Semi-Direct Visual Odometry for a Fisheye-Stereo Camera

Lionel Heng and Benjamin Choi

Abstract—We present a semi-direct visual odometry algorithm for a fisheye-stereo camera. In a tracking thread, we simultaneously track oriented patches and estimate the camera pose. In a mapping thread, we estimate the coordinates and surface normal for each new patch to be tracked. Estimation of the surface normals allows us to track patches over a wide variety of viewpoints. In our algorithm, we do not make use of descriptors and robust descriptor matching to find patch correspondences. Instead, we use photoconsistency-based techniques to find patch correspondences. For tracking, we use sparse model-based image alignment to find the relative motion estimate, and feature alignment to find 2D-3D patch correspondences. For mapping, we use plane-sweeping stereo to find matching patches between stereo images. We also implement a state estimator based on the Extended Kalman Filter (EKF) to fuse inertial measurements and relative pose estimates from our visual odometry implementation. We run experiments in two different outdoor environments to validate our algorithm, and discuss the experimental results. Our implementation runs at an average of 42 Hz on a commodity Intel CPU. To the best of our knowledge, there is no other existing semi-direct visual odometry algorithm for a fisheye-stereo camera.

I. INTRODUCTION

Semi-direct visual odometry methods [5, 7] have been gaining popularity over traditional feature-based visual odometry methods [18, 8] in recent years as they do not require the computationally expensive tasks of extracting descriptors and identifying outliers in a set of feature correspondences. They directly use pixel intensity values instead of hand-crafted high-dimensional feature descriptors. As such, semi-direct visual odometry methods are able to run at significantly higher frame rates, and thus, are well-suited for use on fast-moving vehicles. In contrast, direct visual odometry methods [16] attempt per-pixel depth estimation, and thus, require the use of a commodity GPU for real-time operation. Hence, their real-time operation is restricted to vehicle platforms equipped with GPUs.

Semi-direct visual odometry methods generally follow the simultaneous-tracking-and-mapping paradigm. However, in recent literature, we see two distinct approaches being taken by semi-direct visual odometry methods: pixel-based and patch-based. The pixel-based approach led by Engel et al. [5] leans towards direct visual odometry. Here, a semi-dense depth map is propagated from frame to frame, and updated 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 current frame is used to track the next frame via whole-image alignment. The patch-based approach led by Forster et al. [7] leans towards feature-based visual odometry. Here, a sparse map of patch-based landmarks is maintained instead of a semi-dense depth map. Depth information with respect to landmarks observed in the current frame is used to track the next frame via sparse-model-based image alignment. Feature correspondences are implicitly obtained through feature alignment while the pose estimate and estimated landmark coordinates are optimized in pose and structure refinement respectively. The second approach is inherently faster than the first approach due to the significantly smaller number of pixels for which depth values have to be estimated. Both approaches have been extensively and experimentally validated for monocular pinhole cameras. We now review variants these approaches. For more accurate and robust tracking, several variants make use of a stereo camera, IMU, or even both. Several other variants are targeted at fisheye cameras which have a wider field of view than pinhole cameras.

II. RELATED WORK

A. Monocular vs Stereo

Stereo visual odometry resolves scale ambiguity in monocular visual odometry by estimating metric depth from stereo images with a known baseline. Omari et al. [17] follow the semi-dense 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 semi-direct visual odometry method [5] by incorporating depth measurements from static stereo in addition to those from temporal stereo.

B. Visual vs Visual-Inertial

We can also resolve scale ambiguity inherent in monocular visual odometry by using inertial measurements. Blösch et al. [3] and Tanskanen et al. [20] follow in the footsteps of Forster et al. [7], and use a tightly-coupled filtering-based visual-inertial framework. They estimate the landmark coordinates as part of the filter state, and use intensity errors to update the filter state. However, long-term IMU bias drift can still occur in the absence of absolute depth measurements, and thus, cause scale drift in the long run. Usenko et al. [21] combine the complementary benefits of stereo and inertial measurements. They expand on the work of Engel et al. [6] by tightly integrating inertial measurements into tracking; whole-image alignment involves the minimization of both intensity and IMU errors as a single cost function.
C. 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 obfuscate a pinhole camera’s field of view, making tracking impossible. Semi-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. [4] do a disparity search along the epipolar curve. We can circumvent the problem of epipolar curves by using the naive step of extracting rectified pinhole images from fisheye images at the expense of a significant loss of field-of-view. Nevertheless, significant distortion exists at the periphery of the resulting rectified image which may impede stereo matching. In computer vision literature, plane-sweeping stereo [22, 12] has been extensively used for dense stereo matching without the need for rectification but requires a GPU.

III. CONTRIBUTIONS

The main contribution of this paper is a semi-direct visual odometry algorithm for a fisheye-stereo camera. To the best of our knowledge, no semi-direct visual odometry algorithm exists for a fisheye-stereo camera. Furthermore, there only exists one other visual odometry algorithm for a fisheye-stereo camera; Shen et al. [19] implement feature-based visual odometry algorithm for a fisheye-stereo camera. To the best of our knowledge, no other semi-direct patch-based visual odometry method exists one other visual odometry algorithm for a fisheye-stereo camera. To the best of our knowledge, no other semi-direct patch-based visual odometry method exists for rectification but requires a GPU.

IV. NOTATION

We briefly define the notation to be used throughout this paper.

We denote the world reference frame as $\mathcal{F}_w$, the stereo-camera reference frame as $\mathcal{F}_s$, and the individual camera reference frames as $\mathcal{F}_{C_i}$ and $\mathcal{F}_{C_2}$. Here, $C_1$ is the reference camera in the stereo camera, and thus, $\mathcal{F}_{C_1}$ coincides with $\mathcal{F}_s$. We denote the image from camera $C_i$ at time step $k$ as $\Pi_{C_i}^k$, and the stereo frame at time step $k$ as $F_s^k = \{\Pi_{C_1}^k, \Pi_{C_2}^k\}$. $\Pi_{C_i}^k(u)$ is the intensity of the pixel with image coordinates $u$.

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

$$u = \pi(c_i, p). \tag{1}$$

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

$$c_i, p = \pi^{-1}(u, d_u). \tag{2}$$

Otherwise, if the depth is not known, we recover the ray $f_u$ in $\mathcal{F}_{C_i}$ and passing through the landmark that corresponds to the image point $u$ in $\Pi_{C_i}$:

$$c_i, f_u = \pi^{-1}(u). \tag{3}$$

Each landmark has an associated surface normal $n$.

We denote the stereo camera pose at time step $k$ as a rigid body transformation $T_{c_{SW}}^k \in SE(3)$ from $\mathcal{F}_w$ to $\mathcal{F}_s$ and whose rotation matrix part is $R_{c_{SW}}^k$, corresponding quaternion part is $q_{c_{SW}}^k$, and translation part is $t_{c_{SW}}^k$. Similarly, we denote the pose of camera $C_i$ at time step $k$ with $T_{C_iW}^k = T_{C_iS}^k T_{c_{SW}}^k$. $T_{C_iS}$ is known from calibration. The rigid body transformation $T_{c_{SW}}^k$ maps a 3D point $w \in \mathcal{F}_w$ to a 3D point $c_i, p$ in $\mathcal{F}_{C_i}$:

$$c_i, p = T_{C_iW}^k w. \tag{4}$$

Given the fact that a rigid body transformation 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 $\xi = [v\ \omega]^T$ where $v$ is the linear velocity and $\omega$ is the angular velocity. We use the exponential
map to map the twist coordinates $\xi$ in $se(3)$ to a rigid body transformation $T$ in $SE(3)$:

$$T(\xi) = \exp(\xi). \quad (5)$$

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

$$\xi = \log(T(\xi)). \quad (6)$$

V. ALGORITHM

In this section, we describe our semi-direct visual odometry algorithm for a fisheye-stereo camera. Fig. 1 shows the pipeline of our implementation which encompasses a tracking thread and a mapping thread. The tracking thread tracks the stereo camera using a local map, and the mapping thread adds new landmarks to the local map. The local map is a graph with nodes representing keyframes and landmarks and edges representing pose-point constraints. In the map, we store the 10 most recent keyframes together with landmarks observed in these keyframes. The tracking thread only processes images from the reference camera in thread only processes images from the reference camera in with landmarks observed in these keyframes. The tracking thread adds new landmarks to the local map. The local tracking thread and a mapping thread. The tracking thread in the pipeline of our implementation which encompasses a

A. Sparse Model-based Image Alignment

Whole image alignment works well only if we sum the squared intensity errors over a sufficiently large number of pixels. However, we only know the depth for a sparse set of feature points. Instead of summing over the feature points, we sum over 5x5 image patches $P(\mathbf{u})$ centered at the feature points $\mathbf{u}$. We then find the motion estimate $T_{C_1}^{k,k-1}$ that minimizes the sum of squared intensity errors over all the image patches:

$$\sum_{\mathbf{u}} \sum_{\mathbf{u}' \in P_{C_1}^{k-1}(\mathbf{u})} w(\delta I(T_{C_1}^{k,k-1}, \mathbf{u}')) \delta I(T_{C_1}^{k,k-1}, \mathbf{u}')^2 \quad (7)$$

where

$$\delta I(T_{C_1}^{k,k-1}, \mathbf{u}) = I_{C_1}^k(\pi(T_{C_1}^{k,k-1} \cdot \pi^{-1}(\mathbf{u}, d_\mathbf{u}))) - I_{C_1}^{k-1}(\mathbf{u}) \quad (8)$$

and $w : \mathbb{R} \rightarrow \mathbb{R}$ is a robust weighting function that reduces the influence of outliers from incorrect depth estimates and moving objects on the sum of squared intensity residuals. We use the $t$-distribution [13] for the robust weighting function.

To solve this non-linear least squares problem, we use the iteratively reweighted Gauss-Newton algorithm coupled with a coarse-to-fine approach using several pyramid levels. We use the inverse compositional formulation [11] of the intensity residual:

$$\delta I(\xi, \mathbf{u}) = I_{C_1}^k(\pi(T_{C_1}^{k,k-1} \cdot \pi^{-1}(\mathbf{u}, d_\mathbf{u}))) - I_{C_1}^{k-1}(\mathbf{u}) \quad (9)$$

where $T_{C_1}^{k,k-1}$ is the motion estimate from the previous iteration and $T(\xi)$ is an incremental update to the estimate that is parameterized with a twist $\xi \in se(3)$:

$$T_{C_1}^{k,k-1} = T_{C_1}^{k,k-1} \cdot T(\xi). \quad (10)$$

We use the constant velocity model to compute an initial motion estimate which is fed to the first iteration of the Gauss-Newton algorithm applied to the top pyramid level. In the Gauss-Newton algorithm, we solve the equation:

$$(J^T J) \xi = -J^T \delta I(0, \mathbf{u}). \quad (11)$$

We compute the Jacobian $J$ using the chain rule:

$$J = \frac{\partial \delta I(\xi, \mathbf{u})}{\partial \xi} |_{\xi=0} = -\frac{\partial I_{C_1}^{k-1}(\mathbf{a})}{\partial \mathbf{a}} \bigg|_{\mathbf{a} = \mathbf{u}} \cdot \frac{\partial \pi(T(\xi) \cdot \pi^{-1}(\mathbf{u}, d_\mathbf{u}))}{\partial \xi} |_{\xi=0} \quad (12)$$

The first Jacobian component is the image gradient evaluated at $\mathbf{u}$ while the second Jacobian component is the closed-form image Jacobian that relates the twist to the image velocity of the feature point. The inverse compositional formulation is a computational trick that makes evaluation of both the intensity residual and Jacobian much faster.

B. Feature Alignment

The sparse model-based image alignment algorithm described in the previous section finds an initial value for the motion estimate between the previous image $I_{C_1}^{k-1}$ and the current image $I_{C_1}^k$. We can reduce drift if we estimate the current pose $T_{C_1}^k W_1$ of camera $C_1$ with respect to a local map rather than the previous frame. This reason is the basis for both the feature alignment step, and pose and structure refinement step.

In the feature alignment step, we implicitly establish feature correspondences between $I_{C_1}^k$ and the local map. For each landmark in the local map and which is visible from the current camera pose $T_{C_1}^k W_1$, we reproject the landmark into $I_{C_1}^k$. Assume without loss of generality that the reprojected landmark has image coordinates $\mathbf{u}$ in $I_{C_1}^k$. Because the motion estimate from the sparse model-based image alignment algorithm may have errors, $\mathbf{u}$ may differ from the actual value. We identify the keyframe $I_{C_1}^k$ in the local map and that observes the landmark with the closest observation angle. Then, we optimize $\mathbf{u}$ by minimizing the

---

Fig. 1: Our semi-direct fisheye-stereo visual odometry pipeline.
photometric error between the patch centered at \( \mathbf{u} \) in \( \mathbf{I}_{C_1}^k \) and the reference patch in \( \mathbf{I}_{C_1}^k \), that is warped from \( \mathbf{I}_{R}^k \).

To obtain the reference patch, we use a homography \( \mathbf{H}^{r,k} \) to warp the image patch centered on the landmark reprojection in \( \mathbf{I}_{C_1}^k \) to \( \mathbf{I}_{C_1}^k \). Given that the landmark lies on a plane with distance \( d \) from the current camera pose \( \mathbf{T}_{C_1}^{k} \) and with the normal vector \( \mathbf{n} \) which is equal to the surface normal associated with that landmark, we derive:

\[
\begin{align*}
    d &= d_u(f_u, -\mathbf{R}_{C_1,W}^k \mathbf{n}) \\
    \mathbf{H}^{r,k} &= \mathbf{R}_{C_1}^k \frac{1}{d} t_{C_1}^{r,k}(\mathbf{R}_{C_1,W}^k \mathbf{n})^T
\end{align*}
\]  

(13)

We use a patch size of 5x5. Initially setting \( \mathbf{u}' = \mathbf{u} \), we then optimize \( \mathbf{u}' \) by minimizing the sum of squared intensity errors over the patch:

\[
\mathbf{u}^* = \arg \min_{\mathbf{u}'} \frac{1}{2} \sum_{\mathbf{u}'} (\mathbf{I}_{C_1}^k(\mathbf{u}') - \mathbf{I}_{C_1}^k(\pi(\mathbf{H}^{r,k} \cdot \pi^{-1}(\mathbf{u}))))^2.
\]

(14)

C. Pose and Structure Refinement

In this step, we optimize the pose \( \mathbf{T}_{SW}^k \) of the stereo camera at current time step \( k \), and in turn, the coordinates of the landmarks observed in the current stereo frame at current time step \( k \) and comprising images \( \mathbf{I}_{C_1}^k \) and \( \mathbf{I}_{C_2}^k \).

We solve a least-squares problem which involves minimizing a cost function equivalent to the sum of weighted squared reprojection errors:

\[
\sum_{i=1}^{2} \sum_{j \in J(i,k)} \rho(||\mathbf{e}_{i,j}^k||_2) = \sum_{i=1}^{2} \sum_{j \in J(i,k)} \rho(||\mathbf{e}_{i,j}^k||_2)
\]

where

\[
\mathbf{e}_{i,j}^k = \mathbf{u}_{i,j}^k - \pi(\mathbf{T}_{C_1} \mathbf{T}_{SW} \mathbf{p}_j).
\]

(16)

\( i \) is the camera index of the stereo camera, \( j \) is the landmark index, and the set \( J(i,k) \) denotes the indices of landmarks visible in the \( k \)th frame in \( C_i \). \( \rho \) is a loss function that reduces the influence of outliers on the solution to the least-squares problem; we use the Huber loss function. \( \mathbf{w}_j \) is the coordinates of the \( j \)th landmark in \( \mathbf{F}_{SW} \), and \( \mathbf{u}_{i,j}^k \) is the measured image coordinates of the \( j \)th landmark observed in \( \mathbf{I}_{C_1}^k \).

The pose refinement finds the best value of \( \mathbf{T}_{SW}^k \) that minimizes the cost function. The structure refinement finds the best value of \( \mathbf{w}_j \) for each \( j \in J(i,k) \) that minimizes the cost function.

We mark the current stereo frame \( F_{SW}^k \) as a keyframe if the sum of Euclidean and angular distances between the current camera pose and the camera pose corresponding to the current keyframe exceeds a threshold. In the event \( F_{SW}^k \) is marked as a keyframe, we pass \( F_{SW}^k \) to the mapping thread which finds the depth of new landmarks observed in the image regions that do not contain observations of existing landmarks in the local map.

D. Feature Detection

We use the AGAST corner detector [15] to detect candidate feature points in the image \( \mathbf{I}_{C_1}^k \). Subsequently, we form a 2D grid over the image \( \mathbf{I}_{C_1}^k \), and reproject the landmarks in the local map into the 2D grid. We mark the grid cells containing reprojected landmarks as occupied. If each grid cell that is unoccupied, we select the feature point with the highest Shi-Tomasi score. We use the plane-sweeping stereo algorithm described in the next section to find the depth of all selected feature points, and subsequently, the coordinates of the corresponding landmarks.

E. Plane-Sweeping Stereo

The plane-sweeping stereo algorithm is based on [12] which attempts to estimate the depth for every pixel in an image, and thus, requires a GPU. We achieve real-time performance on a CPU by running the plane-sweeping stereo algorithm for a sparse set of feature points that is selected in the previous feature detection step.

We define a set of plane hypotheses \( \{\mathbf{n}_1, d_1\}, \ldots, \{\mathbf{n}_m, d_m\}, \ldots, \{\mathbf{n}_M, d_M\} \) where \( \mathbf{n}_i \) and \( d_i \) are the normal and depth of the \( m \)th plane in \( \mathbf{F}_{C_1} \). For each feature point in the image \( \mathbf{I}_{C_1}^k \), we evaluate each plane hypothesis by computing the corresponding homography \( \mathbf{H}_{C_2,C_1} \) and using it to warp an image patch from \( \mathbf{I}_{C_2}^k \) to \( \mathbf{I}_{C_1}^k \) such that the warped 5x5 image patch in \( \mathbf{I}_{C_1}^k \) is centered on the feature point:

\[
\mathbf{H}_{C_2,C_1} = \mathbf{R}_{C_2,C_1} - \frac{1}{d_m}(\mathbf{R}_{C_2,C_1} \mathbf{t}_{C_1} \mathbf{s}) \mathbf{n}_m^T.
\]

(17)

For the plane hypotheses, we use all possible permutations drawn from 128 depth values over the range \([0.5, 30]\) m with constant disparity step size, azimuthal angles over the range \( [-\frac{\pi}{4}, \frac{\pi}{4}] \) rad, and polar angles over the range \([0, \pi]\) rad with step size \( \frac{\pi}{16} \) rad.

We compute the similarity score based on zero-mean normalized cross-correlation between the image patch in \( \mathbf{I}_{C_1}^k \) and centered on the feature point, and warped image patch from \( \mathbf{I}_{C_2}^k \). We choose the best plane hypothesis \( \{\mathbf{n}, d\} \) that has the highest similarity score provided that this score exceeds a predefined threshold. We then use this hypothesis together with subpixel interpolation to compute the depth \( d_u \) of the landmark corresponding to the feature point \( \mathbf{u} \) in \( \mathbf{I}_{C_1}^k \):

\[
d_u = -\frac{d}{f_m} \mathbf{n}.
\]

(18)

We also set the surface normal of the landmark equal to the normal of the best plane hypothesis. We further optimize the coordinates of the landmark by applying the feature alignment algorithm described in Section V-B to optimize the image coordinates of the feature point in \( \mathbf{I}_{C_2}^k \) that corresponds to that in \( \mathbf{I}_{C_1}^k \).

VI. EXPERIMENTS AND RESULTS

We test our semi-direct fisheye-stereo visual odometry implementation in two places: a parking lot with a paved surface, and an off-road environment with rocky terrain.
Fig. 2: The top row shows a stereo frame captured in the parking lot. The bottom row shows a stereo frame captured in the off-road environment. The left column shows images from the reference camera in the stereo camera together with feature tracks. 

Fig. 2 shows stereo frames captured in both environments. The left column in this figure shows feature tracks in images from the reference camera in the stereo camera. The shape of the feature tracks is a visual indicator of the terrain roughness; smooth feature tracks indicate smooth terrain while oscillating feature tracks indicate rough terrain. At the same time, we run our loosely-coupled visual-inertial odometry implementation.

We compare both implementations against a state-of-the-art stereo visual odometry implementation: LIBVISO2 [9]. As LIBVISO2 is designed to work with pinhole-stereo cameras, we rectify images from our fisheye-stereo camera such that the resulting pinhole images correspond to a pinhole camera with horizontal and vertical fields of view of $85^\circ$ and $69^\circ$ respectively, and input these pinhole images to the LIBVISO2 implementation. Fig. 3 shows example fisheye images and the corresponding rectified pinhole images.

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 pose estimates from a GPS/INS system, which according to manufacturer specifications, has a position error of 1.2 m in the absence of GPS outages.

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 $45^\circ$ to the left and right respectively. All cameras output 1280x960 color images at 30 fps, and are hardware-time-synchronized with a 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 GPS/INS pose estimates which are used as ground truth.

B. Experiments

For all experiments, we use downsampled 640x480 images from the right fisheye-stereo camera as input to our semi-direct visual odometry implementation and loosely-coupled visual-inertial odometry implementation. For the LIBVISO2 implementation which we compare our implementation to, we use downsampled and rectified 640x480 images from the right fisheye-stereo camera as input. To quantify the performance of the odometry implementations, we use the metrics in [10]. These metrics correspond to the translational and rotational errors averaged over all possible subsequences of length $\{100, \ldots, 800\}$ meters.

We ran the first experiment in a parking lot with a paved surface. We drove the vehicle along a 408 m trajectory and
TABLE I: Translational and Rotational Errors (Parking Lot)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Translational Error (%)</th>
<th>Rotational Error (deg/m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Visual Odometry</td>
<td>0.7950</td>
<td>0.008538</td>
</tr>
<tr>
<td>Our Visual-Inertial Odometry</td>
<td>0.6670</td>
<td>0.003006</td>
</tr>
<tr>
<td>LIBVISO2</td>
<td>3.415</td>
<td>0.06612</td>
</tr>
</tbody>
</table>

with a speed range of 10-15 km/h. Fig. 5 shows the pose estimates from the GPS/INS system, our standalone visual odometry implementation, our loosely-coupled visual-inertial odometry implementation, and the LIBVISO2 implementation in green, blue, red, and magenta respectively. A black circle marks the starting point, while green, blue, red, and magenta circles mark the end of the trajectories estimated by the GPS/INS system, the visual odometry implementation, the loosely-coupled visual-inertial odometry implementation, and the LIBVISO2 implementation. In addition, we tabulate the translational and rotational errors for each implementation in Table I.

Fig. 6 and Fig. 7 plot the position and attitude drifts respectively over distance. We compute the position drift at a given time by computing the norm of the difference between the position-components of the pose estimates from visual(-inertial) odometry and the GPS/INS system at that time. We compute the attitude drift at a given time by computing the norm of the difference between the roll-pitch-yaw components of the pose estimates from visual(-inertial) odometry and the GPS/INS system at that time.

Table I shows that our loosely-coupled visual-inertial odometry implementation achieves the lowest translational and rotational errors followed by our standalone visual odometry implementation and the LIBVISO2 implementation in order. We observe from Fig. 7 that fusion of inertial measurements greatly reduces attitude drift due to the roll and pitch being made observable through inertial measurements.

![Fig. 5: The vehicle is driven along a 408 m trajectory in a parking lot. Green, blue, red, and magenta lines represent the trajectories estimated by the GPS/INS system, visual odometry implementation, loosely-coupled visual-inertial odometry implementation, and LIBVISO2 implementation respectively.](image1)

![Fig. 6: Blue, red, and magenta lines represent the position drift of the pose estimates output by our visual odometry, loosely-coupled visual-inertial odometry, and LIBVISO2 implementations in the parking lot.](image2)

![Fig. 7: Blue, red, and magenta lines represent the attitude drift of the pose estimates output by our visual odometry, loosely-coupled visual-inertial odometry, and LIBVISO2 implementations in the parking lot.](image3)

We ran the second experiment in an off-road environment with rocky terrain. We consider this environment to be more challenging than the parking lot; the rocky terrain induces significant vehicle roll and pitch motions whereas vehicle roll and pitch are almost constant on the smooth planar ground in the parking lot. We drove the vehicle along a 3.89 km trajectory, and up to a maximum speed of 35 km/h. Fig. 8 shows the pose estimates from the GPS/INS system, our standalone visual odometry implementation, our loosely-coupled visual-inertial odometry implementation, and the LIBVISO2 implementation in green, blue, red, and magenta respectively. We tabulate the translational and rotational errors for each implementation in Table II. Fig. 9 and Fig. 10 plot the position and attitude drifts respectively over distance. The attitude drift associated with the LIBVISO2 implementation decreases after the 2800-meter mark due to the fact that the
TABLE II: Translational and Rotational Errors (Off-Road Environment)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Translational Error (%)</th>
<th>Rotational Error (deg/m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Visual Odometry</td>
<td>3.312</td>
<td>0.02538</td>
</tr>
<tr>
<td>Our Visual-Inertial Odometry</td>
<td>1.484</td>
<td>0.001878</td>
</tr>
<tr>
<td>LIBVISO2</td>
<td>8.152</td>
<td>0.08063</td>
</tr>
</tbody>
</table>

yaw drift exceeds 360° at that point and an angle wraparound occurs as a result.

From the table and three figures, we observe that our loosely-coupled visual-inertial odometry implementation outperforms our standalone visual odometry implementation, which in turn, outperforms the LIBVISO2 implementation. We also observe that the benefits of visual-inertial fusion are more pronounced over a longer trajectory.

Fig. 8: The vehicle is driven along a 3.89 km trajectory in an off-road environment. Green, blue, red, and magenta lines represent the trajectories estimated by the GPS/INS system, visual odometry implementation, loosely-coupled visual-inertial odometry implementation, and LIBVISO2 implementation respectively.

Fig. 9: Blue, red, and magenta lines represent the position drift of the pose estimates output by our visual odometry, loosely-coupled visual-inertial odometry, and LIBVISO2 implementations in the off-road environment.

Fig. 10: Blue, red, and magenta lines represent the attitude drift of the pose estimates output by our visual odometry, loosely-coupled visual-inertial odometry, and LIBVISO2 implementations in the off-road environment.

Fig. 11 shows a box plot of the time taken by each iteration of the tracking and mapping threads on a Windows machine (Intel i7, 2.90 GHz) during the run in the off-road environment. The average time taken by each tracking iteration is 24 ms while the average time taken by each mapping iteration is 22 ms. The timing data indicates that our semi-direct fisheye-stereo visual odometry implementation is capable of running at an average of 42 Hz, and is able to process every frame from our fish-eye stereo camera which outputs stereo image pairs at 30 Hz. Thus, our implementation is ideal for use on fast-moving vehicles.

VII. CONCLUSIONS

Our novel semi-direct visual odometry algorithm for a fisheye-stereo camera has been experimentally shown to work over large distances in challenging environments, and
outperform an existing state-of-the-art stereo visual odometry implementation for pinhole-stereo cameras. Furthermore, experimental results demonstrate the significantly higher accuracy of pose estimates from our loosely-coupled visual-inertial odometry implementation which involves the use of a EKF-based estimator to fuse inertial measurements and relative pose measurements from our visual odometry algorithm.

We note that the surface normal is only estimated once and coarsely for each new landmark during mapping. Hence, future work includes iterative refinement of the surface normals. Ongoing work on tightly-coupled semi-direct visual-inertial odometry for a fisheye-stereo camera will lead to more accurate visual pose estimates.

REFERENCES


In European Conference on Computer Vision (ECCV), 2000.


