Journal of Zhejiang University SCIENCE A
ISSN 1673-565X (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

# Environment map building and localization for robot navigation based on image sequences[*]

Ye-hu SHEN[†], Ji-lin LIU, Xin DU

(*Department of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China*)

[†]E-mail: paulsyh@zju.edu.cn

Received Aug. 20, 2007; revision accepted Oct. 22, 2007

**Abstract:** SLAM is one of the most important components in robot navigation. A SLAM algorithm based on image sequences captured by a single digital camera is proposed in this paper. By this algorithm, SIFT feature points are selected and matched between image pairs sequentially. After three images have been captured, the environment's 3D map and the camera's positions are initialized based on matched feature points and intrinsic parameters of the camera. A robust method is applied to estimate the position and orientation of the camera in the forthcoming images. Finally, a robust adaptive bundle adjustment algorithm is adopted to optimize the environment's 3D map and the camera's positions simultaneously. Results of quantitative and qualitative experiments show that our algorithm can reconstruct the environment and localize the camera accurately and efficiently.

**Key words:** Monocular vision, Digital elevation map (DEM), SIFT, Robust adaptive bundle adjustment, SLAM algorithm
**doi:**10.1631/jzus.A071442 **Document code:** A **CLC number:** TP391

## INTRODUCTION

Robot navigation is one of the most important research fields in the robotics community for decades. In order to do the path planning and detect the obstacles, building the map of the environment around the robot and simultaneously localizing it is an essential part. For this reason, SLAM is one of the most important components in robot navigation system. SLAM was originally proposed by Smith *et al.*(1987), since then, various algorithms have been proposed.

The SLAM algorithms can be approximately divided into two classes according to the different sensors applied to detect the range information. One kind of algorithm uses active sensors such as LADAR (Guivant *et al.*, 2000; Hahnel *et al.*, 2003) or sonar (Choi *et al.*, 2005). The map and localization results are obtained consistently based on a Bayesian estimation scheme. Many of the algorithms use EKF (Weingarten and Siegwart, 2005) or particle filters (Montermerlo *et al.*, 2002). Since the accuracy of the range information acquired by LADAR is quite high, this kind of algorithms usually can provide satisfactory results and is quite mature by now. Unfortunately, LADAR is quite large, heavy, power consuming and expensive. So it is not suitable for the planet rover such as lunar rover and Mars rover as the payload and power supply are very restrictive. On the other hand, although sonar is much cheaper, its angular accuracy is poor and it cannot be used on the moon since there is no air on it. So sonar is mainly used in indoor environments.

With the development of computational power and image quality, vision based SLAM gradually emerges. Camera is much smaller, lighter and consumes less power than LADAR. Furthermore, it is an all-solid-state approach that might be more easily space qualifiable. So it is suitable for the planet rover. Many vision based SLAM use the stereo vision (Se *et al.*, 2002; Garcia and Solanas, 2004; Kwolek, 2007).

[*] Project supported by the National Natural Science Foundation of China (Nos. 60534070 and 60502006) and the Science and Technology Department of Zhejiang Province (No. 2005C14008), China

In recent years, some researchers have proposed algorithms based on the monocular vision, which does not need the system for cameras synchronization or mechanical equipment for keeping the relative positions between two cameras. So monocular vision based SLAM is simpler, cheaper, consumes less power and is suitable as a backup system. Davison (2003) proposed a real-time SLAM algorithm for limited indoor environments. It could only track around 100 feature points, which is not enough for outdoor applications. Chekhlov *et al.*(2007) improved the robustness in feature matching, but they still only demonstrated the indoor results. Nister *et al.*(2004) proposed two separate visual odometry algorithms based on both stereo vision and monocular vision, but they only provided the experimental results with stereo vision system. Campbell *et al.*(2005) realized the visual odometry with a consumer grade camera by concentrating on the localization of the rover. They did not discuss the method to construct the map of the environment. Tomono (2005) proposed an automatic baseline selection method based on the analysis of the rank of the coefficient matrix. The author only provided some qualitative results in short distances and the computational cost analysis was not made.

In order to overcome the defects of the abovementioned algorithms, we propose a simultaneous environment reconstruction and localization algorithm based on monocular vision. Our method is realized in a sequential mode, which is to say, we do not need to capture the whole image sequence beforehand. This is especially suitable for robot navigation since we can process the image sequence while the images are being captured. This is different from most of the SFM (structure from motion) methods which are realized in a batch mode. Firstly, we improve the matching accuracy with the help of the SIFT feature points in image sequences. We use a robust pose and location estimation method when a new image is captured. While doing bundle adjustment, we make use of the residue information from the latest optimization results and propose an adaptive bundle adjustment algorithm. It can provide accurate results and eliminate the outliers with relatively low computational cost. Finally, digital elevation map (DEM), which is more suitable for robot navigation, is generated from 3D point clouds. Overview of the algorithm is shown in Fig.1.

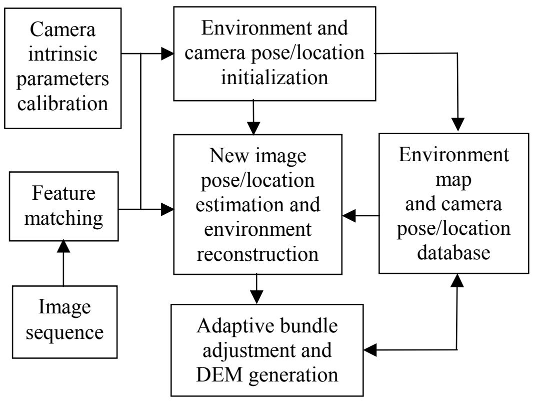

Camera intrinsic parameters calibration
Environment and camera pose/location initialization
Feature matching
New image pose/location estimation and environment reconstruction
Environment map and camera pose/location database
Image sequence
Adaptive bundle adjustment and DEM generation

**Fig.1 Overview of our algorithm**

## ALGORITHM DETAILS

### Camera intrinsic parameters calibration

In ideal condition, the camera satisfies linear model and the intrinsic parameters can be expressed by matrix $\boldsymbol{K}$:

$$\boldsymbol{K}=\begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix},$$

where $f_x$ and $f_y$ are focal lengths of the camera in the unit of the pixel's size in horizontal and vertical directions, respectively. $(u_0, v_0)$ is the optical center of the image plane. $\gamma$ is the skew factor of the pixel. Usually there are some distortions in the image. The relationship between the actual position of image point and the position according to the linear camera model can be described by the following equation:

$$\boldsymbol{x}_\mathrm{d}=distort(\boldsymbol{x}_\mathrm{n})=(1+kc_1r^2+kc_2r^4)\boldsymbol{x}_\mathrm{n}, \quad (1)$$

in which $\boldsymbol{x}_\mathrm{d}=[\boldsymbol{x}_\mathrm{d}(1)\ \boldsymbol{x}_\mathrm{d}(2)]^\mathrm{T}$ is the actual normalized coordinate of image point, $\boldsymbol{x}_\mathrm{n}=[x\ y]^\mathrm{T}$ is the normalized coordinate according to the linear camera model, $r^2=x^2+y^2$. In this paper, we use the method proposed by Zhang (2000) to calculate the intrinsic parameters matrix $\boldsymbol{K}$ and nonlinear distortion coefficients $kc_1$ and $kc_2$.

### Feature matching

In order to improve the quality of image matching and speed up the matching procedure, we first choose some feature points which are easy to match in the image. Then we search for the corresponding points in the next image. We apply the SIFT (Lowe, 2004) for this purpose. SIFT is a successful feature matching method which is robust to image scale and illumination changes. It can also handle affine transformations between images to some extent. With the SIFT feature tracker, we can realize the image matching algorithm in dramatically different viewing directions robustly. Fig.2 shows the matched feature points which are labelled in dots. From the figure we can conclude that most of the matched feature points are correct.

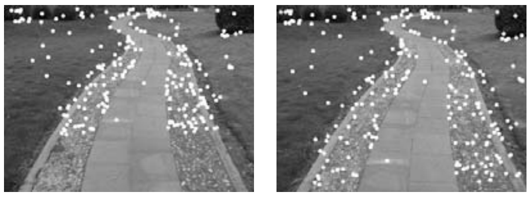


**Fig.2 Example of SIFT matching results between two consecutive images. Features are labelled in dots**

## Environment and camera pose/location initialization

After we have obtained three images, we can initialize the environment map and camera poses/ locations. We set the reference coordinates at the camera coordinates of the first image. We use all the feature points that can be seen in all the three images. Assume $Feat_i$ ($i$=1,2,3) are the sets of homogeneous coordinates of the feature points in the three images. For every $\boldsymbol{x}_c \in Feat_i$ ($i$=1,2,3), we calculate:

$$\boldsymbol{x}_n = distort^{-1}(\boldsymbol{K}^{-1}\boldsymbol{x}_c),$$

where $distort^{-1}()$ is the inverse function of Eq.(1). For the corresponding feature points $\boldsymbol{x}_n$ and $\boldsymbol{x}_n'$ from the first and the third images, we have the epipolar constraint (Ma and Zhang, 2003):

$$(\boldsymbol{x}_n')^T \boldsymbol{E}\boldsymbol{x}_n = 0,$$

where $\boldsymbol{E}$ is the essential matrix. In this paper, the eight-point algorithm (Hartley, 1997) is applied to calculate $\boldsymbol{E}$. We also use the RANSAC (Fischler and Bolles, 1981) to estimate $\boldsymbol{E}$ robustly.

The relationship between the essential matrix and the motion parameters can be expressed by

$$\boldsymbol{E} = [\boldsymbol{t}_u]_\times \boldsymbol{R},$$

where $\boldsymbol{t}_u$ is the translation vector between the first and the third images and $[\boldsymbol{t}_u]_\times$ denotes its corresponding skew symmetric matrix. $\boldsymbol{R}$ is the rotation matrix between the first and third images.

We apply SVD to the essential matrix:

$$SVD(\boldsymbol{E}) \sim \boldsymbol{U}\cdot\text{diag}(1, 1, 0)\cdot\boldsymbol{V}^T \quad (\det\boldsymbol{U}>0, \det\boldsymbol{V}>0).$$

We assume $\boldsymbol{D} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $\boldsymbol{R}_a=\boldsymbol{UDV}^T$, $\boldsymbol{R}_b=\boldsymbol{UD}^T\boldsymbol{V}^T$, $\boldsymbol{t}_u=[u_{13}\ u_{23}\ u_{33}]^T$. There are four possible solutions for the pose and location of the third image: $\boldsymbol{R}_a$ and $\boldsymbol{t}_u$, $\boldsymbol{R}_a$ and $-\boldsymbol{t}_u$, $\boldsymbol{R}_b$ and $\boldsymbol{t}_u$, $\boldsymbol{R}_b$ and $-\boldsymbol{t}_u$. In order to get the true solution, we select a pair of corresponding feature points randomly and triangulate them with pose and location parameters (Ma and Zhang, 2003). If this point is in front of both cameras, the pose and location parameters are exactly what we want. Since we can only determine the structure of the environment up to a scale with monocular vision, we normalize the translation vector $\boldsymbol{t}_u$ so that $||\boldsymbol{t}_u||=1$.

When we have obtained the rotation matrix and translation vector, the projection matrix of the camera can be expressed as: $\boldsymbol{P}=\boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]$. The 3D structure of the corresponding feature points in the first and third images can be triangulated with the projection matrices (Ma and Zhang, 2003). After that, we can obtain the pose and location parameters of the second image (refer to the following subsection). Finally, we optimize the pose and location parameters and 3D structure of the feature points with traditional bundle adjustment (Triggs *et al.*, 2000).

## New image pose/location estimation and environment reconstruction

When a new image has been captured, the pose and location parameters should be estimated. Assume we have processed $N$ images and the new image is $I_{N+1}$, $FS_i$ is the set of homogeneous coordinates of the matched feature points in the $i$th image. $SP$ is the set of 3D structures in the environment map and camera pose/location database. $proj_i(\boldsymbol{X})$ ($\boldsymbol{X}\in SP$) is the projection from the 3D point $\boldsymbol{X}$ to $I_i$. Its theoretical position can be calculated according to the projection matrix $\boldsymbol{P}^i=\boldsymbol{K}[\boldsymbol{R}^i|\boldsymbol{t}^i]$ of $I_i$ and its actual position can be acquired by the feature matching procedure.

First, we calculate $CSP=\{\boldsymbol{X}|\boldsymbol{X}\in SP\wedge proj_{N+1}(\boldsymbol{X})\in FS_{N+1}\}$. After a set $CSP$ has been calculated, we project all the members in $CSP$ to $I_{N+1}$ and calculate the errors with respect to their corresponding detected feature positions in $I_{N+1}$:

$$f_{\mathrm{P}}(\boldsymbol{R}^{N+1},\boldsymbol{t}^{N+1})=\sum_{i=1}^{N_{\mathrm{C}}}\left[r(\boldsymbol{x}_i-proj_{N+1}(\boldsymbol{X}_i))\right]^2,\quad \boldsymbol{X}_i\in CSP,\ \boldsymbol{x}_i\in FS_{N+1},\tag{2}$$

where $N_{\mathrm{C}}=|CSP|$. We can get the pose and location parameters of $I_{N+1}$ by the minimization of Eq.(2). Because SIFT will unavoidably produce false matches, instead of applying the widely used L2 norm $r(\boldsymbol{x})=\|\boldsymbol{x}\|_2$ which is vulnerable to outliers, we choose the robust Huber function:

$$r^2(\boldsymbol{x})=g(\boldsymbol{x})=\begin{cases}|\boldsymbol{x}|^2, & |\boldsymbol{x}|<\delta,\\ 2\delta|\boldsymbol{x}|-\delta^2, & |\boldsymbol{x}|\ge\delta.\end{cases}$$

For small residuals, Huber function acts as traditional squared error, while for large residuals, it performs like L1 norm. In all of our experiments, we choose $\delta$=2. Eq.(2) is optimized by the Levenberg-Marquardt algorithm with pose and location parameters $\boldsymbol{R}^N$ and $\boldsymbol{t}^N$ of the previous image as initialization. Generally speaking, Huber function can produce more robust results at the cost of slightly smaller convergence rate compared with the traditional L2 norm. Fortunately, the number of variables needed to be optimized in Eq.(2) is small (pose and location parameters can be described by six parameters) so that the additional computational cost can be ignored compared with other parts of the algorithm.

Until now we have obtained the pose and location parameters of $I_{N+1}$. As there are some newly matched feature points in every image, the 3D structures of the newly added feature points in $I_{N-1}$ have not been calculated. After we obtain the pose and location parameters of $I_{N+1}$, we triangulate the 3D structures of the feature points which are newly added in $I_{N-1}$ and can still be seen in $I_{N+1}$.

## Robust adaptive bundle adjustment

The abovementioned environment reconstruction and camera pose/location estimation algorithms are done separately. The errors generated in any step will spread further and affect the estimation results. In order to diminish the error diffusion, we apply bundle adjustment after the new image is captured and the scene structure and camera pose/location are estimated. The cost function is:

$$f(C,X)=\frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M}\left[r(\boldsymbol{x}_{ij}-proj_i(\boldsymbol{X}_j))\right]^2,\quad \boldsymbol{X}_j\in X,\tag{3}$$

where $M=|X|$, $C=\{\boldsymbol{R}^i,\ \boldsymbol{t}^i|1\le i\le N\}$ is the set of camera poses and locations for optimization, $X$ is the set of environment structures for optimization, $\boldsymbol{x}_{ij}$ is the detected image coordinate of the $j$th 3D point in the $i$th image, and $r(\boldsymbol{x})$ is the error function. Traditional bundle adjustment (Triggs *et al.*, 2000) takes into account the independence between camera poses/locations and scene structure. The computational cost is greatly reduced compared with the Levenberg-Marquardt algorithm but when the size of the scene structure and camera poses/locations becomes large, the computational cost is still high. Traditional bundle adjustment solves the following equation in each iteration (Hartley and Zisserman, 2003):

$$\begin{bmatrix}\boldsymbol{U}^* & \boldsymbol{W}\\ \boldsymbol{W}^{\mathrm{T}} & \boldsymbol{V}^*\end{bmatrix}\begin{pmatrix}\boldsymbol{\delta}_a\\ \boldsymbol{\delta}_b\end{pmatrix}=\begin{pmatrix}\boldsymbol{\varepsilon}_A\\ \boldsymbol{\varepsilon}_B\end{pmatrix}.$$

The left side is the Hessian matrix $\boldsymbol{H}=\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J}$, whose diagonal is multiplied by $1+\lambda$. $\boldsymbol{\delta}_a$ and $\boldsymbol{\delta}_b$ are increments in each iteration for camera pose/location parameters and scene structure, respectively. $\boldsymbol{\varepsilon}_A$ and $\boldsymbol{\varepsilon}_B$ are residuals. We first solve $\boldsymbol{\delta}_a$ by Eq.(4):

$$[\boldsymbol{U}^*-\boldsymbol{W}(\boldsymbol{V}^*)^{-1}\boldsymbol{W}^{\mathrm{T}}]\boldsymbol{\delta}_a=\boldsymbol{\varepsilon}_A-\boldsymbol{W}(\boldsymbol{V}^*)^{-1}\boldsymbol{\varepsilon}_B.\tag{4}$$

From $\boldsymbol{\delta}_a$ we can get $\boldsymbol{\delta}_b$ by

$$\boldsymbol{\delta}_b=(\boldsymbol{V}^*)^{-1}(\boldsymbol{\varepsilon}_B-\boldsymbol{W}^{\mathrm{T}}\boldsymbol{\delta}_a).$$

Most of the computational cost is spent in calculating the Hessian matrix and solving Eq.(4). For simplicity, we assume there are $n$ feature points in every image on average. The number of cameras for pose and location optimization is $N_O$. We take into account the 2D projections in $N_T$ images. The time complexity for computing the Hessian matrix is $O(nN_T)$. For matrix product $\boldsymbol{W}(\boldsymbol{V}^*)^{-1}\boldsymbol{W}^T$, we first consider the number of not-null blocks of $\boldsymbol{W}(\boldsymbol{V}^*)^{-1}$ which is the same as that of $\boldsymbol{W}$. In the product $\boldsymbol{W}(\boldsymbol{V}^*)^{-1}\boldsymbol{W}^T$, each not-null 6×3 block of $\boldsymbol{W}(\boldsymbol{V}^*)^{-1}$ is used once in the calculation of each block column of $\boldsymbol{W}(\boldsymbol{V}^*)^{-1}\boldsymbol{W}^T$. Thus the computational complexity of the product $\boldsymbol{W}(\boldsymbol{V}^*)^{-1}\boldsymbol{W}^T$ is $O(nN_O^2)$. The computational complexity for solving Eq.(4) is $O(nN_O^3)$. So the total computational complexity is approximately $O(nN_T)+O(nN_O^2)+O(nN_O^3)$. In traditional bundle adjustment, $N_O=N_T=N$. As the number of images $N$ increases, the computational complexity increases rapidly. In order to improve the efficiency of the whole algorithm, we have to tune $N_O$ and $N_T$ so as to find a balance between performance and speed. Furthermore we should make the algorithm immune to outliers. In the above subsection, we have mentioned that the Huber function is more robust than L2 norm and is a candidate for this purpose. But in this subsection, considering the implementation simplicity and kicking outliers explicitly out of the bundle adjustment optimization procedure, we still choose the L2 error function. The procedure of robust adaptive bundle adjustment is as follows:

Step 1: When $N\leq 20$, we choose $N_O=N_T=N$ which is the same as the traditional bundle adjustment. When $N=21$, we choose $N_O=3$ and $N_T=N_O+5$. When $N>21$, we choose $N_O$ and $N_T$ according to the residuals of the last two images:

$$N_O=\begin{cases} N_O-2, & res_{N-1}<res_{N-2}\wedge N_O>3,\\ N_O+2, & res_{N-1}>res_{N-2}\wedge N_O<9,\\ N_O, & \text{otherwise,}\end{cases}$$

$$N_T=N_O+5.$$

Step 2: Do the bundle adjustment so that

$$\{C_{opt},X_{opt}\}=\arg\min_{C,X} f(C,X).$$

Calculate the inlier set:

$$X_{inlier}=\{\boldsymbol{X}_j|\boldsymbol{X}_j\in X_{opt}\wedge r(\boldsymbol{x}_{ij}-proj_i(\boldsymbol{X}_j))<\delta\wedge proj_i(\boldsymbol{X}_j)\in FS_i\wedge N-N_T<i\leq N\}.$$

We choose $\delta=2$ which is fixed in all our experiments. If $X_{inlier}\neq X_{opt}$, do bundle adjustment again:

$$\{C^n_{opt},X^n_{opt}\}=\arg\min_{C_{opt},X_{inlier}} f(C_{opt},X_{inlier}).$$

Otherwise, we simply set

$$C^n_{opt}=C_{opt} \text{ and } X^n_{opt}=X_{opt}.$$

Step 3: Calculate and store the average residuals:

$$res_N=f(C^n_{opt},X^n_{opt}).$$

When $N\leq 20$, the number of images and 3D points in the environment map is not large. The computational cost of the traditional bundle adjustment is not high and it can provide accurate initialization. When the number of images increases, we should constrain $N_O$ and $N_T$. But if the residue of the last image increases, it is not enough to diminish errors with only $N_O$ images, we should add more images for optimization. On the contrary, we can still get reasonable result with fewer images for optimization. We constrain $N_O$ between 3 and 9 for the balance between performance and speed. $N_O$ should not be too small, otherwise the errors will be accumulated rapidly. $N_O$ should also not be too large, which will cause extremely high computational cost. In order to keep the consistency between the optimization results and the previous reconstruction and pose/location estimation results, $N_T$ must satisfy $N_T\geq N_O+2$, so we also choose $N_T=N_O+5$ considering the balance between performance and speed. For 3D points involved in optimization, we choose those which can be detected in at least one of the $N_O$ images.

In order to find the outliers in the environment structure set, we propose a two-step bundle adjustment scheme. In the first step, all the 3D points in the environment structure set are used for bundle adjustment. The residuals of outliers after bundle adjustment must still be large. So we check the reprojection error of every 3D point after optimization. If there is any point whose reprojection error is larger than a predefined threshold, it is deemed as an outlier. If outliers are found in this step, bundle adjustment has to be performed again free of these outliers so that the optimization results will be more accurate. Though bundle adjustment has to be used twice for some images in the sequence which increases the computation time at first glance, fortunately, since the initialization of the second bundle adjustment is better than the first one, fewer iterations are needed in the second bundle adjustment. Furthermore, with our proposed adaptive bundle adjustment procedure, the computation time of bundle adjustment is much shorter than that of the SIFT feature tracker so that the increased computation time is tolerable.

### DEM generation

The above calculated 3D points cloud is not very suitable for robot navigation. They are further processed to create DEM for obstacle detection and path planning. A DEM uses a fixed 2D grid with variable height $z$ to represent the 3D world. In this paper, a uniform grid is used. For every 3D point $\boldsymbol{X}=(x, y, z)^{\mathrm{T}}\in SP$, we first transform it into a world coordinate system. In the world coordinate system, $XY$ plane is parallel to the ground, $Y$ axis is pointing forward and $Z$ axis is pointing upward. Then we assign it to the corresponding grid cell of the DEM according to $x$ and $y$ coordinates of the 3D point. Since the 3D structure set is nearly free of outliers, we simply assign the height of the grid cell in DEM as the point with the largest $z$ coordinate in this cell.

## EXPERIMENTAL RESULTS

### Algorithm effectiveness verification

In order to verify the effectiveness of the main part of our algorithm and show the indoor experimental result, we choose the Dinosaur sequence which can be downloaded from the website (Visual Geometry Group, http://www.robots.ox.ac.uk/~vgg/data.html). This sequence includes 36 images. The resolution is 720×576. The camera is set in front of a turntable. The Dinosaur model is put at the center of the turntable. The camera takes one image after the turntable rotates 10°. The locus of the camera is just on a circle around the Dinosaur model. Figs.3a and 3b show the 1st and 18th frames of the sequence, respectively. Because the 2D feature points in each frame are provided and the camera intrinsic parameters matrix can be calculated from the projection matrix which is also provided in this website by QR factorization, we apply these feature points data directly to the algorithm. Figs.3c and 3d are bird's eye view and side view of the scene reconstruction results, respectively. The white cones stand for the poses and locations of the camera in each image. From the figure we can conclude that the locus of the camera is nearly on the circle, and the reconstruction result is faithful.

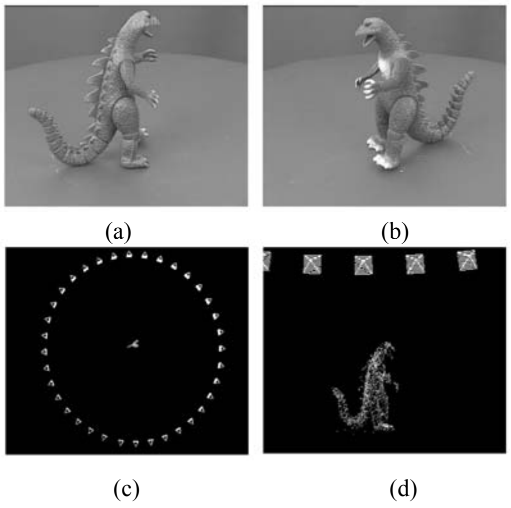


**Fig.3 Dinosaur sequence: (a) The 1st frame; (b) The 18th frame. Reconstruction results: (c) Bird's eye view; (d) Side view**

In order to provide quantitative analysis results, we need the actual locations of the camera. Monocular vision based SLAM can only reconstruct the structure of the environment and estimate the locations of the camera up to a scale. We fit a circle over the estimated locations of the camera by the least square method. The true positions of the camera are spread on the circle evenly. We analyze the errors of positions and poses of the camera separately, and the results are shown in Figs.4a and 4b, respectively. Since the absolute error for the position is meaningless, the relative error (the absolute error divided by the distance traveled by the camera) is analyzed.

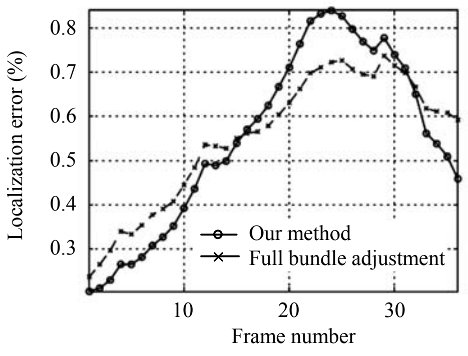
From Fig.4, it can be seen that the performances of the two algorithms are comparable. The maximum localization error of our method is less than 0.9% with the maximum angular error of our method being less than 1.6°. The computation time of our algorithm is 55.3 s, and the traditional method takes 147.4 s. So we find a balance between quality and speed.

### Short distance experimental results

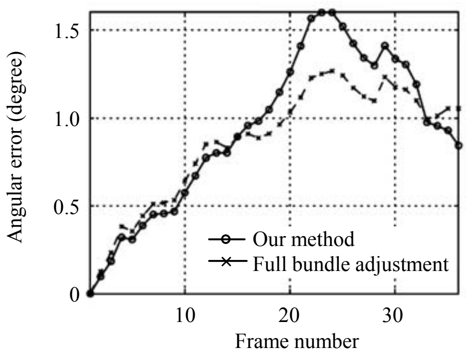
In this subsection we capture images in a planet like terrain test lab with Bumblebee2 digital camera. The FOV of the camera is about 70°, and the resolution is 640×480. The camera is equipped on a prototype planet rover with IMU. The pose of the camera relative to the world coordinate can be measured beforehand. The result is generated by the whole algorithm proposed in this paper automatically without any human intervention. The whole sequence contains 22 images. Figs.5a and 5b show two of them. Figs.5c~5e are scene reconstruction results from different views. Fig.5f is the DEM of the scene. The grid size of the DEM is 0.1 m×0.1 m. From DEM, five rocks can be detected easily. Since the distance traveled is about 2.1 m, IMU can provide reliable estimations in this short distance. We compare localization errors of our method with IMU results. The scale in our method is determined according to the distance traveled between the first two images. It is provided by IMU. The starting point of our result is aligned with that of IMU. Fig.5g is the plot of the localization errors in every image. From the plot we can find that the localization error is less than 0.1 m at the end of the sequence, which is quite small compared with the traveling distance.

## Outdoor experimental results

We use the consumer grade digital camera Canon A630 which is equipped on a tripod. We capture three image sequences. The FOV of the camera is about 53°, and the resolution is 640×480. The pose of the camera relative to the world coordinate can be measured beforehand by inclinometer. All the results are generated by the whole algorithm proposed in this paper automatically without any human intervention.

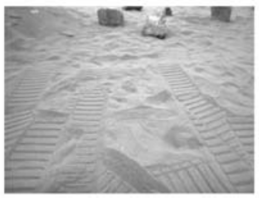


(a)

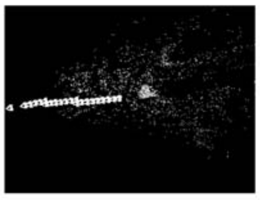


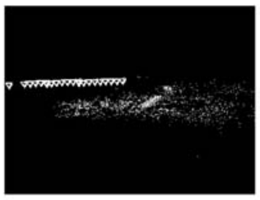
(b)

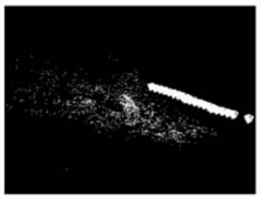
**Fig.4 Localization error analysis. (a) Relative error of the position; (b) Absolute error of the pose**

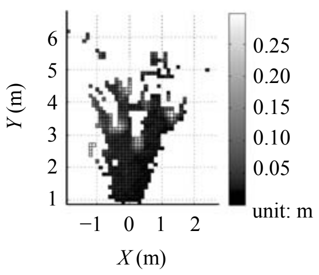

(a)

(b)

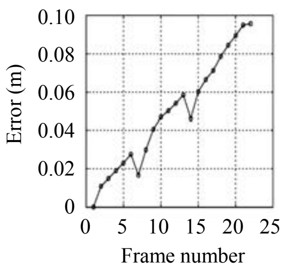

(c)


(d)


(e)



(f)



(g)

**Fig.5 Short distance experimental results. (a) The 2nd image of the image sequence; (b) The 20th image of the image sequence; (c)~(e) 3D scene reconstruction results in three different views; (f) DEM of the scene; (g) Localization errors of every image compared with IMU results**

In all the sequences, the optical axis of the camera is nearly along with the direction of the motion. The camera is a little bit pointing down. The scale in our method is determined according to the whole distances traveled in each sequence.

Fig.6a shows the overview of the whole environment in the first sequence. A person walked along the direction shown by black arrow and went back to the starting point. Figs.6b and 6c are sample images in the sequence. Figs.6d~6f are three different views of the scene reconstruction results. Fig.6g is the DEM of the scene. The grid size of the DEM is 0.3 m×0.3 m. The approximate traveled distance of this sequence is 32 m, which is measured by tapeline. The success of this sequence validates the correctness of our algorithm.

Since we cannot get the actual structure of the environment and the positions/poses of every image, we introduce the loop closure error for quantitative analysis. We assume that there are totally $N$ images. $pos_i$ ($i$=1, 2, …,$N$) is the position of the camera in the $i$th image. The loop closure error is defined as:

$$Err = \frac{\|pos_N - pos_1\|}{\sum_{i=2}^{N}\|pos_i - pos_{i-1}\|} \times 100\%.$$

The smaller the loop closure error is, the better the starting point meets the end point. The quantitative results are shown in Table 1.

Fig.7 shows the result of the second image sequence. This sequence is much longer than the first one and was taken in hilly terrain. This sequence includes lawn and bypath with different elevations. It also contains several sharp turns which are quite challenging. Fig.7a shows the overview of the whole environment. Figs.7b and 7c are sample images in the sequence. Figs.7d~7f are three different views of the scene reconstruction results. From these figures we can clearly find that we are going up and down the hills and turning sharply. Fig.7g is a snapshot from Google Earth$^{TM}$. The bright line with dark dots is our approximate path to capture this sequence. The ruler tool of Google Earth$^{TM}$ is used and it shows that the distance of our trajectory is about 195 m. Fig.7h is the DEM of the scene. The grid size of the DEM is 0.5 m×0.5 m. The DEM also clearly proves the hilly terrain. The success of this sequence shows that our algorithm is suitable for different kinds of terrains with variety of motions. The quantitative results are shown in Table 1.

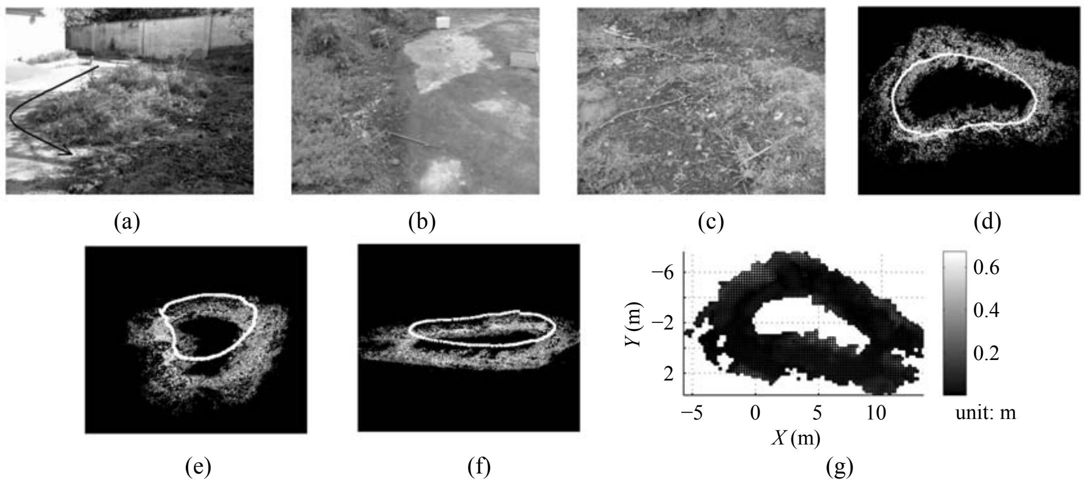


**Fig.6 Experimental results of the 1st sequence. (a) Overview of the environment; (b)~(c) Two sample images in the sequence; (d)~(f) 3D scene reconstruction results in three different views; (g) DEM of the scene**

**Table 1 Quantitative analysis of the three image sequences**

| Sequence index | Loop closure error (%) | Traveled distance (m) | Total image number | Number of 3D points in scene structure |
|---|---|---|---|---|
| 1 | 0.98 | 32 | 196 | 41 252 |
| 2 | 2.62 | 195 | 564 | 63 493 |
| 3 | 3.34 | 360 | 1105 | 133 506 |

Fig.8 shows the result of the third image sequence. This sequence is even longer and was taken in a narrowly paved path. Figs.8a and 8b are sample images in the sequence. Figs.8c~8e are three different views of the scene reconstruction results. Fig.8f is a snapshot from Google Earth$^{TM}$. The bright line with dark dots is our approximate path to capture this sequence. The distance of our trajectory is about 360 m. Fig.8g is the DEM of the scene. The grid size of the DEM is 0.5 m×0.5 m. The success of this sequence shows the robustness of our algorithm. The quantitative results are shown in Table 1.

From Table 1 we can find that the loop closure error is quite small and comparable to the method proposed by Nister *et al.*(2004) with stereo vision system.

**Analysis on computational cost**

The main framework of our algorithm is accomplished in Matlab, while the robust adaptive bundle adjustment is written in C for high efficiency. The platform for testing the algorithm is P4 2.8 GHz with 1 GB RAM running Windows XP. We first captured image sequences by digital camera, and then downloaded data to the computer and tested the algorithm off-line. The image sequence is read sequentially to simulate the image capturing procedure. SIFT feature tracking takes more than 75% of the time. For example, in Sequence 2, the total computation time is 6492 s. SIFT takes about 4913 s. We only analyzed the time used for the second sequence without considering the time for SIFT feature matching. The result is shown in Fig.9.

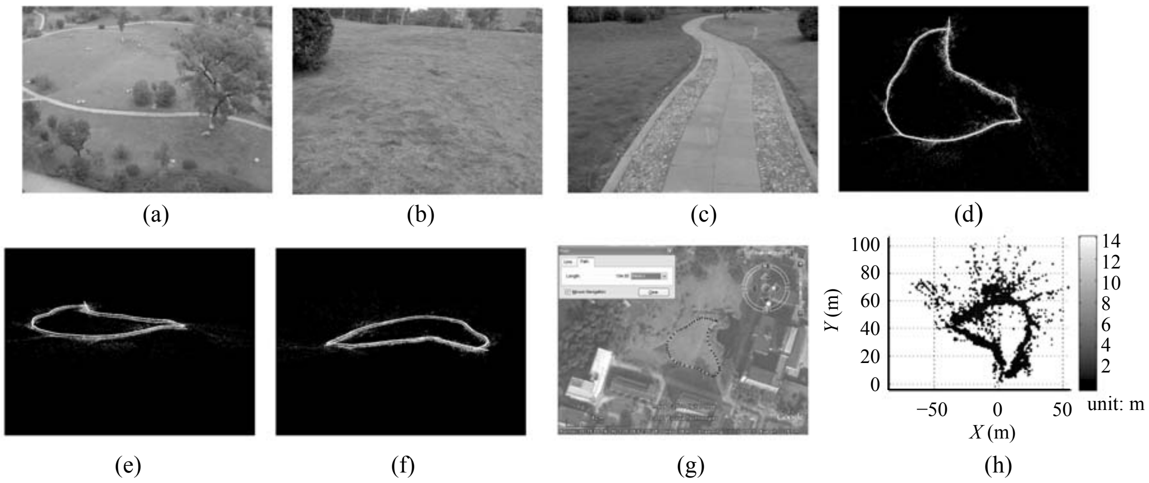


**Fig.7 Experimental results of the 2nd sequence. (a) Overview of the environment; (b)~(c) Two sample images in the sequence; (d)~(f) 3D scene reconstruction results in three different views; (g) Snapshot from Google Earth$^{TM}$. It shows that our path is about 194.85 m; (h) DEM of the scene**

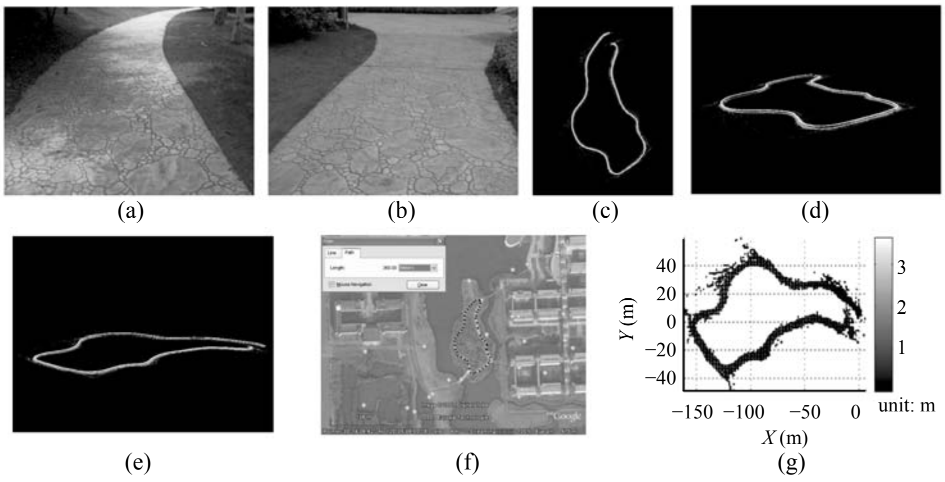


**Fig.8 Experimental results of the 3rd sequence. (a)~(b) Two sample images in the sequence; (c)~(e) 3D scene reconstruction results in three different views; (f) Snapshot from Google Earth$^{TM}$. It shows that our path is about 360 m; (g) DEM of the scene**

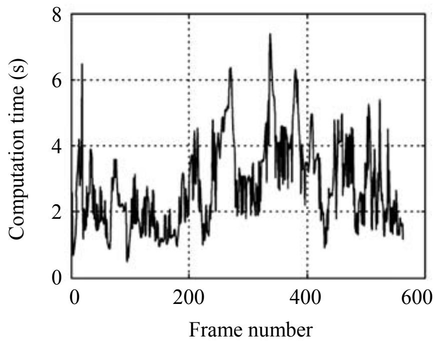


**Fig.9 Analysis of the computation time for every frame in the 2nd sequence without considering SIFT feature matching**

From Fig.9 we can find that, when the number of images is below 20, the computation time grows rapidly. This proves that the computational cost is very high for traditional bundle adjustment if the sequence is long. On the other hand, when the number of images is greater than 20, the computation time of our proposed algorithm does not grow rapidly. At first they are around 2 s per image. Gradually, since 3D points become more and the outliers filtering stage takes longer time, there are some frames near the 350th frame which take extremely long time. This is caused by a huge amount of 3D points included in the optimization procedure. Generally speaking, the computation time of our algorithm does not grow so rapidly with the increasing of the image frames. The computation is greatly accelerated. We find the balance between quality and speed.

## CONCLUSION

This paper proposes a simultaneous environment reconstruction and localization algorithm based on monocular vision and image sequence. We propose a robust camera pose and location estimation method. We use robust adaptive bundle adjustment so that the computation is accelerated and the reconstruction and localization quality is still good. Indoor and outdoor experimental results show that our algorithm can provide accurate 3D environment map and positions/poses of the camera. It is suitable for low speed planet rover which cannot use LADAR such as lunar rover and Mars rover. In the future, we prepare to use the SURF feature tracker (Bay *et al.*, 2006) instead of SIFT for higher speed.

## References

Bay, H., Tuytelaars, T., Gool, L.V., 2006. SURF: Speeded Up Robust Features. European Conf. on Computer Vision, p.404-417.

Campbell, J., Sukthankar, R., Nourbakhsh, I., Pahwa, A., 2005. A Robust Visual Odometry and Precipice Detection System Using Consumer-Grade Monocular Vision. Proc. Int. Conf. on Robotics and Automation, p.3421-3427.

Chekhlov, D., Pupilli, M., Mayol, W., Calway, A., 2007. Robust Real-Time Visual SLAM Using Scale Prediction and Exemplar Based Feature Description. Proc. Int. Conf. on Computer Vision and Pattern Recognition, p.1-7.

Choi, J., Ahn, S., Chung, W., 2005. Robust Sonar Feature Detection for the SLAM of Mobile Robot. Proc. Int. Conf. on Intelligent Robots and Systems, p.3415-3420.

Davison, A., 2003. Real-Time Simultaneous Localization and Mapping with a Single Camera. Proc. 9th IEEE Int. Conf. on Computer Vision, p.1403-1410. [doi:10.1109/ICCV.2003.1238654]

Fischler, M., Bolles, R., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *ACM Trans. on Commun.*, **24**(6):381-395. [doi:10.1145/358669.358692]

Garcia, M., Solanas, A., 2004. 3D Simultaneous Localization and Modeling from Stereo Vision. Proc. Int. Conf. on Robotics and Automation, p.847-853.

Guivant, J., Nebot, E., Baiker, S., 2000. Localization and map building using laser range sensors in outdoor applications. *J. Rob. Syst.*, **17**(10):565-583. [doi:10.1002/1097-4563(200010)17:10<565::AID-ROB4>3.0.CO;2-6]

Hahnel, D., Burgard, W., Fox, D., Thrun, S., 2003. An Efficient FastSLAM Algorithm for Generating Maps of Large-Scale Cyclic Environments from Raw Laser Range Measurements. Proc. Int. Conf. on Intelligent Robots and Systems, p.206-211.

Hartley, R., 1997. In defense of the eight-point algorithm. *IEEE Trans. on Pattern Anal. Machine Intell.*, **19**(6):580-593. [doi:10.1109/34.601246]

Hartley, R., Zisserman, A., 2003. Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge.

Kwolek, B., 2007. Visual Odometry Based on Gabor Filters and Sparse Bundle Adjustment. Proc. Int. Conf. on Robotics and Automation, p.3573-3578.

Lowe, D., 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Computer Vision*, **60**(2):91-110. [doi:10.1023/B:VISI.0000029664.99615.94]

Ma, S., Zhang, Z., 2003. Computer Vision—Computational Theory and Basics of the Algorithms. Science Press, Beijing, China (in Chinese).

Montermerlo, M., Thrun, S., Koller, D., Wegbreit, B., 2002. Fastslam: A Factored Solution to the Simultaneous Localization and Mapping Problem. Proc. National Conf. on Artificial Intelligence, p.593-598.

Nister, D., Naroditsky, O., Bergen, J., 2004. Visual Odometry. Proc. Int. Conf. on Computer Vision and Pattern Recognition, p.652-659.

Se, S., Lowe, D., Little, J., 2002. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *Int. J. Rob. Res.*, **21**(8):735-758. [doi:10.1177/027836402761412467]

Smith, R., Self, M., Cheeseman, P., 1987. A Stochastic Map for Uncertain Spatial Relationships. Proc. Int. Symp. on Robotics Research, p.468-474.

Tomono, M., 2005. 3-D Localization and Mapping Using a Single Camera Based on Structure-from-Motion with Automatic Baseline Selection. Proc. Int. Conf. on Robotics and Automation, p.3342-3347.

Triggs, B., McLauchlan, P., Hartley, R., Fitzgibbon, A., 2000. Bundle Adjustment—A Modern Synthesis. *In*: Vision Algorithms: Theory and Practice. Springer-Verlag, Berlin, p.298-375. [doi:10.1007/3-540-44480-7_21]

Weingarten, J., Siegwart, R., 2005. EKF-based 3D SLAM for Structured Environment Reconstruction. Proc. Int. Conf. on Intelligent Robots and Systems, p.3834-3839.

Zhang, Z., 2000. A flexible new technique for camera calibration. *IEEE Trans. on Pattern Anal. Machine Intell.*, **22**(11):1330-1334. [doi:10.1109/34.888718]