Object recognition and pose estimation from an RGB-D image

BERICHT ZUR FORSCHUNGSPRAXIS
von

Chiraz Nafouki

geb. am 15.09.1992
wohnhaft in:
Willi-Graf-Strasse 17 -0211
80805 München
Tel.: 015175658353

Lehrstuhl für
STEUERUNGS- und REGELUNGSTECHNIK
Technische Universität München

Univ.-Prof. Dr.-Ing./Univ. Tokio Martin Buss

Fachgebiet für
Dynamic Human-Robot-Interaction for Automation Systems
Technische Universität München

Prof. Dongheui Lee, Ph.D.

Betreuer:	M.Sc. Shile Li
Beginn:	07.09.2015
Zwischenbericht:	15.10.2015
Abgabe:	18.01.2016

Abstract

Object recognition and 6-DoF pose estimation play an important role in many robotic applications. They make it possible to identify objects in the environment and accurately estimate their pose in the space.
In this report, a new feature descriptor that combines geometry and color information for object recognition and 6-DoF pose estimation is presented. The training database is obtained by generating synthetic views of the 3D object models. At recognition stage, a segmentation of the scene is performed and the objects and their poses are recognized by searching the k-nearest neighbours in the database.

Contents

Chapter 1

Introduction

Object recognition and 6-DoF pose estimation has gained an increasing interest in recent years. In many robotic applications, such as object grasping, tracking and occlusion handling, the robotic perception should be able to correctly identify objects and accurately estimate their 6-DoF pose in real time. The methods for object recognition can be classified into two main categories: 2D-based methods and 3D-based methods. The 2D-based methods require to first extract keypoints from an image. Then, a descriptor is evaluated for each keypoint, usually based on its surrounding pixels, and the descriptors are saved in a database. The 3D-based methods came to overcome the limits of the 2D-based methods which can not fully represent spatial information. With the advent of low-cost RGB devices such as the Kinect introduced by Microsoft, studying 3D-based techniques has become a more interesting research area.

In order to achieve 3D-object recognition, many feature descriptors were developed. There are two main kinds of descriptors: local descriptors and global descriptors. Like 2D-based methods, local descriptors are also based on extracting keypoints from the 3D-object model. Each local descriptor corresponds to one keypoint and therefore there are as many local descriptors as extracted keypoints. On the other hand, by using a global descriptor, a whole partial view of an object corresponds to one single descriptor. Unlike local descriptors, global descriptors do not require keypoints detection. This results in reduced computational cost compared with local descriptors especially in the matching stage. Therefore, we chose to develop a global descriptor for object recognition and pose estimation because they are faster, more robust against noise and more suitable for real-time applications than local descriptors.

The pipeline of object recognition and pose estimation with global descriptors is presented in Fig. 1.1. The first step of the object recognition pipeline begins with object modelling. In this step, the 3D object models are build. Then, a large amount of synthetic views are generated for each object model. Since we want to be able to recognize each object with different poses and that each viewpoint corresponds to a unique object pose with respect to the camera, we need to generate as many viewpoints as possible in order to provide a more accurate pose estimation. For our experiments, we used 1260 generated synthetic views for each object. After that, a global descriptor is estimated for each synthetic view in the feature extraction step and then stored with the object pose label into a database. For the the testing stage,

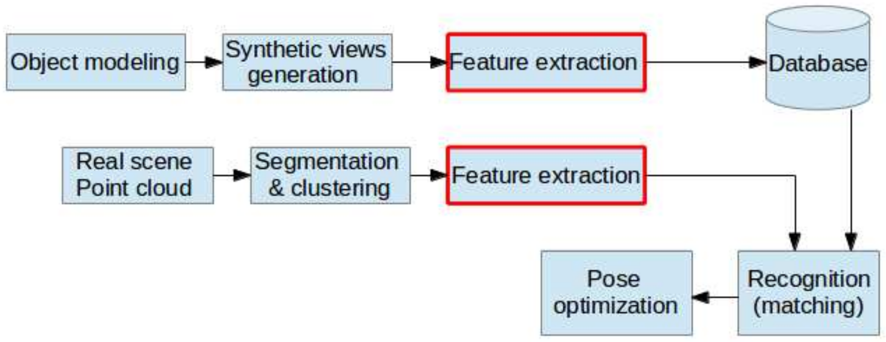


Figure 1.1: Pipeline of object recognition and pose estimation with global descriptors

we consider a real color point cloud scene and the aim is to identify the different objects in the scene and estimate their poses. In order to achieve this aim, a segmentation of the scene should be first performed in order to filter the horizontal surface, on which the objects are supposed to be lying from the scene. The remaining points in the scene should correspond to the different objects.Then, a clustering of these objects is performed based on Euclidean distance, so that each cluster corresponds to one object to be recognized. After that, a global feature descriptor is estimated for each cluster and is compared to the descriptors of the training data. The object recognition and its pose estimation results from finding the best matching i.e. the nearest neighbour in the database with respect to a certain metric.Finally, the pose estimation can be optimized by using an optimization technique such as Iterative Closest Point (ICP).

Our work focused mainly on the feature extraction part. We developed a global descriptor that combines geometry and colour information. Our motivation behind this work was that, in many 2D-based and 3D-based recognition approaches, colour information was proven to improve the recognition rate and and solve the problem of identifying objects having similar shapes but different colours. Logically, this should also stand in the case of global descriptors. However, most of the global descriptors from the state of art use only geometric information and ignore colour information. Therefore , we try to efficiently include colour information in our global descriptor in order to improve the recognition rate and pose estimation accuracy. For this purpose, the Point Cloud Library (PCL), an open-source library that allows 3D point cloud processing [RC11] was used.

This report is organized as follows. In Chapter 2, two global descriptors from the state of art are presented. In Chapter 3, we describe the object modelling of the training data. Our global descriptor and the feature extraction process are both presented in Chapter 4. Object Recognition and pose estimation steps are described in Chapter 5. In Chapter 6, we present our experiment set-up and the experimental results. Finally, we conclude our work in Chapter 7.

Chapter 2

Related Work

In this section, we present two global descriptors from the state-of-the-art which enable to achieve real-time object recognition and pose estimation and explain the underlying principles of each descriptor. Our descriptor was inspired in some aspects by these two descriptors and we aimed at overcoming some of their drawbacks which will be further explained in this section.

2.1 The Viewpoint Feature Histogram

The Viewpoint Feature Histogram (VFH) is a global descriptor which is based on the surflet-pair relation. The surflet-pair relation [WHH03] encodes the geometric properties of the surface of an object by using the surface normals. Let $\mathbf{p_i}$ and $\mathbf{p_j}$ be two arbitrary points of an object and $\mathbf{n_i}$ and $\mathbf{n_j}$ their corresponding surface normals. The surflet-pair relation describes the Euclidean distance between $\mathbf{p_i}$ and $\mathbf{p_j}$, as well as the angles between each surface normal and the connecting line of the two points as shown in Fig. 2.1. There are many local and global descriptors which use the surflet-point-pair relation such as the Point Feature Histogram (PFH) [Rus09], the Fast Point Feature Histogram (FPFH) [RBB09] and the Ensemble of Shape Functions (ESF) [WV11]. The Viewpoint Feature Histogram calculates the

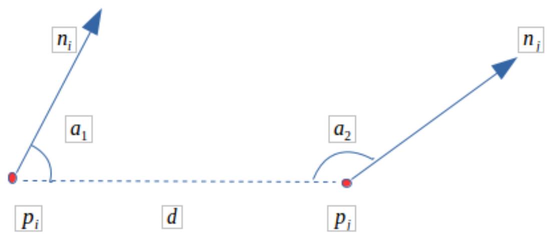


Figure 2.1: Surflet-pair relation between two points $\mathbf{p_i}$ and $\mathbf{p_j}$

surflet-point-pair relation between each point and the centroid of an object. Then, it accumulates the estimated point pair features into a single histogram. The size of the VFH's descriptor is 308 which enables to achieve real time performance in object recognition applications. However the VFH descriptor includes only geometrical information and does not take into account color information. Therefore, it can not distinguish between two objects which have the same shape but different colors such as the two cans in Fig. 2.2. An other issue related to VFH descriptor is that its accuracy is closely related to the estimation of the centroid's position. Therefore, in case the object is partly occluded, the centroid is not correctly estimated, which may affect the recognition results.

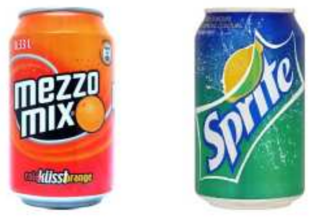


Figure 2.2: An example of two objects with same shape but different colors: a sprite can and a mezzomix can

2.2 The Viewpoint oriented Colour-Shape Histogram

The Viewpoint oriented Colour-Shape Histogram (VCSH) [Li12] is a global descriptor which includes both geometric and color information. It also uses the surflet-point-pair relation for estimating the geometrical feature of the object and adds color information by using a smoothed color ranging method. The geometric part is obtained by calculating four features for each point $\mathbf{p_i}$ of the object: the distance between the point $\mathbf{p_i}$ and the centroid, the angle between the surface normal at $\mathbf{p_i}$ and the viewpoint direction, the angle between the viewpoint direction and the line connecting $\mathbf{p_i}$ to the centroid and the distance between the centroid and its projection on the surface tangent in $\mathbf{p_i}$. For the color information, VCSH uses HSV color space. Unlike RGB color space, HSV is invariant to illumination changes because it separates color from intensity information. The descriptor defines 8 color ranges and in each color range, each of the four geometrical features is encoded in 30 bins. Therefore, the total size of VCSH is 960. Although the correlation between color and geometry in this descriptor allows a more accurate object recognition and pose estimation. However, the large size of the descriptor increases the computational cost especially in the matching stage. Besides, the color histogram used in VCSH considers only the total distribution of colors in the object and does not take into the spatial location of the colors. However, it is possible that two objects have the same color distribution but are perceptually different such as in Fig. 2.3. Our

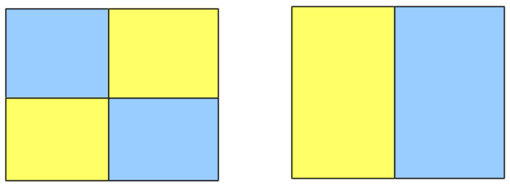


Figure 2.3: Two perceptually different objects with same color distributions.

descriptor aims at overcoming this problem by dividing the object into subregions based on the distance to the centroid and estimating a color histogram for each subregion. For the color histogram, CIELab color space is used. Like HSV, CIELab separates color components from intensity components which makes it invariant to illumination changes. Moreover, CIELab is more numerically stable than HSV at low illumination intensity and is therefore chosen to be used in our work.

Chapter 3

Scene segmentation and Object Modelling

In this section, we describe the preprocessing of testing data by scene segmentation and object clustering and the object modelling of the training data. In the training dataset, the object models are already clustered, so that each point cloud object corresponds to a single object captured from a known viewpoint. A descriptor is then calculated for each object model and stored in the database with the object identity and pose labels. However, in the testing dataset, the objects to be recognized are parts of a complicated real world scene. The objects are not separated beforehand from the background or from other objects like in the case of training data. Therefore, we first need to perform a segmentation of the scene in order to extract the horizontal plane (table, floor ...), on which our objects are supposed to be lying. Then, we cluster the remaining point clouds in the scene so that each cluster should correspond to a single object to be recognized. After that, we estimate a feature descriptor for each object and compare it to the descriptors in the training dataset in order to recognize the object and extract its initial pose estimation.

3.1 Plane Segmentation and object clustering

Plane segmentation consists in identifying the horizontal plane on which the objects lie and extracting this plane from the point cloud in order to facilitate the clustering of the remaining objects. Fig. 3.1 shows an example of a real world scene taken with a Kinect. The scene contains not only the objects that need to be recognized, but also an horizontal white plane which corresponds to a table on which the objects lie. In order to extract the horizontal plane from the scene, Random Sample Consensus (RANSAC) method is used. This method allows the estimation of the plane parameters. If we denote by P the scene point cloud, a plane H can be described

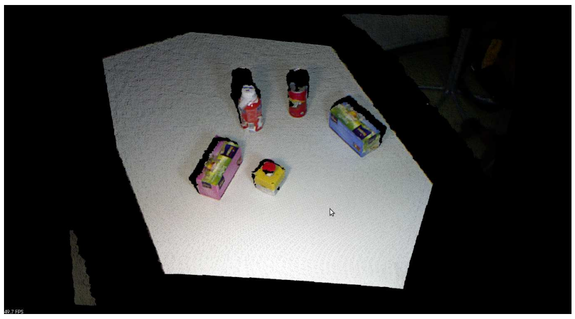


Figure 3.1: An example of a real world scene before segmentation

using four parameters {a,b,c,d} and is defined as follows:

$$H = \{(x, y, z)^T \in P | ax + by + cz + d = 0\} \tag{3.1}$$

To estimate the parameters {a,b,c,d}, RANSAC algorithm first selects three random points from the point cloud P and calculates the four parameters of the plane P_1 defined by the three points. Then the algorithm calculates the Euclidean distances of all points $\mathbf{p} \in P$ to the plane $P1$. A point $\mathbf{p}$ of P is considered to be a point of $P1$ if its distance d to $P1$ satisfies the condition : $d < d_{lim}$ where d_{lim} was set in our experiments to the value 0.01. The algorithm then determines the total number of points that belong to $P1$. This process is repeated 100 times. And the plane model $P_i, i \in [1, 100]$ that contains a maximum number of points is considered as the plane to be extracted and is filtered from the scene.

Once the horizontal plane is eliminated from the scene, the remaining points are clustered into different clusters. The first step of the clustering algorithm is based on a k-d tree search. A k-d tree [Ben75] of a point cloud P is a binary tree, where each node is a k-dimensional point. Every non-leaf node defines a splitting hyperplane that divides the space into two parts based on a specific dimension among the k dimensions. The steps of the clustering algorithm [Rus09] are as follows:

```
1) Generate a k-d tree representation of the input point cloud $P$.
2) Set up an empty list of clusters $C$ and a queue of points that need to be checked $Q$.
3) For every point $\mathbf{p_i} \in P$ perform the following steps:
        1.Add $\mathbf{p_i}$ to the current queue $Q$
        2.For every point $\mathbf{p_i} \in Q$ do:
            • Search for the set $P^i$ of point neighbours of $\mathbf{p_i}$ in a sphere with a radius $r$ to be fixed.
            • For every point in $P^i$, check if the point has already been processed, and if not add it to $Q$.
        3.When all points in $Q$ have been processed, add $Q$ to the list of clusters $C$, and reset $Q$ to an empty list.
```

4) The algorithm is repeated until all points in P have been processed and have become part of the list of point clusters C.
The result of the algorithm should be the different clusters, where each cluster corresponds to an individual point cloud object to be recognized. Fig. 3.2 shows an example of two objects obtained after plane segmentation and object clustering of the scene shown in Fig. 3.1.

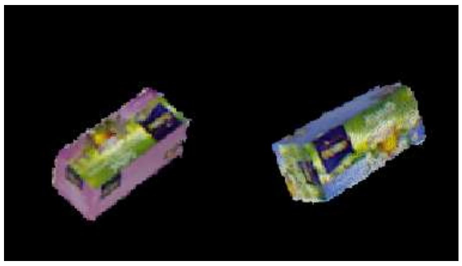


Figure 3.2: An example of two teabags obtained after plane segmentation and object clustering of the scene shown in Fig. 3.1

3.2 Object Modelling

3.2.1 Naïve Sampling

For our experiments, we used the training dataset provided by [Li12]. The object models which form the training dataset are obtained by generating synthetic views from different viewpoints for each 3D object point cloud. To generate the 3D object models, a rotated plane was used on which the objects were put for modelling. Point clouds corresponding to different viewpoints were captured using a Kinect sensor. Once the 3D object models are formed, the synthetic views generation phase begins. A synthetic view means the projection of the 3D object model from a certain viewpoint on a 2D plane. Since each viewpoint corresponds to a certain object pose, the more viewpoints we have, the more accurate the pose estimation will be. For each object model, 1260 synthetic views were generated and stored in a database with the object identity and pose labels. The method of sampling used for synthetic views generation in [Li12] is called naïve sampling [J.K04]. It is based on uniformly sampling each one of the Euler angles independently to get the different rotations of the 3D object. Once the object is rotated, a projection on a 2D plane is performed in order to get the genrated view. The synthetic view generation rate was: every 10° in elevation and every 2° in azimuth. Fig. 3.3 shows an example of some object models which were used for the training dataset. As shown in this figure, some objects have the same shape but different colours and others have similar colours but different shapes. Therefore, these objects are suitable for evaluating the performance of our descriptor in combining colour and geometry information for object recognition and pose estimation.

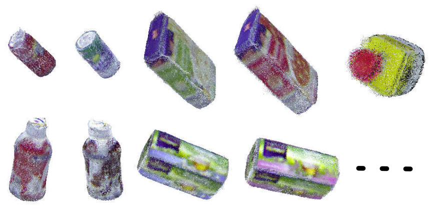
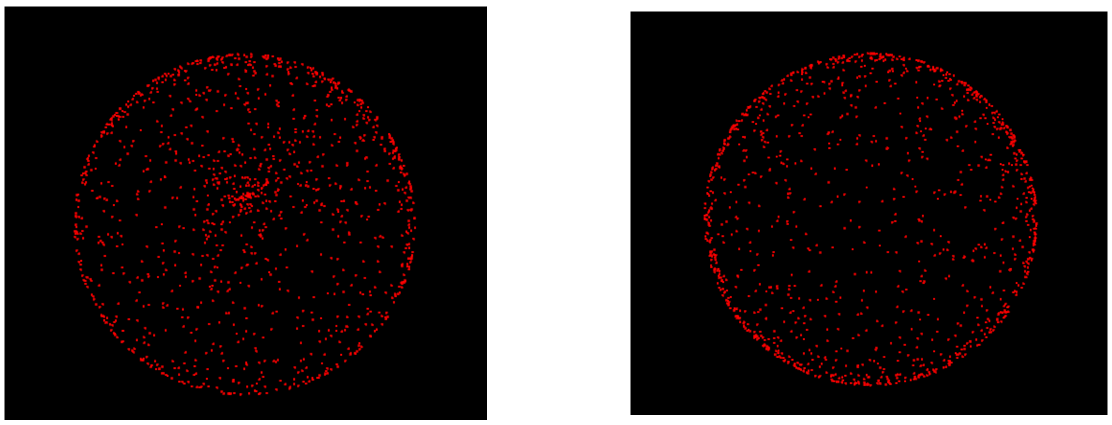


Figure 3.3: An example of object models used for the training dataset


(a) Naïve Sampling

(b) Uniform Sampling

Figure 3.4: Generation of 1260 samples using (a)naïve sampling and (b)uniform sampling. Both figures are shown from an upper view.

3.2.2 Uniform Sampling

The previous approach was based on uniformly sampling each of the elevation and azimuth angles independently. However, this approach does not result in a uniform distribution of the generated viewpoints. In fact, the naïve sampling method presented in the previous paragraph results in a distribution that is heavily concentrated in polar regions [J.K04]. This sampling is therefore inefficient because it results in more samples towards the poles that is needed for the recognition. Moreover, it results in a sparser distribution near the equator which affects the recognition of viewpoints taken at a low elevation angle. Fig. 3.4a shows an upper view of a naïve sampling of a sphere using elevation and azimuth angles for generating 1260 samples as explained in the previous paragraph. A better method of sampling is called uniform sampling [J.K04]. This method allows us to generate a uniformly distributed sampling of rotations. It is based on generating uniform distributions in the range $[-\pi, \pi]$ for the pitch angle, and using the inverse cosine relationship to generate the yaw angle in the range $[0, \pi/2]$. The use of the inverse cosine allows us to avoid oversampling the polar regions. The steps of the uniform sampling algorithm for a single viewpoint generation are given in the following pseudocode:

1) Choose the pitch angle θ as follows: $\theta = 2 * \pi * rand() - \pi$.
2) Choose the yaw angle ϕ as follows:$\phi = arccos(rand())$.

Fig. 3.4b shows an upper view of the result of a uniform sampling of a sphere using pitch and yaw angles for generating 1260 synthetic views. It is clear that the problem of oversampling near polar regions is solved with this method resulting in a uniform distribution of sampling rotations.

Chapter 4

Feature Extraction

In this section, we describe our global feature descriptor. Our feature descriptor is composed of two parts. The first part contains merely geometric information and is inspired from the surflet-pair relation. This part is more robust against occlusions than VFH or VCSH descriptors because it is not based on calculating the surflet-pair relation between the centroid and other points of the object. Instead, it calculates the surflet-pair relation by using randomly chosen point pairs. In the second part, colour and geometry information are correlated by dividing the object into subregions and estimating the colour histogram for each subregion. By this way, the colour locations are better taken into account than in VCSH which only calculates the total color distribution within the object.

Let's consider a point cloud P, which corresponds to an object and contains n points $\mathbf{p_i}$, with $i \in [0, n]$. To calculate the first part of the descriptor, we start by choosing random point pairs $(\mathbf{p_i}, \mathbf{p_j})$ from the point cloud object a certain number of times and calculate their normals $(\mathbf{n_i}, \mathbf{n_j})$ each time. For each randomly chosen point pair , three geometric features inspired from the surflet-pair relation are evaluated. These features were proven to effectively encode geometric information of point cloud surfaces, especially when these point clouds contain many surface variations [WHH03]. The first feature consists in the Euclidean distance $d(\mathbf{p_i}, \mathbf{p_j})$ between the two random points:

$$d(\mathbf{p_i}, \mathbf{p_j}) = ||\mathbf{p_i} - \mathbf{p_j}|| \tag{4.1}$$

The distance feature is encoded in a 30-bin histogram. To define the limits of each bin, we first calculate the maximum distance d_{max} between two points within the object:

$$d_{max} = max\{d(\mathbf{p_i}, \mathbf{p_j}); \mathbf{p_i} \in P, \mathbf{p_j} \in P\} \tag{4.2}$$

Then, we normalize the bins with respect to the maximum distance d_{max}. An example of calculation of the distance histogram is shown in Fig. 4.1. For each two random points chosen in the object, the distance between them is calculated and the corresponding bin is incremented by 1. The second feature consists of the angle between each of the two normals and the connecting line of the two points. Therefore for each point pair $(\mathbf{p_i}, \mathbf{p_j})$, we evaluate two angles a_1 and a_2 as follows:

$$\begin{aligned} a_1 &= \angle(\mathbf{n_i}, \mathbf{p_i p_j}) \\ a_2 &= \angle(\mathbf{n_j}, \mathbf{p_i p_j}) \end{aligned} \tag{4.3}$$

The angles are also encoded in a 30-bin histogram. The third geometric feature

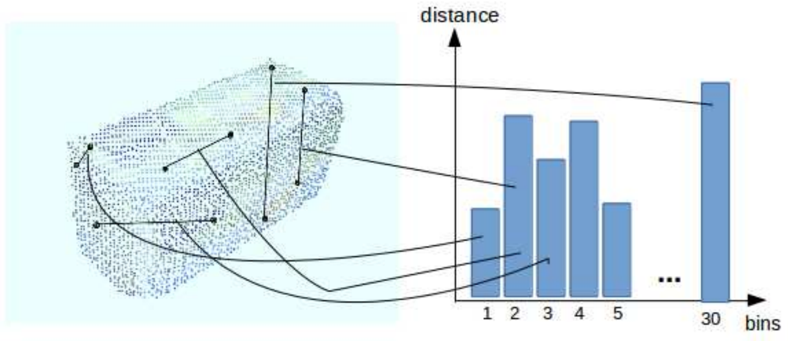


Figure 4.1: Estimation of the distance histogram for a point cloud object. The distance between each randomly chosen point pair is classified into one bin among 30 bins.

corresponds to the angle a_3 between the viewpoint direction $\mathbf{v}$ and the connecting line of the two points:

$$a_3 = \angle(\mathbf{v}, \mathbf{p_i p_j}) \tag{4.4}$$

The viewpoint direction $\mathbf{v}$ is defined as the line connecting the origin of the camera to the centroid of the object $\mathbf{p_i}$. This feature is also encoded in a 30-bin histogram. Therefore, the size of the first part of the descriptor is $30 \cdot 3 = 90$.

The second part of our descriptor includes both colour and geometric information. For this part, we need to divide the object into 10 regions, based on the distance to the centroid. In order to achieve this, we start by calculating the maximum distance d'_{max} to the centroid $\mathbf{c}$:

$$d'_{max} = max\{d(\mathbf{c}, \mathbf{p_i}); \mathbf{p_i} \in P\} \tag{4.5}$$

Then, for each random point $\mathbf{p_i}$, we calculate its distance to the centroid $d(\mathbf{c}, \mathbf{p_i})$. The region number $R_n \in [1, 10]$ to which the point $\mathbf{p_i}$ belongs is determined as follows:

$$R_n = ceil(d(\mathbf{c}, \mathbf{p_i}) \cdot 10/d'_{max}) \tag{4.6}$$

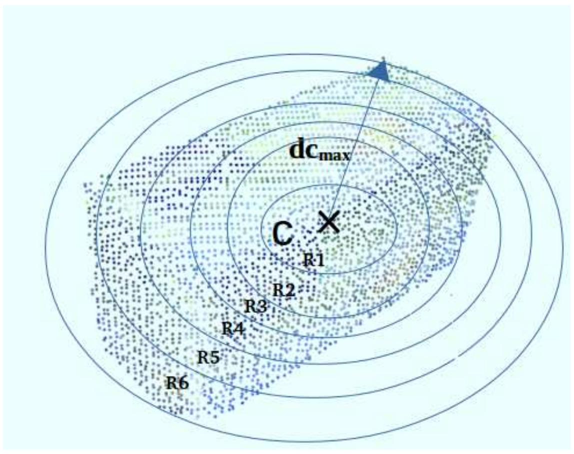


Figure 4.2: Definition of 6 regions for an object characterized by its centroid $\mathbf{c}$ and the maximum distance to the centroid d'_{max}

Fig. 4.2 shows an example of the definition of regions for an object characterized by its centroid $\mathbf{c}$ and the maximum distance to the centroid d'_{max}. For the purpose of simplification, we divided the object into 6 regions only. Once we classified the point $\mathbf{p_i}$ into one region between 1 and 10, we convert its RGB colour to CIELab colour space. CIELab is a three dimensional colour space, which was introduced by the Commission Internationale de l'Eclairage (CIE) and adopted in 1970. Unlike RGB, CIELab is device independent. This means that it is not related to a specific device such as camera, scanner, etc. An other advantage of CIELab compared with other color spaces such as RGB or HSV is that it is perceptually uniform. This means that in this color space, the distance between points is directly proportional to color difference perceived by the human eye [CF97]. CIELab color space can be represented in 3 dimensional space by a sphere with three coordinates designed by L^*, a^* and b^* as shown in Fig. 4.3. The vertical L^* axis represents lightness or gray scale and ranges from 0 to 100. The horizontal a^* and b^* axes represent the chroma and cross each other in the center of the sphere. The a^* axis is green at one extremity, and red at the other.The b^* axis is blue at one extremity, and yellow at the other. The choice of the axes a^* and b^* is based on the color opponent process, which suggests that there are three opponent channels in the human vision system: red versus green, blue versus yellow, and black versus white [CIE04]. If we move horizontally away from the L^* axis, the color saturation increases gradually.In theory there are no maximum values of a^* and b^* but in practice, a^* values range in $[-500, 500]$ and b^* values in $[-200, 200]$. The transformation from RGB color space to CIELab color space is a non-linear transformation. The RGB values are first normalized to the range $[0, 1]$ and then transformed to an intermediate space called

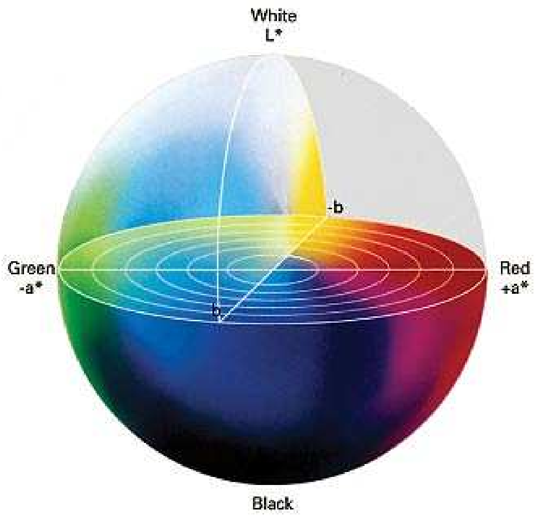


Figure 4.3: CIELab color space.

CIEXYZ color space [CF97] as follows:

$$\begin{aligned} X &= R * 0.4124 + G * 0.3576 + B * 0.1805 \\ Y &= R * 0.2126 + G * 0.7152 + B * 0.0722 \\ Z &= R * 0.0193 + G * 0.1192 + B * 0.9505 \end{aligned} \tag{4.7}$$

After that, the XYZ values are transformed to CIELab colour space using the following equations:

$$\begin{aligned} L^* &= 116 f(Y/Y_0) - 16 \\ a^* &= 500[f(X/X_0) - f(Y/Y_0)] \\ b^* &= 500[f(Y/Y_0) - f(Z/Z_0)] \end{aligned} \tag{4.8}$$

where $X_0 = 95.047$, $Y_0 = 100.000$ and $Z_0 = 108.883$. The function f is defined as follows:

$$f(x) = \begin{cases} \sqrt[3]{x} & \textit{if} \quad x > 0.008856 \\ 7.787x + 0.138 & \textit{else} \end{cases} .$$

For each of the ten regions, we define a color histogram of size 30 bins, where each of the 3 CIELab values is encoded into 10 bins. Therefore the size of the second part of our descriptor is $30 \cdot 10 = 300$ and the total size of the descriptor is $90 + 300 = 390$. Fig. 4.4 shows how our descriptor is divided. The first part contains three geometric features: The distance d between two random points, the angle between a segment connecting two random points and their normals and the angle between the viewpoint direction and a segment connecting two random points. The second part of the descriptor allows a correlation between colour ang geometry information since each random point is first classified into a region based on its distance to the

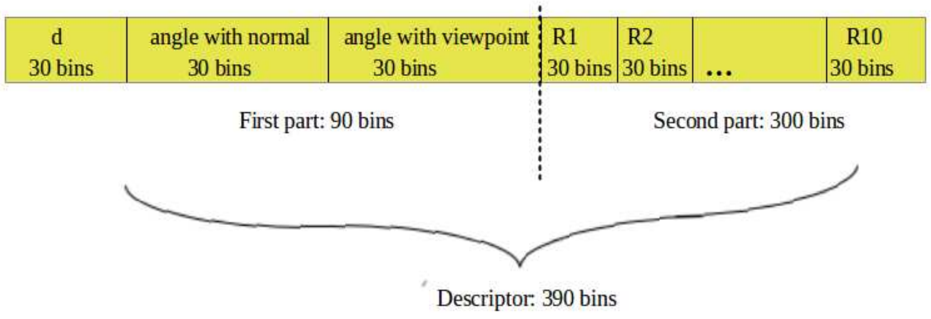


Figure 4.4: Structre of the feature descriptor

centroid and then the colour histogram corresponding to its region is incremented based on its CIELab values.

Chapter 5

Recognition and Pose Estimation

In this section, the object recognition and pose estimation phases are described. The goal of object recognition is to provide a set of candidates from the training dataset that could correspond to an object taken from a real world scene. The 6-DoF pose associated to each candidate is also retrieved from the training data and given as the initial pose estimation of the object. This initial pose estimation can be improved by using an optimization technique such as the Iterative Closest Point (ICP).

5.1 Object Recognition

The recognition problem can be seen as a nearest neighbour estimation problem. To achieve the nearest neighbour search, a k-d tree structure of all the descriptors of the training dataset was used. Then, given a feature descriptor $\mathbf{f}$ of an object to be recognized, the k-nearest descriptors are searched in the k-d tree structure and sorted according to their distances to $\mathbf{f}$. Chi-Squared distance was used for the nearest neighbours search. If $\mathbf{f} = (f_1, f_2, ..., f_{390})^T$, and $\mathbf{g} = (g_1, g_2, ..., g_{390})^T$ are two descriptors, the Chi-Squared distance $d(\mathbf{f}, \mathbf{g})$ between $\mathbf{f}$ and $\mathbf{g}$ is defined by :

$$d(\mathbf{f}, \mathbf{g}) = \sum_{k=1}^{390} \frac{(f_k - g_k)^2}{f_k + g_k}. \tag{5.1}$$

Once the k-nearest neighbours i.e descriptors having the k minimal distances to $\mathbf{f}$ are found, their corresponding pose estimations are retrieved from the database. In our case, we take k=1 and therefore we consider only the nearest neighbour for object recognition and pose estimation. The initial pose estimation gives us an initial guess of the transformation matrix T_{init} from the object-coordinate-system to the camera-coordinate-system.

5.2 Pose Estimation

The initial pose estimation of a recognized object can be optimized using Iterative Closest Point (ICP) algorithm [Zha94]. ICP is used to minimize the difference between two point clouds. One point cloud, known as the reference or target, is kept fixed, while the other point cloud, known as the source, is transformed to best match the reference. The algorithm iteratively changes the geometric transformation, which is a combination of translation and rotation applied on the source, in order to minimize the distance from the source to the reference point cloud. In our case, the reference point cloud is the object in the testing data and the source is the recognized object in the training data taken from a certain viewpoint. After iterating a certain number of times, the transformation matrix $Ticp$ calculated using ICP is estimated. The final transformation matrix from the camera-coordinate-system to the object-coordinate-system is therefore obtained as follows:

$$T_{final} = T_{icp} \cdot T_{init}. \tag{5.2}$$

Once we estimate T_{final}, we can apply the final geometric transformation on the 3D model of the recognized object, visualize the object in its estimated pose and compare the estimated pose with the real pose of the object.

Chapter 6

Experimental Results

In this chapter, we present the results of some experiments that we have conducted to evaluate the performance of our descriptor. Our experimental setting consists of a Kinect sensor and an horizontal plane (floor or table) on which the objects to be recognized lie as shown in Fig. 6.1. Our code was implemented using the open source point cloud library (PCL) and was integrated into a ROS node. This allowed us to receive online frames from Kinect and to publish in real time the identities and pose estimations of the recognized objects. Our experimental results take into

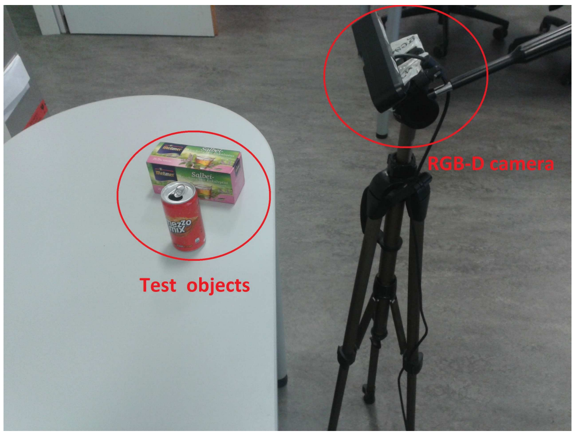


Figure 6.1: Experimental setting

consideration three main aspects: recognition rate, pose accuracy and occlusion handling. We show here that our descriptor is able in most cases to correctly identify objects and estimate their poses, even if they are partially occluded by other objects.

6.1 Recognition rate

In order to estimate the ability of our descriptor to correctly identify an object, we consider for each testing object its nearest neighbour in the training data. we

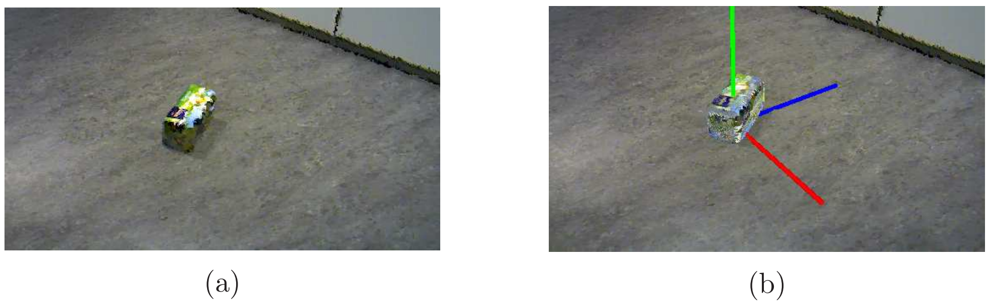


Figure 6.2: Original object (a) and superposition of the recognized object in its estimated pose with the real scene (b).

compare the label of the nearest neighbour to the real label of the object and if the nearest neighbour is misidentified, we consider the recognition to be wrong, otherwise we consider the recognition to be correct. An other possible option was to take several nearest neighbours and to perform a voting process to decide the identity of the object. We took 20 different random partial views of each of the the objects that we want to recognize in real scene. We obtain a recognition rate of 94% which confirms the liability of our descriptor in correctly recognizing objects. We also evaluated the recognition rate using 100 random synthetic viewpoints taken from the training data. We obtain a recognition rate of 100%. We recall that our descriptor is not deterministic since it is based on the choice of N random points from the point cloud object, with N being the total number of points in the object. Therefore, the feature vector obtained from the application of our descriptor to a same object taken from a fixed viewpoint is not constant.

6.2 Pose accuracy

One of the main aims of our descriptor is to give a good estimation of the pose of the object to be recognized. In order to evaluate the accuracy of the pose estimation, we superpose the reconstructed 3D object model in its estimated pose with the object in its real pose as captured in the scene. We compare the superposition of objects with the eye and deduce if the pose is estimated accurately. Fig. 6.2 shows an example of the original object taken in a real scene (Fig. 6.2a) and the superposition between the original object and the recognized object in its estimated pose (Fig. 6.2b). As we can notice from the superposition of the two objects, the estimated pose is very close to the real pose. In most cases our descriptor was able to give a very accurate estimation of the object.

Fig. 6.3 shows an example of object recognition and pose estimation of different objects. Here, the estimated poses are used for the reconstruction of the 3D models.

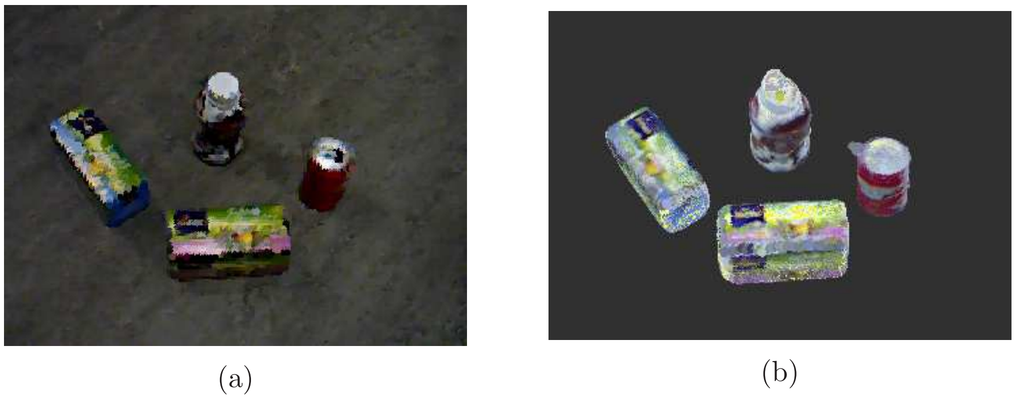


Figure 6.3: Original camera view (a) and reconstructed 3D models based on the estimated poses of the recognized objects (b).

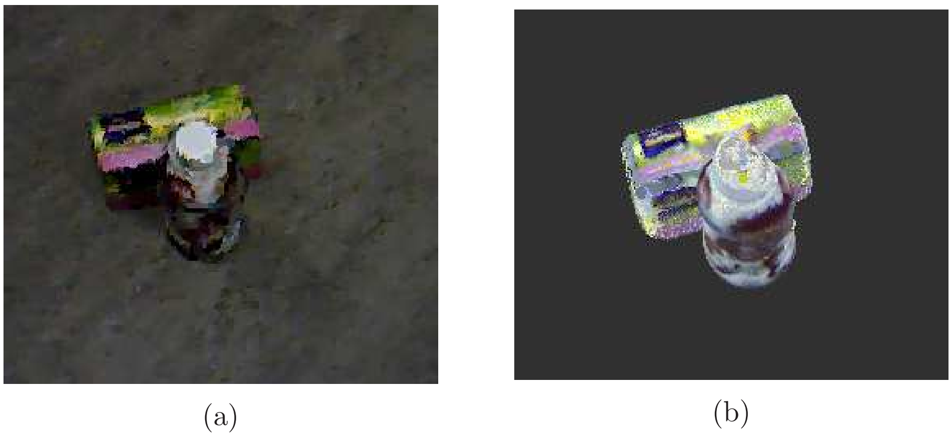


Figure 6.4: Original camera view (a) and reconstructed 3D models based on the estimated poses of the recognized objects in case of partial occlusion (b).

6.3 Occlusion handling

An other feature of our descriptor is its ability to handle partial occlusions. In fact the choice of random points to evaluate the feature vector makes our descriptor more robust against missing points which may result from occlusions or from sensor inaccuracy. In order to estimate the performance of our descriptor in handling partial occlusions, we realized many experiments in which some objects were partially occluded by other objects. Our descriptor was able in most cases to correctly recognize the objects. The estimated poses are less accurate than without occlusions but are still acceptable. Fig.6.4 shows an example of object recognition and pose estimation in case of a partial occlusion.

Chapter 7

Conclusion and future work

In this report, our global feature descriptor that combines color and geometry information for object recognition and pose estimation is presented. Our descriptor is composed of two main parts: the first part is based on the extraction of geometric properties using random point pairs. The second part is based on a correlation between color and geometry information using also random points from the object. Experiments show the ability of our descriptor to recognize and estimate accurately the pose of objects having different shapes and colors. The integration of color information makes our descriptor capable of distinguishing objects having similar shapes but different colours. Moreover, the choice of random points makes our descriptor more robust against occlusions which was also shown in our experiments.

In future work, the occlusion handling could be further improved, mainly by making the second part of our descriptor independent of the centroid estimation. An other possible solution for better occlusion handling is to apply the first part of our descriptor independently to estimate the position of the centroid and use the estimated position in the second part to include the colour information. An other possible improvement for our work is to use Locality-Sensitive Hashing (LSH) algorithm instead of kd-tree structure for the nearest neighbours search. In fact, LSH was proven to be more efficient than kd-tree for high dimensional vectors which is the case for our descriptor.

List of Figures

Bibliography

[Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Magazine Communications of the ACM*, 18:509–517, September 1975.

[CF97] Christine Connolly and Thomas Fliess. A study of efficiency and accuracy in the transformation from rgb to cielab color space. In *IEEE Transactions on image processing, Vol. 6, No. 7*, July 1997.

[CIE04] CIE. Colorimetry, 3rd Ed., Publication No. CIE 15:2004, Vienna, Austria. Technical report, 2004.

[J.K04] James J.Kuffner. Effective sampling and distance metrics for 3d rigid body path planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[Li12] Shile Li. Fast rigid object recognition and 6-dof pose estimation using color and depth information. B.S. Thesis, Technische Universität München, 2012.

[RBB09] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast Point Feature Histograms (FPFH) for 3D Registration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 12-17 2009.

[RC11] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[Rus09] Radu Bogdan Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Technische Universität München, 2009.

[WHH03] Eric Wahl, Ulrich Hillenbrand, and Gerd Hirzinger. Surflet-pair-relation histograms: A statistical 3D-shape representation for rapid classification. In *3-D Digital Imaging and Modeling*, pages 474–482, 2003.

[WV11] Walter Wohlkinger and Markus Vincze. Ensemble of shape functions for 3D object classification. In *ROBIO*, pages 2987–2992, 2011.

[Zha94] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13:119–152, 1994.