PFBIK-Tracking: Particle Filter with Bio-Inspired Keypoints Tracking

Sílvio Filipe
IT - Instituto de Telecomunicações
Department of Computer Science
University of Beira Interior
6200-001 Covilhã, Portugal
Email: sfilipe[at]ubi.pt

Luís A. Alexandre
IT - Instituto de Telecomunicações
Department of Computer Science
University of Beira Interior
6200-001 Covilhã, Portugal
Email: lfbaa[at]di.ubi.pt

***Abstract*—In this paper, we propose a robust detection and tracking method for 3D objects by using keypoint information in a particle filter. Our method consists of three distinct steps: Segmentation, Tracking Initialization and Tracking. The segmentation is made in order to remove all the background information, in order to reduce the number of points for further processing. In the initialization, we use a keypoint detector with biological inspiration. The information of the object that we want to follow is given by the extracted keypoints. The particle filter does the tracking of the keypoints, so with that we can predict where the keypoints will be in the next frame. In a recognition system, one of the problems is the computational cost of keypoint detectors with this we intend to solve this problem. The experiments with PFBIK-Tracking method are done indoors in an office/home environment, where personal robots are expected to operate. The Tracking Error evaluate the stability of the general tracking method. We also quantitatively evaluate this method using a "Tracking Error". Our evaluation is done by the computation of the keypoint and particle centroid. Comparing our system with the tracking method which exists in the Point Cloud Library, we archive better results, with a much smaller number of points and computational time. Our method is faster and more robust to occlusion when compared to the OpenniTracker.**

I. Introduction

Tracking is the process of following moving objects over time using a camera. There is a vast range of applications for tracking, such as, vehicle collision warning and avoidance, mobile robotics, speaker localization, people and animal tracking, tracking a military target and medical imaging. To perform tracking an algorithm analyzes sequential video frames and outputs the location of targets on each frame.

There are two major components of a visual tracking system: target representation and filtering. Target representation is mostly a bottom-up process, whereas filtering is mostly a top-down process. These methods give a variety of tools for identifying the moving object. The following are some common target representation algorithms: Blob tracking, Kernel-based or mean-shift tracking and contour tracking. Filtering involves incorporating prior information about the scene or object, dealing with object dynamics, and evaluation of different hypotheses. These methods allow the tracking of complex objects along with more complex object interaction like tracking objects moving behind obstructions [1]. The following are some common filtering algorithms: Particle filter and Kalman filter.

The interest on using depth information on computer vision applications has been growing recently due to the decreasing prices of 3D cameras like Kinect and Asus Xtion. Depth information facilitates object perception, as it allows for the determination of its shape or geometry.

In this work, we will use directly the information given by a Kinect camera and the Point Cloud Library (PCL) [2]. With this camera, we do not spend time to produce the depth map, since this is given by the camera. In traditional stereo vision, two cameras, placed horizontally from one another are used to obtain two differing views on a scene, in a manner similar to human binocular vision. PCL contains many algorithms that deal with point cloud data, from segmentation to recognition, from search to input/output [3].

In figure 1, we present the setup/pipeline of our recognition system and we will focus on the Particle Filter with Bio-Inspired Keypoints Tracking (PFBIK-Tracking) block. The PFBIK-Tracking block is composed by three main steps: Segmentation, Tracking Initialization and Tracking. The input cloud is segmented using two steps: Pass Through Filter (PTF) and Planar Segmentation (PS). These steps are performed in order to remove the background and keep only the object information. This is used to initialize the tracking, where the Cluster Extraction (CE) and the keypoint extraction are performed. This step (Tracking Initialization) is done only in the first iteration in order to select one cluster and extract the points interest of the selected cluster. In [4], they use a similar tracking method, but it's the user who makes this initialization manually. The Tracking step uses a particle filter in order to follow the keypoints, which represent the object. In [5], we have made an evaluation of keypoint detectors, and concluded that Scale Invariant Feature Transform 3D (SIFT3D) [6] and Intrinsic Shape Signatures (ISS) [7] are the detectors with the best results regarding the rotation, translation and change scale. We use the SIFT3D keypoint detector because of its biological properties mentioned [8].

The Recognition block is presented in this work only in a illustrative way and we will not discuss in this work. But in [3], [9], the author give us a good perspective on how to solve the issue of objects recognition. In [3], the focus is on the descriptors available in PCL (Feature Extraction step). He briefly explain how they work and made a comparative evaluation on publicly available data. It compares descriptors based

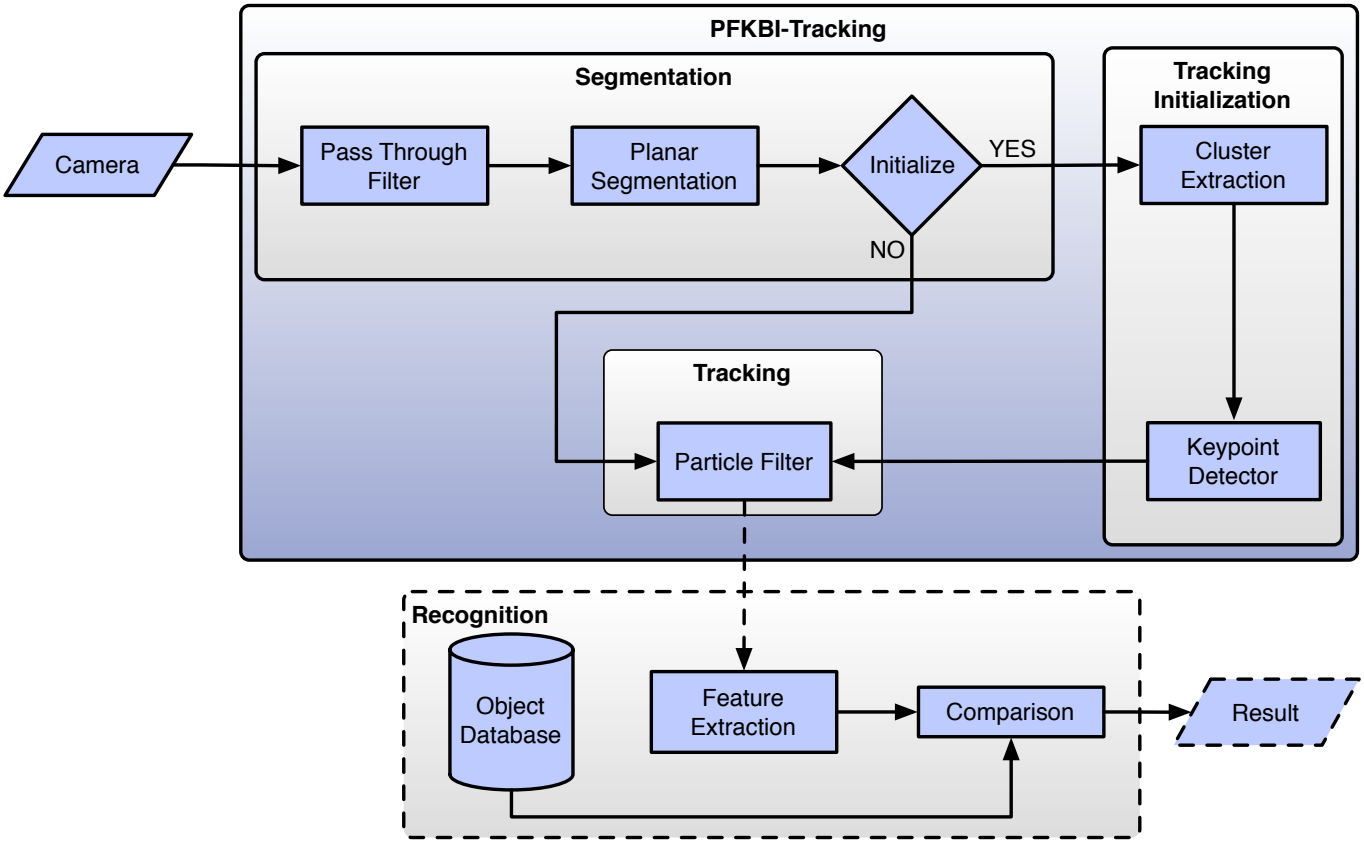


Figure 1. Setup of our recognition system. The diagram presents a complete object recognition system in order to understand better how the communication between the different stages is processed.

on two methods for keypoint extraction: one is a keypoint detector and the second approach consists on sub-sampling the input cloud with two different sizes. One conclusion in this work is that the increased number of keypoints improves recognition results at the expense of size and time. The same author studies the accuracy of the distances both for objects and category recognition and finds that simple distances give competitive results. Our work will use the distance measure with the best accuracy presented in [9].

The paper is organized as follows: the next section presents an overview on keypoint detectors and particle filters; in section III, we describe our Particle Filter with Keypoints Tracking method; and the last two sections will discuss the results obtained and present the conclusions.

II. Overview

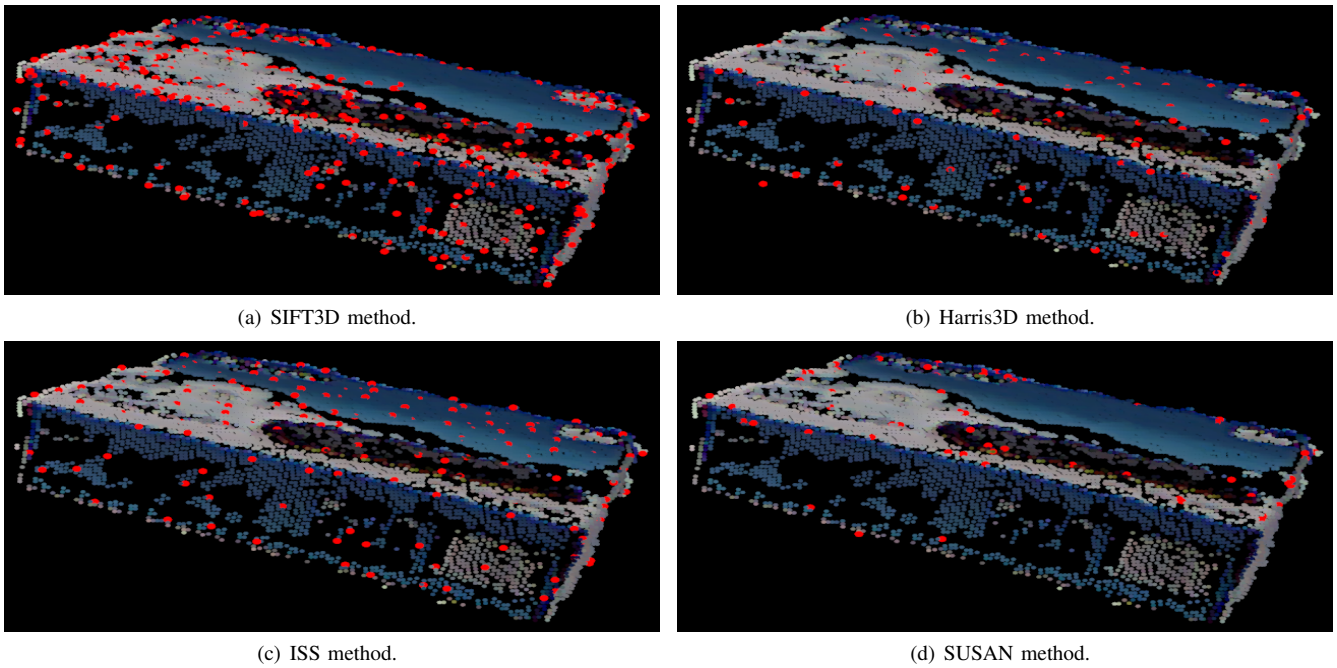
The image analysis methods used in this work can be divided into two areas: the spatial analysis (keypoint detectors) and temporal analysis (particle filters). The spatial analysis are points that stand out from the objects according to certain criteria result. Their coordinates are estimated in every frame of the image sequence, if a temporal analysis is not performed. Temporal analysis detects particles that are connected from frame to frame using another set of criteria to form tracks, removing the need to extract spatial information in all frames.

A. Keypoint Detectors

Keypoint detectors are used to find interest points in 3D point clouds. Interest points have been long used in the context of motion, stereo, and tracking problems. A desirable quality of an interest point is its invariance to changes in illumination and camera viewpoint. In PCL, commonly used interest point detectors include Harris keypoint detector [10], [11] (and its variants Lowe [8] and Noble [12]), Kanade-Lucas-Tomasi (KLT) detector [13], SIFT3D detector [8], [6], Smallest Univalue Segment Assimilating Nucleus (SUSAN) [14] and ISS [7]. For a comparative evaluation of 2D keypoint detectors, we refer the reader [15], [16] and for 3D keypoint detectors to [17], [5]. In [5], we provide a detailed description of the keypoint detectors present int the PCL and focus on the invariance evaluation of the detectors. Example of the most important 3D keypoint detectors applied in a point cloud are shown in figure 2.

B. Particle Filters

Particle filters can be described as a sample-based implementation of a Bayes filter. The posterior probability density over the state space of a dynamic system is estimated by the particle filters. The main idea of the particle filter is to represent the probability density by sets of samples, where each particle has a weight assigned to it that represents the probability of that particle being sampled from the probability



(a) SIFT3D method. (b) Harris3D method.

(c) ISS method. (d) SUSAN method.

Figure 2. Example of keypoints extraction in 3D point clouds using different methods. The keypoints are the red dots.

density function. This representation allows the particle filters to combine the efficiency with the ability to represent a wide range of probability densities [18]. Particle filter methods can be divided into four sub-groups: **Multi-hypothesis methods** represent the state by mixtures of Gaussians. These approaches are able to solve the global localization problem, since each hypothesis is tracked using a Kalman filter; **Topological methods** are based on symbolic, graph structured representations of the environment. The advantage of these approaches lies in their efficiency and in the fact that they can represent arbitrary distributions over the discrete state space; **Grid-based methods** rely on discrete, piece-wise constant representations. Topological and in particular grid-based implementations of Bayes filters for robot localization are often referred to as Markov localization [18]; **Sample-based methods** are represented by sets of samples. Since they focus their resources on regions with high likelihood. The efficiency of particle filters strongly depends on the number of samples used for the filtering process, several attempts have been made to make more efficient use of the available samples [19], [20].

The time complexity of one update of a particle filter algorithm is linear in the number of samples needed for the estimation [18]. Several authors have tried to make a more efficient use of the available samples, allowing sample sets of reasonable size. In [20], they incorporate Markov chain Monte Carlo steps to improve the quality of the sample-based posterior approximation [20]. An auxiliary particle filter is presented in [19]. Initially they applied an one-step lookahead to minimize the mismatch between the proposal and the target distribution. This minimizes the variability of the importance weights, which in turn determines the efficiency of the importance sampler.

III. Particle Filter with Bio-Inspired Keypoints Tracking

In this section, we will focus on the PFBIK-Tracking block presented in the figure 1. Our method is composed by two main steps: Segmentation and Tracking, which we will now describe in detail.

A. Segmentation

The segmentation starts with the PTF. This filter removes depth regions that are not contained on the desired working distances $[d_{min}, d_{max}]$, where d_{min} is the minimum distance at which the system should work and d_{max} the maximum distance. Depth regions that are not included between these distances are considered background and are discarded by our tracking system. By removing these regions (shown in figure 3(b)), which do not have interesting information for the object tracking system, a considerable reduction in the time is obtained.

The second step of the segmentation is the PS, which is based on the Random Sample Consensus (RANSAC) algorithm [21]. It is an iterative method to estimate parameters of a mathematical model from a set of observed data which contains outliers. Basically, the data consists of "inliers" and "outliers". The distribuition of the inliers data can be explained by some set of model parameters, but may be subject to noise and outliers, which are data that do not fit the model. The outliers can come from extreme values of the noise or from erroneous measurements or incorrect hypotheses about the interpretation of data.

Let w be the probability of choosing an inlier each time a single point is selected, that is, $w = \frac{\#inliers}{\#points}$. Using n points, selected independently, for estimating the model, w^n is the probability that all n points are inliers and $1 - w^n$ is the

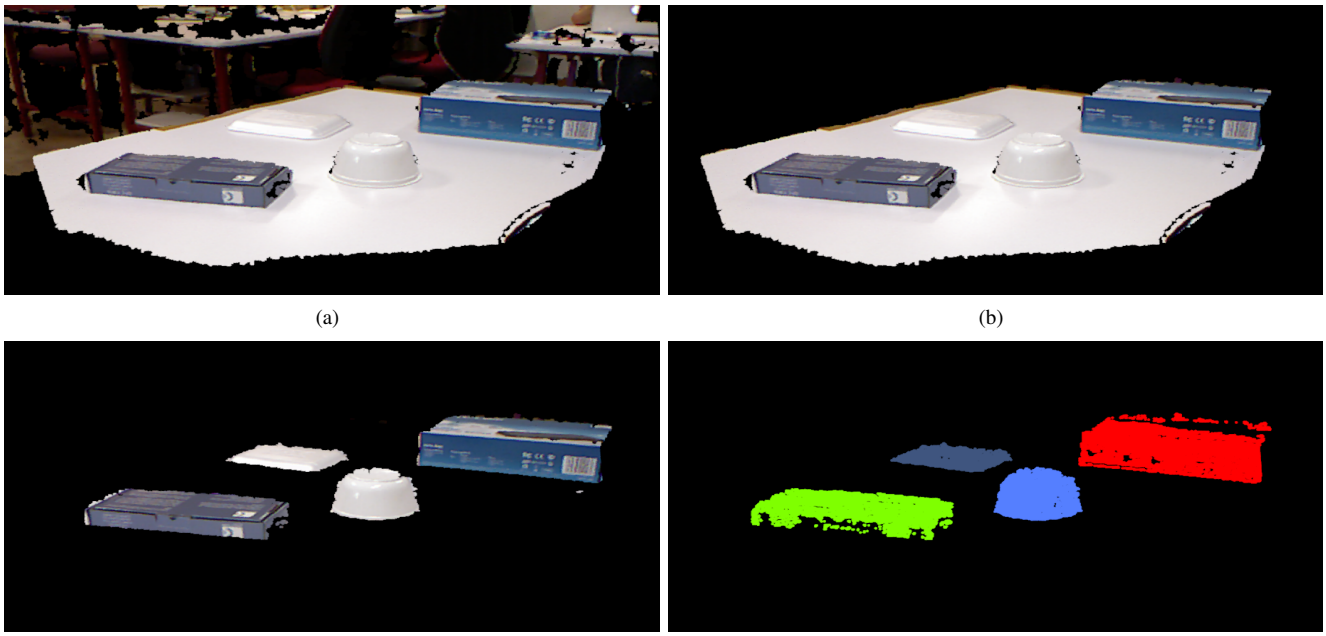


Figure 3. Representation of the segmentation steps. Figure (a) represents a cloud captured by our camera. Figure (b) is the output of the pass through filter with $d_{min} = 0.0$ m and $d_{max} = 1.6$ m, and in (c) the result of the removal of planar regions. Figure (d) are the clusters of the objects, wherein each object is represented by a different color.

probability that at least one of the n points is an outlier. That probability to the power of k (number of iterations) is the probability that the algorithm never selects a set of n points which all are inliers and this must be the same as $1-p$. Where p is the probability that the RANSAC algorithm in some iteration selects only inliers from the input cloud set when it chooses the n points from which the model parameters are estimated. Consequently,

$$1 - p = (1 - w^n)^k \tag{1}$$

which, after taking the logarithm of both sides, leads to

$$k = \frac{\log(1-p)}{\log(1-w^n)}. \tag{2}$$

Given the planar region estimated by RANSAC algorithm, we can remove the planar regions from our cloud, keeping only the remaining objects (shown in figure 3(c)).

B. Tracking Initialization

In the first frame captured, to initialize the tracking, we perform the third step of segmentation, the CE. Clustering is the process of examining a collection of "points", and grouping the points into "clusters" according to some distance measure. That is, the goal is that points in the same cluster have a small distance from one another, while points in different clusters are at a large distance from one another. This step will return a list of the clusters (shown in figure 3(d)), where each one contains the information of an object present in the cloud scene. In this work, we use a Euclidean Clustering method. As the name implies, the distance between two points p_1, p_2 is given by the Euclidean distance:

$$D(p_1, p_2) = \sqrt{(p_{1_x} - p_{2_x})^2 + (p_{1_y} - p_{2_y})^2 + (p_{1_z} - p_{2_z})^2}. \tag{3}$$

This PCL implementation is explained in [22].

As we mentioned earlier, in [5], we presented an evaluation of the keypoint detectors available on PCL. The Scale Invariant Feature Transform (SIFT) keypoint detector was proposed in [8]. The SIFT features are vectors that represent local cloud measurements.

The 3D implementation of SIFT's keypoint detector was presented in [6]. It uses a 3D version of the Hessian to select such interest points. A density function $f(x, y, z)$ is approximated by sampling the data regularly in space. A scale space is built over the density function, and a search is made for local maxima of the Hessian determinant. The input cloud $I(x, y, z)$ is convolved with a number of Gaussian filters whose standard deviations $\{\sigma_1, \sigma_2, \dots\}$ differ by a fixed scale factor. That is, $\sigma_{j+1} = k\sigma_j$ where k is a constant scalar that should be set to $\sqrt{2}$. The convolutions yield smoothed images, denoted by

$$G(x, y, z, \sigma_j), i = 1, \dots, n. \tag{4}$$

The adjacent smoothed images are then subtracted to yield a small number (3 or 4) of Difference-of-Gaussian (DoG) clouds, by

$$D(x, y, z, \sigma_j) = G(x, y, z, \sigma_{j+1}) - G(x, y, z, \sigma_j). \tag{5}$$

These two steps are repeated, yielding a number of DoG clouds over the scale space. Once DoG clouds have been obtained, keypoints are identified as local minima/maxima of the DoG clouds across scales. This is done by comparing each point in the DoG clouds to its eight neighbors at the same scale and nine corresponding neighborhood points in each of the neighborhood scales. If the point value is the maximum or minimum among all compared points, it is selected as a candidate keypoint.

The performance of human vision is obviously far superior to that of current computer vision systems, so there is potentially much to be gained by emulating biological processes. Fortunately, there have been dramatic improvements within the past few years in understanding how object recognition is accomplished in animals and humans [8]. Some features found in inferior temporal cortex (IT) are composed by neurons that respond to various moderately complex object features, and those that cluster in a columnar region that runs perpendicular to the cortical surface respond to similar features [23]. These neurons maintain highly specific responses to shape features that appear anywhere within a large portion of the visual field and over a several octave range of scales [24]. The complexity of many of these features appears to be roughly the same as for the SIFT. The DoG clouds are also similar to the "Place cells", which are pyramidal cells in the hippocampus which exhibit strongly increased firing in specific spatial locations [25]. The feature responses have been shown to depend on previous visual learning from exposure to specific objects containing these features [26]. These features appear to be derived in the brain by a highly computation-intensive parallel process, which is quite different from the staged filtering given by this method. A retinotopic organization, parallel processing, feedforward, feedback and lateral connection are a complex composition of the human visual system [27]. However, the results are much the same: an image is transformed into a large set of local features that each match a small fraction of potential objects yet are largely invariant to common viewing transformations.

C. Tracking

The Tracking block presented in figure 1 is the Particle Filter. For this we use an adaptive particle filter presented in [28], [18]. They presented a statistical approach to adapting the sample set size of particle filters on-the-fly. The number of the particles changes adaptively based on Kullback-Leibler Distance (KLD) sampling [29], where they bind the error introduced by sample-based representation of the particle filter. The samples are generated iteratively until their number is large enough to ensure that the KLD between the maximum likelihood estimate and the underlying posterior does not exceed a pre-specified bound. This method will choose different numbers of samples depending on whether the density of the 3D point cloud. If selects a small number of samples, the density is focused in a small subspace and it selects a larger number of samples, the samples have to cover most of the state space.

IV. Results

To evaluate the performance of our method, we will use the Euclidean distance (equation 3) between the centroid of the keypoints and result of our method, which is our "Tracking Error". The purpose of performing this comparison is that we intend that our system can track the keypoints of an object. This is done in order to not be necessary to apply a keypoint detector in all frames. In a real-time system is not feasible to apply a keypoint detector in each frame, due to the computational time spent on their calculation.

The centroid of a finite set of p points $\mathrm{p}_1, \mathrm{p}_2, \ldots, \mathrm{p}_k$ in $\mathbb{R}^n$ is given by

$$C = \frac{\sum_{i=1}^{k} \mathrm{p}_i}{k} \tag{6}$$

This point minimizes the sum of squared Euclidean distances between itself and each point in the set.

In order to properly evaluate the performance of our method, we will compare it with the sample-based method *OpenniTracker* available in PCL 1.7 (from the trunk). We apply our segmentation step in this tracker, where the output of this step is shown in figure 4. Thus, we will give as input exactly the same data to both methods. The difference between the two methods is the initialization of the particle filter. Whereas we initialize with the results of the keypoint detector, the OpenniTracker only makes a sub-sampling. This is a very important difference in the object recognition frameworks, because the sub-sampling only reduces the number of points in a linear manner, while the keypoint detector is reducing the number of points based on the object characteristics.

The results presented in table I, II and III are obtained using a dataset collected by us (shown in figure 4). This dataset contains 10 different moving objects in a total of 3300 point clouds. We will provide the dataset at: http://socia-lab.di.ubi.pt/~silvio.

In table I, we can observe that our method performed the tracking with a significantly lower number of points. Since our goal is to make the recognition of each object in the scene, our method has a stable performance with fewer points. With this number of points it is possible to think of making recognition in real time. On the other hand, with the number of points shown by the OpenniTracker this is very difficult to archive.

In table II, we present the distance between the cloud of keypoints ('what we expect') and the resulting cloud of points produced by the tracker ('what we estimate'). As already mentioned, the Euclidean distance is calculated based on the centroid of what was estimated and what we really are looking. In this table, we can see that even with a large decline in the number of points (around 50 times), our method has better performance than OpenniTracker.

In table III, we present the mean processing time of the two evaluated methods. These times were obtained on a computer with *Intel®Core™2 Quad Processor Q9300 2.5GHz* with $4\ GB$ of RAM memory. Our method takes longer to initialize

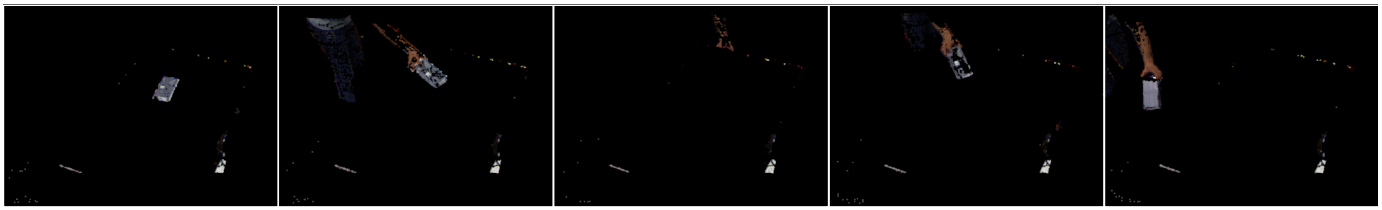
Table I. Mean and standard deviation number of keypoints and particles resulting from the tracker. In the OpenniTracker case, the column keypoints represents the sub-sampled cloud.

	Number of Points	
	Keypoints	Particles
PFBIK-Tracking	116.973 ± 76.251	102.107 ± 92.102
OpenniTracker	2502.536 ± 1325.807	2132.124 ± 1987.516

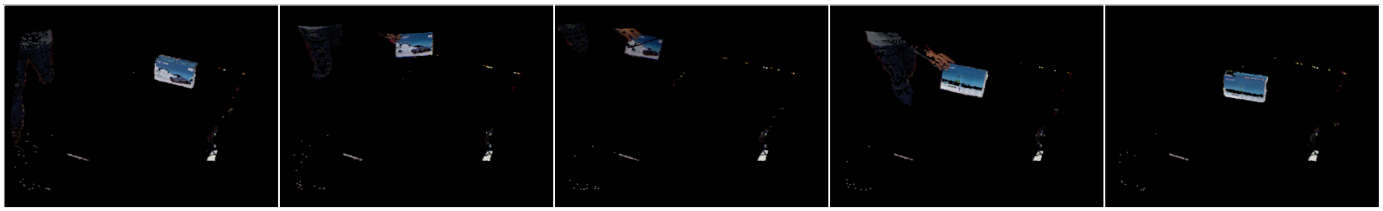
Table II. Euclidean distance between the output of the tracker and the expected result.

	Distance between Centroids			
	X axis	Y axis	Z axis	All axis
PFBIK-Tracking	0.036 ± 0.032	0.013 ± 0.012	0.019 ± 0.019	0.045 ± 0.036
OpenniTracker	0.038 ± 0.023	0.013 ± 0.011	0.027 ± 0.015	0.052 ± 0.022

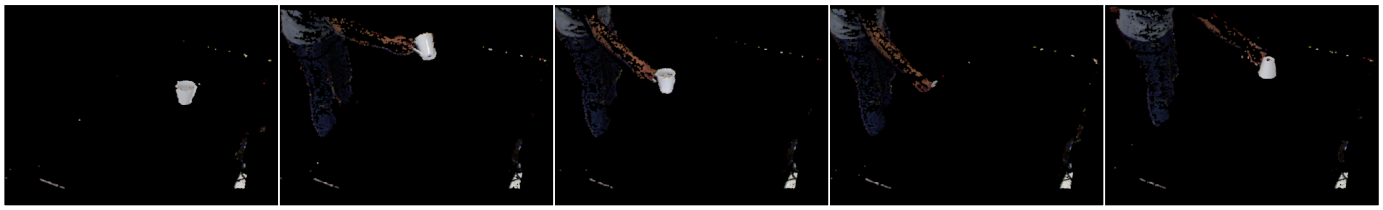
(a) Bowl.



(b) Box_1.



(c) Box_2



(d) Mug.

(e) Plate.

Figure 4. Segmented point cloud sequences of our dataset. These point clouds are the inputs of the presented tracking methods, and these have already been segmented.

Table III. MEAN AND STANDARD DEVIATION OF THE COMPUTATIONAL TIME (IN SECONDS) OF THE EVALUATED METHODS. HERE WE DISCARD THE TIME OF THE SEGMENTATION STEP, BECAUSE IT IS THE SAME IN BOTH METHODS.

	Tracking Initialization	Tracking
PFBIK-Tracking	0.203 $\pm$ 0.162	0.081 $\pm$ 0.057
OpenniTracker	0.173 $\pm$ 0.187	0.186 $\pm$ 0.170

the tracking, but then our tracking becomes 2.3 times faster than the other method presented. The initialization time is not a problem since this is only done once. The initialization is slower due to the fact that we extract keypoints, instead of the sub-sampling process used by the other method. In summary, our method obtains better results in terms of tracking (increased robustness to occlusion), while using less points and resulting in an improvement in terms of processing speed.

V. CONCLUSION

In this work, we present a system to perform the tracking of keypoints. The goal is to remove the necessity of applying a keypoint detector in all frames that we want to analyze. When we do this kind of approach, we would spend a lot computational time and the system could no longer function in

real time. We intend to make the tracking of keypoints because our main goal is to extract the descriptors of a particular object in the scene in order to perform the recognition. In order to do this we present several segmentation steps, so that we can remove all the background and objects become isolated. When we have the objects segmented, we apply a clustering method and the SIFT3D keypoint detector, which is used to initialize the particle filter. We use the SIFT3D keypoint detector because it has similar features to those in IT [8]. Once it is initialized with the intended object, we only need to give as input the output of the segmentation.

With our approach we were able to obtain better results than using the OpenniTracker: we have a faster and more robust method. For future work, we will intend to do the tracking of multiple objects simultaneously.

ACKNOWLEDGMENT

This work is supported by '*FCT - Fundação para a Ciência e Tecnologia*' (Portugal) through the research grant 'SFRH/BD/72575/2010', and the funding from 'FEDER - QREN - Type 4.1 - *Formação Avançada*', subsidized by the European Social Fund and by Portuguese funds through 'MCTES'. We also acknowledge the support given by the *IT - Instituto de Telecomunicações* through 'PEst-OE/EEI/LA0008/2013'.

REFERENCES

[1] J. Black, T. Ellis, and P. Rosin, "A novel method for video tracking performance evaluation," in *In Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2003, pp. 125–132.

[2] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *International Conference on Robotics and Automation*, Shanghai, China, May 2011.

[3] L. A. Alexandre, "3D descriptors for object and category recognition: a comparative evaluation," in *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, Portugal, Oct 2012.

[4] X. Li, Z. Cao, R. Yan, and T. Li, "Forward-looking infrared 3d target tracking via combination of particle filter and sift," 2013.

[5] S. Filipe and L. A. Alexandre, "A Comparative Evaluation of 3D Keypoint Detectors in a RGB-D Object Dataset," in *9th International Conference on Computer Vision Theory and Applications*, Lisbon, Portugal, Jan 2014.

[6] A. Flint, A. Dick, and A. Hengel, "Thrift: Local 3D Structure Recognition," in *9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications*, Dec. 2007, pp. 182–188.

[7] Y. Zhong, "Intrinsic shape signatures: A shape descriptor for 3D object recognition," *International Conference on Computer Vision Workshops*, pp. 689–696, Sep. 2009.

[8] D. Lowe, "Local feature view clustering for 3D object recognition," *Computer Vision and Pattern Recognition*, vol. 1, pp. I–682–I–688, 2001.

[9] L. A. Alexandre, "Set distance functions for 3D object recognition," in *18th Iberoamerican Congress on Pattern Recognition*, ser. LNCS, vol. 8258. Havana, Cuba: Springer, Nov 2013, pp. 57–64.

[10] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey Vision Conference*, Manchester, 1988, pp. 147–152.

[11] I. Sipiran and B. Bustos, "Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes," *The Visual Computer*, vol. 27, no. 11, pp. 963–976, Jul. 2011.

[12] J. Noble, *Descriptions of Image Surfaces*. University of Oxford, 1989.

[13] C. Tomasi and T. Kanade, "Detection and Tracking of Point Features," Carnegie Mellon University, Tech. Rep., 1991.

[14] S. M. Smith and J. M. Brady, "SUSAN – A new approach to low level image processing," *International Journal of Computer Vision*, vol. 23, no. 1, pp. 45–78, 1997.

[15] C. Schmid, R. Mohr, and C. Bauckhage, "Evaluation of Interest Point Detectors," *International Journal of Computer Vision*, vol. 37, no. 2, pp. 151–172, 2000.

[16] K. Mikolajczyk and C. Schmid, "Performance evaluation of local descriptors." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, Oct. 2005.

[17] S. Salti, F. Tombari, and L. D. Stefano, "A Performance Evaluation of 3D Keypoint Detectors," in *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, 2011, pp. 236–243.

[18] D. Fox, "Adapting the Sample Size in Particle Filters Through KLD-Sampling," *The International Journal of Robotics Research*, vol. 22, no. 12, pp. 985–1003, Dec. 2003.

[19] M. K. Pitt and N. Shephard, "Filtering via Simulation: Auxiliary Particle Filters," *Journal of the American Statistical Association*, vol. 94, no. 446, pp. 590–599, 1999.

[20] W. R. Gilks and C. Berzuini, "Following a moving target – monte carlo inference for dynamic bayesian models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 1, pp. 127–146, 2001.

[21] 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.

[22] R. B. Rusu, "Semantic 3d object maps for everyday manipulation in human living environments," Ph.D. dissertation, Computer Science department, Technische Universitaet Muenchen, Germany, Oct 2009.

[23] K. Tanaka, "Inferotemporal cortex and object vision," *Annual Review of Neuroscience*, vol. 19, pp. 109–139, 1996.

[24] M. Ito, H. Tamura, I. Fujita, and K. Tanaka, "Size and position invariance of neuronal responses in monkey inferotemporal cortex," *Journal of Neurophysiology*, vol. 73, no. 1, pp. 218–226, 1995.

[25] T. Madl, S. Franklin, K. Chen, D. Montaldi, and R. Trappl, "Bayesian Integration of Information in Hippocampal Place Cells," *PLoS ONE*, vol. 9, no. 3, pp. e89 762+, Mar. 2014.

[26] N. K. Logothetis, J. Pauls, and T. Poggiot, "Shape representation in the inferior temporal cortex of monkeys," *Curr. Biol*, pp. 552–563, 1995.

[27] A. Beghdadi, M.-C. Larabi, A. Bouzerdoum, and K. Iftekharuddin, "A survey of perceptual image processing methods," *Signal Processing: Image Communication*, vol. 28, no. 8, pp. 811–831, Sep. 2013.

[28] D. Fox, "KLD-sampling: Adaptive particle filters," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 14. MIT Press, 2001.

[29] S. Kullback, *Information Theory and Statistics*, ser. Dover Books on Mathematics. Dover Publications, 1997.