3D Computer Vision: From Points to Concepts

Luís A. Alexandre(✉)

Department of Informatics and Instituto de Telecomunicações,
University of Beira Interior, Covilhã, Portugal
luis.alexandre@ubi.pt

Abstract. The emergence of cheap structured light sensors, like the Kinect, opened the door to an increased interest in all matters related to the processing of 3D visual data. Applications for these technologies are abundant, from robot vision to 3D scanning. In this paper we go through the main steps used on a typical 3D vision system, from sensors and point clouds up to understanding the scene contents, including key point detectors, descriptors, set distances, object recognition and tracking and the biological motivation for some of these methods. We present several approaches developed at our lab and some current challenges.

1 Introduction

There are currently many application fields for 3D computer vision (3DCV). One of the recent pushes to the 3D computer vision was the appearance of cheap 3D sensors, such as the Microsoft Kinect. This was not developed for 3D computer vision but for the (console) video gaming industry. 3DCV is used in games as a means to receive user input. Other applications of 3DCV can be found in biometrics, such as for 3D facial and expression recognition, in robotic vision, industrial quality control systems or even in online shopping[1].

We present the current 3D technologies and the most used sensors in Sect. 2. In Sect. 3 the focus will be on keypoint extraction from 3D point clouds. Section 4 discusses 3D descriptors and the following section presents methods used on 3D object recognition. Section 6 presents a 3D tracking method based on keypoint extraction and Sect. 7 indicates some current challenges in this field. The final section contains the conclusion.

2 3D Sensors

There are several possible technologies for obtaining 3D images. These 3D images are in fact sets of points in space called point clouds. These points have, besides

L.A. Alexandre—This work was partially financed by FEDER funds through the Programa Operacional Factores de Competitividade - COMPETE and by Portuguese funds through FCT - Fundação para a Ciência e a Tecnologia in the framework of the project PTDC/EIA-EIA/119004/2010.

[1] See http://metail.com/.

A. Fred et al. (Eds.): ICPRAM 2015, LNCS 9493, pp. 3–14, 2015.
DOI: 10.1007/978-3-319-27677-9_1

their 3D coordinates, typically at least a gray scale value or RGB value but can have other measures associated, such has a local curvature. The 3D images can also be represented by two 2D images: one containing the illumination intensity of color values for scene locations and the other the respective depth or distance to the sensor.

A basic approach to obtaining 3D images is by inferring the depth from two different views of a scene (parallax). This can be done by using a single camera and positioning it in different locations (for a static scene) or more commonly, by using two cameras, mimicking the animal's visual sensors (eyes) layout, as in Fig. 1. The major difficulty in this approach is identifying the same scene point in both images to obtain the point disparity. Many approaches have been proposed to achieve this[2].

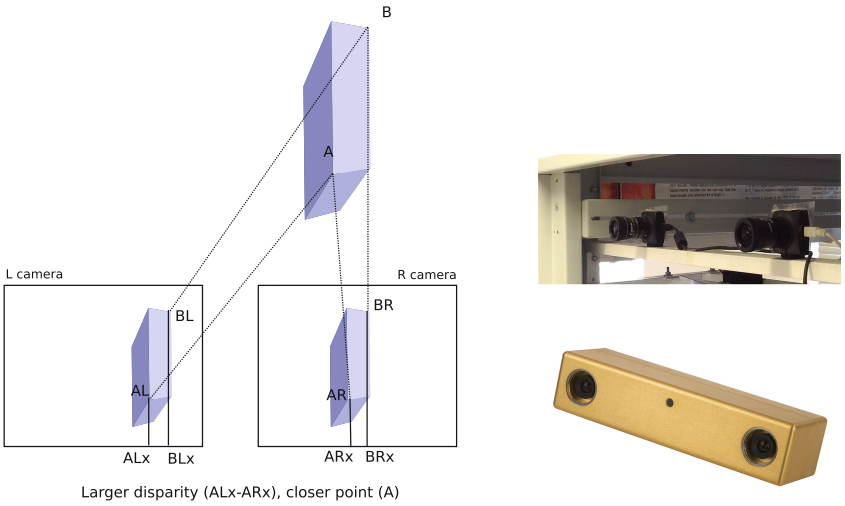

Fig. 1. Parallax-based stereo vision (left) and examples of sensors based on this approach (right-top, a "home-made" stereo vision system and right-bottom, a Bumblebee2 camera).

Another way to obtain 3D visual data is by using active vision and projecting a pattern in the scene that is used to identify the scene points' relative position. This approach is called a structured light approach. Figure 2 presents the idea and shows several sensors based on this approach. The pattern projection is usually made using infrared light such that it doesn't appear in the visible image. A third approach to obtaining 3D images is by inferring the scene points' distance to the sensor by measuring the time light takes to travel from an emitter located near the sensor, to the scene point and returning to the sensor. Since the speed

[2] Check for instance the disparity algorithms at the Middlebury Stereo Vision Page http://vision.middlebury.edu/stereo/.

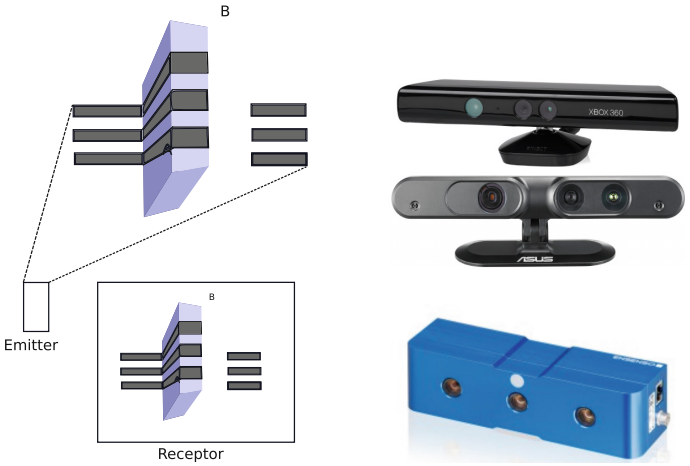

Fig. 2. Projection of a structured light pattern onto the scene (left) and several structured light sensors (from top to bottom): Microsoft Kinect, Asus Xtion Live, IDS Ensenso N20.

Fig. 3. The principle behind ToF sensors (left) and several sensors (from top left to bottom right): Microsoft Kinect 2, DepthSense, Creative Senz3D, Intel RealSense F200, Fotonic, PMD CamBoard pico^s.

of light in the air is known, the time taken is enough to infer the distance, or depth. Figure 3 illustrates this and presents some commercial available sensors based on this idea.

Size and weight have been falling to the point of currently having a 3D sensor inside a cell phone (see project Tango by Google), something that opens the way to many possible new mobile applications.

These sensors eventually produce a point cloud, typically at 30 fps. For 30 k points with RGB at 30 fps (typical Kinect specification), more than 30 MB/s of

data are generated. This can be too much data specially for embedded applications, so some form of sub-sampling must be used to reduce the computational burden of processing this type of data stream.

3 Keypoints

Keypoints are a set of points considered representative of the point cloud. They are extracted from a point cloud when the full data stream is considered too much data for real-time processing. So keypoints are a way to do sub-sampling. Figure 4 presents two different approaches to keypoint extraction: regular spaced sub-sampling using a voxel grid with two different voxel sides (left 1 cm and center 2 cm) and a Harris3D extractor (right). The figure also shows the location of the keypoints (the black dots) and the number of extracted keypoints.

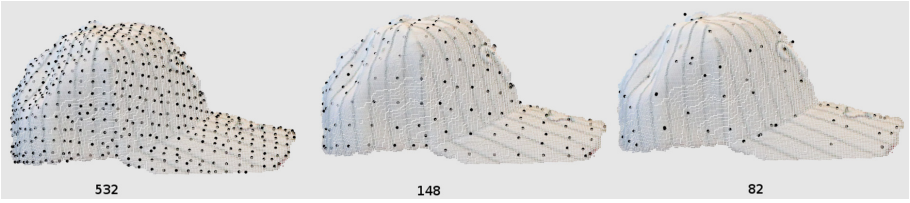

Fig. 4. Examples of keypoint extraction: left and center shows the keypoints obtained using voxel grid sub-sampling; right shows the result of the Harris3D keypoint extractor. Keypoints appear as black dots superimposed on the original point cloud.

Humans don't process every "input pixel", but focus their attention on salient points.

We have recently proposed [6] a 3D keypoint detector based on a computational model of the human visual system (HVS): the Biologically Inspired 3D Keypoint based on Bottom-Up Saliency (BIK-BUS). This approach is inspired on the visual saliency and the method mimics the following HVS mechanisms:

- Center-surround cells: sensitive to the center of their receptive fields and are inhibited by stimuli in its surroundings.
- Color double-opponency: neurons are excited in the center of their receptive field by one color and inhibited by the opponent color (red-green or blue-yellow) while the opposite takes place in the surround.
- Impulse response of orientation-selective neurons is approximated by Gabor filters.
- Lateral inhibition: neighboring cells inhibit each other through lateral connections.

Figure 5 presents a general view of the proposed method. The input point cloud is filtered to obtain color, intensity and normal orientation data. This is then used to build multi-scale representations of these features (Gaussian pyramids) that are combined using a mechanism that simulates center-surround cells

and a normalization operator motivated by lateral inhibition to generated feature maps. From these feature maps, new maps, called conspicuity maps, are generated combining information from multiple scales. The three conspicuity maps are combined into a single saliency map. Finally, from the saliency map, and through the use of inhibition mechanisms, the 3D keypoints can be selected.

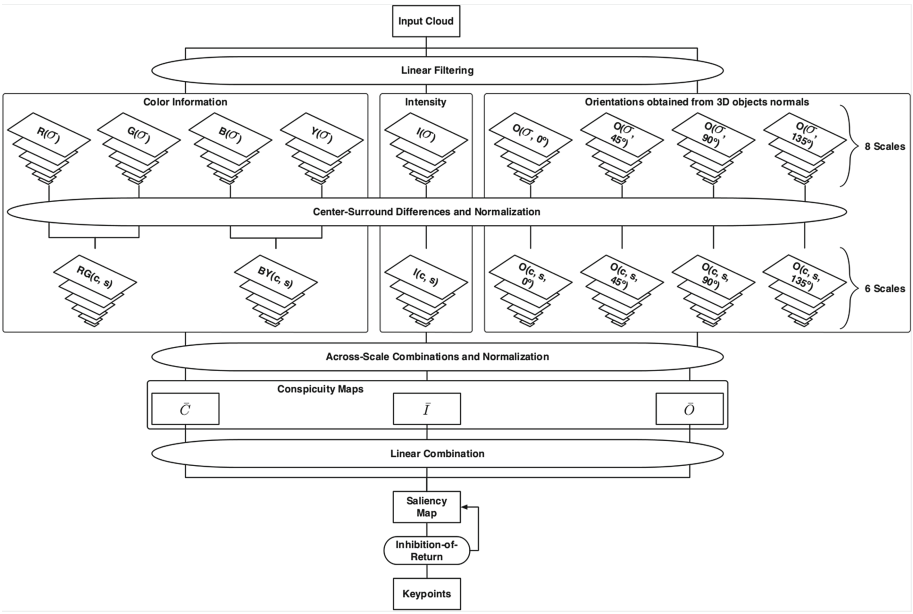

Fig. 5. General view of the BIK-BUS 3D keypoint detection method. Figure from [6].

Table 1. Number of times a keypoint detector came out as the best in the experiments run in [6].

Detector	Category		Object		Total
	AUC	DEC	AUC	DEC	
BIK-BUS	7	9	7	9	32
Curvature	3	2	2	1	8
Harris3D	1	1	0	0	2
ISS3D	2	0	4	2	8
KLT	2	1	1	0	4
Lowe	0	0	1	0	1
Noble	0	1	2	1	4
SIFT3D	0	0	0	1	1
SUSAN	2	1	2	1	6

We evaluated our proposal against 8 state-of-the-art detectors. We performed around 1.6 million comparisons for each pair keypoint detector/descriptor for a total of 135 pairs (9 keypoint detectors × 15 descriptors). The evaluation considered two metrics: area under the ROC curve (AUC) and the decidability (DEC). Table 1 shows the number of times each keypoint detector was the best on the experiments run. BIK-BUS was a clear winner with the second best methods at a considerable distance.

4 Descriptors

4.1 Evaluating Descriptors

A descriptor is a measure extracted from the input data that represents or describes an input data region in a concise manner. They are used to represent input data and allow a system to keep only a condensed representation of the input data (they are the equivalent of features in standard pattern recognition). There is a wide choice of descriptors: which should one use? We made an evaluation of 13 available in PCL [1]. Figure 6 shows the time taken and space used by the evaluated descriptors when they were applied after 3 different keypoint detectors.

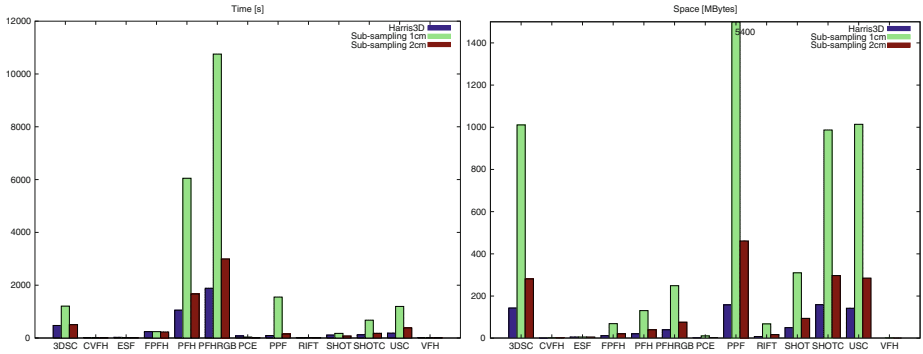

Fig. 6. Evaluation results of 13 descriptors in terms of time and space used in the experiments performed in [1].

Figure 7 shows the Precision-Recall curves for the experiments that used the 1 cm voxel grid sub-sample keypoint detector. Color-based descriptors are better (PFHRGB and SHOTCOLOR). Further details, including the equivalent figures for the remaining 2 keypoint detector approaches can be found in [1].

4.2 Genetic Algorithm-Evolved 3D Point Cloud Descriptor

From the evaluation of the descriptors discussed in the above section, we concluded that accurate descriptors are very computationally intensive and faster

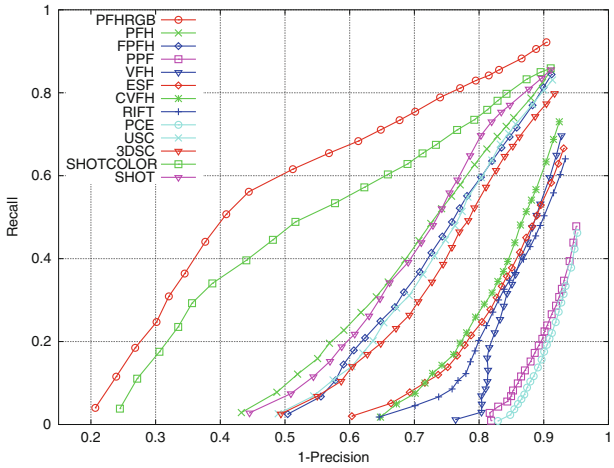

Fig. 7. Precision-Recall curves for the experiments that used the 1 cm voxelgrid sub-sample keypoint detector. Figure from [1].

descriptors use large storage space. For embedded approaches, such as robot-based vision, where computational resources and storage space come at a cost or might not be available in adequate amounts, a simple descriptor is desirable. For this type of application, we developed [8] a genetic algorithm(GA)-based descriptor that is both fast and has a small space footprint, while maintaining an acceptable accuracy.

It works by creating a keypoint cloud by sub-sampling with a voxel grid with leaf size of 2 cm. Two regions around each keypoint are considered: disk (R_1) + ring ($R_2 - R_1$) (see Fig. 8).

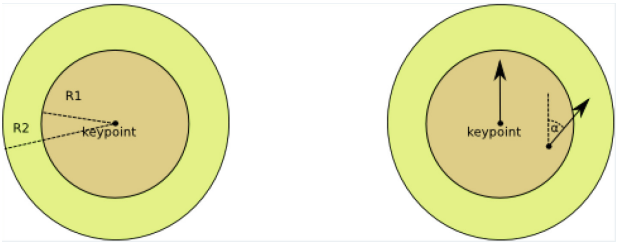

Fig. 8. Left: circular regions considered. Right: angle between the normal at a point in the R_1 region and the normal at the keypoint.

The information stored by the descriptor considers both shape and color information around each keypoint. For the shape, the descriptor records the histogram of angles between normals at keypoint and at each neighbor in region.

For the color information, a (Hue, Saturation) histogram for all points in each region is stored. The used distance between 2 point clouds represented by this descriptor is calculated using: $d = w.d_{shape} + (1 - w).d_{color}$, where the weight w is obtained through the GA optimization procedure. In total, 5 parameters (#shape bins, #color bins, R_1, R_2, w) are searched using the GA on the training data set. The obtained results can be seen in Table 2. This proposal allows for a much faster and lightweight (in terms of space) descriptor, with accuracy comparable to the SHOTCOLOR descriptor, and is thus adequate for use in situations where the computational cost of algorithms is an issue and/ or the available storage space is small.

Table 2. Average error, time and size of the three descriptors evaluated in [8].

Descriptor	Object [%]	Category [%]	Time [s]	Size
PFHRGB	20.25	5.27	2992	250
SHOTCOLOR	26.58	9.28	178	1353
Our	27.43	10.34	72	248

5 3D Object Recognition

The typical 3D object recognition pipeline consists on: obtaining the input data usually in the form of a point cloud; making keypoint detection; finding descriptors at each keypoint that are then grouped into a set that represents the input point cloud. After this, in a test ou deployment phase, incoming point clouds are compared against stored ones in an object database using, for instance, a set distance.

So, each point cloud is represented by a set of descriptors, and each descriptor is n-dimensional. In practice, a given point cloud will can have an arbitrary number of descriptors representing it, so the cardinal of the set of descriptors that represents the input data is not constant. To find the closest object in a database, a match to the input point cloud, we need to use a set distance.

5.1 Set Distances

Set distances are usually built around point distances. Three common point distances are the following: consider $x, y \in \mathbb{R}^n$, then

– City-block:

$$L_1(x, y) = \|x - y\|_1 = \sum_{i=1}^{n} |x(i) - y(i)|$$

– Euclidean:

$$L_2(x,y) = \|x-y\|_2 = \sqrt{\sum_{i=1}^{n}(x(i)-y(i))^2}$$

– Chi-squared:

$$d_{\chi^2}(x,y) = \frac{1}{2}\sum_{i=1}^{n}\frac{(x(i)-y(i))^2}{x(i)+y(i)}$$

Consider that a, b are points and A, B are sets. Let us also consider the following set distances:

– $D_1(A,B) = \max\{\sup\{f(a,B) \mid a \in A\}, \sup\{f(b,A) \mid b \in B\}\}$ with $f(a,B) = \inf\{L_1(a,b),\ b \in B\}$
– D_2 = Pyramid Match Kernel distance [7]
– $D_3(A,B) = L_1(\min_A, \min_B) + L_1(\max_A, \max_B)$ with
$\min_A(i) = \min_{j=1,\ldots,|A|}\{a_j(i)\},\ i = 1,\ldots,n$
$\max_A(i) = \max_{j=1,\ldots,|A|}\{a_j(i)\},\ i = 1,\ldots,n$
and similarly for $\min_B(i)$ e $\max_B(i)$.
– $D_4(A,B) = L_1(c_A, c_B)$ where c_A, c_B are cloud centroids
– $D_5(A,B) = L_2(c_A, c_B)$
– $D_6(A,B) = D_4(A,B) + L_1(std_A, std_B)$ with
$std_A(i) = \sqrt{\frac{1}{|A|-1}\sum_{j=1}^{|A|}(a_j(i) - c_A(i))^2},\ i = 1,\ldots,n$ and similarly for std_B.
– $D_7(A,B) = d_{\chi^2}(c_A, c_B) + d_{\chi^2}(std_A, std_B)$
– $D_8(A,B) = \frac{1}{|A||B|}\sum_{i=1}^{|A|}\sum_{j=1}^{|B|} L_1(a_i, b_j)$

We evaluated [2] these 8 distances using 2 descriptors (PFHRBG and SHOTCOLOR). We used a data set with 48 objects from 10 categories and 1421 point clouds. The keypoint detector used was Harris3D. Figure 9 shows the precision-recall curves for the experiments with both descriptors. Table 3 contains the time it took for the evaluation of the test set on a machine running with 12 threads.

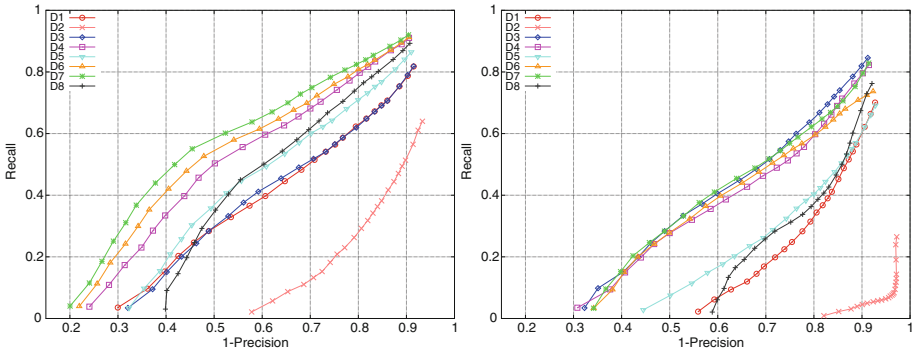

Fig. 9. Results of the distance evaluation using: PFHRBG (left) and SHOTCOLOR (right). Figure from [2].

Table 3. Time in seconds for test set evaluation (12 threads).

Distance	PFHRGB	SHOTCOLOR
D_1	1914	175
D_2	2197	1510
D_3	1889	132
D_4	1876	137
D_5	1886	134
D_6	1885	132
D_7	1883	113
D_8	1914	174

Table 4. Average error and time used on 10 repetitions for the different approaches.

Approach	Error [%]	Time[s]	Approach	Error [%]	Time[s]
RGBD	29.87	714.60	Channel G + TL	37.47	166.60
Channel R	32.15	136.50	Channel B + TL	43.58	95.10
Channel G	44.02	131.60	Channel D + TL	66.32	157.70
Channel B	55.62	110.10	R,G+TL,B+TL,D+TL maj	33.45	555.90
Channel D	65.85	126.30	R,G+TL,B+TL,D+TL mean	**28.80**	555.90
R,G,B,D maj	36.72	**504.50**	R,G+TL,B+TL,D maj	32.63	524.50
R,G,B,D mean	29.58	**504.50**	R,G+TL,B+TL,D mean	29.01	524.50

Simple distances like D_6 and D_7 are a good choice (accurate and fast) better than more common distances such as D_1 and D_2. Additionally, simple distances don't need any parameter search, as is the case with D_2.

5.2 Deep Transfer Learning for 3D Object Recognition

Deep learning is showing great potential in pattern recognition. The idea of transfer learning (TL) is also a very appealing one: learn in one problem and reuse (at least part of) the knowledge in other problems. We used both these ideas in a work where a convolutional neural network learns to recognize objects from 3D data [3]. TL is used from one color channel to the others and also to the depth channel. Decision fusion is used to merge each nets predictions. The results appear in Table 4. As can be seen, the TL approach is successful in obtaining both higher accuracy and shorter time that the baselines considered.

6 3D Object Tracking

The world is dynamic: another step towards understanding it is to follow objects as they move, since movement is a very important visual cue. There are many different approaches to tracking: the most used are particle filter variants [4].

We used a biologically-inspired keypoint extractor to initialize and maintain particles for particle filter-based tracking from 3D [5]. A general overview of the proposed method appears in Fig. 10.

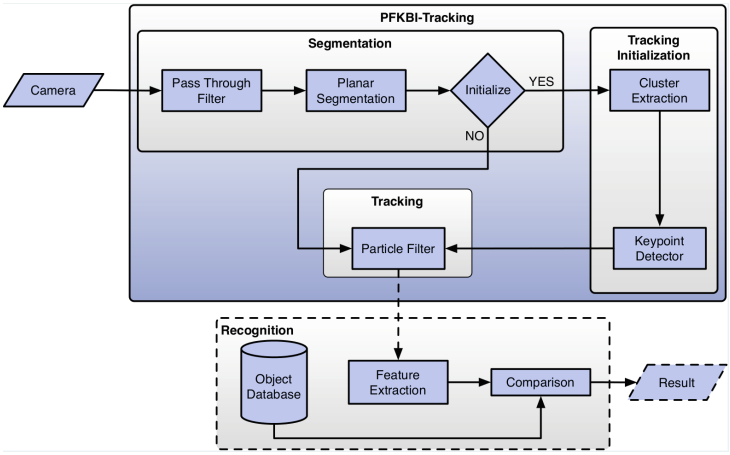

Fig. 10. A general view of PFBIK. Figure from [5].

We compared our tracker against the OpenNI tracker available in PCL. The videos used 10 different moving objects and a total of 3300 point clouds. The results are presented in Table 5. The PFBIK tracker used a much smaller number of particles that enabled it to be much faster during tracking, with the exception of the initialization where it was slower than the OpenNITracker given the necessary keypoint detection (that is only done at the start of the tracking process). The PFBIK was also slightly more accurate as can be seen through the distance error to the tracked object centroid.

Table 5. Results of PFBIK tracking compared to OpenNITracker: number of particles used, initialization and iteration time in seconds and distance error to the tracked centroid in meters.

Method	Particles	Init. time	Track time	Distance error
PFBIK	102 ± 92	0.20 ± 0.16	0.08 ± 0.06	0.045 ± 0.036
OpenNITracker	2132 ± 1988	0.17 ± 0.19	0.19 ± 0.17	0.052 ± 0.022

7 Challenges

Although recent progress in 3DCV has been substantial, there are still many challenges in the field. Some of the current challenges faced by the 3D computer vision community are:

- object representation: in this paper we showed object representations based on sets of descriptors of partial object views (2.5D) but other possibilities might be better (represent an object using a fused view representation, for instance). The best object representation approach may depend on the particular application and is still a important research topic;
- non-rigid object recognition: current keypoint plus descriptor approach is not a good solution when the objects are not rigid. More complex models are needed (3D deformable models);
- activity recognition: what are the best approaches to understand human activities from 3D video? This is currently a hot research topic;
- real-time processing: GPU-based implementations of most algorithms can bring us here but it is still a problem with embedded devices (cloud-based processing requires high bandwidth and permanent connection).

8 Conclusion

This paper summarizes the invited talk present at ICPRAM 2015 where the author reviewed some of the key concepts of 3D computer vision and presented some of the recent work in this field produced by him and his co-authors.

References

1. Alexandre, L.A.: 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 (IROS), Vilamoura, Portugal, October 2012
2. Alexandre, L.A.: Set distance functions for 3D object recognition. In: Ruiz-Shulcloper, J., Sanniti di Baja, G. (eds.) CIARP 2013, Part I. LNCS, vol. 8258, pp. 57–64. Springer, Heidelberg (2013)
3. Alexandre, L.A.: 3D object recognition using convolutional neural networks with transfer learning between input channels. In: Menegatti, E., Michael, N., Berns, K., Yamaguchi, H. (eds.) Intelligent Autonomous Systems 13. AISC, pp. 889–898. Springer, Heidelberg (2014)
4. Del Moral, P.: Mean field simulation for monte carlo integration. Chapman and Hall/CRC, Boca Raton (2013)
5. Filipe, S., Alexandre, L.: Pfbik-tracking: Particle filter with bio-inspired keypoints tracking. In: 2014 IEEE Symposium on Computational Intelligence for Multimedia. Signal and Vision Processing (CIMSIVP), pp. 1–8, Florida, USA, December 2014
6. Filipe, S., Itti, L., Alexandre, L.A.: BIK-BUS: biologically motivated 3D keypoint based on bottom-up saliency. IEEE Trans. Image Process. **24**(1), 163–175 (2015)
7. Grauman, K., Darrell, T.: The pyramid match kernel: efficient learning with sets of features. J. Mach. Learn. Res. **8**, 725–760 (2007)
8. Węgrzyn, D., Alexandre, L.A.: A genetic algorithm-evolved 3D point cloud descriptor. In: Ruiz-Shulcloper, J., Sanniti di Baja, G. (eds.) CIARP 2013, Part I. LNCS, vol. 8258, pp. 92–99. Springer, Heidelberg (2013)