A Genetic Algorithm-Evolved 3D Point Cloud Descriptor ⋆

Dominik Węgrzyn and Luís A. Alexandre

IT - Instituto de Telecomunicações
Dept. of Computer Science, Univ. Beira Interior, 6200-001 Covilhã, Portugal

Abstract. In this paper we propose a new descriptor for 3D point clouds that is fast when compared to others with similar performance and its parameters are set using a genetic algorithm. The idea is to obtain a descriptor that can be used in simple computational devices, that have no GPUs or high computational capabilities and also avoid the usual time-consuming task of determining the optimal parameters for the descriptor. Our proposal is compared with other similar algorithms in a public available point cloud library (PCL [1]). We perform a comparative evaluation on 3D point clouds using both the object and category recognition performance. Our proposal presents a comparable performance with other similar algorithms but is much faster and requires less disk space.

1 Introduction

The current cheap depth+RGB cameras like the Kinect and the Xtion have increased the interest in 3D point cloud acquisition and processing. One of the key steps when processing this type of data are the descriptors, that enable a compact representation of a region of a point cloud. Although there are already several available descriptors [1, 2], the motivation for this work was two-fold: first, many of the available descriptors are computationally demanding, and make it difficult to use them in computationally restricted devices; second, all the descriptors require the adjustment of one or more parameters, which is usually done using a grid search or other similar process, which can be a lengthy process.

The aim of this work is to design a descriptor that is computationally simple and hence fast and that has its parameters obtained using a genetic algorithm (GA), so as to address the two points raised above.

Section 2 presents the pipeline used in this work, from the input clouds to the matching stage. Section 3 explains the ideas behind the proposed descriptor. The following section describes the use of the GA with our descriptor. Section 5 illustrates the results of the new descriptor and compares it to similar available descriptors. The final section contains the conclusion and possible future work.

2 3D Object Recognition Pipeline

In this work the proposed descriptor uses both shape and color information. In order to represent this information histograms were used.

⋆ We acknowledge the financial support of project PEst-OE/EEI/LA0008/2013.

First keypoints are extracted from the input clouds in order to reduce the cost of computing the histograms. The keypoint cloud represents the input cloud by containing only a subset of the original cloud such that an increased processing speed can be achieved.

After computing the keypoints we find the normals of both, the input and the keypoint clouds. The normals are used in the calculation of the shape histograms that will be part of the final descriptor, as described below. The keypoint cloud is obtained from the input cloud using a VoxelGrid [1] with leaf size of 2 cm.

The second part of the descriptor consists in adding color information. For this purpose the RGB colors are transformed into HSV. This model is used because with the HSV color space we can use only the H and S channels and obtain illumination invariance in terms of color representation. We create another histogram for the color component using the hue and saturation channels. For the matching process the input cloud descriptor is compared against the stored descriptors of known objects, using a set distance function. The object recognition pipeline used is presented in figure 1.

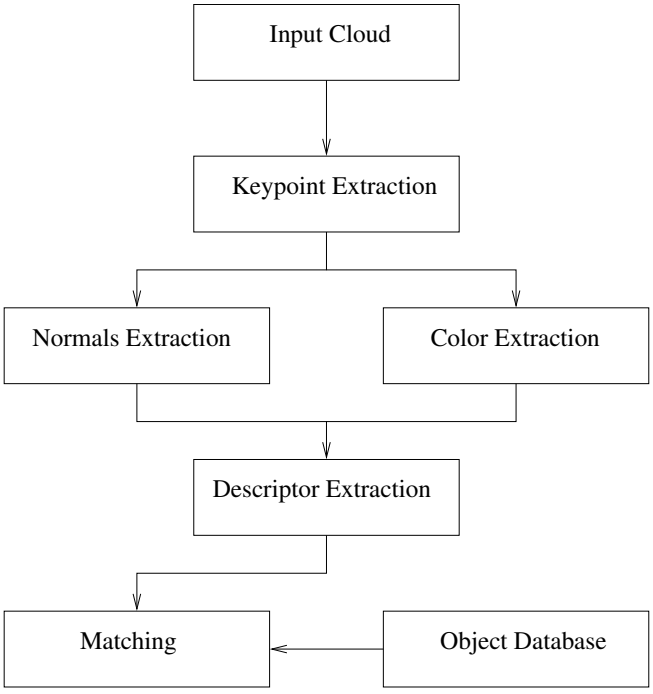

Fig. 1. Object recognition pipeline

3 Descriptor

3.1 Regions around the keypoints

The descriptor contains two parts: one to represent the shape and another to represent the color. The data used are collected from two regions around the keypoints, the first is a disk with radius R_1 and the second is a ring obtained by removing the first disk from the one obtained using radius R_2. This approach was proposed in [3]. These regions are illustrated in figure 2.

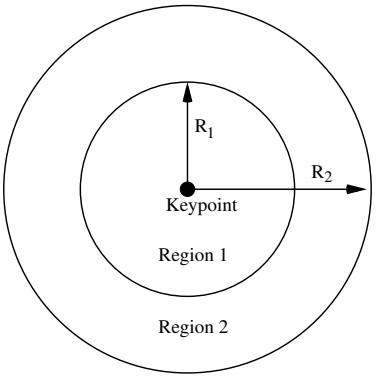

Fig. 2. The two concentric regions used to collect data around a keypoint: a disk (region 1) and a ring (region 2).

The advantage of using this disk and ring regions is that it makes it possible to analyze two areas around the keypoint to create the histograms. They separate the information between points that are very close to the keypoint and the points that further away from it, yielding a better representation of the region around the keypoints.

3.2 Shape information

After computing the keypoint clouds and all normals, for each keypoint we search all of its neighbors inside region 1. This search is done in the original input cloud and for this task the PCL utility KdTreeFLANN [1] is used.
The next step consists in finding the angle between the normal of this neighbor and the normal of the keypoint. This will give us information regarding the object's shape in the keypoint's neighborhood. Equation 1 shows how to calculate the $angle$.

$$angle = \arccos\left(\frac{Normal_{keypoint} \cdot Normal_{neighbor}}{||Normal_{keypoint}|| \cdot ||Normal_{neighbor}||}\right) \tag{1}$$

The angle is used in degrees.

We use an histogram to count the occurrences of the angles in the keypoint's neighborhood. The incremented bin is found using equation 2, where $shape_{bins}$ is the total number of histogram bins.

$$bin = \frac{angle \cdot (shape_{bins} - 1)}{360} \tag{2}$$

After we have found all $angle$ for the points in a keypoint's neighborhood, the histogram is normalized to sum 1.

The process just described is done also for the region 2 and a second shape histogram is obtained, at the same keypoint.

3.3 Color information

The color histogram contains the saturation and hue information.

Again we look at the neighbors of a keypoint for their hue, H, and saturation, S, values and select a bin_h and a bin_s, using the equations (3 and 4), where the total number of bins is m^2.

$$bin_h = \frac{H \cdot m}{360} \tag{3}$$

$$bin_s = \frac{S \cdot m}{100} \tag{4}$$

Now to find the correct $color_{bin}$ to be incremented in the color histogram, we use the coordinates (bin_h, bin_s) in the equation 5.

$$color_{bin} = m \cdot bin_h + bin_s \tag{5}$$

The color histogram is normalized to sum 1.

As we did in the case of the shape information, the process is repeated for region 2, yielding a second normalized color histogram.

The two histograms for regions 1 and 2 are concatenated yielding a single color histogram with a total number of bins equal to $2m^2$.

3.4 Cloud distance

To find the matching objects we need to compute the distances between two clouds.

In order to get the distances between shape histograms from two clouds first we compute the centroid of the shape histograms of each cloud (c_1 and c_2), then using the chi-squared distance [4] (equation 6) we get the distance between the two centroids.

$$d_{cent} = \sum_i \frac{(c_1[i] - c_2[i])^2}{2(c_1[i] + c_2[i])} \tag{6}$$

Then we do a similar step but instead of using the centroids we use the standard deviation of the histograms of each cloud. Equation 7, shows how to find this value for cloud 1, and a similar calculation should be done for the second cloud to be compared.

h_1 is the shape histogram, while c_1 is the centroid and N_1 is the number of keypoints in this cloud. We use the same process (equation 6) to get the distance, d_{std}, between these standard deviations histograms (std_1 and std_2) as we used for the centroid histograms (c_1 and c_2).

$$std_1 = \sqrt{\frac{\sum_i (h_1[i] - c_1[i])^2}{N_1 - 1}} \tag{7}$$

Finally we sum the centroid distance, d_{cent}, with the standard deviation distance d_{std}, [2] for the final shape distance: $d_{sh} = d_{cent} + d_{std}$. The same process is used to compute the color histogram distance, d_{cl}.

Equation 8 shows how we compute the final distance (d_{final}) between two clouds, where d_{sh} is the shape distance between two histograms and d_{cl} is the color distance between the same histograms. We use the weight w to represent the relative importance of the shape and color information regarding the final distance.

$$d_{final} = d_{sh} \cdot w + d_{cl} \cdot (1 - w) \tag{8}$$

The next step is to find which test cloud fits best to each training cloud, this means the one with the smallest distance. The results of the matching of all test point clouds are used to produce the final result of the dataset.

4 Genetic Algorithm

In this work a genetic algorithm [5], tries to find the optimal parameters for the 3D cloud descriptor. The chromosomes encode the following 5 descriptor parameters: $Shape_{bins}$, m, R_1, R_2 and w (which is in fact used in the distance between the descriptors, and not in the descriptors themselves).

The role of each of these parameters in the creation of the descriptor was explained in the previous section.

The GA has some restrictions, which are the intervals in which the parameters lie. The parameter $Shape_{bins}$ is set between 8 and 64. The parameter m is set between 3 and 8. The parameter R_1 is set between 0.5 and 2.0 cm and R_2 is set between 2.1 and 5.0 cm. The R_2 has the maximal value possible set to 5.0 cm as the other descriptors used to compare with our descriptor also use this value. The last parameter optimized by the GA is the weight w, that is allowed to vary between 0 and 1.

The chromosomes are binary encoded. The initial population is set randomly. The population used consisted of 10 chromosomes. This small value was chosen in order to avoid the generations taking too much time to compute. The object error represents the percentage of correctly matched point clouds from the training set 1 among the point clouds from training set 2 and is used as the fitness of the chromosome (training set 1 and training set 2 that are explained in the experiments section). The elitism selection [6] and the roulette-wheel selection [6] were used in order to make the selection. The used crossover technique is the uniform crossover [6]. Mutation makes it possible to search a wide area of solutions in order to find the optimal solution. The mutation used

is the in-order mutation [6]. The mutation probability is set to 40% in the first generation and decreases exponentially after each successive generation.

The GA needs to be able to stop the evolution when the goal is achieved. The stopping criterion is set to either the number of generations reaching 200 generations, or if no better solution in 40 consecutive generations is found, or if a solution with 100% of correct object matches is found.

After each generation we measure the validation error of the best chromosome. This way we avoid the overfitting of the descriptor that could lead to the loss of the ability that the descriptor has to recognize point clouds. For this purpose we check in a validation subset of point clouds (apart from the clouds used by GA to determine the best parameters for the descriptor) for the validation error of the best chromosome. When this error begins to rise, we stop the AG and consider that we have found the best descriptor.

5 Experiments

5.1 Dataset

A subset of the large dataset[1] of 3D point clouds [7] was used to perform experiments. The clouds were divided into four subsets, constituted by two training subsets, one validation subset and one test set. The $test_{clouds}$ subset is composed by 475 different views of 10 objects and is used to calculate the final performance of the descriptor using 943 training clouds as possible match. The $validation_{clouds}$ subset has 239 clouds and is used to avoid the overfitting of the descriptor. We calculate the validation error of the best chromosome of each generation when making use of the GA. For this purpose we check how many clouds from the $validation_{clouds}$ are correctly matched, using 704 training clouds as the matching clouds. On the other hand these 704 training clouds are divided into $training_1$ and $training_2$ subsets. Those two training subsets were used by the GA to get the object error of the chromosomes (both $training_1$ and $training_2$ subsets contain 352 clouds).

5.2 Results

The code used to implement our descriptor can be downloaded online[2].

The best chromosome optimized by the GA has 60 shape bins, $m = 8$, $R_1 = 1.3$ cm, $R_2 = 3.6$ cm and $w = 0.67$. After the matches are done, we check how many of the 475 test clouds were correctly matched to the 943 training clouds. The best descriptor uses 248 bins to represent the point cloud. This descriptor has an accuracy of 72.47% in matching the cloud to the correct object (from the 474 test clouds 344 were correctly matched – one of the test clouds was not used since it had less than 10 points) and 89.66% in matching the cloud to the correct category.

In the paper [2] some available descriptors were tested to get their performance and computational time. Table 1 shows the performance of those descriptors, that were extracted using the same point clouds as the ones used to evaluate the descriptor proposed

[1] http://www.cs.washington.edu/node/4229/

[2] http://www.di.ubi.pt/~lfbaa

in this paper. The column time refers to the necessary time to match the 475 test clouds using 943 point clouds as the training set. The column size refers to the number of real values required to store one descriptor.

Table 1. Descriptors performance: test errors for the object and category recognition tasks along with time and size requirements.

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

As we can see the SHOTCOLOR takes 178 s to compute, while our descriptor takes only 72 s using the same machine. Although we have a slightly lower accuracy, the temporal advantage can be important in real time applications. The PFHRGB has the best accuracy, however it takes 2992 s to compute. In terms of size, we can see that our descriptor uses only 248 real values that is significantly less than the SHOTCOLOR's 1353 and still less than the 250 values per descriptor that PFHRGB uses. Figure 3 contains the recall $\times$ (1-precision) curves for the object recognition experiments.

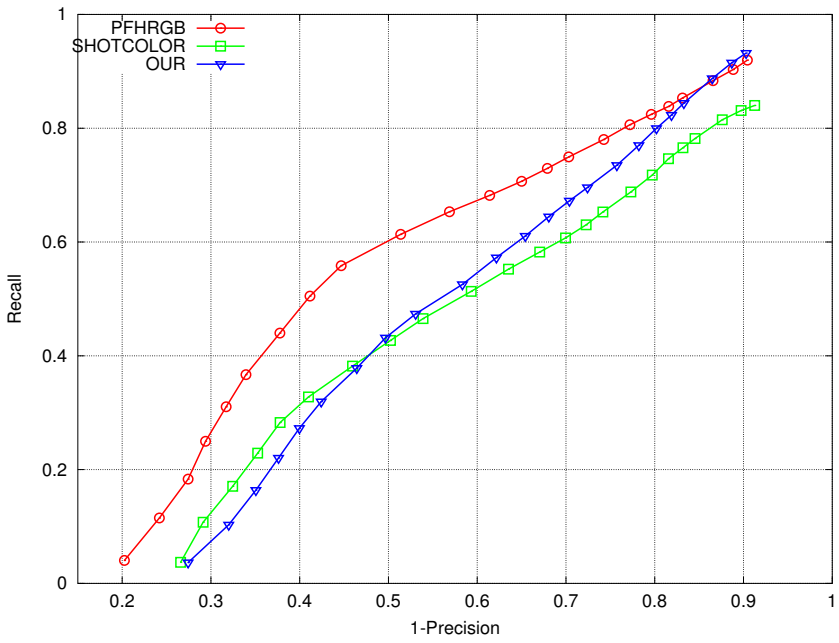

Fig. 3. Recall $\times$ (1-precision) curves for the object recognition experiments.

Although the PFHRGB curve is better than ours, we can see that our curve is close to the SHOTCOLOR curve and when we have a recall larger than 0.35 our curve is better than the SHOTCOLOR's.

6 Conclusion

In this paper we presented a new descriptor for 3D point clouds that takes advantage of a genetic algorithm to find good parameters. It presents a performance similar to other existing descriptors, but is faster to compute and uses less space to store the extracted descriptors.

Our descriptor when compared to the SHOTCOLOR presents a slightly higher error (27.43% versus 26.58% object recognition error) but it is much faster (uses 40% of the time needed by the SHOTCOLOR) and occupies less space. When compared to the PFHRGB, it is substantially faster (uses only 2.5% of the time needed by PFHRGB), uses the same space but has higher error (27.43% error versus 20.25%). So our proposal can be used to replace these descriptors, when extraction speed is important.

A possible way to improve the quality of the descriptor is to let the GA optimize not only the values of the parameters, but also the entire structure of the descriptor (types of angles used, their number, types of regions to consider and their shapes).

References

1. Rusu, R.B., Cousins, S.: 3d is here: Point cloud library (pcl). In: IEEE International Conference on Robotics and Automation (ICRA). (2011)
2. 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)
3. Torsten Fiolka, Jorg Stuckler, D.A.D.S., Behnke, S.: Place recognition using surface entropy features. In: in Proc. of IEEE ICRA Workshop on Semantic Perception, Mapping, and Exploration, Saint Paul, MN, USA. (2012)
4. Puzicha, J., Hofmann, T., Buhmann, J.M.: Non-parametric similarity measures for unsupervised texture segmentation and image retrieval. 2012 IEEE Conference on Computer Vision and Pattern Recognition **0** (1997) 267–272
5. Holland, J.H.: Adaptation in Natural and Artificial Systems. A Bradford Book (1975)
6. Engelbrecht, A.P.: Computational Intelligence, An Introduction. John Wiley & Sons (2002)
7. K. Lai, L. Bo, X.R., Fox, D.: A large-scale hierarchical multi-view rgb-d object dataset. In: Proc. of the IEEE International Conference on Robotics and Automation (ICRA). (2011)