# On-the-fly Particle Filter Registration for Laser Data

Christian Rink, Simon Kriegel, Jakob Hasse, Zoltan-Csaba Marton
Institute of Robotics and Mechatronics, DLR (German Aerospace Center), 82234 Wessling, Germany
Email: christian.rink@dlr.de

***Abstract*—This work is focused on streaming particle filter registration of surface models such as homogeneous triangle meshes and point clouds. Part of the approach is a streaming curvature feature calculation. The investigated approach utilizes a particle filter to incrementally update pose estimates during data acquisition. The method is evaluated in real data experiments with a high-precision laser striper system attached to an industrial robot. During the laser scan, the data is integrated on-the-fly in order to calculate features and based on these to estimate the object's pose. Experiments show the method's competitiveness in accuracy and reliability compared to state-of-the-art offline algorithms.**

## I. Introduction

Object registration is required in a wide range of technical applications, such as computer-assisted surgery, rapid prototyping, reverse engineering, or manipulation tasks in manufacturing processes. The main application of the proposed method is in autonomous 3D modeling scenarios with laser scanners as investigated by Kriegel et al. [1]. The therein presented autonomous modeling approach does not address the problem of replacing the objects. However, this is necessary if e.g. the bottom or other initially occluded parts of the object are to be modeled and requires pose estimation. The advance of low-cost off-the-shelf 3D sensors catalyzed the application of pose estimation techniques. Therefore, many current registration methods are designed for data generated by 3D sensors such as Kinect. However, laser stripers are more accurate and can deal better with shiny or black objects, which is required for high quality 3D-modeling [2]. Although most of the existing registration methods work on laser data as well, they suffer from the time consuming data acquisition process of laser scanners, as they do not exploit the fact that partial data arrives continuously, i.e. in a data stream. Moreover, no global streaming method exists that is based purely on dense depth-images and works with laser stripers.

In this work, we fill this gap by developing a novel method satisfying these requirements. The registration method works on streaming data, starts with data acquisition and updates pose estimates on-the-fly based on new data points. The benefit is twofold: First, time is saved due to parallel processing of data acquisition and pose estimation during laser scanning. Second, a laser scan can be aborted before the complete scan is carried out once the acquired data is sufficient for robust pose estimation. To the best of our knowledge, no feature based global streaming pose estimation method in the literature exists, that is based on pure geometrical information.

978-1-4673-8692-0/16/$31.00 

## II. Related Work

For pose estimation, Fischler et al. [3] introduced the well-known random sampling consensus (RANSAC), and successively Chen et al. [4] demonstrated its application to registration. Variants typically calculate rigid motions from subsets of points or point-normal pairs [5], [6] or higher-dimensional features [7] that are sampled in the datasets. Unfortunately, these RANSAC-based methods cannot be adopted to work with streaming data, as a uniform sampling of points cannot be achieved before all data is acquired.

Another group of algorithms tries to group correspondences [8] or exploit salient points [9]. Also this class of algorithms cannot be adopted to work with streaming data, because global data-sets are needed, especially for feature calculation.

Recently, Rink et al. [10] formulated the approach of Barequet et al. [11] as particle filter problem. They utilize the unique decomposability of rigid motions into rotations and translations. Rotations are sampled and weighted with a measure of how clustered the corresponding set of translations is. In mobile robotics, particles filters based on pure depth images proved to work incrementally on robust, as detailed in the comprehensive overview given by Sturm et al. [12]. Unfortunately, these approaches are not suitable for the stated problem, as the weighting is time consuming. In contrast to classic mobile robotics our estimation problem is not in 2D, and thus we need a much larger number of particles. Additionally, we have to cope with higher update rates.

The weighting of rotations as proposed by [10] and [11] is heuristic and slow. Therefore, we propose sampling on the space of rigid body transformations and weight these with a fast and theoretically sound method. Apart from this, the contributions of this work comprise a streaming curvature calculation and streaming pose estimation, based purely on geometrical information obtained by a moving laser striper. Real data experiments confirm the method's effectiveness and reliability, compared to available state-of-the-art offline methods. We emphasize that the advantage in computation time is obvious due to the streaming nature of the method and therefore this point is not addressed in the experiments. Nevertheless, it is crucial for many applications.

## III. System Overview

The proposed method integrates three modules: the *3D Data Acquisition*, the *Depth Image Stream Processing* and the *Particle Filter* module, as depicted in Fig. 1. The components of the *Depth Image Stream Processing* and the *Particle Filter* are described in more detail in the following sections.

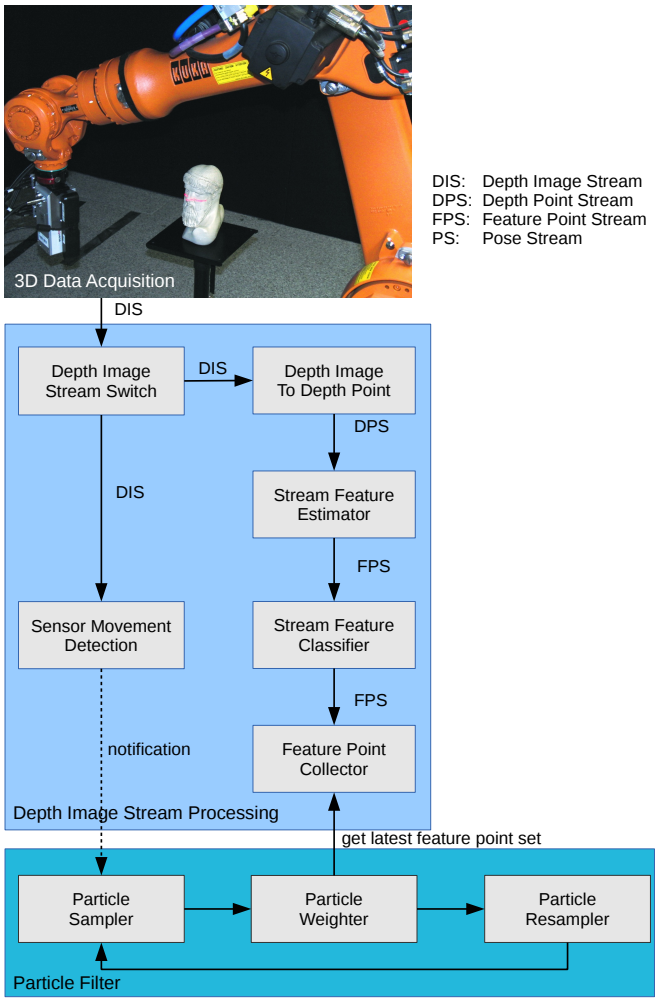



Fig. 1. The streaming pose estimation workflow is divided into three main modules: the *3D Data Acquisition*, the *Depth Image Stream Processing*, and the *Particle Filter*. Each module contains different components.

When data acquisition is started the *3D Data Acquisition* module synchronizes the depth images with the pose information and provides a depth image stream with depth images that contain the pose information [13]. The depth image stream is handled to the *Depth Image Stream Processing* module. Here, the stream is divided into two different streams by the *Depth Image Stream Switch* component.

One of it ends in a notification to the *Particle Sampler* component of the *Particle Filter* module. The notification is based on the pose information contained in the depth images, which is extracted and inspected by the *Sensor Movement Detection* component. The *Particle Sampler* is notified as soon as the pose information indicates that the sensor has moved significantly. This is the case if the translatory or rotatory difference to the last pose used for an update exceeds some predefined thresholds $th_t, th_R$, respectively.

The second stream is used for feature point calculation. Here, the depth images are first converted to depth points and passed to the *Stream Feature Estimator* component as described in Sect. IV. Here, a feature point stream is handled to the *Stream Feature Classifier* component, which classifies the features according to the class borders of the template. Finally, the feature point stream is collected in the *Feature Point Collector* component from which the *Particle Weighter* component acquires only the latest feature points on demand.

The *Particle Filter* module itself contains the *Particle Sampler*, the *Particle Weighter* and the *Particle Resampler* components, and works as described in Sect. V. The *Particle Sampler* component starts sampling the particles when it is notified by the *Sensor Movement Detection* component. It performs a neighborhood sampling of the transformation particles according to Sect. VI. When finished, the *Particle Weighter* component is called, which acquires the latest feature points from the *Feature Point Collector* component. The particle weighting is carried out according to Sect. VI-B. After weighting, the particles are resampled with an importance resampling step.

## IV. Features

In this work, we adopt the streaming mesh reconstruction of Bodenmüller [14] to produce a feature point set instead of a mesh. Similar offline variants of the proposed features have been introduced in [10] for the purpose of offline registration and proved to be robust under noisy depth measurements.

Every feature point $p = (c_p, n_p, v_p) \in \mathbb{R}^3 \times \mathcal{S}^2 \times \mathbb{R}$ consists of a coordinate $c_p$, a surface normal $n_p$, ($\mathcal{S}^2$ being the unit sphere) and a feature value $v_p$. Let in the following $p$ be a point with surface normal $n_p$ and a neighborhood $N(p)$ and define

$$c(p,q) := \cos(n_p, \frac{q-p}{||q-p||})$$

for a neighbor $q \in N(p)$. Then, the mean, maximum and minimum of $\{c(p,q)|q \in N(p)\}$ are called the mean normal cosine (MNC), maximum normal cosine (MaNC) and minimum normal cosine (MiNC) in $p$ with neighborhood $N(p)$, respectively.

The processing pipeline consists of three stages: the *density limitation*, the *normal estimation* and the *feature generation* step. Depth points coming from a real-time data stream are incrementally inserted into the model if they pass a limitation test: each newly acquired point, that is closer than a distance $r_r$ to any point already inserted to the model, is rejected. Thus, the entire Euclidean point density of the model is limited and the computational effort can be controlled. For each point that passes the *density limitation*, a surface normal is estimated using principal component analysis for all points within a spherical neighborhood with radius $r_n$. Only points, for which the *normal estimation* is considered robust (see [14] for details), are transferred to the subsequent *feature generation* step. The proposed angle features are calculated from the point surface normal and the neighborhood points in the model. Consequently, if a stable normal is ready and the new point is inserted into the feature calculation module the MNC, MaNC, or MiNC are calculated in the neighborhood of radius $r_n$ (immediately available from the *normal estimation*

stage). Also, all the points in the neighborhood are updated correspondingly.

## V. Monte Carlo Registration

In contrast to [10], we propose to sample on the complete space of rigid body transformations, since calculating the score function from [10] would be too slow for streaming pose estimation. In the following, we will shortly review the most important facts and describe our new method in detail afterwards.

The space of rigid body transformations is denoted $\mathcal{T}$. We search the transformation $T_i \in \mathcal{T}$ between a template model $P$ and a measured point cloud model $Q$ of a known object at time step $i$. Each particle comprises a transformation $T \in \mathcal{T}$ and a weight $w$. The state transition between two time steps consists of the systematical change described by the transition function $A_i$ and some error $\epsilon_{A_i}$.

In the special case of registration, $A_i$ can be considered the identity, and the observed model is assumed to be constant over time ($A_i = \mathbf{id}$). The error $\varepsilon_{A_i}$ is assumed to be distributed uniformly in a neighborhood of the identity. Each particle $(T, w)$ in each time step $i$ is weighted by the conditional probability density function (**pdf**) $f(Q_i|T, P)$ of observed data $Q_i$ conditional on the state $T$ and the template model $P$. Thus, particle filter registration can be summarized:

1) Sample an initial set of transformations.
2) Weight each $T$ by $f(Q_i|T, P)$
3) Resample all $T$s according to their weights.
4) Optionally: Adapt the sampling neighborhoods (changing $g(\varepsilon)$) and/or the number of particles.
5) Sample transformations in the neighborhoods of the existing transformations. Return to step 2 if not converged.

In step 4 some parameters are adopted, i.e. the sampling neighborhood, the number of particles and the radius for correspondence search. In this work, we reduce each of the parameters by the same factor of 0.9. For each parameter, a starting value and a minimum value is used. The latter defines a lower bound for the reduction. The initial and minimal local sampling radii for translations are denoted $r_T$ and $r_t$, respectively. The initial and minimal number of particles is denoted $n_P$ and $n_p$. The initial and minimal radii for the correspondence search in the particle weighting are denoted $r_S$ and $r_s$, respectively. We are aware that more flexible approaches would help to improve the performance, e.g. increase radii or number of particles if results become worse during the streaming process. We intentionally simplified the setting, for the sake of a clear and tight description of the basic method.

## VI. Sampling Pose-Particles

A common a priori distribution for expressing ignorance is the uniform distribution. So if no a priori knowledge is available, both the rotational and the translational part of the transformation can be sampled uniformly on the whole parameter space. In other cases, there might be some a priori knowledge: Often the space can be restricted to a cuboid subset on which the transformations are sampled uniformly.

### A. Sampling Rotations and Translations

In this work, we concentrate on sampling transformations uniformly. Sampling rotations from a uniform distribution can be performed in various ways. As outlined in [10], sophisticated approximate methods for deterministic sampling exist, but uniform sampling in a statistical sense has proven to be advantageous. Therefore, we use uniform sampling in $\alpha$-neighborhoods of arbitrary rotations. The $\alpha$-neighborhood $N_\alpha(\mathbf{R})$ of a rotation $\mathbf{R}$ is defined as

$$N_\alpha(\mathbf{R}) \coloneqq \{\tilde{\mathbf{R}} \in \mathcal{R} | d(\mathbf{R}, \tilde{\mathbf{R}}) \leq \alpha\}, \text{ where } \alpha \in [0, \pi]$$

with $d(\mathbf{R}, \tilde{\mathbf{R}})$ being the rotational difference between two rotations $\mathbf{R}, \tilde{\mathbf{R}}$, i.e. the angle of the axis-angle representation of $\mathbf{R} \circ \tilde{\mathbf{R}}^{-1}$. For total ignorance the $\pi$-neighborhood of the identity can be used to sample all rotations.

In the iterative process of the proposed particle filter, in every sampling step the neighborhood radius decreases. Note that the transition function $A$ is supposed to be the identity and the error $\varepsilon$ is supposed to be uniformly distributed in a neighborhood of the identity. Therefore, every transformation is sampled in a neighborhood of itself.

### B. Scoring Transformations

Let $P, Q$ be the feature points of the template and the incoming data feature points, correspondingly. Further, we assume the features to be classified, i.e. $v_p$ is as a discrete category for every $p \in P$ (and correspondingly for $Q$). Now consider a particle describing a transformation $T = (R, t)$. Let $q$ be the feature point in the data corresponding to the feature point $p$ of the template model. If the underlying transformation $T$ between data and template is known, it is reasonable to assume $c_q$ to follow a normal distribution with expectation $T(c_p)$ and some covariance matrix $\Sigma = \sigma^2 \cdot \mathbf{id}$. Then, if we the errors are identically and independently distributed and we consider a set of feature points $\mathbf{p} = \{p_1, \ldots, p_n\}$ and a set of correspondences $\mathbf{q} = \{q_1, \ldots, q_n\}$, we end up at the conditional **pdf** of all feature point locations to be:

$$f(\mathbf{q}|T, \mathbf{p}) \propto \exp -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (T(c_{p_i}) - c_{q_i})^2. \tag{1}$$

In practice, the corresponding $p_i$ are approximated by the nearest feature point to $q_i$ with the same feature class:

$$p_i = \arg \min_{p \in P, v_p = v_{q_i}} d(T(c_p), c_{q_i})$$

If the feature class is erroneous for some $q$, no correct corresponding $p$ will be found in the template. The best we can do is to assume that the feature location is distributed uniformly, conditional on a wrong corresponding $p$. For some distance threshold $r_{max}$ we define

$$d_i \coloneqq \min \{d(T(c_{p_i}) - c_{q_i}), r_{max}\} \tag{2}$$

and adopt Equation (1) to

$$f(\mathbf{q}|T, \mathbf{p}) \propto \exp -\frac{1}{2\sigma^2} \sum_{i=1}^{n} d_i^2. \tag{3}$$

which corresponds to (truncated) normally distributed errors if the correspondences are found within a radius $r_{max}$ and uniformly distributed errors if not (with the density equal to that of the truncated normal at its boundary). Note that this is actually an improper **pdf**, because the integral over every single $p_i$ is unbounded. Nevertheless, we can use it for calculating the importance weights in sampling importance resampling.

Therefore, each particle's transformation $T$ is scored with incoming feature points $Q$ as follows. Each point $q_i$ of $Q$ is classified according to the class borders of the template and is transformed to $T^{-1}(c_{q_i})$. Then, the nearest feature point $p_i$ with the same feature class is searched in the template model and the distance $d_i$ is saved. If no such point is found within the search radius $r_{max}$, the distance is set to that radius ($d_i = r_{max}$). When all distances $d_i$ are calculated for the $n$ feature points, we determine

$$w(t) = \exp(-\frac{1}{2\sigma^2}\sum_{i=1}^{n} d_i^2) \tag{4}$$

*Scoring Variants:* Theoretically, a particle filter uses only statistically independent measurements for each update, that is only new incoming feature points are utilized for $Q_i$. In practice, the particles tend to show improvable convergence results, as will be shown in Sect. VIII. Improved convergence can be achieved, if all previously measured feature points are used for $Q_i$, that is $Q_i = \bigcup_i Q_i$. To distinguish the two scoring variants, we call the former *streaming particle filter registration* (SPFR) and the latter *streaming Monte Carlo registration* (SMCR) in the remainder.

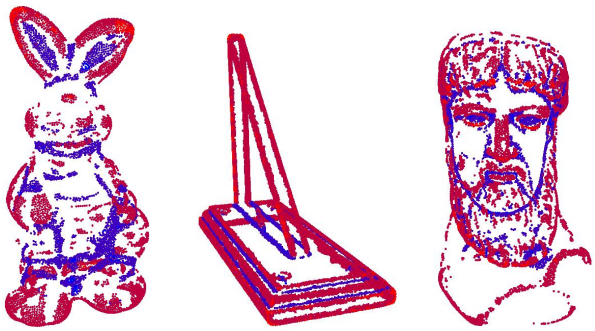
## VII. Test Setup

In this section, experiments with real data are demonstrated. First, the concept is verified and the procedure of streaming pose estimation is visualized with one selected object. Second, we compare the results to available state-of-the-art pose estimation algorithms.

### A. Hardware

As the utilized laser striper is heavy and rather large, the experiments are carried out on an industrial robot. Here, a 6 DOF industrial robot, the Kuka KR16-2, with mounted laser striper is utilized (see Fig. 1 top). For the KR16-2, the absolute positioning error is in millimeter range. The streaming Monte Carlo registration is run on an external computer with Quad Xeon W3520 2.67 GHz CPUs and 6 GB RAM as the Kuka Robot Control 4 (KRC4) is not designed for additional modules. The communication between KRC4 and the external PC is performed at 250 Hz using the Kuka Robot Sensor Interface. The laser striper is a Micro-Epsilon ScanControl 2700-100 which obtains a stripe of 640 depth points in a range of 0.3 m to 0.6 m at 50 Hz with a maximum measuring error of approx. 0.5 mm. During laser scans, the robot pose and range data are synchronized. Note that the depth measurements of the laser striper are very accurate in contrast to RGB-D cameras [2].

(a) The test objects



(b) The template models

Fig. 2. The test objects with corresponding template models used for the experiments: bunny, wooden chevron, Zeus bust (from left to right). Colors describe the different feature classes: light/dark red occur in convex regions, blue/purple in concave regions (plane regions were removed).

### B. Test Objects and Data

All tests were performed with scans from three objects: a bunny, a wooden chevron, and a Zeus bust (see Fig. 2(a)). These represent different application domains, namely household, manufacturing and cultural heritage. The approximate height of the bunny and chevron is 18 cm, and of the bust 22 cm. Ground truth surface models were generated with a commercial 3D modeling system. Based on these models, feature points were calculated in a preprocessing step. Each feature point set was classified with 5 classes and the middle class was removed. The reduced feature point sets served as template models (see Fig. 2(b)) and were used for registration during the laser scans. The template models of the bunny, the chevron and the Zeus consisted of 6714, 13075 and 8771 feature points, respectively.

The method has been evaluated with 10 different scan paths of the Zeus bust, 8 scan paths of the bunny and 5 scan paths of the chevron. The different numbers of scan paths is a consequence of the differences in object shape and the chevron's symmetry. The scan paths were placed all around the objects, in order to ensure independence of the results. In order to achieve a meaningful number of test runs, we repeated each test 100 times. As it is not be feasible to repeat the whole scanning process so often, the tests were performed once on the real robot and the scan data was saved. Then, the tests were repeated on the saved data.

Concerning the quality of the acquired 3D models, the robot's pose error during a scan is usually negligible, However,

TABLE I
OVERVIEW OF USED PARAMETER VALUES

| $r_r$ | $r_n$ | $th_t$ | $th_R$ | $r_T/r_t$ | $n_P/n_p$ | $r_S/r_s$ |
|---|---|---|---|---|---|---|
| 1 mm | 5 mm | 5 mm | 3° | 10/1 mm | 100/10 | 50/8 mm |

significant differences of robot configurations between two different scans lead to considerable pose errors between the acquired 3D scan data. Typically, there are gaps up to $3\,\mathrm{mm}$ in between different scans. Thus, ground truth estimation was necessary for each scan, because the resulting pose estimation accuracy was in the range of millimeters.

Each ground truth estimation was calculated by utilizing the global method from [10], followed by an ICP working on all acquired raw points. A visual inspection by a human operator assured correct results. Additionally the coordinate root mean square error after the ICP was checked to be lower than $0.2\,\mathrm{mm}$ in every case.

### C. Parameters

The feature point calculation radii (as denoted in Sect. IV), the thresholds for the movement detection (Sect. III), the sampling radii (Sect. V) and particle numbers are summarized in Tab. I.

## VIII. RESULTS

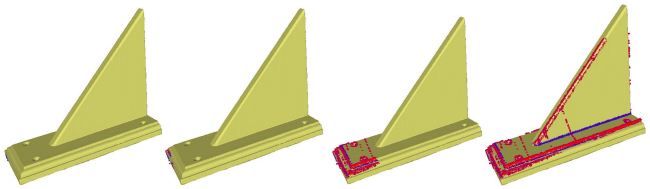
The comparisons in this section are done with respect to the median of rotational and translational error, denoted $m_t$ and $m_R$, respectively, and a success rate. A success is defined, if the final error in translation and rotation is below $8\,\mathrm{mm}$ and $8°$. The ratio between successful runs and total runs is called success rate and denoted **sr**. For the presented objects, the given bounds proved necessary for the ICP to converge reliably (differing from the bounds given in [10]).

### A. SPFR vs. SMCR

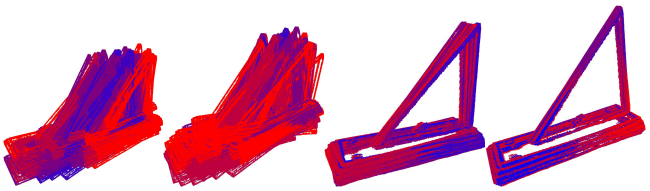
Fig. 3 depicts the typical convergence progress of SPFR and SMCR. Clearly, the convergence is better with SMCR, compared to SPFR. This impression gets confirmed by the convergence characteristics of $m_t$ and $m_R$, depicted in Fig. 4. Therefore, we used this generalized implementation in the remainder and also propose it for small scale objects. However, if too many feature points are used for weighting, the computation can slow down too much. In most cases this can be avoided by using a stricter particle reduction: if the particle number decreases sufficiently, the weighting process for all particles will not slow down too much.

### B. Comparisons with Offline Methods

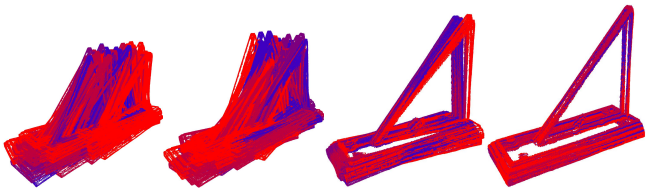
We compared our method to a Monte Carlo registration (MCR) [10], and to Sample Consensus Initial Alignment (SAC-IA) [9], the method that performed best in the comparisons therein. SAC-IA is a method based on sampling consensus, that is sampling the parameter space based on correspondences selected based on a similarity metric. In our previous work, we used the same features as for our method, but here we also tested FPFH as well, a multi-dimensional



(a) collected feature points after after 0, 1, 10 and 50 steps



(b) particle distribution after 0, 10, 20 and 48 steps for SPFR



(c) particle distribution at steps 0, 1, 20 and 46 steps for SMCR

Fig. 3. Particles in SPFR and SMCR. Top row: approximate scanning progress for the depicted particles and online fit to a surface model. Lower rows: the particle distributions, colored from blue to red, with blue being the best and red the worst particle.

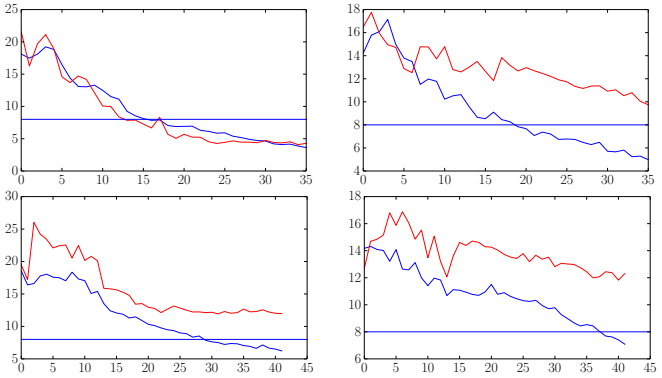


Fig. 4. Exemplary error convergence of SPFR (red) and SMCR (blue) for 100 runs on a bunny (top) and a Zeus (bottom) scan. Left: translational error in mm. Right: rotational error in degree. X-axis: step number. The straight blue line represents the success threshold of 8 degrees or 8 mm.

feature proposed in [7]. It is important to note that SAC-IA is not taking any prior information into consideration, so it needs to perform many trials, resulting in long computation times. As discussed in [10], introducing prior information is possible, but the rejection sampling approach tested there makes the method even slower. Again, we found that SAC-IA works well with 3000 iterations (results can be improved when using tens of thousands of trials, but we did not find that feasible). Tab. II shows the result for 100 runs with all scans. The $b_i, w_i, z_i$ denote the scans of the bunny, the wooden chevron and the Zeus bust, respectively. Unlike in [10], here we have high-quality scans, so the results for SAC-IA were best when using

TABLE II
100 RUNS: **SR**, $m_t$, $m_R$ AND THE MEAN COMPUTATION TIME $\bar{t}$ FOR SMCR, MCR AND SAC-IA.

| | SMCR | | MCR | | | SAC-IA | | |
|---|---|---|---|---|---|---|---|---|
| data | **sr** | $m_t/m_R$ | **sr** | $m_t/m_R$ | $\bar{t}$ | **sr** | $m_t/m_R$ | $\bar{t}$ |
| $b_1$ | 77 | 3.7 / 4.4 | 90 | 2.9 / 2.3 | 6.1 | 2 | 9.8 / 18.0 | 28.4 |
| $b_2$ | 71 | 3.9 / 4.0 | 57 | 2.2 / 7.4 | 7.4 | 6 | 8.2 / 13.0 | 30.3 |
| $b_3$ | 61 | 4.6 / 5.8 | 93 | 2.4 / 2.5 | 6.8 | 42 | 5.4 / 8.4 | 28.5 |
| $b_4$ | 44 | 3.9 / 9.2 | 78 | 2.7 / 6.1 | 4.3 | 22 | 6.8 / 11.9 | 23.0 |
| $b_5$ | 26 | 6.7 / 10.5 | 22 | 5.3 / 9.0 | 4.1 | 70 | 4.1 / 5.8 | 28.8 |
| $b_6$ | 15 | 13.9 / 10.1 | 58 | 4.6 / 4.9 | 4.4 | 20 | 7.5 / 10.4 | 27.7 |
| $b_7$ | 69 | 3.8 / 5.6 | 37 | 4.2 / 11.2 | 4.4 | 17 | 7.5 / 12.3 | 27.6 |
| $b_8$ | 43 | 6.0 / 8.5 | 45 | 4.6 / 8.7 | 4.7 | 2 | 11.0 / 28.0 | 23.4 |
| $w_1$ | 15 | 14.8 / 10.5 | 66 | 3.2 / 4.1 | 30.8 | 15 | 13.9 / 8.1 | 82.5 |
| $w_2$ | 33 | 11.8 / 7.6 | 96 | 1.4 / 1.7 | 23.7 | 14 | 13.3 / 9.7 | 77.0 |
| $w_3$ | 22 | 14.7 / 6.6 | 99 | 1.6 / 3.9 | 16.4 | 4 | 19.2 / 13.2 | 63.5 |
| $w_4$ | 17 | 18.7 / 8.6 | 52 | 7.5 / 4.3 | 15.9 | 1 | 39.7 / 170.0 | 59.2 |
| $w_5$ | 6 | 21.8 / 11.0 | 32 | 11.8 / 6.0 | 15.1 | 0 | 43.0 / 26.8 | 54.5 |
| $z_1$ | 51 | 5.7 / 5.8 | 60 | 5.9 / 6.0 | 8.6 | 4 | 11.3 / 15.6 | 120.4 |
| $z_2$ | 59 | 6.0 / 5.7 | 92 | 3.8 / 3.3 | 14.1 | 26 | 7.9 / 11.1 | 113.1 |
| $z_3$ | 49 | 7.1 / 7.8 | 58 | 4.5 / 6.9 | 14.4 | 24 | 9.1 / 9.2 | 109.0 |
| $z_4$ | 50 | 7.0 / 7.0 | 80 | 2.7 / 4.2 | 14.1 | 47 | 5.5 / 7.8 | 131.7 |
| $z_5$ | 25 | 9.4 / 13.5 | 22 | 17.8 / 52.4 | 10.4 | 92 | 2.5 / 3.4 | 125.8 |
| $z_6$ | 0 | 16.5 / 27.1 | 0 | 19.2 / 176.9 | 7.5 | 44 | 5.6 / 8.2 | 114.1 |
| $z_7$ | 34 | 9.3 / 9.5 | 0 | 20.6 / 169.5 | 7.6 | 58 | 5.2 / 6.0 | 114.8 |
| $z_8$ | 0 | 28.3 / 22.0 | 82 | 6.2 / 3.8 | 8.9 | 11 | 9.3 / 12.8 | 143.2 |
| $z_9$ | 41 | 7.6 / 7.6 | 69 | 4.1 / 4.8 | 10.3 | 5 | 21.5 / 24.8 | 80.4 |
| $z_{10}$ | 4 | 27.3 / 11.8 | 0 | 23.7 / 124.3 | 6.9 | 6 | 14.3 / 143.3 | 79.0 |
| units | % | mm/deg | % | mm/deg | s | % | mm/deg | s |

it on the complete point cloud (downsampled to a density of 3 mm) and the FPFH feature, so those are presented in the table. In Tab. II we highlight one of the cases where both SMCR and MCR perform poorly, but SAC-IA relatively well. Since the $z_6$ scan captures a smooth surface at the back of the statue's head, it contains relatively few feature points, resulting in larger errors. SAC-IA, however, uses all the points and manages to find a good transformation in $44\%$ of the cases. Nonetheless, looking at the distributions of the errors, we can see that while SAC-IA performed better, it failed completely in some of the cases, which did not influence the median too much. In such cases, more points should be considered by SMCR, in order to increase performance at the cost of computation time and introducing more uncertainty.

### C. Interpretation

The results in table Tab. II yield no definite best method. But they clearly show, that the method is competitive in accuracy and robustness with available state-of-the-art methods in many cases. The advantage of no extra computation time is clear, as even for these simple objects the offline methods need up to 2 minutes for getting similar results, whereas our method does not need any extra computation time. This effect will dramatically increase with larger objects, which could not be investigated with the hardware setup in this paper, due to kinematic constraints.

## IX. SUMMARY AND OUTLOOK

A novel streaming feature based pose estimation particle filter has been presented, comprising streaming feature calculation. Two variants for weighting have been presented and the convergence behaviors on real data investigated. An extensive comparison to existing offline state-of-the-art algorithms was performed. The proposed method proved to be comparable concerning accuracy and reliability. In contrast to offline methods, the estimation process is carried out in parallel to data acquisition and no time consuming post processing is required.

In future, scanning time can be saved, if the method reports convergence and data acquisition is stopped accordingly. Further, a priori knowledge can easily be autonomously gathered with depth cameras. In a forthcoming work [15] we apply the method in autonomous 3D modeling and equip it with an optimization in the weighting step. Finally, we want to apply the method in mobile robot localization and integrate it on humanoid or lightweight robots by utilizing more compact laser stripers.

## REFERENCES


[1] S. Kriegel, C. Rink, T. Bodenmüller, and M. Suppa, "Efficient next-best-scan planning for autonomous 3D surface reconstruction of unknown objects," *Journal of Real-Time Image Processing*, vol. 10, no. 4, pp. 611–631, 2015.

[2] S. Kriegel, M. Brucker, Z. Marton, T. Bodenmüller, and M. Suppa, "Combining object modeling and recognition for active scene exploration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Japan, Nov. 3–7, 2013, pp. 2384–2391.

[3] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[4] C. Chen, Y. Hung, and J. Cheng, "RANSAC-based DARCES: A new approach to fast automatic registration of partially overlapping range images," *IEEE Transactions PAMI*, vol. 21, pp. 1229–1234, 1999.

[5] S. Winkelbach, "Effiziente Methoden zum Lösen von 3D-Puzzle-Problemen," *it - Information Technology*, vol. 50, no. 3, pp. 199–201, 2008.

[6] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," in *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, San Francisco, CA, USA, Jun. 13–18, 2010, pp. 998–1005.

[7] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *IEEE International Conference on Robotics and Automation, ICRA*, Kobe, Japan, May 12–17, 2009.

[8] A. Aldoma, Z. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze, "Tutorial: Point cloud library – three-dimensional object recognition and 6 dof pose estimation," *Robotics & Automation Magazine*, vol. 19, no. 3, pp. 80–91, 2012.

[9] R. B. Rusu, N. Blodow, Z. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Acropolis Convention Center, Nice, France, Sep. 22–26, 2008.

[10] C. Rink, Z. Marton, D. Seth, T. Bodenmüller, and M. Suppa, "Feature based particle filter registration of 3D surface models and its application in robotics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Japan, Nov. 3–7, 2013, pp. 3187–3194.

[11] G. Barequet and M. Sharir, "Partial surface and volume matching in three dimensions," *IEEE Transactions PAMI*, vol. 19, pp. 929–948, 1994.

[12] J. Sturm, W. Burgard, and D. Cremers, "Evaluating egomotion and structure-from-motion approaches using the TUM RGB-D benchmark," in *Proceedings of the Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012.

[13] T. Bodenmüller, W. Sepp, M. Suppa, and G. Hirzinger, "Tackling multi-sensory 3D data acquisition and fusion," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. San Diego, USA: IEEE/RSJ, Oct./Nov. 2007, pp. 2180–2185.

[14] T. Bodenmüller, "Streaming surface reconstruction from real time 3D measurements," Dissertation, Technische Universität München, Munich, 2009.

[15] C. Rink and S. Kriegel, "Streaming Monte Carlo pose estimation for autonomous object modeling," in *13th Conference on Computer and Robot Vision (CRV)*. Victoria, Canada: IEEE, May 2016, accepted for publication.