Towards Fully Dense Direct Filter-Based Monocular Visual-Inertial Odometry

Alexander Hardt-Stremayr and Stephan Weiss

***Abstract*— We propose a fully dense direct filter-based visual-inertial odometry method estimating both pixel depth for all pixels and robot state simultaneously, having all uncertainties in the same state vector. Due to the fully dense method, our approach works even in low-textured areas with very low, smooth gradients (i.e. scenes where feature based or semi-dense approaches fail). Our algorithm performs in real-time on a CPU with a time complexity linearly dependent on the amount of pixels in the provided image. To achieve this, we propose complexity reduction methods for fast matrix inversion, exploiting specific structures of the covariance matrix. We provide both simulated and real-world results in low-textured areas with a smooth gradient.**

I. Introduction

Visual-inertial odometry (VIO) methods combine fast inertial measurement readings (IMU) with camera images at a lower frequency to estimate an internal robot state.

In this paper, we present a novel fully dense tightly coupled direct filter-based approach with a computational upper bound of $O(nm^2)$ instead of $O(n^3)$ on a CPU, with n being the number of processed pixels, m begin the size of the robot state and $n \gg m$. We fuse IMU and camera data in an Extended Kalman Filter (tightly coupled EKF) and operate directly on image intensities to minimize the photometric error instead of using features. Our fully dense approach processes all pixels in an image, contrary to using only high-gradient areas (semi-dense) or single pixels (sparse). By adding both depth and intensity to the state vector, we are able to update all entries in a single step, inherently including map uncertainty in the probabilistic estimation. This is in contrast to algorithms which iteratively estimate pose change and scene depth in two repeated steps with disjoint uncertainty information.

Especially in low-texture areas, direct methods minimizing the photometric error are more robust than indirect methods extracting features [1]. Thus, we specifically tested our approach on low-texture areas with smooth gradients as seen in Fig. 1, showing promising results. Based on this proof of concept, this approach can already be used to bridge short featureless areas with low gradients.

Alexander Hardt-Stremayr and Stephan Weiss are with the Department of Smart Systems Technologies in the Control of Networked Systems Group, Alpen-Adria-Universität Klagenfurt, 9020 Klagenfurt, Austria alexander.hardt-stremayr@aau.at, stephan.weiss@ieee.org

The research leading to these results has received funding from the ARL within the BAA W911NF-12-R-0011 under grant agreement W911NF-16-2-0112 and from the Austrian Ministry for Transport, Innovation and Technology (BMVIT) under the grant agreement 855468 (Forest-IMATE).

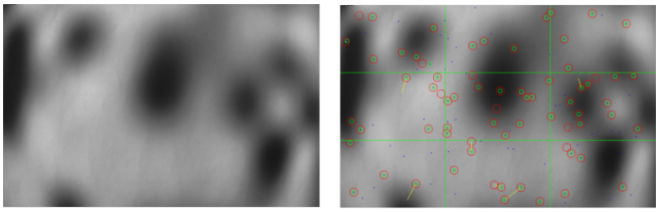

Fig. 1. Left: Camera view of a scenery with smooth intensity gradients, input for our algorithm. Right: No or wrong features found by a feature detector.

Our main contributions are

- Fully dense handling of camera information with combined single-step simultaneous pose and depth estimation.
- Complexity reduction methods to fuse the information on a CPU in real-time.
- State estimation on featureless areas with smooth gradients, different to classical approaches that need strong corners, lines or gradients.

II. Related Work

In their core, direct semi-dense and dense Visual Odometry methods are built upon the dense optical flow. By comparing the predicted intensity value with the measured intensity value and multiplying this residual with the gradient and a subsequent state Jacobian, it is possible to estimate the state. In semi-dense approaches, high-gradient pixels are selected for estimation, while dense approaches use all pixels.

Additionally, as the gradient is a linearization of a potentially highly nonlinear image, the pixel offset should be smaller than one pixel. This is commonly achieved by using an image pyramid. Visual-Inertial Odometry methods incorporate the measurements provided by an IMU to improve the predicted state and to decrease the pixel coordinate offset. However, to predict the expected value, knowledge of the scene depth is needed. Different ways to handle this problem have been introduced in the past.

Using a stereo with known baseline, depth estimation is reduced to a line search and can be accomplished in real-time on a CPU as seen in [2] and [3]. Similar to stereo, a RGB-D camera provides depth information. This has been used to provide dense visual odometry [4]–[6] on a CPU. DTAM [7] popularized GPU-based trajectory and depth estimation. Recent work added inertial measurements [8], [9]. KinectFusion [10] provided GPU-based processing of RGB-D information. Among different extensions [11], [12], integration of IMU information has also been published

[13]. Deep-learning based depth estimation approaches as e.g. [14]–[16] do not yet integrate IMU readings. Two-step algorithms estimate pose change and depth interatively, having disjoint uncertainty information. Beside semi-dense [17] and sparse [18] visual odometry approaches, a pseudo-dense (semi-dense among gradients [19], superpixels for monotonous regions [20]) visual-inertial odometry algorithm [21] has been published. One-step algorithms like ours estimate both depth and pose change pixel intensities. However, existing algorithms have conceptual overlap to feature-based approaches as a set of regions is selected and depth is estimated for each one similarly to sparse features [22]–[24].

To our knowledge, no one-step fully dense direct CPU-based visual-inertial odometry algorithm exists as of now.

III. Dense Direct Visual-Inertial Odometry

A. State Representation

The state of our system is composed of the position ${}_W\boldsymbol{p}_{WI}$ of the IMU in the world frame $\mathcal{W}$, its velocity ${}_W\mathbf{v}_{WI}$ and its attitude quaternion $\bar{\mathbf{q}}_{WI}$, describing a rotation from the world frame $\mathcal{W}$ to the IMU frame $\mathcal{I}$. The state of the IMU sensor is given as its angular velocity bias $\mathbf{b}_{\boldsymbol{\omega}}$ and its linear acceleration bias $\mathbf{b}_{\boldsymbol{\omega}}$. The camera extrinsics are given as the position ${}_I\boldsymbol{p}_{IC}$ of the camera center in the IMU frame and its attitude quaternion $\bar{\mathbf{q}}_{IC}$. In this document, those states are summarized as core states $\mathbf{x}_c$.

$$\mathbf{x}_c = \begin{bmatrix} {}_W\boldsymbol{p}_{WI} & {}_W\mathbf{v}_{WI} & \bar{\mathbf{q}}_{WI} & \mathbf{b}_{\boldsymbol{\omega}} & \mathbf{b}_{\mathbf{a}} & {}_I\boldsymbol{p}_{IC} & \bar{\mathbf{q}}_{IC} \end{bmatrix} \quad (1)$$

Since we compare predicted pixel intensities with measured pixel intensities and the predicted image depends on the depth of the scene, we keep both (inverse) depth per pixel $\boldsymbol{\rho}$ and intensity values $\boldsymbol{i}$ in the state.

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_c & \boldsymbol{\rho} & \boldsymbol{i} \end{bmatrix}^T \quad (2)$$

We use an error state EKF and the error rotation and its covariance as an error quaternion $\widetilde{\bar{\mathbf{q}}} = [\frac{1}{2}\delta\boldsymbol{\theta}^T \;\; 1]^T$ to handle the quaternion in its minimal representation. The error state $\widetilde{\mathbf{x}}$ is:

$$\widetilde{\mathbf{x}} = \begin{bmatrix} \widetilde{\mathbf{x}}_c & \widetilde{\boldsymbol{\rho}} & \widetilde{\boldsymbol{i}} \end{bmatrix}^T \quad (3)$$

For a more thorough description of the error state $\widetilde{\mathbf{x}}_c$ and its application see [25].

B. Process Model

For brevity, we refer to [25] for both continuous and discretized equations for the core states $f(\mathbf{x}_c)$ as well as the description of the corresponding process noise matrix $Q_{\mathbf{x}_c}$. The derivation of the dynamics $\dot{\boldsymbol{\rho}} = f_{\boldsymbol{\rho}}(\mathbf{x})$ and $\dot{\boldsymbol{i}} = f_{\boldsymbol{i}}(\mathbf{x})$ is based on the pixel propagation model. Every pixel intensity i is the result of a look-up of image Img at the position p_{Img} with the homogeneous coordinates $\begin{bmatrix} u & v & 1 \end{bmatrix}^T$. Using the camera matrix K and the inverse depth of the pixel ρ, the pixel coordinate can be calculated based on the three-dimensional position ${}_C\boldsymbol{p}_{CP}$ of the point in the camera frame. ρ is used to scale the coordinate back to a depth of 1, therefore $\rho = \frac{1}{e_z^T\,{}_C\boldsymbol{p}_{CP}}$ with $e_z = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$.

$$i = Img(p_{Img}) = Img([u\ v\ 1]) = Img(K\,{}_C\boldsymbol{p}_{CP}\rho) \quad (4)$$

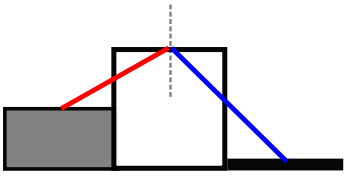

Fig. 2. Image intensities shown as three buckets and corresponding gradients (red, blue). If the prediction is slightly left of the dotted line (red) a correction has opposite sign compared to a prediction right (blue) of the line. To mitigate this, we include a gradient dependent uncertainty term.

The dynamic of the intensity i is then

$$\dot{i} = \frac{\partial Img(p_{Img})}{\partial p_{Img}} \frac{\partial p_{Img}}{\partial t} \quad (5)$$

The first part is a spatial derivation of the intensity value, also known as image gradient $\boldsymbol{g}$.

$$\dot{i} = \boldsymbol{g}K({}_C\dot{\boldsymbol{p}}_{CP}\rho + {}_C\boldsymbol{p}_{CP}\dot{\rho}) \quad (6)$$

$$\dot{\rho} = -\frac{e_z^T\,{}_C\dot{\boldsymbol{p}}_{CP}}{(e_z^T\,{}_C\boldsymbol{p}_{CP})^2} = -e_z^T\rho^2\,{}_C\dot{\boldsymbol{p}}_{CP} \quad (7)$$

Based on $\dot{\boldsymbol{\rho}}$ and ${}_C\boldsymbol{p}_{CP} = K^{-1}p_{Img}\frac{1}{\rho}$, $\dot{i}$ can be reformulated.

$$\dot{i} = \boldsymbol{g}K(I_{d_{3\times 3}} - K^{-1}p_{Img}e_z^T)\rho_C\dot{\boldsymbol{p}}_{CP} \quad (8)$$

The position of a point P as seen by the camera in world coordinates can be described by the concatenation of all translations between world frame $\mathcal{W}$, inertial frame $\mathcal{I}$ and camera frame $\mathcal{C}$ expressed in the world frame. $\mathbf{C}_{WI}$ is the rotation matrix based on quaternion $\bar{\mathbf{q}}_{WI}$.

$${}_W\boldsymbol{p}_{WP} = {}_W\boldsymbol{p}_{WI} + \mathbf{C}_{WI}\,{}_I\boldsymbol{p}_{IC} + \mathbf{C}_{WI}\mathbf{C}_{IC}\,{}_C\boldsymbol{p}_{CP} \quad (9)$$

The time derivative of this expression is

$$\begin{aligned} {}_W\dot{\boldsymbol{p}}_{WP} =& {}_W\dot{\boldsymbol{p}}_{WI} + \dot{\mathbf{C}}_{WI}\,{}_I\boldsymbol{p}_{IC} + \mathbf{C}_{WI}\,{}_I\dot{\boldsymbol{p}}_{IC} + \\ & \dot{\mathbf{C}}_{WI}\mathbf{C}_{IC}\,{}_C\boldsymbol{p}_{CP} + \mathbf{C}_{WI}\dot{\mathbf{C}}_{IC}\,{}_C\boldsymbol{p}_{CP} + \mathbf{C}_{WI}\mathbf{C}_{IC}\,{}_C\dot{\boldsymbol{p}}_{CP} \end{aligned} \quad (10)$$

As the world coordinate of the point is assumed to be stationary ${}_W\boldsymbol{p}_{WP}$ and both attitude $\mathbf{C}_{IC}$ and translation ${}_I\boldsymbol{p}_{IC}$ of the camera with respect to the IMU do not change over time, the corresponding time derivatives can be set to zero. This will yield

$$0 = {}_W\dot{\boldsymbol{p}}_{WI} + \dot{\mathbf{C}}_{WI}\,{}_I\boldsymbol{p}_{IC} + \dot{\mathbf{C}}_{WI}\mathbf{C}_{IC}\,{}_C\boldsymbol{p}_{CP} + \mathbf{C}_{WI}\mathbf{C}_{IC}\,{}_C\dot{\boldsymbol{p}}_{CP} \quad (11)$$

Reformulating and replacing ${}_W\dot{\boldsymbol{p}}_{WI}$ by ${}_W\mathbf{v}_{WI}$, it is possible to derive the change of the point P in the camera frame over time.

$${}_C\dot{\boldsymbol{p}}_{CP} = -\mathbf{C}_{IC}{}^T\mathbf{C}_{WI}{}^T({}_W\mathbf{v}_{WI} + \dot{\mathbf{C}}_{WI}\,{}_I\boldsymbol{p}_{IC} + \dot{\mathbf{C}}_{WI}\mathbf{C}_{IC}\,{}_C\boldsymbol{p}_{CP}) \quad (12)$$

C. Process Noise

The noise for $\mathbf{x}_c$ originates from IMU noise and both its derivation and discretization can be found in [25]. The intensity change rate $\dot{i}$ is linearly dependent on the gradient $\boldsymbol{g}$, assuming $\boldsymbol{g}$ is an exact linearization of the underlying model. However, as Fig. 2 shows, the assumed image intensity function can be highly nonlinear or even discontinuous. When using a simple linear model, the sign of the correction

may depend on whether the left or right gradient of the current pixel is used. This problem is mitigated when using a different interpolation model, e.g. a larger kernel for convolution, but it is not entirely removed.

Instead, we calculate an image gradient based uncertainty. P_g is the gradient covariance matrix. $\sigma_{g_x}^2$ and $\sigma_{g_y}^2$ are the variances of the gradient in x and y-direction respectively, based on either the left and right or the upper and lower pixel of the current one. Similarly, $\sigma_{g_{xy}}$ is the calculated covariance between the gradients.

$$P_g = \begin{bmatrix} \sigma_{g_x}^2 & \sigma_{g_{xy}} \\ \sigma_{g_{xy}} & \sigma_{g_y}^2 \end{bmatrix} \tag{13}$$

The goal is to efficiently include predicted intensity uncertainty arising from uncertain motion propagation. The dynamics of a pixel coordinate $\frac{d}{dt}p_{Img}$ can be integrated over time to yield Δp_{Img}. The Jacobian $F_{px} = \frac{\partial p_{Img}}{\partial \mathbf{x}}$ of the corresponding dynamic model is used to propagate the covariance of the derived state, stored in P, to the covarinace of Δp_{Img}, stored in the newly calculated covariance matrix P_p with p being the subscript denoting Δp_{Img}: $P_p = F_{px} P F_{px}^T$.

$\dot{i}$ may be formulated as $\dot{i} = \boldsymbol{g}\frac{d}{dt}p_{Img}$. Similarly, $\Delta i = \boldsymbol{g}\Delta p_{Img}$. To calculate the discrete intensity process noise variance Q_{ii} for a single pixel, both covariance matrices are combined using multiplicative uncertainty propagation.

$$Q_{ii} = \Delta p_{Img}{}^T P_g \Delta p_{Img} + \begin{bmatrix} 1 & 1 \end{bmatrix} P_p P_g \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{14}$$

To fully complete the multiplicative uncertainty propagation, $\boldsymbol{g} P_p \boldsymbol{g}^T$ would also have to be added. However, this is implicitly covered in the covariance prediction step of the EKF $P^- = FPF^T + Q$.

D. Update Model

On a camera reading of a grayscale image, the image is first undistorted and then provided as measurement input. As the image intensities have been predicted using the estimated depth and the IMU inputs as shown in Eq. (8), the residual y is the difference between the measured pixel intensity z and predicted intensity $\hat{i}$. n_z is the measurement noise of a single pixel intensity:

$$y = z - \hat{i} - n_z \tag{15}$$

The measurement noise for each pixel is assumed to be independent from the other pixels. Therefore, the measurement noise matrix R is given as

$$R = I_{d_{n\times n}} n_z^2 \tag{16}$$

E. Discretization and Linearization

The process Jacobian $F = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ is divided into multiple parts: we use the indices x for $\mathbf{x}_c$, d for $\boldsymbol{\rho}$ and i for $\boldsymbol{i}$.

$$F = \begin{bmatrix} F_{xx} & 0 & 0 \\ F_{dx} & F_{dd} & 0 \\ F_{ix} & F_{id} & F_{ii} \end{bmatrix} \tag{17}$$

The derivation of F_{xx} can be found in [25]. Similarly, for both $\dot{\boldsymbol{\rho}} = f_{\boldsymbol{\rho}}(\mathbf{x})$ and $\dot{\boldsymbol{i}} = f_{\boldsymbol{i}}(\mathbf{x})$, a first-order discretization is performed and higher order terms are omitted. Due to space limitations, we only give two simple examples, $\frac{\partial \dot{\rho}}{\partial_{W}\mathbf{v}_{WI}}$ and $\frac{\partial \dot{i}}{\partial_{W}\mathbf{v}_{WI}}$, how entries of $F = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ may be computed:

$$\frac{\partial \dot{\rho}}{\partial_{W}\mathbf{v}_{WI}} = -e_z^T \rho^2 (-\mathbf{C}_{IC}{}^T \mathbf{C}_{WI}{}^T) \Delta t \tag{18}$$

$$\frac{\partial \dot{i}}{\partial_{W}\mathbf{v}_{WI}} = \boldsymbol{g} K (I_{d_{3\times 3}} - K^{-1} p_{Img} e_z^T) \rho (-\mathbf{C}_{IC}{}^T \mathbf{C}_{WI}{}^T) \Delta t \tag{19}$$

The Jacobian for the intensity measurement $H = \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}$ is

$$H = \begin{bmatrix} 0_{n\times m} & 0_{n\times n} & I_{d_{n\times n}} \end{bmatrix} \tag{20}$$

F. Initialization

Position ${}_{W}\boldsymbol{p}_{WI}$, velocity ${}_{W}\mathbf{v}_{WI}$ and angular velocity bias $\mathbf{b}_{\boldsymbol{\omega}}$ are initialized with zero values assuming a still phase. For the rotation $\bar{\mathbf{q}}_{WI}$, gravity alignment is performed on a set of IMU readings and the remaining acceleration is stored as linear acceleration bias $\mathbf{b}_{\mathbf{a}}$. For the covariance of ${}_{W}\boldsymbol{p}_{WI}$, the entries are set to be very small while for the other covariance entries, values corresponding to the expected initial error are chosen. $\boldsymbol{i}$ is initialized with 0.5, assuming that pixel intensities are scaled to 0 and 1. $\boldsymbol{\rho}$ is initialized with random values. For both, the initial variance is set to a large value. Similarly, whenever a pixel in the current image was not yet seen before, the newly inserted intensity and depth entries are initialized in the same way.

This describes the complete fully dense direct tightly-coupled EKF able to estimate both internal robot state as well as inverse scene depth. Naive implementation would not allow real-time operation on a CPU. In the following section we discuss our approach on complexity reduction.

IV. Complexity Reduction

If the measurement size n is far larger than the state size m, in an EKF, the most costly operation is the inversion of the innovation matrix S with a complexity of $O(n^3)$. When considering all pixels in an image, this would render the approach intractable in practice on a CPU for all image sizes larger than a few pixels. Thus, in the following, we detail a set of assumptions, so the complexity can be reduced to $O(nm^2)$ with m being the size of $\mathbf{x}_c$, enabling real-time use on a CPU.

A. Fast State Propagation

Although both intensity and inverse depth vectors are stored in the state vector, only $\mathbf{x}_c$ is propagated upon IMU readings for performance reasons. On each camera reading, the state change introduced by IMU readings is accumulated (i.e. integrated) and the pixel propagation is calculated before comparing the intensity with the camera measurement. This is comparable to keeping the map in two-step algorithms. However, in our case it is embedded into the probabilistic framework with a shared uncertainty. The rotational change between first and second camera measurement seen from the

frame of the second measurement is calculated in $\mathbf{C}_{21}$ by iteratively applying IMU propagation steps. Likewise, ${}_2\boldsymbol{p}_{21}$ is the same for the translational movement. Similar to IMU-preintegration methods, we also calculate integrated F and Q matrices. Assuming that between camera measurements 1 and 2, additional IMU measurements have occurred, the integrated F_{xx} and Q_{xx} matrices can be calculated as repeated application of different $F_{xx_{ij}}$ and $Q_{xx_{ij}}$ matrices between every two measurements i and j. Propagation of inverse depths and intensities can be calculated as follows:

$$p_{Img_2} = K\frac{\mathbf{C}_{21}K^{-1}p_{Img_1}\rho_1^{-1} + {}_2\boldsymbol{p}_{21}}{\rho_2^{-1}} \tag{21}$$

$$\rho_2 = e_z^T(\mathbf{C}_{21}K^{-1}p_{Img_1}\rho_1^{-1} + {}_2\boldsymbol{p}_{21}) \tag{22}$$

The new pixel position defines the location in both intensity and inverse depth vectors. As the location corresponds to a position on the pixel grid and only allows whole numbers, the resulting position is rounded as seen in Fig. 3 and the intensity is interpolated. The remainder of the rounding

Fig. 3. A pixel intensity value is transferred from the original image position to the predicted image position. As only pixel centers can be destinations, the propagated intensity value is shifted by multiplying the pixel coordinate offset (red arrow) with the gradient.

is multiplied with the gradient to predict the target pixel intensity. If multiple pixels would be propagated onto the same position, the pixel with the smaller inverse depth variance is used. In Eq. (23), the operator $[\cdot]$ is the two-dimensional nearest integer function.

$$i_2 = i_1 - \boldsymbol{g}\Delta p, \Delta p = p_{Img_2} - [p_{Img_2}] \tag{23}$$

As the intensity value is modified using the potentially noisy gradient, an additional factor ΔQ_{ii} is added to the intensity uncertainty based on the gradient covariance matrix P_g.

$$\Delta Q_{ii} = \Delta p^T P_g \Delta p \tag{24}$$

By using the gradient based interpolation, at most one entry per row and column is set in F_{dd} and F_{ii}, corresponding to the coordinate of p_{Img_1} as the column index and the coordinate of p_{Img_2} as the row index. Entries in both F and P are swapped so that F_{dd} and F_{ii} (and subsequently F_{id}) are diagonal matrices. Ensuring this diagonal property is a critical step for the fast matrix inversion shown in section IV-D.

B. Covariance Matrix Assumptions

The covariance matrix P can be divided into a set of smaller matrices, corresponding to $\mathbf{x}_c$ as x, $\boldsymbol{\rho}$ as d and $\boldsymbol{i}$ as i similar to F in III-E. To enable fast matrix inversion, the covariance matrix P has to conform to a certain structure after any covariance update as shown in Eq. (25).

$$P = \begin{bmatrix} P_{xx} & P_{xd} & 0 \\ P_{dx} & P_{dd} & 0 \\ 0 & 0 & P_{ii} \end{bmatrix} \tag{25}$$

Using concatenated F and Q matrices, the filter process can be adapted to a single propagation step followed by a single update step with respect to the inverse depth $\boldsymbol{\rho}$ and intensity $\boldsymbol{i}$ parts of the state and covariance matrix. Based on our simulations using a full covariance matrix P, covariances between image intensities and the remainder of the state vector are negligible and can be approximated to be zero leading to the structure in Eq. 25. Similarly, the off-diagonal entries of P_{ii} are also approximated as zero. P_{xx}, P_{xd} and P_{dx} are stored as dense matrices. For P_{dd}, the diagonal component of the matrix is stored while off-diagonal entries are approximated using matrix rank reduction for rank k to lower storage size and execution time. With $k \leq m$, the upper computational complexity bound is ensured. For results in this paper, we set the off-diagonal entries of P_{dd} to zero ($k = 0$) without significant quality loss but noticeable speed increase. This may of course introduce filter inconsistencies which is subject to further investigations. With those assumptions, supported by our simulations, only P_{xx} and P_{xd} with the corresponding transpose remain block matrices. This is another important property for the fast matrix inversion.

C. Fast Matrix Inversion

Based on the substructure of the covariance matrix shown above, we present a fast matrix inversion which is algebraically exact. With the concatenated F and Q matrices, the covariance matrix P^- corresponding to the predicted state at the time of the update can be calculated using the covariance matrix P after the last update:

$$P^- = FPF^T + Q \tag{26}$$

While we build upon a certain substructure in P as described above, no such substructure assumptions hold for P^- which may be a fully dense matrix. As one of our contributions in this paper, P^- is never calculated in its entirety enabling real-time computation despite its density. Instead, fast update equations are derived based on the definition of P^- and used directly.

The innovation matrix S is defined as $S = HPH^T + R$. As H has been derived to be $\begin{bmatrix} 0 & 0 & I_{d_{n\times n}} \end{bmatrix}$ in Eq. (20) and based on the structure of P^-, S has the following definition:

$$\begin{aligned} S =& F_{ix}P_{xx}F_{ix}^T + F_{id}P_{dd}F_{id}^T + F_{ii}P_{ii}F_{ii} + \\ & F_{ix}P_{xd}F_{id}^T + F_{id}P_{xd}^T Fix^T + Q_{ii} + R \end{aligned} \tag{27}$$

Except for F_{ix} and P_{xd}, which are a $n \times m$ matrices, P_{xx}, which is a $m \times m$ matrix and P_{dd}, which is a combination of a diagonal and $n \times k$ matrices, all used matrices are diagonal matrices. This is the base for the $O(nm^2)$ inversion. This can be reformulated as $S = ABA^T + D$ with

$$\begin{aligned} A &= \begin{bmatrix} F_{ix} & F_{id}P_{xd} & P_{ddA} \end{bmatrix}, \\ B &= \begin{bmatrix} P_{xx} & I_{d_{m\times m}} & \\ I_{d_{m\times m}} & & \\ & & P_{ddB} \end{bmatrix}, \end{aligned} \tag{28}$$

P_{ddA} and P_{ddB} as rank-reduced dense matrices and D summing up all diagonal matrices including the diagonal

component of P_{dd}. A further reformulation yields

$$S = \sqrt{D^{-1}}(I_{d_{n\times n}} + \sqrt{D^{-1}}ABA^T\sqrt{D^{-1}})\sqrt{D^{-1}} \quad (29)$$

To perform a fast matrix inversion, we use the identity of

$$(U\Lambda U^T + I_{d_{n\times n}})^{-1} = U(\Lambda + I_{d_{n\times n}})^{-1}U^T \quad (30)$$

assuming $UU^T = U^TU = I_{d_{n\times n}}$. With a QR-decomposition of $QR = qr(D^{-0.5}A)$,

$$QRBR^TQ^T = D^{-0.5}ABA^TD^{-0.5} \quad (31)$$

As A has a rank of $2m + k$, only $2m + k \times 2m + k$ entries of R will be filled. Furthermore, RBR^T will be a symmetric matrix, so a singular value decomposition $U, \Lambda, V^T = svd(RBR^T)$ will yield $UV^T = U^TV = I$.

Combining QR decomposition and singular value decomposition yields

$$QU\Lambda V^TQ^T = D^{-0.5}ABA^TD^{-0.5} \quad (32)$$

with $QUV^TQ^T = I_{d_{n\times n}}$. Therefore, the inverted innovation matrix is

$$S^{-1} = D^{-0.5}QU(\Lambda + I_{d_{n\times n}})^{-1}U^TQ^TD^{-0.5} \quad (33)$$

As both Λ and $I_{d_{n\times n}}$ are diagonal matrices, inversion is done in linear time.

However, $U(\Lambda + I_{d_{n\times n}})^{-1}U^T$ still has a computational complexity of O(n^3). To enable a speed-up, Q needs to be cut to a $n\times m$ matrix Q_c. Similarly, R_c is the upper triangular rectangular matrix based on the first entries of R.

This would lead to information loss as while $Q_c^TQ_c = I_{d_{m\times m}}$ still holds, $Q_cQ_c^T$ does not evaluate to the identity matrix anymore. By reformulating the equation it is possible to prevent this problem and to ensure fast algebraic exact inversion.

U_c, Λ_c and V_c^T are calculated using the singular value decomposition of $R_cBR_c^T$ and will be square matrices (diagonal in the case of Λ_c) with a size of $m \times m$.

By adding $-D^{-1} + D^{-1}$ and expanding it to $-D^{-0.5}QQ^TD^{-0.5} + D^{-1}$, it is possible to adapt S^{-1} into the formulation given in Eq. (34) and (35).

$$S_{inner} = U_c(\Lambda_c + I_{d_{2m+k\times 2m+k}})^{-1}U_c^T - I_{d_{2m+k\times 2m+k}} \quad (34)$$

$$S^{-1} = D^{-0.5}Q_cS_{inner}Q_c^TD^{-0.5} + D^{-1} \quad (35)$$

Again, and similar to P^-, S is never calculated directly. Instead, it is only used to derive fast update equations.

D. Fast Update Calculation

The Kalman gain K is defined as $K = P^-H^TS^{-1}$. Based on equations 26 for P^- and 20 for H, the gain K can be calculated for all parts of the state. Equation 36 shows the content of the K component for the core state $\mathbf{x}_c$.

$$K_x = (F_{xx}P_{xx}F_{ix}^T + F_{xx}P_{xd}F_{id}^T)S^{-1} \quad (36)$$

By doing repeated right-side multiplication with the residual y, we ensure the intermediate result to always be a vector of at most size n. This ensures a time complexity of $O(nm)$.

The covariance update is calculated independently for P_{xx}, P_{dx}, P_{dd} and P_{ii} due to the structure of P^-. Other entries are not calculated as they are assumed to be zero as discussed in section IV-B. Similar to the state update, certain properties of the matrix structure can be used to simplify the equation and to preserve the upper bound of $O(nm^2)$ for calculation of the updated covariance matrix $P^+ = P^- - P^U$.

$$\begin{aligned} P_{xx}^U =& F_{xx}P_{xx}F_{ix}^TS^{-1}F_{ix}P_{xx}F_{xx}^T + \\ & F_{xx}P_{xd}F_{id}^TS_{-1}F_{id}P_{xd}^TF_{xx}^T \end{aligned} \quad (37)$$

Although S^{-1} is a $n \times n$ matrix, it is possible to multiply either F_{ix} or $P_{xd}F_{id}^T$ with the Q_c component, resulting in a $m \times m$ matrix. By repeating this inner matrix multiplications and by using the diagonal property of the D^{-1} matrix, the update calculations for P_{xx} and P_{xd} never exceed $O(nm^2)$. For P_{ii}^U and the diagonal component of P_{dd}^U, only diagonal update entries are calculated as we assume that off-diagonal entries are zero. Similar to P_{xx}, inner matrix multiplications are repeated until only the outer matrices with $m \times n$ and $m \times n$ remain. Diagonal results are extracted and in case of off-diagonal P_{dd} entries, rank-reduced intermediate matrix multiplication products are stored.

V. RESULTS

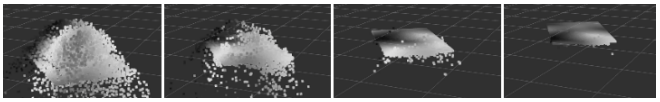

Fig. 4. Estimated pixel depths projected into world frame as a point cloud: The sequence shows the first few updates on the depth estimation and the rapid convergence over the planar scene (synthetic data).

Results presented in this section have been generated on a 2.2 GHz Intel i5-5200U. With non-optimized code and a downscaled image resolution of 94×60 running in a single thread, calculations currently take 125ms per frame. This calculation time scales linearly with the image resolution.

Due to space limitation we largely omit well behaving results from our simulations with synthetic data and directly present our preliminary results with real-world data. Most interesting (and probably best reflecting the (good) performance of our fully dense approach) is the estimated depth per pixel. In Fig. 4, we project the estimated pixel depth as a point-cloud into the world reference frame using the estimated states. The image sequence in the Figure shows this point-cloud soon after the (random initialization) and its rapid convergence towards the simulated planar ground. Fig. 5 shows a similar sequence, but with real-world data, demonstrating the performance of our algorithm in a real environment with real hardware. Smaller images in this Figure additionally show the down-scaled input image, colorized depth estimates per pixel and uncertainty of this estimate. The convergence is well reflected in the increased portions with lower uncertainty (red-ish areas in the small images).

Fig. 6 also shows a similar experiment, but this time with real-world data including a three-dimensional object (i.e. motion estimation over non-flat ground). The depth image

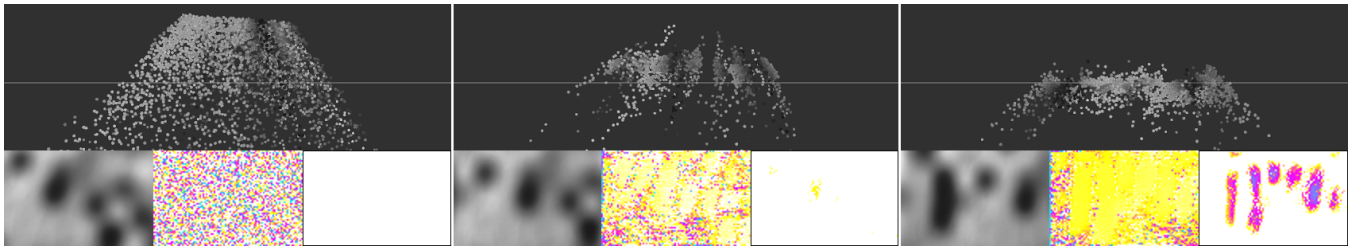

Fig. 5. Similar Figure as Fig. 4 but with real world data over a planar scene: Pixel depth convergence is shown during the first few updates. The top row shows the pixel depth estimates projected as point-cloud in the world frame (left: soon after initialization, middle and right: gradually converging). Bottom row shows for each step: left: the camera image, middle: estimated depth color coded, right: estimated uncertainty for the depth at each pixel (white means large uncertainty).

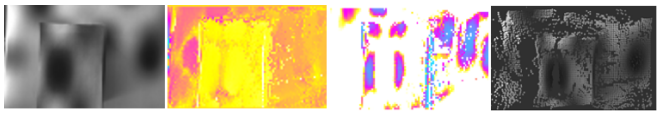

Fig. 6. Motion estimation over non-flat scene: The 3D object (box) is correctly reconstructed within our combined approach to simultaneously estimate the system motion and scene geometry in the same state vector. Left: camera image, middle left: estimated depth color coded, middle right: depth uncertainty, right: top view of reconstructed point cloud. (real-world data)

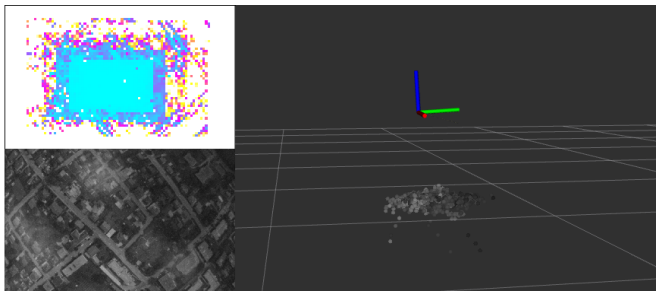

Fig. 7. Depth and motion estimation during take-off on real-world data. Upper left: depth estimation has a lower uncertainty compared to the border regions as the system moves upwards and the border pixels continue to see new regions. Bottom right: camera image. Right: estimated system pose as RGB-tripod and projected (already converged) point-cloud from estimated pixel depths.

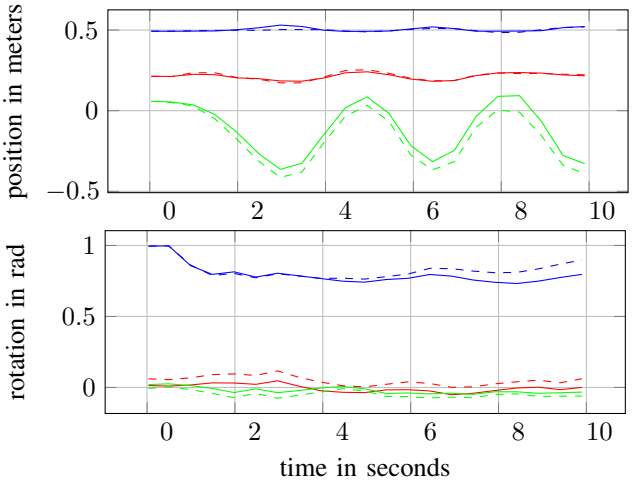

Fig. 8. Estimation of a real-world translational with slight rotational trajectory. The full lines are ground truth (optitrack), the dashed lines are the estimation. Red is the position in x direction in the upper plot and the roll in the lower plot. Likewise green for y and pitch and blue for z and yaw.

(middle left) and point cloud (right) reflect well the perceived environment (left) with outliers having a high uncertainty (middle right).

Fig. 7 is based on real-world data with regular texture and high frequencies in the camera image. Naturally, feature based approaches would also work with such data. The sequence depicts a "take-off" scene where the system moves upwards. Thus, the uncertainty (top left) is lowest in the image center where pixels were already perceived in previous readings the scene. Conversely, at the borders, the uncertainty is large as new scene elements enters the image. The Figure also shows the point cloud already well converged to the planar scene.

Lastly, Fig. 8 shows the estimated pose of the system during a trajectory with motion along the y axis. We observe a position drift of $< 3\%$ which is slightly larger than classical visual-inertial estimators ($< 1\%$) with the difference of being able to use low-texture, low gradient images and to estimate all pixel depths of the image in a fully dense fashion. We omitted evaluation of other approaches on this data set as both feature based and high gradient based approaches have difficulties or fail to extract meaningful information from low gradient areas as shown in Fig. 1.

VI. Conclusion

We demonstrated an initial approach towards a fully dense direct filter-based monocular visual-inertial odometry approach running in real time on a CPU estimating both robot state and inverse depth simultaneously within a shared covariance matrix, using a certain covariance matrix sub-structure.

This approach may be used to bridge short featureless trajectories with reasonable drift. Longer trajectories as well as drift reduction and robustness improvement is subject to further work.

We have shown that this approach works for a selected set of real-world data. This set has been chosen to be difficult for classical approaches. The approach converges correctly and is able to detect depth differences introduced by a three-dimensional object.

REFERENCES

[1] N. Yang, R. Wang, X. Gao, and D. Cremers, "Challenges in monocular visual odometry: Photometric calibration, motion bias, and rolling shutter effect," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2878–2885, 2018.

[2] Y. Ling and S. Shen, "Dense visual-inertial odometry for tracking of aggressive motions," in *Robotics and Biomimetics (ROBIO), 2015 IEEE International Conference on*. IEEE, 2015, pp. 576–583.

[3] V. Usenko, J. Engel, J. Stückler, and D. Cremers, "Direct visual-inertial odometry with stereo cameras," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1885–1892.

[4] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for rgb-d cameras," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3748–3754.

[5] C. Kerl, J. Stuckler, and D. Cremers, "Dense continuous-time tracking and mapping with rolling shutter rgb-d cameras," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2264–2272.

[6] M. Kuse and S. Shen, "Robust camera motion estimation using direct edge alignment and sub-gradient method," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 573–579.

[7] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2320–2327.

[8] K. Qiu and S. Shen, "Model-aided monocular visual-inertial state estimation and dense mapping," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 1783–1789.

[9] Z. Yang, F. Gao, and S. Shen, "Real-time monocular dense mapping on aerial robots using visual-inertial fusion," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 4552–4559.

[10] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison *et al.*, "Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 2011, pp. 559–568.

[11] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, "Real-time large-scale dense rgb-d slam with volumetric fusion," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 598–626, 2015.

[12] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "Elasticfusion: Real-time dense slam and light source estimation," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1697–1716, 2016.

[13] T. Laidlow, M. Bloesch, W. Li, and S. Leutenegger, "Dense rgb-d-inertial slam with map deformations," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 6741–6748.

[14] M. Kuse, S. P. Jaiswal, and S. Shen, "Deep-mapnets : A residual network for 3d environment representation," in *2017 IEEE International Conference on Image Processing (ICIP)*, Sept 2017, pp. 2652–2656.

[15] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, "Codeslam - learning a compact, optimisable representation for dense visual slam," *CoRR*, vol. abs/1804.00874, 2018.

[16] H. Luo, Y. Gao, Y. Wu, C. Liao, X. Yang, and K. Cheng, "Real-time dense monocular slam with online adapted depth prediction network," *IEEE Transactions on Multimedia*, pp. 1–1, 2018.

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

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

[19] A. Concha Belenguer and J. Civera Sancho, "Dpptam: Dense piecewise planar tracking and mapping from a monocular sequence," in *Proc. IEEE/RSJ Int. Conf. Intell. Rob. Syst.*, no. ART-2015-92153, 2015.

[20] A. Concha and J. Civera, "Using superpixels in monocular slam," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 365–372.

[21] A. Concha, G. Loianno, V. Kumar, and J. Civera, "Visual-inertial direct slam," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1331–1338.

[22] X. Zheng, Z. Moratto, M. Li, and A. I. Mourikis, "Photometric patch-based visual-inertial odometry," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 3264–3271.

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

[24] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1053–1072, 2017.

[25] S. Weiss, M. W. Achtelik, S. Lynen, M. C. Achtelik, L. Kneip, M. Chli, and R. Siegwart, "Monocular vision for long-term micro aerial vehicle state estimation: a compendium," *Journal of Field Robotics*, vol. 30, no. 5, pp. 803–831, 2013.