Sung and Lin *EURASIP Journal on Image and Video Processing* (2017) 2017:18
DOI 10.1186/s13640-017-0168-3

EURASIP Journal on Image and Video Processing

RESEARCH Open Access

A fast 3D scene reconstructing method using continuous video

Bo-Yi Sung and Chang-Hong Lin*

Abstract

Accurate 3D measuring systems thrive in the past few years. Most of them are based on laser scanners because these laser scanners are able to acquire 3D information directly and precisely in real time. However, comparing to the conventional cameras, these kinds of equipment are usually expensive and they are not commonly available to customers. Moreover, laser scanners interfere easily with each other sensors of the same type. On the other hand, computer vision-based 3D measuring techniques use stereo matching to acquire the cameras' relative position and then estimate the 3D location of points on the image. Because this kind of systems needs additional estimation of the 3D information, systems with real time capability often relies on heavy parallelism that prevents implementation on mobile devices.

Inspired by the structure from motion systems, we propose a system that reconstructs sparse feature points to a 3D point cloud using a mono video sequence so as to achieve higher computation efficiency. The system keeps tracking all detected feature points and calculates both the amount of these feature points and their moving distances. We only use the key frames to estimate the current position of the camera in order to reduce the computation load and the noise interference on the system. Furthermore, for the sake of avoiding duplicate 3D points, the system reconstructs the 2D point only when the point shifts out of the boundary of a camera. In our experiments, we show that our system is able to be implemented on tablets and can achieve state-of-the-art accuracy with a denser point cloud with high speed.

Keywords: 3D reconstruction, 2D to 3D, Point cloud, Mono camera, Mono vision

1 Introduction

The 3D reconstructing techniques have been widely promoted in these years. These 3D techniques have many ways of application, such as object modeling and 3D printing, architecture, robots, augmented reality, medical, archaeology, or just a 3D record as an alternative of camera photos. There are many 3D acquiring methods, such as ultrasound [1], synthetic aperture radar (SAR) [2], and the most famous LIDAR system [3].

Recently, the 3D applications based on the RGB-D camera (IR laser) are improved dramatically, such as methods using Microsoft Kinect [4, 5] and ASUS Xtion [6, 7], which are capable to construct an exceedingly precise and dense 3D point cloud in real time with a GPGPU (general-purpose computing on graphics processing units) assist. However, since the RGB-D camera and laser scanners are not commonly available products, at the cost of implementing depth-estimating methods, both stereo cameras [8–10] and mono cameras [11–15] are used to achieve the same functionality of a RGB-D camera.

Among these mono cameras, methods [12, 13] are designed for building a 3D point cloud with multiple 2D images. These methods usually took a long time to compute the relation between input images in order to acquire a large amount of cloud points with high accuracy. On the other hand, the SLAM (simultaneous localization and mapping) system [11] is designed for AR applications. Although the system can acquire accurate camera position in real time, the system only tracks very little amount of feature points so that the system is not quite suitable for reconstructing a 3D point cloud. Therefore, we seek a system that is able to build a 3D point cloud within real time.

In this paper, we present a fast 3D reconstructing system which is suitable for 3D scenery recording usage. Considering the structure from motion methods [12, 13], most of

* Correspondence: chlin@mail.ntust.edu.tw
Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology, No. 43, Sec. 4, Keelung Rd, Taipei 106, Taiwan

them take minutes or even hours to finish the work. While the 3D reconstruction of a vast area is not the requirement, such as building a whole city district, we concentrate at reconstructing a single scene at high speed. First, in order to improve the speed of the overall process, we apply the key frame selection technique, which can significantly reduce the processing time of pose estimation by removing duplicate information from neighbor frames in a video sequence. Second, we maintain the precision of the result while the overall computation time is reduced. Because we only cut the neighbor frames, which carry almost same information with each other, the data kept by system can be updated at the time when it is truly needed. Third, the proposed system avoids to reconstruct the duplicated points. The duplicated point appears because the instability of feature points, and this can be caused by the blur, heavy motion, or the illuminance change of the image. Avoiding duplicated points can reduce the time needed for triangulation and save a lot of space for memory in comparison with the system, which do not avoid the duplicated points.

By these improvements, the proposed system is able to be implemented on a single CPU. The computation speed achieves 5 to15 fps based on the size of the video sequence and the number of feature points.

2 Related works

Providing a better experience for users has been an active goal for multimedia research. Images can not only aggregate into a 3D scene but also animate themselves in order to provide more vivid and interesting visuals. This works nowadays are widely implemented in mobile devices, such as Apple Live Photos, Instagram Boomerang, and the "Cinemagraphs" project (available: http://cinemagraphs.com/). Most of them require capturing more than one photograph. However, there are studies that require only one still image to reconstruct the motion of an object in the photograph. A study [16] has carried out to create the cloud motion, which is much harder than animating the rigid objects. On the other hand, in order to reconstruct a 3D scene, localizing the object is critical even if the system uses the RGB-D camera. There are studies that dedicated in researching a more efficient way to track an object. The study [17] compares the performance of the system that uses early or the late fusion with SVM (support vector machine) or other deep learning classifiers. In the case of hand gesture recognizing, the research [18] uses the deep learning to enhance the system to track the moving hand with faster speed without losing precision. While this research is tracking a hand captured by a stationary camera, the 3D reconstructing methods require studies that track stationary objects captured by a moving camera, reversely. There exists a lot of work that reconstructs the 3D scene or tracks the camera positions. Since we are concentrated in the method of using mono cameras, only methods using mono visions will be briefly described in the following paragraphs.

One of the solutions that can reconstruct a dense depth map is segmenting a photo into superpixels [19, 20]. Based on the local appearance of the photo with the global constraints, they build the most likely 3D structure for each segment and use it to build the depth map. Even though their systems give excellent results, their results are still not precise enough and were restricted in a single view that is not suitable for building a scenery around an area.

Instead of building depth map for an image, the method that estimates the positions for a camera can also reconstruct the 3D points from the estimated camera positions. SLAM [11, 21] is the process that a system incrementally builds a consistent map of its environment and uses this map to compute its own location at the same time. Their methods not only work in real time but also maintain the position of the camera precisely. However, in order to keep these efforts simultaneously in real time, most of the approaches track only a few of feature points, while we are interested in a method that tracks the feature points as dense as possible.

Similar to the SLAM method, a method that also estimates the positions of the cameras from video was demonstrated by Akbarzadeh et al. [22]. The main purpose of their method is to reconstruct the 3D urban scene but not maintaining the camera track in the map. On the other hand, the 3D reconstruction with multiple photos that can reconstruct a scenery as large as the entire city of Rome was originally demonstrated by N. Snavely et al. [12, 13]. This kind of technique that processes photo sequences into a 3D point cloud by studying the coherence between photos is called structure from motion (SfM). This kind of method basically computes the relative camera positions between all related photos. After every relative camera position is found, the scheme uses these matrices to reconstruct all feature points using triangulation. Although the results of the method proposed by N. Snavely et al. [12, 13] were very impressive, their method requires a very long time to finish calculating all required matrices. Hence, the other SfM methods, which are VisualSFM [14] and OpenMVG [15], are proposed to improve the processing speed and the robustness of the system. VisualSFM [14] uses the preemptive feature matching, the incremental structure from motion and the re-triangulation techniques. The incremental feature matching can greatly speed up the process because this kind of matching will first sort all feature points and match only first h feature points for each photo. The scheme will not proceed to match whole feature points unless the number of successful matches among first h features is greater than a defined

threshold. Incremental structure from motion also saves the processing time due to not performing the bundle adjustment while a new camera was added. Instead, it performs the bundle adjustment only when number of points increase relatively by a certain ratio. The re-triangulation technique lowers the camera drift caused by the bad camera relative pose, which might have a low ratio between their common points. They re-triangulate these bad camera poses after a sufficient amount of data obtained from new added cameras. OpenMVG [15] also contains incremental structure from motion technique. Besides that, they proposed a new iterative sampling method called a contrario Random Sample Consensus (AC-RANSAC) as a substitution to the original RANSAC in order to acquire higher precision and better performance. The AC-RANSAC using the "a contrario" methodology in order to find a model that best fits the data with a threshold T that adapts automatically to the noise. Hence, it is able to find a model and its associated noise without a fixed threshold.

In this paper, we are interested in reconstructing the observed points as detailed as possible from the video sequence while losing very less real-time performance.

3 Proposed method

The goal of our method is to build a 3D point cloud in real time. We summarize our method in Fig. 1 as a flowchart. This method can be briefly divided into five procedures: feature processing loop, pose estimation, point tracking, triangulation, and bundle adjustment. We will describe each procedure in the following sections: 3.1 to 3.5, respectively.

3.1 Feature processing loop

In this section, the feature processing loop is described. This procedure first does all the process for feature detection and matching and iterates itself to search for the points found in the previous frame. In order to improve the efficiency, it also decides whether a frame is a key frame or not. The key frame would then be used in the motion data processing.

Among all feature detecting and describing algorithms, we tested several popular algorithms including SIFT [23], SURF [24], ORB [25], and the new A-KAZE [26] method. These methods are tested with maximum 5000 points in 250 different images sized 853×480, and the results are shown in Table 1. Based on the result, we found that all these methods are not well suited with our requirements. So we further tried the features from accelerated segment test (FAST) [27] feature detection method along with pyramidal L-K (Lucas-Kanade) feature-tracking method [28]. As the result, the computation speed of this combination fulfills the requirements and the method returns a great number of detected feature points while losing very little precision.

After the matched feature points are found from the feature detecting and matching processes, the average distance of each matched points is now calculated. In order to increase the overall frame rate, the most important trick is to reduce the effort on computing an unnecessary frame, for which the parallax (or baseline) of previous frame and current frame is very small. This kind of frame usually carries information that system has already known by previous frames, so skipping this frame will not only save time but also lose no important information. On the other hand, based on our inquisition, we choose a frame to be a key frame if and only if following situations are matched:

- The average feature distance in pixel must be more than ImageWidth/3.
- Feature point matches did not drop dramatically with respect to previous match.

The first situation makes sure that we will have big enough baseline between the previous and current frame. The second situation makes sure the key frame is not an abruptly moved camera scene, which might contain some motion blur and other noises. In the end, the matched point data will be saved when the key frame was selected. These matched data would then be used in the motion data processing procedure. Simultaneously, a new point is added while it is detected by FAST but not

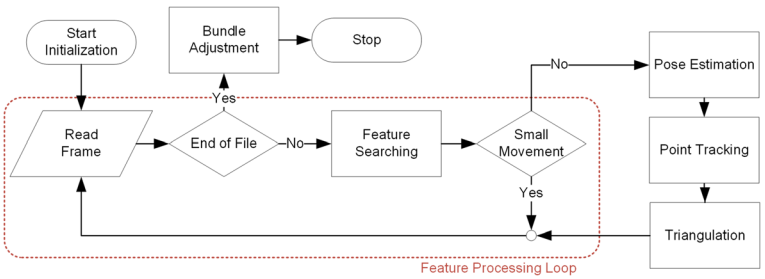

Fig. 1 The workflow of proposed method

Table 1 Results of feature processing methods

	Points	Ms/frame	Ms/point
SIFT [23]	1112	501.49	0.45
SURF [24]	2829	453.43	0.16
ORB [25]	4748	340.68	0.07
AKAZE [26]	1319	285.98	0.22
FAST [27] and L-K [28]	4284	33.71	0.01

found in L-K tracker. As illustrated in Fig. 2, this means that the newly found feature points from the current key frame are joined to the point vector. And the point vector will not be modified until the next key frame is selected.

3.2 Pose estimation

The pose estimation process estimates the camera poses based on the matched feature points from previous procedures. For pose estimation procedure, the projection matrix (or camera matrix) P will be calculated and saved along with the key frame. The projection matrix is used to denote a projective mapping from the world coordinate to a coordinate for a particular image in a pinhole camera.

$$\begin{bmatrix} q_1 \\ q_2 \\ 1 \end{bmatrix} = P_{3\times4} \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ 1 \end{bmatrix} \tag{1}$$

Where $Q = (Q_1, Q_2, Q_3, 1)$ is a representation of a 3D point in homogeneous coordinates and $q = (q_1, q_2, 1)$ is a representation of an image's corresponding point. The projection matrix P can be decomposed into an intrinsic matrix and an extrinsic matrix.

For the intrinsic matrix K, it describes the geometric property of a camera and projects 2D point from the camera coordinates to image coordinates. K is composed of focal length, principle point, and the skew parameter. In short, to get a projection matrix, it is necessary to get the intrinsic and extrinsic matrix in advance. Because intrinsic matrix K is only related to the camera setting, K can be acquired by calibrating the camera. In this case, the extrinsic matrix $[R|t]$, which denotes the coordinate transformations from 3D world coordinates to 3D camera coordinates by the rotation R and translation t, is what will be estimated in this section. The extrinsic matrix describes a camera's "pose" including camera's rotation, pan-and-tilt, and location c in the world coordinate. On the other hand, the fundamental matrix I is the algebraic representation of epipolar geometry. And the epipolar geometry is the projective geometry between two views. Therefore, every extrinsic matrix can be derived from knowing relation between cameras while assuming the first camera is located at the origin $[\mathrm{R}|t] = [\mathbf{I}|0]$.

We adopt the 8-point algorithm [29] to estimate fundamental matrix. Nevertheless, the fundamental matrix that was estimated will not be perfect not only because the 8 points chosen cannot be extremely accurate points, but also because this fundamental matrix is estimated using approximate solutions. Hence, given a current estimate of F, the Sampson Distance [29] d is calculated to estimate the reprojection error between the epipolar lines.

$$d = \sum_i \frac{\left(q_{i'^{\mathrm{T}}} F q_i\right)^2}{(Fq_i)_1^2 + (Fq_i)_2^2 + \left(F^{\mathrm{T}} q_i'\right)_1^2 + \left(F^{\mathrm{T}} q_i'\right)_2^2} \tag{2}$$

Where $(Fq_i)_j^2$ means the square of j^{th} entry of the vector Fq_i. The system iterates itself with random 8 matched points to estimate the fundamental matrix F and tries to search for F with the smallest d using Random Sample Consensus (RANSAC) robust estimation [30]. Since the fundamental matrix was estimated, an essential matrix [29] can be estimated by Eq. (3). Regarding a fundamental matrix, an essential matrix is a specialization of the fundamental matrix to the case of a

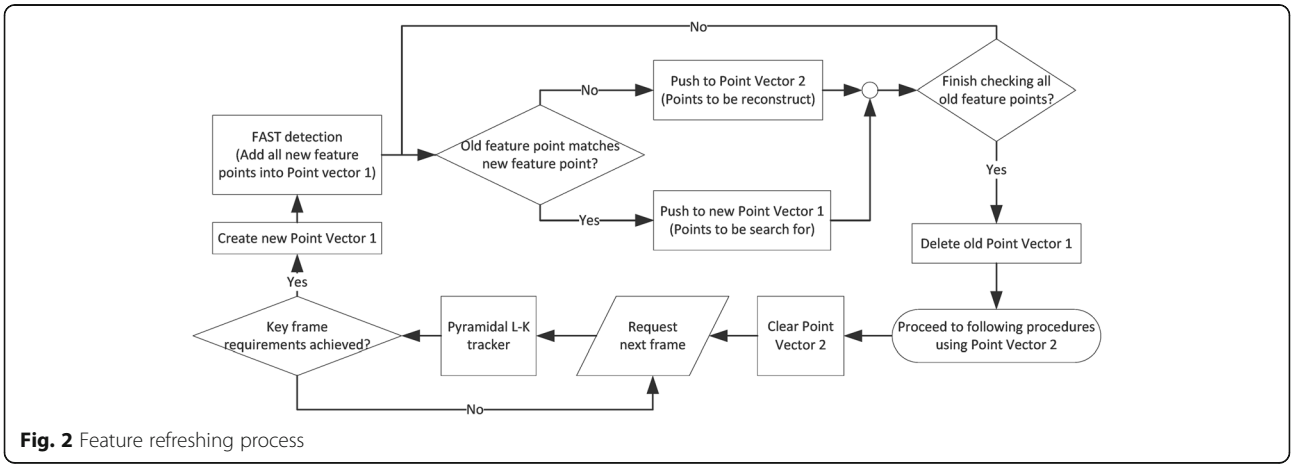

Fig. 2 Feature refreshing process

calibrated camera. Because it does not have projective ambiguity which fundamental matrix contains, it is better to get rotation and translation matrix $[R|t]$ by decomposing an essential matrix.

$$E = K'^{\mathrm{T}} F K \tag{3}$$

Once the essential matrix is determined, the camera rotation and translation matrix $[R|t]$ against the world coordinate can be retrieved. Because the essential matrix is composed of rotation and translation matrices, it can be expressed as:

$$E = [t]_{\times} R = \mathrm{SR} = \mathrm{U}\Sigma\mathrm{V}^{\mathrm{T}} \tag{4}$$

Where $\mathrm{U}\Sigma\mathrm{V}^{T}$ is the SVD of E and $[t]_{\times} = S$ is the 3×3 skew-symmetric matrix for the corresponding 3-vector t. The Hartley & Zisserman essential matrix decomposing method [29] is utilized to acquire $[R|t]$. It first defines an orthogonal matrix W and a skew-symmetric matrix Z:

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad Z = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5}$$

Then, S and R can be directly computed by these possible factorizations:

$$S = \mathrm{UZU}^{\mathrm{T}},\ R = \mathrm{UW}^{\mathrm{T}}\mathrm{V}^{\mathrm{T}}\ or\ \mathrm{UWV}^{\mathrm{T}} \tag{6}$$

Base on the essential matrix constraint [29], a 3×3 matrix is an essential matrix if and only if two of its singular values are equal and the third is zero. And in Eq. (4) $\mathrm{SR} = (\mathrm{UZU}^{\mathrm{T}})(\mathrm{UWV}^{\mathrm{T}}) = U(\mathrm{ZW})V^{\mathrm{T}}$, the singular values $\Sigma = diag(1, 1, 0)$ and $\Sigma = \mathrm{ZW}$ are true as required.

Corresponding to an essential matrix, there are four possible solutions for the extrinsic matrix because of two possible choices of R and the unknown sign of t. It means that the translation vector from the first to the second camera can be reversed and camera can have a rotation 180° about the baseline. In order to decide between all four solutions, it is sufficient to test with input points from previous procedure and see which solution reconstructed most points located in front of both cameras. Reconstruction is done by a simple triangulation, assuming the first camera locating at the origin $[\mathbf{I}|0]$ and the second located at $[R|t]$. Triangulation will be described in Section 3.4.

3.3 Point tracking

Point tracking keep tracks of every point found using a vector that stores tracks. As illustrated in Fig. 3, a track records the information of a feature point's color, 2D locations, and the key frame IDs. Every non-duplicate feature point detected by previous procedure will be saved as a track. Every coordinates of a feature point in key frames are stored in the 2D point vector respectively. In order to tell which projection matrix $P = K[R|t]$ is related to a single 2D point, the first frame ID and the last frame ID are also recorded. And the track also reserves an entry which stores 3D point coordinate. Therefore, it represents a feature point in the real world captured by a camera and will eventually be reconstructed into a 3D point in the world coordinate.

As illustrated in Fig. 4, for every point, the system checks if the point already exists in any track and categorizes each of them as either a new point or an old point. That is, the system checks whether the coordinate of an input point-match matches the newest point stored in any track or not. For example, there is a feature point-match q and its corresponding point q' in the previous key frame (first camera) and current key frame (second camera), respectively. If the point q exists in one of the tracks in the track vector, the point q' is considered as an old point which already has a track and the information of that track will be extended with the point. Extending a track means pushing the point q' into the 2D point vector in the containing track and updating its last frame ID to be current key frame ID. On the contrary, if the point q does not exist in the track, it is considered as a brand new point and the system will create a new track for the point. The new track first saves the color information of point q'. And the 2D points q and q' will be pushed into 2D point vector in an order with respect to previous key frame ID and current key frame ID. These IDs are saved in the first frame ID and the last frame ID, respectively. Finally, the new track is pushed into the track vector.

Besides categorizing every input point, the system also checks the track vector and finds the track which has not been touched in the categorizing step. The track that has not been touched means that its corresponding feature point does not found in the current key frame by the previous procedure in Section 3.1. In this case, we assume that the point has been shifted out of the boundary of the camera and will not appear any more. These untouched tracks will be erased from the vector and pushed to the next procedure in section 3.4 which does triangulation and reconstructs the 3D point.

3.4 Triangulation

The triangulation process reconstructs a 3D point from a pair of known cameras and the corresponding 2D points. Recall Eq. (1), although the wanted 3D point can be calculated directly by reprojecting the 2D point to 3D point, the solution will not be sufficiently correct because there are errors in the measured points q and q'. This means that, by reprojecting the points q and q', it usually does not exist that a point Q satisfies $q = PQ$ and $q' = P'Q$ simultaneously, where q' and P' represent the

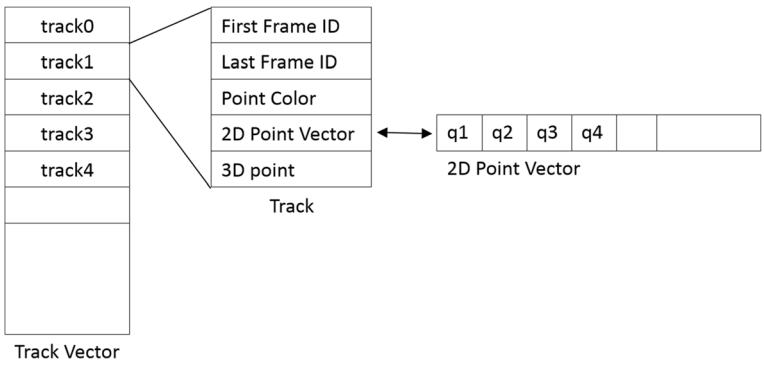

Fig. 3 Data structure of the track vector and a track

2D point and the projection matrix of the second camera. These points q and $q^{'}$ also do not sufficiently satisfy the epipolar constraint $q^{'T}Fq = 0$. Therefore, the direct linear transformation (DLT) [29] is proposed to achieve a closer solution to the ideal Q.

The DLT method combines $q = PQ$ and $q' = P'Q$ into a form of $YQ = 0$. First, it assumes:

$$q \times \mathrm{P}Q = \begin{bmatrix} q_1 \\ q_2 \\ 1 \end{bmatrix} \times \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ 1 \end{bmatrix} = 0 \tag{7}$$

This equation is true because from Eq. (1), we know $q = PQ$ and the cross product of q and q itself, which is $q \times q$ must be a zero vector. By expanding Eq. (7), it gives three equations:

$$\begin{aligned} (\mathrm{p}^{1\mathrm{T}}Q) - q_1(\mathrm{p}^{3\mathrm{T}}Q) &= 0 \\ q_2(\mathrm{p}^{3\mathrm{T}}Q) - (\mathrm{p}^{2\mathrm{T}}Q) &= 0 \\ q_2(\mathrm{p}^{1\mathrm{T}}Q) - q_1(\mathrm{p}^{2\mathrm{T}}Q) &= 0 \end{aligned} \tag{8}$$

Where p^{iT} means the transpose of i^{th} row of the projection matrix P. Hence, an equation of the form $YQ = 0$ can then be composed with:

$$Y = \begin{bmatrix} (\mathrm{p}^{1\mathrm{T}}) - q_1(\mathrm{p}^{3\mathrm{T}}) \\ q_2(\mathrm{p}^{3\mathrm{T}}) - (\mathrm{p}^{2\mathrm{T}}) \\ \left(\mathrm{p}'^{1\mathrm{T}}\right) - q_1'\left(\mathrm{p}'^{3\mathrm{T}}\right) \\ q_2'\left(\mathrm{p}'^{3\mathrm{T}}\right) - \left(\mathrm{p}'^{2\mathrm{T}}\right) \end{bmatrix} \tag{9}$$

The first two equations from (9) have been included for each camera, which provide totally four equations and four homogenous unknowns. This homogenous linear equation $YQ = 0$ can be solved by considering Q as a null space of Y. And because we are using the homogeneous coordinate system, the solution Q, which is a 4-vector, needs to be normalized so that the last coordinate of itself equals to one.

Since the parameters used by the triangulation procedure are measured point-match from Section 3.1 and the estimated projection matrix from Section 3.2, both of them may include noises. In this case, the reconstructed 3D point cannot be reconstructed at an ideal location. The reconstructed 3D point will locate within the area between the rays from the camera through the measured points. The more parallel of these rays become the larger of the shaded area it will be. This means that the small camera movements in all six directions may cause a poor triangulating solution. The small movement can mostly be solved by point tracking in Section 3.3 because we always use the first found key frame and the last found key frame of a feature point to do the

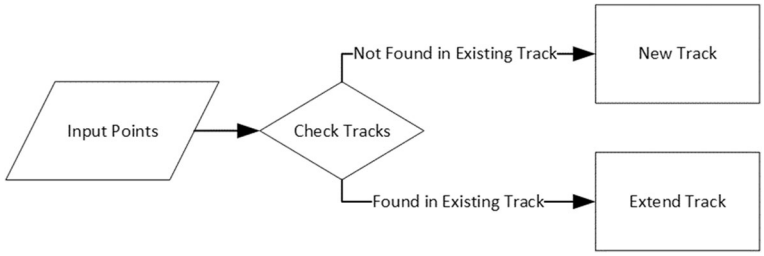

Fig. 4 Point checking checks for every input point

triangulation. This keeps the triangulation procedure that always uses the key frames with longest camera distance. However, the reconstructed 3D points might still contain noises. If the precision is the priority order, it is necessary to do a further refinement in the end of the program on these reconstructed 3D points.

3.5 Bundle adjustment

The bundle adjustment is a method that solves the problem of simultaneously refining the 3D coordinates, the parameters of the camera motion, and the characteristics of cameras. As described in Sections 3.2 and 3.4, if the image measurements are noisy, the camera poses are not flawlessly precise and the Eq. (1) $q = PQ$ will not be satisfied exactly. In this case, an optimization is needed to minimize the reprojection error between the image points of observed and predicted image points.

Consider a reconstructed scene which consisting a 3D point Q_j that is seen by the corresponding cameras with projection matrices P^i. These parameters are estimated from the measured 2D points q_j^i, which are corresponding to the j^{th} 3D point and measured by the i^{th} camera. If the image measurement noise is Gaussian, the bundle adjustment can be referred to a maximum likelihood estimator. The refined 3D points are represented by $\hat{Q_j}$, and the refined projection matrices are represented by $\hat{P}\,i$. These parameters minimize the image distance $d\left(\hat{q_j}\,i, q_j^i\right)$ between the homogeneous points $\hat{q_j}\,i$ and q_j^i. The $\hat{q_j}\,i$ is the reprojected 2D point calculated from $\hat{P}\,i\hat{Q_j}$ by Eq. (1).

$$\underset{\hat{P}i,\hat{Q_j}}{\operatorname{argmin}} \sum_{i,j}^{m,n} d\left(\hat{P}\,i\hat{Q}_j,\ q_j^i\right)^2 \tag{10}$$

The proposed method uses the Levenberg-Marquardt algorithm [29, 31] that replaced the Gauss-Newton iteration with the augmented normal equations:

$$\left(J^T J + \lambda I\right)\delta = -J^T \epsilon \tag{11}$$

Where I is the identity matrix and λ is the adjusting factor that varies from iteration to iteration. If error decreases, then λ gets smaller; otherwise, it gets larger.

Because all 2D points are not necessarily corresponded to every projection matrix, the algorithm became a sparse Levenberg-Marquardt algorithm. In the case of bundle adjustment, there are two sets of parameters, A and B, where A is related to projection matrices and B is related to 3D points.

$$A = \left[\frac{\partial \hat{q}}{\partial P}\right] \text{ and } B = \left[\frac{\partial \hat{q}}{\partial Q}\right] \tag{12}$$

Apply sets A and B into equation $J\delta = \epsilon$. It can get the form of:

$$[A|B] \times diag(1+\lambda) \times \left[\frac{\delta_P}{\delta_Q}\right] = \epsilon \tag{13}$$

Then the normal equation of $J\delta = \epsilon$, which is (14), will be solved under the form of:

$$\left[\begin{array}{c|c} A^T A & A^T B \\ \hline B^T A & B^T B \end{array}\right] \times diag(1+\lambda) \times \left[\frac{\delta_P}{\delta_Q}\right] = \left[\frac{A^T \epsilon}{B^T \epsilon}\right] \tag{14}$$

And it can be abbreviated into:

$$\begin{bmatrix} U & W \\ W^T & V \end{bmatrix} \left[\frac{\delta_P}{\delta_Q}\right] = \left[\frac{\epsilon_A}{\epsilon_B}\right] \tag{15}$$

Where $U = [A^T A] \times diag(1+\lambda)$, $V = [B^T B] \times diag(1+\lambda)$ and $W = [A^T B]$.

In the beginning of solving Eq. (15), both sides of the equation are multiplied on the left by $\begin{bmatrix} I & -WV^{-1} 0I \end{bmatrix}$, where it assumes V as an invertible matrix, resulting in:

$$\begin{bmatrix} U - WV^{-1}W^T & 0 \\ W^T & V \end{bmatrix} \left[\frac{\delta_P}{\delta_Q}\right] = \left[\frac{\epsilon_A - WV^{-1}\epsilon_B}{\epsilon_B}\right] \tag{16}$$

This step eliminates the top right block of the matrix $\begin{bmatrix} U & W \\ W^T & V \end{bmatrix}$, making the top half equation of (16) to be:

$$\left(U - WV^{-1}W^T\right)\delta_P = \epsilon_A - WV^{-1}\epsilon_B \tag{17}$$

Since the error vector can be calculated by computing the error between reprojected position from 3D point and the measured 2D point, δ_P is the only unknown in the equation. Hence, δ_P can be found by solving Eq. (17) and the value of δ_Q can be found by the bottom half equation of (16) while the value of δ_P is already known.

$$V\delta_Q = \epsilon_B - W^T \delta_P \tag{18}$$

After δ_P and δ_Q are computed, the parameters of P and Q are replaced by new $(P + \delta_P)$ and $(Q + \delta_Q)$, respectively. And there will be a new error vector ϵ which is computed from these new parameters. If the error decreases, the system scales the factor λ down and proceeds to the next iteration. Otherwise, it reverts the parameters to the old parameter values and tries again with the scaled up factor λ. In the end, the iteration continues until the error has minimized below the threshold or the maximum number of iterations is reached.

4 Experimental results

The system is tested with a surface tablet with intel Core i3-4020Y running at 1.5 GHz and a personal computer with an intel Core i7-4770 CPU running at 3.4 GHz. In the experiment, our system is able to work faster than 1 fps on tablet and 5 fps on PC depending on the size of input video, the amount of feature points, and the moving speed of the camera. We compare our method to other 3D reconstructing methods, the SfM methods [12, 14, 15]. These SfM methods focus on the precision, and it was designed for reconstructing a vast scenery. Hence, their methods always take minutes or even hours to finish the whole process. The first one we are going to test is the Bundler: SfM for unordered image collections [12, 13]. It is a well-known SfM method that can solve a large amount of images with different intrinsic parameters by checking the EXIF data of every photo. VisualSFM [14] and OpenMVG [15] are similar methods that can also solve a large amount of images, but with different feature detecting, matching, tracking, outlier removing, and distortion recovering technique. Furthermore, they also improve the speed with the multi-thread technique. On the contrary, in the case of using a continuous video taken by a single moving camera, we provide a much faster method running in a single thread that was able to acquire a compromise solution in real time.

There are four different video sequences that are used for testing. Two videos were taken indoors and the other two were taken outdoors. And these sequences were captured by a hand held camera, and every frame was extracted into PNG files. We first compare the timing results in Section 4.1 and then compare the result of 3D reconstruction in Section 4.2.

4.1 Timing results

The timing results for sequence 1 (illustrated in Fig. 5 in Section 4.2) using all frames as the input image are shown in Table 2. As the table shows, the OpenMVG runs very fast because their method can run with all 8 logical cores on our PC. In contrast, both Bundler and VisualSFM provided the results that with denser or more precise point clouds (Fig. 6) at the cost of long computation time. However, our method still runs faster than all others while the point cloud of our result (Fig. 6) is still precise and dense enough. In fact, our method can run fast because it is suited for the continuous video sequences. Therefore, since these SfM methods do not need a continuous video as the input, we further tested their methods with fewer inputs in order to achieve similar time cost with our method. As the timing result shown in Table 3, the computation times are dramatically reduced to less than 3 min in average. Although the timing result of OpenMVG is better than ours, our result of 3D reconstruction is way better than theirs according to test results in Section 4.2. Furthermore, our method is faster than Bundler and VisualSFM while our 3D reconstruction results are better than the result of the Bundler and VisualSFM in Section 4.2. On the other hand, compared with the proposed method using PC, we further tested our method using a tablet listed in Table 4. And in order to show the performance for the real-time usage, as listed in Table 5, our method works more than 5 fps on PC and 1 fps on a tablet.

Table 2 The timing results of using all frames (examined on PC)

	Bundler [12]	VisualSFM [14]	OpenMVG [15]	Proposed
Sequence 1	51 min 36 s	42 min 6 min	7 min 17 s	1 min 24 s

4.2 3D reconstruction results

The 3D reconstructed results are tested with four different video sequences from sequence 1 through sequence 4, and the results from each method are printed with the cloud of points in the following figures in this section.

Sequence 1 is an indoor video sequence, which contains 349 frames as illustrated in Fig. 5. The reconstructed point clouds are illustrated in Fig. 6. We can

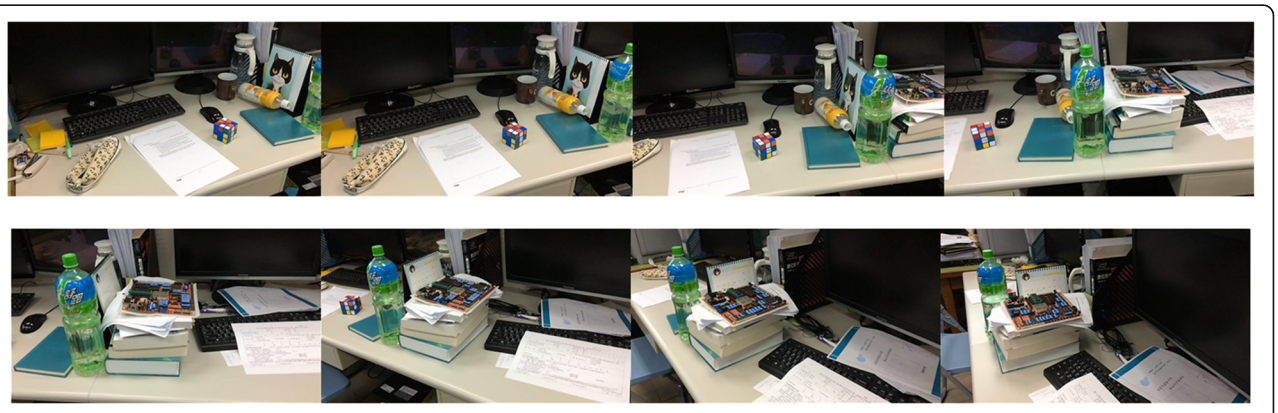

Fig. 5 Snapshot of sequence 1. Indoor, 349 frames, 13.9 s, 853 × 480, 16:9

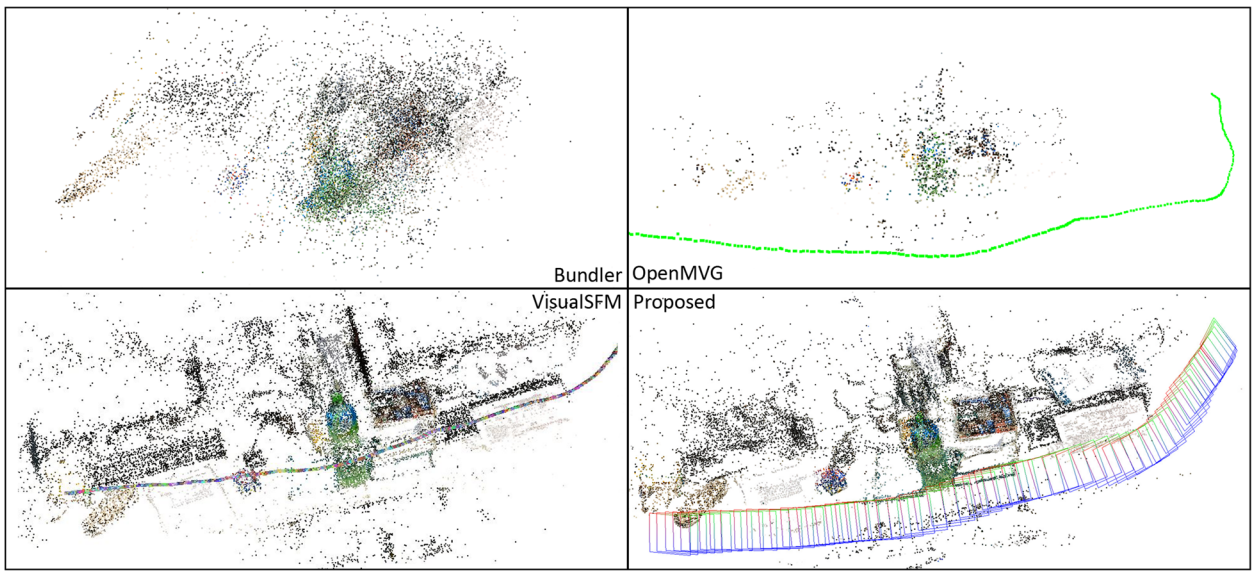

Fig. 6 The comparison of results for sequence 1

see that the results of Bundler and OpenMVG are indistinct, and the VisualSFM gives the rather precise result. On the contrary, the proposed method was also able to give the precise results.

As mentioned in Section 4.1, since these SfM methods do not need a continuous video as the input. We further tested their methods with fewer inputs in order to achieve similar time cost with our method in Fig. 7. So the SfM methods will use one fifth of all frames (70 frames) for sequence 1 as the input images. For the result of OpenMVG with 70 frames, it does not seem to change very much with fewer input images. And the result of the Bundler is too indistinct to distinguish between the stuffs placing on the table. On the other hand, VisualSFM suffers from fewer input images and the result of point cloud becomes sparser than theirs before. In this case, the result of our method is the clearest 3D point cloud than the others.

Table 3 The timing results of using partial frames and the proposed method using all frames (on PC)

	Bundler [12]	VisualSFM [14]	OpenMVG [15]	Proposed
Sequence 1	70 frames			349 frames
	5 min 52 s	2 min 20 s	34 s	1 min 24 s
Sequence 2	67 frames			534 frames
	2 min 57 s	2 min 18 s	32 s	1 min 12 s
Sequence 3	160 frames			1601 frames
	46 min 03 s	11 min 19 s	1 min 19 s	5 min 22 s
Sequence 4	33 frames			486 frames
	2 min 08 s	1 min 49 s	15 s	1 min 24 s

Sequence 2 is an indoor video as illustrated in Fig. 8. There is a finger blocking the left side of view in the video for 22 frames. As for SfM methods, the one eighth of 534 frames (67 frames) are used as the input base on the timing result (Table 3). And there are 3 fingers blocking the frames in those 67 frames.

The result for sequence 2 is printed in Fig. 9. There is a reason that the result of Bundler still gives the ambiguous result even with the manual-preset focal length. The results indicate that the Bundler might be heavily depended on the EXIF data which is included in the file of a photo from digital camera. And in our testing video sequences, there is no photo EXIF data that exists. As in Fig. 9, OpenMVG gives the precise but very sparse result while the result of VisualSFM is much denser than the OpenMVG. However, the result of proposed method is still the clearest among other results.

Sequence 3, as shown in Fig. 10, is an outdoor scene with a pond that can influence the feature-tracking methods. It is because the water may reflect the scenery with ripple and confuse the feature-matching methods. Similarly, there are one tenth of 1601 frames (160 frames) that are used as the input of the SfM methods. And for the results in Fig. 11, OpenMVG produces the scenery that was able to be recognized, but it produces the result with the wrong scale of the distances. The VisualSFM reconstructs a

Table 4 The timing results of the proposed method using all frames (on tablet)

	Sequence 1	Sequence 2	Sequence 3	Sequence 4
Timing	51 min 36 s	42 min 6 min	7 min 17 s	1 min 24 s

Table 5 Time cost per frame with all four video sequences

	Sequence 1		Sequence 2		Sequence 3		Sequence 4	
	PC	Tablet	PC	Tablet	PC	Tablet	PC	Tablet
3D points	18,129	15,561	15,057	22,841	35,717	36,528	39,682	36,012
Feature processing (ms/frame)	72.94	211.95	56.25	204.81	171.20	589.85	122.27	390.18
Pose estimation (ms/keyframe)	117.88	170.85	549.53	2300.3	120.28	286.22	135.42	352.04
Point tracking (ms/keyframe)	57.45	81.92	51.59	166.38	45.26	169.89	64.75	158.944
Triangulation (ms/keyframe)	17.07	53.67	6.62	18.06	20.94	81.60	33.96	55.5213
Average FPS	11.06	4.17	8.21	2.02	5.34	1.57	6.68	2.18

Note that only feature processing procedure will work at every frame and other three procedures will work while the key frame was selected

precise and a large area of results while our result provides a denser result.

Sequence 4 is an outdoor video with the forward camera movement as illustrated in Fig. 12. There are 33 frames used as the input images for SfM methods (one fifteenth of 486 frames). As the result in Fig. 13, both VisualSFM and OpenMVG give the precise results. Nevertheless, the OpenMVG gives sparser results than the VisualSFM, and our proposed method is the densest result with precision among them.

5 Discussion

Based on the inquisition, methods that only use camera work slower than the methods with additional equipment because these vision-based methods need time to calculate the depth information. For mono vision systems, the most important issues are the precision and the speed because the actual camera position can be only estimated from changes in camera frames. In this case, most of the SfM methods [12, 15] use the famous SIFT feature detecting and describing method that costs a lot more time to process, and this is the reason why SLAM [11] systems usually use FAST corner detector that can help them achieve the real-time performance. The SLAM system maintains their robustness by loop closure detection using the map they built. For the proposed method, we also use the FAST corner detector to accelerate the process. Moreover, different from SLAM system, the proposed method did not build the map because doing so usually need to parallelize the program that is more unsuitable to implement on the mobile devices. Since our main priority is the processing speed and the density of points, we simply maintain the precision by carefully choosing the key frame, so the proposed method's available building area will be smaller than SfM or SLAM methods. However, among the systems building the point cloud, the proposed method uses a continuous video, and it can process very fast and maintain precision in a certain area.

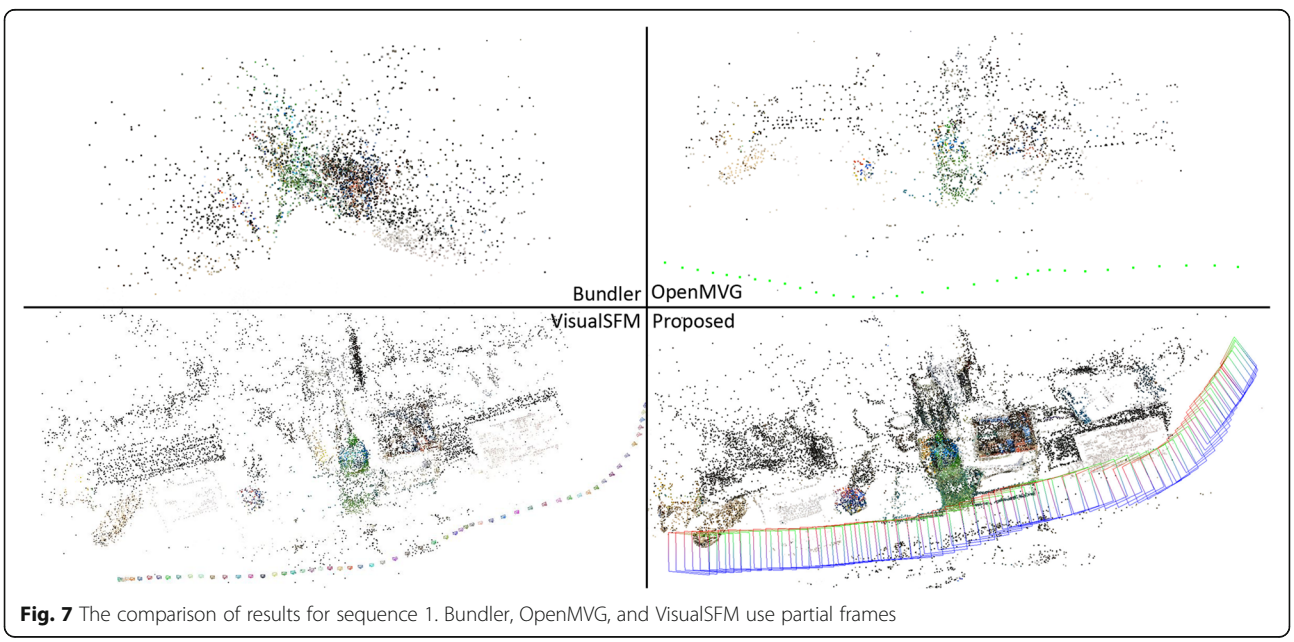

Fig. 7 The comparison of results for sequence 1. Bundler, OpenMVG, and VisualSFM use partial frames

Fig. 8 Snapshot of sequence 2. Indoor, 534 frames, 17.8 s, 853 × 480, 16:9

6 Conclusions

Over the years, the computer vision community has contributed many efforts improving the quality of the reconstructed 3D point cloud. As part of this effort, we have demonstrated a system that generates an accurate point cloud with high speed. Comparing to other existing 3D reconstructing methods, we are able to use only a mono video sequence as an input on a single CPU and reconstruct the 3D point cloud as dense as possible.

The most critical part for a mono 3D reconstructing method is the heavy load on estimating the camera position. Unlike the stereo system, the camera position can only be guessed from the projection between frames. The proposed method is able to lower

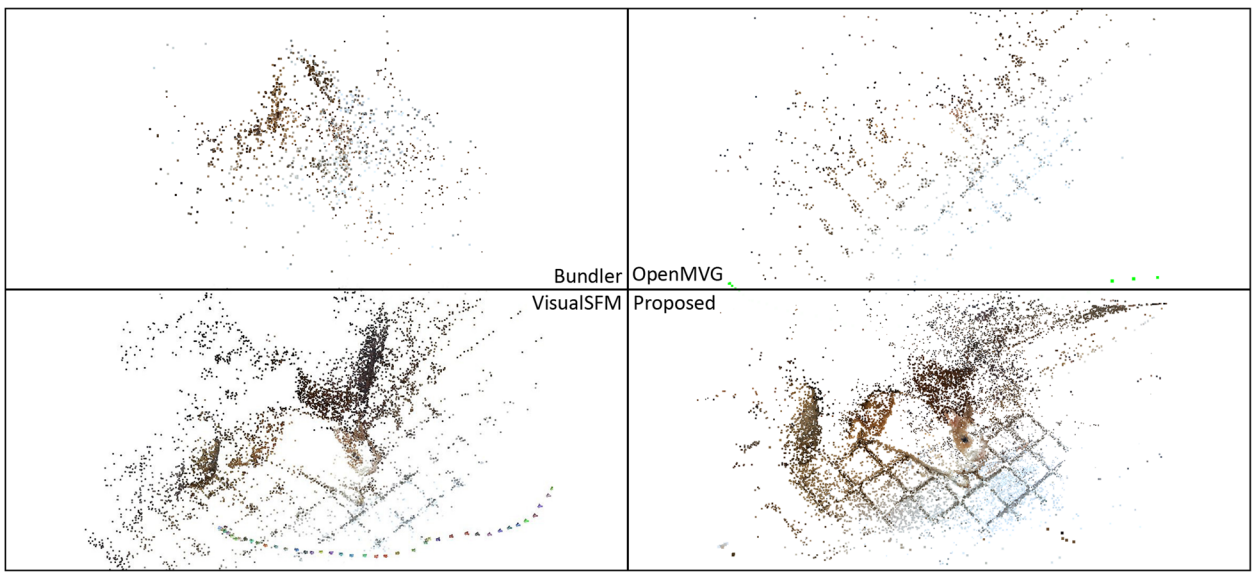

Fig. 9 The comparison of results for sequence 2. Bundler, OpenMVG, and VisualSFM use partial frames

Fig. 10 Snapshot of sequence 3. Outdoor, 1601 frames, 53.4 s, 853 × 480, 16:9

the load on estimating camera position while losing very little precision. Furthermore, with the lack of camera baseline as a reference, the estimated camera position is usually gained not only with noise but also the ambiguous scale between the pixels and the real world. In this case, despite that the proposed system is able to reconstruct a scenery within an area, this system will also encounter some scale drift while the video sequence was recorded along a very long distance.

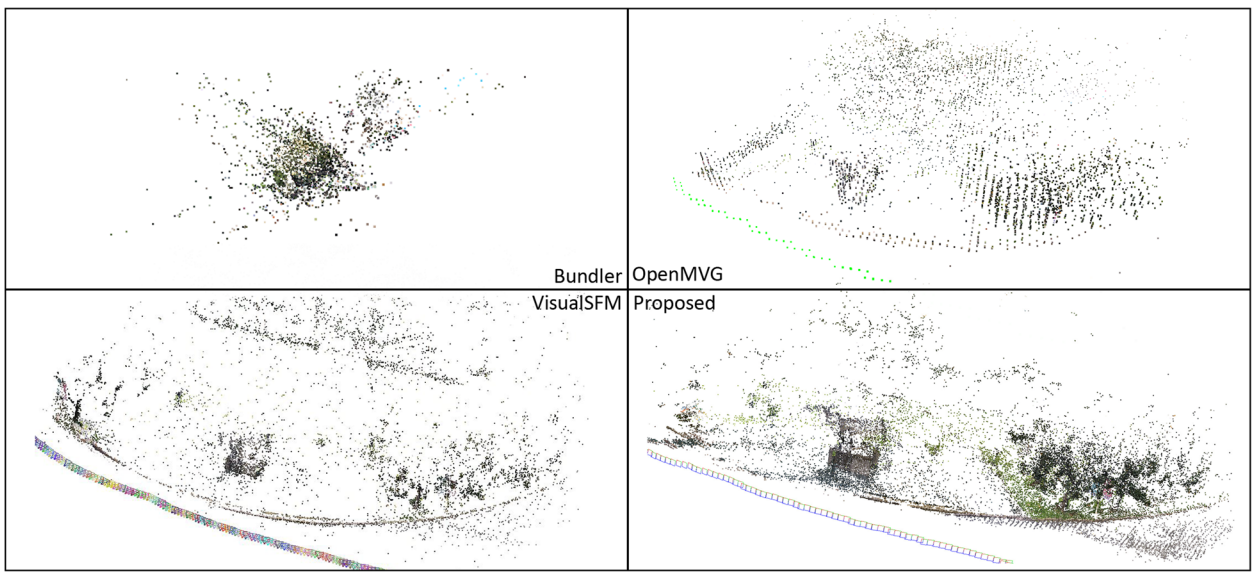

Fig. 11 The comparison of results for sequence 2. Bundler, OpenMVG, and VisualSFM use partial frames

Fig. 12 Snapshot of sequence 4. Outdoor, 486 frames, 16.2 s 853 × 480, 16:9

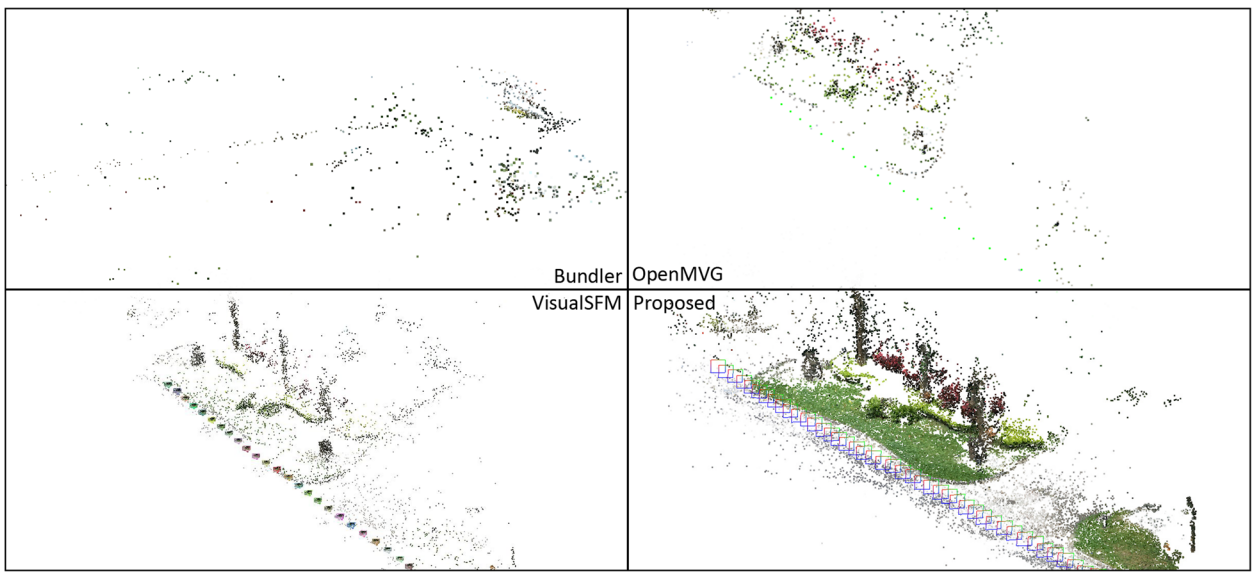

Fig. 13 The comparison of results for sequence 2. Bundler, OpenMVG, and VisualSFM use partial frames

Last but not least, we hope that this kind of fast and accurate 3D reconstructing algorithm can be promoted and become a readily available tool for artist, architect, engineer, and everyone whoever wants to build a 3D scenery.

Acknowledgements
The authors would like to thank the Ministry of Science and Technology in Taiwan for supporting this research under the project MOST104-2220-E-011-001-.

Authors' contributions
B-YS carried out the algorithm studies, platform implementation and the simulation and drafted the manuscript. C-HL participated in the algorithm studies and helped to draft the manuscript. Both authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Received: 2 August 2016 Accepted: 8 February 2017
Published online: 22 February 2017

References

1. A Fenster, DB Downey, Fast parametric elastic image registration. IEEE Eng Med Biol Mag **15**(6), 41–51 (2002)
2. JM Lopez-Sanchez, J Fortuny-Guasch, 3-D radar imaging using range migration techniques. IEEE Trans. Antennas Propag. **48**(5), 728–737 (2002)
3. B Douillard, J Underwood, N Kuntz, V Vlaskine, A Quadros, P Morton, A Frenkel, *On the Segmentation of 3D LIDAR Point Clouds* (IEEE International Conference on Robotics and Automation, Shanghai, 2011), pp. 2798–2805
4. F Endres, J Hess, N Engelhard, J Sturm, *An Evaluation of the RGB-D SLAM System* (IEEE International Conference on Robotics and Automation (ICRA), Saint Paul, 2012), pp. 1691–1696
5. J Chen, D Bautembach, S Izadi, Scalable real-time volumetric surface reconstruction. ACM Trans. Graph. - SIGGRAPH 2013 Conference Proceedings **32**(4), 1–16 (2013)
6. Q-Y Zhou, V Koltun, Dense scene reconstruction with points of interest. ACM Trans. Graph. - SIGGRAPH 2013 Conference Proceedings **32**(4), 1–8 (2013)
7. M Nießner, M Zollhöfer, S Izadi, M Stamminger, Real-time 3D reconstruction at scale using voxel hashing. ACM Trans. Graph. - Proceedings of ACM SIGGRAPH Asia **32**(6), 1–11 (2013)
8. SM Seitz, B Curless, J Diebel, D Scharstein, *A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms*. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, pp. 519–528
9. B Micusik, J Kosecka, *Piecewise Planar City 3D Modeling from Street View Panoramic Sequences* (IEEE Conference on Computer Vision and Pattern Recognition, Miami, 2009), pp. 2906–2912
10. A Geiger, J Ziegler, C Stiller, *StereoScan: Dense 3d Reconstruction in Real-time* (IEEE Intelligent Vehicles Symposium, Baden-Baden, 2011), pp. 963–968
11. G Klein, D Murray, *Parallel Tracking and Mapping for Small AR Workspaces*. ISMAR '07 Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, 2007, pp. 1–10
12. N Snavely, SM Seitz, R Szeliski, Modeling the world from internet photo collections. Int. J. Comput. Vis. **80**(2), 189–210 (2008)
13. S Agarwal, Y Furukawa, N Snavely, I Simon, B Curless, SM Seitz, R Szeliski, Building Rome in a day. Commun. ACM **54**(10), 105–112 (2011)
14. C Wu, *Towards Linear-Time Incremental Structure from Motion* (International Conference on 3D Vision, Seattle, 2013), pp. 127–134
15. P Moulon, P Monasse, R Marlet, *Global Fusion of Relative Motions for Robust, Accurate and Scalable Structure from Motion* (IEEE International Conference on Computer Vision, Sydney, 2013), pp. 3248–3255
16. W-C Jhou, W-H Cheng, Animating still landscape photographs through cloud motion creation. IEEE Trans. Multimedia **18**(1), 4–13 (2016). doi:10.1109/TMM.2015.2500031
17. J Sanchez-Riera, K-L Hua, Y-S Hsiao, T Lim, SC Hidayati, W-H Cheng, A comparative study of data fusion for RGB-D based visual recognition. Pattern Recogn. Lett. **73**, 1–6 (2016)
18. J Sanchez-Riera, Y-S Hsiao, T Lim, K-L Hua, W-H Cheng, *A Robust Tracking Algorithm for 3D Hand Gesture with Rapid Hand Motion Through Deep Learning* (IEEE International Conference on Multimedia and Expo Workshops (ICMEW), Chengdu, 2014), pp. 1–6. doi:10.1109/ICMEW.2014.6890556
19. A Saxena, M Sun, AY Ng, *Learning 3-D Scene Structure from a Single Still Image* (IEEE 11th International Conference on Computer Vision, Rio de Janeiro, 2007), pp. 1–8
20. A Gupta, AA Efros, M Hebert, *Blocks World Revisited: Image Understanding Using Qualitative Geometry and Mechanics*. Computer Vision - ECCV 2010: 11th European Conference on Computer Vision, 2010, pp. 482–496
21. AJ Davison, ID Reid, ND Molton, O Stasse, MonoSLAM: real-time single camera SLAM. IEEE Trans Pattern Anal Mach Intell **29**(6), 1052–1067 (2007)
22. A Akbarzadeh, J-M Frahm, P Mordohai, B Clipp, C Engels, D Gallup, M Pollefeys, *Towards Urban 3D Reconstruction from Video* (Third International Symposium on 3D Data Processing, Visualization, and Transmission, Chapel Hil, 2006), pp. 1–8
23. DG Lowe, Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**, 91–110 (2004)
24. H Bay, T Tuytelaars, LV Gool, Speeded-up robust features (SURF). Comput. Vis. Image Underst. **110**(3), 346–359 (2008)
25. E Rublee, V Rabaud, K Konolige, G Bradski, *ORB: an Efficient Alternative to SIFT or SURF*. International Conference on Computer Vision (IEEE, Barcelona, 2011), pp. 2564–2571. doi:10.1109/ICCV.2011.6126544
26. PF Alcantarilla, J Nuevo, A Bartoli, *Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces* (British Machine Vision Conference (BMVC), Bristol, 2013), pp. 1–11
27. E Rosten, T Drummond, *Machine Learning for High-Speed Corner Detection*. Proceedings of the 9th European conference on Computer Vision, 1, 2006, pp. 105–119
28. J-y Bouguet, *Pyramidal Implementation of the Lucas Kanade Feature Tracker*. Intel Corporation, Microprocessor Research Labs, 2000, pp. 1–9
29. R Hartley, A Zisserman, *Multiple View Geometry in Computer Vision* (second ed) (Cambridge, Cambridge University Press The Edinburgh Building, 2004)
30. MA Fischler, RC Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **26**, 381–395 (1981). doi:10.1145/358669.358692
31. C Zach, Robust bundle adjustment revisited. Comput. Vis. **8693**, 772–787 (2014). doi:10.1007/978-3-319-10602-1_50