2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

Towards Robust Visual Odometry with a Multi-Camera System

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

Abstract— **We present a visual odometry (VO) algorithm for a multi-camera system and robust operation in challenging environments. Our algorithm consists of a pose tracker and a local mapper. The tracker estimates the current pose by minimizing photometric errors between the most recent keyframe and the current frame. The mapper initializes the depths of all sampled feature points using plane-sweeping stereo. To reduce pose drift, a sliding window optimizer is used to refine poses and structure jointly. Our formulation is flexible enough to support an arbitrary number of stereo cameras. We evaluate our algorithm thoroughly on five datasets. The datasets were captured in different conditions: daytime, night-time with near-infrared (NIR) illumination and night-time without NIR illumination. Experimental results show that a multi-camera setup makes the VO more robust to challenging environments, especially night-time conditions, in which a single stereo configuration fails easily due to the lack of features.**

I. Introduction

Visual odometry (VO) is a technique used to estimate camera motion from images. As a fundamental block for robot navigation, virtual reality, augmented reality, and so on, VO has made significant progress over the past few decades. Existing approaches range across conventional feature-based methods [15], hybrid methods [8], and direct methods [7].

To make VO useful for real-world applications, we need to make VO work robustly under various conditions. For example, to serve as a fundamental module for autonomous driving, robust vehicle motion estimation is critical for path/trajectory tracking, and environment perception modules. Motion estimation errors and failures would cause tragic accidents, and limit wide-scale deployment of VO.

Several approaches have been proposed to improve the robustness of VO for specific environments. Alismail et al. [1] propose a dense binary descriptor that can be integrated within a multi-channel Lucas Kanade framework to improve illumination change robustness. Park et al. [21] perform a systematic evaluation of real-time capable methods for illumination change robustness in direct visual SLAM. Zhang et al. [26] propose an active exposure control method for robust visual odometry in high dynamic range (HDR) environments. For each frame, they choose an exposure time that maximizes an image quality metric. In [22], a direct monocular SLAM algorithm based on the Normalized Information Distance (NID) metric is proposed. They show that the information-theoretic NID metric provides robustness to appearance variations due to lighting, weather, and structural changes in the scene.

In this paper, we propose to improve the robustness of VO by using a multi-camera system. A multi-camera system can cover a wide field-of-view and thus provide redundancy for poorly textured environments. Our experimental evaluation demonstrates that VO with a multi-camera system can still estimate vehicle motion reliably in dark environments at night; in contrast, a single stereo configuration is prone to failure due to the lack of texture.

Our algorithm consists of a pose tracker and a local mapper. The tracker estimates the current pose by minimizing photometric errors between the most recent keyframe and the current frame across multiple cameras. The local mapper consists of a sliding window optimizer and a two-view stereo matcher. Both vehicle poses and structure are jointly optimized by the sliding window optimizer. This optimization minimizes long-term pose drift. The depths of newly sampled feature points are initialized by the two-view stereo matcher. We thoroughly evaluate our algorithm under multiple challenging conditions. In particular, we select five different datasets with different characteristics for our evaluations. The datasets are captured in varying lighting conditions, at different vehicle speeds, and over different trajectory lengths.

The remainder of the paper is organized as follows. Section II reviews related work. Section III introduces the notation used throughout the paper. Section IV describes our algorithm in detail. Section V provides an extensive experimental evaluation of our method. Finally, Section VI concludes the paper.

II. Related Work

Many different camera configurations have been used in the literature. The two most commonly used configurations are the monocular and the stereo settings. Approaches which use multiple cameras to build up a generalized camera system have also been proposed in the literature. We summarize these papers below.

To the best of our knowledge, the first work in real-time monocular camera SLAM can be attributed to [3]. They proposed a filtering-based algorithm which runs at 30 Hz using a single thread on standard PC hardware. In contrast to [3], Nister et al. [20] proposed a pure visual odometry algorithm using an iterative optimization approach. The proposed algorithm runs at video rates on a single thread. A system to separate localization and mapping on two parallel threads was proposed in [15], and became a

[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

978-1-5386-8093-3/18/$31.00 ©2018 IEEE

standard paradigm for subsequent VO algorithms. This separation enables the system to produce high-quality maps by using more advanced but computational expensive inference techniques while simultaneously achieving real-time camera pose tracking. However, the proposed system is mainly designed for a small AR workspace. Mur-Artal et al. [19] extend the system to handle large-scale environments by incorporating re-localization and loop closure techniques into the system. Concurrently, both Engel et al. [5] and Forster et al. [8] proposed two novel monocular SLAM systems. They also follow the two-thread paradigm, but with new feature representations and inference techniques in contrast to the aforementioned algorithms. In particular, instead of using conventional sparse corner features, [5] uses all high gradient pixels directly for pose estimation. The camera pose is estimated by minimizing the photometric intensity errors. Recently, Engel et al. [7] proposed a new monocular VO system which has the same front-end as [5] but with a different novel back-end for both pose and structure refinements. In particular, they jointly optimize camera poses, 3D structures and other variables by minimizing the photometric intensity errors directly (aka. direct method).

One of the drawbacks of monocular VO is its inability to recover metric scale. Therefore, to make them more useful for robot applications, additional sensor modalities [25] or cameras [17] are typically employed to capture scale information. For most of the above mentioned monocular systems, there exists a stereo version which allows for metric scale recovery. For example, Mur-Artal and Tardós [18] extend ORB-SLAM from [19] to stereo camera configurations. Both [5] and [8] have also been extended for stereo or multi-camera configurations [6, 9].

For most robot applications, a wide field of view (FoV) or even a 360° FoV is necessary for better situational awareness and perception capabilities. For example, Harmat et al. [12], Heng et al. [14] successfully demonstrated a multi-camera SLAM system for a micro aerial vehicle (MAV).

III. Notation

We use lower case letters (e.g. λ) for scalar variables, bold lower case letters (e.g. $\mathbf{v}$) for vectors, and bold capital letters (e.g. $\mathbf{T}$) for matrices. A coordinate frame x is denoted as $\mathcal{F}_x$. We define $\mathbf{T}_{B_k}^{W}$ as the transformation matrix that transforms vectors from frame $\mathcal{F}_B$ to frame $\mathcal{F}_W$ at timestamp k. We use $\mathbf{p}^{W_k}$ to denote the vector variable $\mathbf{p}$ represented in frame $\mathcal{F}_W$ at timestamp k. We use $\mathbf{b}_k$ to denote the vector variable $\mathbf{b}$ at timestamp k. Furthermore, we denote $\|\mathbf{x}\|_{\mathbf{\Sigma}}$ as a weighted L_2 norm, i.e., $\mathbf{x}^T\mathbf{\Sigma}^{-1}\mathbf{x}$.

Coordinate frames: There are three main coordinate frames used throughout the algorithm derivations. They are a global world coordinate frame $\mathcal{F}_W$, a vehicle-centric body coordinate frame $\mathcal{F}_B$, and a camera coordinate frame $\mathcal{F}_{C_i}$ for each camera C_i. Frame $\mathcal{F}_W$ is defined to be fixed relative to the global earth coordinate frame in which the cameras navigate. Furthermore, we define the initial vehicle-centric body coordinate frame $\mathcal{F}_B$ to coincide with $\mathcal{F}_W$, i.e., $\mathcal{F}_B$ at timestamp 0.

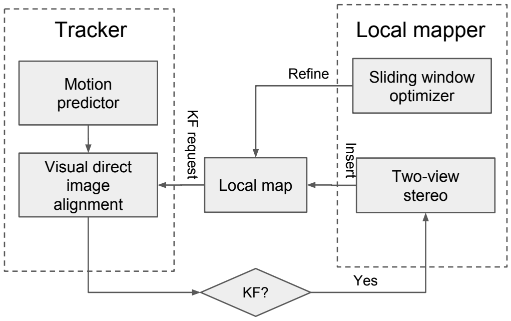

Fig. 1: Schematic representation of our VO pipeline.

Vehicle state: We describe the motion state $\mathcal{S}_k$ at timestamp k by its translation and rotation from frame $\mathcal{F}_B$ to frame $\mathcal{F}_W$. The position is denoted by $\mathbf{p}^{W_k}$. The orientation is represented by a rotation matrix $\mathbf{R}_{B_k}^{W} \in$ SO(3). We obtain the homogeneous transformation matrix $\mathbf{T}_{B_k}^{W} \in$ SE(3) from the vehicle body frame $\mathcal{F}_B$ to the world frame $\mathcal{F}_W$ at timestamp k:

$$\mathbf{T}_{B_k}^{W} = \begin{bmatrix} \mathbf{R}_{B_k}^{W} & \mathbf{p}^{W_k} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} . \tag{1}$$

We denote the image captured at timestamp k by camera C_j as $\mathbf{I}_k^{C_j}$. We further denote the measured pixel intensity at pixel coordinates $\mathbf{u}$ in $\mathbf{I}_k^{C_j}$ as $\mathbf{I}_k^{C_j}(\mathbf{u})$; the pixel intensity is a scalar value for grayscale images. For simplicity, we denote the image captured at the latest keyframe as $\mathbf{I}_{KF}^{C_j}$.

IV. Algorithm

In this section, we describe our VO algorithm for a multi-camera system. We follow [7], which describes a VO framework for monocular cameras that uses direct image alignment based on minimizing a photometric cost function for a sparse set of pixels. We extend the formulation to a multi-camera system which we model as a generalized camera consisting of cameras with multiple centers of projection. As shown in Fig. 1, our algorithm consists of two threads: a tracker and a local mapper. The tracker estimates the vehicle pose in real-time using direct image alignment with respect to the latest keyframe. The local mapper is mainly used to minimize long-term pose drift by refining both the vehicle pose and the estimated 3D point cloud. The local mapper uses a computationally intensive batch optimization method for both pose and structure refinement. Thus, it runs at a much lower frame rate compared to the tracker and only keyframes are processed. For a new keyframe, its newly sampled 3D points are initialized by a two-view stereo algorithm. We will describe the algorithm in detail in the following.

A. Hardware Configurations

To make our algorithm work for different hardware configurations, we assume we have N cameras in total. Each camera can be configured either as a reference camera for motion tracking or as an auxiliary camera for static stereo matching. For ease of reference, we denote using camera C_i

as a reference camera as ${}^{r}C_i$. Without loss of generality, we assume that we have N_r reference cameras and N_a auxiliary cameras where $N_r + N_a = N$.

B. Tracker

As shown in Fig. 1, the tracker consists of two parts: a motion predictor and direct image alignment. Direct image alignment is used to estimate the current vehicle pose with respect to the latest keyframe. We use the vehicle pose provided by the motion predictor to initialize the alignment in order to avoid local minima.

Constant velocity motion prediction model: We use a constant motion model to predict the current vehicle pose. Let us define the current vehicle pose as $\mathbf{T}^{W}_{B_k}$ which is the transformation from the vehicle frame $\mathcal{F}_B$ at timestamp k to the world frame $\mathcal{F}_W$. Similarly, we define the vehicle poses corresponding to the two latest frames as $\mathbf{T}^{W}_{B_{k-1}}$ and $\mathbf{T}^{W}_{B_{k-2}}$ respectively. We assume that the motion velocity from timestamp $k-2$ to k is constant, we thus can predict the current vehicle pose as

$$\mathbf{T}^{W}_{B_k} = \mathbf{T}^{W}_{B_{k-1}}\mathbf{T}^{B_{k-1}}_{B_k} = \mathbf{T}^{W}_{B_{k-1}}\mathbf{T}^{B_{k-2}}_{B_{k-1}} \ , \qquad (2)$$

where

$$\mathbf{T}^{B_{k-2}}_{B_{k-1}} = (\mathbf{T}^{W}_{B_{k-2}})^{-1}\mathbf{T}^{W}_{B_{k-1}} . \qquad (3)$$

Direct sparse tracker: Given the initial pose estimate provided by the motion predictor, we use a direct sparse tracker for refinement. In contrast to the local mapper which will be explained in a later section, only images from reference cameras are used for motion tracking. In particular, we estimate the relative vehicle pose $\hat{\mathbf{T}}^{B_k}_{B_{KF}}$ between the latest keyframe and the current frame by minimizing the following energy function:

$$\hat{\mathbf{T}}^{B_k}_{B_{KF}} = \underset{\mathbf{T}^{B_k}_{B_{KF}}}{\operatorname{argmin}} \sum_{i=1}^{N_r} \sum_{\mathbf{u}\in\mathbf{\Omega}(\mathbf{I}^{{}^rC_i}_{KF})} (\mathbf{I}^{{}^rC_i}_{KF}(\mathbf{u}) - \mathbf{I}^{{}^rC_i}_{k}(\hat{\mathbf{u}}))^2 \ . \qquad (4)$$

Here, N_r is the number of reference cameras and $\mathbf{\Omega}(\mathbf{I}^{{}^rC_i}_{KF})$ is the set of feature pixel points sampled from the keyframe image of reference camera ${}^{r}C_i$. For each feature point $\mathbf{u}$ in the keyframe image, $\hat{\mathbf{u}}$ is the corresponding pixel position in the current frame, obtained as

$$\hat{\mathbf{u}} = \pi(\mathbf{T}^{C_i}_{B} \cdot \mathbf{T}^{B_k}_{B_{KF}} \cdot (\mathbf{T}^{C_i}_{B})^{-1} \cdot \pi^{-1}(\mathbf{u}, d)) \ . \qquad (5)$$

$\mathbf{T}^{C_i}_{B}$ is the relative transformation from vehicle body frame to camera frame. π and π^{-1} are the camera projection function and back projection function, respectively. d is the depth of the feature point $\mathbf{u}$ in the keyframe. The only unknown variable is $\mathbf{T}^{B_k}_{B_{KF}}$ since $\mathbf{T}^{C_i}_{B}$ can be obtained from offline sensor calibration and d can be initialized by stereo matching and refined by the joint optimization performed by the local mapper.

The above formulation follows the standard forward compositional method [2] which requires the Hessians and Jacobians of the residuals to be computed in every iteration of the optimization. For improved efficiency, we adopt the inverse compositional method [2] to minimize the energy function. To robustify the least squares estimator which is sensitive to outliers, we further apply a robust loss function (Huber loss) to all residuals.

Outlier removal: Besides the use of a robust loss function, a specific outlier removal mechanism is also used to robustify both the tracker and the local mapper. We detect outliers based on the template matching scores between the reference frame and the current frame. In our implementation, we use the Zero Mean Normalized Cross-Correlation (ZNCC) score [24]. If the score is smaller than a predefined threshold, we classify a feature match as an oulier and remove it from the optimization. The tracker already uses image patches around the feature points and the warped image patches are already computed during direct image alignment. Thus, the computational overhead of the outlier removal step is marginal.

Relationship with the tracker from [7]: In contrast to the tracker in the Direct Sparse Odometry (DSO) algorithm [7], we do direct sparse pose tracking. To do pose tracking, DSO projects sampled feature points from all recent keyframes to the current keyframe and then do semi-dense pose tracking. Compared to semi-dense pose tracking, the computational complexity of direct tracking with sparse feature points is far smaller than that of semi-dense tracking. Furthermore, an outlier removal step as described in the previous section is included to make the tracking and mapping more stable.

C. Keyframe and Feature Selections

Keyframe selection: One of the implicit assumptions behind motion tracking is that the scene difference between the reference keyframe and the current frame is sufficiently small. If the difference between the current frame and the reference keyframe becomes too large, we instantiate a new keyframe. Intuitively, the difference between frames should be measured based on changes in image content rather than based on absolute pose differences (since the former strongly depends on the depth of the scene while the latter is oblivious to it). Therefore, we use the mean square optical flow as one of the criterions for detecting scene changes. More precisely, we compute

$$f = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{u}_i - \hat{\mathbf{u}}_i\|_2 \ , \qquad (6)$$

where $\|\cdot\|_2$ is the Euclidean norm, $\mathbf{u}_i$ is a feature point in the reference keyframe, and $\hat{\mathbf{u}}_i$ is its corresponding feature point in the current frame. If f is above a threshold, we create a new keyframe.

Feature selection: Once a new keyframe is created, sparse feature points are sampled from all reference cameras. We sample N sparse features uniformly from each image based on their gradient magnitudes. In particular, we divide the image into a grid and select the point with the highest gradient magnitude per grid cell. However, we will not sample a point from this grid cell if the highest gradient magnitude of a grid is smaller than a pre-defined threshold.

Feature representation and depth initialization: As per the implementation in [7], we sample all pixels by following a predefined patch pattern (e.g., 5×5 pattern) for each sparse feature. All these pixels are used for motion tracking and local mapping. In particular, the created visual residuals from the tracker and local mapper are per pixel instead of per patch. Furthermore, pixels from the same patch are assumed to lie on the same 3D plane, which can be parameterized by its inverse plane depth and plane normal. The inverse plane depths and plane normals are initialized by a stereo matching algorithm [11]. The plane depths of the sampled patch instead of ray depths of each pixel are refined during the joint optimization carried out by the local mapper. Each pixel from the same patch has different ray depths, but has the same plane depth. It thus reduces the number of variables to optimize and increases the computational efficiency of the optimizer.

D. Local Mapper

Two-view stereo matching: Stereo matching is used to initialize the depth of each sampled feature from the new keyframe. Rather than performing disparity search along epipolar lines, we use plane-sweeping stereo [11] to compute the depths. This allows us to directly operate on the fisheye images, and thus, avoid a loss of field-of-view from having to undistort and rectify the images.

Based on the implementation from [11], we use the GPU to accelerate plane-sweeping stereo. We sweep planes in two directions: fronto-parallel to the viewing direction of the keyframe and parallel to the ground plane. For each direction, we generate 64 plane hypotheses with a constant disparity step size between them. The planes cover the range $[0.5\ 30]\ \mathrm{m}$ in front of the camera. This results in a total number of 128 plane hypotheses that are evaluated. We compute the ZNCC score between the 7×7 image patches from $\mathbf{I}^{C_i}$ and warped image patches from $\mathbf{I}^{C_j}$. The plane hypothesis with the largest template matching score is selected for each feature point.

Sliding window optimizer: To minimize drift, the local mapper jointly optimizes all vehicle states and the 3D scene geometry. As the vehicle moves, the number of states increases over time. To bound running time, we use state marginalization to remove old states. We only keep a fixed number of previous states, resulting in a constant computational complexity [4]. As shown in [23], a marginalized state will result in all remaining states connected to it to be connected with each other after marginalization. The resulting Hessian matrix is not sparse anymore. This in turn increases the run-time cost of computing the Schur complement during structure optimization. For efficiency, we follow [16, 7] and only fill terms of the Hessian that do not involve geometry terms. For further efficiency, optimization is only performed on selected keyframes.

In particular, we define a sliding window containing k vehicle states. All states outside this sliding window are treated as old states and are removed through partial marginalization [7].

We use $\mathcal{S}_*$ to represent the set of all vehicle states within the sliding window and they are represented in vector form. The local mapper estimates the vehicle states $\mathcal{S}_*$ inside the sliding window, and the set $\mathbf{D}_*$ of inverse plane depths of all the sampled features within the sliding window via

$$\hat{\mathcal{S}}_*, \hat{\mathbf{D}}_* = \underset{\mathcal{S}_*, \mathbf{D}_*}{\operatorname{argmin}}\ \mathrm{E}_0(\mathcal{S}_*) + \mathrm{E}_{\text{vision}}(\mathcal{S}_*, \mathbf{D}_*). \tag{7}$$

Here, $\mathrm{E}_0(\mathcal{S}_*)$ is either a prior energy term obtained from partial marginalization or an initial prior and $\mathrm{E}_{\text{vision}}(\mathcal{S}_*, \mathbf{D}_*)$ is the visual energy term between all keyframes within the sliding window. Optimization is carried out using the Gauss-Newton algorithm. In the following, we describe the individual terms in more detail.

There are two types of **prior terms**. One is from the initial prior. In order to fix the unobservable degrees of freedom of the VO problem, we need to fix the initial state such that all subsequent states are estimated relative to it. Thus, they are usually formulated in the following form:

$$\mathrm{E}_0(\mathcal{S}_0) = \frac{1}{2}\left\|\hat{\mathcal{S}}_0 - \mathcal{S}_0\right\|_{\mathbf{\Sigma}_0}, \tag{8}$$

where $\hat{\mathcal{S}}_0$ is the value of the initial state, $\mathcal{S}_0$ is the inital state variable to estimate and $\mathbf{\Sigma}_0$ is the covariance matrix of the inital state. To fix $\mathcal{S}_0$ as equal to $\hat{\mathcal{S}}_0$, $\mathbf{\Sigma}_0$ is usually selected to have infinitesimal diagonal terms. This prior term only appears in the energy function when the initial state is within the sliding window. Once it leaves the window, it will be marginalized out in the same way as all other energy terms [23, 4].

The other prior term is a direct result of partial marginalization. Partial marginalization introduces priors to all remaining states which are connected to the marginalized states [23, 4]. The information from eliminated states after their removal is stored in prior Hessian and Jacobian matrices such that we can optimally remove them. Thus, we can have the following prior energy term for the remaining states

$$\mathrm{E}_0(\mathcal{S}_*) = \mathbf{J}_m^T(\hat{\mathcal{S}}_{*0} - \mathcal{S}_*) + \frac{1}{2}\left\|\hat{\mathcal{S}}_{*0} - \mathcal{S}_*\right\|_{\mathbf{H}_m{}^{-1}}, \tag{9}$$

where $\hat{\mathcal{S}}_{*0}$ is a set of prior vehicle states of $\mathcal{S}_*$ estimated from previous iterations, $\mathbf{J}_m$ and $\mathbf{H}_m$ are the Jacobian and Hessian matrices respectively, and accumulated from state marginalization.

The **visual energy term** contains the sum of all photometric error residuals and is modeled as

$$\begin{aligned}\mathrm{E}_{\text{vision}}(\mathcal{S}_*, \mathbf{D}_*) \ =\ & \sum_{m=1}^{k}\sum_{i=1}^{N_r}\sum_{\mathbf{u}\in\mathbf{\Omega}(\mathbf{I}_m^{{}^rC_i})}\sum_{n\in\mathbf{\Theta}(\mathbf{u})} \\ & \sum_{j\in\mathbf{\Phi}(\mathcal{S}_m,\mathbf{u})} \left\|r(\mathcal{S}_m, {}^rC_i, \mathcal{S}_n, C_j, \rho)\right\|_{\Sigma}.\end{aligned} \tag{10}$$

Here, k again is the number of keyframes and N_r is the number of reference cameras. ${}^rC_i$ and C_j refer to the i^{th}

and j^{th} camera respectively. $\mathbf{\Omega}(\mathbf{I}_m^{{}^rC_i})$ is the set of feature points sampled from the image of the i^{th} reference camera in the m^{th} keyframe. $\mathbf{\Theta}(\mathbf{u})$ is the set of indices of the vehicle states that observe the feature point $\mathbf{u}$. $\mathbf{\Phi}(\mathcal{S}_m, \mathbf{u})$ is the set of indices of the cameras in the m^{th} keyframe that observe feature point $\mathbf{u}$. ρ is the inverse plane depth of the feature $\mathbf{u}$. $r(\mathcal{S}_m, {}^rC_i, \mathcal{S}_n, C_j, \rho)$ is a pixel intensity residual that measures intensity differences between two projections of the same 3D scene point (see below for a definition of the residual). Furthermore, we use both inter-camera and intra-camera irradiance residuals in our implementation. Finally, Σ is the scalar variance of the photometric error residual which is obtained from each camera's photometric calibration.

Let $\mathbf{I}_m^{{}^rC_i}$ be the image captured by the i^{th} camera in the m^{th} keyframe. Consider a feature point $\mathbf{u}$ sampled from that image. Using plane sweep stereo, we know the inverse depth ρ of its corresponding plane as well as the normal $\mathbf{n}$ of that plane. The 3D point $\mathbf{P_u}$ corresponding to $\mathbf{u}$ is thus given by:

$$\mathbf{P_u} = \frac{1}{\rho \cos\theta} \pi^{-1}_{{}^rC_i}(\mathbf{u}) \ , \tag{11}$$

where $\pi^{-1}_{{}^rC_i}$ is the inverse projection function of the camera and $\cos\theta = -\mathbf{n}^T \cdot \pi^{-1}_{{}^rC_i}(\mathbf{u})$ is the angle between the viewing ray corresponding to $\mathbf{u}$ and the plane normal. The vehicle poses $\mathbf{T}^W_{B_m}$ and $\mathbf{T}^W_{B_n}$ at timestamps m and n are initially estimated by the tracker while the relative transformations $\mathbf{T}^{C_l}_B$ between the vehicle frame $\mathcal{F}_B$ and the camera frames $\mathcal{F}_{C_l}$ can be estimated offline. We use these transformations to compute the pixel $\hat{\mathbf{u}}$ in the j^{th} camera and in the n^{th} keyframe corresponding to the feature $\mathbf{u}$ by projecting $\mathbf{P_u}$ into the image. Given $\mathbf{u}$ and $\hat{\mathbf{u}}$, we use their intensity difference to define the residual $r(\mathcal{S}_m, {}^rC_i, \mathcal{S}_n, C_j, \rho)$:

$$r(\mathcal{S}_m, {}^rC_i, \mathcal{S}_n, C_j, \rho) = \mathbf{I}_m^{{}^rC_i}(\mathbf{p}) - \mathbf{I}_n^{C_j}(\hat{\mathbf{p}}) \ . \tag{12}$$

In order to minimize the visual energy term from Eq. 10, we adjust the motion state parameters (namely the pose $\mathbf{T}^W_{B_m}$) as well as the inverse plane depth of each feature. Rather than looking at a single pixel per feature $\mathbf{u}$, we consider a pixel patch per feature. Each pixel in the patch is warped into the other images and contributes a residual according to Eq. 12. In this setting, parameterizing the depth of $\mathbf{u}$ based on the plane has the advantage that we only require a single parameter per patch.

V. Experimental Evaluations

A. Hardware Setup

We have installed twelve fisheye cameras on our vehicle platform which is shown in Fig. 2. In particular, five cameras are installed at the front of the vehicle, two cameras are installed on each of the left and right sides, and three cameras are installed at the back. We select one stereo pair from each side to evaluate our algorithm. In particular, the front stereo pair has a baseline of 0.722m, the back stereo pair has a baseline of 0.755m, the left stereo pair has a baseline of 0.502m, and the right stereo pair has a baseline of 0.497m. All cameras output 1024×544 gray scale images at 25 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. To compute the transformation between the multi-camera system and the GPS/INS system, we run semi-direct VO [13] for each stereo pair, obtain an initial estimate using hand-eye calibration, triangulate landmark points using the GPS/INS poses and feature correspondences from VO, and refine the initial estimate together with the landmark points by minimising the sum of squared reprojection errors while keeping the GPS/INS poses fixed. 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 and Discussions

Dataset selection: To avoid parameter overfitting, and, at the same time, evaluate our algorithm thoroughly, we select five datasets with different characteristics for evaluation. The first three datasets are collected in a car-park. The first one (Science-park-day) is collected in normal daylight conditions. The second one (Science-park-night-illum) is collected at night with NIR illumination. The third one (Science-park-night) is collected at night without NIR illumination. The other two datasets are collected from a public urban street. One of them (West-coast-day) is collected in day light conditions. Another one (West-coast-night-no-illum) is collected at night without near-infrared illumination. Four sample images are shown in Fig. 3. Furthermore, the characteristics of the datasets are summarized in Table I.

	Length (m)	Max. speed (m/s)
Science-park-day	547.448	3.941
Science-park-night-illum	612.078	3.863
Science-park-night	613.096	3.685
West-coast-day	1179.13	10.68
West-coast-night-no-illum	1224.95	10.23

TABLE I: Characteristics of datasets used for evaluation.

Evaluation metrics: We follow the metric used by the KITTI benchmark [10] for accuracy evaluation. In particular, we compute the position and orientation drifts over sequences of length 200m, 400m, 600m and 800m for each frame, and average drifts over all sequences. We take into account the runtime of both the tracker and the mapper as a metric for evaluating efficiency. Furthermore, we observe that the ground truth heights for all science park datasets are not reliable even though a highly accurate GPS/IMU system was used. For example, for the Science-park-day dataset, the closed-loop ground truth height drift is more than 2.5 m. Since these errors introduce significant bias into the evaluation, we only consider horizontal translations of all three science park datasets for accuracy evaluations. The remaining datasets are evaluated with 3-axis translation errors.

Parameter settings: For all experiments, we sample a total

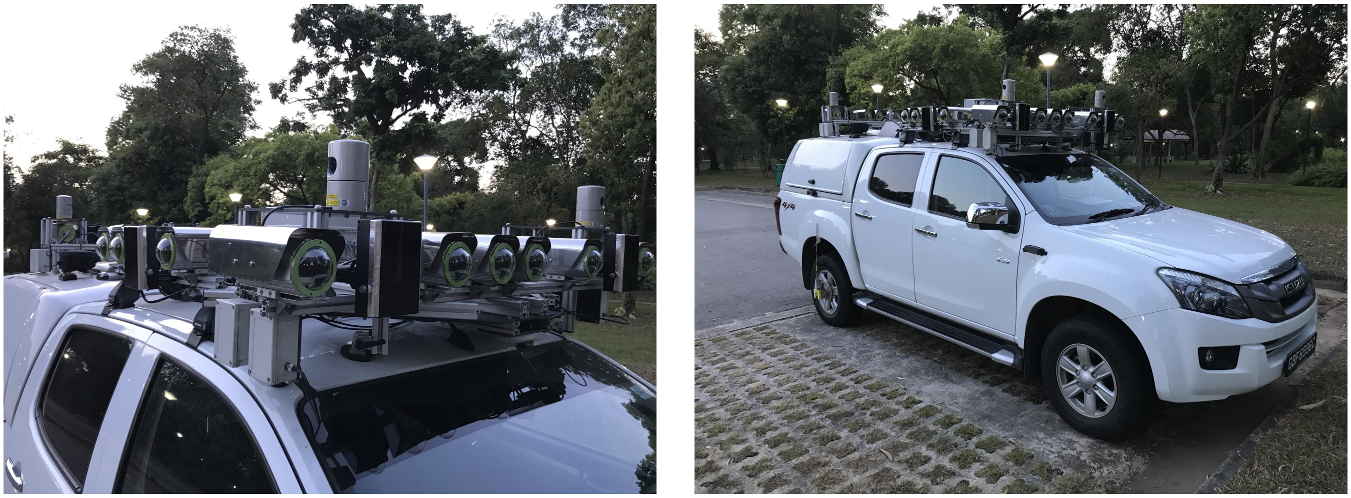

Fig. 2: Two sample images of the vehicle.

(a) Science-park-day

(b) Science-park-night-illum

(c) West-coast-night-no-illum

Fig. 3: Three sample images from our datasets, demonstrating the challenges of the datasets which include strong distortion and absence of features at night.

of 800 5×5 feature patches. All features are uniformly distributed in each camera. For example, if we consider four stereo pairs, then each stereo pair has 200 features. A sliding window with 5 keyframes is used for the local mapper. The optical flow threshold for keyframe selection is 20 pixels. For the Huber parameter, we use 30 for daylight conditions and 10 for night-time conditions.

Baseline algorithm: We choose ORB-SLAM2 [18] as the baseline algorithm for our comparisons. Since ORB-SLAM2 does not support the fisheye camera models, we undistort the fisheye images to generate pinhole images for all experiments.

Our algorithm is evaluated on a laptop with an Intel i7 CPU @ 2.8 GHz. We are able to run our tracking thread at more than 30 Hz and the local mapping thread at around 2 Hz for the current configurations.

Ablation studies: We conduct thorough ablation studies with respect to the hardware configurations. The results can be found in Table II. It shows that a higher number of stereo cameras improves both the accuracy and robustness of our VO algorithm. In particular, both ORB-SLAM2 [18] and our single stereo configuration fail easily for all the night sequences when using only a single or two stereo pairs. Based on our observations, the main reason of the failure is due to the lack of good features. However, using a multi-camera setup which covers a larger field of view can provide the required redundancy necessary for handling poorly textured night environments.

From Table II, we also observe that our algorithm performs better than ORB-SLAM2 [18]. We give two possible reasons: (1) the direct method can do refinement with sub-pixel accuracy which improves the accuracy as long as the algorithm is well-implemented, and (2) the image quality degrades after undistortion and stereo rectification, and can affect ORB-SLAM's performance. In contrast, our method is able to directly operate on the raw fisheye input images.

By doing horizontal comparisons (i.e. same dataset) of the same algorithm for different hardware configurations, we can observe that by using more cameras improves the accuracy of our VO algorithm. We think that spatially well distributed features can give better constraints to the optimization problem, which makes our VO algorithm more accurate.

Furthermore, by doing vertical comparisons (i.e. same hardware configuration) of the same algorithm for different light conditions, we can observe that night-time conditions degrade the performance of VO algorithms. Besides the lack of good features, another reason is that the image quality in night-time conditions is worse than that in daylight conditions. When the vehicle moves fast at night, motion blur is inherent in the captured images, especially for the left and right stereo pairs.

Qualitative evaluations: Fig. 4 shows qualitative evaluation results of our algorithm for all datasets with four pairs of stereo cameras. We plot the x-y trajectories estimated by our VO algorithm against the ground truth trajectories. A supplementary video can also be found on our project website https://cvg.ethz.ch/research/visual-odometry/.

VI. Conclusion

We present a direct sparse visual odometry algorithm for a multi-camera system and robust operation in challenging environments. Our algorithm includes a direct sparse pose tracker and a local mapper. The tracker tracks the current camera pose in real-time. The local mapper jointly optimizes both poses and structure within a sliding window. Instead of minimizing re-projection errors, both the tracker and mapper directly minimize photometric errors. We evaluate our algorithm extensively with five datasets which have different characteristics. Experimental results show that a multi-camera setup makes the VO more robust to challenging night environments and also improves its accuracy.

References

[1] H. Alismail, B. Browning, and S. Lucey. Robust tracking in low light and sudden illumination changes. In *3DV*, 2016.

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

[3] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2003.

[4] T.-C. Dong-Si and A. I. Mourikis. Motion tracking with fixed-lag smoothing: algorithm and consitency analysis. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2011.

[5] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular slam. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2014.

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

[7] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. In *arXiv*, 2016.

[8] C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *Proc. IEEE International Conf. 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*, 33(2), 2017.

[10] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

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

[12] A. Harmat, M. Trentini, and I. Sharf. Multi-Camera Tracking and Mapping for Unmanned Aerial Vehicles in Unstructured Environments. *Journal of Intelligent and Robotic Systems*, 2014.

[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] L. Heng, G. H. Lee, and M. Pollefeys. Self-Calibration and Visual SLAM with a Multi-Camera System on a Micro Aerial Vehicle. In *Proc. Robotics: Science and Systems (RSS)*, 2014.

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

[16] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. T. Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *International Journal of Robotics Research (IJRR)*, 2015.

[17] P. Liu, L. Heng, T. Sattler, A. Geiger, and M. Pollefeys. Direct visual odometry for a fisheye-stereo camera. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[18] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *arXiv preprint arXiv:1610.06475*, 2016.

[19] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5), 2015.

[20] D. Nister, O. Naroditsky, and J. Bergen. Visual Odometry. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[21] S. Park, T. Schöps, and M. Pollefeys. Illumination change robustness in direct visual slam. In *ICRA*, 2017.

[22] G. Pascoe, W. Maddern, M. Tanner, P. Pinies, and P. Newman. NID-SLAM: Robust monocular SLAM using normalized information distance. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[23] G. Sibley, L. Matthies, and G. Sukhatme. Sliding Window Filter with Application to Planetary Landing. *Journal of Field Robotics (JFR)*, 2010.

[24] L. D. Stefano, S. Mattoccia, and F. Tombari. ZNCC-based template matching using bounded partial correlation. *Pattern Recognition Letters*, 2005.

[25] J. Zhang and S. Singh. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[26] Z. Zhang, C.Forster, and D. Scaramuzza. Active Exposure Control for Robust Visual Odometry in HDR Environments. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2017.

Datasets	Alg*	F	B	L	R	FB	FL	FR	BR	BL	LR	BLR	FLR	FBL	FBR	FBLR
SD	Ours	0.449	1.266	1.185	1.194	0.56	0.282	0.632	0.86	0.64	1.24	0.511	0.635	0.838	0.524	0.352
	ORB*	1.168	1.565	x	4.203	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
SN1	Ours	x	3.7	x	x	x	2.05	x	1.01	1.544	2.265	0.625	0.735	0.956	1.193	0.691
	ORB*	3.24	x	x	x	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
SN2	Ours	x	x	x	x	x	x	x	x	x	1.01	1.47	0.815	1.544	1.51	1.03
	ORB*	x	x	x	x	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
WCD	Ours	2.33	1.32	2.17	4.35	0.48	1.79	2.02	3.47	0.96	1.53	1.59	1.97	1.47	0.87	0.9
	ORB*	2.34	1.86	x	x	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
WCN	Ours	x	x	x	x	2.88	11.6	8.57	5.12	4.99	2.84	3	2.23	2.33	2.05	1.74
	ORB*	x	x	x	x	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.

TABLE II: Accuracy evaluations with offline datasets for different hardware configurations (Accuracy (translational drift) is in units of %, the smaller the better; SD: Science-park-day. SN1: Science-park-night-illum. SN2: Science-park-night. WCD: West-coast-day. WCN: West-coast-night-no-illum. Alg*: Algorithms. ORB*: ORB-SLAM2. F: front stereo pair. B: back stereo pair. L: left stereo pair. R: right stereo pair. x: fails to complete whole sequence. N.A.: not available.)

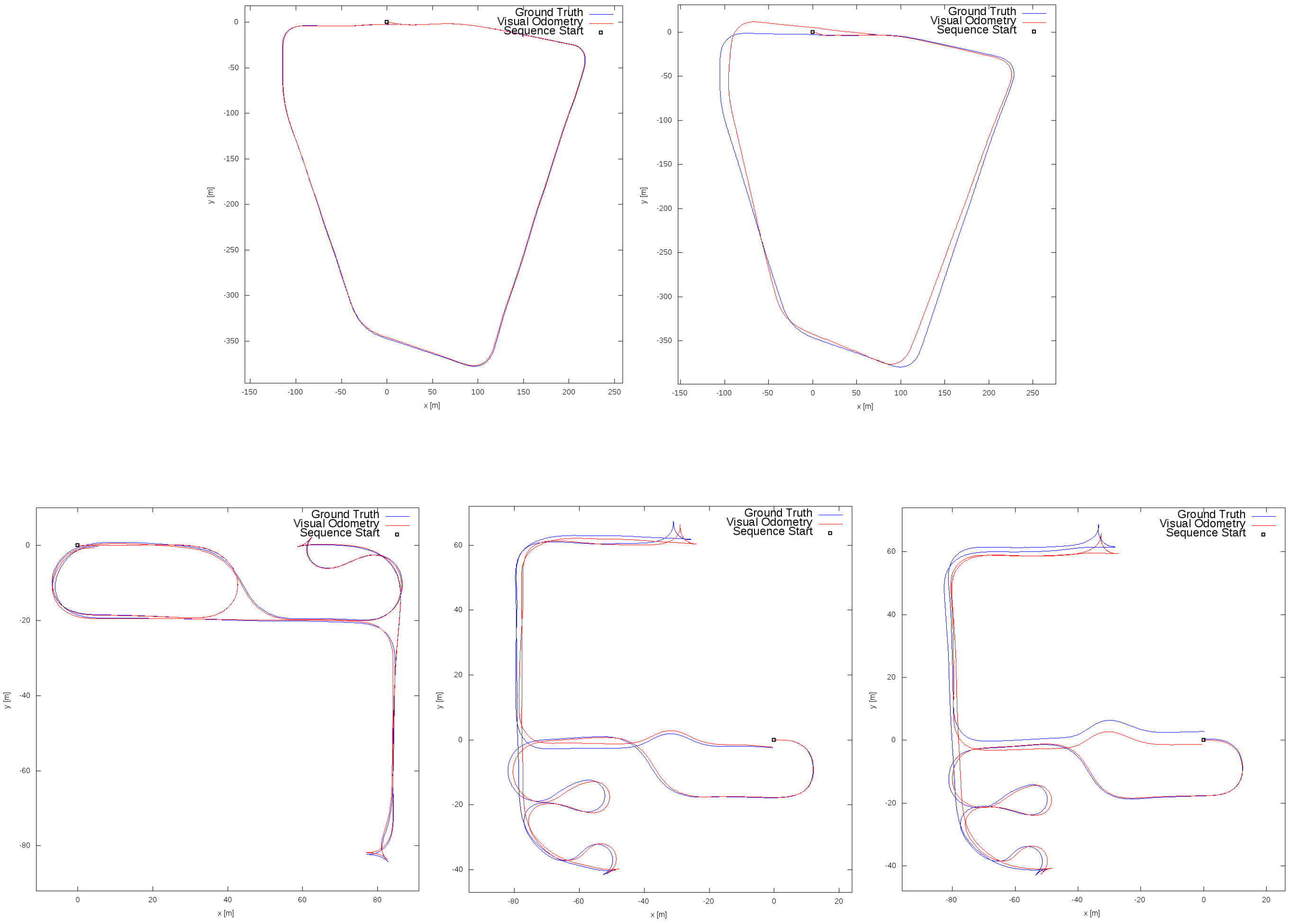

Fig. 4: Trajectories for five selected datasets (Top left: West-coast-day dataset. Top right: West-coast-night-illum dataset. Bottom left: Science-park-day dataset. Bottom center: Science-park-night-illum dataset. Bottom right: Science-park-night dataset).