Features for 3D Object Retrieval

Cristina González Delgado

June 14, 2016

ABSTRACT

This project is focused on the object retrieval challenging field, a critical area for robotics and computer vision systems.

Specifically, this project is trying to address some issues in the object recognition area, which is a key during the object retrieval process.
Recently, low cost RGB-D sensors able to capture not only color information but also depth data, have emerged. This is a great opportunity to increment efficiency and robustness of object retrieval applications by taking advantage of this additional depth information.

In this project I implemented an evaluation system of 3D feature descriptors by using the Point Cloud Library (PCL) , and used it to perform an specific assessment for two of the most widely used descriptors for object recognition: SHOTColor and PFHColor. They were tested in two different datasets containing data extracted with the Kinnect RGB-D sensor. The first dataset consist of 3D point clouds of individual objects and synthetically built scenes, the second one has 3D point clouds corresponding to individual model objects and scenes from the real world. The evaluation process was done considering the following three properties: descriptiveness, robustness to Gaussian noise impact and support radius variation, and efficiency.

As a final result, the outcome of this evaluation process is the performance assessment on those descriptors depending on the different scenarios. SHOTColor descriptors performed better in terms of descriptiveness, robustness to Gaussian noise impact and support radius variations, and efficiency as well. But, they required higher storage capacity than PFHColor descriptors. SHOTColor descriptors seems to be more sensitive to clutter and occlussion effects, since the performance for the real scenes dataset is worsening faster than the PFHColor one.
Given those considerations, SHOTColor descriptors are recommended as the best option for time-crucial applications and also for those ones requiring a strong descriptiveness power. On the other hand, PFHColor descriptors are better for spacial-crucial applications.

ACKNOWLEDGEMENTS

These last four months and a half have been an intense learning period for me, not only in the engineering field, but also in my personal life. I would like to thank who guided and supported me throughout this project.

First, I would like to thank my thesis supervisor at KTH, Markus Flierl, for his advices and his close attitude to me.

Also, I would like to thank Hanwei, who has guided me throughout this project and for being always willing to help me.

I would like to thank my family in Barcelona for its support and for always believing in me.

LIST OF FIGURES

Figure 1 Examples of Point clouds: a) Synthetically built b) Built by using data from actual world 11
Figure 2 Kinnect sensor 13
Figure 3 2D images acquired by using a Kinnect sensor: a) Color b) Depth and c)Point Cloud generated from the above images 13
Figure 4 Kd-tree building process 14
Figure 5 Example of a kd-tree including points in the three dimensional space 15
Figure 6 NN Search using a kd-tree data structure 16
Figure 7 Example of normal estimation in one point p (in blue), by using its k-neighborhood (in black) to estimate the tangent plane. 16
Figure 8 a)Normals inconsistently oriented b) Normals consistently oriented towards the viewpoint 17
Figure 9 Cluster representing an individual object 19
Figure 10 Coordinate system 20
Figure 11 Influence region histogram for PFH. Point p_q is red colored whereas points included in the k-neighborhood are blue 22
Figure 12 k-neighborhood used to compute FPFH of a point p_q 23
Figure 13 SHOT computation diagram 23
Figure 14 Signature structure for SHOT 25
Figure 15 a) VFH viewpoint direction component and b) VFH extended FPFH component 25
Figure 16 VFH histogram 26
Figure 17 Three objects and two scenes included in the synthetically built dataset 29
Figure 18 Three objects and two scenes included in the real world capture dataset 30
Figure 19 Descriptiveness assessment process pipeline 31
Figure 20 A transformation is applied to keypoints obtained from the model to find its physical location in the scene using the groundtruth. 32
Figure 21 Matches between the scene and the model object a)In the dataset including synthetically built scenes b)In the dataset that includes actual scenes 33
Figure 22 Example of a noisy scene (b) compared to the original one (a) in the dataset including actual scenes 36

Figure 23 Example of a noisy scene (b) compared to the original one (a) in the dataset including synthetically built scene 37
Figure 24 Descriptiveness assessment of PFHColor and SHOTColor descriptors in a) the dataset including synthetically built scenes b) the dataset containing actual scenes 39
Figure 25 Robustness to Gaussian noise assessment in dataset including synthetically built scenes: a) PRC by using a standard deviation of 0.003, b) PRC by using a standard deviation of 0.005, c) PRC by using a standard deviation of 0.01, d) AUC for the three former levels of Gaussian noise 41
Figure 26 Robustness assessment to Gaussian noise assessment in dataset including scenes built by using real world captured data: a)PRC by using a standard deviation of 0.003, b) PRC by using a standard deviation of 0.005, c) PRC by using a standard deviation of 0.01 d)AUC for the three former levels of Gaussian noise 42
Figure 27 Robustness to Support Radius assessemnt in dataset including synthetically built scenes: a)PRC by using a radius of 5cm, b) PRC by using a radius of 6cm, c) PRC by using a radius of 7cm, d) PRC by using a radius of 9cm, e)AUC for the three former radius 44
Figure 28 Robustness to support radius assessment in dataset including scenes built by using data from actual world: a)PRC by using a radius of 6cm, b) PRC by using a radius of 7cm, c) PRC by using a radius of 9cm, d)AUC for the three former radius values 45
Figure 29 Efficiency of SHOTColor and PFHColor feature descriptors: a)Computing 163 descriptors b)Computing 363 descriptors 47

ACRONYMS

PCL Point Cloud Library

NNR Nearest Neighbour Ratio

AUC Area Under the Curve

PRC Precision Recall Curve

VFH Viewpoint Feature Histogram

PFH Point Feature Histogram

FPFH Fast Point Feature Histogram

SHOT Signature of Histograms of Orientations

RF Reference Frame

RA Reference Axis

CONTENTS

1 INTRODUCTION 8

I FIRST PART: THEORETICAL CONCEPTS 10

2 FEATURES AND OBJECT RECOGNITION BACKGROUND 11
- 2.1 Point Clouds 11
- 2.2 Point Cloud Library (PCL) 12
- 2.3 Data caption and 3D Point Cloud Generation 12
- 2.4 Cloud data processing 14

3 3D FEATURE DESCRIPTORS 18
- 3.1 Classification 18
- 3.2 Computation 19
  - 3.2.1 PFHColor features 19
  - 3.2.2 Fast Point Feature Histogram (FPFH) 22
  - 3.2.3 Unique Signatures of Histograms for Local Surface Description (SHOT) 23
  - 3.2.4 Viewpoint Feature Histogram (VFH) 25

II SECOND PART: EXPERIMENTAL RESULTS 27

4 PERFORMANCE EVALUATION OF 3D LOCAL FEATURES 28
- 4.1 Selection of 3D features to be assessed 28
- 4.2 Selection of descriptor characteristics to be evaluated 28
- 4.3 Performance evaluation 28
  - 4.3.1 Datasets 29
  - 4.3.2 Descriptiveness assessment 30
  - 4.3.3 Robustness assessment 35
  - 4.3.4 Efficiency assessment 38
- 4.4 Results 38
  - 4.4.1 Descriptiveness assessment 38
  - 4.4.2 Robustness assessment 40
  - 4.4.3 Efficiency assessment: 46
- 4.5 Conclusions 47

1

INTRODUCTION

This project focuses on the 3D feature descriptors used for object retrieval. It is critical for robotics and computer vision systems, and poses quite significant challenges to the current technology.

The recent successful commercialization of affordable RGB-D ("D" refers to a "depth" or "distance" channel) sensors such as the Kinnect[4] one is enabling more efficient methods for computer vision objects management. RGB-D sensors obtain the actual depth information, so we can use 3D feature descriptors based on the underlying 3D geometric data. The combination of the appearance information provided by the color RGB channels, and this 3D geometric information based on actual depth information enables an important performance improvement in object recognition systems.

The project goal consists of providing a recommendation to improve the recall rate for the object recognition process, which is the process of identifying objects from images or video clips, being it crucial for a good retrieval process. It will be deployed by describing the options available with the latest available sensors technology and running the required performance evaluation of 3D features descriptors to obtain the data supporting the investigation done.

The former assessment is performed by using images from two different datasets obtained with a low cost RGB-D camera, Kinnect [8]. One dataset includes actual scenes obtained straight from the reality and the groundtruth information, whereas the other one contains synthetically built scenes and the groundtruth obtained during the building process.

The assessed descriptors are SHOTColor and PFHColor, which capture the information of the surface by accumulating parameters in a histogram. SHOTColor stores cosines of the angles between the normals whereas PFHColor accumulates angles representing the difference between those curves.

Aside of this, we have selected the following properties for our assessment: descriptiveness, robustness to radius variation and Gaussian noise, and efficiency.

The implementation has been carried out by using the Point Cloud Library(PCL) and C++ [4].

This project is organized in two parts: a theoretical explanation in Sections 2 and 3, and a practical activity description in Section 4.
The Section 2 provides information about concepts related to 3D features descriptors: point cloud, information capture using RGB-D cameras and basic cloud processing terminology.
The Section 3 explains the 3D feature descriptors classification, as well as the computation procedure.
The Section 4 describes the methodology followed to perform the assessment in detail, the results obtained (in graphics format), and the outcoming assessment and conclusions.

Part I

FIRST PART: THEORETICAL CONCEPTS

2

FEATURES AND OBJECT RECOGNITION BACKGROUND

2.1 POINT CLOUDS

A Point Cloud is a set of data points in the 3D space, where each data point is defined by its x,y and z coordinate values. Each set represents the geometry of an object. There are two ways to generate the values: computer created for synthetic sets (Figure 1a), or capturing data from real objects (Figure 1b).

For each point there is a data channel for the distance between the sensor and the object (depth), and there could be additional channels for color, lightning or viewpoint information.

The p_x, p_y, p_z coordinate values of each point inside a point cloud P are referenced to a fixed triple-coordinate system. In a real object-scene point cloud, the origin of the coordinates system (0,0,0) represents the position of the sensor used to capture the data.

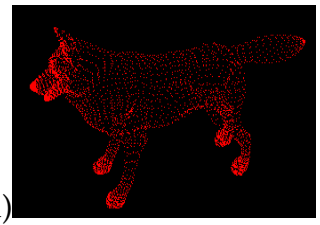
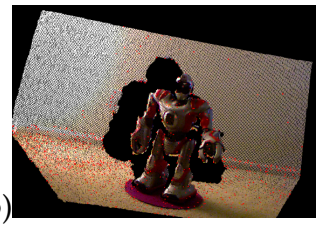
a)  b) 

Figure 1: Examples of Point clouds: a) Synthetically built b) Built by using data from actual world

2.2 POINT CLOUD LIBRARY (PCL)

The Point Cloud Library is a C++ open-source programming library, released under the BSD license, intended for cloud processing and 3D computer vision systems development.

It implements several algorithms to facilitate operations like filtering, feature extraction, surface reconstruction, registration, recognition, features description, model fitting and segmentation.

In addition, it provides support for file formats management and methods to use for performance assessment: PCD file formats, SHOT- Color and PFHColor descriptors, normals estimation, kd-tree structure for Nearest Neighbor search, and point cloud visualization methods.

2.3 DATA CAPTION AND 3D POINT CLOUD GENERATION

There are several options to estimate the depth values required to generate a 3D point cloud depending on the kind of sensor used: stereo- cameras, time-of-flight cameras and structured light sensors.
The selection of the sensor is based on the performance and precision requirements of the application to be used.

RGB-D cameras

RGB-D cameras capture both color and depth information for each pixel.

The RGB-D sensor embeds a camera that captures a 2D color image of the real objects, as well as a depth camera that estimates the distance to the physical scene. The depth camera uses a structured light pattern method to estimate the depth values:

- The infrared transmitter projects the pattern.
- The receiver sensor receives the reflected light
- The distance is calculated from the difference between the projected pattern and the received signal data.

-

In this way we obtain two different 2D images:

- One including the RGB data, which is the base for texture information.
- A second one including the depth data, which provides information about the underlying 3D geometry of the physical object or scene in the actual world.

We combine both of them to generate the joint RGB-D image. Then, the 3D point cloud is built by using the RGB-D data and the predefined calibration parameters of the camera.

In this project Kinnect sensor (Figure 2), a RGB-D kind of device, was used to obtain the available online datasets used for the 3D feature descriptors analysis.
Figure 3 shows an example of the 2D images composing an RGB-D image recorded by using a Kinnect sensor, so you can see the generated point cloud.

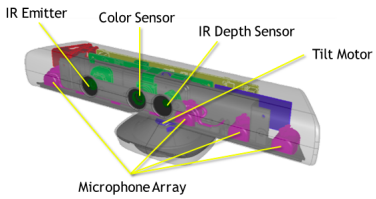


Figure 2: Kinnect sensor

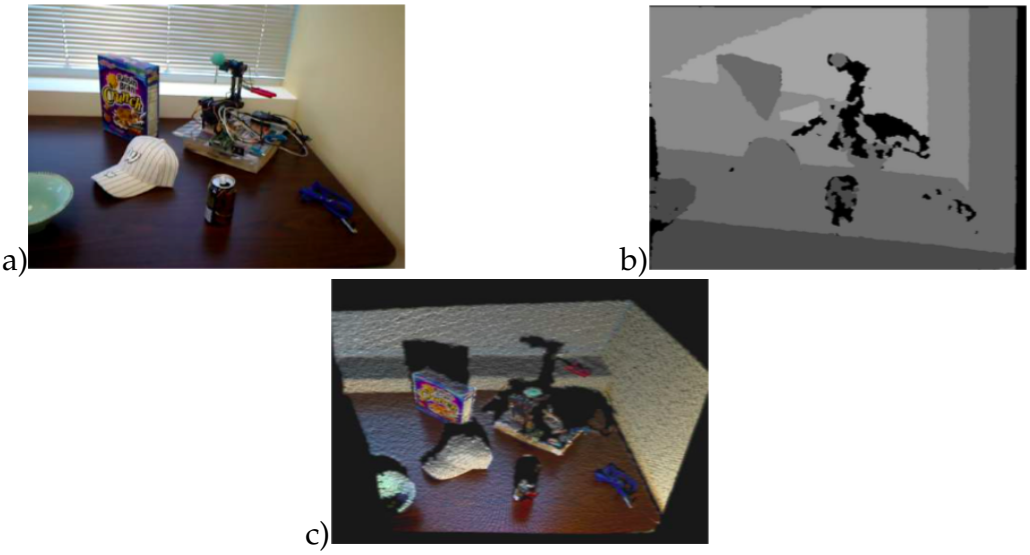


Figure 3: 2D images acquired by using a Kinnect sensor: a) Color b) Depth and c)Point Cloud generated from the above images

2.4 CLOUD DATA PROCESSING

The following sections explain many concepts and methods useful in cloud data processing. They are used throughout the implementation of the performance assessment of 3D feature descriptors.

kd-tree:

A kd-tree (k-dimensional tree) is a space-partitioning data structure for organizing points in a k-dimensional space. It is a generalization of binary search trees since it is implemented as a binary tree where each node is a point in the k-dimensional space [16].

Each node p splits the space in terms of a dimension i, $0 < i <= k$. Then, if the node value is lower than i the component is placed in the left sub-tree, otherwise is placed in the right sub-tree.

This same process is repeated for every dimension.

Figure 4 shows how a kdtree including points in two dimensional space is built.
A node is selected to split the dataset in two groups: points whose x components are lower than the splitting node ones, and another group with the rest of points (Figure 4b).
This dataset splitting process is equivalent to cut the two dimensional space by inserting a hyperplane through one of the points.
In each half space, a kdtree is recursively built by picking some point and inserting a cut line horizontally through it (Figure 4c). Repeating this process, the kdtree obtained is shown in Figure 4d, where the gold node is the root, second level nodes are red, third levels nodes are green, and fourth level nodes are blue.

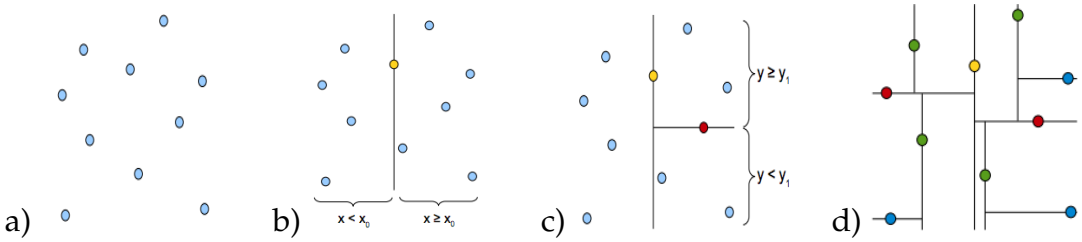


Figure 4: Kd-tree building process

This process can be extended to build a kd-tree out of points in the k-dimensional space . Figure 5 ilustrates how a kd-tree including points in the three dimensional space is built:

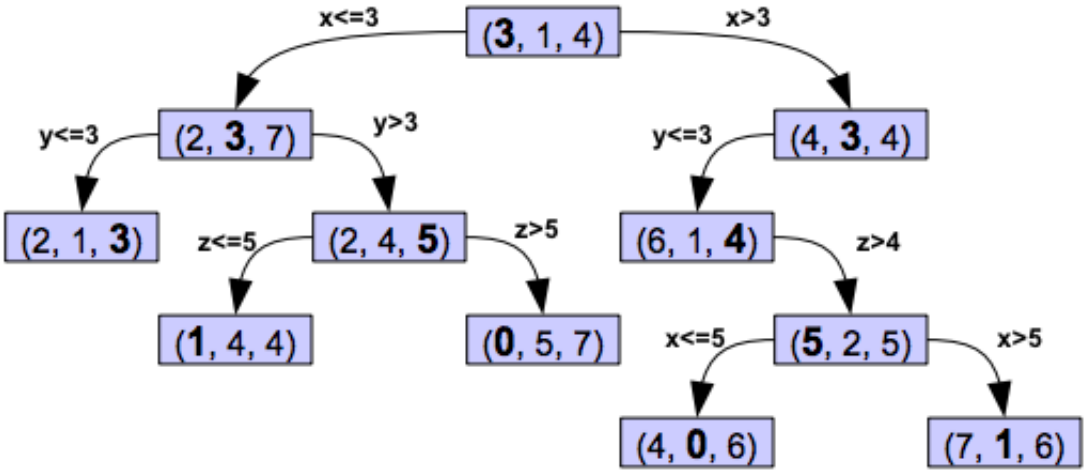


Figure 5: Example of a kd-tree including points in the three dimensional space

Nearest-Neighbor search algorithm for kd-tree structures

The Nearest Neighbor search (NN) algorithm is used to find the point in the tree that is closest to a given query point. This search can be done efficiently by taking advantage of the tree properties to quickly filter large portions of the search space.

The NN search in a kd-tree is performed as follows:

- A point is chosen as a candidate to be the nearest neighbor of the query point p (Figure 6a).
- A hypersphere centered on the query point and passing through the guessed point is built (Figure 6b). If there is a point closer to the query point that our current selection, it must lie in this hypersphere.
- If this hypersphere is fully embedded in one side of a splitting hyperplane, then all the points in the opposite side can be discarded. Obviously, those points are outside the sphere, so they cannot be nearest neighbor candidates. In this way we can quickly remove large branches of the tree
- To determine whether the candidate hypersphere crosses a splitting hyperplane that compares coordinate i, we check whether $|b_i\text{- } a_i| < r$

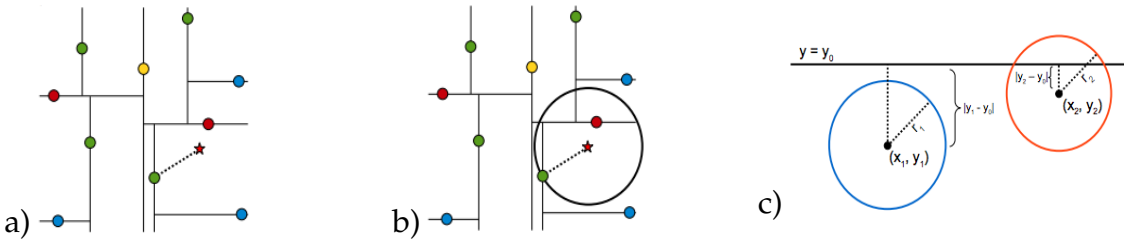


Figure 6: NN Search using a kd-tree data structure

Normal of a point

The normal of a plane is defined as the unitary vector perpendicular to this plane. Extending this concept, the normal vector to a surface is a vector which is perpendicular to the plane tangent to the surface at a given point.

The easiest way to compute the normal in one point is:

- Estimating the tangent plane that best fits the k-neighborhood of p.
- Calculate the normal of the tangent plane.

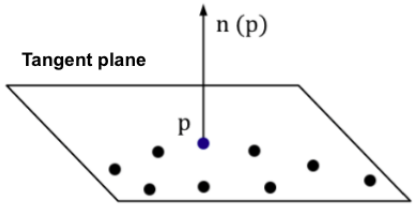


Figure 7: Example of normal estimation in one point p (in blue), by using its k-neighborhood (in black) to estimate the tangent plane.

Based on this, the calculation of normal vectors can be assimilated to the analysis of the eigenvectors and eigenvalues of a covariance matrix created from the nearest neighbors of the query point p.

More specifically, for each point p_i, the covariance matrix (1) is computed as follows:

$$C = \frac{1}{k}\sum_{i=1}^{k} \xi_i (p_i - \bar{p}) \cdot (p_i - \bar{p})^T, C \cdot \vec{v}_j = \lambda_j \cdot \vec{v}_j, j \in 0, i, 2 \quad (1)$$

If the eigenvalues are positive, the first eigenvector $\vec{V}_0$ is the approximation of $\vec{n}$ or $-\vec{n}$.

Unfortunately, the sign of $\vec{n}$ cannot be determined from the analysis of the covariance matrix.
This problem is solved by orientating the normals towards a common viewpoint v_p, that must meet the following condition: $\vec{n_i} \cdot (v_p - p_i) > 0$ for all p_i and its corresponding normals n_i.

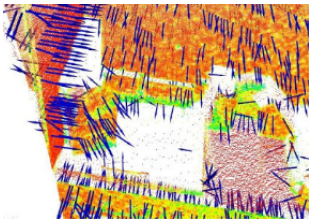
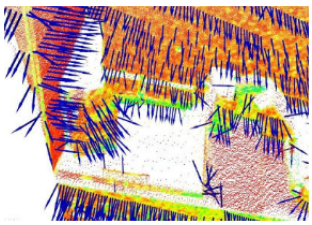
a)  b) 

Figure 8: a)Normals inconsistently oriented b) Normals consistently oriented towards the viewpoint

3

3D FEATURE DESCRIPTORS

3.1 CLASSIFICATION

I propose here two different classification methods of 3D feature descriptors:

First, descriptors can be categorized in two groups regarding their input data: Local descriptors and Global descriptors.

Local descriptors [3] are computed for individual points by using its underlying geometrical information, obtained from the query point k-neighborhood.
They have no information about the object or cluster to be described.
Some examples of local descriptors are Point Feature Histogram (PFH), Fast Point Feature Histogram (FPFH), Radius-Based Surface Descriptor (RSD), 3D Shape Context (3DSC), Unique Shape Context descriptor (USC), Spin Images, Rotation-Invariant Feature Transform (RIFT) and Normal Aligned Radial Feature (NARF).

Global descriptors [3] are not computed for individual points as local descriptors, but they are computed for a cluster of points representing an individual object instead. Hence, they encode the whole object geometry rather than geometrical properties of single points using their k-neighborhood data (Figure 9).
A preprocessing step is required before computing the global descriptors of a given object in order to obtain the cluster that represents the object to be described.
The most popular global descriptors are Viewpoint Feature Histogram (VHF), Ensemble of Shape Functions (ESF), Global Fast Point Feature Histogram (GFPFH).

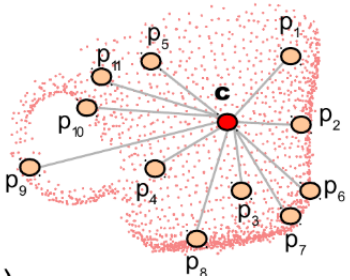


Figure 9: Cluster representing an individual object

Descriptors can also be classified as histogram based descriptors and signature based descriptors.

Histogram-based methods [15] describe the support by building an histogram according to an specific quantized domain, in which geometrical or topological measurements (angles, curvatures, point coordinates...) calculated for individual points are included. Hence, they require the definition of either a Reference Axis (RA) or a local Reference Frame (RF).
As an example, to compute PFH descriptors, for pairs of points in the k-neighborhood of the point of interest, the angles between the normals are computed and included in a general histogram.

The Signature based descriptors [15]describe the support by encoding one or more geometrical measurements computed individually on each point, according to a defined local Reference Frame (RF).

3.2 COMPUTATION

This section describes the computation details of the most widespread 3D feature descriptors, focusing on the ones tested in this project (SHOTColor and PFHColor) but also including other ones widely used (FPFH and VFH).

3.2.1 *PFHColor features*

To simplify the explanation of PFHColor features computation, we will start describing the process without taking into account color information. We will add the implication of color processing later on.

PFH descriptors [12] [3] aim to capture and encode the geometrical properties around a target point by analyzing the difference between the directions of the normal in the k-neighborhood of the point of interest and building multi-dimensional histogram of values.

In its simple way, the computation of a PFH at a point p relies only on the presence of 3D coordinates and estimated surface normals:

- First, the k-neighborhood of p is selected by including the points inside a sphere of r radius centered in point p.
- Second, the angular variations for every pair of points p_s and p_t included in the k-neighborhood are calculated as follows:

  A fixed coordinate system centered in one of the two points is defined as shown in Figure 10. n_s and n_t are the estimated normals computed in the corresponding point p_t and p_t, being p_s the point with a smaller angle between its associated normal and the line connecting the points.

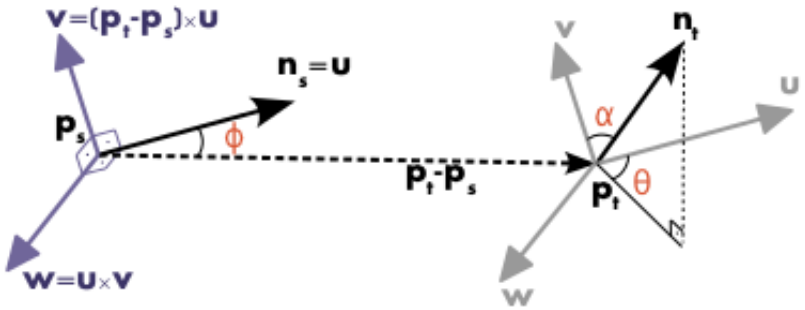


Figure 10: Coordinate system

$$
\begin{aligned}
u &= n_s \\
v &= (p_t - p_s) \times u \\
\omega &= u \times v
\end{aligned}
\tag{2}
$$

Angular variations are finally computed as:

$$\begin{aligned} \alpha &= v \cdot n_t \\ \sigma &= u \cdot \frac{(p_t - p_s))}{||p_t - p_s||} \\ \theta &= \arctan\left(\omega \cdot n_t, u \cdot n_t\right) \end{aligned} \tag{3}$$

- Finally, the binning process is performed as follows.

  For every point, four different features are obtained: three angles (α, σ and θ) and a distance d.
  Each feature's value range is divided into b subdivisions and the number of occurrences in each sub-interval is counted.
  The final histogram encodes in each bin a unique combination of the distinct values for each of the angles, and its dimension (number of bins) can be calculated as:
  $(feature1 \cdot subdivisions1) \cdot (feature2 \cdot subdivisions2) \cdot (feature3 \cdot subdivisions3) \cdot (feature4 \cdot subdivisions4)$
  For instance, in case the number of subdivisions of all four features is the same, the number of bins in the histogram would be equal to the $number\,of\,subintervals^4$.
  Since the assessment of the descriptors performance will be deployed by using code provided by Point Cloud Library (PCL), the number of interval subdivisions per feature used to calculate PFHColor is the one provided by default in this library, this is 5 subdivisions. Hence, the number of bins corresponding to the three angles is $5^3 = 125$ bins.

As mentioned above, the PFH version tested in the performance assessment of the descriptors (PFHColor [1]) includes not only x, y, z information, but also color data channels. This variant includes three additional histograms, one for the ratio between each color channel of p and the corresponding color channel of q. These histograms are binned as the 3 angles of PFH, generating other 125 float values. Adding the 125 bins of the angles and the 125 bins of the three components of color information, the total size of the PFHColor histogram is 250 bins.

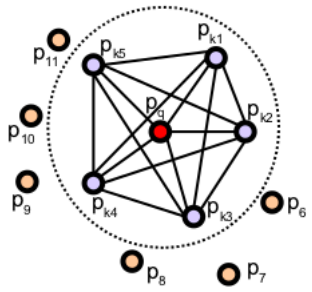


Figure 11: Influence region histogram for PFH. Point p_q is red colored whereas points included in the k-neighborhood are blue

One of the drawbacks of PFHColor is its high computational complexity. For a given point cloud P with n points, its computational complexity is O(nk^2), k standing for the number of neighbors for each point p in P.

3.2.2 *Fast Point Feature Histogram (FPFH)*

FPFH [11] [7] [3] is a simplified version of the PFH described above. It is designed to reduce the computational complexity from O(nk^2) to O(nk) maintaining a similar descriptiveness power.

As explained previously, to obtain the PFH of a given point p_q, the angular variations α, σ and θ are calculated between pairs of points included in a sphere of radius r centered in p_q. This will be called the Simplified Point Feature Histogram (SPFH) of the point p.

To compute FPFH, the SPFH is calculated for every point p_k included in the k-neighborhood or region of interest of the given point p_q.

After calculating the SPFH for every point p_k, the FPFH is computed for the given point p_q by using the SPFH values of its neighbors as a weighting value.

$$FPFH(p_q) = SPFH(p_q) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{wk}SPFH(p_k)$$

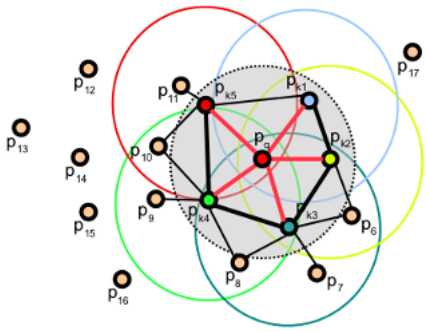


Figure 12: k-neighborhood used to compute FPFH of a point p_q

3.2.3 *Unique Signatures of Histograms for Local Surface Description (SHOT)*

After many performance evaluations of existing 3D features descriptors, most authors agree on that one of the most challenging issue of the existing 3D features descriptors is the definition of a single, unambiguous and stable local coordinate or Local Reference Frame at each point.

To overcome this issue, the SHOT descriptor [15] was released as a new 3D feature descriptor using a new coordinate system.
It includes a proper combination of Signatures and Histograms.

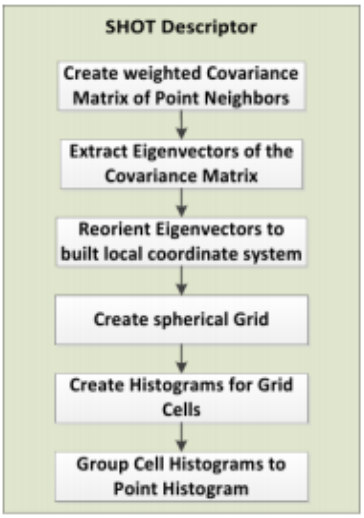


Figure 13: SHOT computation diagram

Its computation includes four steps clearly differentiated (Figure 13): Computation of a coordinate system or RF at the interest point p, creation of a spherical grid, creation of histograms for grid cells and group cell histograms to point histograms.
During the computation of the coordinate system, the weighted matrix of covariance C (4) is computed by using the k neighbors p_i of the interest point p included in the sphere of radius r.

$$\mathrm{C}=\frac{1}{n}\cdot\sum_{i=1}^{n}\left(r-||p_i-p||\right)\cdot\left(p_i-p\right)\cdot\left(p_i-p\right)^T \quad (4)$$

The three eigenvectors that define the coordinate system of point p are the result of the eigenvalues decomposition of C matrix.
The eigenvectors λ_1, λ_2 and λ_3 are sorted by the value of its corresponding eigenvalues $v1, v2$ and $v3$, which represent x-axis, y-axis and z-axis.
The direction of the x-axis an z-axis is determined by the orientation of the vectors from p to the neighboring points p_i, and its cross product determine the direction of y-axis.

Using this local coordinate system, the spatial environment of p is divided with an isotropic spherical grid (Figure 14).
For each point p_i in a cell of the divided sphere the angle $i = p_i \cdot p$ is computed between the points normal and the normal of p. The local distribution of angles is subsequently described by one local histogram for each cell.
In every cell, a histogram is computed including the local distribution of angles of this cell.
The resulting final histogram dimension (number of bins) is given by $k \cdot b$, where k is the number of grills in the sphere and b is the number of bins of each local histogram. These values are normalized to sum to one in order to handle different point densities in different point clouds.

The SHOT descriptor version implemented in this project uses both 3D coordinate information and color information as well, so we need to compute two histograms for each cell in the sphere: one for the angle between normal vectors, and another for the sum of the absolute difference of RGB values between points p_i and p.

SHOTColor descriptor implemented in PCL libry has a size of 1344 bins.

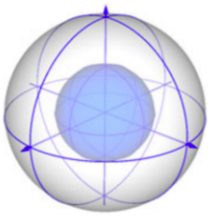


Figure 14: Signature structure for SHOT

3.2.4 *Viewpoint Feature Histogram (VFH)*

The VFH [3] is based in FPFH.

VFH includes a viewpoint direction component and an extended FPFH component (Figure 15 and 16).

To compute the viewpoint direction component, the object's centroid is found by averaging the X, Y and Z coordinates of all the points in the cloud. Then, the vector between this centroid and the viewpoint, given by the position of the sensor, is computed and normalized. Finally, the angle between this vector translated to each point and the normal of the points included in the cluster is calculated and binned into an histogram. Translating the vector makes the descriptor scale invariant.

To compute the second component we use the same algorithm than FPFH. The only difference is that the component is only computed for the centroid using the viewpoint direction vector as the normal of the centroid, and all the points inside the clutter are considered as neighbors.

The histograms obtained are concatenated to build the VFH descriptor. The bins are normalized by default using the number of points in the cluster.

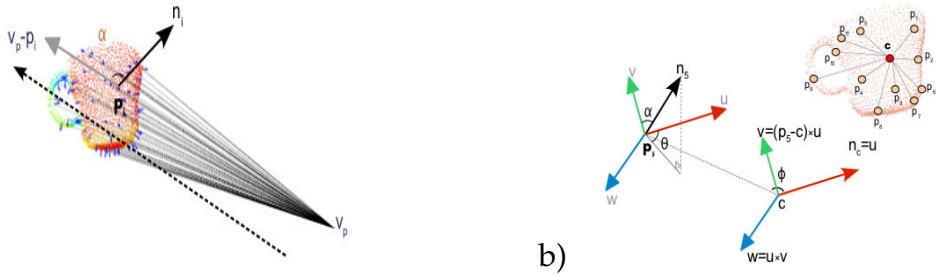


Figure 15: a) VFH viewpoint direction component and b) VFH extended FPFH component

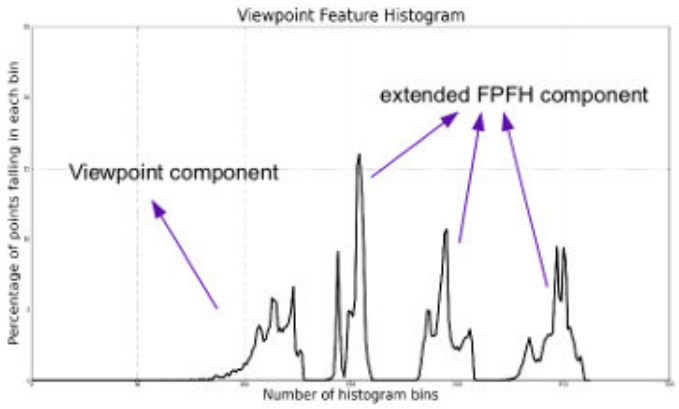


Figure 16: VFH histogram

Part II

SECOND PART: EXPERIMENTAL RESULTS

4

PERFORMANCE EVALUATION OF 3D LOCAL FEATURES

4.1 SELECTION OF 3D FEATURES TO BE ASSESSED

In this section there is a description of the assessment done based on the two most widely used 3D feature descriptors: SHOTColor descriptor and PFHColor descriptor.

Most of 3D performance developments are not including the color data when evaluating the SHOT and PHF descriptors (i.e. [6], [14]) , but in this project even this relevant data will be considered for sake of completeness.

4.2 SELECTION OF DESCRIPTOR CHARACTERISTICS TO BE EVALUATED

3D feature descriptors are going to be evaluated in this project in terms of three different characteristics: descriptiveness, efficiency and robustness, according to previously developed descriptors performance assessments [6] [14] [10] [13] [15].

4.3 PERFORMANCE EVALUATION

The performance evaluation is going to be carried out by using the Point Cloud Library (PCL) and C++.
The Point Cloud Library provides an implementation instance of the descriptors to be evaluated as well as many other useful methods.

4.3.1 *Datasets*

Two online datasets are used as input data in this assessment.
Both datasets include 3D point clouds built from data acquired by a Kinnect camera, which is not very high quality in regular conditions. That is why, to overcome this limitation, the kinnect camera was positioned as close as possible to the objects and in a controlled indoor environment during the data gathering process.

The first data set [5][9] consists of 5 models of individual objects, as well as 5 different scenes. I built these scenes synthetically, and they include instances of the 5 objects distorted by a rigid transformation (translation or rotation).
The groundtruth, rigid transformation between a model and its distance in the scene, is known in advance since it was obtained during the scene building process.

The second dataset [2] also includes 5 models and 5 scenes. In this case, the 5 scenes were acquired directly from actual scenes rather than being synthetically built. They contain both clutter and occlusion.
The groundtruth used to assess the performance is attached in the dataset.

Figure 17: Three objects and two scenes included in the synthetically built dataset

Figure 18: Three objects and two scenes included in the real world capture dataset

4.3.2 *Descriptiveness assessment*

One of the properties of 3D feature descriptors to be tested in order to assess their performance is their descriptiveness.

Descriptiveness stands for the capability of the descriptor to compact the geometrical information around the point of interest p.

Procedure:

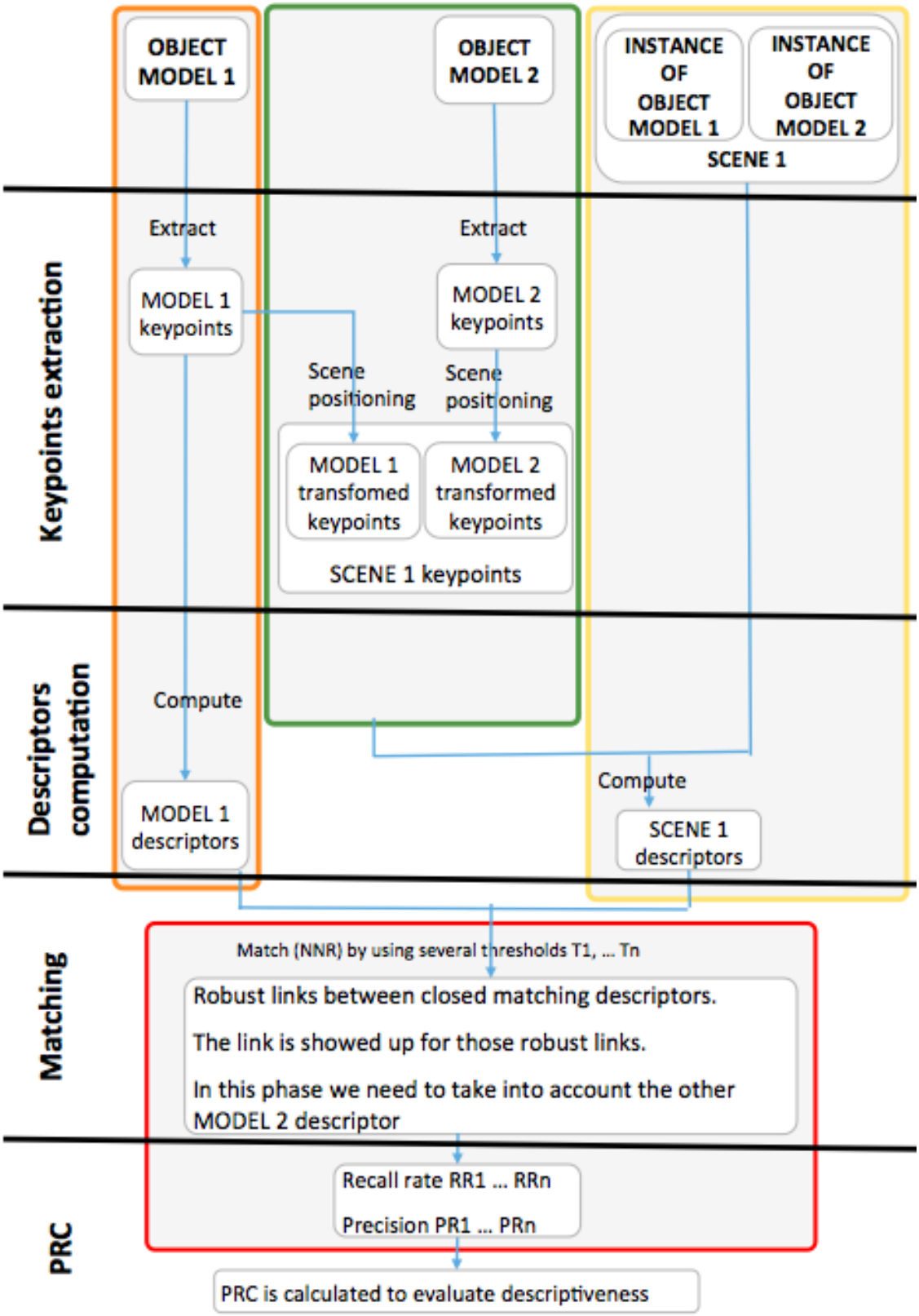


Figure 19: Descriptiveness assessment process pipeline

I will describe the main steps required to the descriptiveness evaluation. For sake of simplification, I will explain a simple concrete case, and I will make a general case later on.

That specific case is the one shown in the scheme 19 above: scope is limited to a unique scene (SCENE1) containing 2 models (MODEL1 and MODEL2), matching flow is described for one model (MODEL1).

This process involves four phases:

The first phase (shown in Figure 19 in orange), consists of extracting the keypoints of the 3D point cloud corresponding to the OBJECT MODEL 1, and afterwards computing the descriptors of those keypoints.

The second phase (shown in Figure 19 in green), consists of placing the keypoints in the scene by applying a transformation to the keypoints extracted from the 3D point cloud of the two model objects that have an instance in the scene, by using the groundtruth information. This particular scene only contains two INSTANCES, one corresponding to MODEL1 and the other one corresponding to MODEL2.

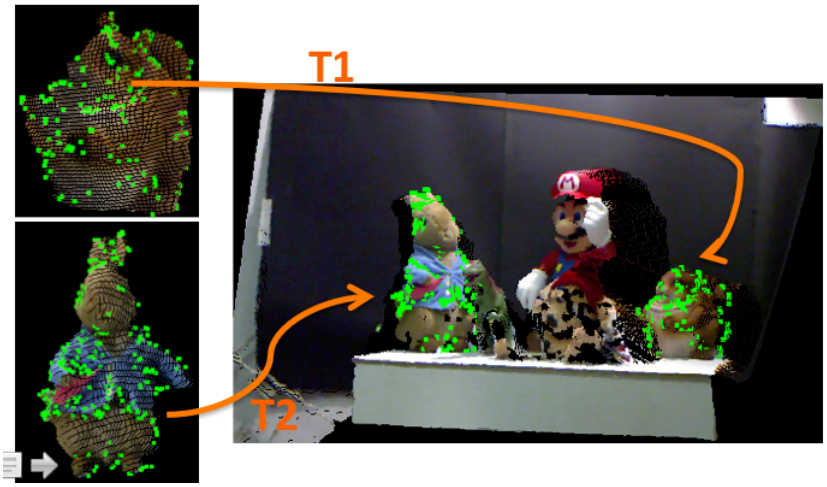


Figure 20: A transformation is applied to keypoints obtained from the model to find its physical location in the scene using the groundtruth.

The third phase (shown in Figure 19 in yellow) involves the computation of the descriptors in the scene, which is done by using the scene 3D point cloud and the keypoints extracted from the scene during the second phase.

Now that we have the descriptors from the model and the scene, we can move to the fifth phase (shown in Figure 19 in red), where those descriptors go through an

intensive matching procedure.
A match is represented by a line displayed in the screen connecting one descriptor from the scene with one descriptor of the MODEL1. Look at the example in Figure 21).

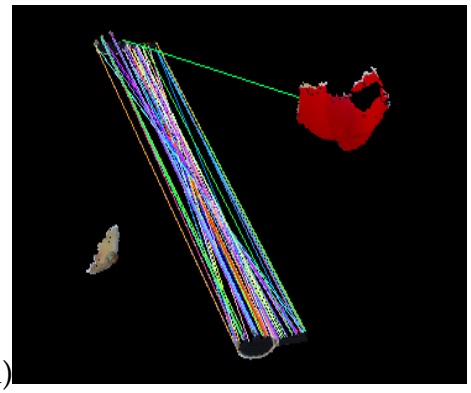
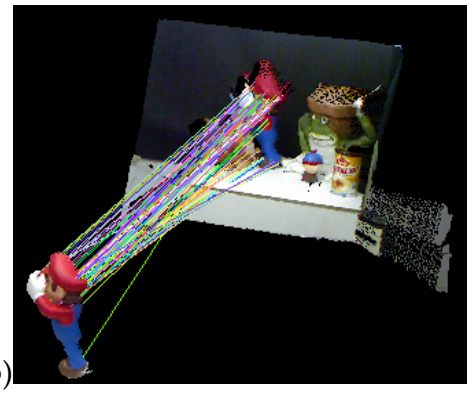
a)  b) 

Figure 21: Matches between the scene and the model object a)In the dataset including synthetically built scenes b)In the dataset that includes actual scenes

First, for each descriptor in the scene, we look for the descriptor in MODEL 1 having the closest value called its nearest neighbor. This search will be done by using a kdtree structure in order to reduce the computational load of the process.
Second, select which of those items will be displayed in the screen, these are called "robust matches". Only a robust matches are displayed in the screen. To decide whether the match between the two descriptors is robust or not, we use Nearest Neighbor Ratio (NNR) (similarly to [6] and [14]).
NNR regards the match as robust if the condition $||N_S - N_M||/||N_S - N'_M|| <= \tau$ is met, where N_S is the descriptor in the scene and N_M and N'_M are its nearest and second nearest neighbor in the MODEL1.
In this phase we also must take into account MODEL2, but this will be explained later to simplify this explanation.

At this moment, for each value of T we compute the recall and precision values:For a fixed value of τ, we can obtain the values of the recall and precision, defined as shown below:

$$Recall = \frac{correct\ matches}{correspondences} \tag{5}$$

$$Precision = \frac{correct\ matches}{total\ robust\ matches} \tag{6}$$

The number of correspondences of an object model is its number of keypoints.
A match is regarded as correct when it matches a object in the dataset with its correct instance in the scene.

Repeating this same calculation for different values of τ let us obtain pairs of values of recall and precision, that we will use to built a curve, called PRC. This curve lets us evaluate how descriptive is a 3D feature descriptor.

Once completed this matching process for Model1, the same procedure is applied to MODEL2. We will repeat the same process but exchanging MODEL1 by MODEL2 in the scheme.

Now we will move to next complexity level to explain how the matching between the scene and the MODEL1 and the matching between the scene and the MODEL2 influence each other.

A descriptor from the scene may match to descriptors of more than one model object, so we could have two different cases:

- A descriptor of the scene matches to a descriptor of one model. In this case the match is robust and displayed.
- A descriptor in the scene matches several descriptors, each one from different models. In this case, only the match between the descriptor in the scene and the closest model descriptor is kept. We must also check whether this selected match is robust, and therefore displayed, or not. To be considered robust, it must met the condition: $||N_S - N_{M1}||/||N_S - N_{M2}|| <= \tau$, where N_{M1} is the selected match (corresponding to MODEL1) and N_{M2} is the nearest neighbour of the descriptor in the scene in MODEL2.

Given the explanation above, one descriptor in the scene can be matched only once with a descriptor of a single model. As a consequence, the sum of the robust matches after performing the NNR technique to all the single model objects and the scene can not exceed the number of keypoints in the scene.

The recall and precision values are also calculated taken into account the two models, this is:

$$\begin{aligned} \textit{correct matches} = \textit{correct matches between scene and model}1+ \\ \textit{correct matches between scene and model}2 \end{aligned}$$

$$total\ matches = total\ matches\ between\ scene\ and\ model1 + total\ matchesbetween\ scene\ and\ model2$$

$$correspondences = correspondences\ between\ scene\ and\ model1 + correspondences\ between\ scene\ and\ model2$$

Finally, we are going to complete our description by extending this specific two-models case to generic N-models process.
In the N-models case, the keypoints in the scene will be obtained by transforming the keypoints extracted from n models rather than only two. Similarly, the process shown in Figure 19 will be repeated n times, one for each model. If there are several scenes, you must repeat the process for each one. In this project we used several scenes, and the final result were obtained as the average of each scene score.

Parameters:
We use a 3cm radius value to calculate the normals required to compute the descriptors.
In this same descriptors calculation the radius value is set to 7cm. This value was chosen according to the dimensions of the objects included in both datasets.
Finally, a subsampling filter is used to reduce the number of points in the 3D point cloud of the models and scene, so that the computational complexity is lower. The leaf of this subsampling filter is 1cm.

4.3.3 *Robustness assessment*

Another important property to test is the robustness of the 3D feature descriptor when there are sources of interferences that may affect their performance: Gaussian noise and the support radius.

Robustness to Gaussian noise impact

Both descriptors are assessed in terms of robustness against Gaussian noise, which is the kind of noise that affects the most the performance of local descriptors.
There are other types of noise affecting the performance of descriptors, such as the Impulse noise. Though, they are not evaluated in this project since they can be easily removed by using a smoothing filter and the impact on local descriptors performance is negligible compared to the Gaussian noise effect.

This robustness assessment is performed in similarly to [6] and [14], by using PRC and Area Under the Curve (AUC).
Area Under the Curve (AUC) is used to compact the information in the PRC curves, so that we can make a better overall analysis, similarly to [6]. The higher the AUC value, the better the performance of 3D feature descriptors, since high recall and precision values lead to larger AUC values.

Procedure:

In each dataset, different levels of Gaussian noise are added to the x, y and z coordinates of the points included in the scenes. This is done by setting the standard deviation of the added noise up to three different values: 0.003, 0.005 and 0.01.
Look at figures Figures 22 and 23 for examples of noisy scenes.

Keypoints from the model are extracted. Then, the keypoints in the noisy scene are placed by applying a transformation to keypoints extracted of the models included in the scene, as explained above in the descriptiveness assessment.

Descriptors are computed from previously extracted keypoints from both the model and the noisy scene.

Finally, the matching between the descriptors of the model and the noisy scene is performed by using NNR with kd-tree and nearestKSearch method.

For each level of noise, a PRC and its AUC is computed.

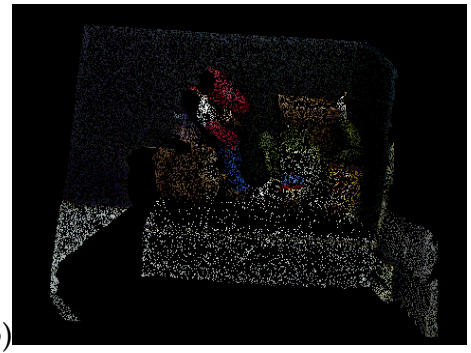
a)  b) 

Figure 22: Example of a noisy scene (b) compared to the original one (a) in the dataset including actual scenes

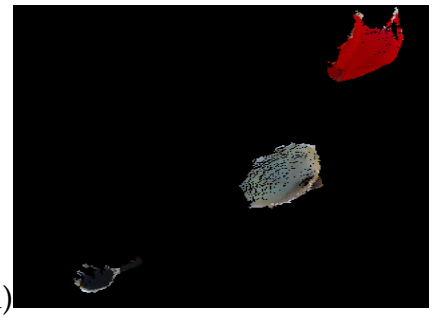
a)  b) 

Figure 23: Example of a noisy scene (b) compared to the original one (a) in the dataset including synthetically built scene

Parameters:

The radius used to calculate the normal vectors involved in the descriptors computation is set to 3cm in both datasets.
The support radius is also set to the same value for testing the SHOTColor and PFHColor descriptors in both datasets: 6 cm.
Since after adding that noise the number of points of the point cloud is almost twice the original one, the sub-sampling filter applied in both datasets has a leaf of 1.2 cm.

Robustness to support radius variation impact:

The robustness assessment also includes testing how robust are descriptors against the support radius variations when computing the local descriptor.

As it was explained in the Section 3, the support radius defines the k-neighborhood considered to calculate the descriptor of a given point p.The bigger the support radius, the larger the number of neighbors of p taken into account to compute its descriptor.

Procedure:

Calculate the descriptiveness for different values of the support radius, computing the PRC and its AUC for each value.

Create a graphical chart of the AUC values depending on the support radius values.

Parameters:

In this assessment, the radius used to compute the normals is set to 3cm.
The support radius varies between 5cm and 9cm in the dataset including syntheti-

cally built scenes and between 6cm and 9cm in the dataset including actual scenes. The range evaluated is higher in the case of the dataset including synthetically built scenes because the results were not enough conclusive for a lower range.The maximum radius tested is set to 9 cm because the models included in the scene have a radius of 1 dm approximately.
The sampling filter has a leaf of 1cm.

4.3.4 *Efficiency assessment*

We also need to test the efficiency of the descriptors, because the more complex calculations the longer time and CPU power is required. It is critical to have a good balance between performance accuracy and computing workload.

Efficiency is defined as the average time necessary to compute the descriptors of a certain number of keypoints depending on the value of the support radius used by the descriptor being tested.

Procedure:

Calculate the average of the time required to build the data of a fix number of descriptors for a benchmark of point clouds obtained from different objects. The test will be run for different number of keypoints and averaging the results obtained for three different object instances.

4.4 RESULTS

4.4.1 *Descriptiveness assessment*

In this section, PRC is used to evaluate the descriptiveness of SHOTColor and PFHColor descriptors in the two different datasets presented in the Section 4.3.1, the first one with synthetically built scenes and the second one with data from actual scenes.

Dataset 1:

Since this dataset was synthetically built, neither occlusion nor clutter is included. Therefore, the performance of the tested descriptors in it is better, than for the dataset including real scenes.

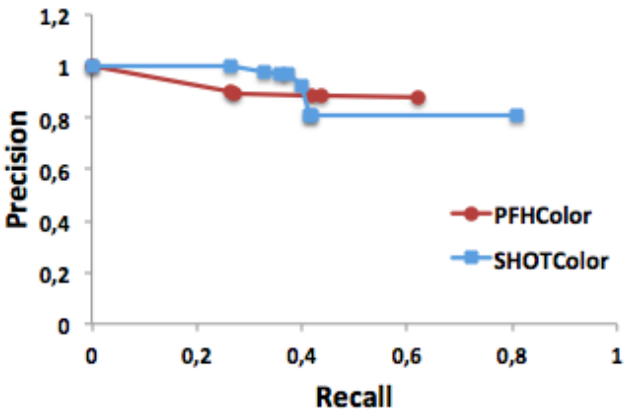
As it can be seen in Figure 24a, SHOTColor feature descriptor is more descriptive than PFHColor feature descriptor because the first one achieves a higher recall rate and keeps higher precision values than the second one. That is, SHOTColor obtains a higher absolute number of correct matches in average for the objects in the scene, as well as a higher percentage of correct matches out of the total number of matches considered robust.

Dataset 2:

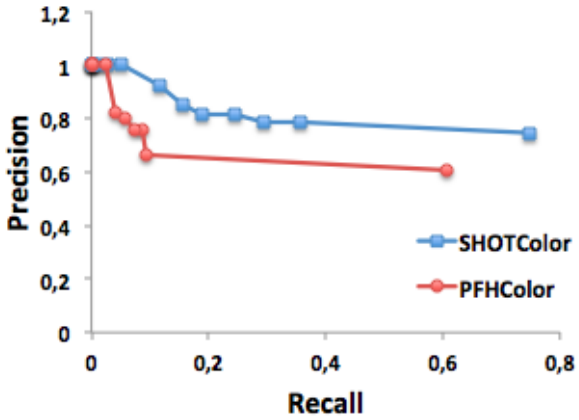
This dataset includes 5 object models and 5 scenes built by using data obtained straight from scenes in the actual world. Hence, clutter and occlusion exist in this dataset.

Figure 24b shows the PRC results of both descriptors in this dataset.
SHOTColor achieves a higher recall and precision rates as in the dataset 1.

It is important to highlight that alhought the SHOTColor descriptor outperforms the PFHColor one in both datasets, its descriptiveness gap is significantly smaller in this second dataset compared to the first one. That is, the SHOTColor descriptor is more sensitive to occlusion and clutter than the PFHColor one.



a)



b)

Figure 24: Descriptiveness assessment of PFHColor and SHOTColor descriptors in a) the dataset including synthetically built scenes b) the dataset containing actual scenes

4.4.2 *Robustness assessment*

Robustness to Gaussian noise impact:

As it was explained in the Section 4.3.3, to evaluate the robustness against Gaussian noise, we compute a PRC for each level of noise.

This comparison method is consistent since the support is common for the PRC. That is, the recall and precision values are changing when τ value is modified in a 0-1 range. The curves are parametrized.

Dataset 1:

Figures 25a, 25b, 25c show that SHOTColor descriptors perform better than PFHColor descriptor for every level of Gaussian noise added.
In the case of PFHColor descriptors, the precision worsen rapidly as the standard deviation of the Gaussian noise increases, whereas its recall is only sensible to high noise levels only.
In the case of SHOTColor descriptors, precision and recall are only affected when high levels of noise are added.

Focusing on the overall analysis, Figure 25d shows the normalized AUC results of these descriptors with respect to different levels of Gaussian noise obtained by varying its standards deviations.
The outcome here is that SHOTColor widely outperforms PFHColor in terms of robustness to Gaussian noise. When the standard deviation increases, the AUC values of PFHColor decrease much faster than the SHOTColor ones.

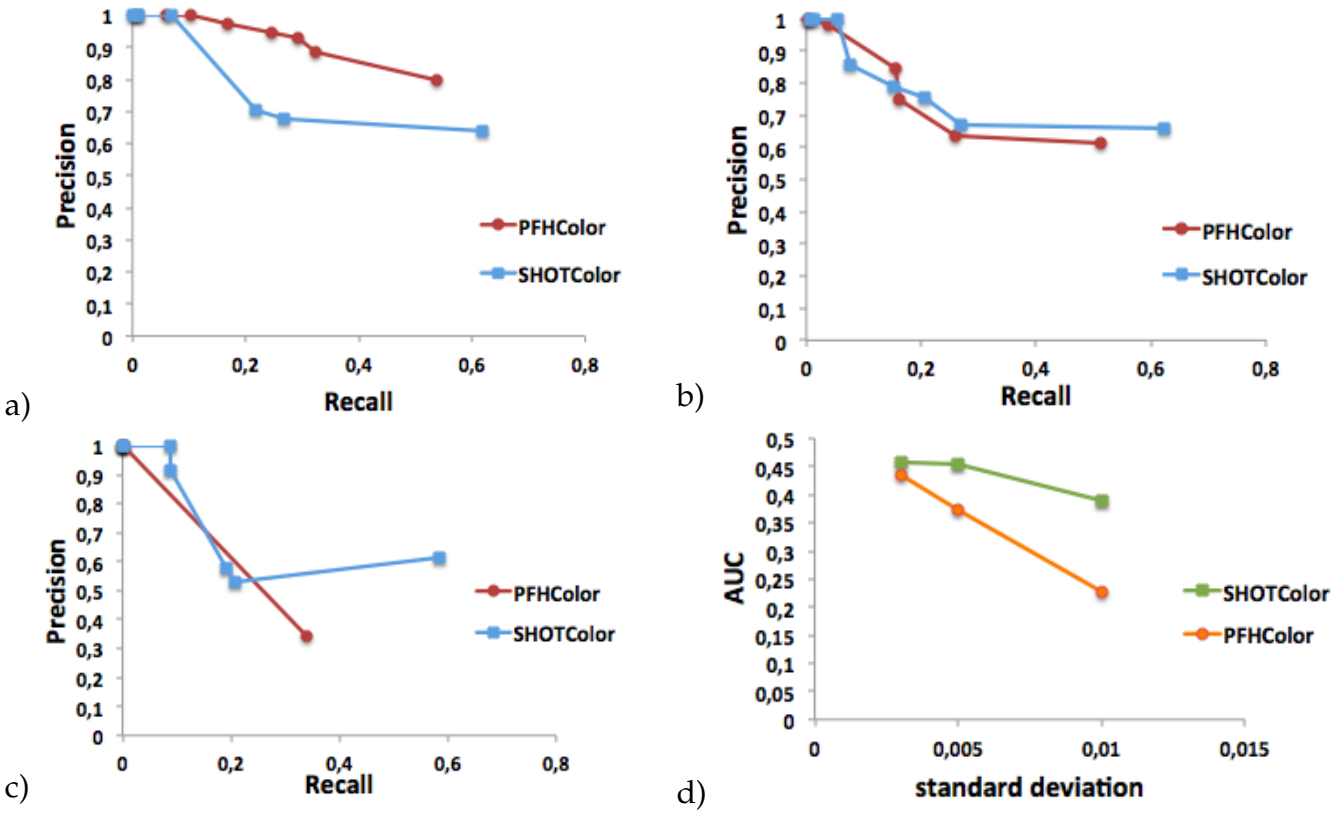


Figure 25: Robustness to Gaussian noise assessment in dataset including synthetically built scenes: a) PRC by using a standard deviation of 0.003, b) PRC by using a standard deviation of 0.005, c) PRC by using a standard deviation of 0.01, d) AUC for the three former levels of Gaussian noise

Dataset 2:

Again here the SHOTColor feature descriptors are more robust against Gaussian noise than PFHColor descriptors.

For low noise levels (Figure 26a), SHOTColor and PFHColor descriptors behave in a similar way.
We also observe that when the noise level increases (Figure 26b and 26c), PFHColor performance suffer an steep dropping caused mostly by the decrement in its recall values.

We can make the following assessment according to the results showed in the Figure 26d: the overall performance of both descriptors is degrading faster than in the dataset 1. This fact can be observed in the SHOTColor descriptors: in dataset 1 the SHOTColor descriptors AUC values keep almost constant for lower levels of noise,

whereas for this dataset the AUC values drop significantly even for very small levels. This is caused by the presence of clutter and occlusion data in dataset 2. Therefore, we can conclude that SHOTColor descriptors are less robust against Gaussian noise when those items are included.
Additionally, it is also observed that PFHColor descriptors robustness to Gaussian noise decreases faster for this dataset. But on the other side, the robustness of this descriptor is not degrading so quickly as noise increases.

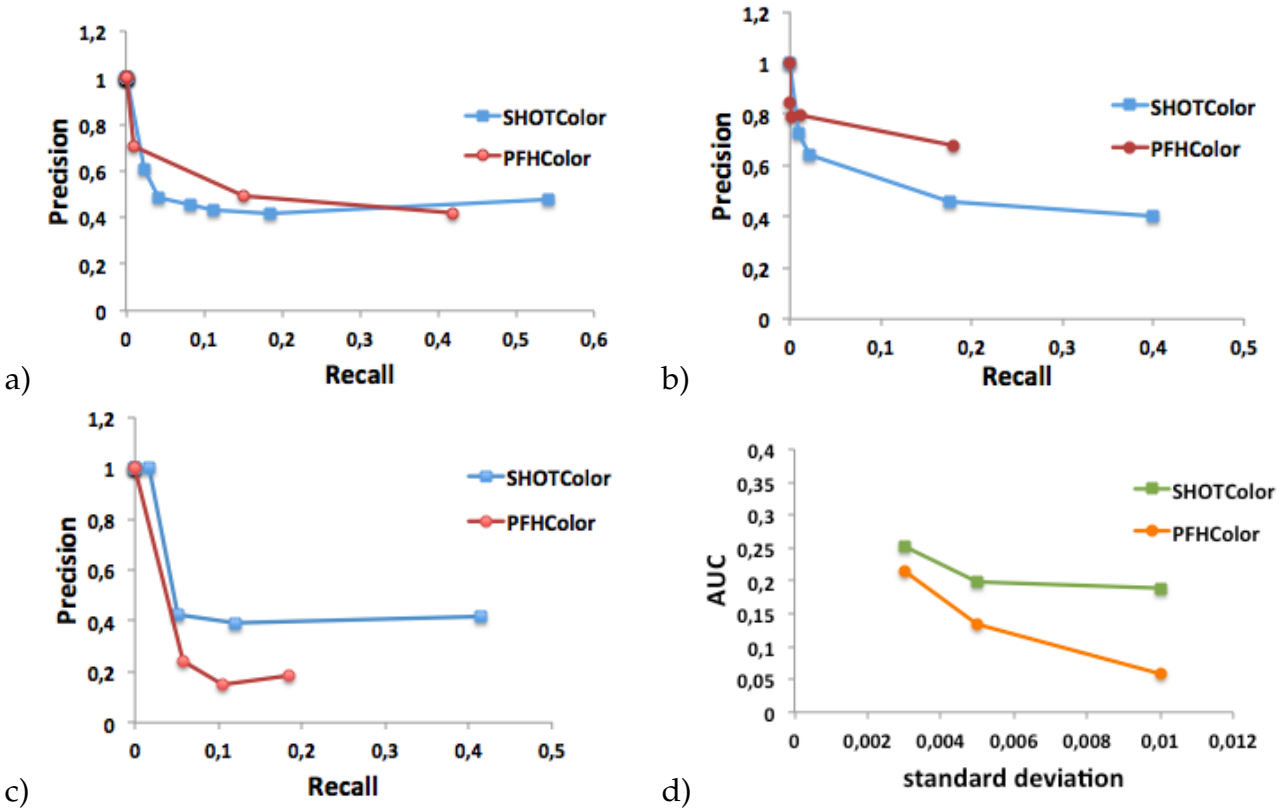


Figure 26: Robustness assessment to Gaussian noise assessment in dataset including scenes built by using real world captured data: a)PRC by using a standard deviation of 0.003, b) PRC by using a standard deviation of 0.005, c) PRC by using a standard deviation of 0.01 d)AUC for the three former levels of Gaussian noise

Robustness to support radius variation:

The AUC is also used when comparing the evolution of the performance of SHOTColor and PHFColor descriptors as the support radius changes, since it compacts the information of PRC obtained for each radius.

On the one hand, the bigger the support radius, the higher the number of neighbors of p taken into account to calculate the descriptor. Hence, a large radius lets descriptors compact more information of the surface, leading to a higher descriptiveness. On the other hand, descriptors using big support radius are less robust against clutter and occlusion.

Dataset 1:

In the dataset including synthetically built scenes, as Figure 27e shows, the performance of both descriptors increases as the support radius increases.
Since in those scenes there is neither occlusion nor clutter, higher radius higher radius values implies compacting a larger amount of information of the underlying local surface. This increases the descriptive power of both descriptors..

Aside of this we found that SHOTColor descriptors perform better than PFHColor descriptors as it can be seen in Figures 27a, 27b, 27c and 27d.

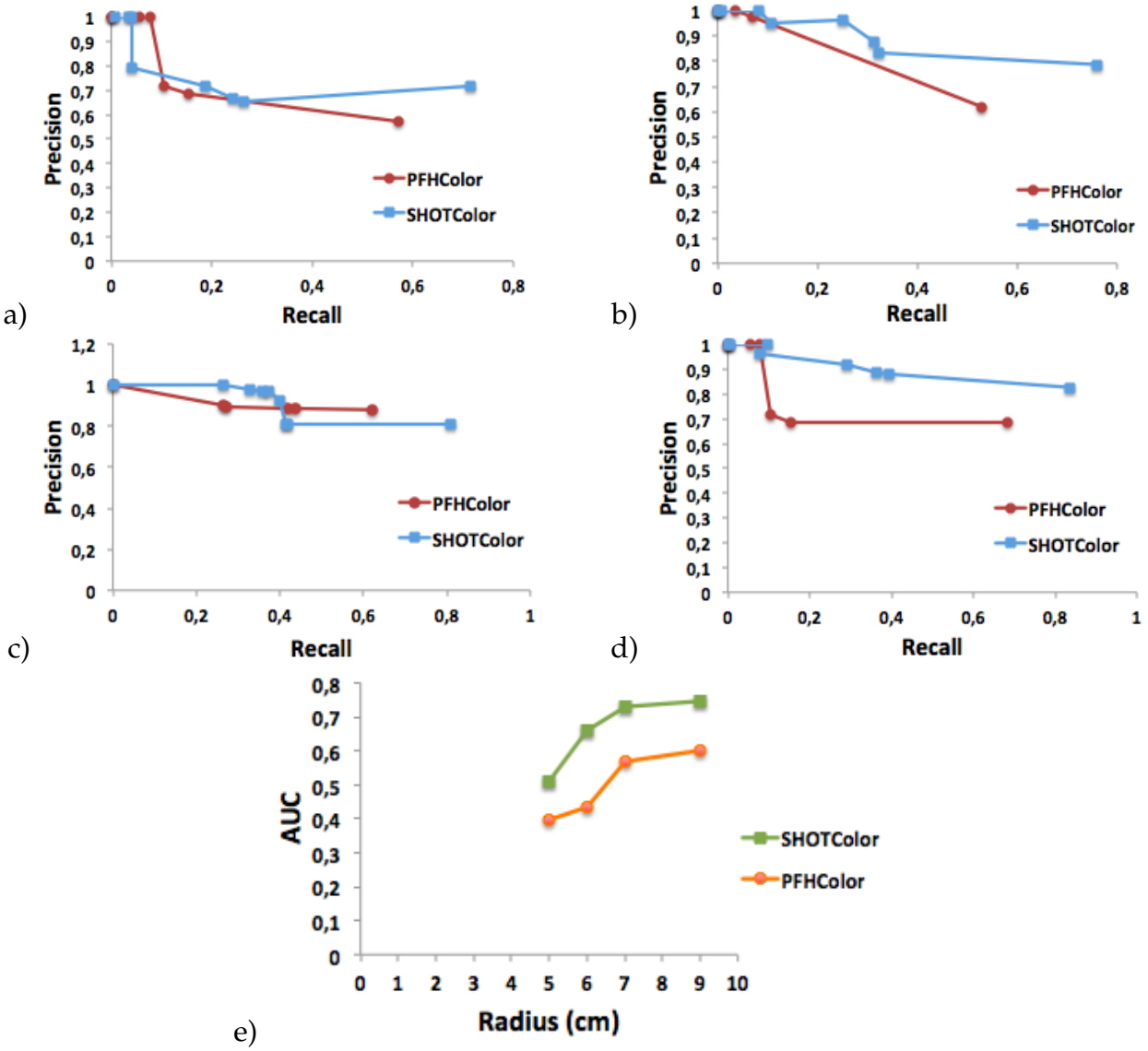


Figure 27: Robustness to Support Radius assessemnt in dataset including synthetically built scenes: a)PRC by using a radius of 5cm, b) PRC by using a radius of 6cm, c) PRC by using a radius of 7cm, d) PRC by using a radius of 9cm, e)AUC for the three former radius

Dataset 2:

In the dataset including actual scenes, the performance of the descriptors decreases consistently as the radius increases from an initial value of 6 cm to a maximum value of 9 cm.

Since the scenes included in this dataset are subjected to clutter and occlusion, high radius values may lead to a worse performance of the descriptors. That is, increasing the support radius means that the descriptors placed in the model boundaries encodes information of the associated surface but also the surface information of adjacent. In short, the wider the scope the more chances to mix with surrounding objects data.

As Figure 28d shows, SHOTColor descriptors performance improves when using radius up to 7cm, compensating the damaging effects of the clutter and occlusion. But for higher radius values (9cm), these effects overcome the positive impact of the radius increase.
Although this is not represented in the Figure 28, the behavior will be similar for PFHColor, but the data peak will happen below 6cm. This means PFHColor descriptors are less robust against support radius variations in this dataset.

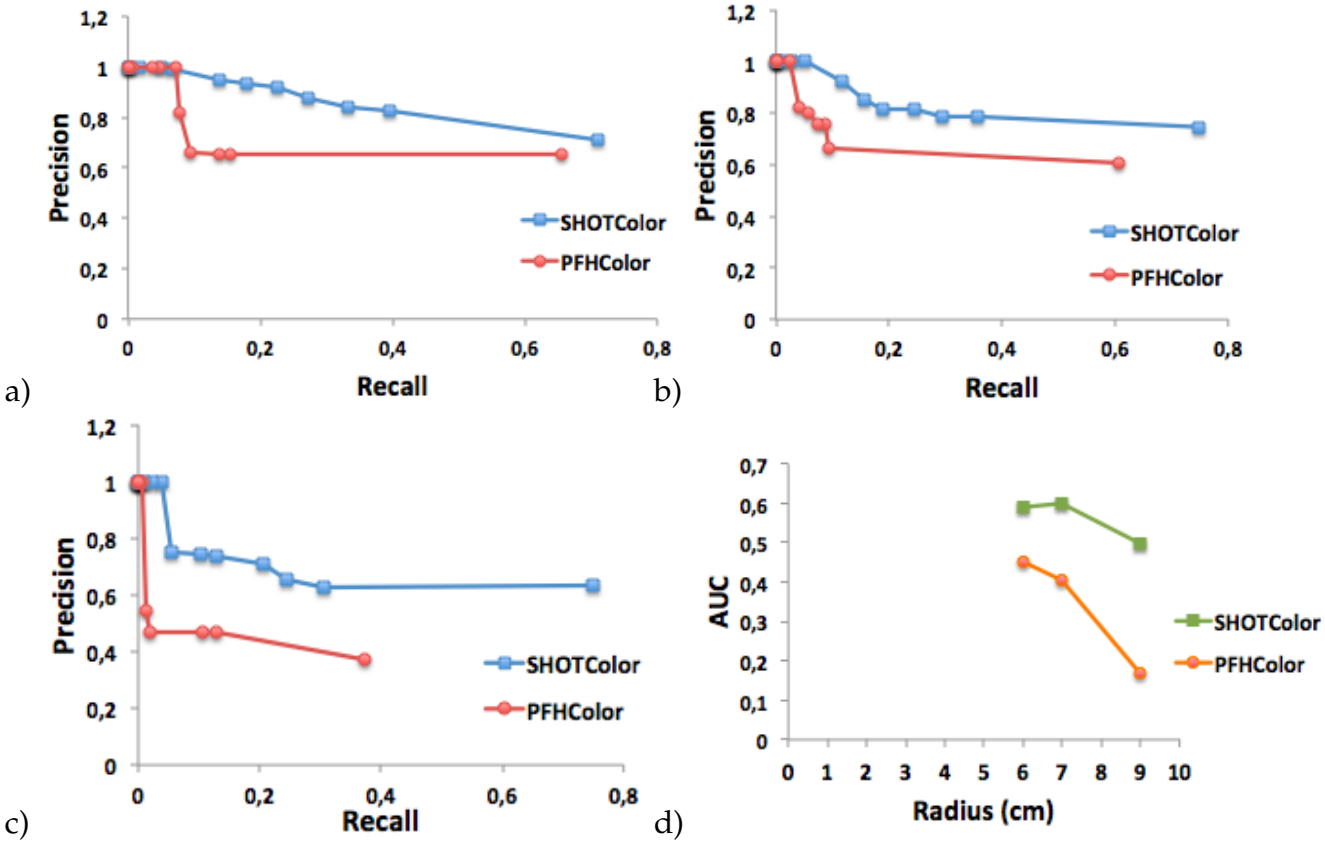


Figure 28: Robustness to support radius assessment in dataset including scenes built by using data from actual world: a)PRC by using a radius of 6cm, b) PRC by using a radius of 7cm, c) PRC by using a radius of 9cm, d)AUC for the three former radius values

4.4.3 *Efficiency assessment:*

The figures 29a and 29b show the relationship between the descriptors computation time and the support radius, for two sets with different range of descriptors: 163 and 363. These values were selected by setting the minimum contrast and number of octaves required to compute the SIFT3D operation: 1 and 8 for the 163 set, 0.5 and 13 for the 363 one. Those values were chosen taking into account the compuational load.

The execution time of SHOTColor descriptor increases linearly as the radius value goes up, whereas PFHColor computation time increases exponentially for a similar ramp.

We can also observe that the time required to compute the descriptors grows faster as the radius increases, for a larger number of points. This trend is strongly visible for PFHColor descriptors.

Hence, it's clear that SHOTColor descriptors are more efficient than PFHColor ones.

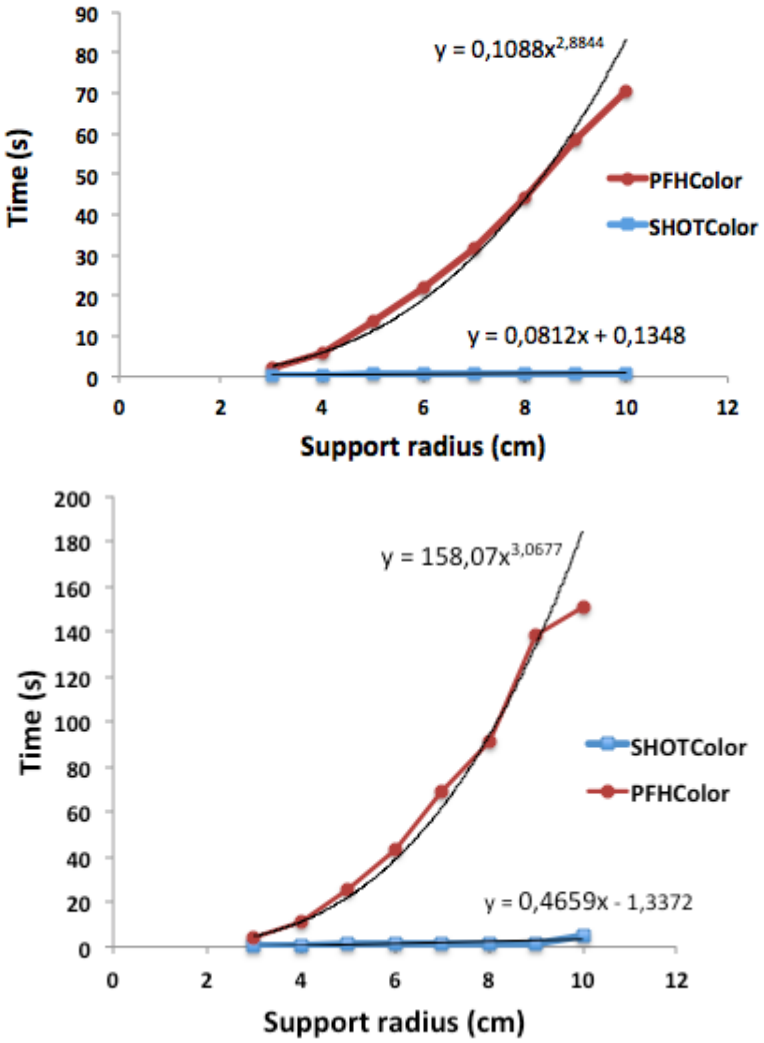


Figure 29: Efficiency of SHOTColor and PFHColor feature descriptors: a)Computing 163 descriptors b)Computing 363 descriptors

4.5 CONCLUSIONS

In terms of computational storage PFHColor descriptors are more convenient since its size is 250 bins, much smaller than the size of SHOTColor descriptors, 1344 bins.

SHOTColor outperforms PFHColor descriptors in terms of descriptiveness in both datasets.

SHOTColor ranks better in terms of robustness to Gaussian noise as well. There is a negative side, though, when the images include clutter and occlusion the effect of Gaussian noise impacts much more its performance than in the case of PFHColor descriptors.

SHOTColor is also able to compact more information than PFHColor descriptors for small support radius values, and it is also able to compensate better the negative effect of clutter and occlusion than PFHColor as the support radius increases.

For time-crucial applications in point cloud, using SHOTColor descriptors would be a good option since they match a computational efficiency and a high descriptiveness power even when there is noise and applied small support radius.

For space-crucial applications PFHColor descriptors is the best option, especially for point clouds including a low number of points: they match a decent descriptive power, have similar computational efficiency to SHOTColor descriptors for small point clouds, and the required storage size is much smaller than the one required by SHOTColor.

For applications requiring a high descriptiveness power, the SHOTColor descriptors would be the best option, specially when the data embeds clutter and occlusion effects. This is because in these cases the chosen support radius value must be low to minimize the negative worsening due to those effects, and for low radius values the SHOTColor performs much better compared to PFHColor descriptors.

BIBLIOGRAPHY

[1] 3d descriptors for object and category recognition: a comparative evaluation. http://www.di.ubi.pt/ lfbaa/pubs/iros2012.pdf.

[2] Descriptor matching - dataset 5 (kinect). http://www.vision.deis.unibo.it/research/78-cvlab/80-shot.

[3] Pcl/openni tutorial 4: 3d object recognition. `http://robotica.unileon.es/mediawiki/index.php/PCL/OpenNI_tutorial_4:_3D_object_recognition_(descriptors)`.

[4] Point cloud library (pcl). `http://www.pointclouds.org`.

[5] Rgb-d object dataset. http://rgbd-dataset.cs.washington.edu.

[6] Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J., and Kwok, N. M. A comprehensive performance evaluation of 3d local feature descriptors. *Springer Science+Business Media New York* (16 April 2015).

[7] Holzbach, A., Rusu, R. B., Blodow, N., and Beetz, M. Fast geometric point labeling using conditional random fields. *IEEE/SRJ International Conference on Intelligent Robots and Systems (IROS), IEEE* (2009), 7–12.

[8] Khoshelham, K. Accuracy analysis of kinnect depth data. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (August 2011).

[9] Lai, K., Bo, L., Ren, X., and Fox, D. A large-scale hierarchical multi-view rgb-d object dataset. *In IEEE International Conference on Robotics and Automation (ICRA)* (May 2011).

[10] Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F. *Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015.

[11] Radu Bogdan Rusu, N. B., and Beetz, M. Fast point feature histograms for 3d registration. *In IEEE Conference on Robotics and Automation (ICRA),IEEE* (2009), 3212–3217.

[12] Rusu, R. B., Blodow, N., Marton, Z. C., , and Beetz, M. Aligning point cloud views using persistent feature histograms. *IEEE/SRJ International Conference on Intelligent Robots and Systems (IROS), IEEE* (2008), 3384–3391.

[13] Salih, Y., Malik, A. S., Walter, N., Desire Sidie2, N. S., and Meriaudeau, F. Noise robustness analysis of point cloud descriptors. *Centre for Intelligent Signal Imaging Research, UniversitiTeknologi PETRONAS, 31750 Tronoh, Malaysia Universite de Bourgogne - Le2i UMR CNRS 6306, 71200 Le Creusot, France* (2013).

[14] Tamas, L., and Jensen, B. Robustness analysis of 3d feature descriptors for object recognition using a time- of-flight camera. *22nd Mediterranean Conference on Control and Automation (MED) University of Palermo* (2004).

[15] Tombari, F., Salti, S., and Stefano, D. Unique signatures of histograms for local surface description. *Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV, Part III. LNCS, vol. 6313 Springer, Heidelberg* (2010).

[16] TOTH, C. D. Binary space partitions: Recent developments. *Combinatorial and Computational Geometry MSRI Publications Volume 52* (2005).