2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Macau, China, November 4-8, 2019

Extending Monocular Visual Odometry to Stereo Camera Systems by Scale Optimization

Jiawei Mo[1] and Junaed Sattar[2]

***Abstract*— This paper proposes a novel approach for extending monocular visual odometry to a stereo camera system. The proposed method uses an additional camera to accurately estimate and optimize the scale of the monocular visual odometry, rather than triangulating 3D points from stereo matching. Specifically, the 3D points generated by the monocular visual odometry are projected onto the other camera of the stereo pair, and the scale is recovered and optimized by directly minimizing the photometric error. It is computationally efficient, adding minimal overhead to the stereo vision system compared to straightforward stereo matching, and is robust to repetitive texture. Additionally, direct scale optimization enables stereo visual odometry to be purely based on the direct method. Extensive evaluation on public datasets (*e.g.*, KITTI), and outdoor environments (both terrestrial and underwater) demonstrates the accuracy and efficiency of a stereo visual odometry approach extended by scale optimization, and its robustness in environments with challenging textures.**

I. Introduction

Localization is an essential feature for autonomous robot navigation; however, it can be challenging in certain environments such as indoors and underwater where GPS signals are unavailable or unique landmarks are difficult to detect. Visual odometry (VO) has been widely used for robot localization, which estimates *ego-motion* using only camera(s). Cameras are *passive* sensors and thus consume less energy compared to *active* sensors such as sonar or laser range-finder (*i.e.*, LiDAR). Mobile robots, particularly those operating outdoors or in unstructured domains, benefit greatly from efficient energy usage as it extends the length of deployments, and also reduces downtime between missions.

Depending on the number of cameras in the system, visual odometry can be categorized into *monocular* or *multi-camera* system. Among multi-camera VO, stereo VO is the most widely used. The classic procedure of a stereo VO starts with *stereo matching*. Stereo matching searches feature correspondences between stereo frames; 3D positions of objects are then estimated instantly by triangulation. Subsequently, the camera pose (position and orientation) is estimated with respect to the 3D points. Since the 3D points are fully recovered, so is the camera pose. However, stereo matching can be computationally expensive. For each feature point, its correspondence is found by searching for the most similar patch along the epipolar line exhaustively. Additionally, many stereo matching algorithms will rectify the stereo pair first, which is also very time-consuming. Another challenge is that when the texture is repetitive and of high-frequency, there could be more than one similar patch that would give rise to ambiguity in best-match determination. These scenes are not uncommon outdoors and are often encountered by field robots, such as underwater or mine-exploration robots.

The authors are with the Department of Computer Science and Engineering, University of Minnesota Twin Cities, Minneapolis, MN, USA. {[1]moxxx066, [2]junaed} at umn.edu.

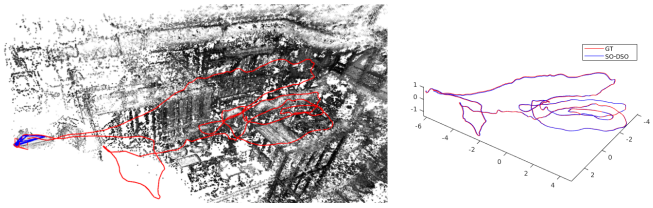


Fig. 1: A demonstration of a stereo VO using the proposed method running on MH01 of EuRoC dataset. Left image shows the trajectory and 3D points. Right image compares the trajectory against ground truth.

On the other hand, without the need to match points among different cameras, monocular VO algorithms (*e.g.,* [1]–[3]) are capable of camera tracking in these repetitive scenes and are computationally less expensive than multi-camera VO. As an example, SVO [2] needs only about 3 milliseconds to process each frame. Low-power platforms such as micro air vehicles benefit greatly from computational efficiency. However, monocular VO is not able to fully estimate camera pose. As the camera projects 3D objects onto 2D images, the distance to (*i.e.*, the depth of) the object is lost during this process. For monocular VO, depth is partially recovered from parallax by moving the camera temporally. However, since the camera movement is unknown as well, both depth and camera pose are estimated up to an unknown scale. The detailed discussion about the unknown scale is found in III-A. Additionally, the scale tends to drift so that it is inconsistent throughout the process. Scale awareness is important for a number of robotic behaviors including, but not limited to, vision-based control and path planning.

To solve the scale problem without intensive computational cost, authors [4]–[6] have fused monocular VO with an inertial measurement unit (IMU) to create visual-inertial navigation systems (VINS). In this case, the IMU provides scale estimation. However, IMU pose propagation is sensitive to measurement noise, thus visual measurements are used to correct the propagated pose. VINS achieve high accuracy and efficiency with a reliable IMU, which needs to be initialized at the beginning.

In this work, we propose a novel approach to solve the scale problem of monocular VO by incorporating an additional camera rather than an IMU. It combines the strengths of stereo and monocular VO in terms of accuracy

978-1-7281-4004-9/19/$31.00 ©2019 IEEE

and performance. Camera poses and 3D points are estimated by a monocular VO running on one camera; the other camera is only used to address the scale problem by projecting the 3D points from the monocular VO onto it. The optimal scale is solved by minimizing the photometric error in stereo projection. The main contributions of this work are the following:

- A novel algorithm to extend monocular VO to stereo,
- Full estimation of camera poses and 3D points with optimized scale,
- High accuracy and computational efficiency,
- Robustness in environments with challenging texture.

In the current implementation, each operation of scale optimization adds only about 2 to 3 milliseconds overhead (with around 2000 points) on average when extending a monocular VO to a stereo VO. We have evaluated an extended stereo VO using the proposed method on standard public datasets, as well as our own (publicly-available) datasets. Using the standard public datasets, we demonstrate that the proposed method achieves accuracy comparable to the state-of-the-art stereo matching-based VO with much less computational cost. In scenarios with challenging textures, the performance of state-of-the-art stereo VO degrades, while the proposed method performs sufficiently well without significant degradation of accuracy or performance (see §IV). An open-source implementation of this work is available online[1].

II. Related Work

Stereo VO has been widely explored, with many approaches [3], [7]–[10] relying on stereo matching. S-PTAM [9] is one of the recent developments in stereo VO, which extends PTAM [11] to a stereo system by using stereo matching to generate new 3D points. Stereo ORB-SLAM [3] is another example of stereo VO that depends on stereo matching. Engel et al. extended their monocular LSD-SLAM [12] to a stereo VO [8]. Monocular LSD-SLAM is purely based on direct method (directly minimizing photometric error, independent of feature matching), but as LSD-SLAM uses stereo matching, it is no longer a fully direct method. VO algorithms with stereo matching often suffer from the problems discussed in §I. They tend to fail if the scene texture is repetitive, and are not computationally efficient.

The stereo matching methods mentioned above mainly focus on the patch appearance (*e.g.*, normalized cross-correlation or feature descriptor) to determine stereo correspondence, referred to as 'local' methods. To improve the robustness of stereo matching, authors have looked at global stereo matching for VO which exploits non-local constraints such as smoothness. One example is the stereo VO developed by Stereolab for their ZED stereo camera [13]. While the localization accuracy of this approach could be improved, real-time performance is achieved by performing stereo matching on a GPU. This adds to energy consumption and increases system complexity, which is not desirable for mobile robots.

[1] https://github.com/jiawei-mo/scale_optimization

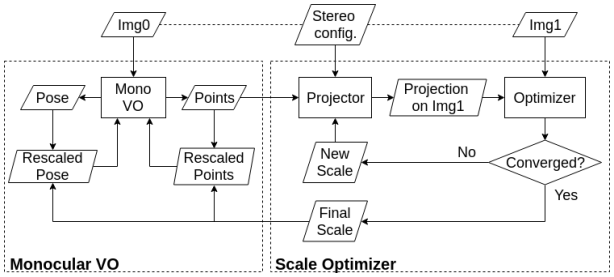


Fig. 2: Method overview. The two components, namely the Monocular VO (left) and the Scale Optimizer (right), run on two different cameras of the stereo pair. The Monocular VO tracks camera pose and reconstructs 3D points, whose scale is estimated/optimized by the Scale Optimizer.

Forster et al. extended their monocular SVO [2] for multi-camera systems [14], though not particularly for a stereo camera. Instead of stereo matching, they couple all cameras into one function to reduce photometric error. This error function is calculated by projecting 3D points onto all visible image frames. The accuracy is further improved and the scale problem is solved implicitly. However, computational cost significantly increases because of this augmented error function. Stereo DSO [15] is a hybrid model, which uses stereo matching to initialize depth for each keyframe; the stereo image is also coupled into the error function. In spite of the computational cost, Stereo DSO is a highly accurate approach to VO.

III. Methodology

Fig. 2 shows an overview of the proposed algorithm. For the current implementation, we adopt DSO [1] to perform monocular VO and enhance it to a two-camera system using the proposed scale optimization method. However, any monocular VO algorithm can be used in this step. DSO was chosen for two reasons. First, as of the time of writing, DSO demonstrates state-of-the-art accuracy among monocular VO methods. The accuracy of the extended stereo VO using scale optimization is strongly dependent on the underlying monocular VO. Second, DSO is one of the few existing monocular VOs which are purely based on direct method. Since scale optimization is also purely based on direct method, the extended stereo VO is thus purely based on direct method.

Notation: Being consistent with DSO, we use lower-case letters (d) to represent scalars, bold lower-case letters ($\mathbf{t}$) to represent vectors, bold upper-case letters ($\mathbf{R}$) to represent matrices, and upper-case letters (I) to represent functions.

A. Monocular VO

As shown in Fig. 2, we use DSO as our Monocular VO to track camera poses and generate 3D points. Here we will briefly introduce DSO and then only focus on the components that are related to the scale. Readers are referred to [1] for other details.

DSO is based on direct method, camera poses are tracked by minimizing the photometric error. Being independent of feature description and matching gives direct visual odometry

the potential of running at high frame rates and makes it robust to repetitive texture. These advantages are inherited by the extended stereo VO with scale optimization. DSO is a keyframe-based VO. Bundle adjustment of camera poses and 3D points are conducted only for keyframes [16]. The other frames are tracked with respect to keyframes, and they are used to refine the 3D points (inverse depth in DSO). Therefore, the proposed scale optimization is only called for keyframes. This further reduces the overhead of extending DSO to a stereo system using scale optimization.

In DSO, the following error function[2] is used at each keyframe to optimize all camera poses and 3D points within the current sliding window:

$$E_{photo} = \sum_{i \in F} \sum_{\mathbf{p} \in P_i} \sum_{j \in obs(\mathbf{p})} E_{\mathbf{p}j} \tag{1}$$

$$E_{\mathbf{p}j} = \sum_{\mathbf{p} \in N_\mathbf{p}} w_\mathbf{p} ||I_j[\Pi_0(\mathbf{R}\Pi_0^{-1}(\mathbf{p}, d_\mathbf{p}) + \mathbf{t})] - I_i[\mathbf{p}]||_\gamma \tag{2}$$

However, Eq. (2) is invariant to scale. If we re-scale the translation $\mathbf{t}$ and 3D points $\Pi_0^{-1}(\mathbf{p}, d_\mathbf{p})$ with a factor of s, the Eq. (2) is unchanged:

$$\begin{aligned} E'_{\mathbf{p}j} &= \sum_{\mathbf{p} \in N_\mathbf{p}} w_\mathbf{p} ||I_j[\Pi_0(\mathbf{R} \cdot s\Pi_0^{-1}(\mathbf{p}, d_\mathbf{p}) + s\mathbf{t})] - I_i[\mathbf{p}]||_\gamma \\ &= \sum_{\mathbf{p} \in N_\mathbf{p}} w_\mathbf{p} ||I_j[\Pi_0(s(\mathbf{R}\Pi_0^{-1}(\mathbf{p}, d_\mathbf{p}) + \mathbf{t}))] - I_i[\mathbf{p}]||_\gamma = E_{\mathbf{p}j} \end{aligned}$$

Thus, monocular DSO is unaware of scale. With more error coming into the system, the scale tends to drift. An example can be seen in Fig. 3. Readers are referred to [8] for notations and detailed treatments.

Stereo DSO [15] solved the scale problem by using stereo matching to initialize depth and extending the error term (1) to:

$$E_{photo} = \sum_{i \in F} \sum_{\mathbf{p} \in P_i} (\sum_{j \in obs(\mathbf{p})} E_{\mathbf{p}j} + \lambda E_{\mathbf{p}s})$$

where $E_{\mathbf{p}s}$ is the photometric error when projecting onto the stereo frame. It is coupled into the system by a weight of λ. The scale problem is implicitly solved by integrating the stereo baseline into the error function. Stereo DSO exhibits high accuracy, but its computational cost is much higher than that of monocular DSO. It adopts stereo rectification for stereo matching, but stereo rectification itself is computationally slow. Also, stereo matching makes Stereo DSO not fully based on direct method.

B. Scale Optimization

With the goal of solving the scale issue of monocular VO with minimal computational cost, while still being robust in challenging, texture-depleted environments, we propose our scale optimization method that extends a monocular VO to a stereo VO effectively and efficiently. As Fig. 2 shows, scale optimization is of modular design which makes it trivial to integrate this into any existing monocular VO algorithm. The inputs to the scale optimization are the 3D points from monocular VO, and the output is the optimized scale of the current frame. The optimized scale is then integrated back into the monocular VO for scale adjustment.

For each keyframe, DSO performs bundle adjustment to optimize the camera poses and 3D points jointly. Subsequently, the optimized 3D points are handed over to the scale optimizer. They are projected onto the stereo frame (*Img1* in Fig. 2) to find an optimal scale such that the photometric error is minimized:

$$E_{scale} = \sum_{\mathbf{p} \in P} E_{\mathbf{p}scale} \tag{3}$$

$$E_{\mathbf{p}scale} = w_\mathbf{p} ||I_1[\Pi_1(\mathbf{T}_{stereo} \cdot s\Pi_0^{-1}(\mathbf{p}, d_\mathbf{p}))] - I_0[\mathbf{p}]||_\gamma \tag{4}$$

For each pixel $\mathbf{p}$ with its depth $d_\mathbf{p}$, it is first back-projected to 3D space by $\Pi_0^{-1}(\mathbf{p}, d_\mathbf{p}))$, then it is re-scaled by current scale s. The re-scaled 3D point is transformed to the stereo camera coordinate by $\mathbf{T}_{stereo}$, which is projected onto the stereo image frame by Π_1. The photometric error is calculated as the Huber norm of pixel intensity difference.

The error term in Eq. (4) is simplified compared to the error term in Eq. (2). Eq. (4) only focuses on the exact pixel at the projection instead of a pattern around the projection as in Eq. (2), in order to further reduce computational cost. The projection Π_1 in Eq. (4) is parameterized by the relative pose between the stereo cameras ($\mathbf{T}_{stereo}$) and current scale of the 3D points (s) (*i.e.*, *Stereo config.* and *New Scale* in Fig. 2). $\mathbf{T}_{stereo}$ is pre-calibrated and fixed, so the scale s is the only variable in the system. Thus, focusing on a single pixel is feasible for scale optimization, which is validated in §IV. The necessity of using the pattern in Eq. (2) is due to its high degrees of freedom including all camera poses and depths.

We use Gauss-Newton optimization [17] to solve Eq. (3). We write the photometric residual as:

$$\begin{aligned} r_{\mathbf{p}scale} &= I_1[\Pi_1(\mathbf{T}_{stereo} \cdot s\Pi_0^{-1}(\mathbf{p}, d_\mathbf{p}))] - I_0[\mathbf{p}] \\ &= I_1[\Pi_1(s \cdot \mathbf{R}_{stereo}[x, y, z]^T + \mathbf{t}_{stereo})] - I_0[\mathbf{p}] \\ &= I_1[\Pi_1(s \cdot [x', y', z']^T + [t_x, t_y, t_z]^T)] - I_0[\mathbf{p}] \\ &= I_1[\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s \cdot x' + t_x \\ s \cdot y' + t_y \\ s \cdot z' + t_z \end{bmatrix}] - I_0[\mathbf{p}] \\ &\doteq I_1[\begin{bmatrix} \frac{s \cdot f_x x' + f_x t_x}{s \cdot z' + t_z} + c_x \\ \frac{s \cdot f_y y' + f_y t_y}{s \cdot z' + t_z} + c_y \end{bmatrix}] - I_0[\mathbf{p}] \end{aligned}$$

where f_x, f_y, c_x, c_y are intrinsic parameters of *Img1* in Fig. 2; $[x', y', z']^T$ is the 3D point rotated by $\mathbf{R}_{stereo}$; and $\mathbf{t}_{stereo} = [t_x, t_y, t_z]^T$. The Jacobian of $r_{\mathbf{p}scale}$ with respect to the scale s is:

$$\mathbf{J_s} = \frac{\partial I_1}{\partial \Pi_1} \cdot \frac{\partial \Pi_1}{\partial s}$$

$\frac{\partial I_1}{\partial \Pi_1}$ is the image gradient at the projection on *Img1*,

$$\frac{\partial \Pi_1}{\partial s} = \frac{1}{(s \cdot z' + t_z)^2} \begin{bmatrix} f_x x' t_z - f_x z' t_x \\ f_y y' t_z - f_y z' t_y \end{bmatrix}$$

[2]The affine brightness terms in Eq. (2) are ignored for simplicity.

At each iteration, the Gauss-Newton algorithm will solve the above system and get a scale increment. The new scale is updated as $s_{new} = s_{old} + s_{inc}$. After convergence, the final scale is fed back into the monocular VO. Consequently, the scale of the system is accurate and consistent.

One note is that the inverse compositional method [18] is not feasible for scale optimization, because changing the scale of 3D points does not change its projection onto the original image (*Img0*).

We use image pyramids to optimize scale from coarse to fine. Coarse-to-fine strategy is especially useful for the first keyframe, where the scale is totally unknown. Alternatively, stereo matching could be called at the first frame to initialize the scale.

Stereo correspondences are implicitly found all at once by optimizing the scale. Compared to explicit stereo matching, the proposed method is more robust to challenging scenes as the 3D points are already partially reconstructed by the monocular VO. Integrating them in a single error function (Eq. (3)) for scale optimization is more robust than individual stereo matching, especially when the scene has repetitive textures. Experimental evaluations described in the following section further underscore this point.

IV. Experimental Evaluation

We evaluate the accuracy and efficiency of scale optimized VO through a number of experiments. These include tests on two publicly-available datasets: the KITTI Visual Odometry dataset [19] and the EuRoC MAV dataset [20]. We compare our extended DSO using scale optimization against Stereo DSO [15]. For naming convenience, we refer to the extended DSO using scale optimization as **SO-DSO**. For Stereo DSO, only third-party implementations are available, since the original authors are yet to publish their code, leading us to choose the best-performing[3] among these. This particular implementation achieves reasonably high accuracy in our experiments. We choose Stereo DSO to compare with so that both algorithms use the same camera tracking algorithm (DSO), which makes it possible to directly compare stereo matching/coupling with scale optimization. We compare the accuracy of visual odometry, as well as the extra cost as stereo systems over the monocular DSO. We adopt the evaluation method used in KITTI VO benchmark, which evaluates the accuracy of the trajectories with different lengths. When testing the run-time, we use the default (slowest) setting of DSO (2000 active points, 7 max keyframes, etc.), not enforcing real-time performance, in order to maximize the accuracy. For clarity, we focus on the run-time of different components between SO-DSO and Stereo DSO, which are the scale optimizer in SO-DSO, stereo matching in Stereo DSO, and the different cost function in bundle adjustment. The experiments are carried out on a single thread of an Intel i7-6700 CPU.

[3] https://github.com/JingeTu/StereoDSO

Seq.	t_{rel} (%)	r_{rel} (deg)	S.O. S.M. (ms)	BA (ms)	TPF (ms)	Pts
00	1.35	0.27	**2.25**	**124.07**	**141.95**	2164.28
	0.83	0.27	12.40	147.63	189.88	1658.76
01	2.72	0.13	**1.85**	**66.38**	**76.42**	1437.18
	1.78	**0.11**	10.75	73.32	123.27	1133.11
02	1.10	0.22	**2.23**	121.42	**164.16**	2019.06
	0.79	**0.21**	11.28	**111.71**	171.52	1426.85
03	3.17	**0.15**	**2.44**	115.32	**97.78**	2241.95
	1.01	0.16	11.34	**109.05**	109.47	1592.40
04	1.73	0.21	**2.09**	**87.40**	**122.44**	1926.45
	1.01	**0.19**	11.15	104.69	160.64	1599.01
05	1.69	0.20	**2.11**	**108.60**	**119.62**	2028.98
	0.82	**0.18**	11.20	113.93	145.99	1647.78
06	**1.66**	0.19	**2.09**	**85.05**	**110.03**	1718.27
	9.19	**0.17**	11.05	90.44	125.06	1270.94
07	2.50	**0.32**	**2.22**	**113.50**	**113.77**	2153.60
	1.03	0.33	10.66	119.33	134.84	1716.57
08	1.72	**0.26**	**2.08**	**109.24**	**126.58**	1945.50
	1.04	0.27	11.15	118.36	155.55	1497.82
09	1.88	0.22	**2.04**	**98.95**	**130.83**	1900.30
	0.98	**0.19**	11.22	102.23	150.79	1457.92
10	1.02	0.21	**1.89**	**88.84**	**102.70**	1775.74
	0.61	**0.19**	10.60	93.38	117.87	1357.89

TABLE I: Error and run-time comparisons on the KITTI dataset. For each sequence, **the upper line is the result of SO-DSO, and the lower line is for Stereo DSO**. t_{rel} is translational RMSE(%); r_{rel} is rotational RMSE (degree per 100m). Results are averaged over $100m$ to $800m$ intervals. S.O. is the run-time of scale optimization; S.M. is the run-time of stereo matching; BA is the bundle adjustment run-time; TPF is the time per frame (not just keyframe); Pts is the number of 3D points in the bundle adjustment.

A. KITTI VO Dataset

KITTI VO Dataset has 22 driving sequences. The vehicle drives around local communities and highways capturing stereo image sequences. The ground truth is provided by a Velodyne laser scanner and a GPS localization system. However, only the first 11 sequences are publicly available with ground truth; the ground truth of Sequences 11 to 21 are reserved for test and ranking of different VO algorithms. We present results from the first 11 sequences for comparison since we do not have full access to the errors on test sequences 11 to 21.

Table I shows the comparison between SO-DSO and Stereo DSO. It is worth noting that the errors of the third party implementation are quite close to the errors reported in the original Stereo DSO paper [15] (The error of Seq. 06 corresponds to the Figure. 4 in [15] with coupling factor 1). In most cases, Stereo DSO achieves higher translational accuracy. This is expected because SO-DSO depends on monocular DSO for generating 3D points, but monocular DSO is not designed for quick camera movement or low camera frame-rate (10Hz). Even worse, there are lots of sharp turns in many sequences. In Stereo DSO, integrating static stereo drastically increases accuracy and robustness for these challenging cases. We suggest static stereo for those cases where monocular VO does not work well. Nevertheless, the

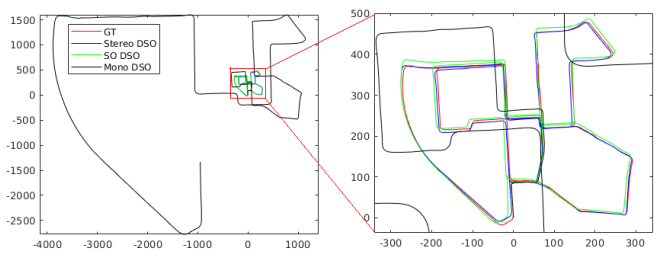


Fig. 3: Effect of scale optimization on KITTI Seq. 00. Trajectories of ground truth (GT), Stereo DSO, SO-DSO, and monocular DSO are shown.

accuracy of SO-DSO is comparable to that of Stereo DSO.

On the efficiency factor, however, scale optimization is approximately 5 times faster than stereo matching. In the current implementation, we use 4 image pyramids for robustness. After the scale is initialized, it is not necessary to use as many pyramids; thus, run-time can be further reduced. On the other hand, Stereo DSO maintains a lower number of 3D points. Stereo matching does not work well for repetitive textures, neither can it triangulate points far away since there is no disparity. The KITTI dataset has plants and far-away objects, which could be challenging for stereo matching. With more points to optimize, the bundle adjustment in SO-DSO is still faster than the one in Stereo DSO. One reason is that Stereo DSO projects points onto both stereo frames, the number of error terms is drastically increased (while improving accuracy). As a system, SO-DSO spends less time per frame (TPF), even with more points, than Stereo DSO. One point to note is that the majority of TPF is taken by monocular DSO. In theory, the overhead of SO-DSO over monocular DSO is the time taken for scale optimization (plus the negligible time needed for accessing 3D points and scale adjustment). Additionally, Stereo DSO requires data pre-processing of stereo rectification, which is also time-consuming.

Fig. 3 demonstrates the effectiveness of scale optimization qualitatively. We initialize the scale of monocular DSO at the beginning only with no further scale optimization, which is labeled as Mono DSO. The trajectory is close to ground truth at the beginning, but completely deviates as the scale drift becomes extremely large. However, with scale optimization throughout, the trajectory (SO-DSO) is always close to the ground truth, though not as close as Stereo DSO.

B. EuRoC MAV Dataset

We also compare SO-DSO with Stereo DSO on the EuRoC dataset. The dataset was recorded by a drone in two scenarios, Machine Hall and Vicon Room. The ground truth for Machine Hall was measured by a Leica MS50 laser tracker, which contains 3D position only. Thus, no rotational error is measured in Machine Hall tests. The ground truth for the Vicon Room was measured by a Vicon motion capture system obtaining both position and orientation.

The results are given in Table II. For Machine Hall tests, both Stereo DSO and SO-DSO work well, at least for 'easy' and 'medium' tests. One example of SO-DSO on Machine Hall tests is given in Fig. 1. Compared with KITTI tests, the error of SO-DSO, in this case, is lower. One factor is the high frame rate (20Hz) of EuRoC dataset. On the other hand, averaging results over smaller intervals could be the reason that Stereo DSO has a higher error rate. For Vicon Room tests, neither work for 'hard' tests; SO-DSO works for 'easy' and 'medium' tests with low error, while Stereo DSO has a noticeable reduction in accuracy. The possible reasons are image blur due to fast camera motion and poor illumination. Note that Stereo DSO has relatively more 3D points in bundle adjustment for EuRoC than that it has in KITTI. The scale of Machine Hall/Vicon Room is smaller than the street scenes in KITTI, which means more points are within the stereo camera range. From Table II, we see that the accuracy of SO-DSO is comparable to, if not better than, Stereo DSO (with the 3rd party implementation).

Seq.	t_{rel} (%)	r_{rel} (deg)	S.O. S.M. (ms)	BA (ms)	TPF (ms)	Pts
MH_01	**0.23**	N/A	**3.45**	**104.69**	**36.81**	2770.68
easy	1.51	N/A	10.72	157.57	60.40	2728.09
MH_02	**0.28**	N/A	**3.42**	**105.42**	**35.03**	2734.76
easy	1.28	N/A	10.74	155.26	55.65	2691.50
MH_03	**0.59**	N/A	**3.32**	**121.23**	**58.24**	2705.65
medium	1.62	N/A	10.57	178.55	91.03	2659.79
MH_04	**0.76**	N/A	**3.42**	**125.00**	**51.09**	2705.13
hard	x	x	x	x	x	x
MH_05	**0.63**	N/A	**3.23**	**116.70**	**46.24**	2674.09
hard	1.22	N/A	10.44	172.04	75.05	2592.49
V1_01	**1.70**	**21.63**	**3.53**	**127.66**	**52.20**	2715.68
easy	2.42	22.50	10.70	186.72	84.04	2654.30
V1_02	**0.78**	**7.06**	**3.08**	**133.08**	**92.49**	2720.20
medium	2.66	43.98	10.27	208.00	139.12	2709.03
V1_03	x	x	x	x	x	x
hard	x	x	x	x	x	x
V2_01	**0.57**	**7.14**	**3.51**	**136.74**	**47.21**	2756.59
easy	13.85	137.62	9.87	158.41	63.85	2232.31
V2_02	**2.50**	**6.76**	**3.61**	**132.38**	**84.83**	2780.82
medium	9.63	152.00	9.77	185.18	102.70	2487.47
V2_03	x	x	x	x	x	x
hard	x	x	x	x	x	x

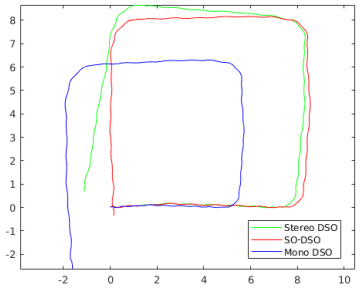
TABLE II: Error and run-time comparison on EuRoC. Same notation as in Table I, except that results are averaged over $10m$ to $80m$ intervals.

Also, the high computational efficiency of our approach is further validated on this dataset. Scale optimization is faster than stereo matching, and bundle adjustment of SO-DSO is faster than that of Stereo DSO.

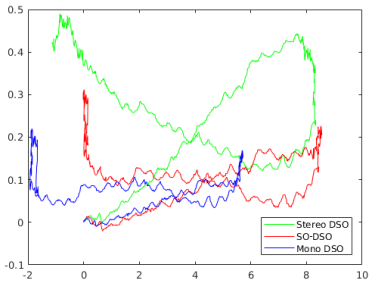
The TPF drops significantly for both algorithms. The drone used in EuRoC moved slower (<1m/s) than the cars in KITTI (>4m/s), and the EuRoC dataset has a higher frame rate. The distance of camera movement is one important factor for DSO to create new keyframes. Since the camera movement is subtle, fewer keyframes are selected in EuRoC dataset. Thus, both algorithms run faster on the EuRoC dataset, with SO-DSO being the faster of the two.

After the comparison of Stereo DSO and SO-DSO on both KITTI and EuRoC datasets, it is evident that extending

(a) A view of the grass dataset.

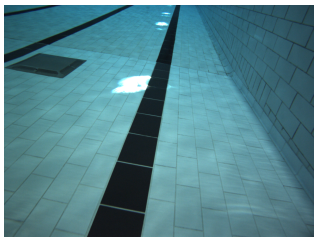


(b) Trajectories (top view, in meters) estimated by the three algorithms.

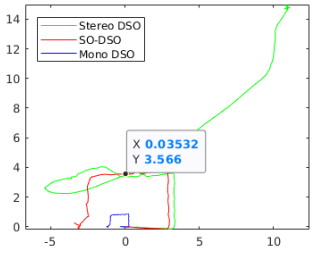


(c) Elevation changes in the trajectories (in meters) estimated by the three algorithms.

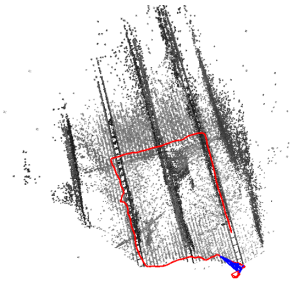
Fig. 4: ZED camera experiment, ground-truth is approximately a zero-elevation, $8m \times 8m$ square.



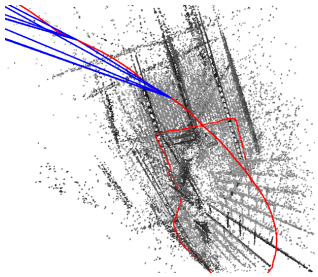
(a) A snapshot of the swimming pool dataset.



(b) Robot trajectories (in meters) estimated by the three algorithms in the pool.



(c) Reconstructed pool (top view) by SO-DSO, with robot trajectory overlaid. SO-DSO converges throughout



(d) Reconstructed pool (top view) by Stereo DSO, with robot trajectory overlaid. Stereo DSO diverges halfway through the trajectory.

Fig. 5: Evaluating VO in a pool environment on an AUV. The width of two swimming lanes combined is about 3.6m.

monocular VO using scale optimization significantly reduces computational cost without a significant loss of accuracy.

C. Terrestrial Data

To validate the robustness of VO with scale optimization in outdoor settings, we use a ZED camera to record a stereo dataset, a snapshot of which is shown in Fig. 4a. The camera is carried by hand on an approximately $8m \times 8m$ square path without much elevation change. The camera is pointed at the grass throughout the trajectory. Because of the sun, the brightness changes drastically when moving into the shadows. This dataset is used intentionally to challenge stereo matching, where the grass is repetitive and of high frequency.

Fig. 4b shows the results. We first run monocular DSO on this dataset, and as the blue trajectory shows, its scale is incorrect and inconsistent. If we run scale optimization on top of monocular DSO (*i.e.*, SO-DSO), the trajectory is roughly a flat $8m \times 8m$ square, and returns to the start position, without significant elevation change (Figure 4c). On the other hand, the trajectory generated by Stereo DSO has a quite accurate scale, but the camera does not return to the start position due to rotational error. A possible reason is that the wrong stereo correspondences degrade the accuracy, which is already mentioned in the Stereo DSO paper [15].

D. Underwater Data

We also evaluate SO-DSO and Stereo DSO on a dataset we collected using the Aqua underwater robot [21] in a swimming pool, illustrated in Fig. 5a. This represents a challenging environment for VO methods because of the reflections that occur on the water and the lack of distinguishable visual features.

The results are shown in Fig. 5b. Monocular VO and SO-DSO generate two similar trajectories but with different scales. With scale optimization, the distance between the two red horizontal lines in Fig. 5b is around 3.566 meters, which is close to the ground truth (which was measured to be approximately $3.6m$). Stereo DSO works quite well at the early stage but fails to converge to the ground truth towards the end, and this behavior was replicated across multiple evaluations on this underwater dataset. Fig. 5c and Fig. 5d show the top-view of the reconstructed swimming pool environment as a qualitative comparison between the two methods. Additionally, a video demonstration of the proposed method accompanies the paper, and the datasets of §IV-C and §IV-D are available online[4].

[4]https://drive.google.com/open?id=1r-vsnkythfqqq0Ly0QZ34_W9Ay8cIJPN

V. Conclusions

In this paper, we proposed a new algorithm for extending monocular visual odometry to a stereo system. It combines the advantages of monocular visual odometry and stereo visual odometry, namely computational efficiency and scale awareness. For demonstration, the monocular DSO is used to track the camera poses and generate 3D points; while the other camera in the stereo setup is used to optimize the scale of DSO. In experimental validations on public datasets and real-world recorded data, we show the proposed scale optimization approach to be very fast and reasonably accurate, and also robust in scenes of challenging texture.

Future extensions to this work will include monocular VO failure detection so that scale optimization and stereo matching can alter in different scenarios, in order to balance accuracy and efficiency. We also intend to include loop-closing into the extended visual odometry to further improve accuracy.

Acknowledgment

We gratefully acknowledge the support of the MnDRIVE initiative for this research.

References

[1] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2018.

[2] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 15–22, IEEE, 2014.

[3] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[4] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[5] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct ekf-based approach," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 298–304, IEEE, 2015.

[6] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, no. 99, pp. 1–17, 2018.

[7] I. Cvišić and I. Petrović, "Stereo odometry based on careful feature selection and tracking," in *Mobile Robots (ECMR), 2015 European Conference on*, pp. 1–6, IEEE, 2015.

[8] J. Engel, J. Stückler, and D. Cremers, "Large-scale direct slam with stereo cameras," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 1935–1942, IEEE, 2015.

[9] T. Pire, T. Fischer, G. Castro, P. De Cristóforis, J. Civera, and J. J. Berlles, "S-PTAM: Stereo parallel tracking and mapping," *Robotics and Autonomous Systems*, vol. 93, pp. 27–42, 2017.

[10] R. Gomez-Ojeda and J. Gonzalez-Jimenez, "Robust stereo visual odometry through a probabilistic combination of points and line segments," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 2521–2526, IEEE, 2016.

[11] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pp. 225–234, IEEE, 2007.

[12] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision*, pp. 834–849, Springer, 2014.

[13] "ZED Stereo Camera." `https://www.stereolabs.com/zed/`. Accessed: 2019-07-29.

[14] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect visual odometry for monocular and multicamera systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2017.

[15] R. Wang, M. Schworer, and D. Cremers, "Stereo DSO: Large-scale direct sparse visual odometry with stereo cameras," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3903–3911, 2017.

[16] H. Strasdat, J. Montiel, and A. J. Davison, "Real-time monocular SLAM: Why filter?," in *2010 IEEE International Conference on Robotics and Automation*, pp. 2657–2664, IEEE, 2010.

[17] A. P. Ruszczyński and A. Ruszczynski, *Nonlinear Optimization*, vol. 13. Princeton university press, 2006.

[18] R. Brooks and T. Arbel, "Generalizing inverse compositional image alignment," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 2, pp. 1200–1203, IEEE, 2006.

[19] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[20] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[21] G. Dudek, P. Giguère, C. Prahacs, S. Saunderson, J. Sattar, L.-A. Torres-Mendez, M. Jenkin, A. German, A. Hogue, A. Ripsman, J. Zacher, E. Milios, H. Liu, P. Zhang, M. Buehler, and C. Georgiades, "Aqua: An Amphibious Autonomous Robot," *IEEE Computer Magazine*, vol. 40, pp. 46–53, 1 2007.