A Method of Registering Virtual Objects in Monocular Augmented Reality System

Zeye Wu[1,2], Pengrui Wang[1], and Wujun Che[1,3(✉)]

[1] Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
{wuzeye2016,wujun.che}@ia.ac.cn
[2] School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 100049, China
[3] AICFVE of Beijing Film Academy, Beijing 100088, China

Abstract. A flexible novel method of registering virtual objects in monocular AR system is presented in this paper. Monocular AR systems use SLAM-related techniques to obtain the camera pose, of which the translation component is on a random scale. We add a scale calibration process to acquire the scale factor from the SLAM map to the real world and provide a closed-form solution of the transformation between two coordinate systems with different scales. We also describe the framework of an AR system based on our method with implementation. The proposed system can easily initialize virtual objects' position, orientation and size by using a known reference in the real scene and the reference is no longer needed in the later process. Our method is flexible, simple to set up and easy to control. The results show the proposed method can apply to real-time interactive AR applications.

Keywords: Augmented reality · Monocular SLAM · Virtual objects registration
Scale calibration

1 Introduction

Augmented Reality (AR) is a technology of overlaying virtual objects on real scenes which can interactive with the environment in real-time. It has huge application potential in the fields of medicine, military, education and entertainment [1–5]. AR registration consists of two aspects: tracking the observer's pose relative to the environment and correctly locating the virtual objects on the real scene.

Tracking is an essential process of AR and has been the most popular research topic of AR in recent years [6]. SLAM (Simultaneous Localization and Mapping) is an attractive option of tracking since it can provide accurate pose estimation in an unknown environment without any marker. Commercial AR solutions including Magic Leap [7] and ArKit [8] have applied SLAM-related techniques in their devices. One of the most convenient sensors for SLAM is a single camera due to its low cost, minimal size and wide deployment in personal terminals. Therefore, researchers have also been interested

Y. Wang et al. (Eds.): IGTA 2018, CCIS 875, pp. 483–493, 2018.
https://doi.org/10.1007/978-981-13-1702-6_48

in monocular visual SLAM-based AR [9–12]. Several monocular visual SLAM solutions have been suggested to use in AR including MonoSLAM [13], PTAM [10], ORB-SLAM [14], RKSLAM [11].

In marker-less monocular SLAM-based AR, a common way to locate virtual objects in the environment is to detect planar surfaces from point clouds recovered by SLAM and register virtual objects to random or manually-selected positions on these surfaces [12, 15, 16]. It makes it inconvenient to register virtual objects accurately.

Besides, such methods have to face the scale ambiguity problem which is inherent for monocular visual SLAM. Since the estimated map by monocular SLAM usually has an unknown scale, it cannot provide sufficient information for registration. For example, when supposed to place an avatar of actual human size, the monocular AR system may display a mini-size avatar. In that case, manual fine-tuning is needed to obtain the exact desired result. To solve the ambiguity, it has been popular to fuse visual camera with extra sensors including IMU or depth sensors [17–20], but these solutions undermine the largest advantages of monocular configuration in equipment size and power cost. As for vision-only solutions, additional prerequisites were introduced to calibrate the scale, including non-holonomic constraints [21], a planar road assumption for indoor or on-road scenes [22, 23]. A closed-form solution using a 2D-3D correspondence was given in [24]. [25] proposed a metric monocular SLAM by initializing the system's scale with a known chessboard pattern.

The main contributions of our work are summarized as follows. First, we propose a registration method with a scale calibration process for monocular AR by combining SLAM-based tracking with marker-based object localization. The scale calibration consists of a theoretical solution and a random sample optimization. Second, we present a monocular AR system based on the proposed registration method. It can flexibly overlay a virtual object in real scenes with easy control of its actual position, orientation and size. Our method is also applied to an interactive AR application to demonstrate its validness and effectiveness.

The paper is organized as follows. Section 2 describes the registration method. Section 3 describes a monocular AR system with the proposed registration process embedded in it. Section 4 provides the experimental results and the conclusion is given in Sect. 5.

2 Method of Scale Calibration and Registration

Visual monocular SLAM has two basic process: tracking and mapping. The tracking process computes the trajectory of the camera through unipolar geometry model, however, losing the depth information of the image points [26]. For example, if scaling up the scene and the motion of the camera at the same time, the observation of the camera will not change. Therefore, the depth of real-scene points is estimated up to a scale factor in mapping process.

We compute camera pose again by solving a PnP problem based on Iterative algorithm [27]. The 3D coordinates of four coplanar points in the real scene are needed so that we can obtain a camera translation with the same scale as the scale of the real world.

The translation information is used to calibrate the scale factor of SLAM coordinate system.

2.1 Relationship Between Coordinate Systems with Different Scales

The coordinate system built by SLAM is denoted by C_s. The coordinate system of the coplanar points is denoted by C_p. Suppose that the camera moves from position 1 to position 2. Figure 1 illustrates the relationship between the coordinate systems mentioned above. By Iterative algorithm, we can solve the two camera poses relative to C_p and the camera coordinate systems in position 1 and position 2 are denoted by C_1 and C_2 respectively. An arbitrary 3D point in coordinate system C_p is denoted by $X_p = (x_p, y_p, z_p)$, and in coordinate system C_1 it is denoted as $X_1 = (x_1, y_1, z_1)$, in coordinate system C_2 it is denoted as $X_2 = (x_2, y_2, z_2)$. The projective coordinates of X_p is denoted by $\overline{X}_p = (x_p, y_p, z_p, 1)$ and similarly, the projective coordinates of X_1 by $\overline{X}_1 = (x_1, y_1, z_1, 1)$, the projective coordinates of X_2 by $\overline{X}_2 = (x_2, y_2, z_2, 1)$. The relationship between $\overline{X}_p$ and $\overline{X}_1$ is expressed as

$$[R_1|T_1]\overline{X}_p = \overline{X}_1. \tag{1}$$

where $[R_1|T_1]$ is the rotation and translation from C_p to C_1.

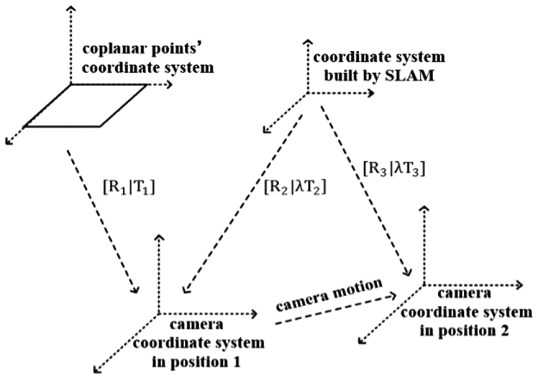

Fig. 1. An explanation of the relationships between the coordinate systems.

The inverse matrix of the camera poses computed by the SLAM system in position 1 and 2 are denoted by $[R_2|T_2]$ and $[R_3|T_3]$ respectively. Let λ denote the scale factor between C_s and the real world coordinate system C_p. The 'real' translation from C_s to C_1 and C_2 are λT_2 and λT_3 respectively. So we have

$$[R_2|\lambda T_2]\overline{X}_s = \overline{X}_1, \tag{2}$$

$$[R_3|\lambda T_3]\overline{X}_s = \overline{X}_2 \tag{3}$$

where $\overline{X}_2$ is the projective coordinates of the 3D point in C_2.

From (1) and (2), we have

$$\begin{aligned}\overline{X}_s &= \left[R_2|\lambda T_2\right]^{-1}\left[R_1|T_1\right]\overline{X}_p = \left[R_2^{-1}|-\lambda T_2\right]\left[R_1|T_1\right]\overline{X}_p \\ &= \left[R_2^{-1}R_1|-\lambda T_2+T_1\right]\overline{X}_p.\end{aligned}$$

Combining with (3), we have

$$\begin{aligned}\overline{X}_2 &= \left[R_3|\lambda T_3\right]\left[R_2^{-1}R_1|-\lambda T_2+T_1\right]\overline{X}_p \\ &= \left[R_3R_2^{-1}R_1|\lambda T_3-\lambda T_2+T_1\right]\overline{X}_p.\end{aligned} \tag{4}$$

2.2 Registering a Virtual Point in an Augmented Image

In formula (4), the scale factor λ is the only unknown parameter. Supposing we have measured the value of λ by some means, $\overline{X}_2$ can be computed using formula (4). Knowing $\overline{X}_2$, the pixel coordinates (u, v) of point X_p are obtained by applying the camera pinhole model:

$$z_2\begin{bmatrix}u\\ v\\ 1\end{bmatrix} = K\overline{X}_2 = \begin{bmatrix}f_x & 0 & c_x & 0\\ 0 & f_y & c_y & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x_2\\ y_2\\ z_2\\ 1\end{bmatrix}, \tag{5}$$

where K is the augmented camera intrinsic matrix.

Consequently, formula (4) and (5) can determine the pose of a virtual object in the SLAM coordinate system by its pose in the world coordinate system.

2.3 Solving the Scale Factor with Optimization

We are now solving the scale factor λ by computing the camera's translation in the coordinate system C_p and C_s. When the camera moves from one position to another, we can obtain two camera poses in C_p with translation vector T and T′. The corresponding poses in C_s have translation vector t and t′, respectively. The scale factor λ makes the following formula true:

$$T - T' = \lambda\left(t - t'\right). \tag{6}$$

The above formulas provide a theoretical method to register a virtual object in the image by assigning a real position for it. However, physical instruments and SLAM system always have errors and deviations. We can refine the result by statistical means. First, to reduce errors caused by small distance, we sample the captured images according the camera translation from last sampled image. Specifically, we ensure that the real distance between two adjacent samples is on a certain range of (0.05 m, 0.2 m) so that we obtain a sequence of translations in C_p, denoted by $\{T_n\}$, and a sequence of

translations in C_s, denoted by $\{T'_n\}$. We compute other two sequences $\{S_n : S_n = T_{n+1} - T_n\}$ and $\{S'_n : S'_n = T'_{n+1} - T'_n\}$. The scale factor's sequence is expressed as $\left\{\lambda_n = \frac{S_n}{S'_n}\right\}$.

Supposing $\{\lambda_n\}$ has errors following a normal distribution, then $\{\lambda_n\}$ also follows a distribution denoted as $N(\mu, \sigma^2)$, where λ is an unknown constant. Take the average value μ_n of $(\lambda_i)_{i=1}^n$ as the n^{th} estimator of μ and the standard deviation σ_n of $(\lambda_i)_{i=1}^n$ as the n^{th} estimator of σ. $\lambda_i (i \in N, i \leq n)$ is seen as an outlier when it is out of the range of $(\mu_n - \sigma_n, \mu_n + \sigma_n)$. We seek to find a normal distribution so that it is a stable distribution with small deviation and also its confidence interval contains most of the sequence data. Therefore, a sampling algorithm inspired by RANSAC [28] is performed as illustrated in Algorithm 1. When the existing data cannot provide a satisfying result, take more samples as input to approximate the accurate result.

Algorithm 1: Random Sample Estimator

Input: Sequence $S = \{\lambda_i : i = 1, \ldots, n\}$
Loop:
  Compute the average value μ, the standard deviation σ of S;
  Update Sequence S by Sampling $S = \{\lambda_i : \lambda_i \in (\mu - \sigma, \mu + \sigma)\}$;
End Loop While ($\sigma <$ threshold or $\#S < 3$)
If ($\sigma <$ threshold & $\#S > 2n/3$)**:**
  Output μ;
Else:
  Add 10 inputs s.t. $S = \{\lambda_i : i = 1, \ldots, n + 10\}$;
  Back to Loop;
End If

3 AR System Based on the Proposed Method

In this section, we describe a monocular AR system with the proposed registration process. As shown in Fig. 2, it consists of four processes: monocular SLAM tracking, scale calibration, virtual object registration and rendering. We use the proposed method in the section above for scale calibration process and registration process.

SLAM Tracking. As mentioned in Sect. 2, monocular visual SLAM computes camera pose in a different scale from the scale of real world. In the AR system, we use ORB-SLAM system, which defines its 'unit length' by the distance between the first two keyframes [14]. We solve the problem by adding a scale calibration process.

Scale Calibration. In scale calibration, the system detects an ArUco marker in the real scene and extracts its four corner points as the four coplanar points [29]. The world coordinate system is defined as shown in Fig. 2c: the red, green and blue axes are the x axis, y axis, and z axis respectively; the origin is the left-top corner point of the marker; the coordinates of other 3 corner points is determined by its real distance relative to the

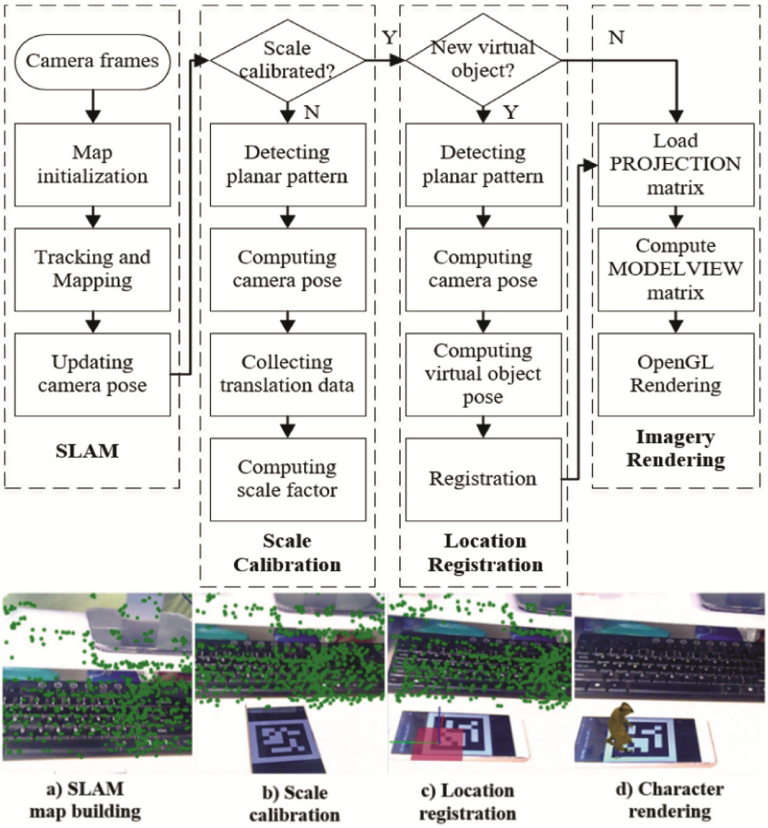

Fig. 2. The overview of our AR system with an image illustration for each process. The points in the image are map points. (Color figure online)

origin. With the four 2D-3D point correspondences, we use Iterative algorithm in [27] to compute an optimal camera pose relative to the marker.

At the same time, the system records the camera translation computed by SLAM module. The local average scale factor is computed based on formula (6) and the post process described in Sect. 2.2. If SLAM module has built the scene map with a closed loop optimization, i.e. has optimized the scale-drift problem, the scale calibration process only need to be carried out once. Otherwise, everytime a new virtual object is added to the scene, the scale calibration process should be performed. That is because the SLAM module has local optimization which makes the scale in a local map be consistent.

Virtual Object Registration. After the scale factor is determined, the ArUco marker is used to assign a real-scene position and orientation as where the virtual object should be. The concrete steps are the following:

(a) place the marker to the desired position,
(b) extracts its four corner points from the captured frame,

(c) solve the marker pose relative to the camera,
(d) compute the marker pose in SLAM coordinate system using formula (4).

This process ensures any virtual object has the same size in the rendering scene as it is in the marker's coordinate system, which means we can easily control the size of it. Once the marker pose is saved, the marker can move out of position. We can register one or more virtual objects by moving one marker to those appointed positions. Moreover, it is feasible to register the location of a moving marker in real-time.

Rendering. We use OpenGL to render virtual imagery over the real-scene image. The MODELVIEW matrix, which describes the relationship between the virtual object to be rendered and the camera, is the projective matrix of the marker pose saved in the registration process. The PROJECTION matrix is computed using the intrinsic parameters of the camera.

4 Experiments

First, we verify the validity of the proposed method by comparing the rendering results of the proposed AR system with and without scale calibration process. Then, we evaluate the registration accuracy of the proposed method by the reprojection errors. Finally, we apply the system to an AR application which can have real-time interaction with the real world. Experiments are run in an Intel i5-347, 3.2 GHz quad-core desktop with 24 GB of RAM and graphics card of NVIDIA GeForce GTX 670, under Windows 10 operating system. The camera frame input is at 30 Hz with resolution of 640 * 480 pixels. The SLAM module in the proposed system is implemented based on the open source code of ORB-SLAM.

4.1 Performance Analysis

Marker-based AR registration method is used as a baseline in our experiments. Marker-based AR usually detects a marker first, then computes the camera pose relative to the marker and finally renders a virtual object over the marker for every frame. We use both methods to render a surrounding rectangle of the marker in the augmented scene as shown in Fig. 3. The ground truth data, i.e. coordinates of the four corner points, are produced by directly detecting the four corner points with manual fine-tuning.

The registration error of one point is defined as the normalized distance between the real point's pixel-coordinates and the computed pixel-coordinates of the registered point. Suppose the pixel coordinate of one of the detected marker's corner is denoted by $P = (u_{pi}, v_{pi}), i = 1, 2, 3, 4$ and the pixel coordinate of the origin computed by our system is denoted by $Q = (u_{qi}, v_{qi}), i = 1, 2, 3, 4$. The registration error for one frame is defined as

$$\text{error} = \sum_{i=1}^{4} \text{sqrt}\left(\left(\frac{u_{pi} - u_{qi}}{w} \right)^2 + \left(\frac{v_{pi} - v_{qi}}{h} \right)^2 \right) / 4,$$

where w, h are the pixel width and height of the input image.

Fig. 3. Sample pictures of registering a surrounding rectangle of the marker in the augmented scene. The pink filled rectangle is registered using our method. The green rectangle is registered by marker-based method. (Color figure online)

Figure 4 shows a comparison of the registration errors of both methods when the marker can be observed. Our method performs better when the camera is moving away from the marker and rotating. In addition, our method still works when the marker disappears or is under occlusion. Figure 5 illustrates this effectiveness by registering a surrounding rectangle of the partly occluded marker, in which the keyboard is taken as the background texture for SLAM mapping. The marker is used to suggest the correct position. Under this condition, the marker-based method cannot detect the marker for registration process. Consequently, we make it more flexible to register a virtual moving character, i.e. once registered, even if the camera cannot observe the marker, the character can have a correct relationship with the environment when moving. See Sect. 4.2 for details.

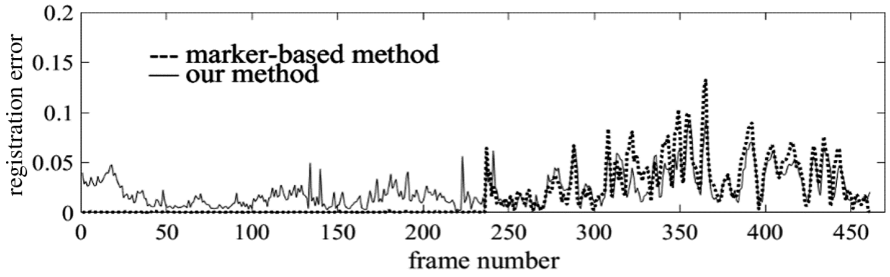

Fig. 4. The relative registration errors of 461 frames in the test. When taking the first 237 frames, the camera is close to its initial position and the marker-based method has a better accuracy. When the camera is moving away from the marker and rotating after 237th frame, our method does better in registration accuracy.

Fig. 5. The left-top image shows the initial position of the marker. The others show registering a surrounding rectangle of a partly occluded marker by our method, in which the marker-based method fails.

The runtime performance has also been tested. The average runtime of scale calibration is 1032.9 ms. Since the scale calibration process is performed only once, its runtime has little negative effect on the real-time rendering of virtual objects. It costs 21.22 ms to compute one virtual object's pose in average for current frame and the maximum computing time is 32 ms, which can satisfy the requirements of real-time (30 fps) applications.

4.2 Interactive Application

Figure 6 shows a simple AR application based on our system. The application can place a virtual character on a selected position in the real scene and control the character's walking directions in real-time using a marker. The character's motion control is based on the PFNN framework by [30]. When the application is running, it will load or build the scene map first. When camera is moving, the application seeks the marker in the

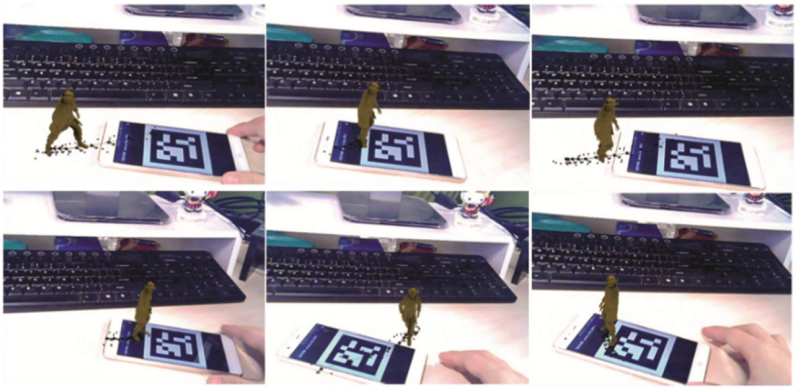

Fig. 6. Real-time interactive AR application using our registration method. The character walks towards the left-top corner point of the marker.

scene and automatically calibrate the scale factor. After that, stop the camera's moving for 2 s and the virtual character appears over the marker. Then the character starts to chase after the marker if user move the marker to another position.

5 Conclusion and Future Work

This paper has proposed a registration method for monocular SLAM-based AR system, which can conveniently register virtual objects in the augmented scene. We use a planar marker to calibrate the scale of the SLAM map and to control the position, orientation and size of the registered object. Different from marker-based AR which needs to capture markers for every frame, the proposed system only uses a marker for calibration and initial localization and no marker is needed for later location process. Therefore, it is more adaptive to dynamic augmented scenes. The proposed method is flexible, simple to set up and easy to control. Experimental results demonstrate that the proposed method provides real-time accurate registration results and can apply to interactive AR applications. Future work will consider using fingers or gestures to register and control virtual objects as a more convenient way of interaction based on the pipeline of the proposed method.

Acknowledgements. This work is supported by National Natural Science Foundation of China (No. 61471359) and National Key R&D Plan of China (No. 2016YFB1001404).

References

1. Kaufmann, H.: Construct3D: an augmented reality application for mathematics and geometry education. In: ACM Multimedia, pp. 656–657. (2002)
2. Bichlmeier, C., Sielhorst, T., Heining, S.M., Navab, N.: Improving depth perception in medical AR. In: Horsch, A., Deserno, T.M., Handels, H., Meinzer, H.P., Tolxdorff, T. (eds.) Bildverarbeitung für die Medizin 2007. Informatik aktuell, pp. 217–221. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71091-2_44
3. Livingston, M.A., et al.: Military applications of augmented reality. In: Furht, B. (ed.) Handbook of Augmented Reality, pp. 671–706. Springer, New York (2011). https://doi.org/10.1007/978-1-4614-0064-6_31
4. Pucihar, K.C., Coulton, P.: Exploring the evolution of mobile augmented reality for future entertainment systems. Conf. Comput. Eur. **11**, 1–16 (2013)
5. Carvalho, C.V.D.A., Lemos, B.M.: Possibilities of augmented reality use in mathematics aiming at a meaningful learning. Creat. Educ. **05**, 690–700 (2014)
6. Feng, Z., Duh, H.B.L., Billinghurst, M.: Trends in augmented reality tracking, interaction and display: a review of ten years of ISMAR. In: 2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality, pp. 193–202 (2008)
7. Detone, D., Malisiewicz, T., Rabinovich, A.: Toward geometric deep SLAM (2017)
8. ARKit Hardware and Software Integration. https://developer.apple.com/arkit/
9. Chekhlov, D., Gee, A.P., Calway, A., Mayol-Cuevas, W.: Ninja on a plane: automatic discovery of physical planes for augmented reality using visual SLAM. In: Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, pp. 1–4. IEEE Computer Society (2007)

10. Klein, G., Murray, D.W.: Parallel tracking and mapping for small AR workspaces. In: international symposium on mixed and augmented reality, pp. 225–234 (2007)
11. Liu, H., Zhang, G., Bao, H.: Robust keyframe-based monocular SLAM for augmented reality. In: 2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 1–10 (2016)
12. Xue, T., Luo, H., Cheng, D., Yuan, Z., Yang, X.: Real-time monocular dense mapping for augmented reality. In: Proceedings of the 2017 ACM on Multimedia Conference, pp. 510–518. ACM, Mountain View (2017)
13. Davison, A.J., Reid, I.D., Molton, N., Stasse, O.: MonoSLAM: real-time single camera SLAM. IEEE Trans. Pattern Anal. Mach. Intell. **29**, 1052–1067 (2007)
14. Murartal, R., Montiel, J.M.M., Tardos, J.D.: ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE Trans. Rob. **31**, 1147–1163 (2015)
15. Concha, A., Civera, J.: DPPTAM: dense piecewise planar tracking and mapping from a monocular sequence. In: Intelligent Robots and Systems, pp. 5686–5693 (2015)
16. Gao, Q.H., Wan, T.R., Tang, W., Chen, L., Zhang, K.B.: An improved augmented reality registration method based on visual SLAM. In: Tian, F., Gatzidis, C., El Rhalibi, A., Tang, W., Charles, F. (eds.) Edutainment 2017. LNCS, vol. 10345, pp. 11–19. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65849-0_2
17. Lupton, T., Sukkarieh, S.: Removing scale biases and ambiguity from 6DoF monocular SLAM using inertial. In: International Conference on Robotics and Automation, pp. 3698–3703 (2008)
18. Kim, O., Kang, D.: A sensor fusion method to solve the scale ambiguity of single image by combining IMU. In: International Conference on Control and Automation, pp. 923–925 (2015)
19. Nutzi, G., Weiss, S., Scaramuzza, D., Siegwart, R.Y.: Fusion of IMU and vision for absolute scale estimation in monocular SLAM. J. Intell. Rob. Syst. **61**, 287–299 (2011)
20. Fujimoto, S., Hu, Z., Chapuis, R., Aufrere, R.: ORB-SLAM map initialization improvement using depth. In: International Conference on Image Processing, pp. 261–265 (2016)
21. Scaramuzza, D.: 1-point-RANSAC structure from motion for vehicle-mounted cameras by exploiting non-holonomic constraints. Int. J. Comput. Vis. **95**, 74–85 (2011)
22. Kitt, B., Rehder, J., Chambers, A., Schönbein, M., Lategahn, H., Singh, S.: Monocular visual odometry using a planar road model to solve scale ambiguity, pp. 43–48 (2011)
23. Song, S., Chandraker, M.: Robust scale estimation in real-time monocular SFM for autonomous driving. In: Computer Vision and Pattern Recognition, pp. 1566–1573 (2014)
24. Esteban, I., Dorst, L., Dijk, J.: Closed form solution for the scale ambiguity problem in monocular visual odometry. In: Liu, H., Ding, H., Xiong, Z., Zhu, X. (eds.) ICIRA 2010. LNCS (LNAI), vol. 6424, pp. 665–679. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16584-9_64
25. Li, Y., Wang, S., Yang, D., Sun, D.: Metric online monocular SLAM by using a known reference. In: World Congress on Intelligent Control and Automation, pp. 3002–3006 (2016)
26. Hartley, R.I., Zisserman, A.: Multiple view geometry in computer vision. Kybernetes **30**, 1333–1341 (2000)
27. Oberkampf, D., Dementhon, D., Davis, L.S.: Iterative pose estimation using coplanar feature points. Comput. Vis. Image Underst. **63**, 495–511 (1996)
28. Matas, J., Chum, O.: Randomized RANSAC with Td, d test. Image Vis. Comput. **22**, 837–842 (2004)
29. Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F.J., Marín-Jiménez, M.J.: Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recogn. **47**, 2280–2292 (2014)
30. Holden, D., Komura, T., Saito, J.: Phase-functioned neural networks for character control. ACM Trans. Graph. **36**, 1–13 (2017)