Closed-form Solution for IMU based LSD-SLAM Point Cloud Conversion into the Scaled 3D World Environment

Sergey Triputen, Kristiaan Schreve, Viktor Tkachev and Matthias Rätsch

Abstract—SLAM is a very popular research stream in computer vision and robotics nowadays. For more effective SLAM implementation it is necessary to have reliable information about the environment, also the data should be aligned and scaled according to the real world coordinate system. Monocular SLAM research is an attractive sub-stream, because of the low equipment cost, size and weight.

In this paper we present a way to build a conversion from LSD-SLAM coordinate space to the real world coordinates using a true metric scale with IMU sensor data implementation. The causes of differences between the real and calculated spaces are explained and the possibility of conversions between the spaces is proved.

Additionally, a closed-form solution for inter space transformation calculation is presented. The synthetic method of generating high level accurate and well controlled input data for the LSD-SLAM algorithm is presented.

Finally, the reconstructed 3D environment representation is delivered as an output of the implemented conversion.

I. INTRODUCTION

Robotic navigation and control is a complex task to deal with. Most positioning, orientation and odometry systems provide only pose data relative to the system start point. Therefore, it is necessary to accurately define the initial position and scale before motion starts. This is vital in order to integrate the SLAM system with other robot systems that interact with the environment. Several SLAM (Simultaneous Localization and Mapping) algorithms have been developed for this task. The main task of these algorithms is to build a map (based on the key-frames with loop closure detection) and also to propose a way to identify the current location of the robot during the localization phase [1]–[3].

Typical visual SLAM systems use either monocular, stereo or RGB-D cameras as a primary input device. Currently, systems using RGB-D cameras are most effective because they provide the depth data (distance to the detected point). The map can be used to calculate the scale factor of the camera path. However, relative to monocular systems, the hardware cost is the main disadvantage. Similar results can be achieved by combining monocular camera data with other sensors [4]–[6].

Another important robot control task is the correct detection of environmental obstacles. Detailed environmental information is necessary for solving path planning tasks. In this paper a solution using a monocular SLAM system for 3D environment reconstruction is proposed, as well as map generation and further processing into metric scale is shown. There are many existing SLAM algorithms described today. Most of them are dealing with standard navigation tasks (e.g. building key-frame based maps, loop closure detecting or current position localization). One of the more popular approaches is PTAM [7], [8] and its variations, such as LSD-SLAM [3], [9] and ORB-SLAM [1], [2].

PTAM and some other SLAM systems use feature point detection algorithms for frame tracking and finding the current camera position. These algorithms are computationally expensive and require significant resources. However some SLAM systems such as described in [8], [10] are using a featureless solution and are optimized for embedded system implementation.

A significant challenge for monocular SLAM systems is the problem of estimating scaled camera positions. Without scaled metric data, the SLAM output cannot be integrated with other robotic systems relying on accurate environmental information. That is why some approaches use RGB-D or stereo cameras [11], [12]. Depth maps generated from RGB-D cameras can be used to find scaled metric camera...
positions. Additionally RGB-D hardware makes 3D environment reconstruction possible [11]. This approach is more expensive than monocular systems.

However, scaled environmental information can be obtained if the monocular camera data is combined with other sensors, e.g. laser scanners, etc. The PTAM algorithm uses an altimeter based solution [13]. PTAM does not provide scaled 3D environmental data, though.

There is no monocular SLAM system that provides a scaled point cloud representation of the environment aligned to some global coordinate system.

The main contribution of the paper is to present a method to combine LSD-SLAM estimated camera position with IMU sensor data in a real world coordinate system (which is perpendicular to the earth’s surface). It is also shown how the aligned (position, orientation and scale) result can be used for octree mesh representation.

II. SPACES ALIGNMENT PROBLEM SOLUTION

A method to convert the estimated LSD-SLAM point cloud scale and orientation into the real world scale and absolute orientation is proposed in this paper. For the analysis of the results’ accuracy we use simulation data, which is obtained from a synthetic world. It takes into account noise measurements of a real IMU sensor.

This chapter presents an overview of the problem and gives the necessary data annotation. Further, the problem solution proposal and the results evaluation criteria are described.

A. Problem definition

The main steps of LSD-SLAM and real world scale and orientation synchronization are taken during the LSD-SLAM algorithm initialization period. We should remember, that LSD-SLAM systems without spacial equipment (such as an external IMU sensor) have no information about the real world camera position, orientation and surrounding object sizes.

The LSD-SLAM camera position and orientation estimating process is based on tracking special points in a particular frame. It is necessary to have depth information for them. Initially, it uses random numbers for the depth hypotheses, i.e. \( d \in [0, 1] \) random numbers for each feature point in the first frame.

Also, for the first frame camera position and orientation the initial position vector \( [0, 0, 0] \) and quaternion \( [1, 0, 0, 0] \) are used.

This random distance is used for camera tracking and for defining the estimated camera position and orientation. At the next step the calculated camera position and orientation error is used in the triangulation task to update the point cloud depth information. After some iterations, the calculation results are converged into an optimal combination of depths and camera tracking data.

It is obvious that after the initialization period the estimated point depths and camera position and orientation data have unpredictable values. The initialization time and final distance scale factor depend on a random distribution of initial hypothetical depths.

Now the task can be formulated: we are looking for a combination of rotation, scaling and translation factors that will map LSD-SLAM estimated coordinate system space to the IMU sensor coordinate system.

It is very important to have the transformation from LSD-SLAM space to the sensor space, because the IMU sensor coordinate system is aligned to the gravity vector. That means it is aligned to the world coordinate system. It makes it possible to use an octree optimized 3D model representation for the estimated point cloud.

B. Problem solution

In order to find a solution, first of all, it is necessary to formally define the input data. LSD-SLAM estimates data as a set of key-frames which describe camera position and orientation with a set of points or coordinates (or estimated distance from camera to each feature point). This will be denoted as \( L \).

\[
L = \{T,\ P\};
\]  

(1)

Where \( T \in \text{Sim}(3) \) such that:

\[
T = \left( \begin{array}{cc}
 sR & t \\
 0 & 1 \\
\end{array} \right) \quad \text{with} \quad R \in \text{SO}(3),
\]

(2a)

\[
t \in \mathbb{R}^3,
\]

(2b)

\[
s \in \mathbb{R}^+.
\]

(2c)

The \( \text{Sim}(3) \) group represents transformation in a three dimensional space. There \( s \) is a scale factor and \( t \) a translation vector in 3D space.

In the equation (2a), \( R \) is an element of the \( \text{SO}(3) \) group that represents rotation. Each element of the \( \text{SO}(3) \) group can be represented in several different forms (e.g. rotation matrix, quaternion or vector and angle combination).

\( T \) represents the camera transformation estimated by LSD-SLAM. Furthermore

\[
P = \{ p = (u, v, d) \mid u \in \mathbb{Z}^+, \quad v \in \mathbb{Z}^+,
\]

(3a)

\[
d \in \mathbb{R}^+ \};
\]

(3c)

where \( [u, v] \) are the pixel coordinates on the frame and \( d \) is a distance from the camera position to the point.

In order to calculate 3D coordinates of the points in a key-frame point cloud, it is also necessary to have the intrinsic camera matrix:

\[
M = \begin{bmatrix}
 f_x & 0 & c_x \\
 0 & f_y & c_y \\
 0 & 0 & 1 \\
\end{bmatrix}
\]

(4)

Where:
\( f = [f_x, f_y] \) - camera focus distance per axis.
\( c = [c_x, c_y] \) - image position of the principle axis in pixel coordinates.

Let \( X = \{ x \mid x \in \mathbb{R}^3 \} \) represent \( P \) in the 3D space of the LSD-SLAM coordinate system. It can be calculated by mapping:

\[
\delta: P \rightarrow X
\]

Where \( \delta \) was specified like:

\[
X = \delta (L, M) = T \left[ \frac{(u - c_x) d}{f_x} \right] \frac{(v - c_y) d}{f_y} 1
\]

(5)

We are receiving additional information from an external sensor, in this case an IMU. Similar to equation (1) for LSD-SLAM estimated data, we will use set \( S \), which describes sensor measured data

\[
S = \{ T', P' \};
\]

For IMU sensor measured data \( T' \in \text{Sim}(3) \) is a transformation into the real world metric coordinate system and \( P' = \emptyset \) is an empty set, because the IMU model do not have any information about the 3D real world geometry. Because \( P' \) is empty, \( X' \) cannot be obtained.

As described in the previous subsection, we are looking for a combination of rotation, translation and scale conversions from \( L \) to \( S \).

Formally that conversion combination can be represented as an element of \( \text{Sim}(3) \):

\[
\Lambda = \begin{pmatrix} sR & t \\ 0 & 1 \end{pmatrix} \text{ with } R \in \text{SO}(3),
\]

\[
t \in \mathbb{R}^3,
\]

\[
s \in \mathbb{R}^+
\]

Finally, we can formalize our task as follows: we are looking for a mapping

\[
\Lambda: L \rightarrow S;
\]

In order to find \( \Lambda \) we can only use the LSD-SLAM camera position and orientation and then relate it to the IMU sensor measurements. As already mentioned \( P' \) (hence \( X' \) as well) is an empty set for the sensor coordinate system. Therefor the equation for finding \( \Lambda \) becomes:

\[
\Lambda^* := \arg \min_{(s, R, t)} \left( \sum_{i=0}^n \left\| T' - \Lambda \times T \right\|^2 \right)
\]

(6)

The element usage is based on the fact that \( T \) and \( T' \) are part of the \( \text{Sim}(3) \). The solution to this equation is discussed in detail in [14, 15].

And so, we are able to reconstruct the point cloud in the IMU sensor coordinate system. With the calculated \( \Lambda^* \), we can transform the LSD-SLAM estimated point cloud to the IMU sensor coordinate system. This can be achieved using equation [5] and the result of equation [6]

\[
X' = \Lambda^* \times X \approx \Lambda^* \times \delta (L, M)
\]

(7)

C. Closed-form solution

Horn [14] gives the closed-form solution for solving equation [6]. This solution is summarized in this algorithm:

1) Find the centroids \( t \) and \( t' \) of the LSD-SLAM estimated \( T \) and IMU sensor measured \( T' \) sets:

\[
\bar{t} = \frac{1}{n} \sum_{i=1}^n t_i
\]

\[
\bar{t}' = \frac{1}{n} \sum_{i=1}^n t'_i
\]

where \( n \) is a frames number.

2) The centroids are subtracted from all elements of sets

\[
\hat{t} = t - \bar{t}
\]

\[
\hat{t}' = t' - \bar{t}'
\]

so that from now on, we deal only with values relative to the centroids.

3) For each pair of coordinates we compute the nine possible products \( t_x t'_x, t_x t'_y, \ldots, t_z t'_z \) of the components of the two vectors. This is added up to obtain

\[
S_{xx} = \sum_{i=1}^n \hat{t}_x \hat{t}'_x,
\]

\[
S_{xy} = \sum_{i=1}^n \hat{t}_x \hat{t}'_y,
\]

and so one. These nine totals contain all the information that is required to find the solution.

4) The rotation quaternion finding \( \hat{e} \) by solving homogeneous equation

\[
[N - \lambda I] \hat{e} = 0
\]

There matrix \( N \) is determined as

\[
N = \begin{bmatrix}
 a & e & h & j \\
 e & b & f & i \\
 h & f & c & g \\
 j & i & g & d
\end{bmatrix}
\]

with elements

\[
a = (S_{xx} + S_{yy} + S_{zz}),
\]

\[
b = (S_{xx} - S_{yy} - S_{zz}),
\]

\[
c = (-S_{xx} + S_{yy} - S_{zz}),
\]

\[
d = (-S_{xx} - S_{yy} + S_{zz}),
\]

\[
e = (S_{yz} - S_{zy}),
\]

\[
f = (S_{xy} + S_{yx}),
\]

\[
g = (S_{yz} + S_{zy}),
\]

\[
h = (S_{xx} - S_{zz}),
\]

\[
i = (S_{xx} + S_{zz}),
\]

\[
j = (S_{xy} - S_{yx})
\]

5) At this point, compute the scale as:

\[
s = \frac{\sum_{i=0}^n \hat{t}' \cdot R(\hat{e})}{\sum_{i=0}^n \| \hat{t}' \|^2}
\]
6) Computation of the translation as the difference between the centroid of the IMU sensor measurements and the scaled and rotated centroid of the LSD-SLAM estimated data

$$r_0 = \bar{t}' - sR(\bar{t})$$

where $r_0 \in \mathbb{R}^3$ is translation vector.

**Note:**

The case, described in [14] assumes the two sets which are well enough synchronized from the start point. In our case, there are several unsynchronized initial sets. It is recommended to exclude these unsynchronized sets. To solve this task we propose to analyze frame to frame vector length factors. These factors should be calculated as:

$$p_k = \frac{\|t'_{k+1} - t'_k\|^2}{\|t_{k+1} - t_k\|^2}$$

where $k$ is the frame number.

As a result we present a method to transform point cloud SLAM data into the IMU sensor coordinate system (gravity vector aligned).

### III. INPUT DATA ESTIMATION

All SLAM algorithms rely on and are very sensitive to input data and ambient conditions (such as color gradient, light, contrast etc.). In order to evaluate the quality of the obtained conversion we need to have precise information of the camera position and orientation as an input. The best way to achieve precise camera position and orientation data is to make a controlled simulation with known parameters (including environment dimensions). An animation application tool which makes this possible is 'Blender'.

![Fig. 3: Test Blender scene for conversion research.](image)

**B. Camera parameters**

In order to have realistic results from LSD-SLAM, we need a definite set of camera parameters. We assume the parameters described below.

- Camera Frame Per Second (FPS).
  FPS determines the achievable camera motion speed. The higher the FPS, the higher camera motion speed may be.
- Frame Resolution.
  Selection of this parameter is a balance between having enough resolution to locate feature points and computational time.
- Lens Focus Distance.
  This parameter is used to calculate the 3D position of the feature point, as shown in equations 4 and 5.
- Lens Distortion Parameters.
  For cameras with a big field of view ("Fish Eye" effect) it is very important to use distortion coefficients (camera undistorted). For our experiments we model a non "Fish eyed" camera, so the mentioned coefficients are equal to zero.

For the simulation we have used camera parameters of the Sony PlayStation Eye 3 as an input to LSD-SLAM. PS Eye 3 was selected since tests in our laboratory have demonstrated its success with LSD-SLAM.

The camera parameters are stated in Table I.

**TABLE I: PS3 EYE CAMERA SPECIFICATIONS**

| Resolutions              | 640 × 480 pixels @ 60 Hz |
|--------------------------|---------------------------|
| Field Of View            | ϕ/5°                      |

**C. External environment parameters**

- Geometry dimensions
  For the present work it is very important to have a method to evaluate point cloud accuracy. It is not a trivial task in real word applications. If tested in real environments, it will be very complicated to compare the point cloud and camera position data with, e.g. 3D scanner or sensor data.

We used the Blender application to build a virtual world with well-known geometric dimensions and to export it into MATLAB for final analysis.
Flexible texture control and light exposition.
In Blender, unlike real world camera perception, it is possible to control the LSD-SLAM inputs (such as light, shade, contrast, etc.). This makes it possible to create precise environmental model scenarios to verify the algorithm results. See Fig. 4 for an example of a typical scene used in this study.

D. Camera position and orientation (IMU or odometry data simulation)

Similar to environmental input for the LSD-SLAM algorithm analysis, it is beneficial to have a controlled simulation of camera position and motion.

- Camera trajectory (length, loops, curvature angels)
  From the beginning of the simulation we have a specified ‘zero’ position. Then, the ground truth path is controlled, taking into account the IMU data. The simulation ensures no motion occurs which makes LSD-SLAM results unpredictable (e.g. camera angular velocity, uncertainty of feature points due to insignificant rotation angle, distance, etc.).
- Camera motion speed (constant or variable along path) is also simulated in a controlled way to allow accurate interpretation of LSD-SLAM results.

In order to eliminate real environment, robot and camera input inaccuracy, simulation with Blender is proposed. The simulation technology provides enough spacial dimension and environment resolution to recreate real environment parameters (e.g. light, shade, gradient).

LSD-SLAM algorithm results with simulated input are tested and delivered with accurate results if compared to the algorithm results with the real camera and environment input data.

IV. RESULTS

The set of frames taken of the Blender scene (Fig.3) is used as input for LSD-SLAM. The results are received from LSD-SLAM and are represented by the LSD-SLAM viewer as shown in Fig.5.

The point cloud in this figure orientation was chosen manually in order to make the presentation more understandable.

A. Basic scene analyzing

In order to make the comparison of the size and orientation of point cloud and camera path, MATLAB is used (Fig.6b). The ground truth path and scene geometry were exported from Blender and imported to MATLAB. It is represented in (Fig.6a).

As expected, there is a big difference in the scale factor and orientation representation between the two figures. It is possible to see this by comparing the axis sizes.

Additionally, 6a and 6b show that XYZ axes (presented with Red, Green and Blue respectively) do not match by orientation.

The basic scene parameters are:

(a) Camera path exported from Blender

(b) Point cloud and camera position and orientation estimated by an original LSD-SLAM project and LSD-SLAM viewer represented
• Total frames number - 7000;
• Key frames number - 123;
• Simulation period - 116.7 seconds;
• Blender path length - 195.12 ground truth meters;
• LSD-SLAM estimated path length - 12.6 LSD-SLAM internal units.

B. Initialization time analyzes

Now we can try to analyze the initialization time. We used equation 9 from chapter II-C. The calculation results are represented in Fig. 7.

The depicted trend is showing the three key stages:
1. Obviously occasional deviations (the starting stage, shown in red dots).
2. Convergence period (the second stage with outlined trend, shown in black dots, 1100 – 4300 counts). Still not stabilized.
3. Stabilized stage, after 4300 count. Still has visible bursts, but trend is kept stable (black dots after 4300).

![Fig. 7: Scale factors analyzing.](image)

C. Conversion results

The conversion results of the position and orientation of the camera and the point cloud coordinates are represented in Fig[8]

Fig[8a] shows that the obtained point cloud coordinates are aligned according to the Blender. As well as the axis directions and scales are aligned.

Fig[8b] shows a fragment of the converted scene.

**CONCLUSION**

According to the achieved results, the targeted transformation shows that the precision for estimated 3D environment for octree map representation is enough according to the objective of this article.

It is important to mention that LSD-SLAM algorithm gives high precision for the point cloud. Consequently the obtained 3D reconstruction is reliable to be used in path planning or follow me tasks.

At the same time, there are several technical points revealed in the current research that are to be considered. One of the mentioned technical points: the scale coefficient is not constant all the time. According to the motion path, if a frame is repeated frequently, the scale should be estimated and adjusted in the process.

**REFERENCES**

[1] R. Mur-Artal and J. Tardos, “ORB-SLAM: Tracking and Mapping Recognizable Features.” Robotics: Science and Systems (RSS) Workshop on Multi View Geometry in Robotics (MVIGRO), 2014.

[2] R. Mur-Artal, Tardos, and Juan, “Probabilistic Semi-Dense Mapping from Highly Accurate Feature-Based Monocular SLAM.” Robotics: Science and Systems., 2015.

[3] J. Engel, T. Schps, and D. Cremers, “LSD-SLAM: Large-Scale Direct Monocular SLAM.” European Conference on Computer Vision (ECCV), 2014.

[4] V. Usenko, J. Engel, J. Stickler, and D. Cremers, “Direct visual-inertial odometry with stereo cameras.” Robotics and Automation (ICRA), 2016.

[5] J. Engel, J. Stueckler, and D. Cremers, “Large-Scale Direct SLAM with Stereo Cameras.” International Conference on Intelligent Robots and Systems (IROS), 2015.

[6] J. Engel, J. Sturm, and D. Cremers, “Scale-Aware Navigation of a Low-Cost Quadrocopter with a Monocular Camera.” Robotics and Autonomous Systems (RAS), 2014.

[7] G. Klein and D. Murray, “Parallel Tracking and Mapping for Small AR Workspaces.” In Proc. International Symposium on Mixed and Augmented Reality (ISMAR‘07, Nara), 2007.

[8] ——, “Parallel Tracking and Mapping on a Camera Phone.” In Proc. International Symposium on Mixed and Augmented Reality (ISMAR’99, Orlando), 2009.

[9] J. Engel, J. Sturm, and D. Cremers, “Semi-Dense Visual Odometry for a Monocular Camera.” IEEE International Conference on Computer Vision (ICCV), 2013.

[10] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “FastSLAM: a factored solution to the simultaneous localization and mapping problem.” Proceeding Eighteenth national conference on Artificial intelligence., 2002.

[11] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, “An Evaluation of the RGB-D SLAM System.” Robotics and Automation, 2012.

[12] R. Maiar, J. Sturm, and D. Cremers, “Submap-based Bundle Adjustment for 3D Reconstruction from RGB-D Data.” In German Conference on Pattern Recognition (GCPR), 2014.

[13] J. Engel, J. Sturm, and D. Cremers, “Camera-Based Navigation of a Low-Cost Quadrocopter.” International Conference on Intelligent Robot Systems (IROS), 2012.

[14] B. Horn, “Closed-form solution of absolute orientation using unit quaternions.” Journal of the Optical Society of America, 1987.

[15] B. Horn, H. Hilden, and S. Negahdarpour, “Closed-form solution of absolute orientation using orthonormal matrices.” Journal of the Optical Society of America, 1988.