DNA-SLAM: Dense Noise Aware SLAM for ToF RGB-D Cameras

Oliver Wasenmüller[1(✉)], Mohammad Dawud Ansari[2], and Didier Stricker[1,2]

[1] German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany
{oliver.wasenmueller,didier.stricker}@dfki.de
[2] University of Kaiserslautern, Kaiserslautern, Germany
m_ansari15@cs.uni-kl.de

Abstract. SLAM with RGB-D cameras is a very active field in Computer Vision as well as Robotics. Dense methods using all depth and intensity information showed best results in the past. However, usually they were developed and evaluated with RGB-D cameras using *Pattern Projection* like the Kinect v1 or Xtion Pro. Recently, *Time-of-Flight* (ToF) cameras like the Kinect v2 or Google Tango were released promising higher quality. While the overall accuracy increases for these ToF cameras, noisy pixels are introduced close to discontinuities, in the image corners and on dark/glossy surfaces. These inaccuracies need to be specially addressed for dense SLAM. Thus, we present a new Dense Noise Aware SLAM (DNA-SLAM), which considers explicitly the noise characteristics of ToF RGB-D cameras with a sophisticated weighting scheme. In a rigorous evaluation on public benchmarks we show the superior accuracy of our algorithm compared to the state-of-the-art.

1 Introduction

Simultaneous Localization And Mapping (SLAM) is the process of determining the pose (position and orientation) of a camera and creating a map of the environment by analyzing the associated camera images. This is a very active research field in both *Computer Vision* and *Robotics*. When using monocular cameras, sparse features [1,2] or dense pixels [3,4] are tracked in the images to estimate the camera pose. Since the depth of all pixels is unknown, this is a challenging task, which comprehends several inaccuracies or requires extensive computations. With the release of low-cost RGB-D sensors (e.g. Microsoft Kinect) the possibility for dense depth measurements of the environment was given. There are two common approaches of depth measurement: *Pattern Projection* and *Time-of-Flight* (ToF). Cameras with *Pattern Projection*, such as Microsoft Kinect v1, Asus Xtion Pro or Occipital Structure, project

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-54407-6_42) contains supplementary material, which is available to authorized users.

C.-S. Chen et al. (Eds.): ACCV 2016 Workshops, Part I, LNCS 10116, pp. 613–629, 2017.
DOI: 10.1007/978-3-319-54407-6_42

a known pattern into the scene and estimate the depth from its distortion. Recently, ToF cameras, such as Microsoft Kinect v2 [5] or Google Tango [6], became very popular, since they claim a higher accuracy in general. ToF cameras measure depth by estimating the time emitted light takes from the camera to the object and back. RGB-D cameras spurred a bulk of research on SLAM. While first approaches extended only sparse monocular algorithms [7,8], best performing algorithms in the literature use dense depth and intensity measurements [9,10]. These dense algorithms try to estimate the rigid camera motion between two images by minimizing the photometric and geometric error. This holds under the assumption of a photometric and geometric consistency. Previous methods were mainly developed and evaluated with RGB-D cameras using *Pattern Projection*, where these assumptions hold although the overall accuracy is lower. However, in some experiments with ToF cameras we detected that the geometric consistency is often violated due to the sensor noise, leading to inaccurate trajectories. Thus, we propose a *Dense Noise Aware Simultaneous Localization And Mapping* (DNA-SLAM), which considers explicitly the noise characteristics of ToF RGB-D cameras with a sophisticated weighting scheme. More precisely, our main contributions in this paper are:

- A robust reliability estimation for each single pixel based on the noise characteristics of ToF cameras.
- The integration of the individual reliabilities into a sophisticated weighting scheme for dense motion estimation.
- A systematic and rigorous evaluation of our algorithm with well-known publicly available benchmarks with real image data.

2 Related Work

Many state-of-the-art methods establish correspondences between sparse features to estimate the cameras motion [8,11,12]. Recently, several dense algorithms [9,10,13,14] emerged for SLAM systems showing better accuracy. Steinbrücker et al. [15] and Audras et al. [16] proposed to minimize the photometric error between consecutive RGB-D frames. This concept was extended by Kerl et al. [9] for their DVO algorithm by weighting photometric errors according to the t-distribution. Klose et al. [17] presented a efficient second order minimization scheme for motion estimation. DVO was extended by a minimization of the geometric error by Kerl et al. [18], since only intensity was used so far. Mailland et al. [19] used for their motion estimation super-resolved keyframes, and Gutierrez et al. [10] parametrized the geometric error by the inverse depth in addition. Several extensions and optimizations of these algorithms exist in the literature, for example for rolling shutter cameras [20], planar scenes [13] and many more. In contrast to all related work, we explicitly consider the noise characteristics of *Time-of-Flight* cameras in a sophisticated weighting. This leads to a superior accuracy as we show in our evaluation in Sect. 5.

3 Basic Idea

The basic idea of DNA-SLAM is to estimate the reliability of each single pixel in a ToF RGB-D image in order to weight its influence on the dense camera motion estimation. In state-of-the-art papers [10,14,18], the camera motion is estimated based on the assumptions of photometric and geometric consistency. This means that neither the scene color nor the scene geometry changes over time in captured images. To verify these assumptions, we analyze in Fig. 1 RGB-D images of a given scene with two different types of cameras, namely a *Time-of-Flight* camera (Microsoft Kinect v2) and a *Pattern Projection* camera (Microsoft Kinect v1). We place both cameras on a stable tripod with very similar viewpoints and capture two consecutive RGB-D images (I_1, D_1) and (I_2, D_2). Ideally, the two consecutive images should not contain any differences in intensity and depth. Thus, the differences between the intensity images and the depth images respectively should be zero. As visible in Fig. 1 the differences in intensity are very small and consequently negligible for both cameras types. The depth images of

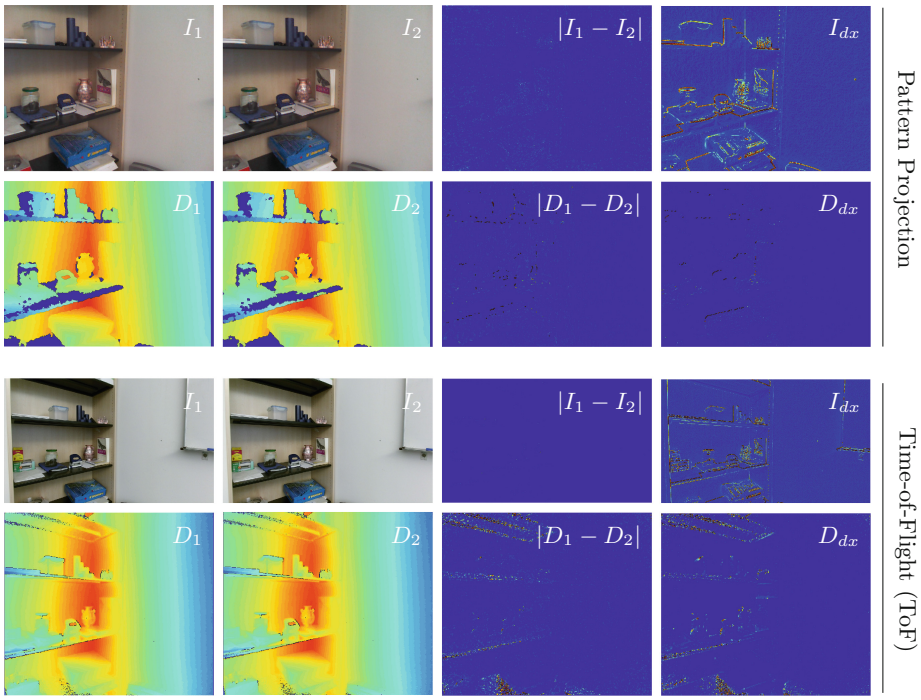


Fig. 1. The basic assumption of dense motion estimation is the photometric and geometric consistency. To verify this assumption we place a *Pattern Projection* camera (top) and a *Time-of-Flight* (ToF) camera (bottom) on a stable tripod and capture two consecutive RGB-D frames (I_1, D_1) and (I_2, D_2). We discover that ToF cameras violate the geometric consistency assumption $|D_1 - D_2| = 0$. However, the depth derivative D_{dx} coincides with these violations. See the supplementaries for large-scale images.

the *Pattern Projection* camera disclose only for some rare pixels bigger differences in depth. However, the ToF camera discloses violations of the geometric consistency assumption for numerous pixels as shown in the lowest row of Fig. 1. These violations for ToF cameras will lead to inaccuracies in the dense motion estimation, unless they are not especially treated. Thus, we propose a new sophisticated weighting scheme in this paper that incorporates the reliability of each single ToF pixel.

In the above mentioned experiment we can detect the violations quite easily by looking at the differences in depth, since the camera motion is zero. However, for unknown camera motions this is not possible. Therefore, we must define a measurement for the reliability of a pixel, which can be estimated on a single image without knowing the camera motion. In our experiments in Fig. 1 we discovered that for ToF cameras the derivatives of the depth image and the violations of the geometric consistency assumption coincide for most pixels. This means, ToF pixels with a high depth derivative are likely to violate the consistency assumption, whereas pixels with a low depth derivative can be considered as reliable. We repeated this experiment for numerous different scenes and all show similar results. We selected the example of Fig. 1 so that it is representative for other scenes.

ToF cameras have several noise sources [21]: dark and glossy scene colors, large scene distances, pixels close to the image boundaries, *flying pixels* close to depth discontinuities, etc. All these noises lead to violations of the geometric consistency assumption. Thus, in the ToF images of Fig. 1 the black shelfs and the four image corners violate the assumption for example. They all are precisely detected by the local derivatives. This is explicable for ToF RGB-D images, because of their noise characteristic [21]. ToF cameras exhibit a high frequent per-pixel noise. This means, in case of imprecise measurements the depth values of neighboring pixels strongly differ. In contrast, *Pattern Projection* cameras exhibit their noise as a per-patch noise or distortions over the surface. This means, neighboring pixels have similar values and errors. The per-pixel noise of ToF cameras can be easily detected by local derivatives, whereas per-patch noise can not. The particular thing is that the local derivatives are able to detect the noise positions independent of its source. This qualifies them to serve as an easy-to-compute measurement for the ToF depth reliability. Depth cameras with *Pattern Projection* enclose much less high-frequent noise, but lack in the overall accuracy [21]. In the supplementary video we show a live depth stream that illustrates the difference between per-pixel and per-patch noise. One might also claim that high derivatives are usual on depth discontinuities, but this is not possible due to the sensor technology. On depth discontinuities the sensor delivers either *flying pixels* (which are detected by the derivatives) or invalid values. In the following section we present a way to transform depth derivatives into a weighting scheme and to integrate them into the camera motion estimation.

4 DNA-SLAM

In this section, we present our DNA-SLAM algorithm. The goal is to estimate the camera motion ξ between consecutive RGB-D images composed of intensity images I_1 and I_2 and depth images D_1 and D_2. In DNA-SLAM we estimate the camera motion ξ according to the basic concept of state-of-the-art papers [10,15,16,18] by minimizing residuals r_i for all n valid pixels by

$$\xi = \arg\min_{\xi} \sum_{i}^{n} w(r_i)(r_i(\xi))^2, \tag{1}$$

where w_i are individual weights for each pixel. In the following sections we describe step-by-step how Eq. 1 is built and solved, while explicitly considering the noise characteristics of ToF cameras. Figure 2 gives an overview of the whole procedure. After describing some preliminaries in Sect. 4.1, we define in Sect. 4.2 the residuals r_i, which contain the photometric and geometric error of a given camera motion ξ. In Sect. 4.3, the noise aware weighting is described, which weights each pixel individually according to its reliability. As motivated in Sect. 3 we use (amongst others) the depth derivatives to compute robust weights. In Sect. 4.4 we detail how the camera motion ξ is computed by solving Eq. 1 with an iterative re-weighted least square algorithm. To enhance the accuracy of the motion estimation, we use a coarse-to-fine strategy.

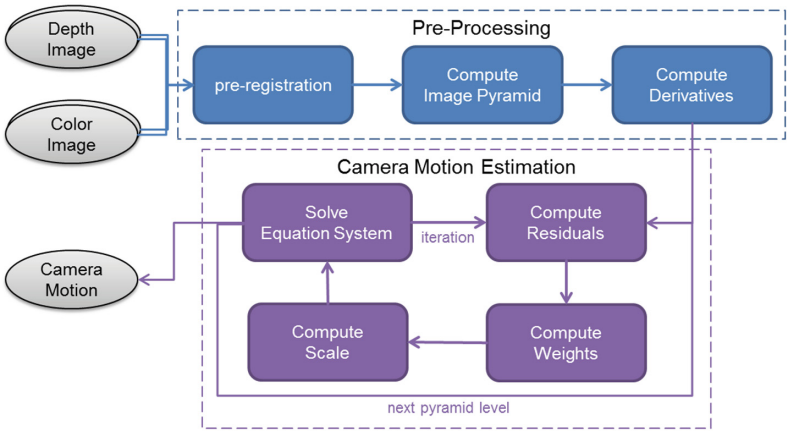


Fig. 2. Overview of DNA-SLAM. As an input the algorithm requires two image tuples (color and depth). After a pre-processing, the iterative motion estimation is performed as described in Sect. 4. For a short summary see also the supplementary video.

4.1 Preliminaries

In this section, we define the basics of the RGB-D camera images I and D, its derivatives $I_{dx,dy}$ and $D_{dx,dy}$ as well as the camera motion ξ. A point p in the 3D

space is defined by homogeneous coordinates as $p = (X, Y, Z, 1)^\top$. The 2D pixel coordinates $\mathrm{x} = (x, y)^\top$ of the 3D point p are defined by the standard pinhole camera model and the projection function π as

$$\mathrm{x} = \pi(p) = \left(\frac{X f_x}{Z} + c_x, \frac{Y f_y}{Z} + c_y \right)^\top, \tag{2}$$

where f_x, f_y are the focal lengths and c_x, c_y is the camera center. Together with the depth $Z = D(\mathrm{x})$ a 3D point p can be reconstructed using the inverse projection function π^{-1} defined by

$$p = \pi^{-1}(\mathrm{x}, D(\mathrm{x})) = \left(\frac{x - c_x}{f_x} D(\mathrm{x}), \frac{y - c_y}{f_y} D(\mathrm{x}), D(\mathrm{x}), 1 \right)^\top. \tag{3}$$

An essential assumption for Eq. 3 is, that the intensity image I and the depth image D coincide. This means that corresponding pixels in the two images are located on the same image coordinates. For some devices (e.g. Kinect v2) this assumption does not hold for the raw images and an additional pre-processing step is necessary. We detail our pre-processing in the Supplementary Material.

The derivatives of an image $A \in \{I, D\}$ are defined via the local neighborhoods. This means, for each pixel coordinate $\mathrm{x} = (x, y)^\top$ the derivative can be computed for x and y direction by writing forward-backward differences as

$$\begin{aligned} A_{dx}(x, y) &= \frac{A(x-1, y) - A(x+1, y)}{2} \quad \text{and} \\ A_{dy}(x, y) &= \frac{A(x, y-1) - A(x, y+1)}{2}. \end{aligned} \tag{4}$$

When applying Eq. 4 on the depth images, one must be careful because of invalid depth measurements. For some pixels the depth sensor is (independent of *Pattern Projection* or ToF) not able to measure the depth leading to invalid values often represented as zero. In that case the step size must be locally increased or decreased in order to achieve correct derivative values.

A 3D point p is transformed to the coordinate frame of the second camera according to the camera motion ξ, which we define as a rigid body motion $\xi \in \mathrm{SE}(3)$. A rigid body motion comprises a rotation $R \in \mathrm{SO}(3)$ and a translation $t \in \mathbb{R}^3$. The rotation R can be expressed by a 3×3 matrix and the translation t by a 3×1 matrix leading to a 4×4 transformation matrix

$$T_{4\times4} = \begin{pmatrix} R_{3\times3} & t_{3\times1} \\ 0 & 1 \end{pmatrix}. \tag{5}$$

Hence, a point p' in the frame of the second camera is given as

$$p' = \xi(p) = Tp. \tag{6}$$

Since T has twelve parameters, whereas a rigid body motion has only six Degrees Of Freedom (DOF) (three DOF of rotation and three DOF for translation),

we use – like many others [9,10,17,18] – a minimal parametrization of ξ using twist coordinates given by the Lie algebra. For more information about the Lie algebra we refer to [22]. Summarizing Eqs. 2, 3 and 6 a warping function τ is given by

$$\tau(\xi, \mathrm{x}) = \pi\left(\xi\left(\pi^{-1}\left(\mathrm{x}, D(\mathrm{x})\right)\right)\right), \tag{7}$$

which maps a pixel x from the first image to the second image using the camera motion ξ.

4.2 Residual Definition

Given the warping function τ we can define – like many state-of-the-art papers [9,13] – for each pixel x the residuals r_I and r_D for the intensity image I and the depth image D respectively as

$$\begin{aligned} r_I(\mathrm{x}, \xi) &= I_2(\tau(\xi, \mathrm{x})) - I_1(\mathrm{x}) \quad \text{and} \\ r_D(\mathrm{x}, \xi) &= D_2(\tau(\xi, \mathrm{x})) - [\xi\left(\pi^{-1}\left(\mathrm{x}, D_1(\mathrm{x})\right)\right)]_Z, \end{aligned} \tag{8}$$

where $[.]_Z$ is the z component of a 3D point, which is equivalent to the depth. Note, the depth residual r_D resembles the geometric error in variants of ICP [23]. In DNA-SLAM we want to explicitly use the derivatives of the depth image to derive a indicator for the reliability of pixels. In general we would have to consider four values, namely $D_{dx,1}$, $D_{dy,1}$ for the first image and $D_{dx,2}$, $D_{dy,2}$ for the second image. To reduce the complexity we also define the influence of the derivatives as residuals r_{Dx} and r_{Dy} by

$$\begin{aligned} r_{Dx}(\mathrm{x}, \xi) &= D_{dx,2}(\tau(\xi, \mathrm{x})) - D_{dx,1}(\mathrm{x}) \quad \text{and} \\ r_{Dy}(\mathrm{x}, \xi) &= D_{dy,2}(\tau(\xi, \mathrm{x})) - D_{dy,1}(\mathrm{x}), \end{aligned} \tag{9}$$

where D_{dx} and D_{dy} are the derivatives as defined in Eq. 4. This has the advantage of considering only two values representing the derivatives for the weight computation. If an unreliable point (indicated by a high derivative) occurs at one of the pixel coordinates x or $\tau(\xi, \mathrm{x})$, the residual will be high too. The case that both pixel coordinates are unreliable is extremely rare as asserted during our tests. Thus, the residuals r_{Dx} and r_{Dy} of Eq. 9 are an authentic indicator for the reliability of the given pixels.

At the end, the residuals are used for both weight computation and motion computation. Since we want to use the derivatives only for the weight computation, we define two residuals vector R and Ω containing the residuals r_I, r_D, r_{Dx} and r_{Dy} of Eqs. 8 and 9 by

$$R(\mathrm{x}, \xi) = \begin{bmatrix} r_I(\mathrm{x}, \xi) \\ r_D(\mathrm{x}, \xi) \end{bmatrix} \quad \text{and} \quad \Omega(\mathrm{x}, \xi) = \begin{bmatrix} r_I(\mathrm{x}, \xi) \\ r_D(\mathrm{x}, \xi) \\ r_{Dx}(\mathrm{x}, \xi) \\ r_{Dy}(\mathrm{x}, \xi) \end{bmatrix}, \tag{10}$$

where $R \in \mathbb{R}^2$ is a two-dimensional vector and $\Omega \in \mathbb{R}^4$ is a four-dimensional vector.

4.3 Noise Aware Weighting and Scale Definition

In this section, we want to determine a weight w for each pixel coordinate $\mathrm{x} = (x, y)^\top$, which states how reliable the pixel at x is. As stated in Sect. 4.2 the weight should be determined based on the residual vector Ω. A straightforward approach would be to combine the single residuals in Ω linearly. State-of-the-art approaches use therefore heuristic [24] or manually chosen weights [14]. However, we want to limit the influence of pixels, where exactly one or combinations of the following holds: high derivatives, high intensity error, or high depth error. Kerl et al. [9] showed already that intensity and depth errors follow the t-distribution with $\nu = 5$ DOF. Inspired by [18] we model the residual vector Ω as a multivariate t-distribution. In Sect. 5.1 we show that also the derivate residual follows the t-distribution with $\nu = 5$ DOF. The weights can then be computed by

$$w(\mathrm{x}, \xi) = \frac{\nu + 1}{\nu + \Omega(\mathrm{x}, \xi)^\top \Sigma_\Omega(\xi)^{-1} \Omega(\mathrm{x}, \xi)}, \tag{11}$$

where Σ_Ω is the covariance matrix of the residual vector Ω. It is defined for all n valid pixels in an image as

$$\Sigma_\Omega(\xi) = \frac{1}{n} \sum_{i}^{n} \Omega(i, \xi) \Omega(i, \xi)^\top \frac{\nu + 1}{\nu + \Omega(i, \xi) \Sigma'_\Omega(\xi)^{-1} \Omega(i, \xi)^\top}, \tag{12}$$

where Σ'_Ω is the covariance matrix of the previous iteration. We will detail the iterative process in Sect. 4.4. Note, while state-of-the-art algorithms [18] work with a 2×2 covariance matrix, we use a 4×4 covariance matrix containing also the derivatives. The covariance matrix Σ_Ω has the task to automatically scale the weights, so that heuristic [24] or manually chosen weights [14] are not necessary.

4.4 Iterative Re-weighted Least Square

In this section, we demonstrate how the camera motion ξ can be estimated with the help of an iterative re-weighted least square. Equation 1 already shows the basic principle: the camera motion ξ is computed by minimizing residuals. For DNA-SLAM we follow state-of-the-art algorithms [13,18,19] and minimize the photometric and geometric error, which is summarized in the residuals $R(\mathrm{x}, \xi)$. Minimizing the residuals $\Omega(\mathrm{x}, \xi)$ is not useful here, because the depth derivatives are not precise enough; even with our sophisticated weighting of Sect. 4.3. The basic formula for the motion estimation in Eq. 1 changes for the multivariate case that we use in DNA-SLAM to

$$\xi = \arg\min_{\xi} \sum_{i}^{n} w(i, \xi) R(i, \xi)^\top \Sigma_R(\xi)^{-1} R(i, \xi). \tag{13}$$

Note, the 2×2 covariance matrix $\Sigma_R(\xi)$ is used here, since also the residuals $R(\mathrm{x}, \xi)$ are used. However, the weights $w(\mathrm{x}, \xi)$ are computed with the residuals $\Omega(\mathrm{x}, \xi)$ and covariance matrix $\Sigma_\Omega(\xi)$ as stated in Eq. 11.

The residuals $R(\mathrm{x}, \xi)$ in Eq. 10 are non-linear in the camera motion ξ. Therefore, we follow [9,15,18] and linearize Eq. 13 around ξ using the first order Taylor expansion. This reformulation is possible, because the motion between the frames is very small as the frame rate of ToF cameras is usually rather high (e.g. Kinect v2: 30 Hz). In addition, we utilize a coarse-to-fine strategy to ensure small camera motion as detailed in Sect. 4.4. The resulting normal equations of the non-linear least square problem are [18]:

$$\sum_{i}^{n} w(i,\xi) J(i,\xi)^{\top} \Sigma_R(\xi)^{-1} J(i,\xi) \Delta\xi = -\sum_{i}^{n} w(i,\xi) J(i,\xi)^{\top} \Sigma_R(\xi)^{-1} R(i,\xi), \tag{14}$$

where $J(i,\xi)$ is the 2×6 matrix containing the derivatives of $R(i,\xi)$ with respect to ξ. The linear equation system in (14) can be efficiently solved for the motion increments $\Delta\xi$ using Cholesky decomposition [9]. In each iteration the equation system is re-weighted by re-estimating the weights $w(i,\xi)$ and the scale matrix $\Sigma_\Omega(\xi)$ as shown in Fig. 2. The trajectory of the camera is determined by concatenating the motion estimations between single frames.

For the Taylor expansion in Eq. 14 we must ensure small camera motions. Thus, we utilize a coarse-to-fine strategy by building an image pyramid. Therefore, we subsample each image tuple (I_i, D_i) by halving the image resolution for each level. For subsampling the depth images it is important to consider the invalid depth values. The camera motion ξ is estimated in the coarsest level first and then propagated to finer levels as a prior. In our experiments we were using four pyramid levels. The finest level is not used, since the results are almost unchanged, but runtime increases significantly.

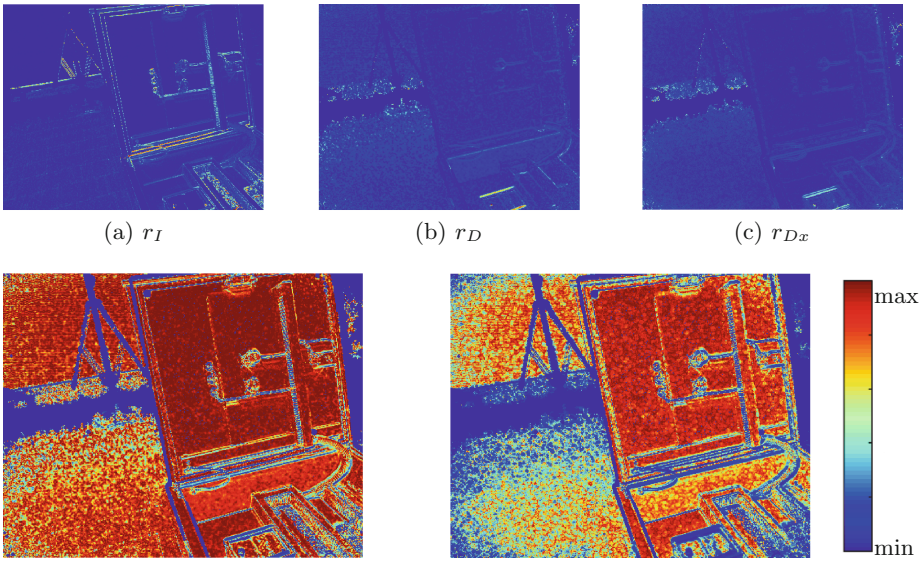
5 Evaluation

In order to obtain meaningful evaluation results we make use of two publicly available benchmarks [25,26]. First, we use the CoRBS benchmark [26], which was recently published and is the only dataset providing real ToF image data of the Kinect v2 together with ground truth trajectories for the camera and ground truth reconstructions of the scene. The ground truth trajectory of CoRBS was acquired with an external motion capture system with sub-millimeter precision. This is an ideal basis for the evaluation of our new algorithm. In addition, we also use the TUM RGB-D benchmark [25], which uses the Kinect v1 and also provides a ground truth camera trajectory. We use this dataset in order to show the applicability of our new algorithm to Kinect v1. Furthermore, many previous algorithms were benchmarked on this dataset.

For the evaluation of the estimated camera trajectory we follow [25,26] and use the Relative Pose Error (RPE). It measures the drift of the estimated trajectory compared to a given ground truth over a fixed time interval Δ in m/s. For more details on that measurement please refer to the Supplementary Material.

5.1 Noise Aware Weights

One of the key contributions in this paper is to estimate the reliability of each pixel individually and integrate it into a sophisticated weighing scheme. As motivated in Sect. 3 we use the depth derivatives to judge the reliability of pixels. Together with the photometric and depth residuals they are transformed to weights as defined in Eq. 11. In Fig. 3(a), (b) and (c) the inputs for the weighting function of our DNA-SLAM are depicted for an exemplary frame of the CoRBS [26] dataset. While the intensity r_I and depth r_D residuals are high close to edges, the new r_{Dx} is high in unreliable regions. These are e.g. points on the reflective floor, in the image corners and flying pixels. In Fig. 3(d) the weights without the new reliability estimation r_{Dx} are shown as used in DVO [18]. Here, only points close to edges are down-weighted, whereas the remaining point have almost equally high weights. In Fig. 3(e) the weights with the new reliability estimation r_{Dx} are shown as applied in DNA-SLAM. Here, the points close to edges are down-weighted too, but also regions like e.g. the floor and the image corners are down-weighted. In total, the different weights are much more distributed and selective compared to Fig. 3(d). From a first impression one might claim the weight values w_R are only lower by a constant value or factor



(a) r_I (b) r_D (c) r_{Dx}

(d) w_R (used e.g. in [9, 18]) (e) w_Ω (used in DNA-SLAM)

Fig. 3. Our DNA-SLAM uses a noise aware weighting scheme. Weights are computed based on (a) intensity residuals, (b) depth residuals and (c) depth derivatives. While (d) the weights in state-of-the-art algorithms are quite clustered, (e) DNA-SLAM estimates the weights for each pixel very selective, representing the individual reliability. Points on the reflective floor, in the image corners and *flying pixels* are clearly down-weighted. (Color figure online)

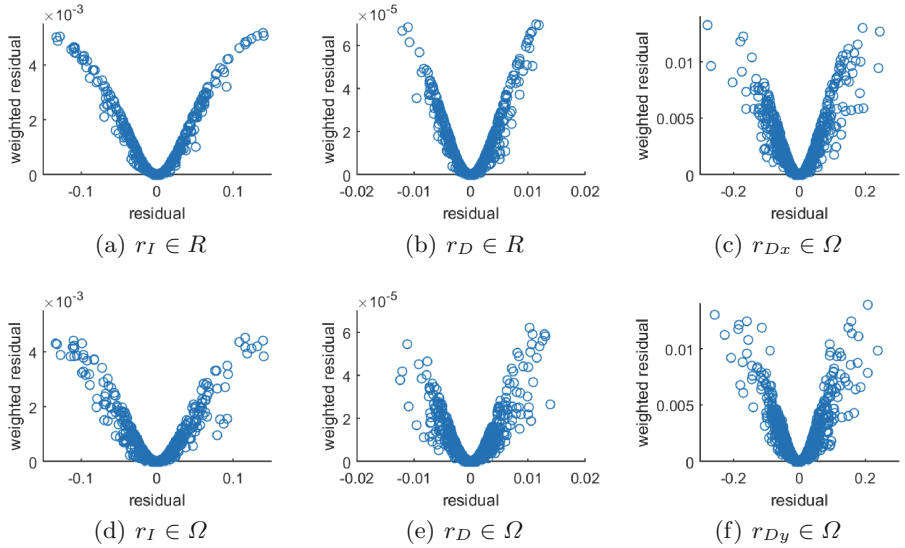


Fig. 4. In DNA-SLAM we weight residuals r_i by Eq. 1 to $w(r_i)r_i^2$. (a) and (b) show the weighted residuals of R, in contrast (d) and (e) use our new residuals Ω including depth derivatives (cp. Eq. 10). (c) and (f) show that also the derivatives follow the t-distribution.

compared to w_Ω. However, this is a non-linear transformation. For example, the cab of the electrical cabinet is transformed from dark red to red, whereas the glossy floor is transformed much stronger from red to green/blue. This shows the effectiveness of the new proposed weighting. Another argument is that the weights are scaled by the (co-)variance (see Eq. 12). A scaling of the weights by a certain value would be neutralized by the variance. The effects of this new weighting on the trajectory accuracy is shown in Sect. 5.2. We also tested binary weights for down-weighting unreliable pixels, but the results were poor.

The definition of weights (cp. Eq. 11) assumes a t-distribution of the used derivative residuals r_{Dx} and r_{Dy}. In Fig. 4 we verified this assumption. (a) and (b) show the weighted residuals, if only R (cp. Eq. 10) is utilized for weight computation as applied in DVO [18]. The plots verify that $r_I, r_D \in R$ follow clearly the t-distribution. If we use our new Ω (cp. Eq. 10) for weight computation, the plots change as illustrated in Fig. 4(d) and (e). The residuals $r_I, r_D \in \Omega$ still follow the t-distribution, but some values are down-weighted. These are values which were detected as unreliable. In addition, Fig. 4(c) and (f) show that our new residuals $r_{Dx}, r_{Dy} \in \Omega$ follow the t-distribution.

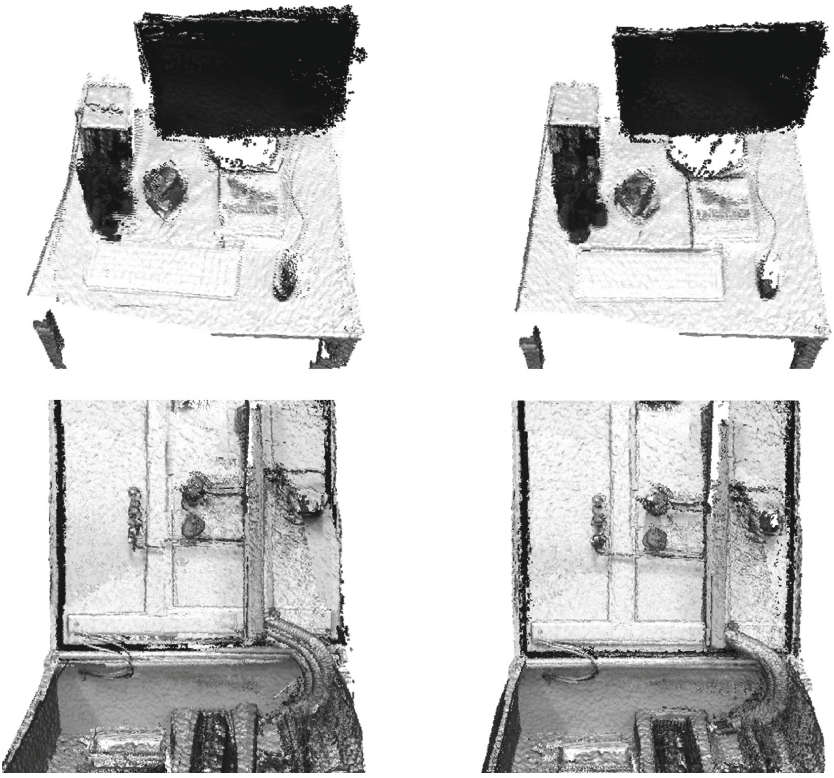
5.2 SLAM Results

In this section, we evaluate our DNA-SLAM with public datasets and compare it against state-of-the-art algorithms. For the implementation of DNA-SLAM we

Table 1. RMSE of the translational and rotational drift (RPE) in m/s and deg/s respectively for different sequences of the CoRBS dataset [26] using the Kinect v2 ToF RGB-D camera. Best result are bold. DNA-SLAM performs superior in most sequences or is at least on par.

Algorithm	D1		D2		D4	
	E_{trans}	E_{rot}	E_{trans}	E_{rot}	E_{trans}	E_{rot}
ICP [23]	0.0443	2.0819	0.0317	1.9360	0.0420	2.3651
DVO [18]	0.0596	2.5411	0.0410	2.1356	0.0335	1.5915
DNA-SLAM (*ours*)	**0.0266**	**0.9702**	**0.0209**	**1.1008**	**0.0223**	**0.9988**
Algorithm	E1		E4		E5	
	E_{trans}	E_{rot}	E_{trans}	E_{rot}	E_{trans}	E_{rot}
ICP [23]	0.0770	4.4267	0.0569	2.9581	0.0453	4.1031
DVO [18]	**0.0335**	1.5915	0.0335	**1.5915**	0.0309	1.7145
DNA-SLAM (*ours*)	0.0349	**1.4264**	**0.0264**	1.7547	**0.0143**	**0.7128**
Algorithm	H1		H2		H3	
	E_{trans}	E_{rot}	E_{trans}	E_{rot}	E_{trans}	E_{rot}
ICP [23]	0.0559	2.3807	0.0412	2.3018	0.0676	2.9232
DVO [18]	0.0616	2.3214	0.0251	1.5564	0.0827	3.2376
DNA-SLAM (*ours*)	**0.0196**	**0.7248**	**0.0124**	**0.6778**	**0.0300**	**1.4728**

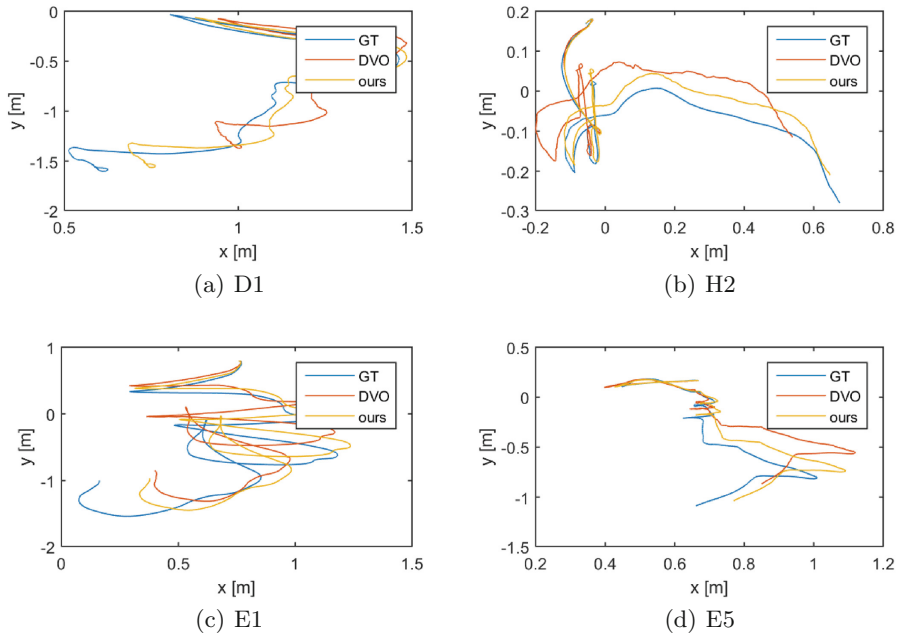
build on top of the open source implementation of DVO [18]. First, we make use of the CoRBS dataset [26], which uses the Kinect v2 ToF RGB-D camera. We run the DNA-SLAM algorithm with several sequences and measure the relative translational as well as the rotational error of the estimated trajectories. In order to evaluate the effect of the new weighting, we compare against the state-of-the-art algorithm DVO [18] as well as ICP [23,27] which is used e.g. in KinectFusion [28] and others [14,29]. We apply DVO and ICP without keyframes, since we want to measure the frame-to-frame drift of the algorithms. The quantitative evaluation in Table 1 verifies the superior accuracy of our DNA-SLAM. The trajectories estimated with ICP exhibit a lower accuracy than the new DNA-SLAM. Compared to DVO the relative translational as well as the rotational error was substantially reduced with DNA-SLAM in most sequences. For the sequences E1 and E4 we achieve an accuracy on par. Figure 6 gives a visual impression of the trajectories. In all sequences DNA-SLAM is closer to the ground truth (GT) than DVO. Figure 5 shows the mappings of two exemplary sequences. The maps of DNA-SLAM are much more accurate, which is a logical consequence of the reduced drift compared to DVO. Summarized, we can conclude that our new weighting scheme substantially reduces the drift in dense motion estimation with ToF cameras.



(a) DVO [18] (b) DNA-SLAM

Fig. 5. Visual comparison of the mapped geometry with DVO [18] and DNA-SLAM using the CoRBS [26] sequences D1 and E5. DNA-SLAM delivers a higher accuracy.

For the sake of completeness we tested DNA-SLAM also with the TUM dataset [25], which utilizes the Kinect v1 using *Pattern Projection* for depth estimation. Since this dataset is already available for some years, several state-of-the-art algorithms [8,10,14,17–19] used it for evaluation. In Table 2 we state the relative translational errors. Here, we are slightly better than DVO, which is the most similar algorithm. Some state-of-the-art algorithms [10,19] deliver more accurate results, but they are also using keyframes adulterating comparison. Summarized, the evaluation shows that our algorithm is also applicable with Kinect v1 delivering state-of-the-art results. However, when using *Time-of-Flight* cameras like the Kinect v2 our algorithm shows clearly superior accuracy.



(a) D1 (b) H2

(c) E1 (d) E5

Fig. 6. Estimated trajectories for different sequences of the CoRBS dataset [26]. DNA-SLAM is closer to the ground truth (GT) trajectories than the state-of-the-art algorithm DVO [18]. See Table 1 for a quantitative evaluation.

Table 2. RMSE of the translational drift (RPE) in m/s for selected sequences of the TUM dataset [25] using the Kinect v1 with *Pattern Projection*. DNA-SLAM achieves state-of-the-art results even with this camera while being designed for ToF cameras; only algorithms with keyframes (marked with *) perform better.

Algorithm	Author	fr1/desk	fr1/desk2	fr1/room	fr2/desk
FOVIS	Huang et al. [8]	0.0604*	-	0.0642*	0.0136*
	Whelan et al. [14]	0.0393	-	0.0622	0.0208
	Klose et al. [17]	0.0302	0.0526	0.0397	0.0147
VP	Meilland et al. [19]	0.0259*	-	0.0351*	0.0147*
	Gutierrez et al. [10]	0.0260*	0.0387*	0.0491*	0.0121*
DVO	Kerl et al. [18]	0.036	0.049	0.058	-
DNA-SLAM	*ours*	0.0333	0.0482	0.0498	0.0195

6 Conclusion and Future Work

In this paper, we proposed our new DNA-SLAM algorithm, which addresses especially ToF RGB-D cameras. We discovered that the noise in the depth image leads to inaccuracies in the camera motion estimation, unless not

especially treated. In our experiments we verified that the local depth derivatives are a good indicator for the depth pixel reliability. Thus, DNA-SLAM estimates the reliability of each pixel individually and transforms that into a weighting scheme. In the evaluation with the public CoRBS benchmark [26] we showed the substantially reduced drift of DNA-SLAM compared to state-of-the-art algorithms. Thus, we demonstrated that a sophisticated weighting scheme can compensate the errors introduced by ToF depth cameras leading to superior localization and mapping results. For a short summary see also the supplementary video.

In future work, we will integrate a keyframe selection and extend the tracking from frame-to-frame to frame-to-keyframe. This is a straightforward procedure and was already applied in several works [10,18,19]. We expect a further improvment in accuracy, which is required for several applications [30]. In addition, we will speed-up the implementation, since our current CPU version requires around 100 ms for a single motion estimation. A major speed-up can be achieved by porting the algorithm either to SSE [18] or to a GPU [31] as already shown in the literature.

References

1. Davison, A.J.: Real-time simultaneous localisation and mapping with a single camera. In: International Conference on Computer Vision (ICCV), pp. 1403–1410. IEEE (2003)
2. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: International Symposium on Mixed and Augmented Reality (ISMAR), pp. 225–234. IEEE (2007)
3. Newcombe, R.A., Lovegrove, S.J., Davison, A.J.: DTAM: Dense tracking and mapping in real-time. In: International Conference on Computer Vision (ICCV), pp. 2320–2327. IEEE (2011)
4. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: large-scale direct monocular SLAM. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8690, pp. 834–849. Springer, Cham (2014). doi:10.1007/978-3-319-10605-2_54
5. Microsoft: (Kinect v2). www.microsoft.com/en-us/kinectforwindows/
6. Google: (Tango). www.google.com/atap/project-tango/
7. Engelhard, N., Endres, F., Hess, J., Sturm, J., Burgard, W.: Real-time 3D visual slam with a hand-held RGB-D camera. In: RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, vol. 180 (2011)
8. Huang, A.S., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., Roy, N.: Visual odometry and mapping for autonomous flight using an RGB-D camera. In: International Symposium on Robotics Research (ISRR), vol. 2 (2011)
9. Kerl, C., Sturm, J., Cremers, D.: Robust odometry estimation for RGB-D cameras. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 3748–3754. IEEE (2013)
10. Gutierrez-Gomez, D., Mayol-Cuevas, W., Guerrero, J.: Dense RGB-D visual odometry using inverse depth. Robot. Auton. Syst. **75**, 571–583 (2016)

11. Brunetto, N., Fioraio, N., Stefano, L.: Interactive RGB-D SLAM on mobile devices. In: Jawahar, C.V., Shan, S. (eds.) ACCV 2014. LNCS, vol. 9010, pp. 339–351. Springer, Cham (2015). doi:10.1007/978-3-319-16634-6_25
12. Belter, D., Nowicki, M., Skrzypczyński, P.: On the performance of pose-based RGB-D visual navigation systems. In: Cremers, D., Reid, I., Saito, H., Yang, M.-H. (eds.) ACCV 2014. LNCS, vol. 9004, pp. 407–423. Springer, Cham (2015). doi:10.1007/978-3-319-16808-1_28
13. Ma, L., Kerl, C., Stueckler, J., Cremers, D.: CPA-SLAM: consistent plane-model alignment for direct RGB-D slam. In: International Conference on Robotics and Automation (ICRA) (2016)
14. Whelan, T., Johannsson, H., Kaess, M., Leonard, J.J., McDonald, J.: Robust real-time visual odometry for dense RGB-D mapping. In: International Conference on Robotics and Automation (ICRA), pp. 5724–5731. IEEE (2013)
15. Steinbruecker, F., Sturm, J., Cremers, D.: Real-time visual odometry from dense RGB-D images. In: International Conference on Computer Vision Workshop (ICCV Workshop) (2011)
16. Audras, C., Comport, A., Meilland, M., Rives, P.: Real-time dense appearance-based slam for RGB-D sensors. In: Australasian Conference on Robotics and Automation (ACRA) (2011)
17. Klose, S., Heise, P., Knoll, A.: Efficient compositional approaches for real-time robust direct visual odometry from RGB-D data. In: International Conference on Intelligent Robots and Systems (IROS), pp. 1100–1106. IEEE (2013)
18. Kerl, C., Sturm, J., Cremers, D.: Dense visual slam for RGB-D cameras. In: International Conference on Intelligent Robot Systems (IROS) (2013)
19. Meilland, M., Comport, A.I.: On unifying key-frame and voxel-based dense visual slam at large scales. In: International Conference on Intelligent Robots and Systems (IROS), pp. 3677–3683. IEEE (2013)
20. Kerl, C., Stueckler, J., Cremers, D.: (Dense continuous-time tracking and mapping with rolling shutter RGB-D cameras)
21. Wasenmüller, O., Stricker, D.: Comparison of kinect v1 and v2 depth images in terms of accuracy and precision. In: Chen, C.-S., Lu, J., Ma, K.-K. (eds.) ACCV 2016 Workshops, Part II. LNCS, vol. 10116, pp. 34–45. Springer, Cham (2017)
22. Ma, Y., Soatto, S., Kosecka, J., Sastry, S.S.: An invitation to 3-d vision: from images to geometric models, vol. 26. Springer Science & Business Media, New York (2012)
23. Besl, P.J., McKay, N.D.: Method for registration of 3-d shapes. In: Robotics-DL Tentative, pp. 586–606. International Society for Optics and Photonics (1992)
24. Tykkälä, T., Audras, C., Comport, A.I.: Direct iterative closest point for real-time visual odometry. In: International Conference on Computer Vision Workshops (ICCV Workshops), pp. 2050–2056. IEEE (2011)
25. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D slam systems. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 573–580. IEEE (2012)
26. Wasenmüller, O., Meyer, M., Stricker, D.: CoRBS: comprehensive RGB-D benchmark for slam using kinect v2. In: IEEE Winter Conference on Applications of Computer Vision (WACV). IEEE (2016)
27. Rusinkiewicz, S., Levoy, M.: Efficient variants of the ICP algorithm. In: IEEE International Conference on 3D Digital Imaging and Modeling, pp. 145–152. IEEE (2001)
28. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohi, P., Shotton, J., Hodges, S., Fitzgibbon, A.: Kinectfusion: real-time dense

surface mapping and tracking. In: IEEE International Symposium on Mixed and Augmented Reality (ISMAR) (2011)
29. Lin, Y.C., Chen, C.Y., Huang, S.W., Huang, P.S., Chen, C.F.: Registration and merging of large scale range data using an improved ICP algorithm approach. In: International Conference Image and Vision Computing (IVCNZ) (2011)
30. Wasenmüller, O., Meyer, M., Stricker, D.: Augmented reality 3D discrepancy check in industrial applications. In: IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 125–134. IEEE (2016)
31. Lee, D., Kim, H., Myung, H.: Gpu-based real-time RGB-D 3D slam. In: International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), pp. 46–48. IEEE (2012)