REAL-TIME LOCALIZATION OF OBJECTS IN TIME-OF-FLIGHT DEPTH IMAGES

Ulrike Thomas
Institute for Robotics and Mechatronics, German Aerospace Center, 82234 Wessling, Germany

Keywords: Object Localization, Pose Estimation, Time-of-Flight Sensors, Ransac.

Abstract: In this paper a *Random Sample Consensus* (Ransac) based algorithm for object localization in time-of-flight depth images is presented. In contrast to many other approaches for pose estimation, the algorithm does not need an inertial guess of the object's pose, despite it is able to find objects in real time. This is achieved by hashing suitable object features in a pre-processing step. The approach is model based and only needs point clouds of objects, which can either be provided by a CAD systems or acquired from prior taken measurements. The implemented approach is not a simple Ransac approach, because the algorithm makes use of a more progressive sampling strategy, hence the here presented algorithm is rather a *Progressive Sampling Consensus* (Prosac) approach. As a consequence, the number of necessary iterations is reduced. The implementation has been evaluated with a couple of exemplary scenarios as they occur in real robotic applications. On the one hand, industrial parts are picked out of a bin and on the other hand every day objects are located on a table.

1 INTRODUCTION

Localization of known objects in images and acquired scenes is of importance for many robotic applications, e.g. in assembly or service domains, where robots have to interact autonomously with their environments. In this context, known objects need to be located within a few seconds with such an accuracy in order to enable robots to manipulate these objects. This problem is widely known as pose estimation in cluttered scenes, where objects might partially be occluded. A robust and fast pose estimation of known objects in such scenarios will support robotic technologies to be brought into human environments. Recently, time-of-flight sensors became very popular and are used for some applications, e.g. for pose estimation of the human body (Plagemann et al., 2010), where real-time requirements have to be met. In this paper, it is shown, how objects can be localized in real-time in depth images acquired from time-of-flight sensors. In the approach described in this paper, no assumptions of inertial object poses are required. The specific sensors applied for object localization in this paper are the Swissranger SR 4000 (Oggier et al., 2004) and the PMDs CamCube, providing intensity and depth images with a frame rate of up to 50 Hz with a resolution of 174×144 and of 204×204 pixels, respectively. The calibration of these sensorshas been described in (Fuchs and Hirzinger, 2008). Figure 1 illustrates intensity and depth images acquired from given scenes by the time-of-flight sensor. In this paper, only the depth images are used for pose estimation. In general the localization problem can be treated as a 6 dof pose estimation problem of a rigid object; hence, the task is to estimate a pose $P := (\mathbf{R}, \vec{t})$ with $\mathbf{R} \in SO(3)$ and $\vec{t} \in \mathbb{R}^3$ of a model in the given depth image. The depth image is here considered as a point cloud $\mathcal{P} = \{\vec{p}_1, \ldots, \vec{p}_n\}$ with its surface normals $\mathcal{N} = \{\vec{n}_1, \ldots, \vec{n}_n\}$ obtained from principle component analysis. The goal is to estimate poses of objects in real time by extracting as much as possible a-priori model knowledge. This paper presents a real-time object localization approach by using on the one hand hash tables of features obtained from models and on the other hand by choosing a suitable sampling strategy so that the *Random Sample Consensus* (Ransac) algorithm is adapted to a more *Progressive Sampling Consensus* (Prosac) approach. The content of this paper is structured as follows: The next section gives an overview about related work. The section next after illustrates how the a-priori knowledge of objects is extracted and processed, so that model knowledge can be used efficiently for real-time pose estimation. Section number four describes the main part of the algorithm in particular the sampling strategy and the hypotheses evaluation function. Section

Thomas U..
REAL-TIME LOCALIZATION OF OBJECTS IN TIME-OF-FLIGHT DEPTH IMAGES.
DOI: 10.5220/0003870707330737
In *Proceedings of the International Conference on Computer Vision Theory and Applications* (VISAPP-2012), pages 733-737
ISBN: 978-989-8565-03-7

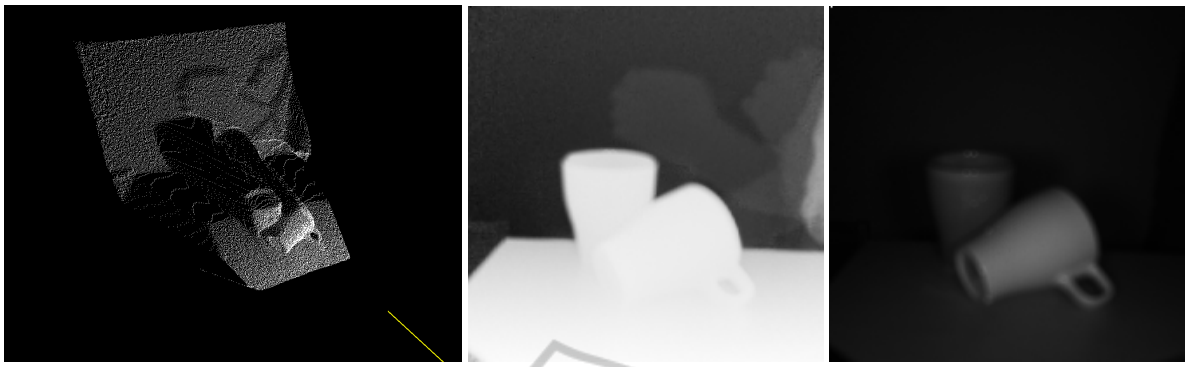

Figure 1: The algorithm is able to estimate object poses in uncluttered scenes. Illustrated a two cups arranged on a table. The left image shows a 3d point cloud obtained from the depth image. The image in the middle shows the depth image itself and on the right side the intensity image is depicted.

number five contains the evaluation part.

2 RELATED WORK

Ransac is well known for matching model data into sensor data (Fischler and Bolles, 1981; Bolles and Fischler, 1981). The original Ransac was illustrated for fitting lines into 2D images. Since then, Ransac has been applied to many applications. (Winkelbach et al., 2004) show that it is possible to solve the 3d puzzle problem using Ransac. As feature vector they use surflet pairs, which are used in this paper, too. (Buchholz et al., 2010) shows how the Ransac algorithm can be applied to solve the bin picking problem using laser scanner data. (Schnabel et al., 2007) apply Ransac for shape primitive detection in point clouds. In order to estimate object poses in stereo data obtained from image correlation (Hillenbrand, 2008) has provided an algorithm. For image registration and motion estimation (Nister, 2003) has applied the Ransac approach. In order to find certain models in uncluttered point cloud scenes (Papazov and Burschka, 2010) provide an algorithm, which uses some different sampling strategies, which can also be regarded as a Prosac approach. Prosac has been successfully used for the registration of images (Chum and Matas, 2005). Altogether, Ransac has become very popular for pose estimation. All these approaches have in common, that they are not real-time and mostly run for a few seconds on state-of-the-art hardware to estimate positions of objects in uncluttered scenes. In contrast to most of the above mentioned algorithms, a progressive sampling strategy is applied here, which takes the advantage of the model based a-priori knowledge.

3 FEATURE EXTRACTION FROM MODEL DATA

For generic object localization in real-time model information needs to be processed prior to execution. In the past, most approaches required images taken from the same sensor afflicted with same sensor noise. In contrast, here, only point cloud data of objects are applied as model data. These point clouds might be obtained from images taken prior to execution or from CAD-systems. Hence, model information is either available as triangle mesh – as a simplest representation in CAD-systems – or as point clouds. Hence, each object $O_1 \dots O_m$ can be modeled by a set of vertexes $O = \{\vec{v}_1, \dots \vec{v}_m\}$ with $\vec{v}_i \in \mathbb{R}^3$ and their corresponding surface normals. The surface normals are estimated by means of principle component analysis on at least five up to eight closest neighbors $N := \{\vec{n}_i, \dots, \vec{n}_m\}$ with $\vec{n}_i \in \mathbb{R}^3$. This kind of model generation is indeed very generic and depends not on a certain set of data. Figure 2 illustrates the original model data and the features drawn from theses models. Remember, the original Ransac algorithm takes two points $\vec{p}_i$, $\vec{p}_j$ from the input data set $\mathcal{P} = \{\vec{p}_1, \dots, \vec{p}_n\}$, and searches for the same two points $\vec{v}_i$, ,$\vec{v}_j$ in the model data set $\mathcal{V} = \{\vec{v}_1, \dots, \vec{v}_m\}$. Figure 2 illustrates a selected tuple $(\vec{p}_i, \vec{p}_j)$. Each point is assumed to be an oriented point with respective point normals $\vec{n}_i, \vec{n}_j$. It is shown how these feature pairs can be used to align the objects. For real-time object localization it is very important to access theses features in near $O(1)$ time complexity, scince the algorithm takes iteratively such features from the acquired data set and seeks for the corresponding features in the model set. Hence, a hash table suites well to provide efficient access to model features. The hash

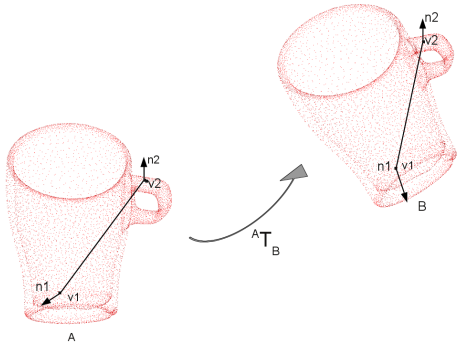

Figure 2: The model features, which are inserted into the hash map. For each pair of points the feature vector is determined and hashed, if the distance of points is not closer than a given threshold.

function must be as unambiguous as possible. Hence, given two vertexes $(\vec{v}_i, \vec{v}_j)$ with their vertex normals $(\vec{n}_i, \vec{n}_j)$ from the model data the feature vector $f \in \mathbb{R}^4$ is defined by:

$$f(v_i, v_j) := \begin{pmatrix} \| \vec{v}_i - \vec{v}_j \| \\ \angle(\vec{n}_i, \vec{v}_j - \vec{v}_i) \\ \angle(\vec{n}_j, \vec{v}_i - \vec{v}_j) \\ atan2(\frac{\vec{n}_i \cdot (\vec{d}_{ij} \times \vec{n}_j)}{(\vec{n}_i \times \vec{d}_{ij}) \cdot (\vec{d}_{ij} \times \vec{n}_j)}) \end{pmatrix} \quad (1)$$

with $\vec{d}_{ij} = \vec{v}_i - \vec{v}_j$. This feature vector has four dimensions. Hence a hash table of size N^4 is applied and N need to be adjusted to the available storage size. For this hash function the angular values between $[0, \ldots, 2\pi]$ are mapped to the range $[0, \ldots, N]$. For the translational parameter of f, the bounds $b_{\min}$ and $b_{\max}$ are chosen according to the model data.

$$\begin{aligned} b_{max} &= \max\{\| \vec{v}_i - \vec{v}_j \| \mid \forall \vec{v}_i, \vec{v}_j \in O \wedge i \neq j\} \\ b_{min} &= b_{max} \cdot 0.33 \end{aligned} \quad (2)$$

Therewith, the range of the hash table is adopted to values between the maximum possible translational distance of two point pairs and the minimum distance, which is bounded by one third of the maximum distance. It has been shown in experiments, that this choice is a reasonable bound. The explanation reads as follows: The smaller the range is for a feature the better are the matches of a point pair in the hash table and consequently the algorithm is able to find good matches much faster. Additionally to adjusting the hash map size to sizes of objects, the bounds are applied to enforce the algorithm to draw samples, which correspond more likely to the same object. Thereby, the random sampling consensus strategy turns more into a progressive sampling strategy, which is explained in more detail in the next section.

Note, that here for each object a separate hash table is used, which is much faster than using only one hash table, because the size of the hash table can be fitted better to the size of the objects. Due to the time consuming filling of the hash table, this is processed in an off-line phase.

4 RANSAC FOR OBJECT LOCALIZATION

Ransac is known very well for model fitting into measured data sets. In general the algorithm draws two samples randomly out of the data sets and seeks for the corresponding feature pair in the model data. Therewith, an object hypothesis is generated and evaluated. For the sampling strategy we have applied a progressive approach, for which we can show that the number of iterations is reduced to find certain objects in a scene. Additionally, an efficient evaluation function is implemented able to estimate the quality of hypotheses very accurately. Therewith our approach can be executed in real-time. In general, the steps of the algorithms can be described as follows:

1. For each object o_i a hash table has to be generated which is filled by feature vectors obtained for each tuple $(\vec{v}_i, \vec{v}_j)$ of oriented point pairs as mentioned above.
2. Start sampling by drawing the first point $\vec{p}_i$ from the measured data set $P = \{\vec{p}_i \ldots \vec{p}_n\}$ at random
3. Sampling the second point $\vec{p}_j$ by applying a ball of radius r with the first sample point as center point. The radius is assigned with the bounds of the object for which the pose should be estimated. Given n objects to be found in a scene, we approximate r as upper bound over all searched objects with $max_{b_{max}} \forall O$.
4. Regarding to the chosen pair of point $(\vec{p}_i, \vec{p}_j)$ determine the feature vector $f(\vec{p}_i, \vec{p}_j)$ and look it up in the hash table with the corresponding key. Let $\mathcal{H}$ be the set of hypotheses found in in the hash table for the chosen feature pairs $f(\vec{v}_i, \vec{v}_j)$, then determine for each hypothesis $h_i \in \mathcal{H}$ the aligned transformation for rigid motion $R \in SO(3)$ and $t \in \mathbb{R}^3$, see Figure 2
5. Evaluate the hypothesis. If it is a suitable hypothesis a confidence value is returned.
6. As long as no good hypotheses for the object is found and the time bound is not reached continue with step 2.

4.1 Progressive Sample Consensus

With the progressive sampling strategy it can be shown that the number of necessary iteration can be reduced. Given a data set of measured range image with N points, and assuming in the data set an object which is modeled by m points, the probability of finding the object is provided as follows. Note, that the number of inliers is set to m where the number of outliers is set to N. The probability p of finding two inliers in a single path is given approximately by the ratio of:

$$\left(\begin{array}{c} m \\ 2 \end{array} \right) / \left(\begin{array}{c} N \\ 2 \end{array} \right) \approx \left(\frac{m}{N}\right)^2 \tag{3}$$

The choice of points is assumed to be independent, because neglecting the points already drawn reduces the expensive deletion of points in hierarchical data sets. Now, let k be the number of iterations, where no two inliers have been chosen. Then we can assign the probability that Ransac has not converged against a solution after k iterations by:

$$1 - p(o,k) = (1 - \left(\frac{m}{N}\right)^2)^k \tag{4}$$

it follows that the number of iteration k necessary to find two inliers is given by the ratio:

$$k = \frac{\log(1 - p(o,k))}{\log(1 - \left(\frac{m}{N}\right)^2)} \tag{5}$$

Assuming the ratio between inliers and outliers is set to 10% and a probability of 50% is expected to fit the model data into the data set, the number of iterations necessary to be executed is given by 68. If we want to reduce the necessary iterations, we might draw the second point out of a ball of size r which contains R outliers with $R \ll N$. The probability of choosing a ball with only R outliers might be given by the ratio of (m/R) with 20% inliers at average. Then, the probability of drawing two inliers after k iterations can be estimated by:

$$p(o,k) = 1 - (1 - (\left(\frac{m}{N}\right) \cdot \left(\frac{m}{R}\right))^k) \tag{6}$$

and hence due to the ratio of 20% the number of iterations is reduced to 34, which is the half. But this holds only if R and respectively the ball size r is chosen in an appropriate way. It is realized by applying a ball search in the here used kd-tree data structure. The ball size is set to the upper bounds of the model data set.

4.2 Aligning Two Hypotheses

If a suitable hypothesis is found in the hash table, the model data has to be aligned to the drawn data. Given the feature vector found in the hash table $f(\vec{v}_i, \vec{v}_j)$ with its corresponding point normals $(\vec{n}_i^v, \vec{n}_j^v)$ and the feature vector obtained from drawing two samples $f(\vec{p}_i, \vec{p}_j)$ respectively with its normals $(\vec{n}_i^p, \vec{n}_j^p)$ the rigid body transformation can be determined in the following way. First we aligning the translational vector $\vec{t_\Delta}$ by:

$$\vec{t_\Delta} = \frac{1}{2}(\vec{v}_j + \vec{v}_i - \vec{p}_j + \vec{p}_i) \tag{7}$$

and for the rotational part, we align the normal vectors and obtain the rotation matrix $Rot \in \mathbb{R}^{3\times3}$

$$Rot^W_{Feat_B} := \left(\begin{array}{ccc} \frac{(\vec{v}_j - \vec{v}_i)\cdot(\vec{v}_j - \vec{v}_i)}{||(\vec{v}_j - \vec{v}_i)\times(\vec{v}_j - \vec{v}_i)||} & \frac{(\vec{x}\cdot\vec{n}_i^v)}{||\vec{x}\times\vec{n}_i^v||} & \frac{(\vec{x}\cdot\vec{y})}{||\vec{x}\times\vec{y}||} \end{array} \right) \tag{8}$$

and respectively the rotation matrix for the features taken from model data $Rot^W_{Feat_A}$. The obtained transformation applied to the model data $\vec{v}_i$ is then given by $Rot^W_{Feat_A} \cdot (Rot^W_{Feat_B})^{-1}\vec{v}_i + \vec{t_\Delta}$. Therewith we get hypotheses from drawing point pairs out of the measured data set which have to be evaluated.

4.3 Evaluating Hypotheses

The evaluation function has to be efficient and accurate, which means that no good hypotheses should be overlooked and that many hypothesis can be evaluated in a reasonable amount of time. For these objectives we define a model point $\vec{v}_i$ with its point normal $\vec{n}$ transformed to $\vec{v}_i'$ by the above given rigid transformation. A model point is defined to be in contact with a measured data point $\vec{p}_j$, if their distances are smaller then a given threshold δ and the normal vectors directs toward the camera coordinate system. The distance can be estimated very fast using hierarchical data structures like kd-trees. The hypothesis error is hence given by following equation:

$$\frac{1}{c^2_{contact}} \sqrt{\sum_{\forall\, \vec{v}_i' \in O} g(\vec{v}_i')} \tag{9}$$

The distance function g is denoted by:

$$g(v_i) := \begin{cases} d = \min_{\forall p_j}\{|| \, v_i - p_j \, ||\} & \text{if } d \le \delta \text{ and} \\ & \vec{v}_i \text{ visible} \\ \delta & \text{else} \end{cases} \tag{10}$$

With this evaluation function, those hypotheses are assigned with lower cost, whose number of contact points is higher.

5 EVALUATION

The evaluation of the algorithm has been carried out by using some different objects. For each object the

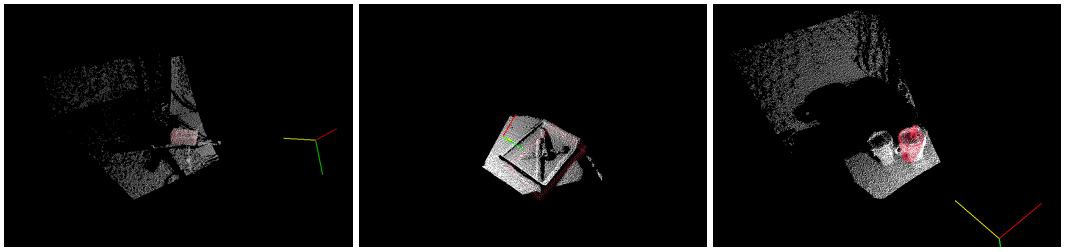

Figure 3: Results for object localization. Left: Finding a box modeled by 625 points has been located. Note, that many symmetries are in the depth image, which makes it much more difficult to find an accurate solution. Middle: For solving the bin picking problem; the bin has been located. Right: Two cups are arranged; the better solution is chosen as it can be seen.

appropriate hash table is filled as described in section three. For evaluating the pose estimation approach, the algorithm has been repeated 20 times and the mean execution time is depicted in Table 1. Additionally, to the original Ransac approach the Prosac based sampling strategy has been applied. The Figure 3 illustrates the results. It is shown that object poses can be estimated very fast with a reasonable position and rotation error. The output generated by our approach should be used as an inertial solution for either the ICP (iterative closed point) algorithm or the Kalman filter, if real-time requirement must be met. The computation times of both Ransac and Prosac are listed and it can be shown that the Prosac approach converges much faster against a reasonable solution.

Table 1: Evaluation time for scenarios illustrated in Figure 3. The computation has been carried out on an Intel Xeon 2,7 GHz CPU with 3 GB Memory.

Scenarios	Ransac	Prosac
example 1	82 ms	49 ms
example 2	748 ms	498 ms
example 3	831ms	501 ms

6 CONCLUSIONS

For object localization a Ransac algorithm has been implemented and is compared to a Prosac approach. With the Prosac approach the number of iterations to find two inliers is reduced. Thereby an algorithm has been provided which can be applied in real-time for pose estimation. We suggest to use it as an inertial pose guess for a particle filter or a Kalman filter. The approach can be adopted to any depth images or point clouds. The presented method will be applied to uncluttered scenes represented as point clouds as well as depth images obtained from stereo vision. Further on it is planned to compare this approach with the fast point feature histograms registration method described in (Rusu et al., 2009).

REFERENCES

Bolles, R. and Fischler, M. A. (1981). A ransac-based approach to model fitting and its application to finding cylinders in range data. In *Proc. IJCAI*, pages 637–643.

Buchholz, D., Winkelbach, S., and Wahl, F. M. (2010). Ransam for industrial bin-picking. In *ISR/Robotics 2010*, pages 1317–1322.

Chum, O. and Matas, J. (2005). Matching with prosac - progressive sampling consensus. In *Proc. Conf. Computer Vision and Pattern Recognition*.

Fischler, M. and Bolles, R. (1981). Random sample consensus: A paradigm for model fitting with applications to image analy- sis and automated cartography. *CACM*, 24(6):381–395.

Fuchs, S. and Hirzinger, G. (2008). Extrinsic and depth calibration of tof-cameras. In *Proc. Conf. Computer Vision and Pattern Recognition*.

Hillenbrand, U. (2008). Pose clustering from stereo data. In *VISAPP International Workshop on Robotic Perception*, pages 23–32.

Nister, D. (2003). Preemptive ransac for live structure and motion estimation. In *ICCV*, pages 199–206.

Oggier, T., Lehmann, M., and R. Kaufmann, M. Schweizer, M. R. (2004). An all-solid-state optical range camera for 3d real-time imaging with sub-centimeter depth resolution(swissranger). In *Proc. SPIE*, volume 5249, pages 534–545.

Papazov, C. and Burschka, D. (2010). An efficient ransac for 3d object recognition in noisy and occluded scenes. In *ICCV*.

Plagemann, C., Ganapathi, V., Koller, D., and Thrun, S. (2010). Real-time identification and localization of body parts from depth image. In *Proc. International Conference on Robotics and Automation*.

Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast point feature histograms (fpfh) for 3d registration. In *ICRA*.

Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient ransac for point-cloud shape detection. In *Eurographics*, pages 1–12.

Winkelbach, S., Rilk, M., Schoenfelder, C., and Wahl, F. (2004). Fast random sample matching of 3d fragments. In *PatternRecognition (DAGM 2004), Lecture Notes in Computer Science 3175*, pages 129–136.