Robotica (2018) © Cambridge University Press 2018
doi:10.1017/xxxx

# Real-time Dense Map Fusion for Stereo SLAM

Taihú Pire†*, Rodrigo Baravalle†, Ariel D'Alessandro† and Javier Civera‡

†*CIFASIS, French Argentine International Center for Information and Systems Sciences (CONICET-UNR), Argentina*
‡*University of Zaragoza, Spain*

(Accepted MONTH DAY, YEAR. First published online: MONTH DAY, YEAR)

**SUMMARY**
A robot should be able to estimate an accurate and dense 3D model of its environment (a map), along with its pose relative to it, all of it in real time, in order to be able to navigate autonomously without collisions.

As the robot moves from its starting position and the estimated map grows, the computational and memory footprint of a dense 3D map increases and might exceed the robot capabilities in a short time. However, a global map is still needed to maintain its consistency and plan for distant goals, possibly out of the robot field of view.

In this work we address such problem by proposing a real-time stereo mapping pipeline, feasible for standard CPUs, which is locally dense and globally sparse and accurate. Our algorithm is based on a graph relating poses and salient visual points, in order to maintain a long-term accuracy with a small cost. Within such framework, we propose an efficient dense fusion of several stereo depths in the locality of the current robot pose.

We evaluate the performance and the accuracy of our algorithm in the public datasets of Tsukuba and KITTI, and demonstrate that it outperforms single-view stereo depth. We release the code as open-source, in order to facilitate the system use and comparisons.

KEYWORDS: Dense Mapping; Visual SLAM; Stereo Vision.

## 1. Introduction

Planning safe trajectories towards a given goal, while moving in an unknown environment, is one of the core components of any autonomous system. In order to achieve such capability, the robot needs to estimate its egomotion and a dense 3D map of the scene from its sensor data.

SLAM (standing for Simultaneous Localization and Mapping) addresses the problem of estimating an incremental map of its environment and, at the same time, the robot ego-pose.[2,4,9] The earliest of the SLAM systems used mainly lasers as sensors.[5] The computational and algorithmic advances of the last decades have enabled the use of cameras as a very convenient –cheap, small, low power– and promising alternative. However, from the early days of visual SLAM and with not many exceptions, the estimated maps are either sparse[8] or semidense,[11] limiting their use for autonomous robots.

Dense, accurate and high-resolution environment models are essential for computing safe robot trajectories. Direct SLAM methods have increased the density of the visual scene models. But textureless areas are not mapped by the so-called semidense visual SLAM methods[11] and hence safe navigation is not possible. The fully dense approaches based on *Total Variation* (TV) regularization ([15,22,30] among others) are expensive, and

* Corresponding author. E-mail: pire@cifasis-conicet.gov.ar

have only been demonstrated at a small scale. Also, their accuracy might be low for large textureless areas.[6]

This work proposes and evaluates a system based on feature-based SLAM, and incorporating a dense local map for robots to navigate safely in their surroundings. We will call such approach *Dense S-PTAM*. We use a stereo camera, a very practical alternative for robotics that (compared against monocular vision) gives metric information from the very first frame and (compared against active RGB-D sensors) can be safely used outdoors without interferences.

To our knowledge, *Dense S-PTAM* is the first locally dense stereo algorithm that runs on CPU real time. Our experimental results validate our multi-view depth fusion proposal, showing that we outperform the accuracy of single-view dense stereo depth. We release the code of our system, integrated with ROS, as open-source to facilitate its use and comparison[1].

The pipeline of our system can be summarized as follows. We use S-PTAM,[25,26] a feature-based stereo SLAM system, in order to have a globally consistent camera pose estimation. We use LIBELAS[14] to estimate an efficient and real-time dense depth from every stereo keyframe. And we propose an efficient depth fusion algorithm to improve the single-view stereo depth and produce locally consistent and accurate depth maps.

The structure of the rest of the paper is as follows: Section 2 details the current state of the art in stereo dense environment reconstructions. Section 3 presents our *Dense S-PTAM* method. Section 4 shows and discusses our experimental results. Finally, in Section 5 we conclude and outline the future work.

## 2. Related Work

### *2.1. Sparse/semidense stereo SLAM*

Feature-based stereo odometry and SLAM is nowadays a mature field, with several systems achieving high accuracy and robustness in large-scale scenes (two of the most recent ones are[21,25]). There are also recent works that use direct methods for semidense stereo SLAM[11] and odometry.[34] Very interestingly, direct methods have shown recently a better accuracy than feature-based ones for incremental motion estimation,[10] so they seem a promising direction for future work in SLAM.

### *2.2. Dense depth from stereo*

Dense matching/correspondence/disparity/depth/reconstruction from a single stereo pair is a classical problem in the robotics and computer vision research community with a huge literature corpus available. Tippetts, et al.[32] is a recent survey on some of the most relevant algorithms, with emphasis on real-time performance. We refer the reader to this reference for details on the state of the art.

### *2.3. Dense large-scale visual SLAM*

There are not many approaches to dense stereo SLAM for large-scale environments.[12] is a seminal paper on this area. The recent[17] maintains a globally consistent semidense map and a locally dense one. However, they use a variational approach to achieve smooth multiview stereo reconstructions, which is computationally expensive (it runs in GPU real time). Variational methods have been used for local GPU-real-time visual SLAM in several works, e.g.[22,27] Our algorithm, based on fusing dense stereo depth maps, is able to estimate dense local maps from stereo in real time on a standard CPU.

Tanner, et al.[31] uses a variational formulation to estimate large-scale maps from stereo data, but again uses GPU processing and do not present real-time results. Alcantarilla, et al.[1] and Sengupta, et al.[29] also show large-scale results and accuracies similar to our system. Their reported computational times are, however, much larger than ours. None

[1] `https://github.com/CIFASIS/dense-sptam`

of the works mentioned so far in this subsection is associated to open-source code for comparisons.

Dense visual maps have been also achieved by different sensor combinations. Schöps, et al.[28] and Klingensmith, et al.[16] use the monocular, inertial and depth sensors of Google Tango. However, they achieve real-time performance using GPU processing. Both use the Truncated Signed Distance Function (TSDF). Oleynikova, et al.[23] also uses the TSDF and stereo images for MAV navigation. Concha, et al.[7] fuses inertial and monocular data and assumes a multiplanar environment. There are also several works that combine stereo and laser readings,[19,20] among others.

More sophisticated approaches estimate jointly the 3D reconstruction of a scene and its semantic labeling.[3,18,29,33] They do not show, however, CPU real time results, their code is not available for comparisons and their robustness has not been thoroughly tested in multiple environments.

## 3. Dense S-PTAM

### *3.1. Globally accurate feature-based SLAM*

We use S-PTAM[25,26] as our globally accurate stereo SLAM[2]. We will briefly summarize its main components here for completion and refer the reader to the original paper for more details.

S-PTAM is composed of the following tracking and mapping modules.

**Stereo camera tracking:** The motion of the current stereo frame, $\boldsymbol{\mu}_t$, is estimated by minimizing the reprojection error $\boldsymbol{\Delta}\mathbf{z}_i$ for each tracked point $\mathbf{x}_i$.

$$\hat{\boldsymbol{\mu}} = \arg\min_{\boldsymbol{\mu}} \sum_i \rho(\mathbf{J}_i \boldsymbol{\mu}_t - \boldsymbol{\Delta}\mathbf{z}_i) \tag{1}$$

$\mathbf{J}_i = \frac{\partial \boldsymbol{\Delta}\mathbf{z}_i}{\partial \boldsymbol{\mu}_t}$ is the Jacobian of the reprojection error $\boldsymbol{\Delta}\mathbf{z}_i$ with respect to the camera motion $\boldsymbol{\mu}_t$ –the map points $\mathbf{x}_i$ are fixed in the tracking thread.

Once the current camera pose is estimated, its associated stereo frame becomes a keyframe candidate and it is added to the map if certain heuristics hold (related to the camera motion and time since the last keyframe was added, and to the overlap between the map and the current field of view).

The tracking initialization results straightforward thanks to the stereo sensor. The local frame of the first stereo pair is the global reference frame. And the initial map is created by triangulating salient points from this initial stereo pair.

**Stereo Mapping:** The map estimated by S-PTAM is composed by a sparse set of $N$ 3D points $\{\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_N\}$ and $M$ keyframes $\{\mathcal{K}_1, \ldots, \mathcal{K}_j, \ldots, \mathcal{K}_M\}$ with their corresponding poses $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_j, \ldots, \boldsymbol{\mu}_M\}$. The local map is estimated by a Bundle Adjustment over a local window of keyframes $\mathcal{W}$ and the points falling within their field of views $FOV(\mathcal{W})$

$$\{\hat{\boldsymbol{\mu}}_{\mathcal{W}}, \hat{\mathbf{x}}_{FOV(\mathcal{W})}\} = \arg\min_{\mathbf{x}_i, \boldsymbol{\mu}_j} \sum_{\mathcal{W}} \sum_{FOV(\mathcal{W})} \rho(\mathbf{J}_{i,j} \boldsymbol{\mu}_j - \boldsymbol{\Delta}\mathbf{z}_{i,j}) \tag{2}$$

The maintenance of the incremental map is as follows. When a new keyframe is added from the tracking thread, S-PTAM triangulates new 3D points from its stereo matches. The camera pose is also added to the points-poses map.

Once this is done, the mapping thread actively searches for point correspondences between keyframes, in order to strengthen the constraints of the point-pose graph.

[2] Code available at `http://github.com/lrse/sptam`

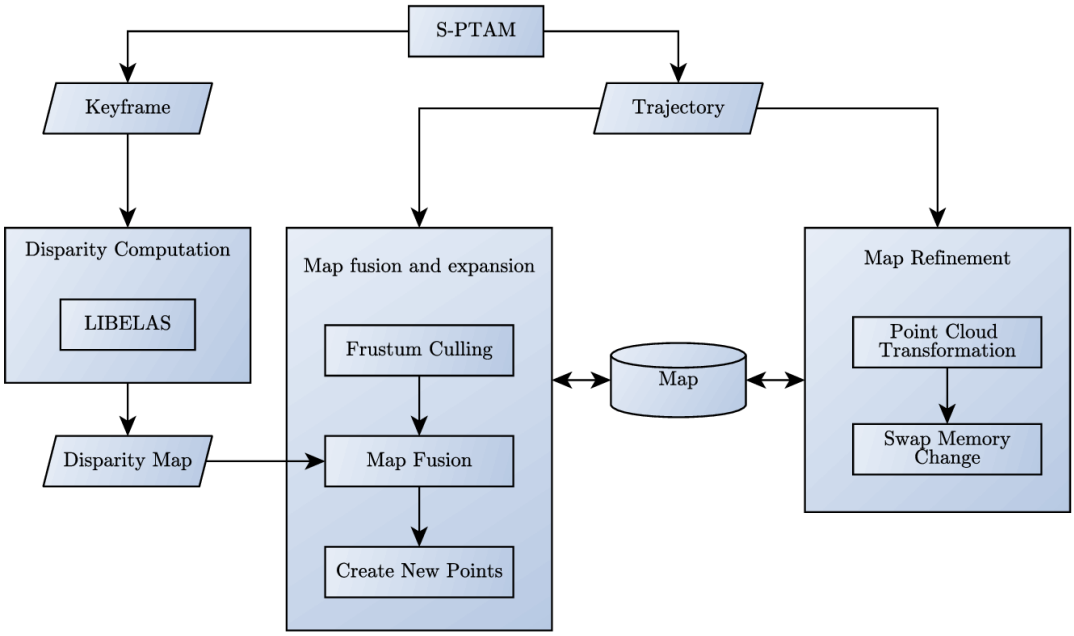



Fig. 1: Overview of the proposed *Dense S-PTAM*.

Immediately after, the mapping thread performs a Local Bundle Adjustment (LBA) over the point-pose subgraph, defined by the last added keyframes.

### *3.2. Dense Local Mapping*

The approach presented in this paper, *Dense S-PTAM*, adds a dense local map over the feature-based map of S-PTAM. It uses the pose estimation from S-PTAM and the disparity from the stereo keyframes, and fuses the dense 3D point clouds in an efficient manner. Figure 1 shows a scheme of the *Dense S-PTAM* pipeline.

Our method starts by estimating a disparity map $\mathcal{D}_j$ for every keyframe $\mathcal{K}_j$ when it is added to the map by the tracking thread of S-PTAM (*Disparity Computation* box in Figure 1). Using the stereo calibration, we transform the disparity map $\mathcal{D}_j$ into a point cloud $\mathcal{P}_j$.

From these point clouds $\mathcal{P}_j$, estimated for each keyframe, the ones closest to the current pose are fused in the *Map Fusion and Expansion* procedure. Points that are close in 3D space are fused based on their respective covariances.

In order to maintain a dense map consistent with the S-PTAM estimation, every time S-PTAM refines a keyframe pose, the *Map refinement thread* updates both the keyframe pose and its corresponding point clouds.

We implemented *Dense S-PTAM* in a separate ROS node, making it easier to reuse it with other SLAM implementations. In the next subsections, we describe in detail the specific formulation of the *Dense S-PTAM* fusion.

### *3.3. Disparity Computation*

A disparity map $\mathcal{D}_j : \mathbb{R}^2 \rightarrow \mathbb{R}$ is a function that, for each pixel $(u, v)^\top$ of rectified stereo pair, gives as output its disparity value $d$. In this work we use LIBELAS[14] to compute disparity maps efficiently and accurately[3] .

LIBELAS follows a Bayesian approach, computing robust matches between the stereo images –support points–, and then triangulating them to form a prior distribution. This helps to reduce stereo matching ambiguities when compared to other disparity methods.

[3] Code available at `http://www.cvlibs.net/software/libelas/`

Also, it does not need global optimization, achieving near real time frame rates on high resolution images. The stereo camera is undistorted and rectified for an efficient disparity map computation. For more details on the LIBELAS disparity computation the reader is referred to the original source.[14]

### *3.4. Map Fusion and Expansion*

From the disparity map $\mathcal{D}_j$ we can estimate an inverse depth map straightforwardly. The inverse depth $\rho_i$ for each pixel $i$ is

$$\rho_i = \frac{d_i}{fb} \tag{3}$$

being $f$ the focal length and $b$ the baseline of the rectified stereo pair. The backprojection of the pixel $i$ at such inverse depth $\rho_i$ results in the 3D point $\mathbf{x}_i$. And the reconstruction of all the pixels in the keyframe $\mathcal{K}_j$ gives us the dense point cloud $\mathcal{P}_j = \{\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_P\}$. As the camera moves, new areas of the 3D scene will appear in the images. And also, the reconstruction will be more accurate from viewpoints that closer to the scene than farther ones. Our aim is to estimate a local point cloud $\mathcal{P}_{(j-J):(j)}$ accumulating the dense reconstructions from the last $J+1$ stereo keyframes $\{\mathcal{P}_{j-J}, \ldots, \mathcal{P}_j\}$.

We will do the fusion sequentially. Fig. 2 shows a schematic view for clarification. Given $\mathbf{x}_{\text{previous}} \in \mathcal{P}_{(j-J):(j-1)}$, a point from the local point cloud up to the $(j-1)$ keyframe, and $\mathbf{x}_{\text{current}} \in \mathcal{P}_j$ that belongs to the point cloud of the $j^{th}$ keyframe, we consider that they correspond to the same 3D point if:

- Their projection on close stereo keyframes falls into the same pixel coordinates.
- The Euclidean distance between their 3D coordinates falls below a certain threshold $\epsilon$.

If the two conditions hold, we fuse the two points. The result is $\mathbf{x}_{\text{fusion}} \in \mathcal{P}_{(j-J):(j)}$, that is calculated as

$$\mathbf{x}_{\text{fusion}} = \frac{1}{\rho_{\text{fusion}}} \frac{\mathbf{x}_{\text{current}}}{\|\mathbf{x}_{\text{current}}\|} = \frac{1}{\rho_{\text{fusion}}} \mathbf{n}_{\text{current}} \ . \tag{4}$$

The direction of the fused point is the same as the direction of the point in the current keyframe $\mathbf{n}_{\text{current}}$. The inverse depth of the new point $\rho_{\text{fusion}}$ is the average inverse depth over the $k$ keyframes in which such point was imaged ($k \leqslant J+1$)

$$\rho_{\text{fusion}} = \frac{k-1}{k} \rho_{\text{previous}} + \frac{1}{k} \rho_{\text{current}} \ . \tag{5}$$

$\rho_{\text{current}} = \dfrac{1}{\|\mathbf{x}_{\text{current}}\|}$ and $\rho_{\text{previous}} = \dfrac{1}{\|\mathbf{x}_{\text{previous}}\|}$ are the inverse depths of the points $\mathbf{x}_{\text{current}}$ and $\mathbf{x}_{\text{previous}}$ respectively.

Notice in equation 3 that the inverse depth $\rho_i$ has a linear relation with the disparity $d_i$. A first order propagation gives us constant inverse depth covariance, and hence the average in equation 5 is taking into account the stereo depth uncertainty.

It is also worth remarking that the assumptions we make in our fusion algorithm are two. First, that the uncertainty from LIBELAS comes from the geometric propagation of the matching error, and the contribution of other processes (e.g., smoothing) is negligible. We consider this is true in most of the cases, as stereo depth is usually smoothed but without altering significantly the stereo depth in textured regions.

A second assumption is that the inverse depth of a point is similar from two different views. This is a reasonable assumption if the local keyframes are close. And it holds in our case, as we incrementally fuse the current local point cloud $\mathcal{P}_{(j-J):(j-1)}$ with the cloud from the latest keyframe $\mathcal{P}_j$. Let $\rho_i^{\mathcal{K}_j}$ and $\mathbf{n}_i^{\mathcal{K}_j}$ be the inverse depth and projection ray

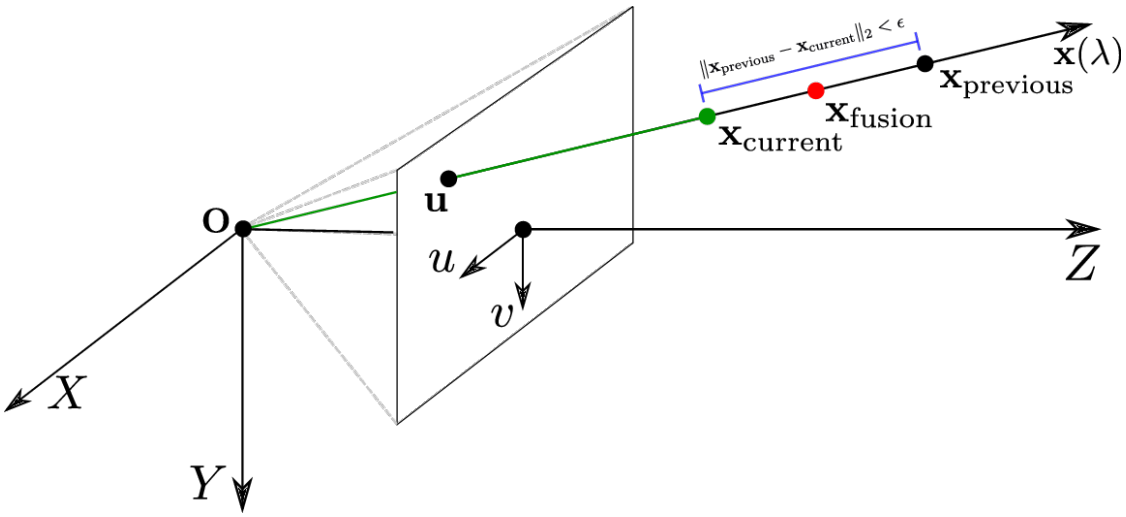



Fig. 2: Illustration for the depth fusion algorithm. $\mathbf{x}_{\text{current}}$, $\mathbf{x}_{\text{previous}}$ and $\mathbf{x}_{\text{fusion}}$ are respectively the triangulated point from the current keyframe, the point from previous depth fusions, and the result of the current fusion. The image also shows the threshold criterion $\epsilon$.

of point $i$ in the reference frame of $\mathcal{K}_j$ and $\mathbf{t}_{\mathcal{K}_1\mathcal{K}_2}$ and $R_{\mathcal{K}_1\mathcal{K}_2}$ the translation vector from keyframe $\mathcal{K}_1$ to keyframe $\mathcal{K}_2$. The following holds

$$\frac{1}{\rho_1^{\mathcal{K}_1}}\mathbf{n}_1^{\mathcal{K}_1} = \mathbf{t}_{\mathcal{K}_1\mathcal{K}_2} + R_{\mathcal{K}_1\mathcal{K}_2}\frac{1}{\rho_1^{\mathcal{K}_2}}\mathbf{n}_1^{\mathcal{K}_2} \tag{6}$$

Solving for the vector modules and making the assumption of close keyframes $\mathbf{t}_{\mathcal{K}_1\mathcal{K}_2} \approx 0$

$$\|\frac{1}{\rho_1^{\mathcal{K}_1}}\mathbf{n}_1^{\mathcal{K}_1}\| = \|\mathbf{t}_{\mathcal{K}_1\mathcal{K}_2} + R_{\mathcal{K}_1\mathcal{K}_2}\frac{1}{\rho_1^{\mathcal{K}_2}}\mathbf{n}_1^{\mathcal{K}_2}\| \tag{7}$$

$$\rho_1^{\mathcal{K}_1} \approx \rho_1^{\mathcal{K}_2} \tag{8}$$

The experimental results section shows that this fusion algorithm performs a proper fusion under the assumptions taken, and the fused depth values are more accurate than the input depths.

Finally, the densification thread we propose initializes new map areas using depth estimations from recent keyframes. This happens in two cases: Points not having a projection in the previous dense local map, and points not holding the constraint on the distance threshold $\epsilon$. In the first case, the densification thread triangulates and adds a new point to the map. In the second case, if the current point $\mathbf{x}_{\text{current}}$ is closer to the camera than the existing point $\mathbf{x}_{\text{previous}}$, this is an indication of an occlusion and also a new point is added to the local dense map.

### *3.5. Map Refinement thread*

Every time the S-PTAM mapping thread refines the pose of a keyframe, the local dense map should also be updated. This helps to maintain a more accurate 3D dense reconstruction. Let $\mathcal{K}_b$ and $\mathcal{K}_a$ be the keyframe poses before and after the mapping update, respectively, and $\mathbf{E}^{\mathcal{K}_b W}$ and $\mathbf{E}^{\mathcal{K}_a W}$ the $\mathbf{SE(3)}$ matrices transforming points from the world coordinate system $W$ to both keyframes' reference frames. Then, the thread

updates the point cloud of keyframe $\mathcal{K}$ as follows

$$\mathbf{E}_{ref} = \mathbf{E}^{W\mathcal{K}_a}\mathbf{E}^{\mathcal{K}_b W}, \tag{9}$$

where $\mathbf{E}^{W\mathcal{K}_a}$ is the inverse of $\mathbf{E}^{\mathcal{K}_a W}$.

This refinement thread permanently moves point clouds from distant keyframes' to swap memory, allowing the system to run for long distances and reconstructing up to ten million points. It also keeps the point cloud of local keyframes in RAM memory, until they leave the local environment. The *Disparity Computation* and *Map Fusion and Expansion* threads are *CPU-bound* (intensive use of the CPU) and the *Map Refinement* thread is *I/O-bound* (limited by input-output operations).

## 4. Experiments

We evaluated our *Dense S-PTAM* on the Tsukuba[24] and KITTI[13] public datasets. Tsukuba is a synthetic dataset, with a stereo camera moving around a rendered room for over a minute. The camera motion is fast and contains several loops and pure rotations. The dataset contains 1800 pairs at 30 frames per second. The stereo baseline is 10 cm and the resolution $640 \times 480$ pixels.

KITTI is a standard benchmark for visual odometry and SLAM systems in urban scenes, composed by a set of 23 real sequences of a car driving on urban environments. A forward-looking stereo camera mounted on the vehicle acquires the images. The camera resolution is $1226 \times 370$ and captures images at a frame rate of 10 frames per second. The stereo baseline is 60 cm.

Our experiments were run on a desktop Intel(R) Octa-Core(TM) i7-7700HQ (2.80GHz) with 8GB RAM, using ROS (Kinetic).

### *4.1. Ground truth depth, baseline and metrics*

We evaluated the reconstruction accuracy of our system by comparing our depth maps against the ground-truth depth in both datasets.

In the Tsukuba case, the authors provide the ground-truth disparity maps for each stereo frame, and from that we extracted the depth maps. For the KITTI dataset, we extracted a ground-truth depth map for each keyframe by projecting the Velodyne point clouds on the stereo reference frame, using the calibration parameters provided by the authors.

We use the depth maps extracted by LIBELAS[14] in a single stereo pair as a baseline, in order to show the improvement of our fusion algorithm.

Our locally dense maps are estimated by applying *Dense S-PTAM* to 30 overlapping keyframes, forwards and backwards with respect to the current keyframe, and referred to the left camera.

### *4.2. Results*

Fig. 3 and Fig. 4 show several tridimensional reconstructions estimated by *Dense S-PTAM*, in the KITTI and Tsukuba datasets respectively, with illustrative purposes. Notice the high accuracy of the estimated reconstructions. These results can be better appreciated in the videos accompanying the paper. The first video[4] contains results of *Dense S-PTAM* on sequence 06 of the KITTI dataset. The first part of the video shows the system running in real-time in such sequence. In the second part of the video we show the complete 3D reconstruction estimated by our system, after all the sequence was processed. The second video[5] shows a detailed view of the dense point clouds obtained by Dense S-PTAM in the KITTI sequences 00, 03, 04 and 07; for further illustration on the accuracy that can be achieved by our algorithm.

[4] Video_1: `https://youtu.be/xZSscfjzV90`
[5] Video_2: `https://youtu.be/yPAoFu_LhhA`

Fig. 3: Sample 3D reconstructions using Dense S-PTAM (KITTI dataset, sequence 06).

Fig. 4: Sample 3D reconstructions using Dense S-PTAM (Tsukuba dataset)

Fig. 5 shows a comparison between the LIBELAS depth, estimated from a single stereo frame, and the *Dense S-PTAM* depth, fused over several keyframes. The figure shows the ground truth depth for a specific frame, the LIBELAS and *Dense S-PTAM* depths and their errors.

Notice that the depth obtained by *Dense S-PTAM* (Fig. 5e) is closer to the ground truth (Fig. 5b) than the one from LIBELAS (Fig. 5c). This can also be appreciated in the error figures 5d and 5f. This noticeable improvement in the accuracy is mainly the result of the depth fusion from different viewpoints. The KITTI images contain distant areas, for which the stereo error is large. For such areas the error can be reduced if the stereo depth is fused with the depth of a closer view.

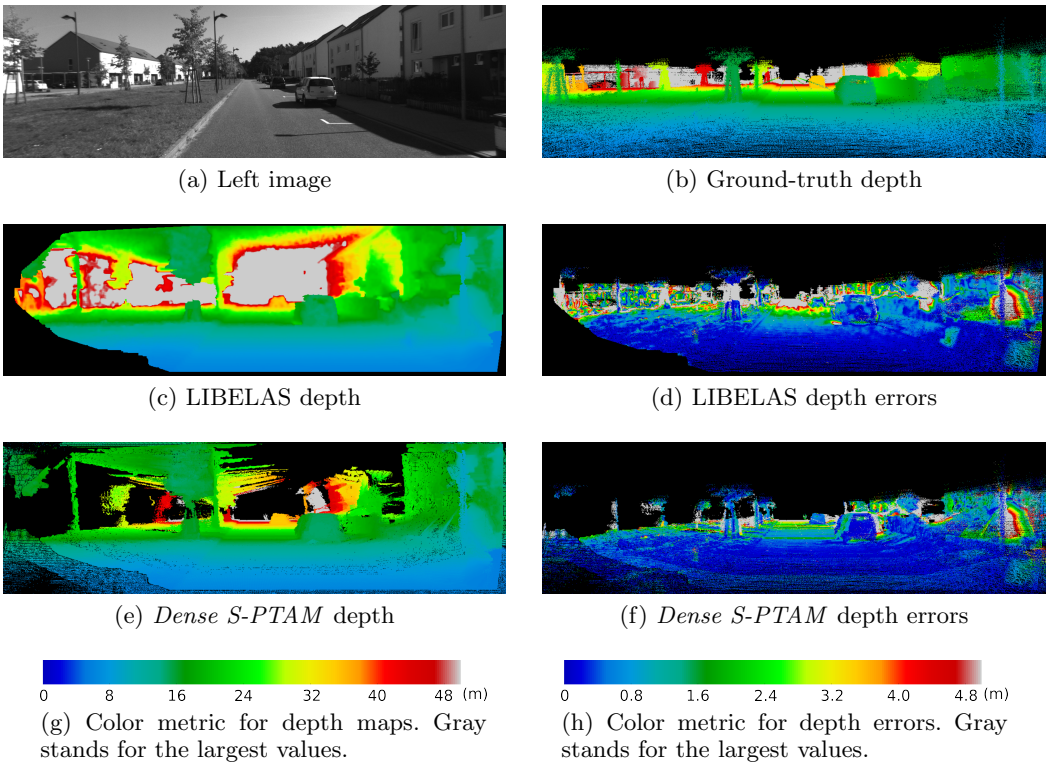
Observe in Fig. 6 and Fig. 7 that LIBELAS and *Dense S-PTAM* errors are more similar for the Tsukuba dataset. The reason is the lack of distant areas in the indoor scene rendered for the Tsukuba dataset. The depth from a single stereo pair is here highly accurate, and the gain obtained by the fusion with another viewpoint is not so evident.

For a more quantitative analysis of our algorithm, we present a detailed analysis of the depth error for the whole KITTI sequence 06. Fig. 8 and Fig. 9 show the error distribution (in the vertical axis, as box-and-whiskers diagrams) for each ground truth depth (in the horizontal axis), and for both LIBELAS and *Dense S-PTAM* respectively. The errors start at 5-meters depth, as this is the minimum distance of the Velodyne sensor we use as ground truth.

The figure shows that *Dense S-PTAM* obtains lower median errors than the LIBELAS estimation. The growth of the median depth error with the depth is smaller for *Dense S-PTAM*, due to the fusion from different viewpoints. The extent of the error distribution is, however, similar for both cases. The main reason is the occlusions, that are not addressed by our algorithm. In any case, both aspects affect a small percentage of the pixels, and hence the median should not be distorted.

For a better evaluation, Fig. 10 compares the *Dense S-PTAM* and LIBELAS median errors for every depth. Notice that the errors are similar for small depths. As the parallax is high the estimated depth is already accurate and extra stereo views do not add much information. The gain in the stereo fusion is appreciated at large depths, where a single stereo pair produces noisy results and multi-view fusion is able to reduce the error.

Fig. 11 and Fig. 12 show the depth errors of LIBELAS and *Dense S-PTAM*, respectively, for the Tsukuba sequence. As before, we also show the median depth error




(a) Left image

(b) Ground-truth depth

(c) LIBELAS depth

(d) LIBELAS depth errors

(e) *Dense S-PTAM* depth

(f) *Dense S-PTAM* depth errors

(g) Color metric for depth maps. Gray stands for the largest values.

(h) Color metric for depth errors. Gray stands for the largest values.

Fig. 5: Comparison of LIBELAS and *Dense S-PTAM* depth maps against the ground truth, for a single frame (KITTI dataset).

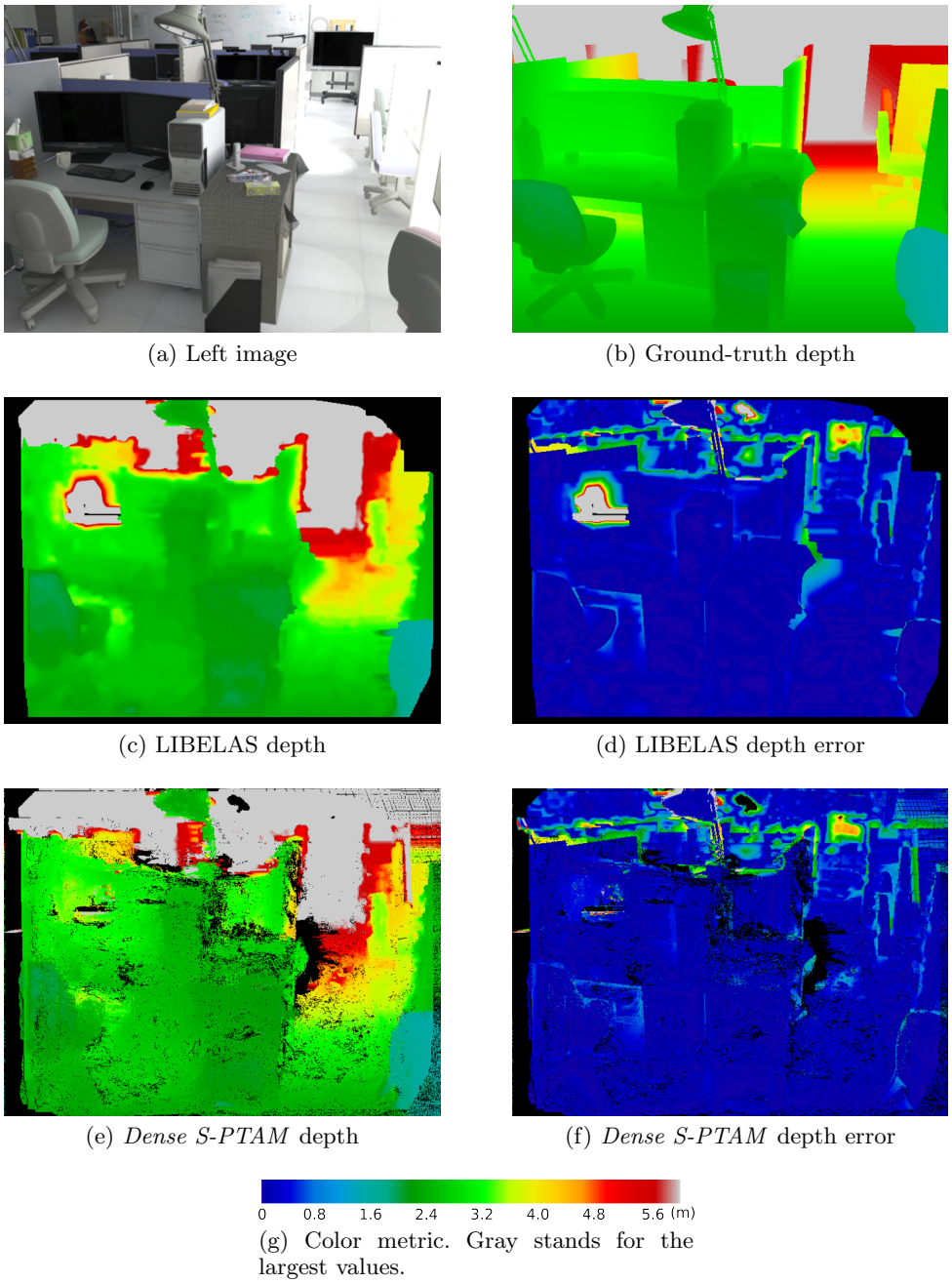
for this approaches (Fig. 13). In contrast to the results for the KITTI sequence, in this case the accuracy of *Dense S-PTAM* is only slightly better than the LIBELAS case.

The reason for that was already discussed: The depth of the rendered indoor scenario in the Tsukuba dataset is small (compared to the outdoor streets of KITTI). The baseline of its stereo sensor is big enough to produce accurate depth estimations, and hence the depth fusion with other stereo pairs do not offer a significant gain. As a conclusion, depth from multiple stereo pairs significantly improves the reconstruction accuracy for high depth-baseline ratios. And the improvement is very limited in the opposite case.

Fig. 14 and Fig. 15 show the number of triangulated and fused points by *Dense S-PTAM* for the KITTI and Tsukuba datasets. We denote as *hypotheses* the points triangulated from a single stereo pair, that become *validated* points when our method fuses their depth with another view. The figure also shows the total number of depth fusions that our algorithm performed for the whole sequence.

The values in the figure demonstrate how *Dense S-PTAM* reduces the map size compared with the naïve approach of registering the stereo point clouds without fusion. Notice how the number of points is higher for the KITTI sequence, as the camera runs for a larger distance than in the Tsukuba sequence. The number of depth fusions results higher in Tsukuba, however, as the scene is revisited multiple times (the trajectory in the KITTI sequence is purely exploratory).

Finally, in Fig. 16 and Fig. 17 we show the localization accuracy of *Dense S-PTAM* for the KITTI and Tsukuba datasets. Specifically, we plot the trajectory estimated by our system compared against the ground-truth; and the relative translation and rotational errors. Notice that these errors are small, comparable to the ones reported by state-of-the-art systems. Observe that the *Dense S-PTAM* error at the final frames of Tsukuba is large. This is because, in this sequence, the camera points to a closed door that covers almost the entire image, and suddenly the door opens.




(a) Left image

(b) Ground-truth depth

(c) LIBELAS depth

(d) LIBELAS depth error

(e) *Dense S-PTAM* depth

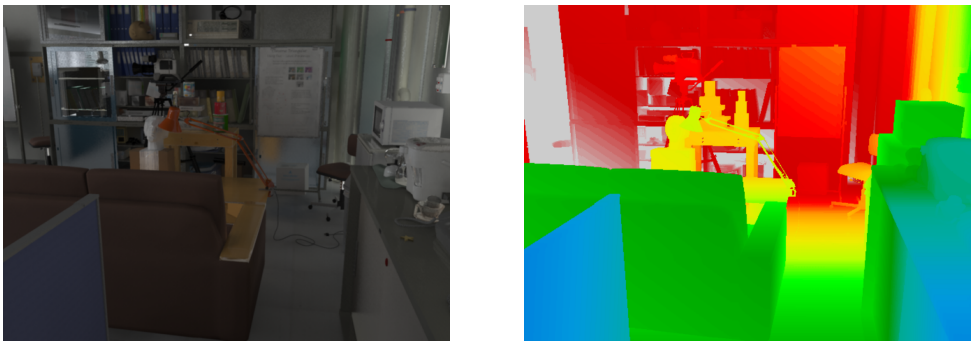
(f) *Dense S-PTAM* depth error

(g) Color metric. Gray stands for the largest values.

Fig. 6: Comparison of LIBELAS and *Dense S-PTAM* depth maps against the ground truth, for a single frame (Tsukuba dataset).

### *4.3. Robustness to Dynamic objects*

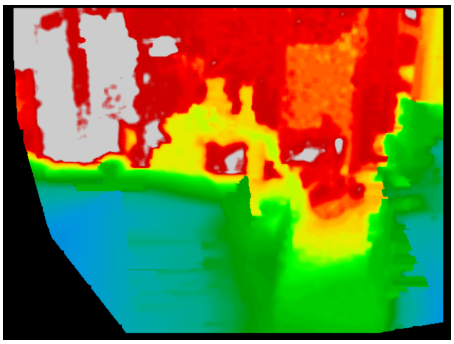
*Dense S-PTAM* is robust to dynamic objects, i.e., they are not included in the dense reconstruction. This can be observed for example in the KITTI sequence 04 shown in the second of our videos (Video_2). The reader can observe a car moving in front of the sensorized KITTI vehicle. Despite the car is moving and appears in the image for a long period of time, it is filtered out of the reconstruction by our system. This behavior is a consequence of the matching rules defined in section 3.4. Points are fused only if the distances between their individual 3D estimations are below a threshold. This does not hold for points in dynamic objects, like the ones in the car, that are not consistent along



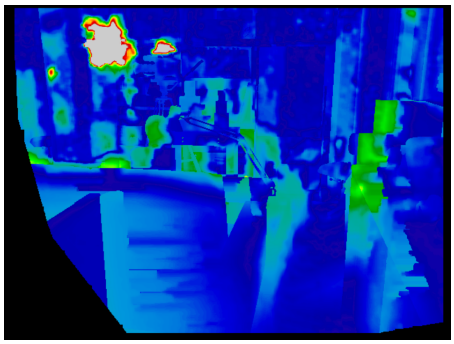
(a) Left image



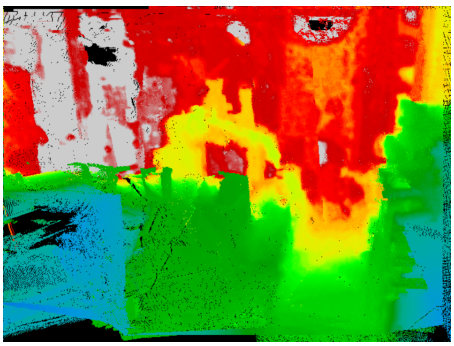
(b) Ground-truth depth



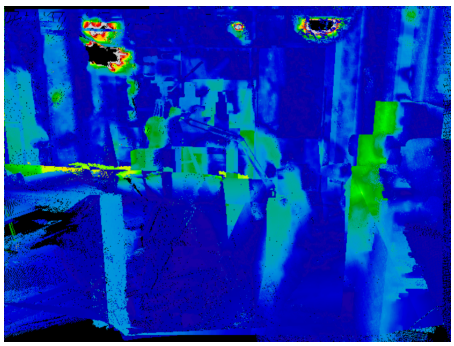
(c) LIBELAS depth



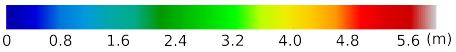
(d) LIBELAS depth error



(e) *Dense S-PTAM* depth



(f) *Dense S-PTAM* depth error




(g) Color metric. Gray stands for the largest values.

Fig. 7: Comparison of LIBELAS and *Dense S-PTAM* depth maps against the ground truth, for a single frame (Tsukuba dataset).

consecutive keyframes. When this happens we label those points as outliers and remove them.

### *4.4. Computational Cost and Memory Requirements*

Fig. 18 shows the computational time statistics of the most relevant steps of our algorithm —disparity computation, map fusion and expansion, and map refinement— per keyframe. The figure shows that *Dense S-PTAM* is suitable for robot navigation, as it can run in real time at 3 fps for the KITTI images and at 5 fps in the Tsukuba case. The difference between both datasets is due to their different image resolution, which causes

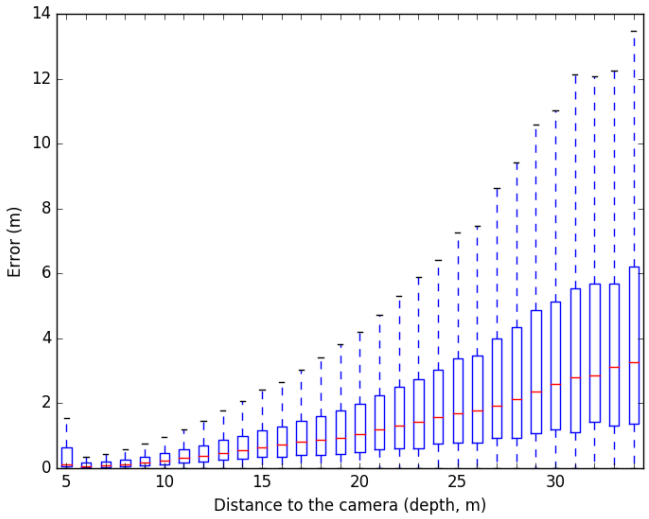



Fig. 8: LIBELAS depth errors per depth, KITTI sequence 06.

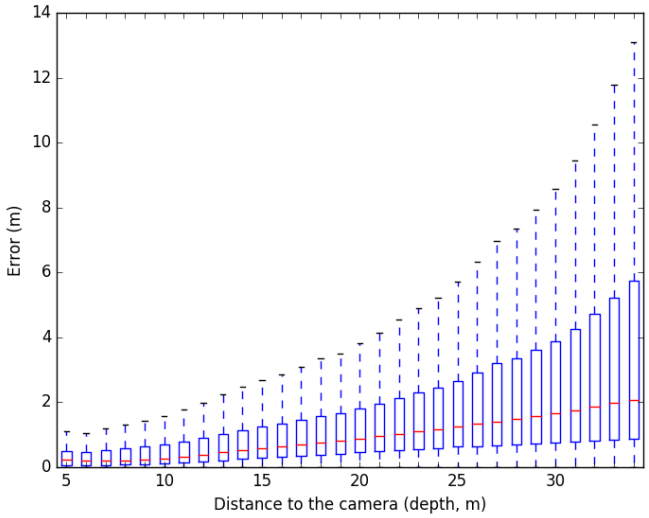



Fig. 9: *Dense S-PTAM* depth errors per depth, KITTI sequence 06.

the disparity map computation done by LIBELAS to double for the KITTI images. Notice, however, that the computation time of the map fusion step is approximately equal in both cases. The map refinement computation is already negligible compared with the other two; and the computation time for the rest of the algorithm is even lower. Our computer has a multi-core processor and our implementation uses several of the cores, with an average processor use around 40%.

The average RAM memory used in our experiments is around 800 MB in the Tsukuba dataset and 2,5 GB in the KITTI dataset. The growth of the memory requirement is approximately linear at exploration; and it is constant when revisiting. We implemented a module to save the farthest parts of the point cloud in the hard drive when it exceeds a limit. The above numbers correspond to such limit in our experiments, that can be set differently.

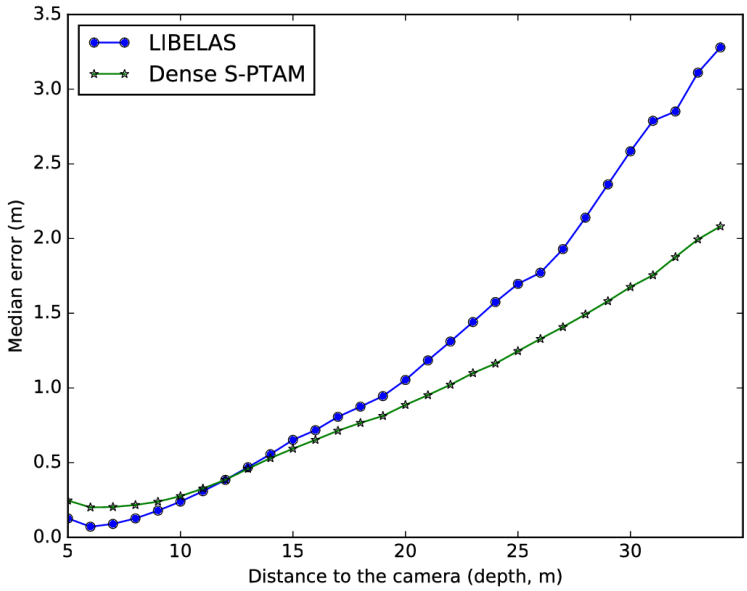



Fig. 10: Median depth errors per depth, LIBELAS and *Dense S-PTAM*, KITTI sequence 06.

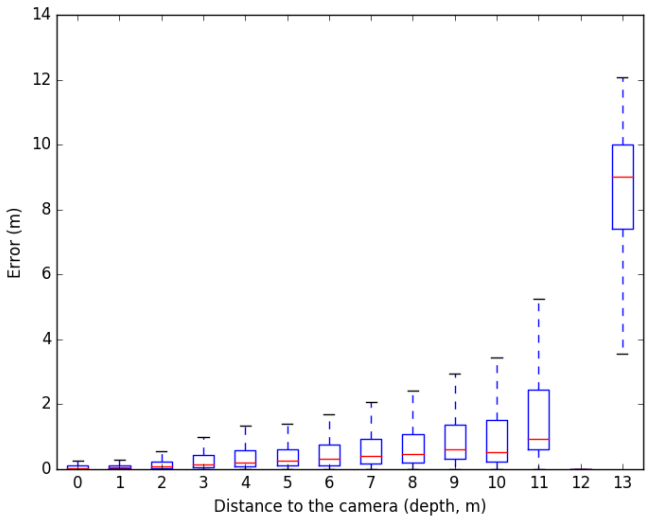



Fig. 11: LIBELAS depth errors per depth in the Tsukuba sequence.

## 5. Conclusions and Future Work

We presented an efficient stereo-based densification method for SLAM systems, that we called *Dense S-PTAM*, capable of generating a locally dense map in CPU real time. We have built a ROS implementation coupled with the stereo SLAM system S-PTAM, proposed in,[26] and have released the code as open source.

We have evaluated our method in a simulated indoor environment (Tsukuba[24]), and in a standard dataset imaging urban scenes (KITTI[13]). Our experiments show that the depth error of our system is lower than the depth error from a single stereo pair, demonstrating an effective fusion of several registered stereo maps. The low computational requirements (we demonstrated three frames per second with standard hardware) make our method suitable for robot navigation.

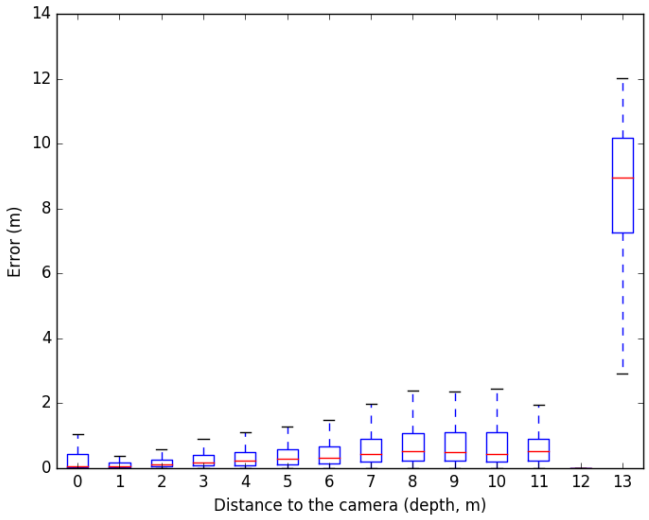



Fig. 12: *Dense S-PTAM* depth errors per depth, Tsukuba sequence.

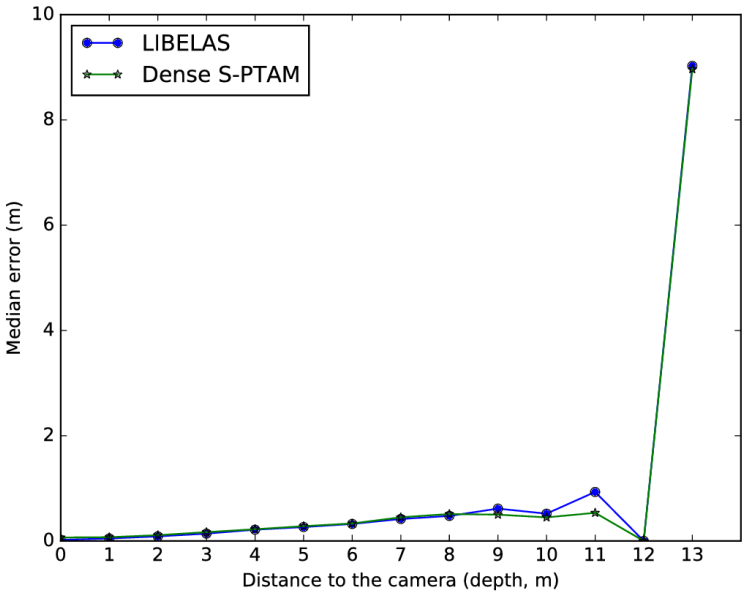



Fig. 13: Median depth errors per depth, LIBELAS and *Dense S-PTAM*, Tsukuba sequence.

Our future work includes the system implementation on an actual robot, in order to develop and evaluate vision-based navigation methods. On the algorithmic side, we plan to improve the reconstruction accuracy using appearance information. Thanks to the loop closure capabilities of S-PTAM, recently added on,[25] a short-term plan is to adapt our dense module to work correctly and consistently with loop closure adjustments.

Finally, we also aim to keep building on top of this system to improve its accuracy at a low computational cost. For example, one of our research lines is clustering the points into higher order structures (e.g., planes).

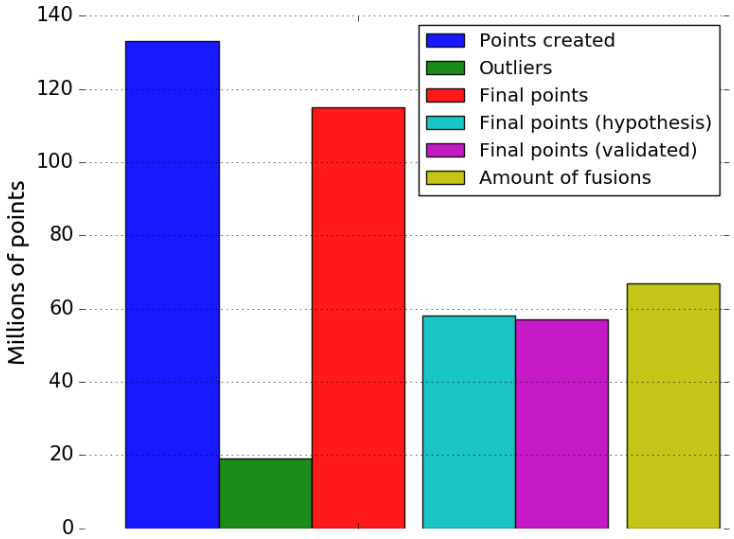



Fig. 14: Total number of points in the *Dense S-PTAM* reconstruction (generated, discarded, present on final reconstruction—hypotheses (saw one time) and validated (fused multiple times)—, and number of fusions) on KITTI 06 sequence.

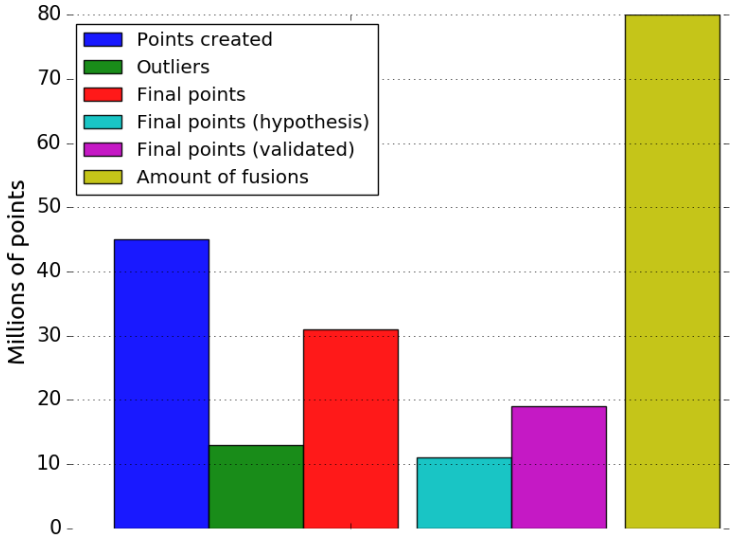
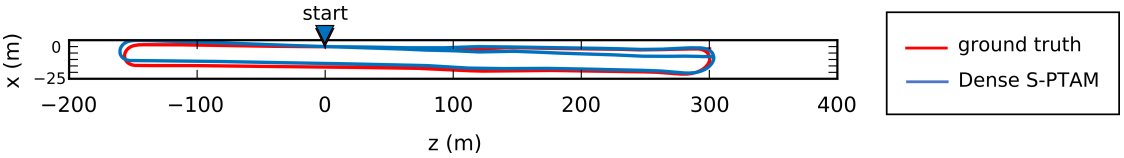
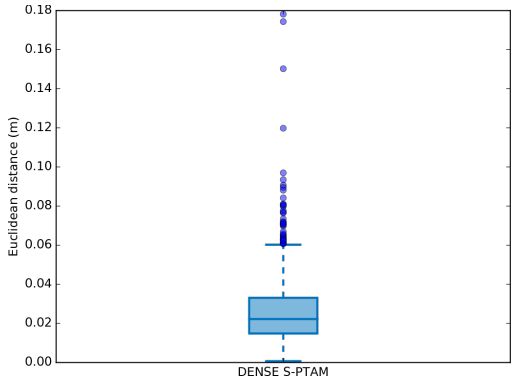


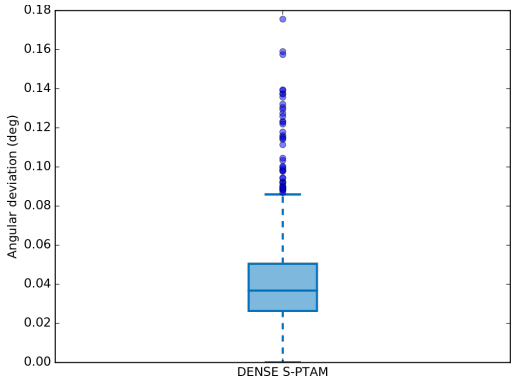

Fig. 15: Total number of points in the *Dense S-PTAM* reconstruction (generated, discarded, present on final reconstruction—hypotheses (saw one time) and validated (fused multiple times)—, and number of fusions) on Tsukuba sequence.




(a) Trajectory

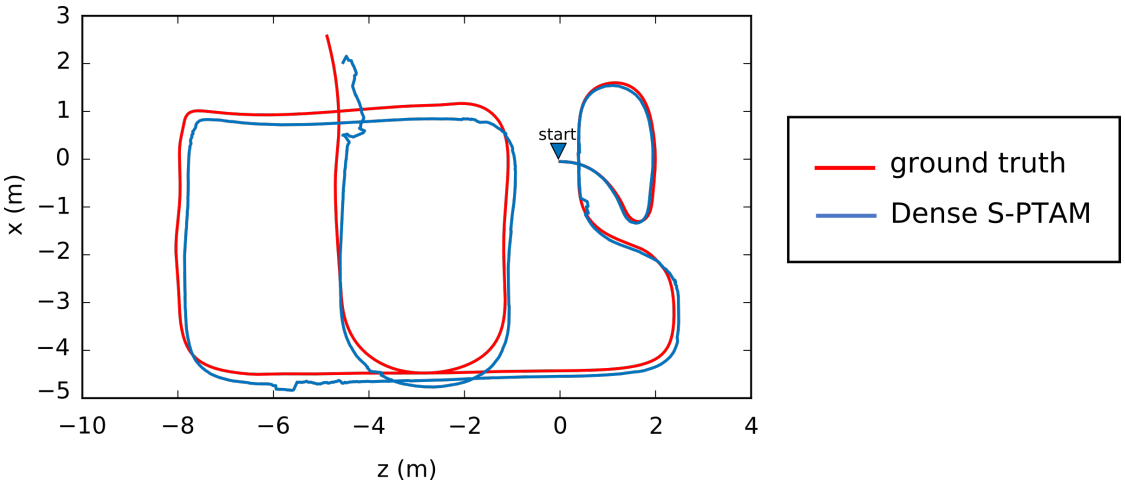



(b) Relative Translation Error

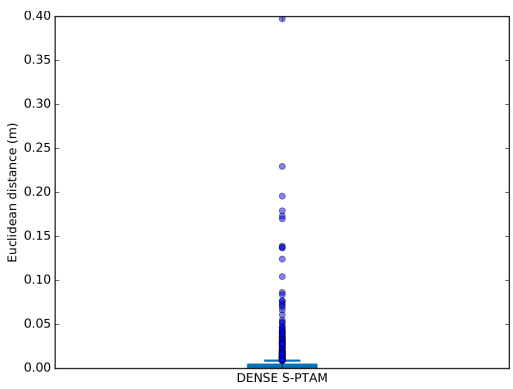



(c) Relative Rotation Error

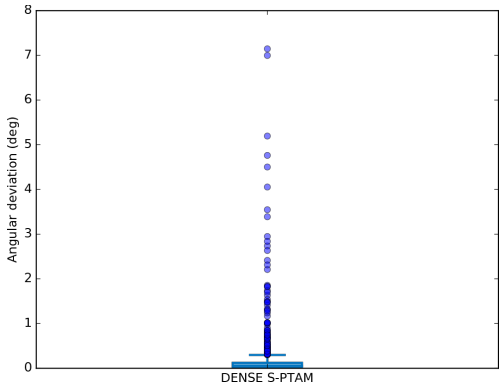
Fig. 16: Accuracy of *Dense S-PTAM* on the sequence 06 of the KITTI dataset. Boxes represent interquartile range ($IQR$), whiskers reach to $-1.5 \times IQR$ and $1.5 \times IQR$, and the points represent data beyond those ranges, considered outliers. The line inside the box represents the median.




(a) Trajectory

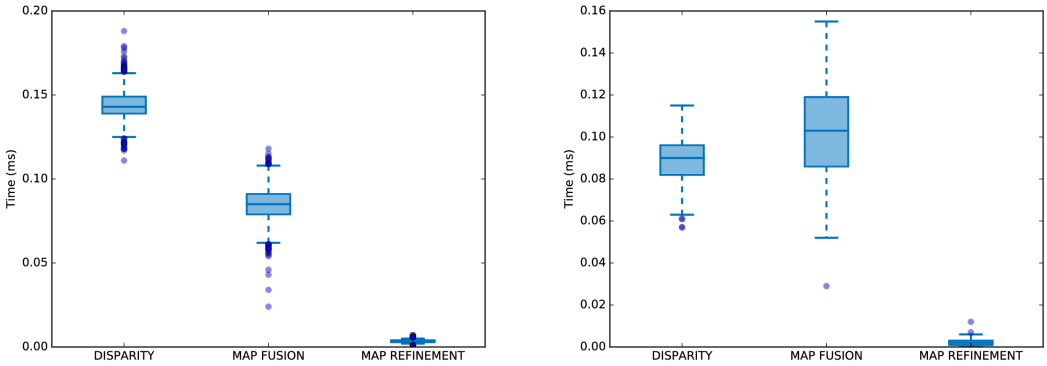



(b) Relative Translation Error




(c) Relative Rotation Error

Fig. 17: Accuracy of *Dense S-PTAM* on the Tsukuba dataset. Boxes represent interquartile range ($IQR$), whiskers reach to $-1.5 \times IQR$ and $1.5 \times IQR$, and the points represent data beyond those ranges, considered outliers. The line inside the box represents the median.




(a) KITTI

(b) Tsukuba

Fig. 18: Computation time boxplots for the main steps of our algorithm. Boxes represent interquartile range ($IQR$), whiskers reach to $-1.5 \times IQR$ and $1.5 \times IQR$, and the points represent data beyond those ranges, considered outliers. The line inside the box represents the median.

## References


1. Pablo Alcantarilla, Chris Beall, and Frank Dellaert. Large-scale dense 3D reconstruction from stereo imagery. In *5th Workshop on Planning, Perception and Navigation for Intelligent Vehicles (PPNV)*. Georgia Institute of Technology, November 2013.
2. Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (SLAM): part II. *IEEE Robotics Automation Magazine*, 13(3):108–117, September 2006.
3. Sid Yingze Bao, Manmohan Chandraker, Yuanqing Lin, and Silvio Savarese. Dense Object Reconstruction with Semantic Priors. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 1264–1271, Washington, DC, USA, 2013. IEEE Computer Society.
4. Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J. Leonard. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Transactions on Robotics*, 32(6):1309–1332, December 2016.
5. David M Cole and Paul M Newman. Using laser range data for 3D SLAM in outdoor environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1556–1563, May 2006.
6. Alejo Concha, Wajahat Hussain, Luis Montano, and Javier Civera. Manhattan and Piecewise-Planar Constraints for Dense Monocular Mapping. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.
7. Alejo Concha, Giuseppe Loianno, Vijay Kumar, and Javier Civera. Visual-inertial direct SLAM. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1331–1338, May 2016.
8. Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, June 2007.
9. Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part I. *IEEE Robotics Automation Magazine*, 13(2):99–110, June 2006.
10. Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.
11. Jakob Engel, Jörg Stückler, and Daniel Cremers. Large-scale direct slam with stereo cameras. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942. IEEE, 2015.
12. A. Geiger, J. Ziegler, and C. Stiller. StereoScan: Dense 3D reconstruction in real-time. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 963–968, June 2011.
13. Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision Meets Robotics: The KITTI Dataset. *The International Journal of Robotics Research, IJRR*, 32(11):1231–1237, September 2013.
14. Andreas Geiger, Martin Roser, and Raquel Urtasun. *Efficient Large-Scale Stereo Matching*, pages 25–38. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
15. Gottfried Graber, Thomas Pock, and Horst Bischof. Online 3D reconstruction using convex optimization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 708–711, November 2011.
16. Matthew Klingensmith, Ivan Dryanovski, Siddhartha Srinivasa, and Jizhong Xiao. Chisel: Real Time Large Scale 3D Reconstruction Onboard a Mobile Device using Spatially Hashed Signed Distance Fields. In *Proceedings of Robotics: Science and Systems*, volume 4, Rome, Italy, July 2015.
17. Georg Kuschk, Aljaž Božič, and Daniel Cremers. Real-time variational stereo reconstruction with applications to large-scale dense SLAM. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1348–1355, June 2017.
18. Lubor Ladický, Paul Sturgess, Chris Russell, Sunando Sengupta, Yalin Bastanlar, William Clocksin, and Philip Torr. Joint Optimization for Object Class Segmentation and Dense Stereo Reconstruction. *International Journal of Computer Vision*, 100(2):122–133, 2012.
19. Will Maddern and Paul Newman. Real-time probabilistic fusion of sparse 3D LIDAR and dense stereo. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2181–2188. IEEE, 2016.
20. Ondrej Miksik, Yousef Amar, Vibhav Vineet, Patrick Prez, and Philip Torr. Incremental dense multi-modal 3D scene reconstruction. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 908–915, Sept 2015.
21. Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, October 2017.
22. Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. DTAM: Dense Tracking and Mapping in Real-time. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, ICCV '11, pages 2320–2327, Washington, DC, USA, 2011. IEEE Computer Society.
23. H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto. Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning. In *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1366–1373, September 2017.

24. M. Peris, S. Martull, A. Maki, Y. Ohkawa, and K. Fukui. Towards a simulation driven stereo vision system. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR)*, pages 1038–1042, November 2012.
25. Taihú Pire, Thomas Fischer, Gastón Castro, Pablo De Cristóforis, Javier Civera, and Julio Jacobo Berlles. S-PTAM: Stereo Parallel Tracking and Mapping. *Robotics and Autonomous Systems (RAS)*, 93:27 – 42, 2017.
26. Taihú Pire, Thomas Fischer, Javier Civera, Pablo De Cristóforis, and Julio Jacobo Berlles. Stereo parallel tracking and mapping for robot localization. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1373–1378, September 2015.
27. Matia Pizzoli, Christian Forster, and Davide Scaramuzza. REMODE: Probabilistic, monocular dense reconstruction in real time. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2609–2616. IEEE, 2014.
28. Thomas Schöps, Torsten Sattler, Christian Häne, and Marc Pollefeys. Large-scale Outdoor 3D Reconstruction on a Mobile Device. *Computer Vision and Image Understanding*, 157(C):151–166, April 2017.
29. Sunando Sengupta, Eric Greveson, Ali Shahrokni, and Philip Torr. Urban 3D semantic modelling using stereo vision. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 580–585, May 2013.
30. Jan Stühmer, Stefan Gumhold, and Daniel Cremers. *Real-Time Dense Geometry from a Handheld Camera*, pages 11–20. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
31. Michael Tanner, Pedro Pinies, Lina Maria Paz, and Paul Newman. DENSER Cities: A System for Dense Efficient Reconstructions of Cities. *arXiv preprint arXiv:1604.03734*, 2016.
32. Beau Tippetts, Dah Jye Lee, Kirt Lillywhite, and James Archibald. Review of stereo vision algorithms and their suitability for resource-limited systems. *Journal of Real-Time Image Processing*, 11(1):5–25, January 2016.
33. V. Vineet, O. Miksik, M. Lidegaard, M. Nießner, S. Golodetz, V. A. Prisacariu, O. Kähler, D. W. Murray, S. Izadi, P. Pérez, and P. H. S. Torr. Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction. In *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 75–82, May 2015.
34. Rui Wang, Martin Schwörer, and Daniel Cremers. Stereo DSO: Large-scale direct sparse visual odometry with stereo cameras. *arXiv preprint arXiv:1708.07878*, 2017.