Direct Visual Odometry for a Fisheye-Stereo Camera

Peidong Liu[1], Lionel Heng[2], Torsten Sattler[1], Andreas Geiger[1,3], and Marc Pollefeys[1,4]

***Abstract*— We present a direct visual odometry algorithm for a fisheye-stereo camera. Our algorithm performs simultaneous camera motion estimation and semi-dense reconstruction. The pipeline consists of two threads: a tracking thread and a mapping thread. In the tracking thread, we estimate the camera pose via semi-dense direct image alignment. To have a wider field of view (FoV) which is important for robotic perception, we use fisheye images directly without converting them to conventional pinhole images which come with a limited FoV. To address the epipolar curve problem, plane-sweeping stereo is used for stereo matching and depth initialization. Multiple depth hypotheses are tracked for selected pixels to better capture the uncertainty characteristics of stereo matching. Temporal motion stereo is then used to refine the depth and remove false positive depth hypotheses. Our implementation runs at an average of 20 Hz on a low-end PC. We run experiments in outdoor environments to validate our algorithm, and discuss the experimental results. We experimentally show that we are able to estimate 6D poses with low drift, and at the same time, do semi-dense 3D reconstruction with high accuracy. To the best of our knowledge, there is no other existing semi-dense direct visual odometry algorithm for a fisheye-stereo camera.**

I. INTRODUCTION

Forster et al. [8] and Engel et al. [5] started a new wave of semi-direct and direct visual odometry (VO) methods respectively for high-speed motion estimation. Previously, traditional feature-based visual odometry methods [19, 10] were dominant, but have fallen out of favor as they involve the computationally expensive tasks of feature descriptor computation and outlier identification in a set of feature correspondences. These tasks have been made redundant by semi-direct and direct visual odometry methods which directly use pixel intensity values instead of hand-crafted high-dimensional feature descriptors. Consequently, these methods are able to run at high frame rates and with less computational resources, and are most suited for computationally-constrained mobile robotic applications.

Semi-direct and direct visual odometry methods generally follow the simultaneous-tracking-and-mapping paradigm [14]. These two types of methods are similar in the aspect that pixel intensity values are used for both motion estimation and stereo matching. However, semi-direct methods use indirect pose and structure optimization which minimizes feature-based reprojection errors while direct methods use direct pose and structure optimization which minimizes photometric errors. The semi-direct approach of Forster et al. [8] has both direct and feature-based characteristics. The direct method of image alignment is used to estimate the initial motion. At the same time, feature-based methods are used to implicitly obtain feature correspondences through feature alignment, and feature-based optimization is used to optimize the pose and landmark coordinates based on feature correspondences over a window of frames. In contrast, Engel et al. [5] exclusively uses direct techniques. A semi-dense depth map is propagated from frame to frame, and refined with depth values that are obtained from variable-baseline stereo matching for pixels with high gradient values. At the same time, the depth map with respect to the most recent keyframe is used to track current frame via direct image alignment. Another difference between the semi-direct approach of Forster et al. [8] and the direct approach of Engel et al. [5] is the number of pixels used for motion estimation. Similarly to conventional feature-based VO methods, the semi-direct approach samples features very sparsely while the direct approach uses a much larger number of pixels with high gradients. In other words, the direct approach is able to do motion estimation and semi-dense 3D reconstruction simultaneously. This motivates us to follow the formulation of the direct method proposed by Engel et al. [5] for our algorithm.

The use of a monocular pinhole camera in [5, 8, 20] leads to scale drift. Researchers have proposed the use of inertial measurement units (IMUs) [15, 17] and stereo cameras [4, 16] to eliminate scale drift. We see a stereo camera as more robust than an IMU; noisy accelerometer measurements on vehicle platforms with considerable amounts of vibration do not allow precise observations of scale whereas a stereo camera is immune to effects of vibration as long as the mount does not allow the vibration to distort the extrinsic stereo parameters. Furthermore, fisheye cameras improve robustness by significantly increasing the number of image pixels corresponding to static areas in dynamic environments and which can be used for accurate motion estimation.

Therefore, we propose a direct visual odometry algorithm for a fisheye-stereo camera in this paper. The proposed algorithm enables mobile robots to do motion estimation and semi-dense 3D reconstruction simultaneously. Real world experimental results show that our algorithm not only gives low-drift motion estimation but also is able to do accurate semi-dense 3D reconstruction.

The rest of the paper is organized as follows. Section II provides an overview of the related work. Section III lists the main contributions of our method. Section IV and Section V describes the formulations of our algorithm in detail.

[1]Computer Vision and Geometry Group, Department of Computer Science, ETH Zürich, Switzerland

[2]Robotics Autonomy Lab, Manned - Unmanned Programme, Information Division, DSO National Laboratories, Singapore

[3]Autonomous Vision Group, MPI for Intelligent Systems, Tübingen, Germany

[4]Microsoft, Redmond, USA

Section VI discusses the evaluation results of our proposed method with real-world datasets.

II. Related Work

A. Monocular vs. Stereo

Forster et al. [9] and Engel et al. [7] make extensions to their seminal work in semi-direct [8] and direct visual odometry [5] respectively for monocular cameras moving at high speeds. In [9], edgelet features are tracked in addition to corner features. In turn, motion estimates are more accurate, especially in environments lacking in corner features. They use motion priors from either a constant velocity model or gyroscopic measurements to make tracking more robust and faster. With motion priors, the initial motion estimate for image alignment is closer to the actual value, resulting in convergence at the global minimum, especially in the presence of local minima. Furthermore, convergence is faster due to a fewer number of iterations in optimisation. In contrast to Forster et al. [9] which both minimizes photometric and geometric (reprojection) errors, Engel et al. [7] only minimizes photometric errors. In this case, for more accurate results, a photometric calibration is used in addition to a standard geometric calibration that models image projection and lens distortion. Such a photometric calibration accounts for a non-linear response function, lens vignetting, and exposure time. Besides camera poses, they also optimize affine brightness parameters, inverse depth values, and camera intrinsics.

Stereo visual odometry resolves scale ambiguity in monocular visual odometry by estimating metric depth from stereo images with a known baseline. Forster et al. [9] expand their original semi-direct visual odometry method [8] for monocular cameras to multi-camera systems. Omari et al. [18] follow the direct approach advocated by Engel et al. [5] but neither propagate nor update a semi-dense depth map. Instead, they directly obtain the depth map from stereo block matching applied to rectified stereo images, and uses this depth map to track new frames. Engel et al. [6] improve on their original direct visual odometry method [5] by incorporating depth measurements from static stereo in addition to those from temporal stereo.

B. Pinhole vs. Fisheye

With a fisheye camera comes a large field of view. This can be advantageous in urban settings with many moving vehicles which can otherwise occlude a pinhole camera's field of view, making tracking impossible. Both semi-direct and direct visual odometry methods for pinhole cameras typically perform disparity search along the epipolar line. However, epipolar curves instead of epipolar lines exist in fisheye images. A fisheye camera brings added computational complexity to epipolar stereo matching as epipolar curves are much more expensive to compute. Caruso et al. [3] do a disparity search along the epipolar curve. We can circumvent the problem of epipolar curves by using the naïve step of extracting rectified pinhole images from fisheye images at the expense of a significant loss of field-of-view. Heng and Choi [13] propose a semi-direct approach for a fisheye-stereo camera and which directly operates on fisheye images. They use a fisheye camera model in both image alignment, and pose and structure refinement, and leverage plane-sweeping stereo for direct stereo matching with fisheye images. However, there exists no direct approach for a fisheye-stereo camera to the best of our knowledge.

III. Contributions

The main contribution of this paper is a direct visual odometry algorithm for a fisheye-stereo camera. To the best of our knowledge, no direct visual odometry algorithm exists for a fisheye-stereo camera.

Our paper is most similar in spirit to that of Engel et al. [5] with three key differences:

1) We use fisheye cameras instead of pinhole cameras.
2) We avoid scale ambiguity by using depth measurements from wide-baseline stereo.
3) We keep track of multiple depth hypotheses when initializing depth values for new pixels to track.

In wide-baseline stereo, it is common to encounter many local minima, making it difficult to find correct depth values. We solve this problem by keeping track of multiple depth hypotheses corresponding to local minima, and eliminating incorrect hypotheses over time until we are left with a single depth hypothesis.

IV. Notation

We denote the world reference frame as $\underrightarrow{\mathcal{F}}_W$, the stereo-camera reference frame as $\underrightarrow{\mathcal{F}}_S$, and the individual camera reference frames as $\underrightarrow{\mathcal{F}}_{C_1}$ and $\underrightarrow{\mathcal{F}}_{C_2}$. Here, C_1 is the reference camera in the stereo camera, and thus, $\underrightarrow{\mathcal{F}}_{C_1}$ coincides with $\underrightarrow{\mathcal{F}}_S$. We denote the image from camera C_i at time step k as $\mathbf{I}^k_{C_i}$, and the stereo frame at time step k as $F^k_S = \{\mathbf{I}^k_{C_1}, \mathbf{I}^k_{C_2}\}$. $\mathbf{I}^k_{C_i}(\mathbf{u})$ is the intensity of the pixel with image coordinate $\mathbf{u}$.

The camera projection model $\boldsymbol{\pi} : \mathbb{R}^3 \mapsto \mathbb{R}^2$ projects a landmark with coordinates ${}_{C_i}\mathbf{p} = [x\ y\ z]^T$ in $\underrightarrow{\mathcal{F}}_{C_i}$ to an image point $\mathbf{u} = [u\ v]^T$ in $\mathbf{I}_{C_i}$:

$$\mathbf{u} = \boldsymbol{\pi}({}_{C_i}\mathbf{p}). \tag{1}$$

We use the unified projection model [2, 11] whose intrinsic parameters are known from calibration. Given the inverse projection function $\boldsymbol{\pi}^{-1}$, we recover the coordinates of the landmark corresponding to an image point $\mathbf{u}$ in $\mathbf{I}_{C_i}$ for which the depth $d_\mathbf{u} \in \mathbb{R}$ is known:

$${}_{C_i}\mathbf{p} = \boldsymbol{\pi}^{-1}(\mathbf{u}, d_\mathbf{u}). \tag{2}$$

If the depth is not known, we recover the ray $\mathbf{f}_\mathbf{u}$ in $\underrightarrow{\mathcal{F}}_{C_i}$ and passing through the landmark that corresponds to the image point $\mathbf{u}$ in C_i:

$${}_{C_i}\mathbf{f}_\mathbf{u} = \boldsymbol{\pi}^{-1}(\mathbf{u}). \tag{3}$$

We denote the stereo camera pose at time step k as a rigid body transformation $\mathbf{T}^k_{SW} \in SE(3)$ from $\underrightarrow{\mathcal{F}}_W$ to $\underrightarrow{\mathcal{F}}_S$ and whose rotation matrix part is $\mathbf{R}^k_{SW}$, corresponding quaternion part is $\mathbf{q}^k_{SW}$, and translation part is $\mathbf{t}^k_{SW}$. Similarly, we denote the pose of camera C_i at time step k with

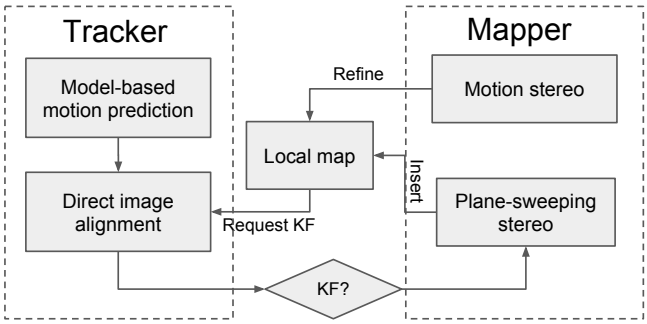

Fig. 1: System overview

$\mathbf{T}^k_{C_iW} = \mathbf{T}_{C_iS}\mathbf{T}^k_{SW}$. $\mathbf{T}_{C_iS}$ is known from calibration. The rigid body transformation $\mathbf{T}^k_{C_iW}$ maps a 3D point ${}_W\mathbf{p}$ in $\underrightarrow{\mathcal{F}}_W$ to a 3D point ${}_{C_i}\mathbf{p}$ in $\underrightarrow{\mathcal{F}}_{C_i}$:

$$ {}_{C_i}\mathbf{p} = \mathbf{T}^k_{C_iW}\,{}_W\mathbf{p}. \tag{4}$$

Given the fact that a rigid body transformation matrix is over-parameterized, on-manifold optimization requires a minimal representation of the rigid body transformation. In this case, we use the Lie algebra $se(3)$ corresponding to the tangent space of $SE(3)$. We denote the algebra elements, also known as twist coordinates, with $\boldsymbol{\xi} = [\mathbf{v}\ \boldsymbol{\omega}]^T$ where $\mathbf{v}$ is the linear velocity and $\boldsymbol{\omega}$ is the angular velocity. We use the exponential map to map the twist coordinates $\boldsymbol{\xi}$ in $se(3)$ to a rigid body transformation $\mathbf{T}$ in $SE(3)$:

$$ \mathbf{T}(\boldsymbol{\xi}) = exp(\boldsymbol{\xi}). \tag{5}$$

Similarly, we use the logarithm map to map a rigid body transform $\mathbf{T}$ in $SE(3)$ to twist coordinates $\boldsymbol{\xi}$ in $se(3)$:

$$ \boldsymbol{\xi} = log(\mathbf{T}(\boldsymbol{\xi})). \tag{6}$$

V. ALGORITHM

In this section, we describe our semi-sense visual odometry algorithm for a fisheye-stereo camera. As shown in Fig. 1, we follow the simultaneous-tracking-and-mapping paradigm [14], which encompasses a tracking thread and a mapping thread. The tracking thread tracks the current frame with respect to a keyframe. The mapping thread initializes and refines the depth map corresponding to the current keyframe; binocular stereo and temporal stereo are used for depth initialization and refinement respectively. The tracking thread only processes images from the reference camera in the stereo pair for efficiency considerations while the mapping thread processes images from both cameras. We describe each step in the pipeline in detail.

A. Semi-Dense Image Alignment

The tracking thread of our pipeline tracks the current frame against a keyframe by using semi-dense direct image alignment. In particular, we estimate the camera pose corresponding to the stereo-camera reference frame at time step r by minimizing the following cost function which is the sum of squared photometric errors:

$$ \mathbf{E}(\mathbf{T}^{\mathbf{k,r}}_{\mathbf{C_1}}) = \sum_{i\in\Omega(\mathbf{I}^r_{C_1})} r_i^2 \tag{7}$$

$$ = \sum_{i\in\Omega(\mathbf{I}^r_{C_1})} (\mathbf{I}^r_{C_1}(\mathbf{u}_i) - \mathbf{I}^k_{C_1}(\boldsymbol{\pi}(\mathbf{T}^{k,r}_{C_1}\boldsymbol{\pi}^{-1}(\mathbf{u}_i, d_{\mathbf{u}_i})))^2, \tag{8}$$

where $\Omega(\mathbf{I}^r_{C_1})$ is the set of high-gradient pixels in the keyframe, and $\mathbf{T}^{k,r}_{C_1}$ is the transformation from the keyframe to the current frame at time k. The camera pose corresponding to the current frame can be recovered from the known camera pose corresponding to the keyframe.

The above cost function is minimized iteratively by using the Gauss-Newton method. To achieve better efficiency, we employ the inverse compositional algorithm from [1]. Here, we do a one-time computation of the Jacobian and Hessian matrices instead of re-computing them for each iteration. Specifically, we minimize the following cost function for each iteration,

$$ \mathbf{E}(\boldsymbol{\xi}) = \sum_{i\in\Omega(\mathbf{I}^r_{C_1})} (\mathbf{I}^r_{C_1}(\boldsymbol{\pi}(\mathbf{T}(\boldsymbol{\xi})\boldsymbol{\pi}^{-1}(\mathbf{u}_i, d_{\mathbf{u}_i})) - \tag{9}$$

$$ \mathbf{I}^k_{C_1}(\boldsymbol{\pi}(\mathbf{T}^{k,r}_{C_1}\boldsymbol{\pi}^{-1}(\mathbf{u}_i, d_{\mathbf{u}_i})))^2. \tag{10}$$

Using the chain rule, we derive the Jacobian matrix:

$$ \mathbf{J}_i = \frac{\partial r_i(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} = \nabla \mathbf{I}^r_{C_1} \cdot \frac{\partial \boldsymbol{\pi}}{\partial_{C_i}\mathbf{p}_i} \cdot \frac{\partial(\mathbf{T}(\boldsymbol{\xi})\,{}_{C_1}\mathbf{p}_i)}{\partial \boldsymbol{\xi}}. \tag{11}$$

According to the Gauss-Newton method, we can compute the update $\boldsymbol{\xi}$:

$$ \boldsymbol{\xi} = -(\sum_{i\in\Omega(\mathbf{I}^r_{C_1})} \mathbf{J}_i^T\mathbf{J}_i)^{-1} \sum_{i\in\Omega(\mathbf{I}^r_{C_1})} r_i\mathbf{J}_i^T. \tag{12}$$

The motion estimate between the current frame and the keyframe is then updated:

$$ \mathbf{T}^{k,r}_{C_1} = \mathbf{T}^{k,r}_{C_1}\mathbf{T}(\boldsymbol{\xi})^{-1}. \tag{13}$$

We use the Huber robust weighting scheme for the residual error of each pixel to suppress the effect of outliers. To achieve better convergence performance, a coarse-to-fine approach using multiple pyramid levels is used in the optimization method. Furthermore, the initial camera pose for the above optimization procedure is estimated using a constant velocity motion model as shown in Fig. 1.

B. Plane-Sweeping Stereo

For a fisheye-stereo frame, pixel correspondences lie on epipolar curves instead of straight epipolar lines. Thus, conventional epipolar line disparity search algorithms cannot be used for fisheye stereo matching. To use fisheye images directly without rectifying them to pinhole images, we use the plane-sweeping stereo algorithm [12] for fisheye stereo matching to initialize the depth map of the keyframe. Plane-sweeping stereo assumes that scenes are locally planar and tests a set of plane hypotheses. These plane hypotheses are

used to warp pixels from a reference view to the current view for similarity matching. The plane hypothesis which gives the maximum similarity measure is recorded and used to compute the ray depth for the corresponding pixel. The algorithm is implemented based on [12], and we will discuss the details as follows.

We define a set of plane hypotheses $\{\mathbf{n}_1, d_1\}$,..., $\{\mathbf{n}_m, d_m\}$, ...,$\{\mathbf{n}_M, d_M\}$ where $\mathbf{n}_m$ and d_m are the normal and depth respectively of the m^{th} plane in $\underrightarrow{\mathcal{F}}_{C_1}$. For each high-gradient pixel in the keyframe image $\mathbf{I}^r_{C_1}$, we evaluate each plane hypothesis by computing the corresponding homography $\mathbf{H}_{C_2C_1}$ and using it to warp an image patch from $\mathbf{I}^r_{C_1}$ to $\mathbf{I}^r_{C_2}$ such that the warped 7x7 image patch in $\mathbf{I}^r_{C_1}$ is centered on the selected pixel point:

$$\mathbf{H}_{C_2C_1} = \mathbf{R}_{C_2C_1} - \frac{\mathbf{t}_{C_2C_1}\mathbf{n}_m{}^T}{d_m}. \tag{14}$$

For the plane hypotheses, we use all possible permutations drawn from 64 depth values over the range $[0.5\ 30]$ m with a constant disparity step size, and fronto-parallel and ground plane orientations. In total, 128 plane hypotheses are generated.

We compute the similarity score based on zero-mean normalized cross-correlation (ZNCC) between the image patch in $\mathbf{I}^r_{C_1}$ centered on the selected high-gradient pixel point, and warped image patch from $\mathbf{I}^r_{C_2}$. For a given plane hypothesis $\{\mathbf{n}_i, d_i\}$, the depth $d_\mathbf{u}$ of the selected pixel $\mathbf{u}$ in $\mathbf{I}^r_{C_1}$ can be further computed as:

$$d_\mathbf{u} = -\frac{d_i}{\mathbf{f}_\mathbf{u} \cdot \mathbf{n}_i}. \tag{15}$$

Stereo matching usually has ambiguities due to repetitive textures, occlusions, and so on. Experimentally, we find the estimated depth map using a winner-takes-all approach to be rather noisy. To better capture the uncertainty characteristics of the estimated depth, we propose to keep track of multiple depth hypotheses for each pixel. Since the procedure to get these depth hypotheses are the same for all pixels, we illustrate the concept for one pixel. For each pixel, we accumulate a 1D (ρ_i, S_i) volume, where ρ_i is the inverse depth and S_i is its corresponding ZNCC score. We select only the elements in the volume such that their ZNCC scores S_i are larger than a pre-defined threshold. We use a threshold value of 0.85 in our experiments; a ZNCC score of 1.0 is considered perfect. All the selected elements from this volume are then clustered according to ρ_i. In particular, we sort the elements according to ρ_i. We separate two neighboring elements ρ_i and ρ_j into two different clusters if their inverse ray depth difference $|\rho_i - \rho_j|$ is larger than a pre-defined separation distance. We use a cluster separation distance of 0.1. We fit a Gaussian model to each cluster as

$$\mu = \frac{\sum_i S_i \cdot \rho_i}{\sum_i S_i}, \tag{16}$$

$$\sigma^2 = \frac{\sum_i S_i \cdot (\rho_i - \mu)^2}{\sum_i S_i}. \tag{17}$$

Furthermore, we choose the best score among all S_i belonging to the current cluster as the representative score for the current cluster. The cluster with the best score for each pixel is used for direct image alignment. All clusters for each pixel are tracked for temporal motion stereo so that depth ambiguities can be eliminated and the ray depth can be refined iteratively.

C. Temporal Motion Stereo

We use temporal motion stereo to refine the estimated depth and to eliminate depth ambiguity in the case that there is more than one inverse depth cluster for each pixel. Due to the epipolar curve issue for fisheye images mentioned earlier, we cannot use the conventional epipolar line disparity search method for stereo matching. Instead, we propose a technique to refine the mean depth estimate for each ray segment corresponding to a cluster, which is bounded by a two-sigma distance from the mean inverse depth of that cluster. Since this technique is applied in the same way to each cluster and each pixel, we use one cluster to explain the concept.

Each cluster or each depth hypothesis is parameterized by five parameters: its mean inverse ray depth μ_ρ, the variance of its inverse ray depth σ^2_ρ, its best matching score S_{best}, a good matching count $nInliers$, and a bad matching count $nOutliers$. μ_ρ, σ^2_ρ and S_{best} are initialized from plane-sweeping stereo. Both $nInliers$ and $nOutliers$ are initialized to be 2 in our experiment to avoid the zero division problem when we compute either the inlier ratio or the outlier ratio. Based on the initial μ_ρ and σ^2_ρ, we assume that the correct inverse ray depth would lie in the interval $[\mu_\rho - 2\sigma_\rho, \mu_\rho + 2\sigma_\rho]$, which has a probability of 95.45% statistically. To get subpixel projection matching accuracy, we subdivide this interval iteratively until the pixel distance between two neighboring projected pixels falls below 1 pixel. For all the sampled inverse ray depths within this interval, we compute their ZNCC matching scores between the reference image and the current image in the same way we compute the scores in plane-sweeping stereo. We use the fronto-parallel and ground plane orientations and current sampled inverse ray depth to compute the homography matrices. Using these homography matrices, we then compute the ZNCC scores for a 7×7 patch with the current refined pixel centered at the patch. As in plane-sweeping stereo, each cluster will then accumulate its own local volume. If this volume is not empty, we fit a new Gaussian model for it and fuse it with the original model:

$$\mu_{refined} = \frac{\frac{\mu_{prev}}{\sigma^2_{prev}} + \frac{\mu_{cur}}{\sigma^2_{cur}}}{\frac{1}{\sigma^2_{prev}} + \frac{1}{\sigma^2_{cur}}}, \tag{18}$$

$$\sigma^2_{refined} = \frac{1}{\frac{1}{\sigma^2_{prev}} + \frac{1}{\sigma^2_{cur}}}. \tag{19}$$

If no good matching (i.e., $S_{best} < 0.85$) is found, then we increment $nOutliers$ by 1. Otherwise, we increment $nInliers$ by 1. If the outlier ratio (i.e., $\frac{nOutliers}{nInliers+nOutliers}$) is larger than a pre-defined threshold, we label the current cluster or depth hypothesis as an outlier and stop tracking it. If only one cluster remains after several motion stereo

refinement iterations, and its corresponding variance falls below a certain threshold, we mark the current hypothesis as the true depth estimate and further mark this pixel's depth as converged. If all hypotheses are marked as outliers, we label this pixel as a bad pixel and consider its ray depth estimation to have diverged. Only pixels not labeled as either converged or diverged are refined in the next iteration.

VI. EXPERIMENTS AND RESULTS

We evaluate our direct fisheye-stereo visual odometry pipeline using real world datasets. In addition, we compare our direct fisheye-stereo visual odometry pipeline against the semi-direct fisheye-stereo visual odometry pipeline of Heng and Choi [13]. The data was collected on a ground vehicle as shown in Fig. 4. Datasets from two different environments are used to evaluate our pipeline: a vegetated environment and an urban environment. Fig. 2 and Fig. 3 show sample images captured on our ground vehicle in both environments. We describe our testbed platform in Section VI-A and show and discuss experimental results in Section VI-B. For ground truth data, we use the post-processed pose estimates from a GPS/INS system, which according to manufacturer specifications, has a position error of 2 cm in the absence of GPS outages.

Fig. 2: A view of the vegetated environment.

Fig. 3: A view of the urban environment.

A. Testbed Platform

We have two fisheye-stereo cameras with a 50 cm baseline on our vehicle platform. Fig. 4 shows the locations of the fisheye-stereo cameras marked by red ellipses. The left and right fisheye-stereo cameras look 45° to the left and right respectively. All cameras output 1280×960 color images at 30 frames per second, and are hardware-time-synchronized with the GPS/INS system.

We calibrate the multi-camera system using a grid of AprilTag markers, and use hand-eye calibration to compute the transformation between the reference frames of the multi-camera system and the INS. This transformation allows direct comparison of visual odometry pose estimates with the post-processed GPS/INS pose estimates which are used as ground truth.

B. Experiments

To achieve real-time performance, we use downsampled 640×480 images from the left fisheye-stereo camera as input to our direct visual odometry pipeline. The vehicle was driven with a speed range of 10-15 km/h for both datasets.

Fig. 4: DSO's Isuzu D-Max platform equipped with two fisheye-stereo cameras which are shown enclosed in red ellipses.

The trajectory lengths are 289 m and 199 m for the vegetated and urban datasets respectively. Fig. 5 and Fig. 8 show the post-processed pose estimates from the GPS/INS system, our visual odometry implementation, and a state-of-the-art semi-direct visual odometry implementation from Heng and Choi [13] in red, green and blue respectively. A black circle marks the starting point, while red, green and blue circles mark the end of the trajectories. Fig. 6 and Fig. 9 plot the absolute $x-y$ position errors against traveled distance for the vegetated and urban environments respectively. Similarly, Fig. 7 and Fig. 10 plot the absolute yaw errors for the vegetated and urban environments. We compute the position error at a given time by computing the norm of the difference between the $x-y$ components of the pose estimates from visual odometry implementations and the GPS/INS system. We compute the yaw error at a given time by computing the absolute difference between the estimated yaw from visual odometry and the GPS/INS system at that time.

	xy-drift	yaw-drift
VO-vegetated-ours	**0.86**%	**0.013** deg/m
VO-vegetated-Heng	0.96%	0.0138 deg/m
VO-urban-ours	**0.6**%	**0.0063** deg/m
VO-urban-Heng	1.6%	0.011 deg/m

TABLE I: Pose accuracy comparison between our direct visual odometry implementation and [13]

We compare the accuracy of our direct visual odometry implementation against that of [13] using position and orientation drift metrics. We compute the $x-y$ and yaw errors averaged over all possible subsequences of length $\{100, 200\}$ meters. Table I shows that our direct visual odometry implementation outperforms Heng and Choi [13] for both the vegetated and urban environments. In addition, we are able to generate a semi-dense point cloud that can be used for dense mapping. Furthermore, Fig. 11 shows the reconstructed and colored point cloud generated by our visual odometry pipeline. The point cloud only includes points which are labeled as converged by our motion stereo. To

better show the reconstruction quality of our pipeline, we choose different viewpoints for purposes of comparison. A supplementary video showing the experimental results can be found at `https://youtu.be/NOiIVO0jzuc`.

Our pipeline is implemented and evaluated on a PC with an Intel i7 CPU @ 2.9 GHz and a low-end Nvidia GTX 480 GPU. We run our tracking thread at 20 Hz on one CPU core. Both plane sweeping stereo and motion stereo run on the GPU at around 5 Hz.

One possible reason that our algorithm performs better than that of [13] could be the use of more information from observed images. Our algorithm uses most of the high gradient pixels (>20k) while [13] only samples several hundred sparse feature points. The use of more information increases the robustness of the whole algorithm, especially in poorly textured environments where [13] does not perform well as shown in Fig. 10.

VII. CONCLUSIONS

We present a direct visual odometry algorithm for a fisheye stereo camera in this paper. Our algorithm uses fisheye images directly instead of converting them to conventional pinhole images, and in turn, reducing the field of view. Plane-sweeping stereo is used to do stereo matching for fisheye images which cannot be processed by traditional epipolar line disparity search algorithms. The estimated depths are further refined via temporal stereo. By using the refined semi-dense depth map, a direct image alignment tracker is used to estimate the current camera pose in real-time. Experimental results show that our pipeline not only achieves significantly accurate motion estimates but also outputs a high-quality point cloud at the same time. In other words, our pipeline can facilitate real-time localization and dense mapping for mobile robots.

REFERENCES

[1] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004.

[2] J. Baretto and H. Araujo. Issues on the geometry of central catadioptic image formation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[3] D. Caruso, J. Engel, and D. Cremers. Large-scale direct slam for omnidirectional cameras. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[4] B. Clipp, J. Lim, J.-M. Frahm, and M. Pollefeys. Parallel, Real-Time Visual SLAM. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[5] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[6] J. Engel, J. Stückler, and D. Cremers. Large-scale direct slam with stereo cameras. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[7] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2017.

[8] C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[9] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. SVO: Semi-direct visual odometry for monocular and multi-camera systems. *IEEE Transactions on Robotics (T-RO)*, PP(99):1–17, 2016.

[10] F. Fraundorfer and D. Scaramuzza. Visual odometry: Part ii - matching, robustness, and applications. *IEEE Robotics And Automation Magazine (RAM)*, 19(2):78–90, 2012.

[11] C. Geyer and K. Daniilidis. A unifying theory for central panaromic systems and practical implications. In *European Conference on Computer Vision (ECCV)*, 2000.

[12] C. Häne, L. Heng, G. H. Lee, A. Sizov, and M. Pollefeys. Real-time direct dense matching on fisheye images using plane-sweeping stereo. In *International Conference on 3D Vision (3DV)*, 2015.

[13] L. Heng and B. Choi. Semi-direct visual odometry for a fisheye-stereo camera. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[14] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *International Symposium on Mixed and Augmented Reality*, 2007.

[15] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. T. Furgale. Keyframe-based visualinertial odometry using nonlinear optimization. *International Journal of Robotics Research*, 2015.

[16] J. Lim, J.-M. Frahm, and M. Pollefeys. Online Environment Mapping. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[17] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[18] S. Omari, M. Blösch, P. Gohl, and R. Siegwart. Dense visual-inertial navigation system for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[19] D. Scaramuzza and F. Fraundorfer. Visual odometry: Part i - the first 30 years and fundamentals. *IEEE Robotics And Automation Magazine (RAM)*, 18(4):80–92, 2011.

[20] P. Tanskanen, T. Ngeli, M. Pollefeys, and O. Hilliges. Semi-Direct EKF-based Monocular Visual-Inertial Odometry. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

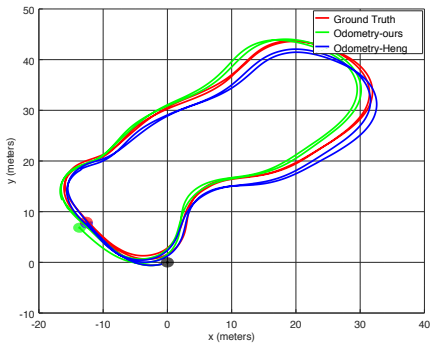

Fig. 5: Red, green, and blue lines represent the trajectories estimated by the GPS/INS system, our implementation, and [13] respectively in the vegetated environment.

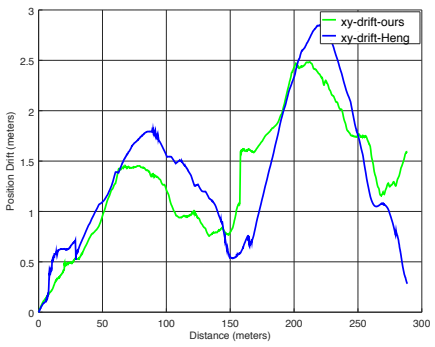

Fig. 6: Red and green lines represent the $x-y$ position drift of the pose estimates output by our implementation and [13] in the vegetated environment.

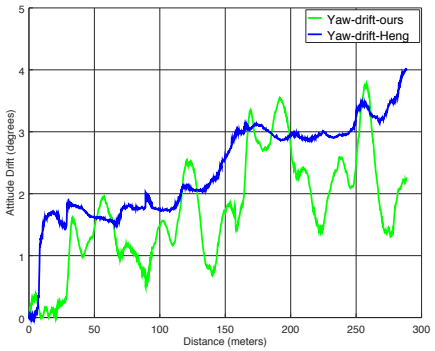

Fig. 7: Red and green lines represent the yaw drift of the pose estimates output by our implementation and [13] in the vegetated environment.

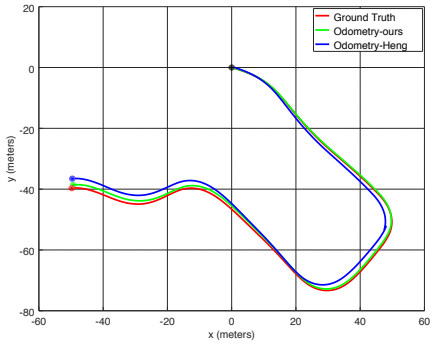

Fig. 8: Red, green, and blue lines represent the trajectories estimated by the GPS/INS system, our implementation, and [13] respectively in the urban environment.

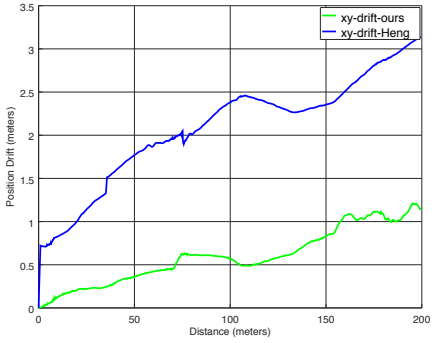

Fig. 9: Red and green lines represent the $x-y$ position drift of the pose estimates output by our implementation and [13] in the urban environment.

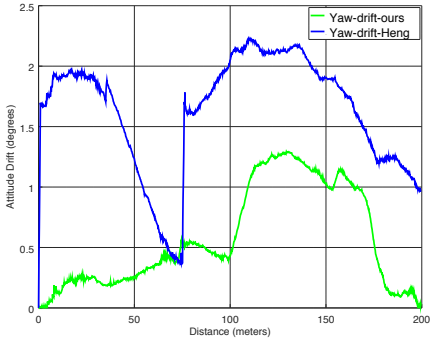

Fig. 10: Red and green lines represent the yaw drift of the pose estimates output by our implementation and [13] in the urban environment.

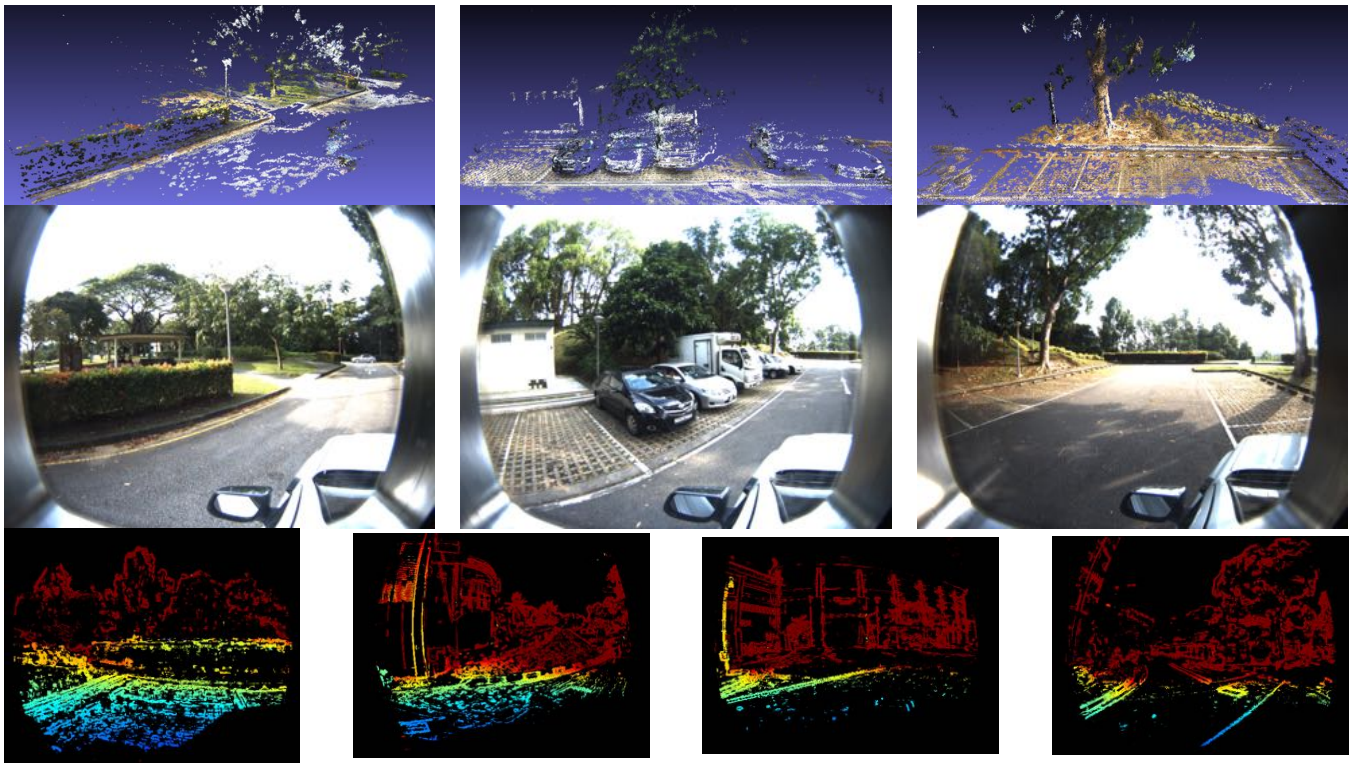

Fig. 11: Each subfigure shows a reconstructed point cloud from our VO pipeline and the corresponding image as well as estimated semi-dense depth maps.