Set Distance Functions for 3D Object Recognition[⋆]

Luís A. Alexandre

Instituto de Telecomunicações, Univ. Beira Interior, Covilhã, Portugal

Abstract. One of the key steps in 3D object recognition is the matching between an input cloud and a cloud in a database of known objects. This is usually done using a distance function between sets of descriptors. In this paper we propose to study how several distance functions (some already available and other new proposals) behave experimentally using a large freely available household object database containing 1421 point clouds from 48 objects and 10 categories. We present experiments illustrating the accuracy of the distances both for object and category recognition and find that simple distances give competitive results both in terms of accuracy and speed.

1 Introduction

There is a growing interest in the use of 3D point cloud images for many tasks, since the recent introduction of cheap sensors that produce RGB plus depth images, such as the Microsoft Kinect or the Asus Xtion.

One of the most challenging tasks to be achieved with such data is to recognize objects in a scene. An important part of the process of recognition is to be able to compare the representations of the input (test or probe) data against stored (train or gallery) data. The objects are usually represented by sets of descriptors. Several distances exist that are able to work with sets of descriptors, notably the Pyramid Match Kernel [1], for object recognition from images.

It is important to obtain a quantitative notion of the performance of such distance functions. In this paper we present a comparison between 8 distance functions for 3D object recognition from point clouds. Two types of descriptors are used and the relative distance performance is similar in both cases. We show both the object and category accuracies that can be obtained from these distances and also the computational cost in terms of the time it takes to process the test set used. From the experiments we conclude that good performance can be obtained using quite simple distance functions, both in terms of accuracy and speed.

The rest of the paper is organized as follows: the next section presents an overview of the 3D object recognition pipeline used in this paper, the following section explains the descriptors used; section 4 presents the distances that are evaluated; section 5 contains the experiments and the paper ends with the conclusions in section 6.

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

2 The 3D Object Recognition Pipeline

The input cloud goes through a keypoint extraction algorithm, the Harris3D keypoint detector implemented in PCL [2]. The covariance matrix of the surface normals on a point neighborhood is used to find the point's response to the detector. Then descriptors are obtained on the extracted keypoints and these form a set that is used to represent the input cloud. This set is matched against sets already present in the object database and the one with largest similarity (smallest distance) is considered the match for the input cloud.

3 Descriptors

In this paper we use the two descriptors that produced the best results in the comparative evaluation performed in [3]. They both use color information.

The first one is the Point Feature Histograms (PFH) [4]. This descriptor's goal is to generalize both the surface normals and the curvature estimates.

Given two points, p and q, a fixed reference frame, consisting of the three unit vectors (u, v, w), is built centered on p using the following procedure: 1) the vector u is the surface normal at p; 2) $v = u \times \frac{p-q}{d}$ 3) $w = u \times v$; where $d = \|p-q\|_2$. Using this reference frame, the difference between the normals at p (n_p) and q (n_q), can be represented by : 1) $\alpha = \arccos(v \cdot n_q)$; 2) $\phi = \arccos(u \cdot (p - q)/d)$; 3) $\theta = \arctan(w \cdot n_p, u \cdot n_p)$.

The angles α, ϕ, θ and the distance d are computed for all pairs in the k-neighborhood of point p. In fact, usually the distance d is dropped as it changes with the viewpoint, keeping only the 3 angles. These are binned into an 125-bin histogram by considering that each of them can fall into 5 distinct bins, and the final histogram encodes in each bin a unique combination of the distinct values for each of the angles. One of these 125-bin histograms is produced for each input point.

The version of PFH used in this paper includes color information and is called PFHRGB. This variant includes three additional histograms, one for the ratio between each color channel of p and the same channel of q. These histograms are binned as the 3 angles of PFH and hence produce another 125 float values, giving the total size of 250 values for the PFHRGB descriptor.

The second descriptor used is the SHOTCOLOR [5]. This descriptor is based on the SHOT descriptor [6], that obtains a repeatable local reference frame using the eigenvalue decomposition around an input point. Given this reference frame, a spherical grid centered on the point divides the neighborhood so that in each grid bin a weighted histogram of normals is obtained. The descriptor concatenates all such histograms into the final signature. It uses 9 values to encode the reference frame and the authors propose the use of 11 shape bins and 32 divisions of the spherical grid, which gives an additional 352 values. The descriptor is normalized to sum 1. The SHOTCOLOR adds color information (based on the CIELab color space) to the SHOT descriptor. It uses 31 bins each with 32 divisions yielding 992 values, plus the 352 from the SHOT which gives the total of 1344 values (plus 9 values to describe the local reference frame). The histograms in this case store the L_1 distance between the CIELab color of a point and the color of its neighbors.

4 Set Distances

The focus of this paper is on the distance function that should be used when comparing two point clouds that are represented by sets of descriptors. Note that the word "distance" should be interpreted loosely since some of the functions presented below do not verify all the conditions of a norm (for instance, D_4 and D_5 can produce a value of zero even if the two input clouds are not the same).

A descriptor can be seen as a point in $X \subset \mathbb{R}^n$. We investigate the performance of functions that receive two sets of descriptors, $A \subseteq X$ and $B \subseteq X$, with a possible different number of elements, $|A| \neq |B|$, and return a (distance) value in $\mathbb{R}$.

We will use below the following distances between descriptors (not sets) $x, y \in X$:

$$L_p(x,y) = \left(\sum_{i=1}^{n} \mid x(i) - y(i) \mid^p \right)^{1/p} \quad , \; p = 1, 2$$

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

We will assign a code to each set distance in the form D_z, where z is an integer to make it easier to refer to the several distances throughout the paper.

4.1 Hausdorff distance

Consider $S(X)$ to be the set of subsets of X that are closed, bounded and non-empty. Let $A, B \in S(X)$. The Hausdorff distance, D_1, between sets A and B is defined as

$$D_1(A,B) = \max\{\sup\{d(a,B) \mid a \in A\}, \sup\{d(b,A) \mid b \in B\}\}$$

where $d(a,B)$ is a distance between a point a and a set B, defined by

$$d(a,B) = \min\{d(a,b_i), \; i = 1, \ldots, |B|\}$$

and $d(a, b_i)$ is the distance between two points a and b_i in $\mathbb{R}^n$. In our case we use the L_1 distance between two points.

4.2 Pyramid Match Kernel

The pyramid match kernel (D_2) [1] uses a hierarchical approach to matching the sets. It finds the similarity between two sets as the weighted sum of the number of feature matchings found at each level of a pyramid.

Consider the input space X of sets of n-dimensional vectors bounded by a sphere of diameter D. The feature extraction function is

$$\Psi(x) = [H_{-1}(x), H_0(x), \ldots, H_L(x)]$$

where $L = \lceil \log_2 D \rceil + 1$, $x \in X$, $H_i(x)$ is a histogram vector formed over data x using n-dimensional bins of side length 2^i. Then, the pyramid referred above is given by:

$$K_\Delta(\Psi(y), \Psi(z)) = \sum_{i=0}^{L} N_i / 2^i$$

where N_i is the number of newly matched pairs at level i. A new match at level i is defined as a pair of features that were not in correspondence at an finer level $(j < i)$ became in correspondence at level i. To become in correspondence means that both fall in the same histogram bin.

4.3 Other set distances

We propose to evaluate also the following set distances, that are all variations around the same theme: use statistical measures like the mean, standard variation, maximum and minimum of the points in each set to develop simple representations for the set.The goal is to search for a simple set distance that produces accurate results and at the same time is fast, such that, other things permitting (the time the keypoints take to be detected plus the time the descriptor takes to extract) would allow for real time cloud processing.

Below we use $a_j(i)$ to refer to the coordinate i of the descriptor j.

The distance D_3 is obtained by finding the minimum and maximum values for each coordinate in each set and sum the L_1 distances between them

$$D_3 = L_1(\min_A, \min_B) + L_1(\max_A, \max_B)$$

where

$$\min_A(i) = \min_{j=1,\ldots,|A|} \{a_j(i)\},\ i = 1, \ldots, n$$

and

$$\max_A(i) = \max_{j=1,\ldots,|A|} \{a_j(i)\},\ i = 1, \ldots, n$$

and likewise for $\min_B(i)$ and $\max_B(i)$.

The next two distances are simply the distance between the centroids of each set, c_A and c_B respectively, using the descriptor distances L_1 and L_2:

$$D_4 = L_1(c_A, c_B) \ \text{ and } \ D_5 = L_2(c_A, c_B)\ .$$

Distance D_6 is the sum of D_4 with the L_1 distance between the standard deviation for each dimension (coordinate) of each set:

$$D_6 = D_4 + L_1(std_A, std_B)$$

where

$$std_A(i) = \sqrt{\frac{1}{|A|-1}\sum_{j=1}^{|A|}(a_j(i) - c_A(i))^2}, \; i = 1, \ldots, n$$

and likewise for std_B.

Distance D_7 is similar to D_6 but instead of using the L_1 distance uses the d_{χ^2} distance between two vectors:

$$D_7 = d_{\chi^2}(c_A, c_B) + d_{\chi^2}(std_A, std_B) \; .$$

The final distance to be evaluated consists on the average L_1 distance between all points in one set to all the points in the other (the normalized average linkage set distance):

$$D_8 = \frac{1}{|A||B|}\sum_{i=1}^{|A|}\sum_{j=1}^{|B|} L_1(a_i, b_j) \; .$$

5 Experiments

5.1 Dataset

We used a subset of the large dataset of 3D point clouds from [7]. The original dataset contains 300 objects from 51 different categories captured on a turntable from 3 different camera poses. We used 48 objects representing 10 categories. The training data contain clouds captured from two different camera views, and the test data contains clouds captured using a third different view. The training set has a total of 946 clouds while the test set contains 475 clouds. Since for each test cloud we do an exhaustive search through the complete training set to find the best match, this amounts to a total of 449.350 cloud comparisons for each of the evaluated descriptors and each of the distance functions used.

5.2 Setup

The code used in the experiments was developed in C++ using the PCL library [2] on a linux machine. The code used for D_2 was from [8]. We used the `UniformPyramidMaker` with the following parameters obtained from experiments with a 10% subset of the one used in the final evaluation: `finest_side_length` = (1/250, 10^{-4}), `discretize_order`=(3, 3) and `side_length_factor`=(2, 2) for (PFHRGB, SHOTCOLOR), respectively. To make a fair comparison between the distances, all steps in the pipeline are equal.

The descriptors are found on the keypoints obtained using the Harris3D keypoint detector with the following parameters: the radius for normal estimation and non-maxima supression (Radius) was set to 0.01 and the sphere radius that is to be used for determining the nearest neighbors used for the keypoint detection (RadiusSearch) was also set to 0.01.

The only parameter needed for the descriptor calculation is the sphere radius that is to be used for determining the nearest neighbors used in its calculation. It was set at 0.05 for both descriptors.

Table 1. Category and object recognition accuracy and the time used for evaluating the test set in seconds, for the different distances and descriptors.

	PFHRGB			SHOTCOLOR		
	Accuracy[%]			Accuracy[%]		
Distance	Category	Object	Time[s]	Category	Object	Time[s]
D_1	91.14	70.04	1914	67.72	44.09	175
D_2	63.92	42.19	2197	26.58	17.93	1510
D_3	88.82	67.93	1889	$\underline{88.82}$	67.72	132
D_4	90.93	75.95	1876	87.97	$\underline{69.20}$	137
D_5	82.70	67.72	1886	79.75	55.49	134
D_6	93.88	78.06	1891	87.76	65.82	134
D_7	$\underline{94.73}$	$\underline{79.96}$	1894	88.19	65.82	127
D_8	77.64	60.13	1914	71.73	41.35	174

5.3 Results

Table 1 and figure 1 contain the results of the experiments done.

An object is considered to be recognized when an input cloud is matched by one of the views of the same object in the database, whereas a category is considered to be recognized when the input cloud is matched to a view of any of the objects that are in the same category as the input object. So, category recognition is an easier task than that of object recognition, since in the latter case the system needs to distinguish between the (similar) objects within a given category. That category recognition is easier than object recognition can be seen in table 1. For all distance functions, category accuracy is always higher than object recognition.

Regarding the accuracies obtained, these results show the importance of choosing a good distance function. For a given descriptor there are considerable variations in terms of accuracy: in terms of object recognition the results for the PFHRGB vary from around 42% to almost 80% whereas for the SHOTCOLOR descriptor the results vary from around 18% to over 69%.

The best results are obtained for the PFHRGB with distance D_7 and for the SHOTCOLOR with distance D_3 for category recognition and D_4 for object recognition.

From the recall × (1-precision) curves in figure 1, we note that the results can be grouped into three sets: the best results for both descriptors, and with similar curves, are obtained with distances D_4, D_6 and D_7 (for SHOTCOLOR, D_3 is also on this first group). The second group contains the distances D_1, D_5 and D_8 (D_3 is in this second group for PFHRGB) that show a decrease in performance when compared with the first group. The difference in performance from group 1 to group 2 is larger with SHOTCOLOR than with PFHRGB. This might have to do with the fact that SHOTCOLOR works on a much higher dimensional space (1344) than PFHRGB (250). Distance D_2 is the sole member of the third group with a poor performance. We believe this might have to do with a poor choice of parameters. But having to choose 3 parameters for a distance that is very heavy

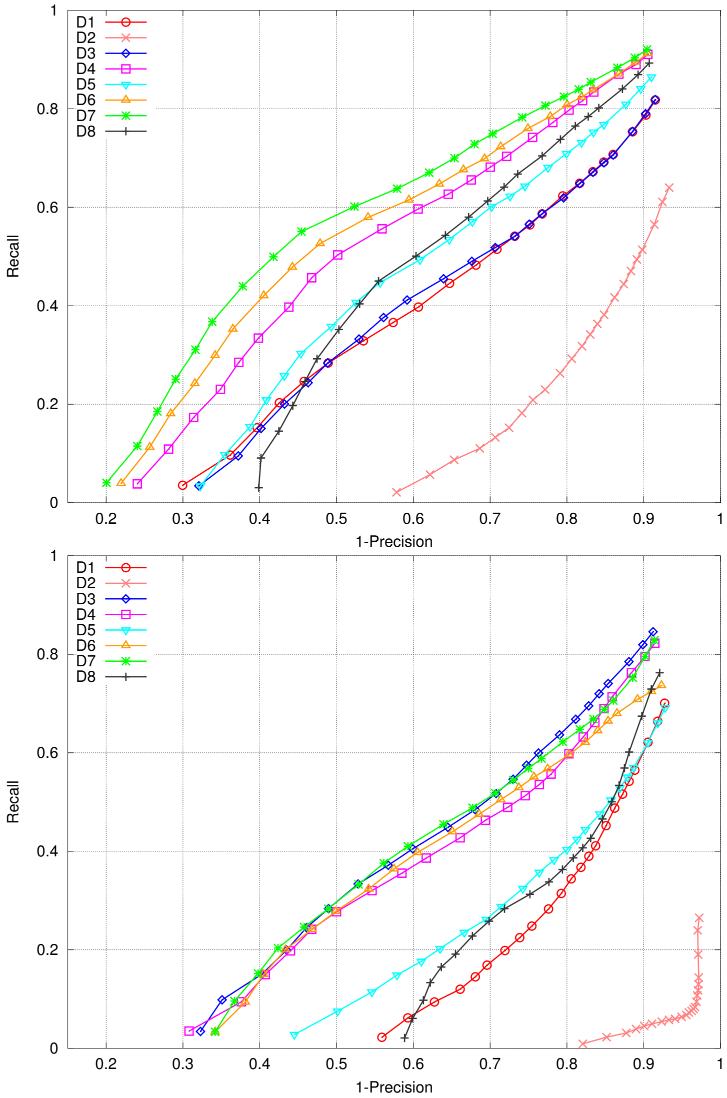

Fig. 1. Recall × (1-Precision) curves for the object recognition experiments using the PFHRGB (top) and SHOTCOLOR (bottom) descriptors (best viewed in color).

from a computational point of view is not an easy task and we might needed to spent more time searching for the optimal parameters to obtain a better result.

Distance D_4 is better than D_5 (these are simply the L_1 and L_2 distances between cloud centroids) for both descriptors, confirming the fact that the Euclidian distance is not appropriate for these high dimensional spaces.

The fifth and seventh columns of table 1 contain the time in seconds that took to run the evaluation (test set) on a 12 thread version using a i7-3930K@3.2GHz

CPU on Fedora 17. The PFHRGB is much more demanding in terms of computational complexity than the SHOTCOLOR, hence the time it takes is around 10 times more than the time used by the SHOTCOLOR. In terms of time taken to complete the tests, D_2 is much slower than the rest. Given its time overhead, D_2 should only be used if it could provide an improved accuracy when compared to the remaining distances, but that was not the case.

6 Conclusions

An important part of a 3D object recognition setup is the distance function used to compare input data against stored data. Since there are many possible distance functions that can be used in this scenario, the user is faced with a tough decision regarding which distance to choose. The obvious way is to make experiments comparing these functions for their particular descriptor and data, but this can be a time consuming task.

This paper presents an evaluation of 8 distance functions on a large point cloud dataset using two descriptors. From the results of the experiments made we conclude that simple distances (such as D_3, D_4, D_6 and D_7) can be a good choice since their performance both in terms of accuracy as in terms of speed surpasses other more common used ones such as D_1 and D_2. The former distances also benefit by not requiring the adjustment of parameters.

References

1. Grauman, K., Darrell, T.: The pyramid match kernel: Efficient learning with sets of features. Journal of Machine Learning Research **8** (2007) 725–760
2. Rusu, R., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China (2011)
3. 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 (2012)
4. Rusu, R., Blodow, N., Marton, Z., Beetz, M.: Aligning point cloud views using persistent feature histograms. In: International Conference on Intelligent Robots and Systems (IROS), Nice, France (2008)
5. Tombari, F., Salti, S., Di Stefano, L.: A combined texture-shape descriptor for enhanced 3D feature matching. In: IEEE International Conference on Image Processing. (2011)
6. Tombari, F., Salti, S., Di Stefano, L.: Unique signatures of histograms for local surface description. In: Proceedings of the 11th European conference on computer vision conference on Computer vision: Part III, Berlin, Heidelberg, Springer-Verlag (2010) 356–369
7. Lai, K., Bo, L., Ren, X., Fox, D.: A Large-Scale hierarchical Multi-View RGB-D object dataset. In: Proc. of the IEEE International Conference on Robotics & Automation (ICRA). (2011)
8. Lee, J.J.: Libpmk: A pyramid match toolkit. Technical Report MIT-CSAIL-TR-2008-17, MIT Computer Science and Artificial Intelligence Laboratory (2008)