Towards a Stereo Approach to PTAM

Taihú Pire and Julio Jacobo Berlles

Departamento de Computación, Facultad de Ciencias Exactas y Naturales,
Universidad de Buenos Aires, Argentina.
tpire@dc.uba.ar, jacobo@dc.uba.ar

Abstract. This work presents the partial results obtained during the development of a system for local Simultaneous Localization and Mapping (SLAM) using a stereo camera. We follow the parallel tracking and mapping spirit that has become a standard in the community, and divide the computation into two threads (PTAM). Our experiments with this system show its real-time performance and the accuracy of the estimated metric maps. The main contribution of this work is the introduction of the stereo constraints in a local non-linear Bundle Adjustment and tracking operations and the release of the code under the name of *STAM* –standing for Stereo Tracking and Mapping– for its use as a basic local feature-based stereo SLAM.

Student Level: PhD, Expected date of conclusion: December 2016

1 Introduction

SLAM, standing for Simultaneous Localization and Mapping, is a key area for the development of truly autonomous robotic systems. Basically, the goal of SLAM is the online estimation of the model of a scene and the robot trajectory from the readings of a set of sensors and in a fully automated manner. Such scene models, and the relative pose of the robot on them, are essential for the robot to safely interact with the environment.

Since the birth of the problem in the 80's ([1] is usually referenced as one of the first SLAM papers), SLAM has become a classic problem and one of the most studied in robotics. The literature on SLAM is huge and a complete review of the literature is out of the scope of this work. But, as a brief and uncomplete summary, some of the research lines have studied the use of several sensors –laser, sonar, visual or RGB-D sensors–; efficient approaches for the estimation, large mapping, semantic mapping, life-long mapping [2] [3] [4] [5].

In this work we focus on SLAM using visual sensors. [6] demonstrated that the optimal –optimal meaning the one getting more information per processing unit– approach to visual SLAM with the current hardware platforms was the Parallel Tracking and Mapping (PTAM) algorithm of [7]. There, the SLAM problem is partitioned into two threads with small communications between them. A tracking thread that estimates *for every frame* the camera pose given an estimation of the map; and a mapping thread that estimates *for a reduced set* *of keyframes* the keyframe poses and the map. PTAM has become the standard technique for small-scale camera tracking and scene mapping where the most recent research builds on.

The contribution of this work is the development of an algorithm for small-scale camera tracking and scene mapping for stereo cameras. The main advantages of this approach with respect to the monocular one are: create a real-scale representation of the environment; the use of the left and right measurements to improve the refinement performed by Bundle Adjustment during mapping phase; initialize 3D points even during rotational movements only. We will call our approach STAM, standing for Stereo Tracking and Mapping.

The rest of the work is structured as follows. Section 2 introduces the preliminary concepts and notation. Section 3 describes the estimation of the pose of the stereo camera. Section 4 describes the estimation of the map using stereo and multiview constraints. Section 5 refers to the initialization of 3D points using the stereo constraint and also the support of the rest of the views. Finally, 6 shows the experimental results and section 7 presents the conclusions and lines for future work.

2 Previous Concepts

In this section we introduce the notation we will use along the report.

Camera Pose $\boldsymbol{E}^{\mathrm{CW}} = [\boldsymbol{R} \mid \boldsymbol{t}]$ where $\boldsymbol{R}$ is a rotation matrix and $\boldsymbol{t}$ is a translation vector. $\boldsymbol{E}$ is a transformation (belonging to the Lie Group, $SE(3)$, the group of rigid-body motions in 3D) which transform a point in world coordinates frame $\boldsymbol{x}^{\mathrm{W}} = \left[x^{\mathrm{W}}\ y^{\mathrm{W}}\ z^{\mathrm{W}}\ 1\right]^{\top}$ to camera coordinates frame $\boldsymbol{x}^{\mathrm{C}} = \left[x^{\mathrm{C}}\ y^{\mathrm{C}}\ z^{\mathrm{C}}\ 1\right]^{\top}$, that is:

$$\boldsymbol{x}^{\mathrm{C}} = \boldsymbol{E}^{\mathrm{CW}} \boldsymbol{x}^{\mathrm{W}} \tag{1}$$

Motion Matrix noted with $\boldsymbol{M}$, is a 4×4 matrix (belongs to $SE(3)$) which represents the changes in camera pose by left-multiplication, $\boldsymbol{E}^{\mathrm{CW}} = \boldsymbol{M}^{\mathrm{C}} \boldsymbol{E}^{\mathrm{CW}}_{\mathrm{prev}}$. In Lie Groups, the motion matrix $\boldsymbol{M}$ could be represented by a six-vector $\boldsymbol{\mu} = (t_x, t_y, t_z, \theta^{\mathrm{roll}}, \theta^{\mathrm{pitch}}, \theta^{\mathrm{yaw}})$, where the first three elements correspond to translation and the last three to rotation angles respectively. The motion vector $\boldsymbol{\mu}$ and motion matrix $\boldsymbol{M}$ are related by:

$$\boldsymbol{M} = \exp(\boldsymbol{\mu}) = e^{\sum_{j=1}^{6} \mu_j \boldsymbol{G}_j} \tag{2}$$

here $\boldsymbol{G}_j$ with $j = 1 \cdots 6$ are the group generator matrices. They result from the partial derivatives of motion matrices with respect to the motion parameters evaluated in $\boldsymbol{\mu} = 0$, that is, $\frac{\partial \boldsymbol{M}}{\partial \mu_j} = \boldsymbol{G}_j$. For further information on Lie Groups the reader is referred to [8].

Measurement noted with letter $\boldsymbol{z}$, is the true 2D position that matches with the projected 3D point on the camera's image plane.

Map Point noted with p, is an ordered pair $\left(\boldsymbol{x}^{\mathrm{W}}, \boldsymbol{d}\right)$ which contains the 3D point $\boldsymbol{x}^{\mathrm{W}}$ and its associated descriptor $\boldsymbol{d}$.

Stereo KeyFrame is a stereo pair of images with the associated stereo camera pose.

Map noted with m, is defined as the set of Map Points and the set of stereo KeyFrames.

A point in camera reference frame, $\boldsymbol{x}^{\mathrm{C}}$, projects into the image as:

$$\begin{bmatrix} u \\ v \end{bmatrix} = CamProj\left(\left[x^{\mathrm{C}}\ y^{\mathrm{C}}\ z^{\mathrm{C}}\ 1\right]^{\top}\right) \tag{3}$$

To perform the projection, a *pin-hole* camera model is used:

$$CamProj\left(\left[x^{\mathrm{C}}\ y^{\mathrm{C}}\ z^{\mathrm{C}}\ 1\right]^{\top}\right) \overset{\text{def}}{=} \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \left[\boldsymbol{I}_3 \mid 0\right] \begin{bmatrix} x^{\mathrm{C}} \\ y^{\mathrm{C}} \\ z^{\mathrm{C}} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f_u x^{\mathrm{C}}}{z^{\mathrm{C}}} + u_0 \\ \frac{f_v y^{\mathrm{C}}}{z^{\mathrm{C}}} + v_0 \end{bmatrix} \tag{4}$$

3 Camera Pose Tracking

In this section, the approaches used to localize the stereo camera given a map are presented.

3.1 Features Extraction

In the literature, several image feature detection algorithms (such as SIFT, SURF, FAST, STAR, ORB) and descriptor extractor algorithms (such as SIFT, SURF, BRIEF, ORB) are presented. In this work, different Detector-Descriptor combinations were tried, reaching the conclusion that the ORB feature detector and its descriptor gives good results providing a good trade-off between computing speed and robustness, given that ORB is invariant to rotation [9]. Furthermore, to achieve scale invariance, features are extracted and described over an 8-level scale pyramid.

3.2 Features Tracking

The process of map tracking consists in determining the projections on the camera frames of the 3D static points (the map), based on a sequence of camera poses. The problem here is to find the correspondences between the 2D projections and the 3D points at each camera pose. In order to accomplish this task, at each step of the sequence, the following steps are performed:

1. Image frames from the stereo-camera pair are undistorted and rectified
2. The camera pose is predicted using a motion model
3. Left camera's field-of-view is computed based on its predicted pose

4. 3D-points inside the view frustum are projected
5. A search inside the neighborhood of each projection is performed in order to find the correct match (using descriptors)
6. The matches resulting from previous step are used to update the camera pose

First of all, the upcoming stereo frames are undistorted and rectified using intrinsic and extrinsic parameters. Then, a *decaying velocity model* is used to predict pose of the camera [10]. Then, the view frustum is computed using the predicted pose. All the 3D points which fall into the frustum are projected to the left camera frame. It is important to note that only the left camera is tracked, this is done because tracking with both cameras would render the system too slow. In the next step, the detection of ORB points inside the projections' neighborhoods are performed. The size of each neighborhood is predefined as the 0.4% of the image size, which means that for an image of 640×240 pixels, the neighborhood will have 25×25 pixels. In order to find the match between the projections and the ORB key-points, their descriptors are compared. The last step, consist in a refinement of the camera pose by minimization of the re-projection error. As [7] proposes, ten iterations of the Gauss-Newton method are enough to perform this task. To reduce the effect of outliers, an *M-Estimator* cost function is used.

3.3 Pose Update

To estimate the current camera pose $\boldsymbol{E}^{\mathrm{CW}}$, it is necessary to find the motion matrix $\boldsymbol{M}^{\mathrm{C}}$, as proposed in [11], which satisfies

$$\boldsymbol{E}^{\mathrm{CW}} = \boldsymbol{M}^{\mathrm{C}} \boldsymbol{E}^{\mathrm{CW}}_{\mathrm{prev}} \tag{5}$$

where $\boldsymbol{M}^{\mathrm{C}}$ is the small motion in the camera frame and $\boldsymbol{E}^{\mathrm{CW}}$ is the previous pose. To find $\boldsymbol{M}^{\mathrm{C}}$, (10) is used to express its dependence on the motion vector $\boldsymbol{\mu} = (t_x, t_y, t_z, \theta_{roll}, \theta_{pitch}, \theta_{yaw})$. Then, the Jacobian matrix $\boldsymbol{J}$ which describes the effect of each element of $\boldsymbol{\mu}$ on each element of the re-projection error $\boldsymbol{\Delta z}_i(\boldsymbol{\mu})$, is calculated

$$\begin{aligned} J_{ij} = \frac{\partial \boldsymbol{\Delta z}_i(\boldsymbol{\mu})}{\partial \mu_j} &= \frac{\partial \left(\begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix}_i - CamProj\left(\exp(\boldsymbol{\mu}) \boldsymbol{E}^{\mathrm{CW}}_{\mathrm{prev}} \boldsymbol{x}^{\mathrm{W}}_i\right) \right)}{\partial \mu_j} &(6) \\ &= -\frac{\partial CamProj\left(\boldsymbol{x}^{\mathrm{C}}_i\right)}{\partial \boldsymbol{x}^{\mathrm{C}}_i} \frac{\partial \boldsymbol{x}^{\mathrm{C}}_i}{\partial \mu_j} &(7) \end{aligned}$$

where,

$$\frac{\partial CamProj\left(\boldsymbol{x}^{\mathrm{C}}_i\right)}{\partial \boldsymbol{x}^{\mathrm{C}}_i} = \begin{bmatrix} \frac{f_u}{z^{\mathrm{C}}} & 0 & -\frac{f_u x^{\mathrm{C}}}{z^{\mathrm{C}2}} \\ 0 & \frac{f_v}{z^{\mathrm{C}}} & -\frac{f_v y^{\mathrm{C}}}{z^{\mathrm{C}2}} \end{bmatrix} \tag{8}$$

and

$$\frac{\partial \boldsymbol{x}^{\mathrm{C}}_i}{\partial \mu_j} = \boldsymbol{G}_j \boldsymbol{E}^{\mathrm{CW}}_{\mathrm{prev}} \boldsymbol{x}^{\mathrm{W}}_i \tag{9}$$

The motion vector $\boldsymbol{\mu}$ is found by solving the equation:

$$\boldsymbol{J}\boldsymbol{\mu} = \boldsymbol{\Delta z}(\boldsymbol{\mu}_{\text{prev}}) \tag{10}$$

In order to do this, given a set $S = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$ of matched measurements, the new value for $\boldsymbol{\mu}$ is the obtained by minimizing an objective function as follows

$$\boldsymbol{\mu}' = \underset{\boldsymbol{\mu}}{\operatorname{argmin}} \sum_{i \in S} \rho_{\sigma_T}\left(\frac{\boldsymbol{J}_i\boldsymbol{\mu} - \boldsymbol{\Delta z}_i(\boldsymbol{\mu}_{\text{prev}})}{\sigma_i}\right), \tag{11}$$

where $\rho_{\sigma_T}(.)$ is Tukey's robust function and σ_T is the robust standard deviation estimate [12]. For each $\boldsymbol{z}_j$, the variance of the measurement noise is $\sigma_j^2 = 2^{2l}$, where l is the pyramid level of the detected feature. The minimization in (11) is performed using the *Gauss-Newton* method, as proposed in [7]. Tukey's robust function is used to reduce the effect of outliers.

4 Stereo Mapping

In this section we present the Stereo Mapping algorithm that uses the multiple view and stereo constraints to produce a sparse map of features and stereo keyframes from the information in the stereo images. Our approach follows the monocular system presented in [7] and extends it with the stereo constraints.

Bundle Adjustment is a particular case of least squares estimation, and consists in the refinement of a set of camera poses (keyframes) and 3D points (the map) by reducing the re-projection error of every point in every image. The problem can be stated as follows: given a initial set of camera poses $\{\boldsymbol{E}_1, \ldots, \boldsymbol{E}_N\}$, an initial set of 3D points $\boldsymbol{x} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$ and a family of measurement sets $\{S_1, \ldots, S_N\}$ where each set S_j contains the measurement $\boldsymbol{z}_{ij}$ of the point i-th in the j-th keyframe, the simultaneous estimation of the multiple cameras and the point cloud is achieved solving

$$\boldsymbol{J}\begin{bmatrix}\boldsymbol{\mu}\\ \boldsymbol{x}\end{bmatrix} = \boldsymbol{\Delta z}\left(\boldsymbol{\mu}_{\text{prev}}, \boldsymbol{x}_{\text{prev}}\right), \tag{12}$$

where $\boldsymbol{\Delta z}\left(\boldsymbol{\mu}, \boldsymbol{x}\right) = \boldsymbol{z} - \mathit{CamProj}\left(\exp(\boldsymbol{\mu})\boldsymbol{E}^{\text{CW}}\boldsymbol{x}^{\text{W}}\right)$ is the reformulated re-projection error where the dependence of the 3D point is included. In a way analogous to the minimization used in Sec. (3.3), we must minimize the double summation in (13)

$$\begin{aligned}&\{\{\boldsymbol{\mu}'_2, \ldots, \boldsymbol{\mu}'_N\}, \{\boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_M\}\} = \\ &\underset{\{\{\boldsymbol{\mu}\},\{\boldsymbol{x}\}\}}{\operatorname{argmin}} \sum_{j=1}^{N}\sum_{i \in S_j} \rho_{\sigma_T}\left(\frac{\boldsymbol{J}_{ji}\begin{bmatrix}\boldsymbol{\mu}_j\\ \boldsymbol{x}_i\end{bmatrix} - \boldsymbol{\Delta z}_i\left(\boldsymbol{\mu}_{prev,j}, \boldsymbol{x}_{prev,i}\right)}{\sigma_{ji}}\right),\end{aligned} \tag{13}$$

observe that $\boldsymbol{\mu}_1$ is fixed during Bundle Adjustment refinement. This is because the first keyframe is given zero uncertainty, as it defines the world reference frame. Given that the vector of parameters is divided in two groups (cameras and points) the jacobian could be decompose as $J = \left[\frac{\partial \boldsymbol{\Delta z}(\boldsymbol{\mu},\boldsymbol{x})}{\partial \boldsymbol{\mu}} \middle| \frac{\partial \boldsymbol{\Delta z}(\boldsymbol{\mu},\boldsymbol{x})}{\partial \boldsymbol{x}} \right]$. So the jacobian is computed as explained bellow:

The Jacobian's coefficients corresponding to camera pose parameters have the form given by (6). The Jacobian's coefficients corresponding to points parameters have the form:

$$\boldsymbol{J}_{ji} = \frac{\partial \boldsymbol{\Delta z}(\boldsymbol{\mu}_j, \boldsymbol{x}_i^{\mathrm{W}})}{\partial \boldsymbol{x}_i^{\mathrm{W}}} = \frac{\partial \left(\begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix}_i - CamProj \left(\exp(\boldsymbol{\mu}_j) \boldsymbol{E}_{\mathrm{prev}}^{\mathrm{C}_j\mathrm{W}} \boldsymbol{x}_i^{\mathrm{W}} \right) \right)}{\partial \boldsymbol{x}_i^{\mathrm{W}}} \tag{14}$$

$$= - \frac{\partial CamProj \left(\boldsymbol{x}_i^{\mathrm{C}_j} \right)}{\partial \boldsymbol{x}_i^{\mathrm{C}_j}} \frac{\partial \boldsymbol{x}_i^{\mathrm{C}_j}}{\partial \boldsymbol{x}_i^{\mathrm{W}}} \tag{15}$$

The first partial derivative is given by (8), the second partial derivative results from

$$\frac{\partial \boldsymbol{x}_i^{\mathrm{C}_j}}{\partial \boldsymbol{x}_i^{\mathrm{W}}} = \frac{\partial \left(\boldsymbol{M}^{\mathrm{C}_j} \boldsymbol{E}_{\mathrm{prev}}^{\mathrm{C}_j\mathrm{W}} \boldsymbol{x}_i^{\mathrm{W}} \right)}{\partial \boldsymbol{x}_i^{\mathrm{W}}} = \boldsymbol{R}. \tag{16}$$

To this point, we have addressed the Bundle Adjustment technique for the one-camera case. However it is possible to add a second camera that will become the right component of the stereo pair. The relationship between the poses of the left and the right cameras is constant, so we can obtain the pose of the right camera from the left camera using (17)

$$\boldsymbol{E}^{\mathrm{RW}} = \boldsymbol{E}^{\mathrm{RL}} \boldsymbol{M}^{L} \boldsymbol{E}_{\mathrm{prev}}^{\mathrm{LW}}. \tag{17}$$

Now, we can use the right camera measurements to add stereo constraints to Bundle Adjustment. These constraints are given by

$$\boldsymbol{z}^{\mathrm{R}} = CamProj^{\mathrm{R}} \left(\boldsymbol{E}^{\mathrm{RL}} \boldsymbol{M}^{L} \boldsymbol{E}_{\mathrm{prev}}^{\mathrm{LW}} \boldsymbol{x}^{\mathrm{W}} \right). \tag{18}$$

Summing up, a 3D-2D point constraint is modelled with (4) in the left camera but with the equation (18) in the right camera. The Jacobian rows related to right measurements have the form

$$\boldsymbol{J}_{ji}^{\mathrm{R}} = \frac{\partial \boldsymbol{\Delta z}^{\mathrm{R}}(\boldsymbol{\mu}_j, \boldsymbol{x}_i^{\mathrm{W}})}{\partial \boldsymbol{x}_i^{\mathrm{W}}} = \frac{\partial \left(\begin{bmatrix} \hat{u}^{\mathrm{R}} \\ \hat{v}^{\mathrm{R}} \end{bmatrix}_i - CamProj^{\mathrm{R}} \left(\boldsymbol{E}^{\mathrm{RL}} \exp(\boldsymbol{\mu}_j) \boldsymbol{E}_{\mathrm{prev}}^{\mathrm{L}_j\mathrm{W}} \boldsymbol{x}_i^{\mathrm{W}} \right) \right)}{\partial \boldsymbol{x}_i^{\mathrm{W}}} \tag{19}$$

$$= \begin{bmatrix} \frac{f_u}{z^{\mathrm{R}}} & 0 & -\frac{f_u x^{\mathrm{R}}}{{z^{\mathrm{R}}}^2} \\ 0 & \frac{f_v}{z^{\mathrm{R}}} & -\frac{f_v y^{\mathrm{R}}}{{z^{\mathrm{R}}}^2} \end{bmatrix} \boldsymbol{R}^{\mathrm{RL}} \boldsymbol{R}. \tag{20}$$

Observe if the stereo camera is rectified, then the transformation between cameras is a pure translation in axis x (baseline) and the intrinsic parameters are the same, therefore $y^{\mathrm{L}} = y^{\mathrm{R}}$ and $z^{\mathrm{L}} = z^{\mathrm{R}}$ and (19) can be rewritten as follow

$$\boldsymbol{J}^{\mathrm{R}}_{ji} = \begin{bmatrix} \frac{f_u}{z^{\mathrm{L}}} & 0 & -\frac{f_u x^{\mathrm{R}}}{z^{\mathrm{L}^2}} \\ 0 & \frac{f_v}{z^{\mathrm{L}}} & -\frac{f_v y^{\mathrm{L}}}{z^{\mathrm{L}^2}} \end{bmatrix} \boldsymbol{R}. \tag{21}$$

Finally the Jacobian can be expressed as

$$\boldsymbol{J}_{ji} = \begin{bmatrix} \frac{f_u}{z^{\mathrm{L}}} & 0 & -\frac{f_u x^{\mathrm{L}}}{z^{\mathrm{L}^2}} \\ 0 & \frac{f_v}{z^{\mathrm{L}}} & -\frac{f_v y^{\mathrm{L}}}{z^{\mathrm{L}^2}} \\ \frac{f_u}{z^{\mathrm{L}}} & 0 & -\frac{f_u x^{\mathrm{R}}}{z^{\mathrm{L}^2}} \end{bmatrix} \boldsymbol{R}. \tag{22}$$

The Fig. 1 shows the constraints used by Bundle Adjustment.

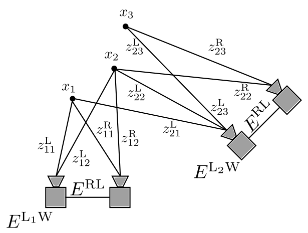

Fig. 1: The constraints (measurements z_{ji} and extrinsic parameters $\boldsymbol{E}^{\mathrm{RL}}$) used by Bundle Adjustment to refine the points x_i and left camera poses $\boldsymbol{E}^{\mathrm{L_jW}}$.

5 Map Points Initialization

In this section the method used for point initialization is presented. There are two ways to create 3D points. One, uses the stereo pair, the other one uses frames from different keyframes, in both case the same triangulation method is performed. To compute 3D points, the following steps are performed:

1. Make a Mask to detect features which do not belong to current Map.
2. Compute Fundamental Matrix, $\boldsymbol{F}$, given known Projection Matrices
3. Keep correct matches using First order geometric error (Sampson Distance).
4. Create 3D points using Linear triangulation method

Given a pair of frames and the poses where they have been taken, the points inside Frustums are projected to images planes and binary masks are made to avoid the creation of repeated 3D points. Using the mask, ORB features are detected and their descriptor are computed. Then, a brute-force descriptor matcher is used to match features from the first frame to the second one. For each descriptor in the first frame, the matcher finds the closest descriptor in the second frame by trying each one. Then, first order approximation to geometric error (Sampson Distance) for each $x_i \leftrightarrow x'_i$ correspondences is computed.

$$\frac{(\boldsymbol{x}'_i \boldsymbol{F} \boldsymbol{x}_i)^2}{(\boldsymbol{F}\boldsymbol{x}_i)_1^2 + (\boldsymbol{F}\boldsymbol{x}_i)_2^2 + (\boldsymbol{F}^\top \boldsymbol{x}'_i)_1^2 + (\boldsymbol{F}^\top \boldsymbol{x}'_i)_2^2} < \text{threshold} \tag{23}$$

where $(\boldsymbol{F}\boldsymbol{x}_i)_j^2$ represents the square of the j-th entry of the vector $\boldsymbol{F}\boldsymbol{x}_i$ and $d(.,.)$ is the euclidean distance. In this way, the points which satisfy (23) are inlier matches and are triangulated. It is important to note that using an stereo camera, it is possible to initialize 3D points by triangulation even during rotational movements only, this is a great advantage with respect to mono camera systems as [7].

6 Preliminary Experiments

The accuracy of the proposed method was tested by using a checker-board with known dimensions. The reconstructed 3D coordinates were then compared to the expected values. A Minoru stereo camera was used to record the sequences for our experiments. The camera has a 60 mm baseline and was configured at a resolution of 640×480 pixels at 15 fps. A laptop computer with an Intel Core $i5 \sim 2.67$ GHz processor with 4 GB of RAM, was used for running the implementation. With this hardware, a processing speed of ~ 20 fps was achieved. Table. 1 and Table. 2 show the performance of STAM tracking and mapping phases respectively. Figure 2(a) shows the initialization of the new map points from the stereo camera. Figure 2(b) illustrates the tracking of the map after its expansion, the yellow marks correspond to predicted pose projections and the red marks correspond to the measurements found in the current frame. Figure 3 exposes the point cloud; new points are depicted in green and old points are depicted in red. To check the accuracy of the method, eight distances between two

Table 1: tracking phase processing time.

Tracking 140 Points	Time in ms
Undistortion and Rectification	9
Pose Prediction	0.04
Get Points (inside Frustum)	0.1
Points Projection	2
ORB KeyPoints Detection	9
ORB Descriptor Computation	9
Matching	2
Pose Update	20
Total	~ 50

Table 2: Bundle Adj. processing time.

Map Points	Keyframes	Time in ms
242	2	75.134
313	3	212.537
306	4	773.285

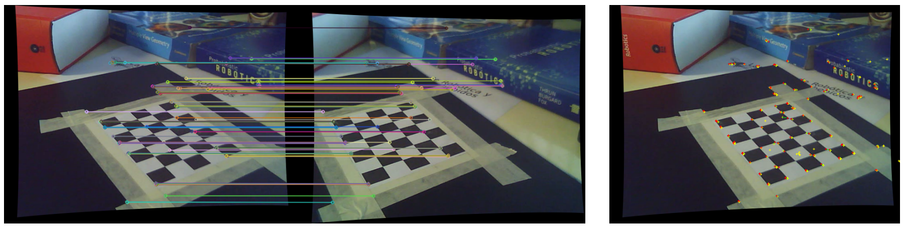
(a) (b)

Fig. 2: (a) shows the point initialization (matching between left and right features) from a keyframe. (b) shows the tracking after the Map expansion.

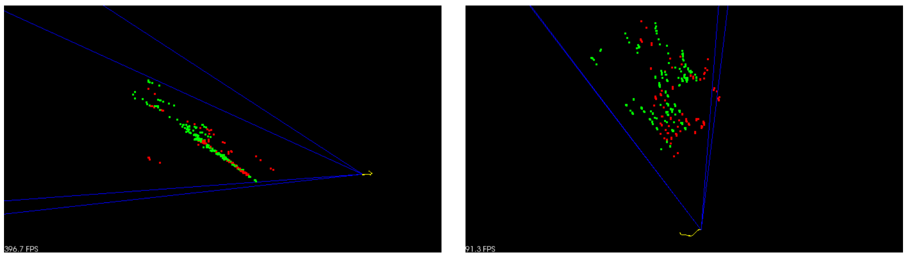
(a) (b)

Fig. 3: views of the reconstructed Map, (a) from side view, (b) from up view.

real points were used as ground-truth. Then, the euclidean distances between the points reconstructed by the STAM were computed. The Table. 3 and Fig. 6 show the accuracy of STAM method. The average error was 2.27 mm.

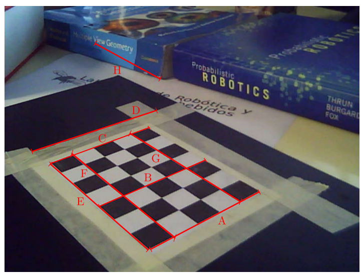

Fig. 4: ground truth measurements.

Table 3: comparison between STAM and the ground truth. All the measurements are in cm.

Segment	STAM	Ground Truth	Absolute error
A	8.29	8.6	0.31
B	8.58	8.6	0.02
C	8.74	8.6	0.14
D	11.33	11.44	0.11
E	12.68	12	0.68
F	12.17	12	0.17
G	12.02	12	0.02
H	10.98	11.35	0.37

7 Conclusions and on-going work

In this work we presented a stereo SLAM system that we call *STAM*, standing for Stereo Tracking and Mapping. This system estimates a local map of the surroundings using a stereo camera and tracks the camera pose from the stereo sequence in real-time. We incorporate the stereo constraints into the formulation of the system, so the maps produced are at metric scale. Specifically, we use both images of the stereo keyframes for map estimation and point initialization. As a result, we are able to overcome the well-known initialization problem [13, 14] even with a stationary or a rotating camera. The preliminary experiments presented in this work demonstrate the real-time performance of the proposed methodology and the accuracy of the estimated metric map. As future work we will perform experiments with real robots using a stereo rig with cameras that support 30 fps of frame rate and can be synchronized between them. We plan to improve the code in order to keep good real-time performance even for the case of large maps.

References

1. Smith, R.C., Cheeseman, P.: On the representation and estimation of spatial uncertainty. The international journal of Robotics Research **5**(4) (1986) 56–68
2. Civera, J., Davison, A.J., Montiel, J.M.M.: Inverse depth parametrization for monocular SLAM. IEEE Transactions on Robotics **24**(5) (October 2008) 932–945
3. Durrant-Whyte, H.F., Dissanayake, M.W.M.G., Gibbens, P.W.: Toward deployments of large scale simultaneous localisation and map building (SLAM) systems. In: International Symposium on Robotics Research (ISRR). (1999) 121–127
4. Konolige, K., Bowman, J.: Towards lifelong visual maps. In: International Conference on Intelligent Robots and Systems, 2009. IROS 2009, IEEE (2009) 1156–1163
5. Bao, S.Y., Savarese, S.: Semantic structure from motion. In: 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2011) 2025–2032
6. Strasdat, H., Montiel, J., Davison, A.: Real-time Monocular SLAM: Why Filter? In: IEEE International Conference on Robotics and Automation (ICRA). (2010)
7. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: International Symposium on Mixed and Augmented Reality. (2007)
8. Varadarajan, V.: Lie Groups, Lie Algebras and Their Representations. Number 102 in Graduate Texts in Mathematics. Springer - Verlag (1974)
9. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: An efficient alternative to sift or surf. In: International Conference on Computer Vision (ICCV). (2011)
10. Hoppe, C., Pirker, K., Rüther, M., Bischof, H.: Large-Scale Robotic SLAM through Visual Mapping. In: oagm2011.joanneum.at. (November 2010)
11. Klein, G.: Visual Tracking for Augmented Reality. PhD thesis, University of Cambridge (2006)
12. Rousseeuw, P.J., Leroy, A.M.: Robust Regression and Outlier Detection. John Wiley & Sons, Inc., New York, NY, USA (1987)
13. Civera, J., Davison, A., Montiel, J.: Interacting multiple model monocular SLAM. In: IEEE International Conference on Robotics and Automation (ICRA). (2008)
14. Gauglitz, S., Sweeney, C., Ventura, J., Turk, M., Hollerer, T.: Live tracking and mapping from both general and rotation-only camera motion. In: International Symposium on Mixed and Augmented Reality, IEEE (2012) 13–22