Visual perception for the 3D recognition of geometric pieces in robotic manipulation (this is a manuscript draft)

Printed version available: http://link.springer.com/content/pdf/10.1007%2Fs00170-015-7708-8.pdf

On-line access: http://link.springer.com/article/10.1007/s00170-015-7708-8

C.M. Mateo, P. Gil, F. Torres

Physics, Systems Engineering and Signal Theory Department

University of Alicante

PO Box 99, 03080, Alicante. Spain.

Carlos.mateo@ua.es, Pablo.Gil@ua.es, Fernando.torres@ua.es

Abstract: During grasping and intelligent robotic manipulation tasks, the camera position relative to the scene changes dramatically because the robot is moving to adapt its path and correctly grasp objects. This is because the camera is mounted at the robot effector. For this reason, in this type of environment, a visual recognition system must be implemented to recognize and 'automatically and autonomously' obtain the positions of objects in the scene. Furthermore, in industrial environments, all objects that are manipulated by robots are made of the same material and cannot be differentiated by features such as texture or colour. In this work, first, a study and analysis of 3D recognition descriptors has been completed for application in these environments. Second, a visual recognition system designed from a specific distributed client-server architecture has been proposed to be applied in the recognition process of industrial objects without these appearance features. Our system has been implemented to overcome problems of recognition when the objects can only be recognized by geometric shape and the simplicity of shapes could create ambiguity. Finally, some real tests are performed and illustrated to verify the satisfactory performance of the proposed system.

Keywords: 3D object recognition, 3D shape detection, pose estimation, robotic manipulation, geometric objects, surfaces

1. Introduction

Robotic manipulation has continually developed in recent years. This task requires scene interpretation in unstructured environments from two types of sensors, contact sensors and contactless sensors. Contact sensors include force and tactile sensors, while contactless sensors include visual sensors based on images. Data from both types of sensors are fused and/or combined to perform tasks of grasping of unknown objects and to address the problems and challenges in the field of autonomous robots, including motion planning, grasping planning, handling and intelligent manipulation of objects. In general, automated robotic grasping tasks require a priori knowledge of the objects to be manipulated. Thereby, visual sensors are applied in these tasks to identify objects in the environment by matching them with models stored in a database, to locate their positions and orientations and to recognize features for the reconstruction of the object geometry to evaluate its surface. Visual sensors that allow us to obtain three-dimensional information on the environment include stereo systems, laser-camera systems, time of flight cameras (ToF), and RGBD cameras.

Different methods have been proposed in the literature to manipulate or grasp objects by robotic hands or claws in industrial [1-4] and/or household environments [5-7]. Grasping in industrial environments is mainly focused on resolving pick and place problems. Recent works have tried to efficiently solve the problem of bin-picking using robot arms equipped with ToF sensors [8]. To accomplish this goal, they presented a computer vision algorithm combined with a robot control scheme. In household environments, first, Papazov et al. [5] proposed a recognition method based solely on 3D geometry information and an efficient localized RANSAC-like sampling strategy. Second, Ciocarlie et al. [6] combined aspects of perception such as scene interpretation from 3D data and aspects of manipulation such as grasp planning and motion planning as well as identification and recovery of grasping failure using tactile sensors. In both cases, the image processing was made with only a depth map to extract geometry information. Another work [9] used a fusion between a structured light sensor and a ToF camera by means of 3D mapping for object localization in industrial applications. Adán et al. [10] introduced and analysed a 3D recognition/pose strategy based on depth gradient image models (DGI). This representation synthesizes both surface and contour information.

In this paper, we address the problem of interpretation of a scene to recognize 3D unknown rigid objects and estimate the location in six degrees of freedom from a database of object models previously created from CAD software, in contrast with the models used by [10]. The proposed method for recognition is focused on industrial applications such as [2] but using classification descriptor methods. In contrast with the approach shown in [5] in which the RANSAC strategy is used to localize models within a scene, a strategy based on scene segmentation plus a process of finding a complex model of features in a data structure is used to localize models. In our approach, the scene is registered in range images from a lone RGBD sensor instead of the method used in [9] with sensorial fusion; we also use two levels of resolution at the same time. Another highlight of the method is the exclusive use of 3D data such as the structured point clouds, in contrast with [11]

and [2] that present a previous segmentation using 2D data and use unstructured point clouds, respectively.

The rest of this paper is organized as follows: Section 2 discusses and analyses 3D descriptors based on geometry, mainly surface normal-based descriptors from related works. Section 3 describes the architecture for the proposed object recognition system. In section 4, we present the proposed method to recognize objects determining their location in the environment. In section 5, we describe the types of objects and their features and how the database for the evaluation and training of our recognition system was generated. Furthermore, in this section, the behaviour and precision of the proposed method is shown in several experiments where the method is integrated as a sensorial subsystem of a robotic handling system. Finally, we show the conclusions of this work in Section 6.

2. Analysis and evaluation of 3d geometry descriptors

2.1 Methods based on normal to surfaces

There are many normal-based strategies to describe the underlying surface of a point cloud. These strategies describe the point cloud by checking the relationships among their normal vectors. These relationships are searched within either a local or global reference frame. Thus, a Darboux frame $\langle \alpha, \phi, \theta \rangle$ is often used as a reference frame for the PFH (Point Feature Histogram) [12] descriptors family. However, a more simplistic reference frame is used for SHOT (Signature of Histogram of OrienTations) [13]. This is based on the computation of $\cos(\theta)$, where θ is the angle between $\mathbf{n}_i$ and $\mathbf{n}_j$, where $\mathbf{n}_i$ is the normal vector of point p_i, and p_j likewise has the normal vector $\mathbf{n}_j$.

The PFH descriptors family is mainly composed of PFH, FPFH (Fast Point Feature Histogram), VFH (Viewpoint Feature Histogram) and CVFH (Clustered Viewpoint Feature Histogram) [12]. PFH is a local descriptor calculated by using local neighbourhood areas. It computes a tuple $\langle \alpha, \phi, \theta \rangle$ for each relationship among the points of a same neighbourhood area. Each tuple represents the relationship among one and all normal vectors in their neighbourhood, according to the Darboux frame computed by equation (1).

$$\begin{cases} \alpha = \mathbf{v}^T \cdot \mathbf{n}_j \\ \phi = \mathbf{n}_j^T \cdot \dfrac{p_i - p_j}{\left\| p_i - p_j \right\|_2} \\ \theta = \arctan(\mathbf{w}^T \cdot \mathbf{n}_i, \mathbf{n}_i^T \cdot \mathbf{n}_j) \end{cases} \tag{1}$$

where vector $\mathbf{v}^T$ is the director vector from p_i to p_j and vector $\mathbf{w}^T$ is a vector perpendicular to $\mathbf{n}_i$ and $\mathbf{v}^T$.

All descriptors of the PFH family describe the shape of a surface by means of a signature. To compute this signature, the PFH method takes into consideration the relationships among all points within the neighbourhood environment. In this case, the computational complexity is $O(nk^k)$. In addition, the signature dimensionality is set by b^3, where the unknown b is the number of bins. This parameter was set to 5 in this work, as Aldoma *et al.* proposed in [12], and hence the dimensionality is 125.

FPFH is based on the same idea as PFH, to accelerate the algorithm by decreasing the computational cost. It uses a Darboux frame to make relationships among pairs of points within a neighbourhood with radius r for computing each local surface signature. This approximation changes the relationships among a point and its neighbours located with a distance smaller than r by adding a specific weight according to the distance between the point and every neighbour. This descriptor generates a linear complexity in the number of neighbours $O(nk)$. FPFH has a signature dimensionality smaller that PFH, and it is set with the relation $b{\cdot}3$. For this work, b was set to 11, as the authors proposed in [12], and hence the dimensionality was 33.

VFH is based on FPFH. Each signature consists of a histogram with two components; one represents the Darboux frame angles, which are calculated as the angular relationship between a normal vector at a point and the normal vector at the point cloud centroid, and the other represents the angles between each point's normal and the director vector determined by the surface centroid and viewpoint. Moreover, information about the distance from each point to the centroid is added in this second component. In this case, the descriptor dimensionality was $308 = b{\cdot}4 + 128$. The dimensionality is divided into two parts (two different-sized bins) because this descriptor has two components. The bin size b is 45 for the first, and 128 for the second, as Rusu *et al.* proposed in [13]. The value 4 represents the number of Darboux frame components and the supplementary distance information. For this reason, the VFH descriptor has a complexity of $O(n)$, as $k = 1$ whether the cost is compared with FPFH.

CVFH is an improved version of VFH. The underlying idea is to split an object into a set of smooth and continuous regions or clusters. Then, the clusters are used to recognize the object by processing and detecting the cluster shapes. The rest of the object such as edges, ridges and other discontinuities on the surface are not considered useful parts of the object because they are more affected by noise when processing techniques are applied. Afterwards, a VFH descriptor is computed for each and every one of the clusters obtained from the object. In particular, VFH describes a surface as a histogram in which each histogram item represents the centroid to the surface and the average of the normal vectors among all points of the surface. The inconvenience in this descriptor is the increase of the computational cost to $O(n\log(n))$.

SHOT is a descriptor that does not belong to the FPH family. SHOT was presented as hybrid signature and histogram local feature descriptor by its authors. This descriptor uses a partitioned spherical grid as a local reference frame. Generally, this spherical grid is split into 32 grid sectors. In particular, 2 divisions were chosen for elevation, 8 for azimuth and 2 for radial, as Salti *et al.* proposed in [14]. This local reference frame is both unique and unambiguous. The histogram counts the differences between the normal vector at each point of the surface within the local

supports (local reference frame) and the normal vector at the query feature point (the centre of the local support). The difference between the two normal vectors was computed by $\cos(\theta) = \mathbf{n}_i \cdot \mathbf{n}_j$. This difference was binning into 11 classes, so the signature dimensionality is $352 = \#bins + \#grid\_sectors$. The computational complexity is $O(nk)$ in this case.

Both bin and division parameters influence the behaviour of the descriptors (Table 1). On the one hand, an increase of these parameters provides more detail in the representation of the objects. But also, it causes an increase of the computational cost. If we want to work in real time, this is not recommended. On the other hand, if we reduce the divisions or use few bins, the runtime is smaller (computational cost is decremented) but the representation of the descriptors is less accurate. This can increase the number of ambiguities in the recognition process. For example, two objects whose shape is very different may be represented by a similar descriptor.

Table 1. Accuracy and runtime depending on the bin and division parameters

	Number of bins								
	1	3	5	7	9	11	13	15	17
Accuracy (%)	0.25	0.25	0.43	0.63	0.81	**0.94**	0.91	0.94	0.95
Runtime (ms)	627	650	734	782	761	**767**	875	825	853

Among all descriptors based on normal features, SHOT and CVFH are the best fits for the issues addressed in this work. The first is a local surface feature extraction method, inasmuch as it looks for surface features within the neighbourhood environment around key points, while the second is likely the most relevant global surface features extraction method [12]. This last has good rate accuracy vs. time. CVFH looks for surface features in an entire point cloud as a unique neighbourhood environment.

Figure 1. Confusion matrices show the classification capability of a descriptor for: **(a)** PFH **(b)** FPFH **(c)** VFH **(d)** CVFH **(e)** SHOT and **(f)** the runtime cost. The dataset used for the tests is shown more in detail in the experiments of section 5.

Nonetheless, both have strengths and weaknesses. On the one hand, SHOT has a great accuracy rate for classifying objects, and it also has a favourable capability to retrieve object pose information. On the other hand, CVFH also has great accuracy rates but also improves the runtime. The SHOT weaknesses are the high runtime and its dependence on the used key-points extraction method. The main problem of CVFH lies in its having to use an auxiliary method to retrieve the roll rotation, being that CVFH only has 5 DoF (degrees of freedom). OUR-CVFH (Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram) [15] was designed to add one more degree of freedom, for a total of 6 DoF, to overcome this problem.
Some researchers in [15] describe the descriptor OUR-CVFH as an enhancement of CVFH using a reference frame for each cluster so it becomes unambiguous. To accomplish this improvement,

they first simplified the descriptor by reducing the encoding of the viewpoint component to 64 bins and removing the distance information. Second, they added a new component for the encoding of the surface transformation. This new component was divided into 8 octants, and each was binned in 13 classes, so that the total new dimensionality was 303

2.2 Comparison of descriptors

An analysis and evaluation of the normal-based surface descriptors, shown in section 2.1, was presented in previous works [12] and [16]. Both works test the classification capability of the descriptors to be applied in the recognition process of real objects. In [12], a similarity measure based on a distance metric was used to test the recognition process. In this case, the Euclidean distance was considered because it is the most traditional metric for a matching process. In [16], these works were extended to different similarity measures such as Manhattan, Euclidean, Chi-squared and Hellinger distances. Both studies uses metrics to measure the similarity among objects and models stored in a database by comparison of descriptors. The Manhattan distance gives the best results if all metrics are compared as shown [16]. Another difference between the two works is that the database models are real objects in [12] vs. object synthetic views generated from CAD models in [16], but in both works, the models are compared with real partial views captured from a depth sensor. In that last work, these objects were printed using a 3D printer, and printed object partial views were captured and digitalized from a range camera to be compared with database information with the aim of finding the best match among CAD models and views. In summary, both the type of descriptor and the distance metric affect the searching to find a model that looks similar to a partial view of the object.

Figure 2. Distributed architecture of a robot vision system.

On the one hand, the confusion matrices shown in Figures 1(a)-1(e) indicate that both SHOT and CVFH are the best descriptors to recognize and classify geometric shapes without colour, texture, or roughness. On the other hand, Figure 1(f) shows the high computational cost of SHOT vs CVFH. For this reason, we can conclude that the best descriptor to be applied in recognition processes in robotic manipulation tasks or grasping should be CVFH. Consequently, if we also wish to overcome its limitations to retrieve the pose in a world where the items are positioned in 6 DoF and not 5 DoF, we will have to use OUR-CVFH.

Figure 3. (a) Mathematical transformations among robot-vision and segmented scene. (b) 3D image processing of the scene from a sensor.

3. Proposed architecture for the object recognition system

The hardware used was a Mitsubishi PA-10 industrial robot arm. This robot has 7 DoF. For visual sensing, we acquired images from a Microsoft Kinect(TM) RGBD range sensor mounted on the robot's end-effector.

The general system architecture is shown in Figure 2. The PA-10 robot is controlled as a slave in a software architecture client-server. It is connected with a module server installed on a personal computer ('PA-10 controller'), and both items are communicated via ARCNET (COM20020 controller chip plus the HYC4000 line driver). The robot is always waiting for commands in robot machine language from the 'PA-10 controller'. These commands are generated from the orders given by the computer 'ROSCORE' that manages the tele-operation interface with the robot via User Datagram Protocol (UDP). The range sensor was directly connected with another computer called 'ROSCORE' by a USB interface. The range data sequence, which is captured from 'ROSCORE' by the range sensor, is sent to the 'MAIN' computer via Local Area Network (LAN) based on TCP/IP architecture using Ethernet as a link layer. The data sequence captured by the range sensor is sent by means of the communications strategy implemented within the Robotic Operative System (ROS), which is based on a message structure called 'rostopic' that was installed in both the 'ROSCORE' and 'MAIN' devices. Consequently, the 'MAIN' computer processes the range data sequence sent by rostopic from 'ROSCORE' to obtain the 3D data of the scene so a 3D point cloud can be rendered. This setup was chosen because the range data are less heavy, so they occupy less bandwidth than 3D point clouds when transmitted. Therefore, an additional data compression step does not need to be added to our implementations. The main task of the MAIN computer is to launch the recognition process to determine what objects can be detected in the scene.

The recognition and data acquisition processes were distributed into two computers, 'MAIN' and 'ROSCORE'. There are two reasons for this implementation. First, the RGBD sensor is mounted at the robot end-effector and it works with a limitation of distance caused by its USB interface. Second, two computers provide a modular implementation and then the acquisition algorithm is less dependent on the processing algorithms.

Furthermore, in the MAIN computer, a Graphical User Interface (GUI) was implemented using the 'rviz' tool. This GUI allows users to monitor and track the robot movement and the general views from an external observer (Figure 3a) to the task and the viewpoint of the sensor (Figure 3b). Thereby, the viewpoint of the sensor determines what the sensor is capturing and how the objects are observed from its viewpoint. The general views determine how the scene is seen from an external observer who is not participating in the operation or task. The GUI visualizes the working environment and each of the involved items in 3D. The view of the work environment is reconfigurable from the GUI, so the user can change the viewpoint by means of rotations, translations and zoom operations.

Figure 4. Our detection and location method. (a) Details of the recognition process to identify objects and surfaces during a task. (b) General steps executed by the system in a monitoring task of grasping.

GUI presents two main displays in Figure 3. The bottom row is a real-time capture from the Kinect camera mounted on PA-10, and the top row is a skeleton of the robot with the main coordinate systems, such as robot base O_{rb} and end-effector O_{re}, in the current posture. Jointly, a 3D point cloud showing a snapshot of the render object from Kinect has been included in the same figure. Some initial assumptions were made about how our system is able to work. In the scene, only the presence of one object is allowed (it must be isolated and separated from other objects) and it must be static (i.e., motionless relative to the robot base coordinate system). We made that assumption, inasmuch as the system goal is to test the recognition of some isolated objects, to classify them according to shape and to determine their pose within the world scene. But, at, present, some improvements are being performed to recognize several objects simultaneously.

4. Proposed recognition process

The recognition method was implemented using a software development methodology framework structured on two levels of abstraction; both of them are shown in Figure 4. These levels of abstraction are required to generalize by reducing the information content and to show only the blocks with relevant information. The first layer is the recognition process (Figure 4a). This layer is only computed in the initial frame and under demand, when required. This process is required in two cases: a) in the initial step, to recognize and localize a new object in the scene and b) if the scene changes because of new objects placed in the scene while the grasping task is being developed. The second layer is the main loop (Figure 4b), which explains how the recognition process and the global object pose estimation are computed.

The "*global object pose estimation*" considers both the local pose estimated in the fourth step of the recognition process and the knowledge of robot kinematics. That is to say, the system knows the type of object and where it was located in a previous image f_{t-1}. Additionally, the robot movement from one position to another is known from the "*robot kinematics*". Therefore, the object pose for a new image f_t can be computed from the initial recognition process by applying spatial transformations. Thus, once the objects are clustered in the image f_t by means of the "*acquisition & segmentation*" sub-process, the system is able to estimate the global pose of a whole object relative to the world coordinates system set O_{rb} at the PA-10 first joint, applying the transformation of the robot kinematics according to:

$$\mathbf{T}_{rb}^{o} = \mathbf{T}_{rb}^{re} \times \mathbf{T}_{re}^{s} \times \mathbf{T}_{s}^{o} \tag{2}$$

wherein $\mathbf{T}_{rb}^{re}$ is the transformation matrix between the robot end-effector O_{re} and the world coordinates O_{rb}, $\mathbf{T}_{re}^{s}$ is the transformation between the optical sensor O_s and the end-effector and $\mathbf{T}_{s}^{o}$ is the transformation between the object pose O_o and the optical sensor coordinate system O_s. Later, a further transformation is calculated to determine the object pose in the current image f_t relative to the reference frame, which is the model pose for that object previously stored in our database.

$$\mathbf{T}^{o}_{rb_{relative}} = \mathbf{T}^{o}_{rb_t} \times \mathbf{T}^{o}_{rb_{t-1}} \tag{3}$$

wherein $\mathbf{T}^{o}_{rb_t}$ is computed by equation (2) for the current f_t, and $\mathbf{T}^{o}_{rb_{t-1}}$ is computed in the last iteration f_{t-1}. Thus, each new pose of an object is always computed from a previous pose that had been determined from the recognition process, applying the set of transformations suffered by the robot when it is moved. For this reason, the system does not need to compute a whole traditional pipeline recognition process when there are small changes in the scene. In this way, the proposed approach looks for a relationship between accuracy and time in the recognition process to be used in real time systems.

The main loop is used to calculate the Euclidean distance between the centroids of the object and its model stored in our database to compare their positions. If this distance is greater than a threshold, λ, the recognition process must be executed again, but only including the comparison with the previously selected model and not from all of the models in the database. In our experiments, a good threshold was 0.05 m. We assume that λ determines two situations: a) the object is the same but its position slightly changed. It was moved when the grasping process was executed; b) the object has been replaced by another or the recognition process has failed. In this last case, the whole 'Recognition Process' is executed again.

The recognition process consists of four steps (Figure 4a). These steps allowed us to define a hierarchy and dependence among implemented modules: acquisition, segmentation, classification and local estimation pose. Each represents a program component that defines functions and control items in our approach.

4.1 Acquisition

To obtain the best relationship between accuracy and computation time, two depth-image acquisition streams were launched, as seen in Figure 4a. Both are captured by the same sensor at the same time. The first is a low-resolution stream, QQVGA $(160px \times 120px)$, and the second is a high-resolution stream, VGA $(640px \times 480px)$. Obviously, the advantage is to decrease the computation time, inasmuch as the data are directly captured from a sensor with the suitable specific resolution without requiring additional processing steps such as stored data within the *octree* data structure. This is a software strategy often used for sampling the point cloud P at several detail levels. In contrast, the strongest point of our approach lies in using a processing multi-thread for acquiring two depth-image streams together.

The acquisition step is a single process used to obtain a 3D data structure "*compute cloud*". Its purpose is to compute, from a depth-image, an organized points cloud P as well a called depth-map, inasmuch as each pixel contains depth information such as a Euclidean distance d_{ij} as well as sensor intrinsic information $\mathbf{K}$, equation (4). P is stored in matrixes n and m, where it can be either $(160,120)$ or $(640,480)$ depending on the resolution used.

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 1 & 0 \end{bmatrix} \tag{4}$$

as (f_x, f_y) is the focal length and (c_x, c_y) is the principal point of the camera. Therefore, the points of P can be calculated as follows:

$$p(i,j) = \begin{cases} p_z(i,j) = d(i,j) \\ p_x(i,j) = (i - c_x) \cdot d(i,j) \cdot f_x \\ p_y(i,j) = (j - c_y) \cdot d(i,j) \cdot f_y \end{cases} \quad , i < n \wedge j < m, p \in P \tag{5}$$

4.2 Segmentation

In this step, the resolution used by the sensor is mainly QQVGA. This resolution is enough to achieve a successful result in the segmentation process. This process is composed of five processing steps: i) normal vector estimation, ii) segmentation, iii) object clustering, iv) convex hull and v) crop hulls.

Our approach uses the optimization-based method presented in [17] for the "*normal vector estimation (Integral method)*". It was chosen because it has a very efficient computation and the computational costs are not dependent of the area of the region used to estimate the normal vector. This method is based on the concept of the integral image. An integral image $\mathbf{I}$ corresponding to an image $\mathbf{I}'$ allows us to calculate the sum of all values of image within a certain rectangular region by accessing only four data elements in the memory by equation (6).

$$\mathbf{I}(i,j) = \mathbf{I}'(i,j) + \mathbf{I}(i-1,j) + \mathbf{I}(i,j-1) - \mathbf{I}(i-1,j-1) \tag{6}$$

The normal vector $\mathbf{n}_p$ at a point $p(i,j)^T$ can be computed as the cross product between two director vectors: a vector that describes the horizontal line with the left neighbour called $\mathbf{v}_{p,x}$ and a vector that describes the vertical line with the top neighbour called $\mathbf{v}_{p,y}$ (Figure 5), as follows:

$$\mathbf{n}_p = \mathbf{v}_{p,x} \times \mathbf{v}_{p,y} \tag{7}$$

Figure 5. Representation of director vectors

Once the normal vectors were computed, the "*worktable segmentation*" is performed by means of the Trevor method [18]. In that work, the segmentation problem is accurately and quickly solved by means of fitting points of the cloud by planes defined on the Hessian-normal form whose representation is $ax + by + cz + d = 0$. Then, this algorithm works by splitting up P into a set of point groups called segments S that are identified with a label. In this way, each point $p(i,j)$ that belongs to each S has the same label denoted as $l(i,j)$. To assign labels, the point cloud is compared point to point with the comparator C_1, defined as follows:

$$C_1(p,q) = \begin{cases} true & if\,(\mathbf{v}_p \cdot \mathbf{n}_q < \lambda_n) \wedge (|d_p - d_q| < \lambda_r) \\ false & otherwise \end{cases} \tag{8}$$

wherein $p,q \in P$, both λ_n and λ_r are thresholds, $\mathbf{n}_p = (a,b,c)$ is the normal vector to the plane in p and the unknown d_p is the d variable on the plane $ax + by + cz + d = 0$, which is calculated as $d_p = p \cdot \mathbf{n}_p$. Similarly, the unknown d_q is obtained. This is conducted for all points by examining their neighbouring, as:

$$p(i-1, j) \wedge p(i, j-1) \tag{9}$$

Assuming that the objects to be recognized must be sustained on a surface, in our case, the worktable, the object clustering process is carried out by means of processing of the survival points. Those are points that do not lie on the plane of the worktable, although its projection p' must to lie over this plane, i.e., $p(i, j)$ is a point belonging to the object surface over the worktable plane when its projection satisfices $p' = p - d' \cdot \mathbf{n}_p$, where $d' = \mathbf{v} \cdot \mathbf{n}_p$ and $\mathbf{v}$ is the vector from point to plane. Thus, if components p'_x and p'_y are inside the boundaries of the plane, then point p lies over the plane.

Finally, the points are grouped into clusters, i.e., points in the same cluster have the same label. To label a pair of points, it goes through P using equation (9) and is compared with the predicate C_2. This predicate determines if two points belong to same group or category.

$$C_2(p,q) = \begin{cases} true & if\,\|p,q\|_2 < \lambda_d \\ false & otherwise \end{cases} \tag{10}$$

Where λ_d is the allowable distance between two points p and q. This distance depends on the density of the cloud point. In this work, the minimum distance between two points is 1mm, and then λ_d was set to ten times that distance as tolerance value.

Subsequently, two processes are executed, one after another. They are the *"convex hull"* and *"crop hull"* processes. Both modules are implemented as a part of the segmentation process but they use point clouds with different resolutions (Figure 4a). Both point clouds are structured because they were acquired from a RGBD sensor and stores like a matrix. Therefore, the points are ordered and they are easily accessible from rows and columns without using the Euclidean position in the space. Then, we can extract the points of the 2D region occupied by the objects with a 2d processing instead of using a 3d processing. Hence, the convex hull process computes a 2D convex polygon for each clustered object. This polygon wraps the set of clusters that establish an object. Additionally, each polygon consists of a set of points called vertex $v(x, y)$. Then, the convex polygons are used to extract the objects from P by means of the crop hull process. This process filters points that are lying inside of these convex polygons.

4.3 Classification

To perform the classification process, three processing steps are used: i) normal vector estimation, ii) feature descriptor estimation and iii) matching.

Initially, the normal vectors of the surface of each clustered objects are computed as discussed in the previous section *"normal vector estimation (integral method)"*. Remember that the clustered objects were obtained from the crop hull process. The normal vector estimation for the database objects models was computed using the average method *"normal vector estimation (average method)"*. This method obtains the eigenvector with the lowest eigenvalue of the covariance matrix. This 3×3 matrix represents the variation of the surface distribution, and it can be computed by equation (11).

$$C=\frac{1}{k}\sum_{i=0}^{k}(p_i-\hat{p})(p_i-\hat{p})^T,\ \hat{p}=\frac{1}{k}\sum_{i=0}^{k}p_i \tag{11}$$

where the number of points k is determined by the neighbourhood radius, which was experimentally chosen as 0.03m. In this case, the method used for calculating the normal vectors is different because it depends on the type of view. Here, the views are obtained from CAD models of the database using a virtual camera. The virtual views establish unstructured point clouds, and the real views captured from a real sensor establish structured point clouds. Finally, the feature descriptor for each view is computed similarly in both cases, real and virtual views.

OUR-CVFH feature descriptor was chosen from all the surface descriptors commented upon in Section 2 for two reasons. OUR-CVFH avoids calculating the angle of the roll rotation for the orientation. Additionally, this feature descriptor provides better recognition rates than other global feature descriptors, as was shown in [15]. OUR-CVFH defines a histogram that enables describing the object shape. Note that the classification process requires that an offline process have previously been conducted to build the knowledge database. The database stores the views for each object model, and each view is saved as an OUR-CVFH descriptor. Therefore, the matching process is a comparison process among the OUR-CVFH of the clustered object and the set of OUR-CVFH for each object model. Moreover, each object model was saved as a set of OUR-CVFH, one for each view registered of the model.

The comparison is performed by means of a *kdtree* data structure implemented with FLANN. It is a library for searching for the nearest neighbour according to the appearance similarity in high-dimensional spaces. At this point in time, the objects are not classified yet. In fact, the classification is carried out by a *"local pose estimation"* step. Just like that, the matching step returns an ordered list. This represents the best matching among the clustered object and the views saved of the object model in the database. Equation (12) is used to compare the descriptor associated with clustered object C_j with some descriptor of the object models O_i.

$$F(O_i,C_j)=\min_{o^{ik}\in O_i}\left(\left\|o^{ik},C_j\right\|_1\right) \tag{12}$$

wherein $o^{ik} \in O_i$ is the descriptor associated with view k of model i.

4.4 Local pose estimation

Supposing we have, similar to our hypothesis, an ordered list with the k best matches among a clustered object and the views of the objects model. This step consists of i) the align process and ii) the hypothesis verification process.

First, the *"align"* process is carried out by using an ICP (Iterative Closest Point) algorithm [19] without the use of texture, in contrast with work [18]. This algorithm is well known in SLAM (Simultaneous Localization And Mapping) problems in mobile robotics. Basically, the ICP algorithm computes the closest point, i.e., for each point p of the clustered object, look for the closest point in the hypothesis. Later, it computes the transformation needed to go from point p to its closest point q (known as registration). Next, this transformation is applied to the whole set of points of the clustered object. Finally, the algorithm is terminated when the mean-square error is smaller than a threshold λ_{icp} that defines the desired precision of the registration. This algorithm is launched for each and every hypothesis.

Finally, the "*hypothesis verification*" is performed, similar to [20] but using a greedy strategy. In brief, this process searches for the best hypothesis, which has the best relationship between the matching score and a smaller transformation that generates the lowest movement. The strategy consists in two steps. First, the list of candidate objects is sorted from the matching scores. Second, the first object within the list that fulfils that e^2 is less than $\lambda_{\mathbf{H}} = |O_i|^{-1} \sum e^2 - 2\sigma$ is selected:

$$e^2 = \sum_q \sum_k \min_p \left(\frac{1}{n} \sum_{d=1}^{n} \| q_d - \mathbf{H} p_d \|^2 \right) \tag{13}$$

Where $\mathbf{H}$ is the homography matrix between two three dimensional points, then n is 3. This homography matrix represents a three-dimensional similarity transformation. Also, q and p are points of the object and of the model view, respectively; and $|O_i|$ is the number of candidate objects, σ is the standard deviation of the transformation errors.

5. Results and discussion

The goal of this section is to measure the recognition capability of the proposed system to be applied in industrial environments in looking for the recognition of geometric pieces without texture, with same colour and made of the same material. To do it, an object database is created from CAD models, and some experiments are performed to recognize this type of geometric objects as part of a monitoring process to help in a robotic grasping process. In spite of what people may think, the simplicity of the shapes increases the complexity of recognition from partial

views due to ambiguity (the object cannot bet seen fully). Generally, processes with many changes are recognized more easily because there are more geometric differences in partial views.

5.1 Creating database

For the experimentation, a set of five basic objects has been used. They can be grouped as follows: i) objects without curved surfaces, ii) objects with both curved and non-curved surfaces, and iii) objects without non-curved (planar) surfaces. Both cubes and prisms represent objects of the first group. The second group consists of a cylinder and a cone, and the third is a sphere.

The reason for this selection of objects was to find objects that have neither information of colour nor texture, and thereby they have a high level of ambiguity, such as in many manufacturing processes.

The database stores 210 views for each model, as shown in Figure 6(a). They have been created by means of a virtual camera positioned at the vertices of an icosphere that has 42 vertices and 80 faces. Figure 6(c) shows the arbitrary camera poses represented by green tetrahedrons, and Figure 6(b) is the representation of the render point cloud created for each view captured from the different camera poses at the icosphere. In order to release the recognition pipeline of a pre-filter step, Gaussian noise has been added for each render point cloud because the RGBD sensor used in this work has significant levels of noise. The database also stores the camera pose used for each virtual view.

Figure 6. (a) Object Models stored in the database. (b) Partial views of the object surfaces according to the viewpoints of the sensor. (c) Three arbitrary camera poses to extract the render view shown on the column b)

5.2 Object recognition tests

In the robotic grasping process, the robot effector performs a path from its initial position to desired position. This path can change depending on the location of the object in the scene. The initial position is determined by the resting pose, and the desired position is estimated as is the most suitable position to carry out grip tasks. Often, the vision system is used to control the robot positions during the performance of the path as was shown in [21] where a robot is equipped with range sensor at the end of the robot close to the effector for visual servoing task. However, here, the vision system must also monitor the scene during the path followed by the robot effector in order to check that there are not changes in the type or position of objects. The recognition results during the development of this type of task are shown in this section.

Figure 7. Simulation of the nearest neighbours (within the environment of the features extracted from the views of the models) of the object features captured from the camera.

Five different objects are located in the same position in the scene in five different tests. For all of them, the robot performs the same path, which is repeated 5 times. Some robot poses are arbitrarily selected within the trajectory to measure the performance of the proposed recognition system. In

these positions, a comparison between the classification output of the proposed system and the expected classification is performed.

The proposed recognition system is dependent on the rank. The rank is the number of the models views selected to find matches in the classification step (see section 4.3) and it is noted here as NNs (Nearest Neighbour views) (Figure 7). Therefore, NNs determines the number of matches between camera view and models views sorted by similarity level. Then, when the rank is equal to 1, the matcher only returns one model view. When the rank is equal to 2, the matcher returns two model views, and when the rank is equal to n, the matcher returns n model views. This parameter provides more flexibility because the algorithm output is not subordinated to a unique solution that could be a false positive in the recognition process. For this reason, an appraisal of the system is required to test its accuracy dependent on the rank used.

Figure 8. Results of the recognition process (a) Confusion matrices using NN2. (b) Confusion matrices using NN16.

Figure 8 shows the recognition process results for 43 different arbitrary camera poses when the rank used is $k = 2$ and $k = 16$. Note that each is shown as two different confusion matrices. On the left side is shown the confusion matrix with the number of classes that are retrieved by the system. Although the number of predicted classes is the same for all tests, the system is not always able to return a classification output. In this case, the classification process could not be performed with a success rate. For example, Figure 8(a) shows 23 outputs of 42 for a cylinder. In contrast, on the right side of figure is shown the results of the test weighted over the total number of camera poses. In both cases when *k* is 2 or 16 the sphere class gives the best results, but the rest of object classes have not the same behaviour. The reason is because the sphere is very different to the rest of the objects. It has not vertices and edges. But, if the objet geometry is similar then the ratio of the matching process is move away from best. This fact occurs in the cases of cube and prism. In this case, the size of the neighbourhood determines the accuracy of matching. Thus, NN2, which considers 2 neighbour views, obtains better result than NN16, which are considered 16 views. The size of neighbourhood represents the views of the objects more similar to the model. Therefore, if k increase then the differences among views and models is greater. The system takes more views to do the matching process and the error increases. The bigger is k there are more likely that the new views have not a suitable representation of the object.

5.3 Object location tests

Furthermore, the robotic manipulation tasks require that the vision system compute not only the type of the object but also its pose, that is, the position and orientation of the object in the scene. For the evaluation of this capacity, the robot has been moved along some trajectories in 3D Cartesian space. Thus, a typical approximation trajectory to carry the end-effector to the object for a grasping task is shown in Figure 9(b)-8(d). In this case, the robot effector describes a path as shows Figure 9, that, it was previously computed by MoveIt [22] tool. More specially, Figure 9(b) shows a gap between the robot effector and the object location. This gap is due to limitations of sensor FOV (field of view). When the range sensor works closer than 0.6 meters, then the object is

lost because it is outside its FOV. Hence when the object is lost the recognition system ends in favour of a control system without visual information.

More in detail, Figure 9(c)-9(d) show the location accuracy by means of error analysis for two tests of recognition: a test for objects without curved surfaces (Figure 9(c) and 9(d) left) and another for objects with curved and non-curved surfaces (Figure 9(c) and 9(d) right). In these cases, cube and cylinder have been the type of objects chosen, respectively. In particular, these graphs show the position and orientation errors. The error in position is defined as:

$$\varepsilon = \sqrt{(x_t - x_s)^2 + (y_t - y_s)^2 + (z_t - z_s)^2} \tag{14}$$

where x_t, y_t, z_t is the real position of the object relative to the robot base, and x_s, y_s, z_s is the position estimation relative to the robot base.

Figure 9. Results of the location process without the recognition step (a) Robot workspace with an object to grasp (b) Path of robot effector with a sensor mounted at the end. (c) Position error for both tests, cube and cylinder (d) Orientation error for both tests.

On the one hand, the average position error for both cube and cylinder is between 0 and 0.015 m. The greatest peaks, approximately 0.03 m are caused by a bit problem with the processes synchronization for linking the points cloud and the proper information about the sensor pose. This problem is a consequence of UDP messages used to send data among some processing nodes in our LAN topology. This type of transmission is not an acknowledgment message service to avoid delays. In future, we expect to implement the transmission between MAIN PROCESS and PA 10 SERVER to control the flow and error of the transmission packets.

On the other hand, the orientation errors depend on the shape complexity that obstructs a suitable object recognition process. Therefore, issues such as the ambiguity views or lack of edges among sides which determine the object complexity, affect to the pose estimation process. Consequently, the result shows errors greater for objects of type i) than objects of type ii).

The objects with curved surfaces of type i), such as surfaces of revolution, cause errors greater than other objects of type ii) such as polyhedrons (cubes, pyramids, prisms, etc.). In general, the surface of objects created by rotating a curve around an axis is smooth and their gradients change little between neighbour points. On the contrary, the polyhedrons have flat polygonal faces, edges and vertices. The gradient slightly change if it is computed in the faces but it change abruptly if it is close to edges or vertices. Against more abrupt can be the variations the better representation has the descriptor. This is, because the used descriptors in this manuscript are strongly dependent on the normal vector to the surface and they describe the variation among normal vectors.

5.4 Runtime and performance

To demonstrate the use of the proposed recognition system in the development of tasks in real time, this section shows a runtime analysis of our proposal. In addition, the runtimes allow comparing and quantifying the performance of the different modules of the system. This is especially critical in a system used for recognition and monitoring of intelligent robotic

manipulation tasks.

Figure 10. (a) Runtime of the process for the best case, with NN2. (b) Comparison of times depending on NNs

Two tests have been designed to help us quantitatively evaluate the performance. The first one is used to determine the worst case in the recognition process if the parameter evaluated is the runtime. This is what type of object spends the highest runtime (Figure 10(a)). The second one is designed to evaluate the behaviours of the recognition system depending on the number of nearest neighbours for the recognition (see section 5.1) and what impact will be in the runtime if it is increased (Figure 10(b)). Both results are obtained for all objects in a database. In addition, the recognition process is divided in the most critical subprocess or threads. Figure 11(a) shows more different levels of nearest neighbour views. It is possible to view that the accuracy does not change significantly between different ranks.

The experiments demonstrate that the runtime grows proportionally up to the number of the NNs (rank). In addition, the align subprocess of the "*Local pose estimation*" of the algorithm (see section 4.4) is slower than the rest and consequently it is the most dependent. In particular, the test (Figure 10(b)) shows that it is more than 6 times lower when 16 neighbours are used instead of 1. That is because the method must align all the NNs with the object to be recognized. Summarizing, the results show that a NNs between 1 and 4 are a good selection in terms of runtime.

Additionally, the results show that the runtime is not only dependent on the number of NNs used but also dependent on the feature descriptor estimation step. This fact is a trace of the dependence between the descriptor and the number of points of the surface in each view. Although all objects have a similar volume, the curved surfaces usually have more points in each view because there is more visible surface.

Once known, these two capabilities of our recognition system, a high accuracy and a low runtime, it becomes relevant to know the relationship between both them. Thus, Figure 11(b) illustrates this relationship. It represents the relationship *accuracy vs 1 - fps'* where the parameter *fps'* is the inverse of frames/seconds normalized using the weight of the rate maximum the capture sensor, in our case 30.

Figure 11. (a) Comparison of the accuracy results depending on NNs (b) Accuracy vs runtime.

The recognition system is slower and less accurate when the representation of its test is shown lying around the bottom-right corner. Positions against, the recognition system is fastest and more accuracy when its representation lies on the top-left corner. Then, the best recognition system in terms of runtime was when $\mathrm{NNs} = 1$. In contrast, the best one in terms of accuracy was when the number of NNs is 5. Finally, in terms of accuracy and runtime together, the best is with $\mathrm{NNs} = 2$.

6. Conclusions

This paper has presented an approach for the implementation of 3D perception systems for robot arms. The results reveal that the implemented 3d recognition system achieved a high success rate to recognize geometric objects made of the same material and without using additional information such as the colour or texture of surfaces, in contrast to other works in the literature where these features are used. The lack of these features is very common in industrial applications of manipulation.

The recognition process was tested in sensing experiments, detecting and localizing objects when an interpretation of a scene is required in intelligent handling and grasping tasks. Thus, the proposed method enables the detection of the object from arbitrary partial views and its spatial position and orientation relative to the robot used in the handling tasks. The method proposed is not dependent of external factors like changes in brightness, luminosity and other typical handicaps of the traditional computer vision used in the industry. It can work in scenes with bad light and/or with shadows and it retrieves the 6 DoF pose of the geometric objects in real-time from a view partially.

An analysis of the recognition experiments reveals that there is a significant correlation between the success rate and the object shape complexity. Thus, sometimes simple objects with few faces, such as objects with revolution surfaces, can cause ambiguity. This can be corrected if the number of views selected from the knowledge database is incremented. This is to choose a high number of NN. Similarly, there is a significant correlation between NN and runtime. Many comparisons in the matching process slow down the 3d recognition system, and consequently, there could be a delay in the robotic handling tasks.

It is not the purpose of this work to supervise the task during the grip process, only throughout the approach path. We consider that the object does not change when it is being manipulated. Besides, the proposed system mixes both supervision and recognition processes. Both sensor data and robot kinematics are used to determine the objects pose and to estimate where the object will be in the next iteration in the robot path.

Future possibilities for improvement lie in extending the study to deformable objects. These were not considered in this study, in which the objects have always been rigid and homogeneous surfaces (i.e., without texture).

Summarizing, the proposed visual system works in real time and its methods and algorithms allow us to identify 3D geometric pieces although they don’t have appearance information and/or the location in the workspace.

Acknowledgements

The research leading to these result has received funding from the Spanish Government and European FEDER funds (DPI2012-32390) and the Valencia Regional Government (PROMETEO/2013/085).

Compliance with Ethical Standards

Funding: This work was funded by DPI2012-32390 and PROMETEO/2013/085.

Conflict of Interest: The authors declare that they have no conflict of interest.

Ethical responsibilities: The authors declare that they have read and understand the ethical responsibilities to publish in IJAMT. This manuscript has not been submitted to other journals for simultaneous consideration and it has not been published previously.

References

1. Ferreira M, Costa P, Rocha L (2014) Stereo-based real-time 6-DoFwork tool tracking for robot programing by demostration. Int J Adv Manuf Technol, 2014, doi:10.1007/s00170-014-6026-x
2. Fuchs S, Haddadin S, Keller M, Parusel S, Kolb A, Suppa M (2010) Cooperative Bin-Piching with Time-of-Flight Camera and Impedance Controlled DLR Lightwight Robot III. The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 4862-4867.
3. Shmukler A, Fischer A (2009) Verifiaction of 3D freeform parts by registration of multiscale shape descriptors. Int J Adv. Manuf Technol. doi:10.1007/s00170-009-2447-3
4. Zheng XJ, Wang YS, Teng HF, Qu FZ (2008) Local scale-based 3D model retrieval for design reuse. Int J Adv Manuf Technol. doi:10.1007/s00170-008-1701-4
5. Papazov C, Haddadin S, Parusel S, Kriege K, and Burschka D (2012) Rigid 3D geometry matching for grasping of known objects in cluttered scenes. SAGE. The International Journal of Robotics Research (ijrr), vol. 0, no. 0, pp. 1-16, doi:10.1177/0278364911436019.
6. Ciocarlie M, Hsiao K, Gil-Jones E, Chitta S, Rusu R, Sucan I.A (2014) Towards reliable grasping and manipulation in household. In Experimental Robotics, Springer tracts in advanced robotics; O. Khatib, V. Kumar, G. Sukhatme, Eds.; Publisher: Springer-Verlag Berlin, Germany, Volume 79, pp. 241–252.
7. Duncan K, Sarkar S, Alquasemi R, Dubey R (2013) Multi-scale superquadractic fitting for efficient shape and pose recovery of unknown objects. IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, pp 4238-4243.
8. Sun K, Heß R, Zhihao X, Schilling K (2014) Real-time robust six degrees of freedom object pose estimation with a time-of-flight camera and a color camera. Journal of Field Robotics, doi:10.1002/rob.21519.
9. Pfitzner C, Antal W, Hess P, May S (2014) 3D Multi-Sensor Data Fusion for Object Localization in Industrial Applications. In Proceedings of ISR/Robotik, 41st International Symposium on Robotics, pp 1-6, ISBN: 978-3-8007-3601-0
10. Adán A, Merchán P, Salamanca S (2011) 3D scene retrieval and recognition with Depth Gradient Images. Pattern Recognition Letters, vol. 32, no. 9, pp 1337-1353, doi:10.1016/j.patrec.2011.03.016
11. Bellandi P, Docchio F, Sansoni G (2013) Roboscan: a combined 2D and 3D vision system for improved speed and flexibility in pick-and-place operation. International Journal of Advanced Manufacturing Technolog, v69, pp 1873-1886.
12. Aldoma A, Marton ZC, Tombari F, Wohlkinger W, Potthast C, Zeisl B, Rusu RB, Gedikli S, Vincze M (2012) Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation. IEEE Robotics & Automation Magazine, 1070 (9932/12), pp80-91.
13. Rusu R.B., Bradski G., Thibaux R., Hsu J. (2010) Fast 3D recognition and pose using the Viewpoint Feature Histogram. In Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, Issue Date: 18-22 Oct. 2010, pp. 2155 - 2162, ISBN 978-1-4244-6674-0
14. Salti S, Tombari F, Di-Stefano L (2014) SHOT: Unique signatures of Surface and texture description. ELSEVIER. Computer Vision and Image Understanding, vol. 125, pp. 251-264, doi:10.1016/j.cviu.2014.04.011
15. Aldoma A, Tombari F, Rusu R, Vincze M (2012) OUR-CVFH – Oriented, Uniue and Repeatable Clustered Viewpoint Feature Histogram for Object Recognition and 6DOF Pose

Estimation. In Pattern Recognition. Joint 34th DAGM and 36th OAGM Symposium, Graz, Austria. Proceedings; v.7476, pp.113-122.
16. Mateo CM, Gil P, Torres F (2014) A Performance Evaluation of Surface Normals-Based Descriptors for Recognition of Objects using CAD-Models. 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Viena, Austria, pp428-435.
17. Holzer S, Rusu RB, Dixon M, Gedikli S, Navab N (2012) Adaptive Neighborhood Selection for Real-Time Surface Normal Estimation from Organized PointCloud Data Using Integral Images IEEE/RSJ International Conference on Intelligent Robots and Systems, Portugal. doi: 10.1109/IROS.2012.6385999
18. Trevor A, Rusu RB, Christensen HI (2013) Efficient organized point cloud segmentation with connected components, in: Proceedings of the 3rd Workshop on Semantic Perception,
19. Besl PJ, McKay ND (1992) A Method for Registration of 3-D Shapes. IEEE Transactions on pattern analysis and machine intelligence, vol. 14, no. 2, pp.239-256, IEEE Log Number 9102686
20. Mian A, Bennamoun M, Owens R (2006) Three-Dimensional model-based object recognition and segmentation in cluttered scenes. IEEE Trans. PAMI (10) doi: 10.1109/TPAMI.2006.213
21. Pomares J, Gil P, Torres F (2010) Visual control of robots using range images. Sensors, vol. 10, no. 8, pp. 7303-7322. doi: doi:10.3390/s100807303
22. Sucan IA, Chitta S (2011) “MoveIt!”, [Online] Available: http://moveit.ros.org

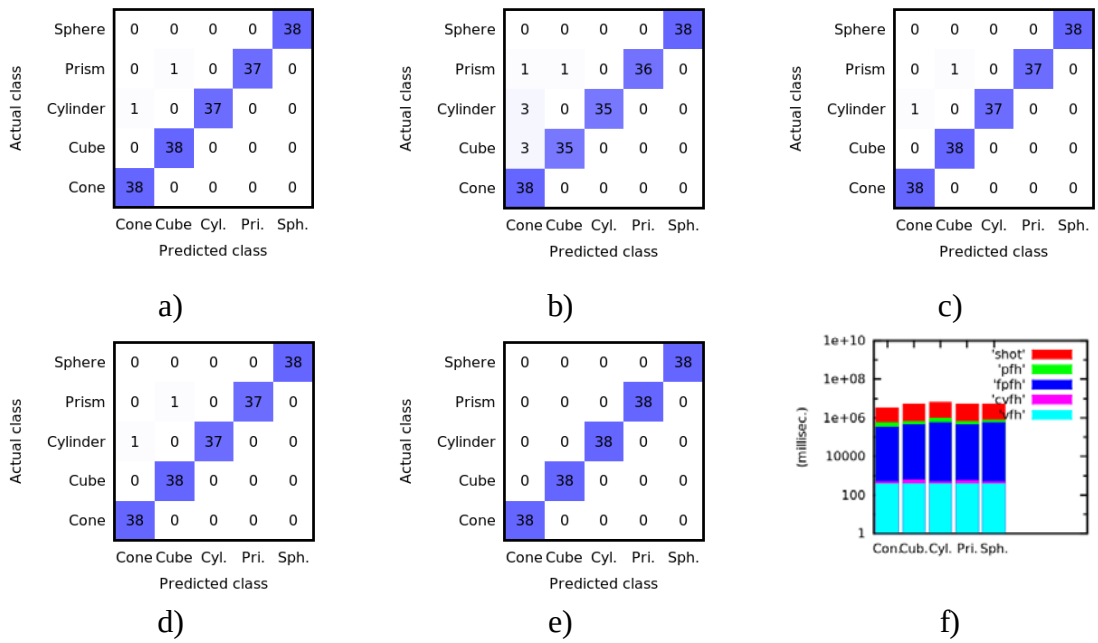


Figure 1. Confusion matrices show the classification capability of a descriptor for: **(a)** PFH **(b)** FPFH **(c)** VFH **(d)** CVFH **(e)** SHOT and **(f)** the runtime cost. The dataset used for the tests is shown more in detail in the experiments of section 5.

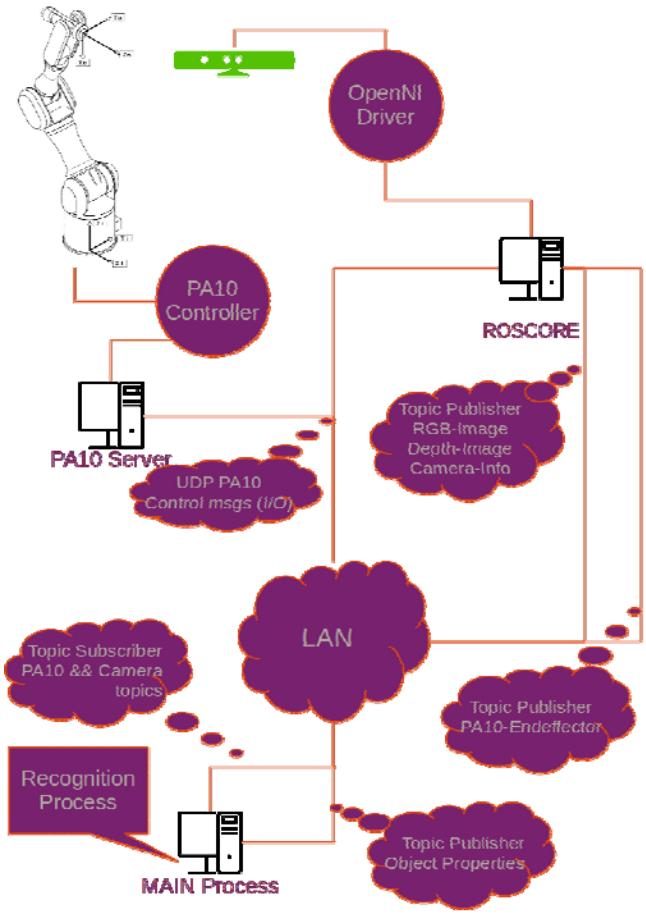


Figure 2. Distributed architecture of a robot vision system.

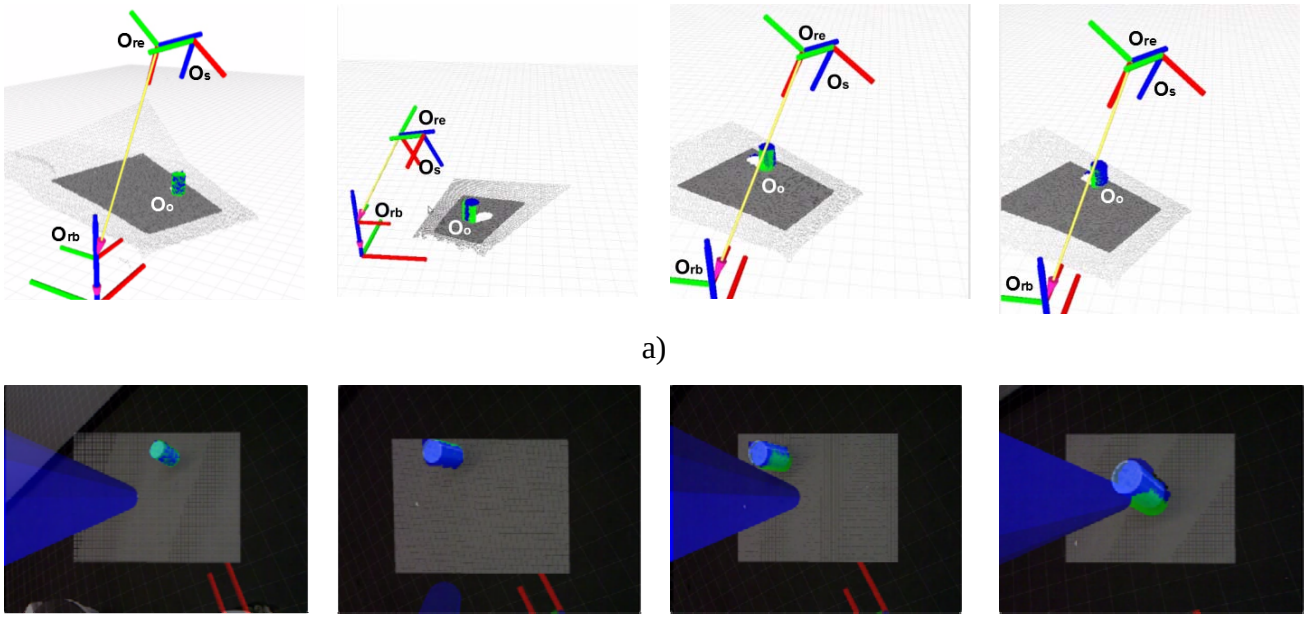
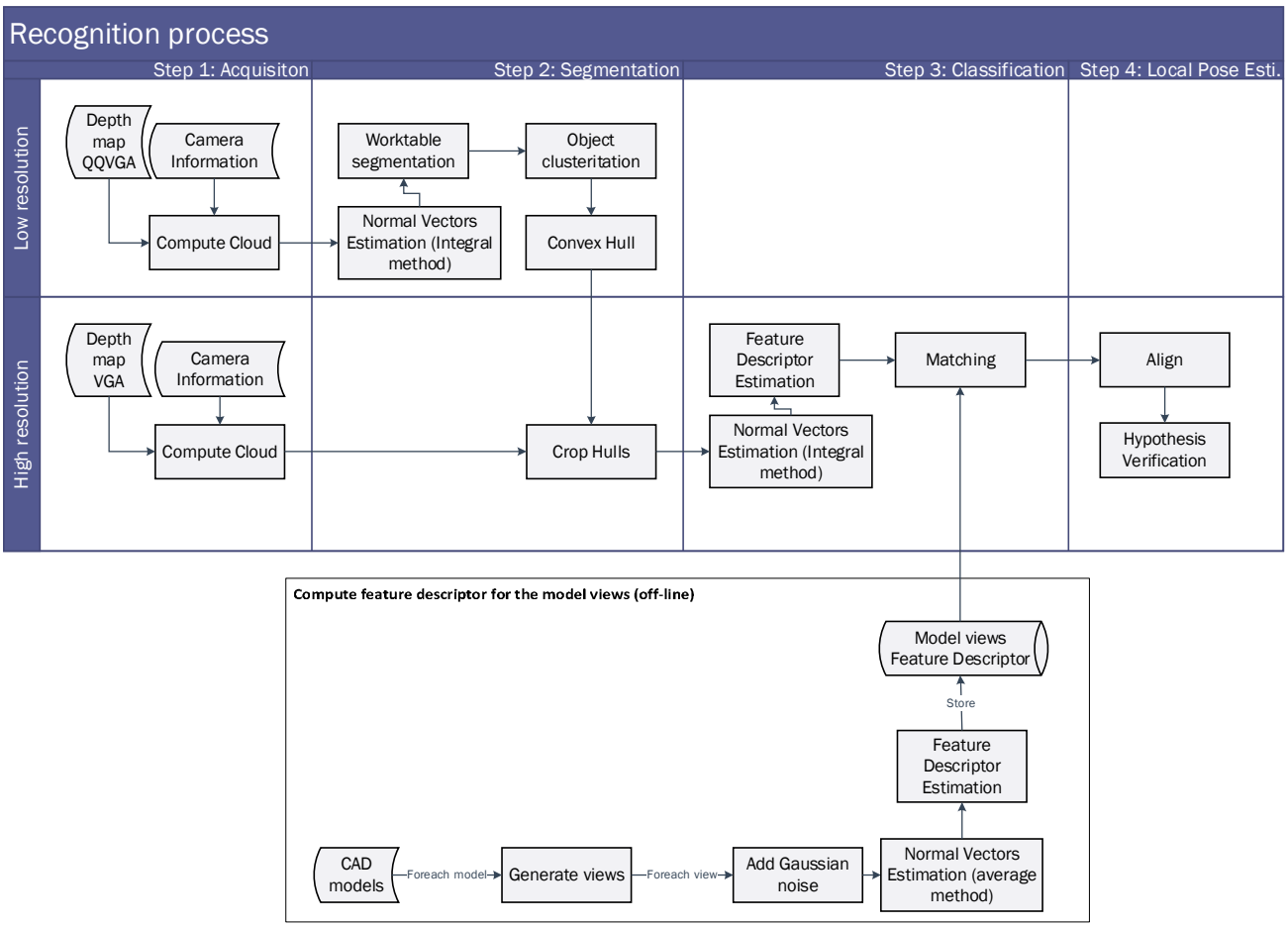
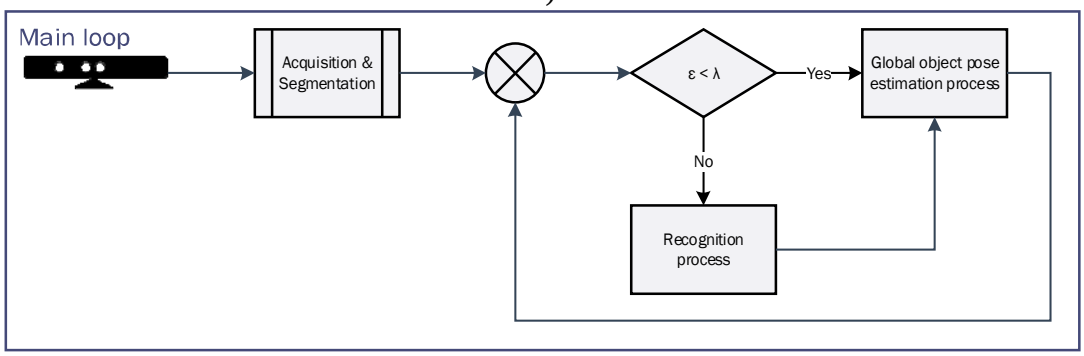


Figure 3. (a) Mathematical transformations among robot-vision and segmented scene. (b) 3D image processing of the scene from a sensor.



a)



b)

Figure 4. Our detection and location method. (a) Details of the recognition process to identify objects and surfaces during a task. (b) General steps executed by the system in a monitoring task of grasping.

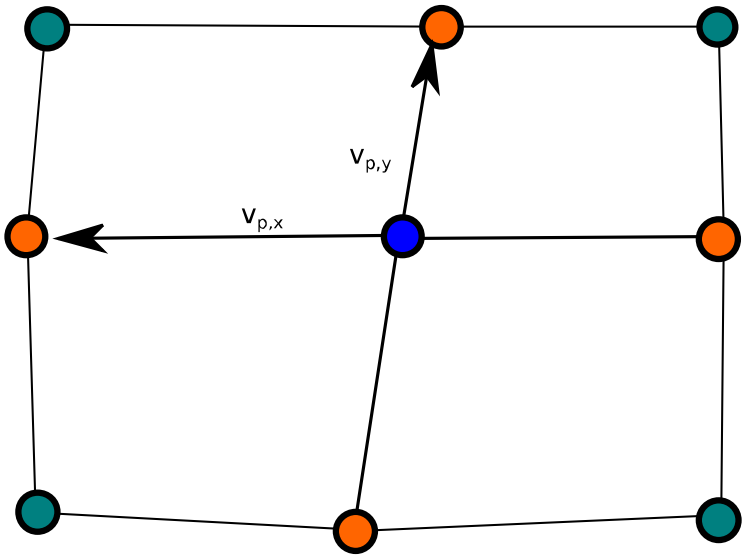
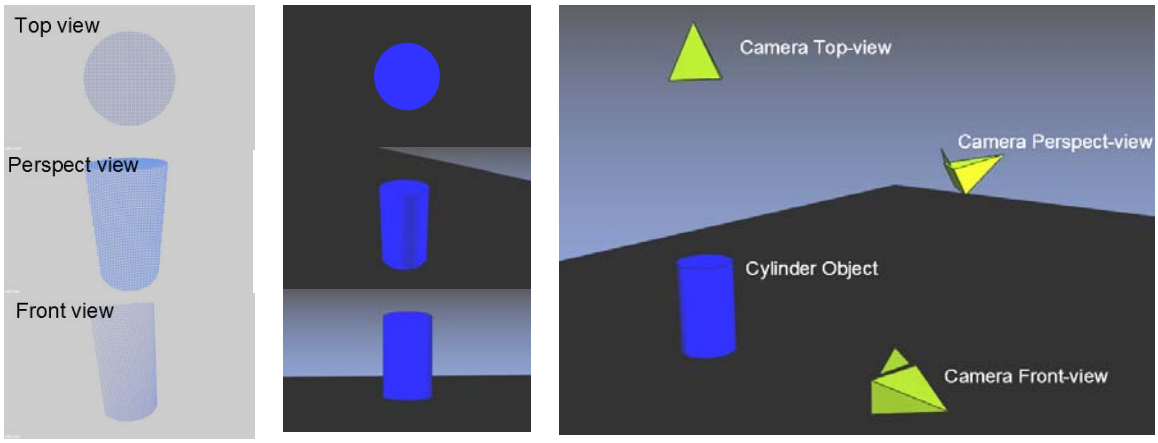


Figure 5. Representation of director vectors



a) b) c)

Figure 6. (a) Object Models stored in the database. (b) Partial views of the object surfaces according to the viewpoints of the sensor. (c) Three arbitrary camera poses to extract the render view shown on the column b)

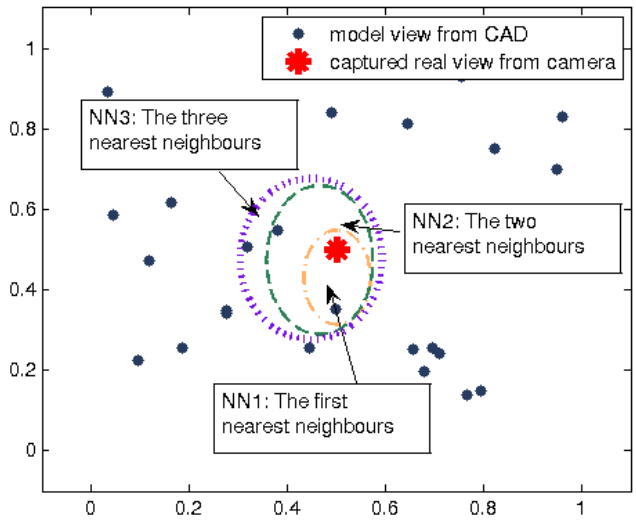
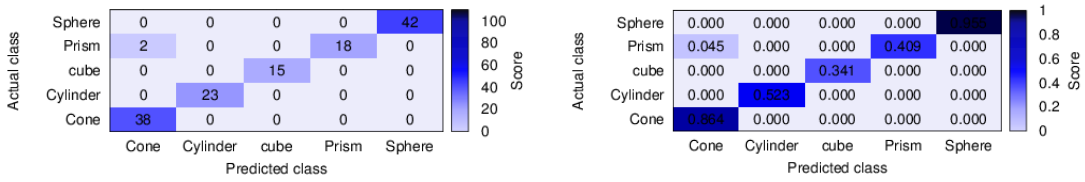
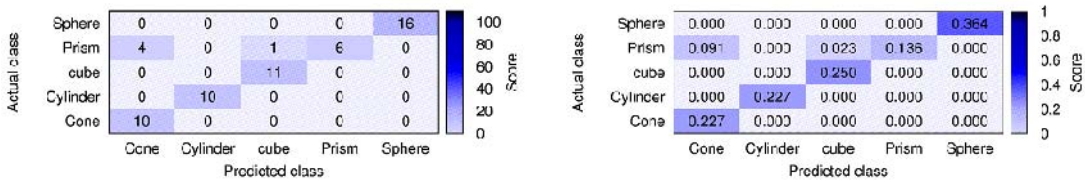


Figure 7. Simulation of the nearest neighbours (within the environment of the features extracted from the views of the models) of the object features captured from the camera.



a)



b)

Figure 8. Results of the recognition process (a) Confusion matrices using NN2. (b) Confusion matrices using NN16.

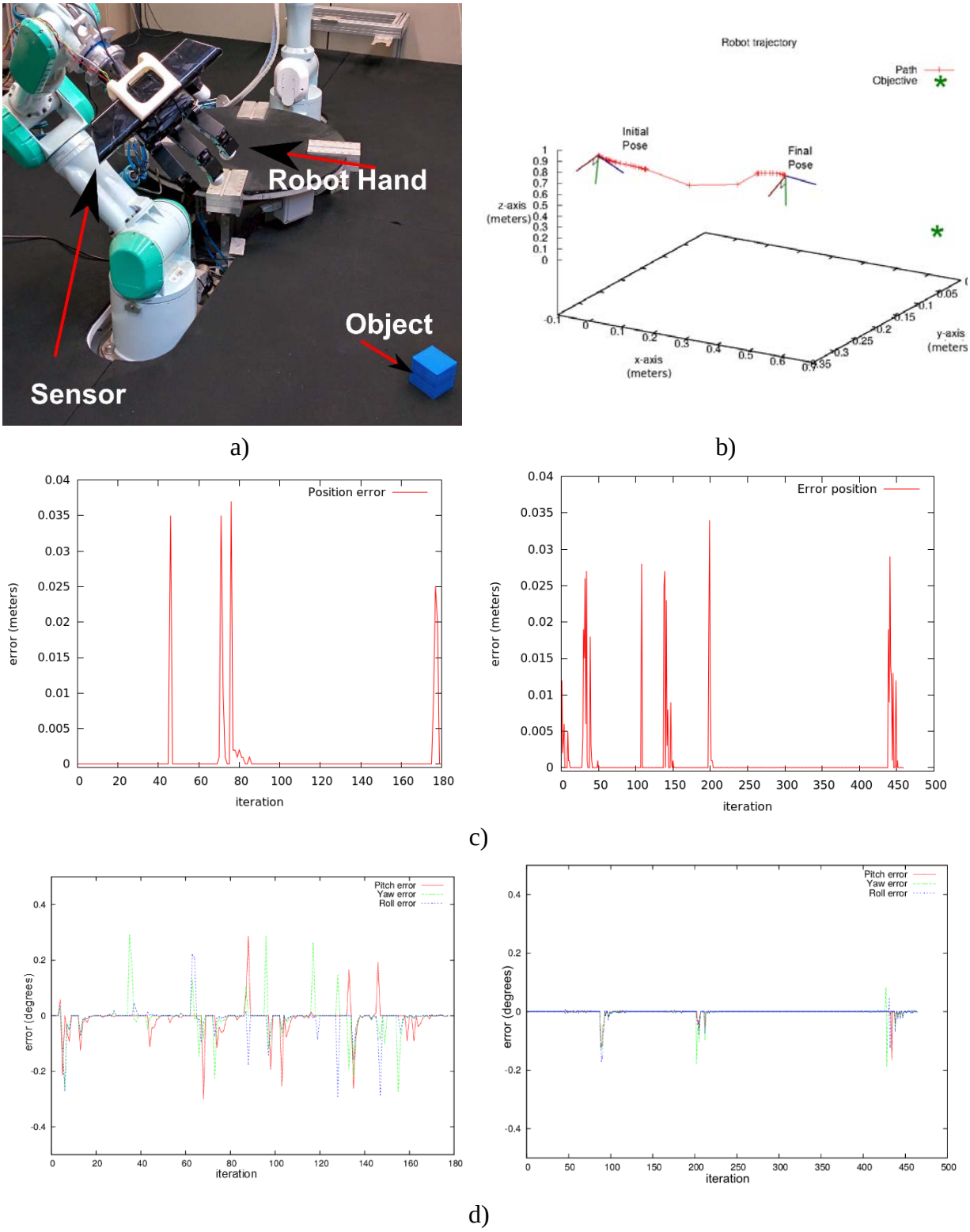
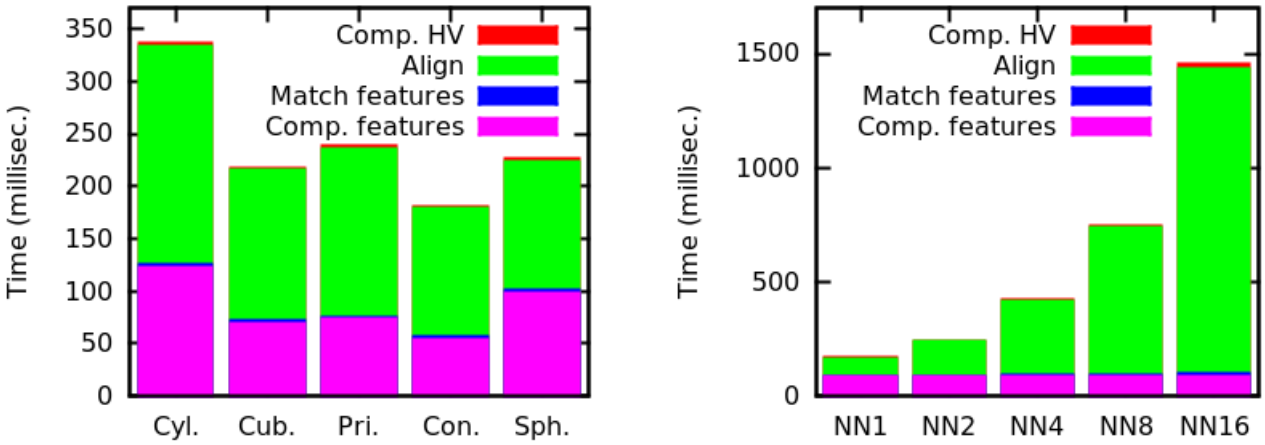
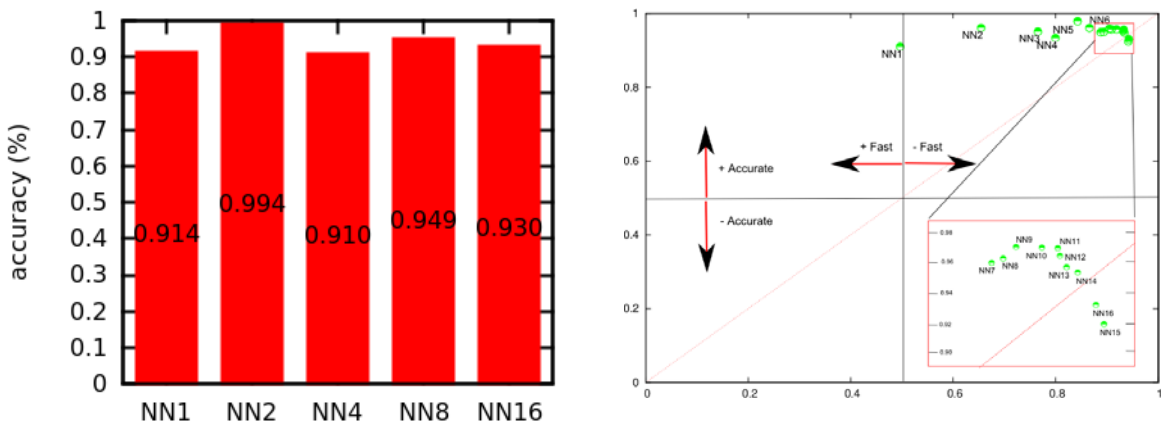


Figure 9. Results of the location process without the recognition step (a) Robot workspace with an object to grasp (b) Path of robot effector with a sensor mounted at the end. (c) Position error for both tests, cube and cylinder (d) Orientation error for both tests.



a) b)

Figure 10. (a) Runtime of the process for the best case, with NN2. (b) Comparison of times depending on NNs .



a) b)

Figure 11. (a) Comparison of the accuracy results depending on NNs (b) Accuracy vs runtime.