam video typically captures the viewing direction parallel to the road direction, we crop each panoramic image into two “reference images” (i.e., two directions per road), each with  $36^\circ$  vertical FoV and  $72^\circ$  horizontal FoV (i.e., one fifth of the FoV of a panoramic image). We use the following techniques to measure the similarity of a pair of images.

- Holistic feature similarity. For both the reference images and frames, we detect sparse SIFT keypoints and represent each keypoint using SIFT descriptor [22]. Then, we use Fisher vector encoding [23] to generate our holistic feature representation for all images. The cosine similarity between a pair of Fisher vectors is used to efficiently measure the similarity of a pair of reference image and frame.
- Geometric verification. The similarity between a pair of reference image and frame can be more reliably confirmed by applying geometric verification method consisting of (1) raw SIFT keypoint matches with ratio test, and (2) RANSAC matching with epipolar geometric verification. However, geometric verification is more computationally expensive than calculating similarity using holistic feature.

In our application, we assume the reference images densely cover a region on the map containing the path of the dashcam. This implies that we only need a small set of reference images for the path. Therefore, we start from each frame and use holistic feature to retrieve its top  $K$  similar reference images. Then, we apply geometric verification only on the retrieved images to re-order them according to the number of inlier matches which is referred to as the “confidence score”. At this point, we obtain a ranked list of pairs of frames and their corresponding reference images. Since some frames have very generic appearance such that they are similar to many reference images, we use the standard ratio test to keep only discriminative frames. In particular, the ratio of the confidence score between the top-1 and the second similar reference image needs to be larger than  $\gamma$ .

**Patch-Level Matching.** Given a matched image pair, we further want to obtain the matched image patch pairs between the frame and matched reference images. Note that every image patch in a dashcam frame is associated with a 3D location in Dashcam 3D coordinate. Similarly, every image patch in a reference image is associated with a 3D location in StreetView 3D coordinate. Hence, we can establish 3D correspondences between Dashcam and StreetView 3D coordinates given matched image patch pairs. Although we already obtain

SIFT keypoint matches while obtaining image-level matching, we find it is not very reliable and the keypoints are very sparse. In order to improve both the precision and recall of 3D correspondence between Dashcam and StreetView 3D coordinates, we apply a state-of-the-art dense matching method [24] to all matched image pairs and obtain a much more reliable set of 3D correspondences than other keypoint-based matches. To further increase the precision, we remove all pairs of images with less than  $Q$  matched image patches.

**3D Alignment.** Given the 3D correspondence between Dashcam and StreetView 3D coordinates, we aim to estimate the best 3D transformation to align these two coordinates. This task can be solved using a classical Random Sample Consensus (RANSAC) with 3D transformation method. However, we have found this classical approach often generates alignment results where the ground planes from both coordinates are not the same after alignment. Therefore, we propose to first make sure that the ground planes in both Dashcam and StreetView 3D coordinates occupy the  $(x, y)$  plane. Next, to ensure the aligned ground plane still occupies the  $(x, y)$  plane, we propose the following “ground-prior” 3D transformation to be estimated as described below,

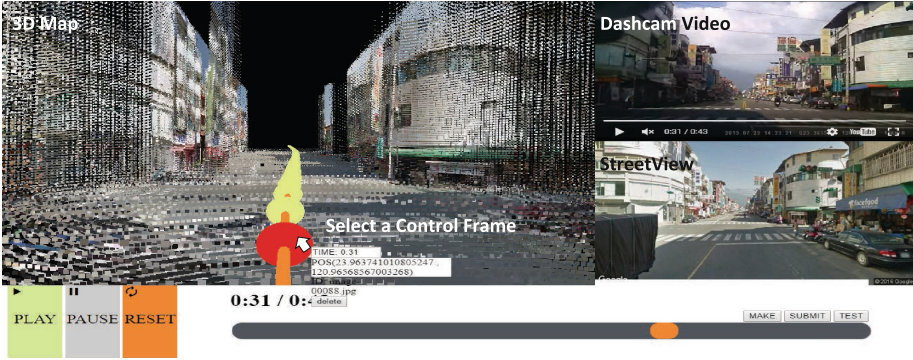
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & e \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} + \begin{bmatrix} f \\ g \\ 0 \end{bmatrix}, \quad (5)$$

where  $[x, y, z]$  is a location in StreetView 3D,  $[\hat{x}, \hat{y}, \hat{z}]$  is a coordinate in Dashcam 3D, parameters  $[a, b, c, d, f, g]$  encode 2D affine transformation in the  $x, y$  plane, and parameter  $e$  encodes scaling in  $z$  axis. Given many 3D correspondences (i.e.,  $[x, y, z]$  and  $[\hat{x}, \hat{y}, \hat{z}]$ ), we can estimate parameters  $[a, b, c, d, e, f, g]$  by solving least squares problem, and a reliable set of parameters can be estimated by using RANSAC with “ground-prior” 3D transformation. Note that our proposed transformation only have 7 degrees of freedom compared to 12 and 9 for 3D affine and rigid transformations, respectively. Moreover, we do not assume that the output point cloud of the sfm is isotropic (i.e., scalings in x,y,z direction are equal). Our proposed “ground-prior” 3D transformation allows different scaling factor for each axis. As a result, we found our proposed “ground-prior” 3D transformation achieves much smaller alignment error.

### 3.4 Implementation Details

The value of  $M$ , which controls the number of Google StreetView panoramas to be considered, depends on the uncertainty of the known geo-locations. In our case, we found  $M = 500$  m is a good value considering noise in consumer-level GPS. For image-level matching, We found that  $K = 3$  and  $\gamma = 1.3$  retrieves sufficient number of good matched image pairs. For patch-level matching, we found  $Q = 10$  gives us good results.





**Fig. 4.** Annotation interface. Left panel: StreetView 3D map, where yellow circle denotes the 3D location of a specific frame. Top-Right panel: Dashcam video. Bottom-Right panel: StreetView cropped image. All three panels are synchronized according to the 3D trajectory. By marking a yellow circle into a red circle, a user can drag it to modify the 3D location of a specific frame in 3D map. (Color figure online)

## 4 Experiments

We evaluate our proposed method on real-world dashcam videos downloaded from the Internet, and show that our method achieves state-of-the-art global metric 3D trajectory results compared to other baseline methods.

### 4.1 Data and Annotation

We harvest 50 dashcam videos on YouTube with good video quality and resolution higher than or equal to 720p. We also retrieve one GPS location for each video as mentioned in the description of the video or provided by the video owner. From the GPS locations, we know these videos are captured in a diverse set of 11 cities in Taiwan. In order to have the ground-truth global metric 3D trajectories, we build a web-based user annotation interface to compare the rendered view of StreetView 3D Map and a dashcam frame (Fig. 4). For each road segment, the users uniformly select at least 15 control frames and use the interface to drag the camera pose of each frame to the correct location in 3D Map. The dashcam videos and annotations will be released once the paper is accepted.

### 4.2 Evaluation Metric

To measure the quality of the predicted 3D trajectory, we calculate the distance between the ground truth and the predicted locations in Google StreetView 3D coordinate for the control frames. The smaller the distance the better the quality of the prediction. We refer to this distance as error.

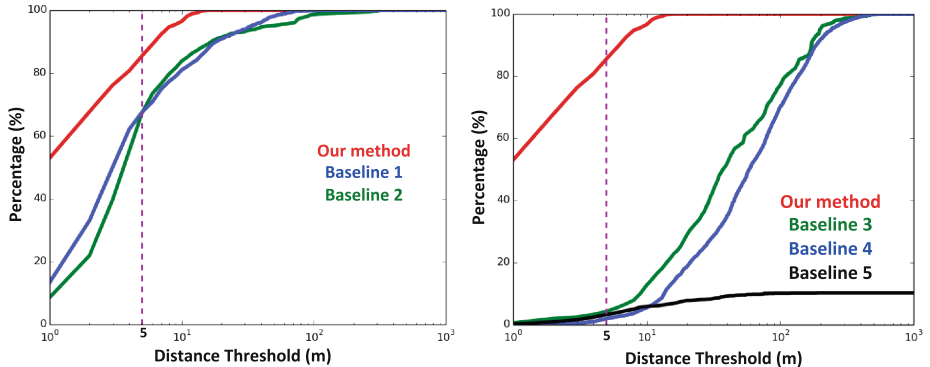
**Table 1.** List of methods for comparison with their detailed steps, where steps differing from our proposed method are highlighted using bold fonts. Average error (Avg.) and standard deviation (SD) are reported at the last two columns, where the smallest values are highlighted using bold fonts. Geo-Verification stands for geometric verification. DHOG stands for dense HOG. DSIFT stands for dense SIFT. SfM stands for Structure-from-Motion. \* indicates that the performance is evaluated on successfully estimated control frames which are only 10.2% of all control frames.

	Feature-match	3D estimation	SfM	Avg	SD
Our	Deepmatch	ground prior + 3D alignment + RANSAC	Yes	<b>2.05</b>	<b>2.88</b>
Baseline 1	Deepmatch	<b>3D Rigid alignment</b> + RANSAC	Yes	7.44	12.04
Baseline 2	Deepmatch	<b>3D Affine alignment</b> + RANSAC	Yes	10.52	28.23
Baseline 3	<b>DHOG match</b>	ground prior + 3D alignment + RANSAC	Yes	67.96	75.26
Baseline 4	<b>DSIFT match</b>	ground prior + 3D alignment + RANSAC	Yes	85.01	80.39
Baseline 5	Deepmatch	<b>PnP RANSAC</b>	No	17.12*	22.8*

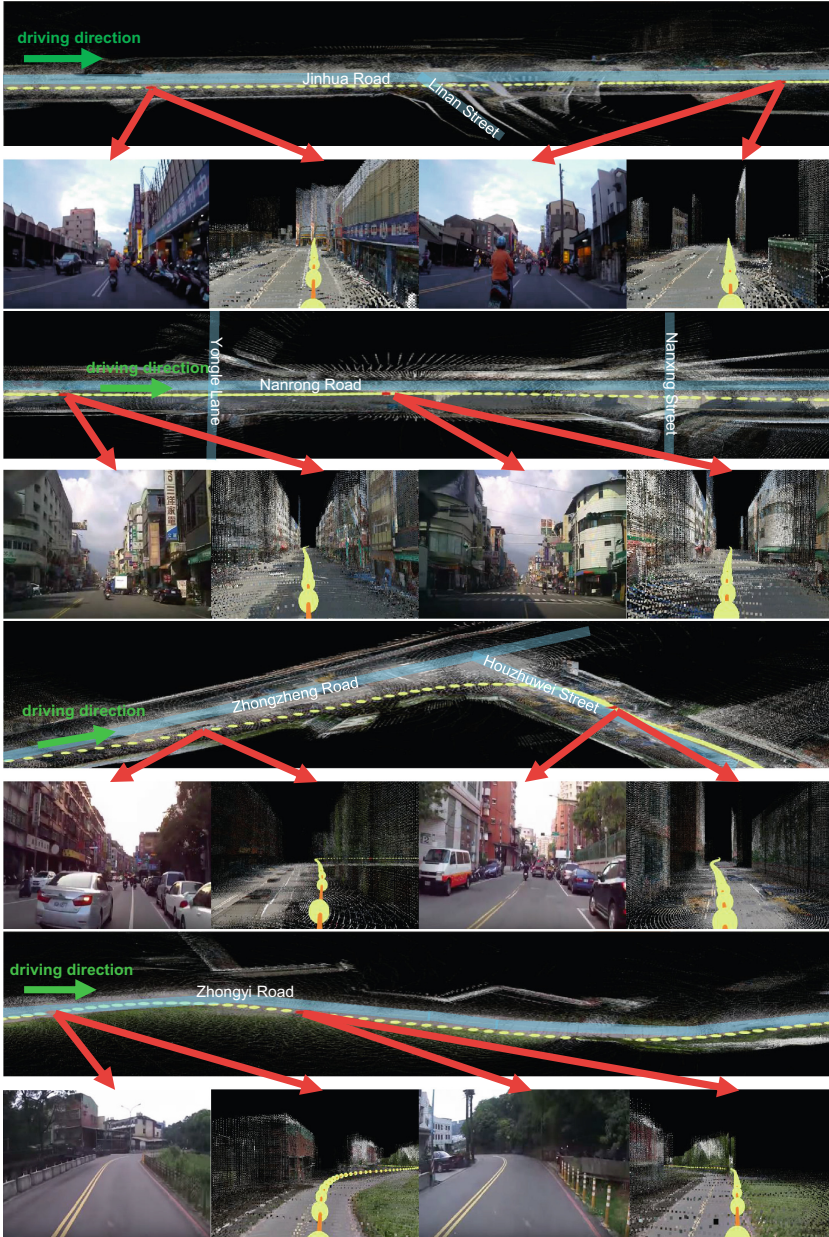
### 4.3 Baseline Methods

We compare our method with several variants (Table 1):

- Baseline 1. Estimating 3D rigid transformation with no ground-prior.
- Baseline 2. Estimating 3D affine transformation with no ground-prior.
- Baseline 3. Using dense HOG-based matching rather than [24].
- Baseline 4. Using dense SIFT-based matching [25] rather than [24].
- Baseline 5 [2]. Each frame is matched to one panoramic image. If there are enough patch correspondences, we use PnP [16] to estimate the 3D camera pose in StreetView 3D coordinate.



**Fig. 5.** Percentage of control frames vs. error threshold. We highlight the percentage with 5 m as threshold using a dash purple line. (Color figure online)



**Fig. 6.** Typical examples. Top-panel: the bird's eye view. Bottom-panel: rendered view compared with dashcam frame. Yellow dot denotes the location of each frame. Two consecutive dots are connected by a dark yellow edge. Green arrow indicates the driving direction. Light blue edges denote the road information. (Color figure online)

## 4.4 Results

For each method, we calculate the average error and standard deviation over all control frames (Table 1). Our method achieves the smallest average error of 2.05 m and standard deviation of 2.88 m. We confirm that our proposed ground-prior 3D alignment outperforms 3D rigid (Baseline 1) and affine (Baseline 2) transformations. Moreover, deepmatch [24] significantly outperforms other dense matching methods (Baseline 3 and 4). Finally, We also report the percentage of control frames with error lower than a threshold. By changing the threshold, we can plot a curve with percentage at the vertical axis (Fig. 5). Our method predicts 85.5% locations with error no more than 5 m. We show many typical examples in Fig. 6. In each example, we show the bird’s eye view of the global metric 3D trajectory in StreetView 3D coordinate, where a yellow dot denotes the location of each frame, two consecutive dots are connected by a dark yellow edge, the green arrow indicates the driving direction, and the light blue edges denote the road information. Two comparisons between rendered view and dashcam frame are also shown for each example. From these examples, we can see that our estimated trajectories are accurate at lane-level. For instance, the car is driving on the outside lane in the first example of Fig. 6.

## 5 Conclusion

We propose a robust method to obtain global metric 3D vehicle trajectories from dashcam videos in the wild. Our method potentially can be used to crowd-source driving behaviors for developing self-driving car simulator. On 50 dashcam videos, our proposed method achieves an average error of 2.05 m and 85.5% of them have error no more than 5 m. Our method significantly outperforms other vision-based baseline methods and is a more accurate alternative method than the most widely used consumer-level Global Positioning System (GPS).

**Acknowledgements.** We thank MOST 104-3115-E-007-005 in Taiwan for its support.

## References

1. Gargallo, P., Kuang, Y.: Opensfm. <https://github.com/mapillary/OpenSfM/>
2. Cavallo, M.: 3d city reconstruction from google street view. *Comput. Graph.* J. (2015)
3. Robertson, D., Cipolla, R.: An image-based system for urban navigation. In: *BMVC* (2004)
4. Zhang, W., Kosecka, J.: Image based localization in urban environments. In: *3DPVT* (2006)
5. Schindler, G., Brown, M., Szeliski, R.: City-scale location recognition. In: *CVPR* (2007)
6. Hays, J., Efros, A.A.: Im2gps: estimating geographic information from a single image. In: *CVPR* (2008)

7. Zamir, A.R., Shah, M.: Accurate image localization based on google maps street view. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6314, pp. 255–268. Springer, Heidelberg (2010). doi:[10.1007/978-3-642-15561-1\\_19](https://doi.org/10.1007/978-3-642-15561-1_19)
8. Vaca-Castano, G., Zamir, A., Shah, M.: City scale geo-spatial trajectory estimation of a moving camera. In: CVPR (2012)
9. Cao, S., Snavely, N.: Graph-based discriminative learning for location recognition. In: CVPR (2013)
10. Bettadapura, V., Essa, I., Pantofaru, C.: Egocentric field-of-view localization using first-person point-of-view devices. In: WACV (2015)
11. Irschara, A., Zach, C., Frahm, J., Bischof, H.: From structure-from-motion point clouds to fast location recognition. In: CVPR (2009)
12. Li, Y., Snavely, N., Huttenlocher, D.P.: Location recognition using prioritized feature matching. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6312, pp. 791–804. Springer, Heidelberg (2010). doi:[10.1007/978-3-642-15552-9\\_57](https://doi.org/10.1007/978-3-642-15552-9_57)
13. Sattler, T., Leibe, B., Kobbelt, L.: Fast image-based localization using direct 2d-to-3d matching. In: ICCV (2011)
14. Li, Y., Snavely, N., Huttenlocher, D., Fua, P.: Worldwide pose estimation using 3d point clouds. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7572, pp. 15–29. Springer, Heidelberg (2012). doi:[10.1007/978-3-642-33718-5\\_2](https://doi.org/10.1007/978-3-642-33718-5_2)
15. Yu, L., nad Guillaume Bresson, C.J., Moutarde, F.: Monocular urban localization using street view. Arxiv (2016)
16. Lepetit, V., Moreno-Noguer, F., Fua, P.: Epnnp: an accurate o(n) solution to the pnp problem. *Int. J. Comput. Vis.* **81**(2), 155 (2009)
17. Badino, H., Huber, D., Kanade, T.: Real-time topometric localization. In: ICRA (2012)
18. Taneja, A., Ballan, L., Pollefeys, M.: Never get lost again: vision based navigation using StreetView images. In: Cremers, D., Reid, I., Saito, H., Yang, M.-H. (eds.) ACCV 2014. LNCS, vol. 9007, pp. 99–114. Springer, Heidelberg (2015). doi:[10.1007/978-3-319-16814-2\\_7](https://doi.org/10.1007/978-3-319-16814-2_7)
19. Lategahn, H., Schreiber, M., Ziegler, J., Stiller, C.: Urban localization with camera and inertial measurement unit. In: Intelligent Vehicles Symposium (IV) (2013)
20. Floros, G., van der Zander, B., Leibe, B.: OpenStreetSLAM: global vehicle localization using openstreetmaps. In: ICRA (2013)
21. Brubaker, M., Geiger, A., Urtasun, R.: Lost! leveraging the crowd for probabilistic visual self-localization. In: CVPR (2013)
22. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision* **60**(2), 91–110 (2004)
23. Jegou, H., Perronnin, F., Douze, M., Snchez, J., Perez, P., Schmid, C.: Aggregating local image descriptors into compact codes. *TPAMI* **34**, 1704–1716 (2011)
24. Weinzaepfel, P., Revaud, J., Harchaoui, Z., Schmid, C.: DeepFlow: Large displacement optical flow with deep matching. In: ICCV (2013)
25. Kim, J., Liu, C., Sha, F., Grauman, K.: Deformable spatial pyramid matching for fast dense correspondences. In: CVPR (2013)