2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Robust Odometry Estimation for RGB-D Cameras

Christian Kerl, Jürgen Sturm, and Daniel Cremers

Abstract— **The goal of our work is to provide a fast and accurate method to estimate the camera motion from RGB-D images. Our approach registers two consecutive RGB-D frames directly upon each other by minimizing the photometric error. We estimate the camera motion using non-linear minimization in combination with a coarse-to-fine scheme. To allow for noise and outliers in the image data, we propose to use a robust error function that reduces the influence of large residuals. Furthermore, our formulation allows for the inclusion of a motion model which can be based on prior knowledge, temporal filtering, or additional sensors like an IMU. Our method is attractive for robots with limited computational resources as it runs in real-time on a single CPU core and has a small, constant memory footprint. In an extensive set of experiments carried out both on a benchmark dataset and synthetic data, we demonstrate that our approach is more accurate and robust than previous methods. We provide our software under an open source license.**

I. Introduction

Visual odometry is an important sensor modality for robot control and navigation in environments when no external reference system, e.g. GPS, is available [1]–[3]. Especially quadrocopters operating in cluttered indoor environments need pose updates at high rates for position control. At the same time they are only capable to carry sensors and processors with limited weight and power consumption. The lightweight commodity RGB-D cameras that became available in recent years are well suited for such application scenarios. For example, the Asus Xtion Pro Live sensor provides the 3D geometry and the visual appearance of the scene in VGA resolution at video frame rates. As the sensor weighs only 77 gram and consumes less than 2.5 Watt, it is well suited for the application on indoor flying robots [4], [5].

In our work, we are interested in methods to estimate the motion of an RGB-D camera and to use these estimates for local navigation and position control. Given this application scenario, the challenge is to compute motion updates at high frame rates with low latency, and to make them robust to outliers and as failure-safe as possible. However, only few approaches have been proposed so far that fully exploit both the intensity and the depth information provided by RGB-D sensors.

Odometry methods based on visual features such as SIFT or SURF are too slow for being used in low-latency applications [6], [7]. Patch-based approaches such as the KLT tracker or the PTAM tracker are fast [8], [9], but neglect major parts of the image and thus do not optimally exploit the available sensor data. Recently, promising approaches have been presented that compute the camera motion directly and densely from the RGB-D frame [10], [11]. The advantage of these methods is that they give precise motion estimates and are fast enough to run in real-time on a single CPU core.

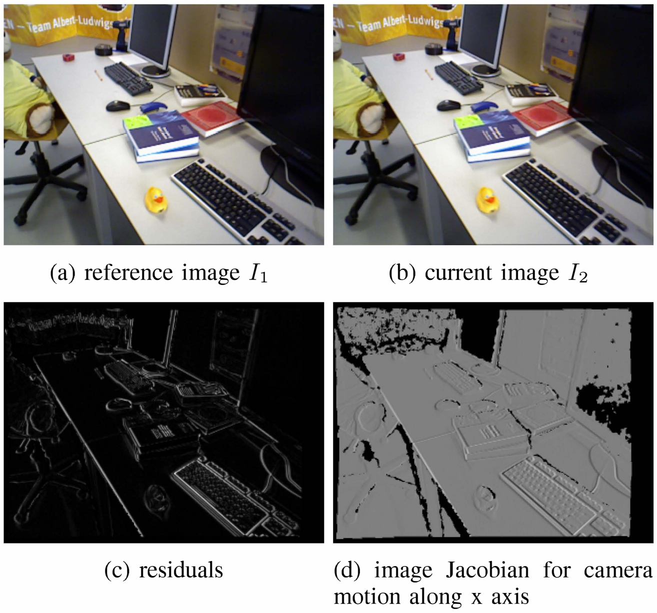

(a) reference image I_1
(b) current image I_2
(c) residuals
(d) image Jacobian for camera motion along x axis

Fig. 1: We compute the camera motion between two consecutive RGB-D frames (a+b) by minimizing the photo metric error (c) based on motion-induced brightness changes (d).

In this paper, we propose a robust, real-time odometry method based on dense RGB-D images. We compute the camera motion by aligning two consecutive RGB-D images as shown in Fig. 1 and minimizing the photometrical error between them. In contrast to sparse feature-based methods we use all color information of the two images and the depth information of the first image. Our method is an generalization and extension of our recent work [10]. The code and a video of our approach are available at:

vision.in.tum.de/data/software/dvo

The main contributions of this paper are:

- a probabilistic formulation for direct motion estimation based on RGB-D data,
- a robust sensor model derived from real world data,
- the integration of a temporal prior,
- an open-souce implementation that runs in real-time (30 Hz) on a single CPU core,
- the rigorous evaluation on benchmark data demonstrating the high accuracy and robustness of our approach.

All authors are with the Computer Vision Group, Department of Computer Science, Technical University of Munich {christian.kerl, juergen.sturm, daniel.cremers}@in.tum.de

978-1-4673-5643-5/13/$31.00 ©2013 IEEE

II. Related Work

Visual odometry methods typically track features in monocular or stereo images and estimate the camera motion between them [1], [12], [13]. Robustness is achieved by using RANSAC to ignore false or inconsistent feature matches. To increase the precision, features are tracked over multiple frames, which leads to the so-called simultaneous localization and mapping (SLAM) or structure-from-motion problem (SfM). Frequently, bundle adjustment is used to refine the pose estimates [14]. Various well-working systems based on this approach have been presented over the past years [4], [9], including our own recent works [3], [15]. Yet, all of these approaches ignore most of the image as features are only extracted sparsely, i.e., at a few (typically 50-500) interest points in the image. Through this pre-selection step, much valuable information is lost.

In contrast, dense methods aim at using the whole image. Köser et al. [16] showed that given a textured 3D model, the camera pose can be estimated efficiently by minimizing the photometric error between the observed image and a synthesized image. This approach was extended by Comport et al. [17] for use with a stereo camera. In principle, their approach can be seen as an extension of the 2D Lukas-Kanade tracker to three dimensions [8], [18]. Recently, both Steinbrücker et al. [10] and Tykkälä et al. [11] extended this approach to register RGB-D images obtained from a Microsoft Kinect sensor, and demonstrated that using this approach highly accurate visual odometry can be computed for static scenes.

Instead of aligning images one can also align 3D point clouds. Often, variations of the iterative closest points (ICP) algorithm [19], [20] are applied, which is however computationally costly as in each iteration the nearest neighbors between two point clouds have to be determined. [21] combine depth and color information from an RGB-D camera into an octree model, use feature descriptors to find correspondences between two octrees and run ICP to align them.

Recent work indicates that the accuracy of dense alignment algorithms can be increased by matching the current image against a scene model instead of the last image. The model is continuously updated with new images during camera tracking. Examples of those approaches are KinectFusion [22] and Dense Tracking and Mapping (DTAM) [23]. KinectFusion uses a variant of ICP for image to model alignment, where as DTAM uses a similar photometric error as [10], [11]. Both DTAM and KinectFusion achieve real-time performance but require state-of-the-art GPUs for the computations. To achieve robustness against outliers many approaches use binary thresholding [20], [23] to segment the image into inliers and outliers, or continuous methods from robust statistics [24], [25]. Robustified error functions are also common in other application areas like bundle adjustment [14].

Several visual odometry approaches use a non-uniform prior on the motion estimate to guide the optimization towards the true solution. These priors can be derived from assumptions about the motion, e.g. small motion or constant velocity [26]. Alternatively, measurements from other sensors (e.g., an IMU) or predictions from a filter (e.g., a Kalman filter) can be used.

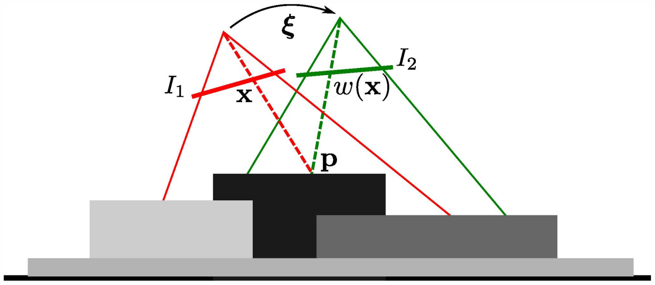


Fig. 2: Our approach is based on the photo-consistency assumption. The goal is to estimate the camera motion $\boldsymbol{\xi}$ such that the warped second image matches the first image.

In contrast to all previous work on dense direct motion estimation, we provide a probabilistic derivation of the model. This formulation allows us to choose a suitable sensor and motion model depending on the application. In particular, we propose to use a robust sensor model based on the t-distribution, and a motion prior based on a constant velocity model.

III. Direct Motion Estimation

In this section, we introduce our direct motion estimation approach from RGB-D data. The approach is a generalized version of recent work including our own [10], [11]. In contrast to these previous works, we provide here for the first time a probabilistic derivation and illustrate how priors on the motion and the sensor noise can easily be integrated using this formulation.

Our goal is to estimate the camera motion by aligning two consecutive intensity images I_1 and I_2 with corresponding depth maps Z_1 and Z_2 obtained from an RGB-D camera. Our approach is based on the photo-consistency assumption, as illustrated in Fig. 2: A world point $\mathbf{p}$ observed by two cameras is assumed to yield the same brightness in both images, i.e.,

$$I_1(\mathbf{x}) = I_2(\tau(\boldsymbol{\xi}, \mathbf{x})). \tag{1}$$

Here, $\tau(\boldsymbol{\xi}, \mathbf{x})$ is the *warping* function that maps a pixel coordinate $\mathbf{x} \in \mathbb{R}^2$ from the first image to a coordinate in the second image given the camera motion $\boldsymbol{\xi} \in \mathbb{R}^6$. Our goal is to find the camera motion $\boldsymbol{\xi}$ that best satisfies the photo-consistency constraint over all pixels. This approach is complementary to that of [27] for real-time dense geometry from a handheld camera. In both cases a photo-consistency error is minimized. While in [27] the geometry is estimated with known camera poses, here we estimate the camera pose for known geometry.

In the remainder of this section, we derive the warping function, specify the error function based on all pixels, and provide an efficient minimization strategy using a coarse-to-fine scheme. Subsequently, we extend this approach in Section IV by adding a weight to each pixel and by incorporating a motion prior.

A. Warping Function

We construct the warping function as follows: First, we reconstruct the 3D point $\mathbf{p}$ corresponding to the pixel $\mathbf{x} = (u, v)^\top$ using the inverse of the projection function π as

$$\begin{aligned} \mathbf{p} &= \pi^{-1}(\mathbf{x}, Z_1(\mathbf{x})) && (2) \\ &= Z_1(\mathbf{x}) \left(\frac{u + c_x}{f_x}, \frac{v + c_y}{f_y}, 1 \right)^\top && (3) \end{aligned}$$

where $Z_1(\mathbf{x})$ is the depth of the pixel, and f_x, f_y and c_x, c_y denote the focal length and optical center of the pinhole camera model, respectively. In the coordinate frame of the second camera, the point $\mathbf{p}$ is rotated and translated according to the rigid body motion $g \in \mathrm{SE}(3)$. A rigid body motion comprises a rotation represented as a 3×3 orthogonal matrix $R \in \mathrm{SO}(3)$ and a translation represented as a 3×1 vector $\mathbf{t} \in \mathbb{R}^3$. Correspondingly, the point $\mathbf{p}$ in the frame of the second camera is given as

$$T(g, \mathbf{p}) = R\mathbf{p} + \mathbf{t}. \qquad (4)$$

To have a minimal parametrization of g we use twist coordinates, i.e.,

$$\boldsymbol{\xi} = (\nu_1, \nu_2, \nu_3, \omega_1, \omega_2, \omega_3)^\top \in \mathbb{R}^6, \qquad (5)$$

where ν_1, ν_2, ν_3 are also called the linear velocity and $\omega_1, \omega_2, \omega_3$ the angular velocity of the motion. The rotation matrix and translation vector of g can be calculated from $\boldsymbol{\xi}$ with the exponential map relating Lie algebra $\mathfrak{se}(3)$ to Lie group $\mathrm{SE}(3)$:

$$g(\boldsymbol{\xi}) = \exp(\hat{\boldsymbol{\xi}}) \qquad (6)$$

A closed form solution to compute the matrix exponential $\exp(\hat{\boldsymbol{\xi}})$ exists. For more details on the Lie algebra and the exponential map, we refer the interested reader to the book of Ma et al. [28].

When the second camera observes the transformed point $T(g, \mathbf{p}) = (x, y, z)^\top$, we obtain the warped pixel coordinates

$$\pi(T(g, \mathbf{p})) = \left(\frac{f_x x}{z} - c_x, \frac{f_y y}{z} - c_y \right)^\top. \qquad (7)$$

To summarize, the full warping function is given by

$$\begin{aligned} \tau(\boldsymbol{\xi}, \mathbf{x}) &= \pi(T(g(\boldsymbol{\xi}), \mathbf{p})) && (8) \\ &= \pi(T(g(\boldsymbol{\xi}), \pi^{-1}(\mathbf{x}, Z_1(\mathbf{x})))). && (9) \end{aligned}$$

B. Likelihood function

For the moment, we assume that the photo-consistency assumption as stated in (1) holds equally for all n pixels $\mathbf{x}_i$ with $i = 1, \ldots, n$ in the image. We define the *residual* of the i-th pixel as the difference in brightness between the first and the warped second image, i.e.,

$$r_i(\boldsymbol{\xi}) := I_2(\tau(\boldsymbol{\xi}, \mathbf{x}_i)) - I_1(\mathbf{x}_i). \qquad (10)$$

In Fig. 1a and 1b two exemplary input images are shown. Their residual image is depicted in figure 1c where brighter pixels indicate larger errors. Ideally, the residuals would be zero, however, due to sensor noise, the residuals will be distributed according to the probabilistic *sensor model* $p(r_i \mid \boldsymbol{\xi})$. By assuming that the noise of all pixels is independent and identically distributed, the likelihood of observing the whole residual image $\mathbf{r} = (r_1, \ldots, r_n)^\top$ becomes

$$p(\mathbf{r} \mid \boldsymbol{\xi}) = \prod_i p(r_i \mid \boldsymbol{\xi}). \qquad (11)$$

Using Bayes' rule, we obtain the a posteriori likelihood of a camera motion $\boldsymbol{\xi}$ given a residual image $\mathbf{r}$, i.e.,

$$p(\boldsymbol{\xi} \mid \mathbf{r}) = \frac{p(\mathbf{r} \mid \boldsymbol{\xi}) p(\boldsymbol{\xi})}{p(\mathbf{r})}. \qquad (12)$$

Note that $p(\boldsymbol{\xi})$ denotes the prior distribution over camera motions. Possible choices include a uniform prior (all camera motions are equally likely), a motion prior from an additional sensor like an IMU, or the prediction of a Kalman filter.

C. Maximum A Posteriori (MAP) estimation

We now seek for the camera motion $\boldsymbol{\xi}$ that maximizes the posterior probability, i.e.,

$$\boldsymbol{\xi}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{\xi}} p(\boldsymbol{\xi} \mid \mathbf{r}). \qquad (13)$$

By plugging in (12) and (11), and dropping the term $p(\mathbf{r})$ as it does not depend on $\boldsymbol{\xi}$, we obtain

$$\boldsymbol{\xi}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{\xi}} \prod_i p(r_i \mid \boldsymbol{\xi}) p(\boldsymbol{\xi}). \qquad (14)$$

By minimizing instead the negative log-likelihood, we can equivalently write

$$\boldsymbol{\xi}_{\mathrm{MAP}} = \arg\min_{\boldsymbol{\xi}} - \sum_i \log p(r_i \mid \boldsymbol{\xi}) - \log p(\boldsymbol{\xi}) \qquad (15)$$

To avoid clutter in the notation, we drop the motion prior $\log p(\boldsymbol{\xi})$ in the remainder of this section. We discuss it in more detail in the next section.

The minimum is found by taking the derivative of the log likelihood and setting it to zero, i.e.,

$$\sum_i \frac{\partial \log p(r_i \mid \boldsymbol{\xi})}{\partial \boldsymbol{\xi}} = \sum_i \frac{\partial \log p(r_i)}{\partial r_i} \frac{\partial r_i}{\partial \boldsymbol{\xi}} = 0. \qquad (16)$$

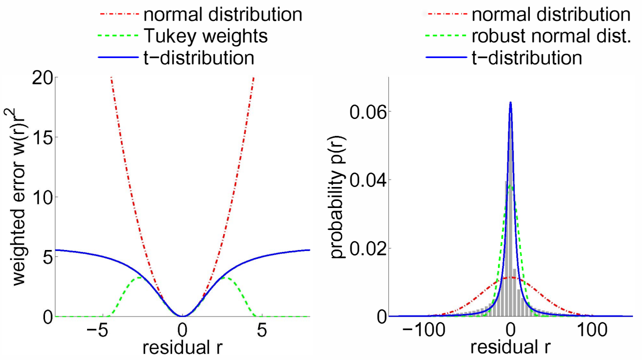
By defining $w(r_i) = \partial \log p(r_i) / \partial r_i \cdot 1/r_i$, we obtain

$$\frac{\partial r_i}{\partial \boldsymbol{\xi}} w(r_i) r_i = 0 \qquad (17)$$

which minimizes the weighted least squares problem:

$$\boldsymbol{\xi}_{\mathrm{MAP}} = \arg\min_{\boldsymbol{\xi}} \sum_i w(r_i)(r_i(\boldsymbol{\xi}))^2. \qquad (18)$$

The function $w(r_i)$ is often called the *weighting function*, as it describes how strongly a particular residual is considered during minimization. Note that if $p(r_i) \propto \exp(r_i^2/\sigma^2)$ is normally distributed, then $w(r_i)$ is constant, leading to normal least squares minimization. For non-Gaussian error models, that we will consider in the next section, the weighting function will be non-constant. Fig. 3a shows the weighted



(a) weighted error (b) residual histogram

Fig. 3: (a) Depending on the weight function residuals have different influence on the optimization. (b) We found that the distribution over residuals (gray) is not well approximated by a Gaussian distribution (red, green). In contrast, a t-distribution matches the observed residuals nicely (blue).

quadratic error for different weight functions. The advantage of this formulation is that also other sensor models can be applied to allow for non-Gaussian noise (e.g., outliers). To solve this minimization problem, we apply the iteratively re-weighted least squares (IRLS) algorithm, where the computation of weights and the estimates of $\boldsymbol{\xi}$ is alternated until convergence.

D. Linearization

To find the best camera motion, we have to solve (17). As the residuals $r_i(\boldsymbol{\xi})$ are non-linear in $\boldsymbol{\xi}$, we use the Gauss-Newton method to iteratively build and solve a system of linear equations. For this, we need to compute the first order Taylor approximation of $r_i(\boldsymbol{\xi})$.

$$r_{\text{lin}}(\boldsymbol{\xi}, \mathbf{x}_i) = r(\mathbf{0}, \mathbf{x}_i) + \left.\frac{\partial r(\tau(\boldsymbol{\xi}, \mathbf{x}_i))}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\mathbf{0}} \Delta\boldsymbol{\xi} \tag{19}$$

$$= r(\mathbf{0}, \mathbf{x}_i) + J_i \Delta\boldsymbol{\xi}, \tag{20}$$

where $J_i \in \mathbb{R}^{1\times 6}$ is the Jacobian of the i-th pixel with respect to the 6-DOF camera motion. By plugging this into (17) and writing all constraints in matrix notation, we obtain the normal equations

$$J^T W J \Delta\boldsymbol{\xi} = -J^T W \mathbf{r}(\mathbf{0}), \tag{21}$$

where $J \in \mathbb{R}^{n\times 6}$ is the stacked matrix of all J_i pixel-wise Jacobians and W is the diagonal matrix of weights with $W_{ii} = w(r_i)$. In Fig. 1d the first column of the Jacobian matrix for the two example images is given, corresponding to the derivative along the camera x axis. The linear system of equations in (21) can be efficiently solved, for example using Cholesky decomposition. At each iteration k of the Gauss-Newton algorithm, we compute an increment $\Delta\boldsymbol{\xi}^{(k)}$ using (21) with which we update our motion estimate using $\boldsymbol{\xi}^{(k+1)} = \log(\exp(\boldsymbol{\xi}^{(k)})\exp(\Delta\boldsymbol{\xi}))$. Note that we do not need to re-linearize (17) at every iteration, because we warp the image I_2 with our current estimate $\boldsymbol{\xi}$ towards I_1. Therefore, we only need the Jacobian matrix at the identity.

As the linearization is only valid for small $\boldsymbol{\xi}$, we apply a coarse-to-fine scheme: First we build an image pyramid where we half the image resolution at each level. Then we estimate the camera motion at each pyramid level as described above and use it as an initialization for the next level. In this way, even large translational and rotational motions can be handled.

IV. Robust Motion Estimation

The approach described in Section III is very flexible, as it allows us to choose a suitable sensor model $p(\mathbf{r} \mid \boldsymbol{\xi})$ and motion prior $p(\boldsymbol{\xi})$. For example, the robustness of motion estimation can be increased by using a sensor model that allows for outliers. Furthermore, if additional information on the camera motion is available, it can be plugged into the motion model.

A. Sensor Model

In [10], no weighting is used. This is equivalent to assuming normal distributed errors $\mathbf{r}$. In our recent studies, however, we found that this assumption is often violated: This is exemplified in Fig. 3b where a typical residual histogram from the "fr1/desk" sequence is depicted (gray bars). As can be seen from this plot, the normal distribution (depicted in red) fits the data poorly. [11] recently proposed to use a M-estimator to fit the Gaussian distribution more robustly (depicted in green). In our analysis, however, we found that the normal distribution independent of the choice of σ does not fit the residual distribution well. The problem is that the normal distribution assigns too low probabilities to large and very small residuals, but too high probabilities in between. Therefore, we follow [29], [30] and assume t-distributed errors where in addition to mean μ and variance σ^2 also the so-called *degrees of freedom* ν of the distribution can be specified. The t-distribution is suited to model data distributions with outliers, because of its heavy tails covering the outliers with low probability. As can be seen from the figure, the fitted t-distribution (depicted in blue) matches nicely the residual distribution.

The weight function $w(r)$ derived from the t-distribution is

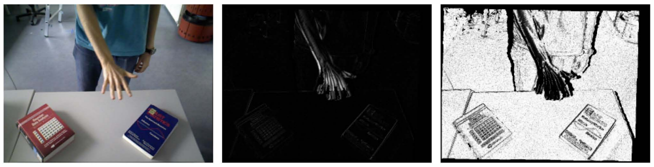
$$w(r_i) = \frac{\log p(r_i)}{\partial r_i}\frac{1}{r_i} = \frac{\nu + 1}{\nu + (\frac{r_i}{\sigma})^2} \tag{22}$$

Based on our experiments, we determined degrees of freedom to $\nu = 5$. In each iteration of IRLS, we compute the variance σ^2 using

$$\sigma^2 = \frac{1}{n}\sum_i r_i^2 \frac{\nu + 1}{\nu + (\frac{r_i}{\sigma})^2}. \tag{23}$$

This equation has to be solved iteratively, because it is recursive, but it converges in few iterations.

The effect of weighting is illustrated in Fig. 4 where a hand moves in a different direction than the camera causing outliers in the residuals (cf. 4b). The outliers get down-weighted as can be seen in Fig. 4c where darker pixels indicate lower weights.



(a) scene (b) residuals (c) weights

Fig. 4: In this experiment, a hand moves through the scene (a) which causes large residuals (b). By using a robust weighting function, the outlier pixels are ignored (dark) for motion estimation (c).

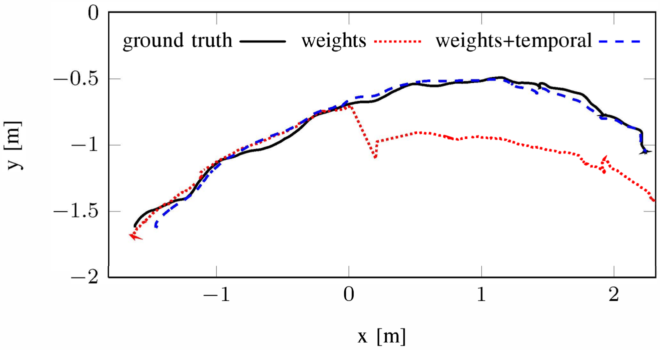


Fig. 5: A temporal prior stabilizes motion estimation significantly.

B. Motion Prior

With the approach described so far, the camera motion can be estimated accurately when the RGB-D frames contain sufficient texture and structure. However, feature-poor input images, motion blur or dynamic objects may lead to increased noise and even divergence of the motion estimate. In particular, in our previous implementation [10], we treated all motions equally likely. As a result, we occasionally observed jumps in the estimated trajectories as illustrated in Fig. 5.

We assume a constant velocity model with a normal distribution, i.e., $p(\boldsymbol{\xi}_t) = \mathcal{N}(\boldsymbol{\xi}_{t-1}, \Sigma)$, where $\boldsymbol{\xi}_{t-1}$ is the camera speed from the previous time step and $\Sigma \in \mathbb{R}^{6\times 6}$ is a diagonal covariance matrix that defines how quickly it may change.

When we derive the normal equations similar to (21) from (15) including the motion prior, we obtain

$$(J^T W J + \Sigma^{-1})\Delta\boldsymbol{\xi} = -J^T W \mathbf{r}(0) + \Sigma^{-1}(\boldsymbol{\xi}_{t-1} - \boldsymbol{\xi}_t^{(k)}), \tag{24}$$

where $\boldsymbol{\xi}_t^{(k)}$ is the motion estimate after the k-th iteration at time step t. As can be seen from this equation, a large covariance matrix Σ will decrease the influence of the motion prior with respect to the image-based residuals, and vice versa.

V. Evaluation

In this section, we demonstrate in a series of experiments that the robust sensor model and a temporal motion model strongly improve the accuracy of the estimated odometry. We evaluated our approach using the TUM RGB-D benchmark [31] and on self-generated synthetic datasets with perfect ground truth. For each experiment we calculated the root mean square error (RMSE) of the drift in meters per second. Furthermore, the average runtime for matching a pair of images was measured. All timing results were obtained on a PC with Intel i5 670 CPU (3.46 GHz) and 4 GB RAM. It should be noted that for the computations only one CPU core was utilized, and the memory footprint is less than four times the size of the input images (for storing the image pyramid and the image gradient at each level).

For comparison, we computed the camera trajectories also with our previous version [10] and a re-implementation of [11] using the Tukey weight function. We ran our previous implementation with parameters optimized for speed, denoted as “reference” in the tables.

A. Synthetic Sequences

We generated two synthetic sequences with perfect ground truth, one on a static scene (“static”) and one with a small, moving object in it (“moving”). To generate the images, we simulated a moving camera along a sampled trajectory on a real RGB-D frame from the “fr1/desk” sequence. The motion in the camera plane was sampled from a uniform distribution between ±0.01 m to emulate a camera moving at speed similar to ones in [31], i.e., 0.3 m/s. Additionally, a rotation around the camera axis was sampled from a uniform distribution between ±5°. For the “moving” sequence, we moved a patch of the original image along a trajectory independent of the simulated camera motion.

Further, we used our implementation with two parameter sets, one suited for realtime performance and one for maximum precision. The most notable difference between both parameter sets is, that the one optimized for speed only uses images up to a resolution of 320 × 240 pixels, where the other uses full resolution of 640 × 480. Other parameters control the number of iterations performed by the algorithm.

Tab. I and Tab. II show the results. For the “static” sequence, all methods perform almost equally well, in particular when we allow for a large number of iterations and operate at the full image resolution (precision parameter set). For the “moving” sequence, this changes dramatically: even with as many iterations as possible, least squares yields a drift of 5.0 cm/s. The Tukey variant yields a drift of 2.7 cm/s while the t-distribution has only 1.3 cm/s. In all cases, the t-distribution always outperforms the unweighted variant.

It should be noted that the weighted variants roughly compute twice as long (56 ms and 47 ms) as the unweighted version (30 ms) with the realtime settings. When more iterations are allowed, this computation time even increases to 130 ms for the unweighted variant up to 508 ms for the t-distribution.

From this, we conclude that (1) a robust weighting function is particularly useful in the presence of noise and (potentially consistent) outliers and (2) that the t-distribution is better suited than a Gaussian distribution.

B. TUM RGB-D Benchmark Sequences

Additionaly, we evaluated our robust visual odometry method on several sequences of the TUM RGB-D bench-

TABLE I: On a simulated, static scene the improvement provided by robust weighting is negligible. However, we found that the Tukey weight function generally degrades the performance, because it overestimates the number of outlier pixels and thus neglects valuable information.

Method	Parameter set	RMSE [m/s]	Improvement	⌀Runtime [s]
reference		0.0751	0.0 %	0.038
no weights	realtime	0.0223	70.0 %	0.032
Tukey weights	realtime	0.0497	33.8 %	0.050
t-dist. weights	realtime	**0.0142**	81.1 %	0.043
no weights	precision	0.0145	80.7 %	0.115
Tukey weights	precision	0.0279	62.8 %	0.230
t-dist. weights	precision	**0.0124**	83.5 %	0.405

TABLE II: When an object moves through a simulated scene, weighting clearly increases the robustness. The t-distribution leads to the lowest error. In direct comparison to the static scene in Tab. I, the moving object does not significantly degrade the performance.

Method	Parameter set	RMSE [m/s]	Improvement	⌀Runtime [s]
reference		0.1019	0.0 %	0.035
no weights	realtime	0.0650	36.2 %	0.030
Tukey weights	realtime	0.0382	62.5 %	0.056
t-dist. weights	realtime	**0.0296**	71.0 %	0.047
no weights	precision	0.0502	50.7 %	0.130
Tukey weights	precision	0.0270	73.5 %	0.279
t-dist. weights	precision	**0.0133**	87.0 %	0.508

TABLE III: Results for the four "desk" sequences. The t-distribution with the temporal prior performs on average best.

Method	fr1/desk2 [m/s]	fr1/desk [m/s]	fr2/desk [m/s]	fr2/person [m/s]
reference	0.3416	0.5370	0.0205	0.0708
no weights	0.1003	0.0551	0.0231	0.0567
Tukey weights	0.2072	0.1740	0.1080	0.1073
t-dist. weights	0.0708	**0.0458**	0.0203	0.0360
t-dist. w.+temporal	**0.0687**	0.0491	**0.0188**	**0.0345**
avg. camera velocity	0.413	0.426	0.193	0.121

TABLE IV: On the four "desk" sequences of TUM RGB-D benchmark, the t-distribution improves the accuracy significantly.

Method	⌀Drift [m/s]	⌀Improvement
reference	0.2425	0.00 %
no weights	0.0588	75.74 %
Tukey weights	0.1491	38.50 %
t-dist. weights	0.0432	82.18 %
t-dist. weights+temporal	**0.0428**	**82.35** %

TABLE V: We evaluated the impact of dynamic objects in the scene on the six "fr3/sitting" sequences, where two persons are sitting at a table and chatting. Robust estimation in combination with the temporal prior has the lowest error.

Method	⌀Drift [m/s]	⌀Improvement
reference	0.0800	0.00 %
no weights	0.0350	56.23 %
Tukey weights	0.1038	-29.66 %
t-dist. weights	0.0470	41.33 %
t-dist. weights+temporal	**0.0316**	**60.55** %

mark. In these experiments we focused on the realtime parameter set, because it is more relevant for our application on a quadrocopter.

Note that we measure the drift in m/s in contrast to earlier publications [10], [21] where the drift was measured per frame. We found this measure to be inaccurate given that the noise of the motion capture system has the same level of magnitude [31]. Moreover, instead of the median drift we measure the RMSE drift, which is much more influenced by large, occasional errors in the estimate. Therefore, low RMSE drift values indicate a continously high tracking quality.

For evaluation, we chose the four "desk" datasets, because they contain real world scenery with structure and texture, different velocity profiles, and have recently been used by several other authors for evaluation [7], [10], [21], [32], [33].

The results for the individual sequences are given in Tab. III and a summary is given in Tab. IV. The variant with t-distribution based weights performs best similarly as on the synthetic datasets. In comparison to the synthetic datasets, the weighted versions perform better than the unweighted one even on these datasets with static scenery. Our conclusion is that real data even from static real-world scenes contains significant amounts of noise and outliers that need to be dealt with. From inspection, we found that these outliers partially stem from occlusions and reflections in the scene that violate the photo-consistency assumption. On the "desk" sequences, we found that the temporal prior does not further improve the accuracy of our method. We think that the reason for this is that the "desk" sequences are sufficiently feature rich, so that the minimization of the residuals is the driving factor.

We also studied the behavior of all variants on the "sitting" and "walking" sequences, where two persons are in the field of view in front of a desk while the handheld camera is being moved along different trajectories. On the "sitting" sequences, we found that the t-distribution in combination with the temporal motion prior performs best, see Tab. V. This is an expected result, as the prior provides a valuable initialization for the minimization and guides it in the right direction. However, overshooting at turns in the trajectory diminishes the overall improvement. Yet, the prior is in particular useful when many outliers (due to dynamic objects) are present in the input data such as in these sequences.

In the "walking" sequences, outlier pixels dominate the images, as substantial parts of most images consist of moving persons. On these datasets, we found that all discussed methods fail. In particular, even the best performing method (Tukey weights) still yields a drift of 0.25 m/s, which is larger than the average camera speed. Therefore, it remains an open challenge to compute accurate motion estimates in the presence of large dynamic objects. Note that for this work, we explicitly focused on state-less visual odometry, but in such cases it might be necessary to estimate a (local) map of the static background similar to [22].

In sum, we showed in our experiments that the t-distribution improves the accuracy significantly over a broad range of real and synthetic sequences. Furthermore, we demonstrated that the temporal prior has a positive effect in scenes with a large number of outliers, few photometric information or high velocity of camera.

VI. Conclusion

In this paper, we derived a probabilistic formulation to directly estimate the camera motion from RGB-D images. Our formulation allows the use of custom probability distributions for the sensor and the motion models. For our application, we demonstrated that the t-distribution better matches the observed residuals and leads to higher accuracies. Furthermore, a motion prior is incorporated in the minimization problem to guide and stabilize motion estimation in the presence of dynamic objects. In exhaustive experiments on real and simulated data, we demonstrated that our approach is highly accurate, robust, and outperforms previous methods. In the near future, our goal is to apply our algorithm on a flying quadrocopter to compensate for short-term motion drift. Our software is available under an open source license.

References

[1] K. Konolige, M. Agrawal, R. Bolles, C. Cowan, M. Fischler, and B. Gerkey, "Outdoor mapping and navigation using stereo vision," in *Intl. Symp. on Experimental Robotics (ISER)*, 2007.

[2] S. Weiss, M. Achtelik, M. Chli, and R. Siegwart, "Versatile distributed pose estimation and sensor self-calibration for an autonomous MAV," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2012.

[3] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," in *Intl. Conf. on Intelligent Robot Systems (IROS)*, 2012.

[4] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Intl. Symposium of Robotics Research (ISRR)*, 2011.

[5] S. Shen, N. Michael, and V. Kumar, "Autonomous indoor 3d exploration with a micro-aerial vehicle," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2012.

[6] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments," in *Intl. Symp. on Experimental Robotics (ISER)*, 2010.

[7] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2012.

[8] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, 1981.

[9] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *IEEE and ACM Intl. Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.

[10] F. Steinbrücker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense RGB-D images," in *Workshop on Live Dense Reconstruction with Moving Cameras at the Intl. Conf. on Computer Vision (ICCV)*, 2011.

[11] T. Tykkälä, C. Audras, and A. Comport, "Direct iterative closest point for real-time visual odometry," in *Workshop on Computer Vision in Vehicle Technology: From Earth to Mars at the Intl. Conf. on Computer Vision (ICCV)*, 2011.

[12] A. Chiuso, P. Favaro, H. Jin, and S. Soatto, "3-D motion and structure from 2-D motion causally integrated over time: Implementation," in *European conf. on Computer Vision (ECCV)*, 2000.

[13] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[14] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[15] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard, "Real-time 3D visual SLAM with a hand-held camera," in *RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum (ERF)*, 2011.

[16] K. Koeser, B. Bartczak, and R. Koch, "An analysis-by-synthesis camera tracking approach based on free-form surfaces," in *DAGM conf. on pattern recognition*, 2007.

[17] A. Comport, E. Malis, and P. Rives, "Real-time Quadrifocal Visual Odometry," *International Journal of Robotics Research (IJRR)*, vol. 29, no. 2-3, 2010.

[18] S. Baker and I. Matthews, "Lucas-Kanade 20 years on: A unifying framework," *Intl. Journal of Computer Vision (IJCV)*, vol. 56, no. 3, 2004.

[19] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, vol. 14, no. 2, 1992.

[20] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Intl. Conf. on 3-D Digital Imaging and Modeling (3DIM)*, 2001.

[21] J. Stückler and S. Behnke, "Model learning and real-time tracking using multi-resolution surfel maps," in *AAAI Conf. on Artificial Intelligence (AAAI)*, 2012.

[22] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *IEEE Intl. Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[23] R. A. Newcombe, S. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Intl. Conf. on Computer Vision (ICCV)*, 2011.

[24] P. Huber, *Robust Statistics*, ser. Wiley Series in Probability and Statistics. John Wiley & Sons, 2003.

[25] A. Comport, E. Marchand, and F. Chaumette, "Statistically robust 2D visual servoing," *IEEE Trans. on Robotics (T-RO)*, vol. 22, no. 2, 2006.

[26] S. Lovegrove, A. J. Davison, and J. I. Guzman, "Accurate visual odometry from a rear parking camera," in *Intelligent Vehicles Symposium*, 2011.

[27] J. Stühmer, S. Gumhold, and D. Cremers, "Real-time dense geometry from a handheld camera," in *DAGM conf. on pattern recognition*, 2010.

[28] Y. Ma, S. Soatto, J. Kosecká, and S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*, ser. Interdisciplinary applied mathematics: Imaging, vision, and graphics. Springer, 2003.

[29] K. L. Lange, R. J. A. Little, and J. M. G. Taylor, "Robust statistical modeling using the t distribution," *Journal of the American Statistical Association (JASA)*, vol. 84, no. 408, 1989.

[30] A. Gelman, J. Carlin, H. Stern, and D. Rubin, *Bayesian Data Analysis*. Chapman & Hall/CRC, 2003.

[31] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Intl. Conf. on Intelligent Robot Systems (IROS)*, 2012.

[32] T. Stoyanov, M. Magnusson, and A. Lilienthal, "Point set registration through minimization of the L2 distance between 3D-NDT models," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2012.

[33] P. Osteen, J. Owens, and C. Kessens, "Online egomotion estimation of RGB-D sensors using spherical harmonics," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2012.