Stereo-based Visual Odometry: A Comparative Study*

Anand Uday Gokhale[†1], Sumanth R Hegde[†1], and Sourav Sahoo[†1]

[1]Department of Electrical Engineering, Indian Institute of Technology Madras
{ee17b158,ee17b032,ee17b040}@smail.iitm.ac.in

Abstract

In this work, we conduct a comparative study on three existing stereo visual odometry systems: 1) real-time stereo visual odometery (RT-SVO), 2) stereo parallel tracking and mapping (S-PTAM) and 3) stereo direct sparse odometry (S-DSO). The RT-SVO algorithm differs from the traditional visual odometry algorithms as it does not make any assumptions on camera motion and it operates on dense stereo disparity images computed by a stereo algorithm. S-PTAM follows a parallel-tracking-and-mapping strategy : a tracking thread estimates the camera pose at frame rate while a mapping thread updates a keyframe-based map at a slower speed. S-DSO differs significantly from the above two methods as it is a direct method, operating on the pixel intensities directly and minimizes a photometric error. We conduct extensive experiments for all three algorithms on KITTI Visual Odometry Benchmark to evaluate their performances. The codes for all the experiments in this work is made available at https://github.com/AnandGokhale/CS6790_Project.

1 Introduction

Traditional robotic systems have depended on LASERs and LiDARs for active sensing of their environment and perform self localization and mapping (Castellanos et al., 1999). Several autonomous ground vehicles have also relied on combination of wheel odometry and inertial sensing for this purpose. However, both of these approaches have their limitations. The inertial sensing methods are prone to drift and wheel odometry fails in rough and uneven terrain (Howard, 2008). LASER and LiDAR based systems are accurate but they are highly expensive due to which their practical uses are limited. So, there has been a shift towards camera-based simultaneous localization and mapping (SLAM) methods in the recent past. Compared to the traditional methods, visual odometry techniques provide multiple advantages: higher resolution, higher sensing range, smaller size and lower cost. Hence, visual SLAM methods have been extensively studied in the recent times.

We first focus on a real-time stereo visual odometry (RT-SVO) system (Howard, 2008). RT-SVO is well suited for autonomous ground vehicles such as small robots. Stereo cameras are used instead of monocular cameras in this algorithm to avoid scale drift issues with

*Course project report for CS6790 Geometry and Photometry based Computer Vision, Indian Institute of Technology Madras, Spring 2020.

†Authors are mentioned alphabetically.

monocular SLAM. The algorithm does not require any initial motion estimate and hence can handle large image translations. RT-SVO also uses *non-invariant* feature detectors instead of invariant ones such as SIFT (Lowe, 2004) and SURF (Bay et al., 2006) as they are about an order of magnitude faster to compute. Hence, the algorithm can fail in case the agent undergoes large motions around the optical axis, which we discuss in greater detail in Section 3. This drawback is of little consequence for the intended application of the algorithm which was for ground vehicles that undergo very small movements around the optical axis and move at low speeds.

The second algorithm that we take up is the stereo parallel-tracking-and-mapping (S-PTAM) (Pire et al., 2017). It is essentially an extension of the existing parallel tracking and mapping (PTAM) (Klein and Murray, 2007) algorithm for stereo domain. S-PTAM consists of two different threads for camera tracking and map optimization. The tracking thread matches features, creates new map points and estimates the pose for each frame whereas the mapping thread refines the nearby point landmarks that compose the map. In this work, as we focus primarily on the odometry and thus do not detail the mapping algorithm. The main advantage of S-PTAM algorithm is parallelizing the computation across multiple threads to achieve real-time performance. In addition to that, it can also be used simultaneously with wheel odometry to improve accuracy and robustness.

The third algorithm, Stereo Direct Sparse Odometry (S-DSO) (Wang et al., 2017) differs from the above mentioned algorithms significantly. Unlike RT-SVO and S-PTAM that use keypoints, S-DSO is a direct method that computes geometry and motion parameters directly from the image intensities, bypassing the need of an intermediate keypoint selection step. S-DSO minimizes the error arising from both temporal multi-view stereo and static stereo to achieve better performance. There are several other advantages of combining both of them which is discussed in Section 2.3.

Our **contributions** include a complete implementation of the RT-SVO algorithm in python. We use an open-source python-based implementation of S-PTAM for our experiments. Since the official S-DSO code has not yet been released, we make use of an open-source implementation available here. Since we found no obvious shortcomings in either implementations, we consider the performance of each to be representative of the true performance of each method. More details are outlined in Section 3.

The rest of this report is organized as follows: The technical details of the three algorithms are briefly discussed in Section 2. We describe the dataset and the experiments we conducted in Section 3. We discuss the results in Section 4 and finally conclude in Section 5.

2 Methodology

We briefly discuss the technical details of the algorithms in this section. For the sake of brevity, we do not discuss all the three methods in their entirety. The full algorithm in RT-SVO would be presented while we will only skim over the methodology of S-PTAM and S-DSO.

2.1 RT-SVO

The RT-SVO algorithm differs from sliding window/batch techniques used in common Structure from Motion (SfM) or SLAM algorithms. The basic algorithm is as follows : (1) obtain feature points for the pre-filtered and rectified stereo pairs (2) match features between frames (3) find the largest set of self-consistent matches (inliers) and (4) find the frame to frame

motion by minimizing a re-projection error for features in the inlier set. A key feature of the algorithm is the inlier detection step used to filter out good matches. A simple stereo rigidity constraint is used in the inlier detection step, following Hirschmuller et al. (2002), as opposed to the commonly used RANSAC algorithm (Fischler and Bolles, 1981). The algorithm is detailed below :

2.1.1 Preprocessing

The basic preprocessing steps required are : (1) rectification (2) pre-filtering : smoothing the rectified stereo pair and (3) disparity : The disparity image is obtained using block matching (based on sum-of-absolute differences/ SAD) performed on the pre-filtered image. Let the pre-filtered and disparity images be denoted by J_a and D_a, with a indexing the frame, captured at some time t_a. Let J_b and D_b denote the corresponding images at time t_b.

2.1.2 Feature detection

In this step, an efficient feature detector such as FAST (Rosten and Drummond, 2006) or Harris (Harris et al., 1988) is used to detect salient corners. Using the generated disparity maps, we then find the world co-ordinates for these points. Since the disparity map is bound to have "holes" where the simple block-matching algorithm has failed, corner points for which the disparity values are unknown are discarded.

2.1.3 Feature Matching

Howard (2008) proposes to construct a descriptor using the pixel intensities in a 7×7 window around the pixel. These are then matched by computing all pairwise distances (using SAD scores) between feature vectors in J_a and J_b. In our implementation, we chose to perform matching by searching over image pyramids, to account for large motion. Let f_a and f_b denote a match between features in frame a and b and M denote the set of all matches.

2.1.4 Find consistent inliers

Given the set of matches M, a consistency matrix W is calculated by considering all the matches pair-wise and checking a simply rigidity constraint. A pair of matches is deemed consistent if the distance between two features (in world co-ordinates) in frame a is identical to the distance between the two features in frame b. If the matched features (f_a, f_b) has world coordinates (w_a, w_b) and another pair ($f_a^{'}$, $f_b^{'}$) has co-ordinates ($w_a^{'}$, w'_b), then the pair is consistent if

$$\text{abs}(|w_a - w_a^{'}| - |w_b - w'_b|) < \delta$$

where δ is a chosen tolerance. If a pair of matches is consistent, then the corresponding entry in the consistency matrix W is 1, and 0 otherwise. Using W, the largest set of consistent inliers boils down to finding the maximum clique in a graph defined by the adjacency matrix W. Since this is an NP-complete problem, the following inexact strategy is adopted:

1. Initialise the clique using the node with maximum degree.
2. Find the matches compatible with the existing matches in the clique.
3. Add the match with the largest number of consistent matches.

4. Repeat step 2 and 3 till there are no more compatible matches.

Let Q denote the set of consistent matches found using this algorithm.

2.1.5 Estimate motion

Using the set of consistent matches Q, the frame-to-frame rotation and translation is estimated by minimizing the re-projection error :

$$\epsilon = \sum_{(f_a, f_b) \in Q} (j_b - P\Delta_{ab} w_b)^2 + (j_a - P\Delta_{ab}^{-1} w_a)^2 \tag{1}$$

where j_a and j_b are image co-ordinates and Δ_{ab} is transformation matrix associated with the relative motion between frame a and b. We minimized (1) using the Levenberg-Marquardt (LM) algorithm (Levenberg, 1944; Marquardt, 1963).

2.1.6 Implementation details

We used `python` to implement RT-SVO for ease of experimentation. Wherever possible, we opted to use optimised C++ implementations for speed. Specifically, we've utilised `opencv` extensively while also resorting to `scipy` for optimisation. We have mainly used the FAST feature detector, but we do make a comparison with GFTT (Shi and Tomasi, 1994) in Section 3. We have also employed a patch-wise filtering of feature points to reduce computation. On average, the per-frame processing speed on the 376×1241 resolution images of KITTI is 2.2s-2.5s on a standard laptop.

2.2 S-PTAM

S-PTAM is a visual SLAM algorithm mainly built upon the approach in Klein and Murray (2007) . As the name suggests, to do parallel tracking and mapping, the method utilizes 2 threads. Another thread is used for loop detection. We shall discuss only the tracking thread in detail here as we mainly focus on visual odometry performance.

Similar to RT-SVO, this method is indirect and sparse. To optimize for time, S-PTAM also assumes that the images are pre-rectified. (In the cases when they are not, this process of rectification is run on a separate thread). Pose is estimated using feature correspondences between the existing 3D map and 2D features of the frame in consideration. The feature detector chosen is GFTT, after considerable evaluation over different choices.

An initial pose estimate is calculated either using a decaying motion model (or wheel odometry, depending on its availability) and each feature point in the image is projected using the predicted stereo pose. A match is searched for in the neighbourhood of this point on the map, and saved. After this, the following cost function is minimized to estimate the new pose, using the LM optimization algorithm:

$$\mu^* = \underset{\mu}{\operatorname{argmin}} \sum_{i \in S} \rho(\mathbf{J}_i \mu - \Delta \mathbf{z}_i(\mu_{prev}))$$

where $\mathbf{J}$ is the Jacobian of the re-projection error, μ is the vector of relative motion parameters (parameterised in the Lie algebra), $\Delta \mathbf{z}(\mu)$ is the re-projection error evaluated for motion parameters μ and S is the set of matched measurements/ features. $\rho(\cdot)$ denotes the Huber loss, opted for robustness against outliers.

To reduce the number of computations, only points from particular keyframes, which satisfy certain criteria in terms of the number of new features detected are added to the map. This initial estimate of the map point is later corrected using bundle adjustment in the mapping thread.

S-PTAM has been proved to be robust in multiple adversarial situations like dynamic objects, low-light conditions, fast camera motions combined with low frame rates. One of the drawbacks of S-PTAM, which also common in a lot of existing SLAM/VO systems, is that abrupt changes in camera motion can cause localization failures, as we show in Section 4. To this end, an improved version of S-PTAM combined with inertial measurement unit (IMU) has been attempted recently (Fischer et al., 2016).

2.3 S-DSO

S-DSO does not involve the feature matching step like S-PTAM and RT-SVO. Instead, when a new stereo frame is fed to the system, direct image alignment (Engel et al., 2013) is performed to track the new frame with respect to the newest keyframe in the active window. Suppose a point set $\mathcal{P}_i$ in a reference frame I_i is observed in I_j, then direct image alignment can be formulated as:

$$E_{ij} = \sum_{\mathbf{p} \in \mathcal{P}_i} \omega_{\mathbf{p}} \left\| I_j[\mathbf{p}'] - I_i[\mathbf{p}] \right\|_\gamma \tag{2}$$

where $\|\cdot\|_\gamma$ is the Huber norm and $\omega_{\mathbf{p}}$ is a weighting that down-weights high image gradients as given in (3).

$$\omega_{\mathbf{p}} = \frac{c^2}{c^2 + \|\nabla I_i(\mathbf{p})\|_2^2} \tag{3}$$

where c is some constant. Stereo coupling is performed to balance the energy function term due to multi-view stereo and static stereo. So, the final energy function is given as:

$$E = \sum_{i \in \mathcal{F}} \sum_{\mathbf{p} \in \mathcal{P}_i} \left(\sum_{j \in \text{obs}(\mathbf{p})} E_{ij}^{\mathbf{p}} + \lambda E_{is}^{\mathbf{p}} \right) \tag{4}$$

where $\mathcal{F}$ is the set of the keyframes in the current window, obs($\mathbf{p}$) is the set of keyframes where $\mathbf{p}$ is observable, $E_{is}^{\mathbf{p}}$ is the energy belonging to static-stereo residuals, and λ is the stereo coupling factor.

There are several advantages of stereo coupling as listed below:

- Absolute scale can be directly calculated from static stereo from the known baseline of the stereo camera.
- Static stereo can provide initial depth estimation for multi-view stereo.
- The two energy terms complement each other in degenerate cases where edges are parallel to the epipolar lines.

The energy function is minimized using Gauss-Newton iterations on the image pyramid in a coarse-to-fine order. In contrast to previous direct monocular VO systems that use random depth values for initialization (Engel et al., 2013, 2014), in S-DSO static stereo matching is done to estimate the dense depth map of the first frame. Unlike S-PTAM, S-DSO does not incorporate loop detection for global pose refinement. However, it's performance on sequences with loops is still comparable to S-PTAM, as we show in Section 4.

3 Experiments

3.1 Dataset

All the experiments are performed on the KITTI Visual Odometry Benchmark (Geiger et al., 2013). It contains a total of 22 driving sequences. The first 11 sequences are provided with ground truth poses as the training set and the rest 11 comprise the testing set. As neither of the models are data-driven, we evaluate them on the training set itself. The training sequence contains $\sim$ 23K frames captured at 10 Hz.

3.2 Experiments

We ran experiments on all 3 algorithms by first using the default configurations. On both sparse methods, we attempted to understand which feature descriptors would work the best, mainly in terms of accuracy. Specifically, in RT-SVO, we experimented with 2 feature detectors, namely GFTT and FAST. The original paper proposed the use of FAST. However, since more recent works such as S-PTAM found GFTT to be optimal, we chose to evaluate GFTT against FAST (Appendix B).

In the case of S-PTAM, we have attempted to test few of the combinations of different feature detectors and descriptors evaluated by the authors. We have only evaluated binary descriptors as they are efficient, low-dimensional alternatives to floating-point based descriptors such as SIFT (Lowe, 2004), making them particularly attractive for visual SLAM. Binary Robust Independent Elementary Features (BRIEF) (Calonder et al., 2010) was one of the first binary descriptors, found to be highly discriminative while also being efficient. This was improved upon by Oriented fast and Rotated BRIEF (ORB) (Rublee et al., 2011), which incorporated rotational invariance. One of the more recent descriptors is Learned Arrangement of Three Patches (LATCH) (Levi and Hassner, 2016), which, contrary to BRIEF and ORB which compute intensity differences over smoothed pixel values, compares pixel *patches*. We've used GFTT as the feature detector for BRIEF and ORB while using ORB's detector itself for the ORB descriptor.

S-DSO being a direct method does not use a feature detector/descriptor. Since S-DSO boasts to have impressive performance even at real-time execution, we ran it in two modes: One where speed was prioritized, and another where accuracy was prioritized. The exact details regarding the 2 modes are given in Appendix A.

3.3 Evaluation Metrics

We evaluate the translational error (in %) and rotational error (in deg/m) for all the sequences in all the different configurations of the VO algorithms according to (5) and (6). It is to be noted that each of the three papers use substantially different metrics for evaluating their methods. However, we use EVO package (Grupp, 2017) and tools provided by the TUM Computer Vision Group[1] for all the algorithms for fair comparison of their performance.

$$\text{Translational Error} = \frac{\text{Mean translational deviation in m}}{\text{Total ground truth path length in m}} \times 100 \tag{5}$$

$$\text{Rotational Error} = \frac{\text{Mean rotational deviation in deg}}{\text{Total ground truth path length in m}} \times 100 \tag{6}$$

[1] https://vision.in.tum.de/data/datasets/rgbd-dataset/tools

4 Results and Discussion

We compare the best variant of each of the tracking algorithms in Table 1. For RT-SVO we take the FAST variant, for S-PTAM, we choose the GFTT-BRIEF variant and for S-DSO, we use the one with `Default` setting. A complete evaluation of the all the variants of all the algorithms is presented in Appendix B. We also plot the trajectories for two sequences in Figure 1 and 2.

Table 1: Comparison of translational and rotational error of the three VO algorithms

	RT-SVO		S-PTAM		S-DSO	
Sequences	**Trans.**	**Rot.**	**Trans.**	**Rot.**	**Trans.**	**Rot.**
00	5.56	0.84	0.18	0.05	**0.16**	**0.03**
01	23.71	1.37	26.36	0.32	**0.57**	**0.02**
02	6.41	1.14	0.31	0.06	**0.20**	**0.03**
03	4.54	1.35	0.32	0.09	**0.32**	**0.05**
04	3.02	1.62	0.32	0.14	**0.20**	**0.05**
05	6.21	0.79	0.16	0.04	**0.16**	**0.03**
06	1.68	0.63	0.36	0.10	**0.33**	**0.10**
07	3.13	1.60	**0.48**	**0.13**	0.60	0.22
08	13.71	1.73	0.23	0.06	**0.21**	**0.04**
09	10.25	1.70	0.49	0.08	**0.39**	**0.06**
10	8.43	2.79	0.31	0.12	**0.21**	**0.06**

where Trans. = translational error and Rot. = rotational error and values in bold represent the best performing algorithm.

From Table 1, we can conclude that S-DSO is superior to the other two algorithms in almost all trajectories. This observation validates the recent move towards direct methods over feature-based methods. For sequence # 01, we notice that the results for RT-SVO and S-PTAM are particularly poor as it records a very fast moving car in a highway with very low feature scenes which makes it ill-conditioned for these algorithms. Direct methods are thus particularly attractive for such environments. RT-SVO performs almost an order of magnitude worse than the other two algorithms. This is because RT-SVO was originally designed for small robots and automated ground vehicles such as rovers and not for fast moving environments such as the ones seen in KITTI Odometry dataset.

5 Conclusion

In this work, we do a comparative study of three visual odometry algorithms: RT-SVO, an algorithm meant for slow-moving ground rovers that performs frame-to-frame motion estimation; S-PTAM, a stereo adaptation of the parallel tracking and mapping methodology with loop closure; and S-DSO, a stereo-based direct and sparse method that samples map points uniformly throughout the image and minimises a photometric error. We described all three methodologies in brief, with particular focus towards RT-SVO. We conducted extensive experiments on the KITTI dataset for evaluating the algorithms and show that the direct method S-DSO outperforms the feature-based methods in almost all situations.

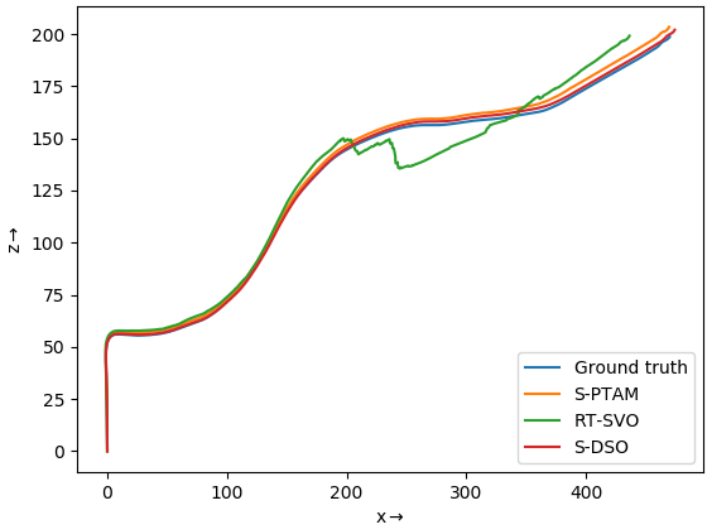

Figure 1: **Trajectory for sequence # 03**. RT-SVO performs almost on par with the other VO algorithms for linear paths and slow-moving camera. Error accumulates with time due to plain frame-to-frame estimation without batch optimization.

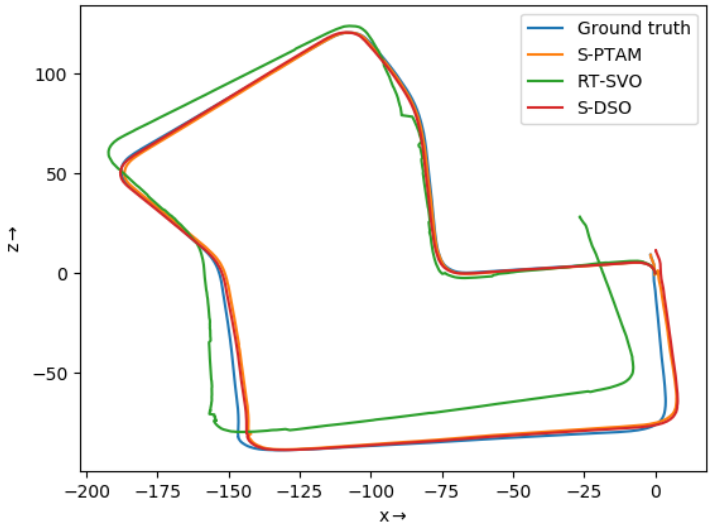

Figure 2: **Trajectory for sequence # 07**. In trajectories with loops, such as the one shown here, RT-SVO fails due to error accumulation and no global pose refinement. Unlike S-PTAM, S-DSO does not have a loop-closure component but still performs reasonably well.

References

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010.

Jose A Castellanos, JMM Montiel, José Neira, and Juan D Tardós. The spmap: A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on robotics and Automation*, 15(5):948–952, 1999.

Jakob Engel, Jurgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1456, 2013.

Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.

Thomas Fischer, Taihú Pire, Petr Čížek, Pablo De Cristóforis, and Jan Faigl. Stereo vision-based localization for hexapod walking robots operating in rough terrains. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2492–2497. IEEE, 2016.

Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

Michael Grupp. evo: Python package for the evaluation of odometry and slam. https://github.com/MichaelGrupp/evo, 2017.

Christopher G Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

Heiko Hirschmuller, Peter R Innocent, and Jonathan M Garibaldi. Fast, unconstrained camera motion estimation from stereo without tracking and robust statistics. In *7th International Conference on Control, Automation, Robotics and Vision, 2002. ICARCV 2002.*, volume 2, pages 1099–1104. IEEE, 2002.

Andrew Howard. Real-time stereo visual odometry for autonomous ground vehicles. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3946–3952. IEEE, 2008.

Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE, 2007.

Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2):164–168, 1944.

Gil Levi and Tal Hassner. Latch: learned arrangements of three patch codes. In *2016 IEEE winter conference on applications of computer vision (WACV)*, pages 1–9. IEEE, 2016.

David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

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*, 93:27–42, 2017.

Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

Jianbo Shi and Carlo Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

Rui Wang, Martin Schworer, and Daniel Cremers. Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3903–3911, 2017.

Appendices

A Stereo DSO Configurations

For S-DSO, evaluations are done in two configurations: `Default` and `Fast`. The differences in the two configurations are listed in Table 2.

Table 2: Differences between `Default` and `Fast` configuration of S-DSO

Attributes	Default	Fast
Real-time enforcing	No	5×
Active Points	2000	800
Active Frames	5-7	4-6
LM iterations/KF	1-6	1-4
Image Resolution	Actual	424×320

where LM iteration/KF = number of Levenberg-Marquardt iterations per keyframe.

B Additional Results

Additional results for the different configurations of the visual odometry algorithms are provided in this section. Results for RT-SVO is presented in Table 3, S-PTAM is in Table 4 and S-DSO is detailed in Table 5.

Table 3: Translational and rotational error for different configurations of RT-SVO

	GFTT		FAST	
Sequences	**Trans.**	**Rot.**	**Trans.**	**Rot.**
00	11.91	1.81	**5.56**	**0.84**
01	29.67	1.46	**23.71**	**1.37**
02	26.11	1.74	**6.41**	**1.14**
03	**3.87**	1.71	4.54	**1.35**
04	4.64	2.29	**3.02**	**1.62**
05	**3.23**	1.56	6.21	**0.79**
06	2.49	0.79	**1.68**	**0.63**
07	18.26	6.81	**3.13**	**1.60**
08	**7.40**	**1.60**	13.71	1.73
09	**9.76**	**1.40**	10.25	1.70
10	8.48	3.09	**8.43**	**2.79**

where Trans. = translational error and Rot. = rotational error and the values in bold represent the best performing variant.

Table 4: Translational and rotational error for different configurations of S-PTAM

Sequences	GFTT-BRIEF		GFTT-LATCH		ORB-ORB	
	Trans.	**Rot.**	**Trans.**	**Rot.**	**Trans.**	**Rot.**
00	**0.18**	**0.05**	0.23	0.06	0.29	0.07
01	**26.36**	**0.32**	34.54	1.02	35.68	0.94
02	**0.31**	**0.06**	0.50	0.10	0.62	0.15
03	**0.32**	**0.09**	0.52	0.13	0.55	0.17
04	**0.32**	**0.14**	0.39	0.14	1.10	0.30
05	**0.16**	**0.04**	0.18	0.07	∞	∞
06	**0.36**	**0.10**	0.42	0.14	0.42	0.13
07	**0.26**	**0.13**	0.29	0.16	1.59	0.60
08	**0.23**	**0.06**	0.30	0.07	0.44	0.06
09	**0.49**	**0.08**	0.50	0.09	1.09	0.24
10	**0.31**	0.12	0.33	**0.11**	0.48	0.19

where Trans. = translational error and Rot. = rotational error and the values in bold represent the best performing variant. For sequence #05, tracking failed mid-way for ORB-ORB variant, hence the errors are mentioned as ∞.

Table 5: Translational and rotational error for different configurations of S-DSO

Sequences	Default		Fast	
	Trans.	**Rot.**	**Trans.**	**Rot.**
00	0.16	0.03	**0.15**	**0.03**
01	**0.57**	**0.02**	1.08	0.04
02	0.20	**0.03**	**0.14**	0.03
03	**0.32**	**0.05**	0.39	0.05
04	**0.20**	**0.05**	0.32	0.06
05	**0.16**	**0.03**	0.16	0.03
06	**0.33**	0.10	0.33	**0.09**
07	0.60	0.22	**0.35**	**0.12**
08	**0.21**	**0.04**	0.23	0.04
09	0.39	0.06	**0.30**	**0.04**
10	**0.21**	**0.06**	0.21	0.06

where Trans. = translational error and Rot. = rotational error and the values in bold represent the best performing variant.