Real-time Dense Stereoscopic Visual Odometry

Sammy Omari *†, Michael Bloesch *, Michael Burri *, Pascal Gohl *†, Markus W. Achtelik * and Roland Y. Siegwart *
*Autonomous Systems Lab, ETH Zurich, Switzerland, first.last@mavt.ethz.ch
†Skybotix AG, Luegislandstrasse 105, Zurich, Switzerland, first.last@skybotix.com

***Abstract*—Real-time dense mapping and pose estimation is essential for a wide range of navigation tasks in mobile robotic applications. We propose an odometry and mapping system that leverages the full photometric information from a stereo-vision system in a probabilistic framework while running in real-time on a single low-power Intel CPU core. Instead of performing mapping and localization on a set of sparse image features, we use the complete, dense image intensity information in our navigation system. By incorporating a probabilistic model of the stereo sensor, we can robustly estimate the ego-motion as well as a dense 3D model of the environment in real-time. The probabilistic formulation of the joint odometry estimation and mapping process enables to efficiently reject *temporal* outliers in ego-motion estimation as well as *spatial* outliers in the mapping process. The performance of the proposed dense scheme is evaluated in a hand-held experiment setup against a state-of-the-art sparse stereo framework. It is shown that the the dense framework exhibits superior tracking performance in terms of accuracy as well as robustness in scenarios with highly dynamic, jerky motion patterns - while retaining a relatively small computational footprint. This makes it an ideal candidate for a navigation framework on mobile robotics platforms with power- and weight constraints.**

I. Introduction

Traditional stereo-vision-based SLAM systems are usually based on estimating the ego-motion using sparse image features [8], [1]. The sparse 3D map can then be augmented into a dense map by projecting the depth images of individual stereo-frame-pairs into 3D using their corresponding pose estimates [6], [9], [13]. These depth images are traditionally obtained either from a stereo-based dense reconstruction algorithm such as block-matcher [11], SGM [7] or ELAS [5] or directly from an RGBD sensor. These serial, two-step approaches usually project the depth-images in a feed-forward fashion into 3D and do not leverage this additional information for state-estimation.

With the advent of RGBD cameras, simultaneous ego-motion estimation and dense mapping frameworks that are leveraging the full intensity and depth images [4], [10] have become increasingly popular. While there are significant contributions in this new field using stereo cameras [3], these dense approaches are mostly based on RGBD cameras since these sensors directly provide relatively high-quality depth images at negligible computational cost. Compared to traditional VO-systems where a few hundred sparse image points are used for optimization, these dense approaches have a significantly larger error term count since for each pixel, an intensity- (and possibly depth-) error term is computed. As a consequence, these dense approaches tend to have a large computational footprint and are therefore mostly performed on a GPU which prohibits their use on mobile robots with constraints in terms of power consumption and weight. Recently, first real-time, CPU-based approaches for dense motion estimation using RGBD cameras have been proposed [10] that employ highly optimized SSE-vectorized implementations of the most time-consuming algorithmic subsets.

However, while RGBD cameras are an interesting option for performing state estimation and mapping they come with a set of significant drawbacks. Since most RGBD cameras are based on active IR-measurements, such solutions tend to fail when used in direct sunlight, which can be a major issue when working on a wide range of (field-) robotic systems. Additionally, most RGBD cameras are equipped with rolling shutter cameras which makes it difficult to integrate them on highly dynamic systems such as walking robots or multicopter systems. We propose a stereo-vision based dense

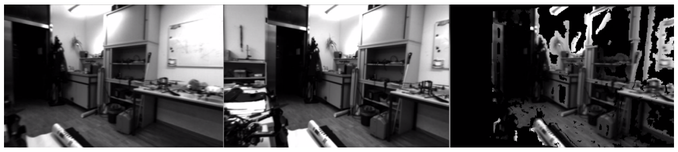

Fig. 1. *Inverse Compositional* dense visual odometry approach predicting keyframe image (right) using current image (left) and keyframe image (middle) for motion estimation.

visual navigation system to overcome the inherent limitations due to the use of an RGBD camera while retaining a small computational footprint. Compared to a RGBD-camera based system, a stereo-based framework is facing a set of challenges. As the computation of the disparity image is costly, a simple reconstruction algorithm is to be employed that might produce spatial outliers in the resulting disparity images. Therefore, compared to a RGBD-based scheme, we have to take these outliers into account in the ego-motion and mapping process. Additionally, we should minimize the amount of computed disparity images by only performing this computation when necessary. To the best of our knowledge, this is the first approach that presents dense real-time motion estimation and mapping using a stereo camera in real-time on a low-power CPU making it an interesting alternative to traditional sparse ego-motion estimation frameworks.

II. Dense Visual-Odometry and Mapping

In this section, we discuss the underlying theoretical framework for dense stereo-based ego-motion estimation.

A. Stereo Camera Modeling

We assume that the stereo camera is fully calibrated and that we can obtain undistorted and rectified image pairs.

We can map each pixel coordinate $\mathbf{x} = (u, v)$ and its corresponding disparity $\mathbf{D}(\mathbf{x})$ into the corresponding 3D point $\mathbf{p} \in \mathbb{R}^3$ using the inverse projection operator $\boldsymbol{\pi}^{-1}$ as

$$\mathbf{p} = \boldsymbol{\pi}^{-1}(\mathbf{x}, \mathbf{D}(\mathbf{x})) = \frac{fb}{\mathbf{D}(\mathbf{x})} \begin{bmatrix} \frac{u-c_u}{f} \\ \frac{v-c_v}{f} \\ 1 \end{bmatrix} \tag{1}$$

where b is the camera baseline, f is the focal length, c_u and c_v the horizontal and vertical camera center. The projection operator that maps a 3d point to a pixel coordinate can be defined accordingly. The disparity $\mathbf{D}(\mathbf{x})$ is estimated from a stereo image pair using a dense reconstruction algorithm [7], [11]. The warping operator $\tau(\mathbf{T}, \mathbf{x})$ maps the pixel location of a projected point from one image to another using the corresponding relative camera transformation $\mathbf{T}$ by

$$\tau(\mathbf{T}, \mathbf{x}) = \boldsymbol{\pi}(\mathbf{T} \cdot \boldsymbol{\pi}^{-1}(\mathbf{x}, \mathbf{D}(\mathbf{x}))). \tag{2}$$

The warping operator first projects the disparity into a 3D point, then transforms it into the other camera frame using the transformation $\mathbf{T}$ and finally projects it back into the other camera frame. By assuming that a 3D point results in the same image brightness irrespective of the camera viewpoint (photoconsistency) and that occlusions due to camera motion are negligible, we can write

$$\mathbf{I}_1(\tau(\mathbf{T}_{12}, \mathbf{x})) \approx \mathbf{I}_2(\mathbf{x}) \tag{3}$$

where $\mathbf{I}(\mathbf{x})$ is the intensity value at pixel coordinate $\mathbf{x}$.

B. Dense Visual Motion Estimation

As frame-to-frame approaches tend to accumulate drift even when the sensor remains static, we propose a frame-to-keyframe approach that is locally drift free. We can use relation (3) to perform motion estimation. As we do not have access to the transformation between the frames in Eq. (3), we can now use this relation in an optimization framework. We can obtain an estimate of the transformation $\mathbf{T}$ between the current- and the keyframe image, $\mathbf{I}_c$ and $\mathbf{I}_k$ respectively, by minimizing the intensity error over the full intensity image

$$\hat{\mathbf{T}} = \underset{\mathbf{T}}{\operatorname{argmin}} \sum_{\mathbf{x} \in \mathbf{I}_k} w(e) e(\mathbf{T}, \mathbf{x})^2 \tag{4}$$

where

$$e(\mathbf{T}, \mathbf{x}) = \mathbf{I}_k(\mathbf{x}) - \mathbf{I}_c(\tau(\mathbf{T}, \mathbf{x})) \tag{5}$$

is the intensity error image and $w(e)$ is a robust weighting term, described in more detail in Sec. II-D.

While we could define another error-term, i.e. the transformation and warping of the keyframe intensity image to the current one, this *inverse-compositional* approach has several benefits. For one, we have to compute the disparity image only for the keyframes while the other images only have to be undistorted and rectified. Additionally, we will show that in order to solve this nonlinear optimization problem, we have to calculate the image derivatives only once for every keyframe.

C. Linearization and Solving

We can incrementally solve for the optimal transformation using Gauss Newton by stacking all error terms in the form

$$\mathbf{J}^\top \mathbf{W} \mathbf{J} \Delta \mathbf{T} = -\mathbf{J} \mathbf{W} \mathbf{e}(\mathbf{T}) \tag{6}$$

where $\mathbf{J} \in \mathbb{R}^{n \times 6}$ and $\mathbf{e}(\mathbf{T}) \in \mathbb{R}^n$ are the stacked matrices of all jacobians and intensity errors respectively and n is the pixel count of the image. The derivation of the individual jacobians is detailed in Sec. III-D. The matrix $\mathbf{W}$ contains the weights of each intensity error on its corresponding diagonal entry.

We can now solve for $\Delta\mathbf{T}$ and add it to the current pose estimate[1]. We iteratively reapply this scheme by first recomputing the error image (5) using the new pose estimate and then solving for the new motion increment until convergence.

In order to increase the region of convergence for the estimation process, we apply a coarse-to-fine scheme by building an image pyramid by reducing the image resolution for each level by a factor of two. We start the estimation on the coarsest level and use the resulting motion estimate as an initial guess for the next level. The reader is referred to the video supplement of the paper, where the inverse compositional rendering approach is detailed.

D. Stereo Camera Noise Modeling

We assume that the intensity error is caused by additive noise on the estimated disparities n_d and by a general noise term n_r that is taking the remaining unmodelled effects into account, e.g. imprecise camera intrinsics:

$$e(\mathbf{T}, \mathbf{x}) = \mathbf{I}_k(\mathbf{x}) - \mathbf{I}_c(\boldsymbol{\pi}(\mathbf{T} \cdot \boldsymbol{\pi}^{-1}(\mathbf{x}, \mathbf{D}(\mathbf{x}) + n_d))) + n_r \tag{7}$$

The noises are modeled as zero-mean, white Gaussian noises with standard deviation σ_d and σ_r accordingly. By using a linear error propagation model, we can obtain the error distribution as

$$\sigma_e^2 = \sigma_r^2 + \mathbf{J}_d^\top \sigma_d^2 \mathbf{J}_d \tag{8}$$

where $\mathbf{J}_d$ is the Jacobian of the error $e(\mathbf{T}, \mathbf{x})$ w.r.t. the disparity $\mathbf{D}(\mathbf{x})$. Using this estimate of the error variance σ_e for each individual pixel, we can efficiently compute robust Tukey weights that are subsequently employed in (6).

E. Temporal Disparity Map Outlier Rejection

To evaluate the reliability for a single disparity estimate, we assume that outliers in the disparity map cause large intensity errors in the estimation process as the camera undergoes translational motion. More precisely, we can use the Mahalanobis distance of the tracking error $\xi = \frac{e^2}{\sigma_e^2}$ to detect erroneous disparity estimates. During tracking, those parts of the keyframe image that result in errors that exceed a certain threshold of the Mahalanobis distance $\xi > \alpha$ are flagged as outliers.

[1]Please note that since the pose resides on the SE(3) manifold, we have to take special care for handling rotations

When a new keyframe is generated, a new disparity map is computed. We can match this new keyframe to the old one using Eq. (3). Those pixels that do not exceed the Mahalanobis distance threshold α are considered to be inliers and are directly used for mapping. Those pixels that are projected in an outlier region in the old keyframe are considered to be neither inlier- nor outlier. Instead, we consider them as inliers when these pixels result in an error below α when the camera performed some translational motion.

III. Implementation

A. Image Processing & Warping

The standard openCV block-matching (BM) algorithm [11] is used for disparity map estimation with sub-pixel accuracy. As we want to keep the computational load low, we chose the BM over other algorithms such as ELAS or SGM which outperform the BM in terms of reconstruction quality. While ELAS and SGM significantly outperform the BM especially in texture-poor image areas due to the local nature of the BM, this does not significantly affect the estimation performance. This is mostly attributed to the fact that only image regions with strong image gradients are used for estimation, as discussed in Sec. III-C.

For image warping, the highly optimized, SSE-vectorized implementation of the *dvo*-framework [10] is used. It is noted that, while we use the stereo pair for disparity map estimation, only one camera is used for tracking in order to reduce the computational load.

For keyframe selection, we use a simple heuristic approach. We select a new keyframe when the angle between the optical axis of the keyframe- and current frame is above a g threshold. Translational motion triggers a new keyframe when the ratio between translational motion and average scene depth is above a given threshold.

B. Linearized Motion Estimation

To compute the motion increment of Eq. (6), both the matrix $\mathbf{J}^\top\mathbf{W}\mathbf{J} \in \mathbb{R}^{6\times 6}$ and the column vector $-\mathbf{J}\mathbf{W}\mathbf{e}(\mathbf{T}) \in \mathbb{R}^{6\times 1}$ have to be computed. One option to construct these terms could be the multiplication of the full jacobian matrices $\mathbf{J} \in \mathbb{R}^{6\times n}$. As n can be a a few 100k elements, depending on the pixel count of the image, the construction of the individual jacobian matrices and the subsequent multiplication can be expensive, especially in terms of memory access. Instead, for each error term, we can compute the outer product $w_i\mathbf{J}_i^\top\mathbf{J}_i$ and subsequently sum up all resulting 6x6 matrices (in parallel) to obtain the left-hand-side of the linear equation system. More precisely, by storing the jacobian matrix in CPU registers, the outer product is computed and directly added to the temporary matrix sum which resides in L1 cache. As the individual error terms are independent of each other, this process is ideal for both parallelization as well as loop unrolling, the latter ensuring the exploitation of instruction level parallelism (ILP) of modern Intel CPUs. Additionally, using this approach, we ensure a minimum number of cache misses and costly (higher level) memory accesses.

Once the system of equations is constructed, we can efficiently solve for the motion increment $\mathbf{T}$ using Cholesky decomposition since the matrix $\mathbf{J}^\top\mathbf{W}\mathbf{J}$ is positive definite. For each iteration we apply the motion increment by $\mathbf{T} \leftarrow \mathbf{T}\Delta\mathbf{T}$. We iteratively re-apply this scheme until convergence for each image pyramid scale.

C. Error-term reduction

Incorporating all pixels in the motion estimation in the form of error terms results in considerable computational load. Instead, we can employ a pixel selection-process that ensures that only parts of the image are used for tracking that contribute significantly to the estimation process. The underlying idea of the pixel selection process is that the subset of pixels with a large magnitude of $\mathbf{J}$ in at least one degree of freedom dominate in the motion estimation [4].

By inspection of (10), we know that the image gradients have a signficant influence on the magnitude of the Jacobian. Therefore, instead of selecting the pixels based on each of the 6 degrees of freedom of the Jacobian as done in [4], we use a selection function that is purely based on the image gradients

$$S(\mathbf{x}) = \|\nabla_x \mathbf{I}_k(\mathbf{x})\| + \|\nabla_y \mathbf{I}_k(\mathbf{x})\| \tag{9}$$

We can now select those pixels whose gradient magnitudes $\|S(\mathbf{x})\|$ are above a time-varying threshold. To ensure a proper pixel distribution in the image, we subdivide the image into smaller section (*bucketing*) and select the highest scoring pixels in each section. This ensures that a certain amount of pixels are also taken into account from low-texture image regions. Additionally, we temporally adapt the gradient threshold using a proportional controller, thus driving the pixel count to a user-defined value. While only a subset of pixels is used for tracking, it is noted that stochastic mapping process, as discussed in Sec. II-E, is still performed on the full intensity image.

As we follow an inverse-compositional warping approach, the pixel selection process is performed only at keyframe-level, thus reducing the computational load. To ensure proper convergence across all pyramid scales, the pixel selection process is performed individually at each level.

D. Jacobian Derivation

In order to estimate the transformation given the minimization criterion of Eq. (7), the Jacobian of the intensity error w.r.t. relative camera transformation is to be derived. Using the chain rule, for an individual pixel error, with some abuse of notation, we can write

$$\mathbf{J} = \mathbf{J}_\mathbf{I}\mathbf{J}_\pi\mathbf{J}_\mathbf{T} \tag{10}$$

where $\mathbf{J}_\mathbf{I}$ is the image derivative of the warped image

$$\mathbf{J}_\mathbf{I} = \left.\frac{\partial \mathbf{I}_c(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\tau(\mathbf{T},\mathbf{x})} \approx \left(\nabla_x \mathbf{I}_k \;\; \nabla_y \mathbf{I}_k\right) \tag{11}$$

By assuming that the warped image is sufficiently close to the keyframe image, we can approximate the derivative of the warped image by the keyframe image derivative. This results in a significant speed-up, as the the image derivatives are only calculated for the keyframe instead of every warped image (which is generated at every iteration of the optimization scheme).

The term $\mathbf{J}_\pi$ corresponds to the Jacobian of the projection function (1) evaluated at the 3D coordinates of the observed point expressed in the current camera frame

$$\mathbf{J}_\pi = \left.\frac{\partial \pi(\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\tau(\mathbf{T},\mathbf{x})}. \tag{12}$$

The term $\mathbf{J_T}$ is the Jacobian of the 3d coordinates of the point in the expressed in the current frame w.r.t. the transformation $\mathbf{T}$

$$\mathbf{J_T} = \frac{\partial(\mathbf{T}\cdot\boldsymbol{\pi}^{-1}(\mathbf{x},\mathbf{D}(\mathbf{x})))}{\partial \mathbf{T}}. \tag{13}$$

IV. Experiments

In this section, we evaluate our *dense* stereo VO-framework in a set of hand-held experiments to a *sparse* VO-framework. The performance of the dense framework in terms of accuracy and runtime is compared to the state-of-the art sparse stereo-vision framework fovis [9]. It is noted that, in the context of this workshop paper, this is by no means a conclusive comparison, however, it should provide the reader with an intuition on the relative performance of these frameworks.

A. Sensor Suite

The data for the hand-held experiments were recorded with the VI-Sensor [12], equipped with two time-synchronized, global-shutter, wide-VGA $1/3$ inch imagers in a fronto-parallel stereo configuration with an 11 cm baseline as depicted in Fig. 2. The cameras are equipped with lenses with a diagonal field of view of 120 degrees and are calibrated for a standard pinhole projection model and a radial-tangential distortion model. While the VI-Sensor is equipped with a timesynchronized inertial measurement unit (IMU), it is not used in this experiment.

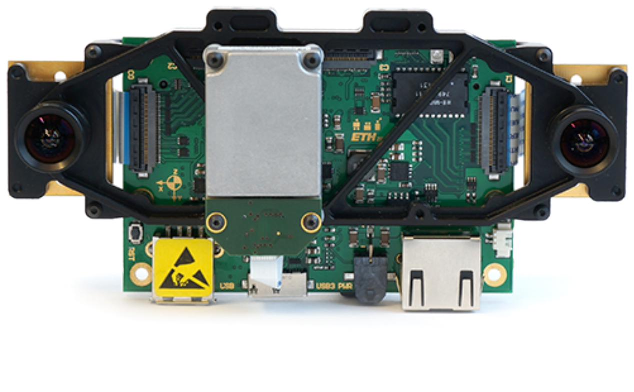

Fig. 2. Visual-Inertial (VI)-Sensor in fronto-parallel stereo configuration with 11 cm baseline. While this sensor is equipped with a timesynchronized IMU, it is not used in the context of this paper.

B. Processing Platform

For the runtime measurements, we run both frameworks on a single core of an Intel Core 2 Duo, 1.86 GHz, 4 GB RAM. The system is running Ubuntu 13.04, as compiler we use the gcc 4.73 with the flags *-O3 -march=native -fno-strict-aliasing*. For fovis, we use the publicly available open source implementation. For the dense framework, the second and third lowest image pyramid scales (376 x 240 & 188 x 120) are used for tracking. However, in the mapping process, we the full resolution intensity and disparity images are used. Therefore, the timings include the rectification and block-matching processes of the full size images.

C. Ground Truth

For this version of the paper, we do not provide ground truth by a motion tracking system. Instead, we simply move the sensor to the same spot as when the experiment started. As we do not have any loop-closing capabilities, this presents a preliminary indicator for the relative accuracy of the VO frameworks.s

D. Description Experiments

In the handheld scenario, we first perform relatively smooth trajectory with no jerky motions. Here, we want to evaluate the local accuracy of the frameworks as well as the noise level of the state estimation by numerically differentiating its translational trajectory to obtain a (naive) velocity estimate. In the second hand-held scenario, we emulate a jerky motion with significant rotations around all camera axes as well as large translational accelerations. This is done to evaluate the robustness of the state estimators to jerky motions with significant motion blur. It is noted that the dataset was recorded in an indoor setting with all regular light sources in the room switched on.

E. Results

1) Slow Experiment: Figure 3 and 4 show the trajectory evolution of the dense VO framework as well as fovis. As the position and orientation is estimated relative to the starting point, the final values should be close to zero. The dense VO

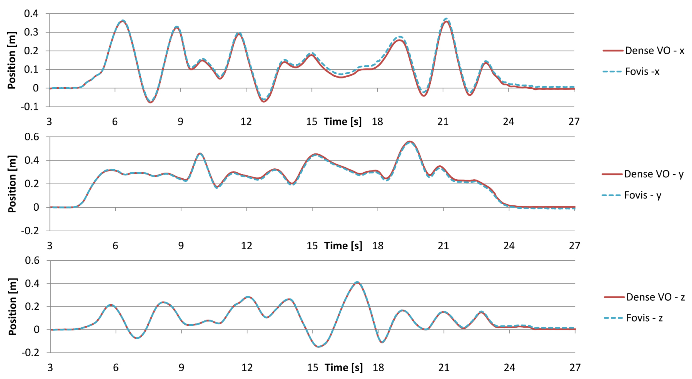

Fig. 3. Translation estimate of the *slow* dataset by the fovis and the dense VO stereo framework.

framework outperforms the fovis framework in terms of final RMS position error of $3.9\,mm$ to $10.1\,mm$. The rotational RMS error is $0.22\,deg$ for the dense VO and $0.62\,deg$ for fovis.

This results in a drift of $0.19\,mm/s$ and $0.011\,deg/s$ for the dense VO and $0.51\,mm/s$ and $0.031\,deg/s$ for fovis over the duration of the movement. We only count the time where the sensor is moving as both frameworks are keyframe based and should not drift locally as long as no new keyframes are generated.

It is noted that the RMS position and rotation error of fovis is in the same region as in the experimental validation of the original fovis paper [9]. We also evaluate the translational and orientational velocity estimates of both frameworks in a phase where the sensor is completely static. The velocity estimates are generated by taking finite differences of the pose estimate. We make sure that the keyframes for both frameworks are generated at the same time to have a fair comparison. After keyframe generation, the sensor is moved by a few cm and slightly pitched to generate a bit of relative motion to the keyframe to ensure sufficient image motion.

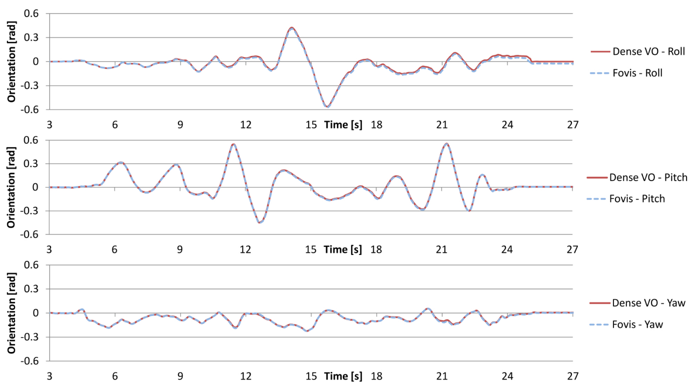

Fig. 4. Rotational estimate of the *slow* dataset by the fovis and the dense VO stereo framework.

The noise level of the sparse approach is significantly higher with an RMS error of $3.0\,mm/s$ for translation and $0.24\,deg/s$ tor rotational speed. The dense VO framework performs with an RMS error with $1.3\,mm/s$ for translation and $0.19\,deg/s$ rotation respectively. We believe that this is due to the fact that the minimum of the cost function of the dense framework is more distinctive and sharper than the sparse counterpart, thus yielding a more stable local minimum for the optimizer.

Please note that the local velocity noise level is larger than the drift over the whole experiment ($0.19\,mm/s$ to $3\,mm/s$ for the dense VO). Due to the fact that we employ a key-frame based state-estimator, this velocity noise is only accumulated into a drift of the whole experiment when new keyframes are generated.

2) Fast Experiment: The second, *fast* experiment is considerably jerkier than the first experiment with rotation rates up to $160\,deg/s$. This rapid and jerky motion induces significant motion blur, challenging the robustness of the state estimation frameworks. The video submission to the paper shows the full experiment, including the pose estimates and the live video stream.

As there are significant dropouts or failures of the fovis framework due to the fast and jerky motion, in Fig. 5, we depict the velocity estimates expressed in the body (sensor-) frame. This enables to evaluate the *relative*, local accuracy of the frameworks as failures or sudden jumps in the pose estimation affect the velocity estimates only locally. As in the first experiment we return the sensor to its initial state towards the end of the experiment to evaluate the amount of drift of the frameworks. For the sake of readability, in Fig. 5, only the first 14 seconds of the experiment are depicted.

From Fig. 5, it is clearly visible that the sparse framework has difficulties to properly handle yawing motions (rotation around the optical axis). Between $t = 4s$ and $t = 8s$, the sensor is undergoing significant yawing motion which causes the fovis framework to fail regularly while the dense VO framework seems to be nicely tracking the trajectory. As no ground truth is available, this is somewhat difficult to determine. However, it is noted that the dense VO framework does not fail during the full experiment and that the error over the full dataset in terms of drift of position and orientation with $0.25\,mm/s$ and $0.015\,deg/s$ is only slightly worse than the first, slow dataset. Additionally, it is noted that the velocity estimates of the fovis framework are significantly noisier than during slow motions, as seen around $t = 3s$ and between $t = 8s$ and $t = 13s$.

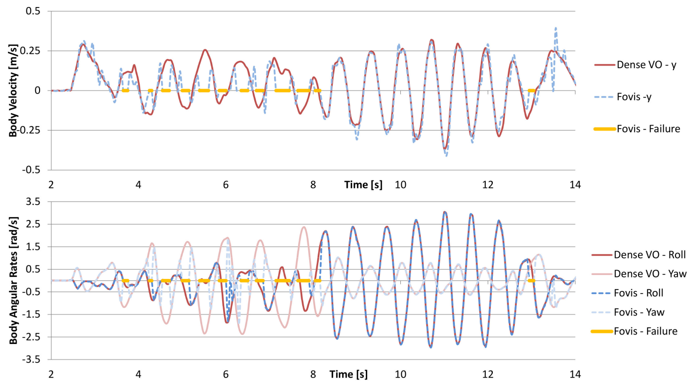

Fig. 5. Translational and rotational velocity estimates for the *fast* dataset. As there are significant dropouts or failures of the fovis framework due to the fast and jerky motion, we depict the velocity estimates expressed in the body (sensor-) frame. The bold, yellow line shows fovis failures detected by the framework itself. The figure only shows the first 14 s of the experiment - the full experiment is added as a video supplement.

F. Timings

In Fig. 6, the timings for the sparse and dense visual-odometry algorithms are depicted. The fovis framework requires an average of $22\,ms$ for the complete processing loop, including feature detection, extraction, matching, outlier rejection and motion estimation. This timing is in line with the original fovis publication [9]. The dense VO framework requires $29\,ms$ for the complete processing loop, when no keyframe is generated. The comparable processing times can be explained by the fact that significant time can be saved since we can avoid any kind of feature detection, extraction or matching due to the dense nature of the algorithm. As the inter-frame motion is relatively small, only very little image warpings have to be calculated as the dense approach will converge after a few iterations while going down the image scale pyramid.

Creating a new keyframe in the dense VO framework requires $69\,ms$. This includes the computation of the depth image using the block matcher which is fairly costly. Even when we use every frame as keyframe (e.g. frame-to-frame odometry), we still have a framerate of over $10\,Hz$. As we are running the algorithm on a dual-core processor, and we can easily outsource the keyframe generation to the second core, the algorithm runs at the $20\,Hz$ framerate of the stereo camera.

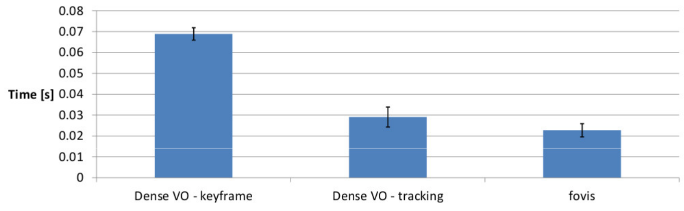

Fig. 6. Timings of sparse and dense visual odometry framework for the slow dataset. The error bars depict 1 sigma bounds. For the dense VO, we differentiate between tracking and keyframe generation as the latter requires the costly computation of the depth image using the block matching algorithm.

V. Conclusion

This paper presents a real-time dense mapping and pose estimation that enables robust state estimation even under jerky sensor motions while retaining a small computational footprint. We propose an odometry and mapping system that leverages the full photometric information from a stereo-vision system in a probabilistic framework and is running in real-time on a single low-power Intel CPU core. We present several approaches and simplifications on how to make this photometric, dense approach real-time capable. Additionally, we present a probabilistic formulation of the joint odometry estimation and mapping process that enables to efficiently reject *temporal* outliers in ego-motion estimation as well as *spatial* outliers in the mapping process. We adapt state-of-the-art ideas from dense RGBD state estimation and apply it to stereo-vision based state estimation which has to deal with significant challenges such as increased computational complexity (since the depth images have to be computed first) and more outliers in the depth image (when a low-complexity, local block matcher is used). The performance of the proposed scheme is evaluated in a hand-held setup against the state-of-the-art stereo-based VO framework fovis. It is shown that the dense scheme can cope with highly dynamic motions in situations where sparse approaches are prone to failure. Additionally, it is shown, that due to the structure of our dense VO algorithm, its computational footprint is comparable to state-of-the-art sparse VO approaches, making it an ideal choice as a navigation backend for weight- and power-restricted mobile robots.

References

[1] Markus Achtelik, Abraham Bachrach, Ruijie He, Samuel Prentice, and Nicholas Roy. Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments. *SPIE Defense, Security, and Sensing. International Society for Optics and Photonics*, 2009.

[2] Motilal Agrawal. Localization and mapping for autonomous navigation in outdoor terrains: A stereo vision approach. *WACV'07*, 2007.

[3] A.I. Comport, E. Malis, and P. Rives. Real-time Quadrifocal Visual Odometry. *The International Journal of Robotics Research*, 29, 2010.

[4] Andrew I Comport. Real-time dense appearance-based SLAM for RGB-D sensors. In *Australasian Conf. on Robotics and Automation.*, 2011.

[5] Andreas Geiger, Martin Roser, and Raquel Urtasun. Efficient large-scale stereo matching. *Computer Vision ACCV 2010*, pages 25 – 38, 2011.

[6] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. *IEEE Intelligent Vehicles Symposium (IV)*, 2011.

[7] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[8] A. Howard. Real-time stereo visual odometry for autonomous ground vehicles. *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2008.

[9] Albert S Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Dieter Fox, and Nicholas Roy. In *Int. Symposium on Robotics Research*.

[10] Christian Kerl, Jurgen Sturm, and Daniel Cremers. Robust odometry estimation for RGB-D cameras. *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[11] Kurt Konolige. Small vision systems: Hardware and implementation. *Robotics Research*, 1998.

[12] Janosch Nikolic, Joern Rehder, Michael Burri, Pascal Gohl, Stefan Leutenegger, Paul T Furgale, and Roland Siegwart. A Synchronized Visual-Inertial Sensor System with FPGA Pre-Processing for Accurate Real-Time SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[13] Korbinian Schmid and Heiko Hirschmuller. Stereo vision and IMU based real-time ego-motion and depth image computation on a handheld device. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.