Monocular Visual Odometry and Dense 3D Reconstruction for On-Road Vehicles

Menglong Zhu[1] Srikumar Ramalingam[2] Yuichi Taguchi[2] Tyler Garaas[2]

[1]University of Pennsylvania, Philadelphia, PA, USA
[2]Mitsubishi Electric Research Labs (MERL), Cambridge, MA, USA

Abstract. More and more on-road vehicles are equipped with cameras each day. This paper presents a novel method for estimating the relative motion of a vehicle from a sequence of images obtained using a single vehicle-mounted camera. Recently, several researchers in robotics and computer vision have studied the performance of motion estimation algorithms under non-holonomic constraints and planarity. The successful algorithms typically use the smallest number of feature correspondences with respect to the motion model. It has been strongly established that such minimal algorithms are efficient and robust to outliers when used in a hypothesize-and-test framework such as random sample consensus (RANSAC). In this paper, we show that the planar 2-point motion estimation can be solved analytically using a single quadratic equation, without the need of iterative techniques such as Newton-Raphson method used in existing work. Non-iterative methods are more efficient and do not suffer from local minima problems. Although 2-point motion estimation generates visually accurate on-road vehicle trajectory, the motion is not precise enough to perform dense 3D reconstruction due to the non-planarity of roads. Thus we use a 2-point relative motion algorithm for the initial images followed by 3-point 2D-to-3D camera pose estimation for the subsequent images. Using this hybrid approach, we generate accurate motion estimates for a plane-sweeping algorithm that produces dense depth maps for obstacle detection applications.

Key words: Visual odometry, 2-point motion estimation, pose estimation, plane sweeping

1 Introduction and Related Work

Accurate ego-motion estimation of a vehicle from video sequences is a challenging and important problem in robotics and computer vision [11]. Existing algorithms can be classified based on the camera model (monocular, stereo) and motion model (planar, non-planar). To compute the relative motion between images, it has been shown that using a minimum number of feature correspondences in a hypothesize-and-test framework such as RANSAC produces accurate and robust results in the presence of outliers.

Several SLAM techniques for general motion sequences exist in the robotics community [9, 13]. In this paper, we focus on ego-motion estimation from monocular video sequences [22, 24, 18] for almost planar road sequences. We present a

novel analytical solution to the planar motion estimation problem using 2 points. Existing approaches solved this problem using an iterative algorithm. Our non-iterative solution is more efficient and does not have local minima problems.

Dense depth estimation from video sequences using a car-mounted camera can be extremely useful for car safety applications such as obstacle detection. In this work, we use the 2-point motion estimation algorithm followed by 3-point 2D-to-3D pose estimation to compute the camera poses, and then reconstruct a dense depth map at each frame. Instead of using just two images and motion-based stereo, we use several images, typically of the order of 20 to 30, to reconstruct depth maps using a plane-sweeping algorithm [4]. Plane sweeping can be implemented on GPUs and it can be performed in real time [16].

Minimal Solutions: Nistér's 5-point algorithm [14] with a RANSAC framework has been established as the standard method for motion estimation in the presence of outliers. In the case of relative motion between two cameras, there are 6 degrees of freedom (DOF) in the motion parameters: 3 DOF for rotation and 3 DOF for translation. For conventional cameras with a single center of projection, only 5 parameters can be estimated, i.e., the translation can only be estimated up to a scale. Accordingly, we need a minimum of 5 feature correspondences to estimate the motion parameters. The feature correspondences can be obtained using, e.g., Harris corners, SIFT, or KLT. Usually, minimal approaches lead to a finite number of solutions for the motion and the correct motion is chosen based on physical realizability or additional point correspondences.

Minimal solutions have been proposed for several calibration and 3D reconstruction problems: the five point relative pose problem [14], the six point generalized camera problem [23], point-to-plane registration using six correspondences [17], pose estimation for stereo setups using either points or lines [2, 3]. The last few years have seen the use of minimal problems in various applications [21] and there are even unification efforts to keep track of all the existing solutions[1].

Restricted Motion Models: In real-world scenarios, the relative motion of a camera is usually constrained by the associated application. For example, a camera mounted on a car does not generally have all 6 DOF. If the road is planar, the camera can only undergo 3 DOF (2 DOF of translation and 1 DOF of rotation). Recently, Scaramuzza et al. [19] have shown that there exists a class of vehicles (cars, bicycles, differential-drive robots) whose motion can be parameterized using only one parameter and thus a 1-point algorithm can be developed. The underlying idea is to use the fact that there exists an instantaneous center of rotation (ICR) and the vehicle follows a circular course around this point. When inertial measurement unit (IMU) is available, one can get two measurement angles using the gravity vector. The remaining unknowns are just three parameters (1 DOF of rotation and 2 DOF of translation). Fraundorfer et al. [5] solved this 3-point motion estimation problem using a quartic equation. This motion estimation algorithm can be useful if one uses cell phone cameras to capture images.

[1] http://cmp.felk.cvut.cz/minimal/

Several researchers have compared the performance of minimal algorithms such as 1-point, 2-point and 5-point algorithm. As shown in many of the recent results, 2-point is comparable to 1-point and 5-point [18]. However, the existing algorithm used for 2-point is an iterative one. Ortin and Montiel [15] proposed a 2-point motion estimation algorithm for planar motion sequences. This is applicable for indoor robot ego-motion estimation when the camera mounted on the robot moves on a plane. The number of degrees of freedom is 3 (1 DOF of rotation and 2 DOF of translation). However, the relative motion can be recovered only up to a scale. In the RANSAC framework, the number of iterations required is usually smaller when we decrease the number of points required to compute the motion. Given the complexity of the equations, Ortin and Montiel determined the solutions iteratively with the Newton-Raphson method and this iterative approach is still used in several recent results [18]. In this paper, we show that 2-point motion estimation can be solved easily with a simple quadratic equation. Independently, Booij and Zivkovic have developed an analytical solution for 2-point motion estimation using a trignometric approach in their unpublished technical report [1]. Similar to their approach, we also obtain a quadratic polynomial for the 2-point problem. However, our formulation is different and we show an entire pipeline including dense 3D reconstruction for driver assistance in car-navigation.

Organization: In Section 2, we present a minimal 2-point relative motion estimation algorithm. In Section 3, we detail our system pipeline and experimental results, including the calibration procedure, 2-point motion estimation followed by 3-point 2D-to-3D pose estimation, and dense depth reconstruction using plane sweeping for obstacle detection application.

2 2-Point Motion Estimation

In this section, we first describe some background of motion estimation problems and then present our analytical solution to the 2-point motion estimation problem.

2.1 Background

Motion estimation refers to estimating the relative pose between two images. Corresponding feature points $\mathbf{p}$ and $\mathbf{p}'$ in two images are related by the essential matrix E by the relation ${\mathbf{p}'}^T\mathsf{E}\mathbf{p} = 0$. Note that $\mathbf{p}$ and $\mathbf{p}'$ are expressed as unit vectors in spherical image coordinates ($\mathbf{p}$ and $\mathbf{p}'$ are pixels back-projected onto a unit sphere such that $||\mathbf{p}|| = ||\mathbf{p}'|| = 1$). This is always possible when the camera is calibrated. The essential matrix E can be computed using the relationship $\mathsf{E} = [\mathbf{T}]_\times\mathsf{R}$, where R is the 3×3 rotation matrix and $[\mathbf{T}]_\times$ is the skew symmetric matrix of the 3×1 translation vector $\mathbf{T}$.

Planar Motion: As shown in Figure 1, we assume that the camera moves on a plane. In the case of a camera mounted on a car, we assume that the road is parallel to the XZ plane and the camera moves on the XZ plane, as shown

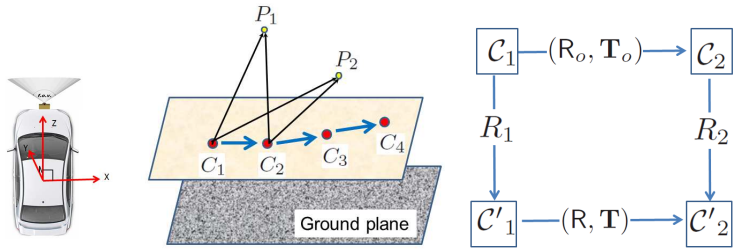

Fig. 1. *Two-point motion estimation for planar motion of an on-road vehicle. (a) The motion of the car is assumed to be on the XZ plane. (b) The camera moves on a plane parallel to the ground plane. We show the projection rays for two 3D points P_1 and P_2 from two camera positions C_1 and C_2. (c) The general idea behind our coordinate transformation technique for the 2-point motion estimation algorithm. Our goal is to compute the motion $(\mathsf{R}_o, \mathbf{T}_o)$ between the coordinate frames $\mathcal{C}_1$ and $\mathcal{C}_2$. We transform the coordinate frames $\mathcal{C}_1$ and $\mathcal{C}_2$ to two intermediate coordinate frames $\mathcal{C}'_1$ and $\mathcal{C}'_2$ respectively. We compute the motion $(\mathsf{R}, \mathbf{T})$ between $\mathcal{C}'_1$ and $\mathcal{C}'_2$, which is much simpler than computing $(\mathsf{R}_o, \mathbf{T}_o)$ directly.*

in Figure 1(a). Accordingly, the rotation matrix and the translation vector are given by

$$\mathsf{R} = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} = \begin{pmatrix} \alpha & 0 & \beta \\ 0 & 1 & 0 \\ -\beta & 0 & \alpha \end{pmatrix}, \quad \mathbf{T} = \begin{pmatrix} T_x \\ 0 \\ 1 \end{pmatrix} \tag{1}$$

The above rotation matrix represents a rotation around the Y axis by an angle θ. We rewrite the rotation matrix by replacing $\cos\theta$ and $\sin\theta$ with α and β. According to the orthonormality constraint, we have $\alpha^2 + \beta^2 = 1$. Since the camera moves on the XZ plane, the Y coordinate of the translation vector is 0. Since the absolute scale cannot be computed, we choose the scale of the motion by fixing $T_z = 1$. Due to the coordinate transformation we later perform in the algorithm, this assumption will hold true even if the motion is along the X direction. We use the essential matrix to compute the unknown parameters (T_x, α, β). Although there are three variables, the number of independent variables is only 2 since $\alpha^2 + \beta^2 = 1$. By directly solving the essential matrix for two sets of points correspondences, we obtain two quadratic equations in three variables (T_x, α, β). Using the orthonormality constraint on α and β, we may get 8 solutions or less. By using algebraic geometry tools like Groebner basis [10], one can directly get two solutions. On the other hand, below we show that a coordinate transformation approach will lead to a simple quadratic equation for computing the motion.

2.2 Analytical Solution

Our goal is to compute the motion between the first camera coordinate frame $\mathcal{C}_1$ to the second camera coordinate frame $\mathcal{C}_2$. Let the required motion be $(\mathsf{R}_o, \mathbf{T}_o)$. Instead of directly computing the motion between these two coordinate frames, we pre-rotate both $\mathcal{C}_1$ and $\mathcal{C}_2$ to intermediate reference frames $\mathcal{C}_1^i$ and $\mathcal{C}_2^i$ respectively. Figure 1(c) shows our transformation approach. We choose these intermediate reference frames such that the motion estimation equations become as simple as possible. Once we compute the motion between these intermediate reference frames $(\mathsf{R}, \mathbf{T})$, we can find the motion in the original coordinate frames using a simple post-rotation.

Intermediate Reference Frames: Let the two point correspondences be $(\mathbf{p}_1, \mathbf{p}'_1)$ and $(\mathbf{p}_2, \mathbf{p}'_2)$. Let us rotate the first camera coordinate system $\mathcal{C}_1$ by a rotation matrix R_1 such that the z coordinate of the first point $\mathbf{p}_1$ becomes 0. Similarly, let us rotate the second camera coordinate system $\mathcal{C}_2$ by a rotation matrix R_2 such that the z coordinate of the second point $\mathbf{p}'_2$ becomes 0. Let the new reference frames be $\mathcal{C}'_1$ and $\mathcal{C}'_2$. Let the new correspondences be $(\mathbf{a}_1, \mathbf{b}_1)$ and $(\mathbf{a}_2, \mathbf{b}_2)$ as $\mathbf{a}_i = \mathsf{R}_1\mathbf{p}_i,\ \mathbf{b}_i = \mathsf{R}_2\mathbf{p}'_i$ where $i = \{1, 2\}$. In the new reference frames, we have

$$\mathbf{a}_1 = \begin{pmatrix} a_{1x} \\ a_{1y} \\ 0 \end{pmatrix}, \quad \mathbf{a}_2 = \begin{pmatrix} a_{2x} \\ a_{2y} \\ a_{2z} \end{pmatrix}, \quad \mathbf{b}_1 = \begin{pmatrix} b_{1x} \\ b_{1y} \\ b_{1z} \end{pmatrix}, \quad \mathbf{b}_2 = \begin{pmatrix} b_{2x} \\ b_{2y} \\ 0 \end{pmatrix}. \tag{2}$$

The rotation matrices R_1 and R_2 can be easily computed as they are just equivalent to rotating the coordinate frames around the Y axis such that the Z coordinate of the point becomes 0:

$$\mathsf{R}_i = \begin{pmatrix} \cos\theta_i & 0 & \sin\theta_i \\ 0 & 1 & 0 \\ -\sin\theta_i & 0 & \cos\theta_i \end{pmatrix}. \tag{3}$$

Here $\theta_1 = \tan^{-1}(p_{1z}/p_{1x})$ and $\theta_2 = \tan^{-1}(p'_{2z}/p'_{2x})$.

Solution: Using Equation (1), we obtain the essential matrix

$$\mathsf{E} = [T]_\times \mathsf{R} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & -T_x \\ 0 & T_x & 0 \end{pmatrix} \begin{pmatrix} \alpha & 0 & \beta \\ 0 & 1 & 0 \\ -\beta & 0 & \alpha \end{pmatrix} = \begin{pmatrix} 0 & -1 & 0 \\ \alpha + \beta T_x & 0 & \beta - \alpha T_x \\ 0 & T_x & 0 \end{pmatrix}. \tag{4}$$

The equation based on the essential matrix becomes $\mathbf{b}_i^T \mathsf{E} \mathbf{a}_i = 0$ for $i = \{1, 2\}$ after the coordinate transformation. When $i = 1$, we have

$$\begin{pmatrix} b_{1x} \\ b_{1y} \\ b_{1z} \end{pmatrix}^T \begin{pmatrix} 0 & -1 & 0 \\ \alpha + \beta T_x & 0 & \beta - \alpha T_x \\ 0 & T_x & 0 \end{pmatrix} \begin{pmatrix} a_{1x} \\ a_{1y} \\ 0 \end{pmatrix} = 0, \tag{5}$$

resulting in

$$g_1 \beta T_x + g_2 T_x + g_1 \alpha + g_3 = 0, \tag{6}$$

where $g_1 = a_{1x}b_{1y}$, $g_2 = a_{1y}b_{1z}$ and $g_3 = -a_{1y}b_{1x}$. When $i = 2$, we have

$$\begin{pmatrix} b_{2x} \\ b_{2y} \\ 0 \end{pmatrix}^T \begin{pmatrix} 0 & -1 & 0 \\ \alpha + \beta T_x & 0 & \beta - \alpha T_x \\ 0 & T_x & 0 \end{pmatrix} \begin{pmatrix} a_{2x} \\ a_{2y} \\ a_{2z} \end{pmatrix} = 0, \tag{7}$$

resulting in

$$f_1\alpha T_x + f_2\beta T_x + f_2\alpha - f_1\beta + f_3 = 0, \tag{8}$$

where $f_1 = -a_{2z}b_{2y}$, $f_2 = a_{2x}b_{2y}$ and $f_3 = -a_{2y}b_{2x}$. Using Equations (6) and (8), we get the following relation for T_x:

$$T_x = \frac{-g_1\alpha - g_3}{g_1\beta + g_2} = \frac{-f_2\alpha + f_1\beta - f_3}{f_1\alpha + f_2\beta} \tag{9}$$

$$(-g_1\alpha - g_3)(f_1\alpha + f_2\beta) = (g_1\beta + g_2)(-f_2\alpha + f_1\beta - f_3) \tag{10}$$

By simplifying the above equation, we get

$$h_1\alpha + h_2\beta + h_3 = 0, \tag{11}$$

where $h_1 = g_3f_1 - f_2g_2$, $h_2 = f_1g_2 - f_3g_1 + f_2g_3$ and $h_3 = f_1g_1 - f_3g_2$. Using the orthonormality constraint $\alpha^2 + \beta^2 = 1$ to replace all β's in Equation (11), we obtain the following quadratic equation:

$$(h_1^2 + h_2^2)\alpha^2 + (2h_1h_3)\alpha + (h_3^2 - h_2^2) = 0. \tag{12}$$

We have two solutions for α by solving the above quadratic equation. Once we compute α, we obtain the corresponding two solutions for β. We can then compute T_x using Equation (9). Note that there will be two solutions for (T_x, α, β) and we can find the correct solution using additional correspondences. Finally we perform the following operations to obtain the motion between the original coordinate frames $\mathsf{R}_o = \mathsf{R}_1^T\mathsf{R}\mathsf{R}_2$ and $\mathbf{T}_o = \mathsf{R}_1^T\mathbf{T}$.

2.3 Sensitivity Analysis of Planarity Assumption

We studied the effect to the planarity assumption on the accuracy of our algorithm. In accordance with our algorithm, we assume that the camera moves on the XZ plane. The only rotation the algorithm can compute is the one around the Y axis. In simulations, we considered rotations that had an off-plane rotation along with the rotation around the Y axis. Any rotation around the X or Z axes can not be computed. In Figure 2(a) we show the error in rotation. Since the translation can only be up to a scale, the error shown is with respect to the direction of the motion on the plane. In many car navigation applications and localization problems, the rotation error around the Y axis is more important. We also studied the error in rotation around the Y axis and this is much lower than the overall rotation error. We considered images of size 100×100 pixels with a focal length of 100. The scene size is a cube of dimension 100 units. We added Gaussian noise with standard deviation of 0.2 in the simulations.

(a) (b) (c) (d)

Fig. 2. *(a) Simulations to study the sensitivity to out of plane rotation. The red and blue curves show the error in rotation and translation respectively. (b) We show the epipole (white circle) obtained from the intersection of the line segments showing the point correspondences from two consecutive images. For a car moving on a planar road, these epipoles should lie on the horizon. (c) We show the horizon (red line) and a trajectory of epipoles. The green line segments show the trajectory of the epipoles for several consecutive pairs of images. Note that the epipoles do not lie on the horizon. (d) We computed the motion parameters for* 3400 *images in a video sequence from a GoPro camera mounted on the front of a car. We project the computed trajectory on the Google Earth's aerial image. The entire trajectory is approximately 1 km long.*

3 System Pipeline and Experimental Results

3.1 Setup and Calibration

The experiments were conducted with Hero-GoPro cameras, which are extremely small in size and easy to mount on cars. We tested several video sequences by mounting the camera on both front and rear side of a car. We calibrated this camera using the omni-directional toolbox of Scaramuzza et al. [20]. The calibration is used to rectify the video sequence captured from the camera. The original image resolution is 1920×1080 pixels. Using the calibration we constructed rectified images of 1000×500 pixels. All the experiments shown in this paper use these rectified images. Once the camera is calibrated and the images are rectified, the algorithmic components shown in this paper will apply to any central camera. We compute the pose of the ground plane in the camera coordinate frame using a large calibration grid on the ground.

Feature Computation: We compared Harris corners, SIFT features, and KLT features. We observed that KLT produced more evenly distributed features compared to SIFT and Harris, thus used KLT features in our experiments. In this work we performed the feature-matching on the rectified images, but one can also do it directly on the original images [12].

3.2 Motion and Pose Estimation Algorithms

In our real experiments, we found that the planarity assumption holds for a short distance, but is violated for a long distance on most roads. For a camera moving on a plane with one degree of rotation, the epipole should always lie on the horizon. Note that the epipole can be computed by the intersection of

line segments joining the point correspondences from two images as shown in Figure 2(b). We computed the epipoles for several image pairs as shown in Figure 2(c) and found that the epipoles do not lie on the horizon line.

In all the video sequences, therefore, we compute the 3 DOF planar motion estimates for the first 20 images using the 2-point motion estimation algorithm, and then estimate the 6 DOF poses for the subsequent images using the standard 3-point camera pose estimation algorithm [7]. The initial camera poses given by our 2-point algorithm are used to triangulate the feature correspondences and obtain a sparse point cloud. As the ground plane is the dominant plane, we use homography to identify points on the ground. The reconstruction is updated to global scale using the knowledge of the ground plane in the camera coordinate system. Using this sparse reconstruction, we compute the 6 DOF poses of the subsequent images [7]. This partial sparse 3D point cloud is updated as new 3D points become available in the subsequent frames. Such a hybrid approach has been used in many real-time 3D reconstruction pipelines for large scale structure-from-motion problems [16]. In Figure 2(d), we show the motion estimated for a sequence of 3400 images.

3.3 Dense Depth Reconstruction

Given the camera poses, we compute a dense depth map at each frame using a plane-sweeping algorithm [4]. Plane sweeping provides a simple and efficient way to reconstruct a depth map using any number of images and their camera poses as the input. The algorithm is suitable for GPU implementation [25] and has been used for dense 3D reconstruction from vehicle-mounted cameras [6, 16].

In our implementation, we define a set of front-parallel planes with depths $d_i (i = 1, \ldots, D)$ in the coordinate system of the current frame. For each depth layer d_i, we project the current image and $N-1$ previous images using projective texture mapping on the GPU [25] and compute a matching cost $C(\mathbf{x}, d_i)$ for each pixel $\mathbf{x}$. As the matching cost, we compute the absolute intensity difference among all combinations of the N images for each pixel and take an average of the smallest 50% values, which makes the cost robust against occlusions [8]. We then smooth the cost in each depth layer with a small local window (11×11 pixels). We finally compute the optimal depth by simply finding the minimum cost for each pixel as $d(\mathbf{x}) = \arg\min_i C(\mathbf{x}, d_i)$.

We tested the plane-sweeping algorithm on several video sequences taken from both indoor and outdoor scenes as shown in Figure 3. We show that it is possible to generate depth maps from monocular video sequences. Such depth maps can be used for car-navigation applications like obstacle detection. Our algorithm is accurate enough to reconstruct small objects (10 cm wide poles and boxes of dimensions 30 cm) at close distances (less than 2 meters).

4 Conclusion

We presented a complete system for relative motion estimation and dense 3D reconstruction of nearby scenes from a monocular video sequence captured by an

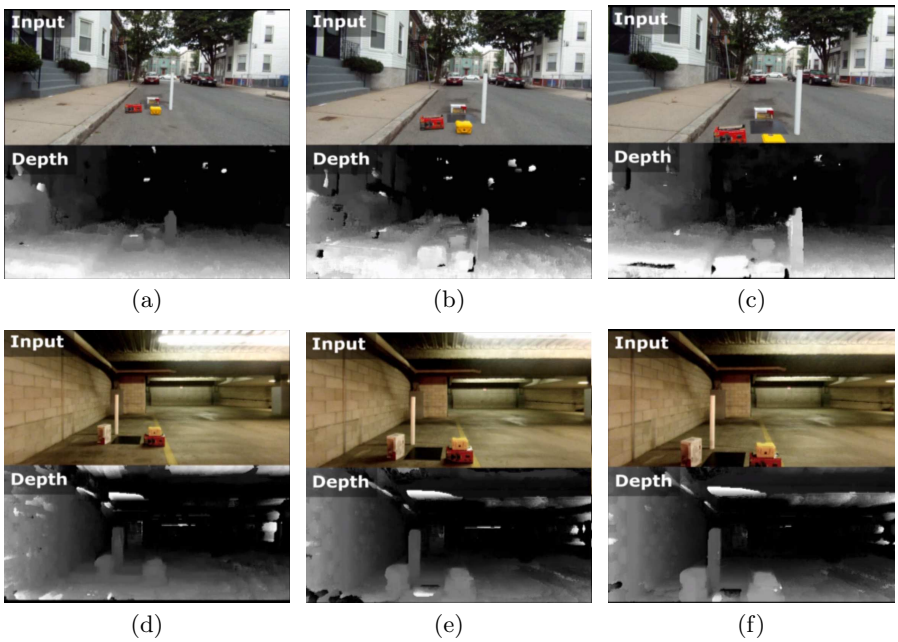

Fig. 3. *Dense depth reconstruction results. We tested video sequences consisting of* 100 *frames and computed a depth map at each frame using the plane-sweeping algorithm with N images and D depth layers. The distances of the objects vary from more than* 5 *meters to less than* 2 *meters. (a), (b) and (c) show the snapshots of the depth maps at three different places in the video for an outdoor scene (*$N = 20$ *and* $D = 50$*). (d), (e) and (f) show the depth maps for an indoor garage sequence (*$N = 30$ *and* $D = 80$*).*

omni-directional camera mounted on a car. We proposed a simple non-iterative solution for the planar 2-point relative motion estimation algorithm. Using a plane-sweeping algorithm along with the motion estimation, we compute a sequence of dense depth maps of the scene. Most of the code is written in Matlab and unoptimized. It currently takes 0.2 seconds for motion estimation, 0.4 seconds for KLT tracking, and 5 seconds for plane sweeping. We believe that the entire system can easily be made real time by using C++ and GPUs.

Acknowledgments: We would like to thank Jay Thornton, Amit Agrawal, Joseph Katz, Hiroshi Kage, Makito Seki, and Shintaro Watanabe for their valuable feedback, help and support. This work was done at and supported by MERL. Menglong Zhu contributed to the work while he was an intern at MERL.

References

1. O. Booij and Z. Zivkovic. The planar two point algorithm. In *IAS technical report IAS-UVA-09-05, University of Amsterdam*, 2009.

2. M. Chandraker, J. Lim, and D. Kriegman. Moving in stereo: Efficient structure and motion using lines. In *ICCV*, 2009.
3. B. Clipp, C. Zach, J. Frahm, and M. Pollefeys. A new minimal solution to the relative pose of a calibrated stereo camera with small field of view overlap. In *ICCV*, 2009.
4. R. T. Collins. A space-sweep approach to true multi-image matching. In *CVPR*, pages 358–363, June 1996.
5. F. Fraundorfer, P. Tanskanen, and M. Pollefeys. A minimal case solution to the calibrated relative pose problem for the case of two known orientation angles. In *ECCV*, 2010.
6. D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *CVPR*, 2007.
7. R. Haralick, C. Lee, K. Ottenberg, and M. Nolle. Review and analysis of solutions of the three point perspective pose estimation problem. *IJCV*, 13(3):331–356, 1994.
8. S. B. Kang, R. Szeliski, and J. Chai. Handling occlusions in dense multi-view stereo. In *CVPR*, volume 1, pages 103–110, 2001.
9. N. Karlsson, E. Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. Munich. The vslam algorithm for robust localization and mapping. In *ICRA*, 2005.
10. Z. Kukelova, M. Bujnak, and T. Pajdla. Automatic generator of minimal problem solvers. In *ECCV*, 2008.
11. H. Longuet-Higgins. A computer program for reconstructing a scene from two projections. *Nature*, pages 133–135, 1981.
12. M. Lourenco, J. Barreto, and F. Vasconcelos. srd-sift: Keypoint detection and matching in images with radial distortion. *IEEE Trans. on Robotics*, 2012.
13. R. Newcombe, S. Lovegrove, and A. Davison. Dtam: Dense tracking and mapping in real-time. In *ICCV*, 2011.
14. D. Nistér. An efficient solution to the five-point relative pose problem. In *CVPR*, volume 26, pages 756–770, 2003.
15. D. Ortin and J. Montiel. Indoor robot motion based on monocular images. *Robotica*, 2001.
16. M. Pollefeys, D. Nistér, J. M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles. Detailed real-time urban 3D reconstruction from video. *IJCV*, 78(2–3):143–167, July 2008.
17. S. Ramalingam, Y. Taguchi, T. K. Marks, and O. Tuzel. P2Π: A minimal solution for the registration of 3D points to 3D planes. In *ECCV*, 2010.
18. D. Scaramuzza. Performance evaluation of 1-point-ransac visual odometry. *Journal of Filed Robotics*, 2011.
19. D. Scaramuzza, F. Fraundorfer, and R. Siegwart. Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC. In *ICRA*, pages 4293–4299, May 2009.
20. D. Scaramuzza, A. Martinelli, and R. Siegwart. A toolbox for easy calibrating omnidirectional cameras. In *IROS*, 2006.
21. N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: Exploring image collections in 3D. *ACM Trans. Graphics*, 2006.
22. H. Stewenius, C. Engels, and D. Nister. Recent developments on direct relative orientation. *ISPRS J. of Photogrammetry and Remote Sensing*, 2006.
23. H. Stewenius, D. Nister, M. Oskarsson, and K. Astrom. Solutions to minimal generalized relative pose problems. In *OMNIVIS*, 2005.
24. J. Tardif, Y. Pavlidis, and K. Daniilidis. Monocular visual odometry in urban environments using an omnidirectional camera. In *IROS*, 2008.
25. R. Yang and M. Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware. In *CVPR*, volume 1, pages 211–217, June 2003.