The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Representations for Object Grasping and Learning from Experience

Oscar J. Rubio, Kai Huebner, and Danica Kragic

***Abstract*— We study two important problems in the area of robot grasping: i) the methodology and representations for grasp selection on known and unknown objects, and ii) learning from experience for grasping of similar objects. The core part of the paper is the study of different representations necessary for implementing grasping tasks on objects of different complexity. We show how to select a grasp satisfying force-closure, taking into account the parameters of the robot hand and collision-free paths. Our implementation takes also into account efficient computation at different levels of the system regarding representation, description and grasp hypotheses generation.**

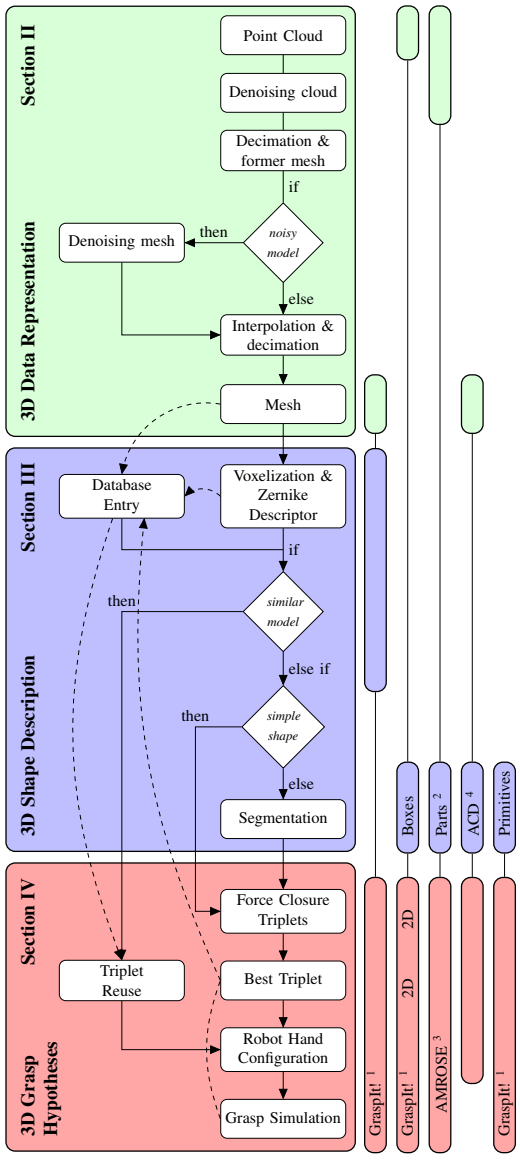
I. INTRODUCTION

A robot grasping cycle involves data representation, shape description and grasp hypotheses generation, Fig. 1. Most of the recent work on robotic grasping relies on 3D data although there are approaches producing grasp hypotheses using 2D image features, e.g. [1]. A range of state-of-the-art methods synthesize 3D object shapes from point clouds by using superquadrics [2] or other shape primitives such as boxes [3]. Assuming that an arbitrary point cloud has to be approximated, a single primitive is obviously not enough for many objects. The more complex the shape is, the more primitives have to be used to represent its individual parts. Multiple methods approach this problem by a variety of segmentation methods, [2], [3], [4].

A major issue is that for *unknown objects*, grasps need to be evaluated from data a robot can extract on-line. This is a difficult problem due to the (i) high dimensionality of the problem, (ii) incomplete and uncertain information about the environment and the objects to be grasped, and (iii) lack of generalizable measures of quality for grasp planning, i.e. (*"What is a good grasp?"*).

Many systems rely on object recognition and/or shape registration. This requires a database of objects or shapes, as for example in [5], or objects combined with grasps, [6]. To approach the recognition problem, an object has to be described using a shape descriptor meeting some desirable requirements: it should be primarily able to cope with real-time and real-world requirements: it should be compact, invariant under transformations, and fast to calculate. An example are the Zernike descriptors, [7].

Overall, there has been a lot of work on grasp planning on different levels: path planning [8], planning on 3D mesh models [9] and databases [6], planning on shape primitives [10], [5]. A general problem is that for any kind of single-view 3D sensor system, a generated point cloud of an object or a scene is not complete. Grasp selection may also benefit from assessing the shape complexity. If complexity is high, it is worthy to segment the object into graspable parts. The segmentation of a model into its parts is also necessary for task-constrained grasping of simple objects.



[1] GraspIt!: a robot grasp simulator [14]
[2] Parts: Range image segmentation + pairwise matching + merging/update
[3] AMROSE: `http://www.amrose.dk`
[4] ACD: Approximate Convex Decomposition

Fig. 1. Structure of our approach, and related approaches' foci.

This work was supported by EU through the project CogX, IST-FP6-IP-027657; GRASP, IST-FP7-IP-215821 and Swedish Foundation for Strategic Research. The authors are with the Computer Vision and Active Perception Lab, Centre for Autonomous Systems, School of Computer Science and Communication, KTH, Stockholm, Sweden. {ojrm,khubner,danik}@kth.se.

However, the decision if an effort in terms of a segmentation or part-based decomposition is in fact worthy is often not treated in the literature. Rarely, a set of differently granulated decision criteria and representations is evaluated to see if some partitioning of the object in fact is reasonable or not.

In this paper, we present a system for robot grasp selection that copes with known and unknown objects where the focus is not to find the most stable grasp, but a grasp that is force-closure and feasible, i.e. collision-free and constrained by the parameters of the robotic hand, given available sensor data.

II. 3D Data Representation

In our work, the input to the system is a sparse or dense 3D point cloud. We then use a mesh representation to interpolate and assign volume to the data. In the following subsections, we describe: denoising, fast point cloud decimation and interpolation, mesh building, convex hull computation and convexity estimation.

A. Denoising

Before mesh building, we perform noise removal using the ANN library [15]: we first calculate d_i as the sum of distances to the 10 nearest neighbors for each point i. We then obtain the number of points n_{iso} which hold $d_i > 3 \cdot mean(d)$ and tag them as isolated vertices to calculate the ratio $r = n_{iso}/n \in [0, 1]$. If $r < 0.99$, we perform noise removal. After building a mesh (Section II-C), we delete faces with longest edges, since residual noisy points will be linked by long edges to the surface of the model. For this reason, the perimeter of all the faces of the mesh will be calculated and those ones whose perimeter is more than twice as long as the average will be deleted. Consequently, if there were groups of noisy faces, they will be isolated and removed from the model. This may cause cracks and holes in the mesh, therefore it is advisable to perform an interpolate-decimate step.

B. Fast Point Cloud Decimation and Interpolation

Obtaining a mesh from a point cloud is a non-trivial task: dense data results in high-quality meshes, but needs more time to be processed. A sparse set of points may be processed fast, but can result in incomplete or erroneous meshes.

1) Decimation: We limit the input point cloud O to a minimum number of points n, and decimate large point clouds by using a filter, preserving constant density in the output point cloud. This increases the accuracy of the axis extraction and benefits the grasp search as well: since all the faces in the mesh representation will be similar sized, all the areas of the object have similar probability of becoming grasp points. The filtering is performed by removing neighbors of each point using kd-tree-search, [15] in a radius

$$rad = \sqrt{surfaceArea(convexHull(O))/(k \cdot n)}, \quad (1)$$

where n is the number of points to keep and the factor $k = 2.43$ was found empirically. We allow a tolerance of 5% to the requested number of output points. In case of dense point clouds, a random reduction is performed first to speed up the decimation.

2) Interpolation: The above procedure ensures a good result by decimating the most redundant points in the cloud but it does not create new points in sparse areas. We approach those cases taking advantage of the mesh representation by (i) iteratively subdividing all the triangles of the mesh, until the number of vertices exceeds a number n, (ii) iteratively subdividing only those triangles t_i with $perimeter(t_i) > 1.2 \cdot mean(perimeter(t))$, and (iii) decimating the resulting cloud to n points. To subdivide a triangle into four similar sized triangles, we use triangle edge bisection. In the evaluation section, we will motivate our choice to set $n = 2000$.

C. Mesh Building

The mesh building process enables the subsequent voxelization and the grasp search including optional segmentation as well. Being more than a pure requirement, meshing is important since the quality of the mesh greatly affects the estimated quality of the final grasp. We use a PowerCrust-based algorithm [16] to acquire a tightly closed *triangular* mesh for each point cloud. We stress the importance of performing the manifold extraction, i.e. deleting badly oriented triangles and ensuring that all remaining triangles are roughly parallel to the surface.

An inappropriate meshing will cause omitting of existing surfaces (false negatives) or adding non-existing surfaces (false positives) in the grasp search. However, both problems can be treated: the first can be detected by checking the percentage of points from the point cloud included in the mesh; in our approach, we reject meshes with a confidence value below 90%, leaving a margin of 10% for small details and/or outliers. The second problem can be avoided or at least minimized after simple post-processing.

D. Convex Hull Computation and Convexity Estimation

We estimate the minimum convex hull enclosing the cloud and its volume using an implementation of the Quickhull algorithm [17], which has shown itself as the most efficient. Based on the mesh, we can measure the convexity of our model. If the object is convex, the volume of the object mesh and the volume of its convex hull are equal. Moreover, the more concave an object O is, the lower the convexity ratio

$$conv = volume(O)/volume(convHull(O)) \in [0, 1]. \quad (2)$$

We exploit this ratio to evaluate the complexity of an object: the simpler the object shape, the closer its convexity to 1.

III. 3D Shape Segmentation

The segmentation divides complex shapes into simpler, independently graspable parts. We consider a body to be complex if its convexity measure (2) is lower than 0.85. Different object segmentation methods have been proposed in the literature: spectral clustering [18], minimum volume bounding box (MVBB) decomposition [3], hierarchical segmentation based on primitives [19]. However, most of these are either time-consuming or not suitable for integrating with grasp selection. We develop a new algorithm for segmentation, fulfilling the requirements to be grasp-oriented, fast,

simple and robust. We use core extraction as a starting point, add a system to carry out several cuts in the mesh using a criterion to find out which one segments best. Although this strategy is similar to MVBB decomposition, there are two main differences: first, we use convex hulls instead of boxes for the decomposition in order to get higher flexibility; and second, our starting point is the hull enclosing the core instead of a box surrounding the whole object.

Our algorithm works as follows:

1) Center-of-Mass Extraction: Assuming that the density in an object O is constant, we estimate the volume integrals by tetrahedrons generated from the mesh representation. Based on the tetrahedrons, we approximate the center of mass of O. First, the center point of each tetrahedron t_i is computed from the ith mesh triangle $\Delta_i = (\mathbf{v}_1(i), \mathbf{v}_2(i), \mathbf{v}_3(i))$ by

$$\mathbf{c}(t_i) = (\mathbf{v}_1(i) + \mathbf{v}_2(i) + \mathbf{v}_3(i) + \mathbf{p})/4 \tag{3}$$

where $\mathbf{p}$ is a random point inside the model. Secondly, we calculate the each tetrahedron's signed volume

$$V(t_i) = (\mathbf{v}_1(i) - \mathbf{p}) \cdot ((\mathbf{v}_2(i) - \mathbf{p}) \times (\mathbf{v}_3(i) - \mathbf{p}))/6 \tag{4}$$

before averaging the distances using the signed volume of each tetrahedron to approximate the object's center of mass

$$\mathbf{c}(O) = \sum_{i=0}^{n} V(t_i)\mathbf{c}(t_i) / \sum_{i=0}^{n} V(t_i) \tag{5}$$

Note that for (4), we have to assure that all triangles in the mesh are defined with the same orientation, clockwise or counter-clockwise. The simple steps result in a good approximation even with a very reduced number of input points, avoiding overestimation of denser areas in the cloud.

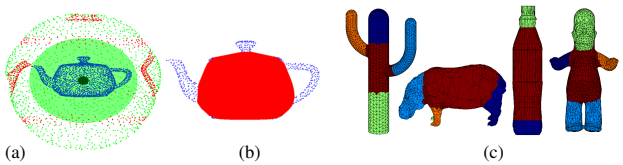
2) Core Extraction via Spherical Mirroring: Spherical mirroring aims at reversing the situation, in such a way that vertices of the core become external and easily extractable. To achieve this, all vertices of the mesh are mirrored on a minimal bounding sphere. Thereby, vertices of the core component are identified as residing on the convex hull of the mirrored vertices. With $\mathbf{c}(O)$ the center of the sphere, the mirrored vertices $\mathbf{v}'$ are given by

$$\mathbf{v}' = \mathbf{v} + 2\tfrac{r - d(\mathbf{v})}{d(\mathbf{v})}(\mathbf{v} - \mathbf{c}(O)) \text{ , with} \tag{6}$$

$$d(\mathbf{v}) = \|\mathbf{v} - \mathbf{c}(O)\| \text{ and } r = \max_{\mathbf{v}} d(\mathbf{v}) \text{ .} \tag{7}$$

An example for core extraction is shown in Fig. 2 (a)-(b).

3) Cut Trials: We associate core points with their corresponding triangles in the mesh. Then, we apply an algorithm based on the connectivity filter of the graphical library VTK to cut the mesh into parts. We found that the size of the core enclosing the hull is usually not optimal to break the mesh in a suitable way, thus the process is repeated scaling the size of the hull by several scales within a range between 1 and 2. The best segmentation is defined by the scale minimizing the sum of convex hull volumes over all parts. Some models segmented using this method are presented in Fig. 2 (c).



(a) (b) (c)

Fig. 2. Core segmentation by spherical mirroring. (a) original point cloud, center of mass, mirroring sphere (tightly enclosing the teapot) and mirrored points. (b) Core (solid) and points outside (dotted). (c) Some examples.

4) Cut Refinement: After segmentation, the different parts are studied and catalogued as *graspable* or *non-graspable*. We define two requirements for *graspable* parts as (i) being larger than 1/100 of the size of the whole object, and (ii) having some curvature, since flat regions are not suitable for grasping. The non-graspable parts in a segmentation are iteratively merged with their closest neighbor.

IV. 3D Grasp Hypotheses Generation

Obtaining a 3D grasp hypothesis for a given object is the primary purpose of a grasping system. The two major policies to acquire are either (i) the search for and selection of best candidate for a new object, or (ii) the adaptation of a learned grasp on a similar or familiar object.

A. Generation of Force-Closure Triplets

The search for force-closure property is aimed at collecting a list of triplet candidates, where each triplet representing the the fingertip positions of a three-fingered robotic hand should reach on the object's surface to result in a stable grasp. We base our search on the method described in [20], adapting the idea from four-fingered hands to three-fingered hands. We acquire a set of N contact triplets using Algorithm 1.

We highlight that the probability of selecting a grasp with stability higher than the average human grasp quality is very

Algorithm 1: Triplet-from-Mesh Computation.

input : Set of triangle mesh $M = \{(\Delta_1, ..., \Delta_m\}$, their outwards normals $\mathbf{n}(\Delta_i)$ and the friction coefficient μ.
output: Set of triplets $\{T_1(M), ..., T_N(M)\}$
begin
  for $n \leftarrow 1$ **to** N **do**
    Choose random Δ_r and center as 1. contact point: **finger 1**
    $\mathbf{f}_1 \leftarrow (\mathbf{v}_1(r) + \mathbf{v}_2(r) + \mathbf{v}_3(r))/3 : r = rand(1, m)$
    Sample a ray $\mathbf{r_1}$ departing from $\mathbf{f}_1$ and deviating from the negative surface normal of Δ_r, using the friction cone angle as standard deviation: **finger 2**
    $\mathbf{r_1} \leftarrow rotate(-\mathbf{n}(\Delta_r), \alpha)$:
    $\alpha = normrnd(0, \tan^{-1}(\mu))$
    Find the intersections of $\mathbf{r_1}$ with the mesh M:
    $\{\mathbf{i}_1, ..., \mathbf{i}_l\} \leftarrow intersect(\mathbf{r_1}, M)$
    If there is more than one such point, we choose one at which the surface is penetrated outwards:
    $\mathbf{f}_2 \leftarrow$ **if** $(l = 1)$ **then** $\mathbf{i}_1$ **else** $chooseOutwards(\mathbf{i}_1, ..., \mathbf{i}_l)$
    Sample a ray $\mathbf{r_2}$ perpendicular to the line given by $\mathbf{f}_1, \mathbf{f}_2$ with origin at the middle point: **thumb**
    $\mathbf{r_2} \leftarrow\perp (\overline{\mathbf{f}_1\mathbf{f}_2})$
    Find the intersections of $\mathbf{r_2}$ with the mesh M:
    $\{\mathbf{i}_1, ..., \mathbf{i}_l\} \leftarrow intersect(\mathbf{r_2}, M)$
    If there is more than one such point, we choose one at which the surface is penetrated outwards:
    $\mathbf{f}_0 \leftarrow$ **if** $(l = 1)$ **then** $\mathbf{i}_1$ **else** $chooseOutwards(\mathbf{i}_1, ..., \mathbf{i}_l)$
    $T_n(M) \leftarrow (\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2)$ **triplet**

high after $N = 100$ randomly generated grasps, from [21]. Since that study was done with few objects corresponding to primitives shapes, thus different from our input data, we extend our search to $N = 400$ grasps and stop at the first sample exceeding a quality threshold. If none of the grasps exceeds this threshold, the best one is taken. We choose the number of triplets to be generated on each part to be proportional to its relative size.

Taking advantage of the mesh representation which provides faces in a mesh and their corresponding normals, it is possible to evaluate the quality of a grasp created from a given triplet. Since contact points and their normals are thus given, we can easily approximate the friction cones, estimate the convex hull of the Grasp Wrench Space and obtain a quality measure.

B. Reuse of Triplets

When dealing with similar shaped objects, we want to reuse the stored grasp hypotheses. This is realized by finding the affine transformation that resizes and reorients the original model M to match the current object O, allowing for the same with its corresponding triplet. The information we need is the original triplet, $T(M) = (\mathbf{f}_{M,0}, \mathbf{f}_{M,1}, \mathbf{f}_{M,2})$, the center-of-mass ($\mathbf{c}$), the volume ($V$) and the main axes of inertia ($\mathbf{A}$) of both M and O. Since we obtain high quality meshes after interpolation-decimation, Principal Component Analysis (PCA) is suitable for the latter.

C. Hand Configuration

The next step is to translate a triplet of contact points $(\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2)$ into a robotic hand configuration, which in our case is the Barrett hand [22]. The hand has 10 degrees of freedom: 6 for the pose of the wrist, 1 for the spread angle of the fingers and 3 for the proximal joint angles. The choice of the triplet points according to the method described in IV-A permits obtaining an optimal configuration of the Barrett hand after Algorithm 2.

Algorithm 2: Triplet-to-Barrett Hand Adaptation.

input : Triplet $T = (\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2)$.
output: Hand configuration $(\mathbf{p}, \mathbf{o})$ (pose), (Θ, e) (spread, extension).
begin
Compute the normal vector of T:
$\mathbf{n}_T \leftarrow (\mathbf{f}_1 - \mathbf{f}_0) \times (\mathbf{f}_2 - \mathbf{f}_0)$.
After finding the circle $(\mathbf{c}, r)$ passing through $\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2$ on the plane described by T, acquire the hand configuration:
$(\Theta, e) \leftarrow (0.5 \cdot \measuredangle\, \mathbf{f}_1\mathbf{c}\mathbf{f}_2, r)$.
There are two possible configurations:
$(p, o)_1 \leftarrow (\mathbf{c} + (-0.953 \cdot e + 128.8) \cdot \mathbf{n}_T, -\mathbf{n}_T)$.
$(p, o)_2 \leftarrow (\mathbf{c} - (-0.953 \cdot e + 128.8) \cdot \mathbf{n}_T, \mathbf{n}_T)$.
Choose the one holding that the palm is further from the model (and outside):
$(\mathbf{p}, \mathbf{o}) \leftarrow (p, o)_i$:
$dist(p_i, model) > dist(p_j, model),\ dist(p_i, model) > 0$.

Note that for the Barrett hand, a change in a proximal link implies a change in the corresponding distal link as well. Therefore, we use the linear estimation from [11] to compute $\mathbf{p}$, causing little loss in precision compared to calculating the actual inverse kinematics for the hand.

V. System Evaluation

As a suitable dataset to evaluate the algorithms described in this paper, we apply all models from the Princeton Shape Benchmark (PSB) [23]. Similar to [6], we rescale all 1,815 point-clouds and consider them as graspable toys to get a complete overview of the system's performance. To provide a reference for the efficiency, all processes are performed on a 2GHz dual-core processor laptop, running Ubuntu 9.04.

A. Point Cloud Interpolation-Decimation

The time used in the reduction of large point clouds clearly depends on the number of points n that we want to keep (see Fig. 3 left). The higher n, the more selective we delete and the lower the number of points we remove in each step. We found only one algorithm, the k-means by Huang [24], to result in similar performance. However, the time k-means needs to reduce to more than 1000 points is over 100 seconds. The time spent in the interpolation by using the mesh representation is negligible ($\ll 0.1s$) as it is done by simple triangle edge bisection.

B. Mesh Building

In Fig. 3 center, the time spent to mesh point clouds of different sizes is plotted. We found out that decimated point clouds with more than 3000 points do not result in better depicted meshes, thus we only present the range from 600 to 3000 points in Fig. 3. As point cloud reduction and meshing take longer time as the number of points increases, we determine the minimum necessary size ensuring good meshing. By considering that the confidence of well meshed models does not improve when increasing the number of points, we empirically found 2000 points to be an optimum value for the PSB object models. Meeting this value in Fig. 3 center, we can infer that reduction and meshing of a point cloud can be done in around 1.7 seconds.

C. Shape Descriptors

A comparative study of descriptors, e.g. Fourier descriptors, curvature scale space descriptors, Zernike and grid descriptors, [25], encourages the use of Zernike moments. We evaluate the Zernike descriptors through an object classification experiment on the PSB. Following [7], we use precision-recall diagrams (see Fig. 3 right) to find a good combination $(N_{voxels},\ O_{Zernike}) \in \mathbb{N} \times \mathbb{N}$ for retrieving the shape of a model. We use the diagrams to measure the ability of different combinations to separate sets of objects belonging to a shape class given only one object from a class. The quality measure we use for each class C is the integral of the normalized precision-recall diagram averaged over the members of C, P_o^C. The better the descriptors represent the models of a class, the closer this value to 1.

We selected 6 different classes from the PSB. The purpose of each experiment was to separate 2 or 3 sets of models composed of between 20 and 50 elements (each set), using 4 different voxelizations $N_{voxels} \in \{48, 64, 128, 256\}$ and 21 different Zernike orders $O_{Zernike} \in \{5,...,25\}$.

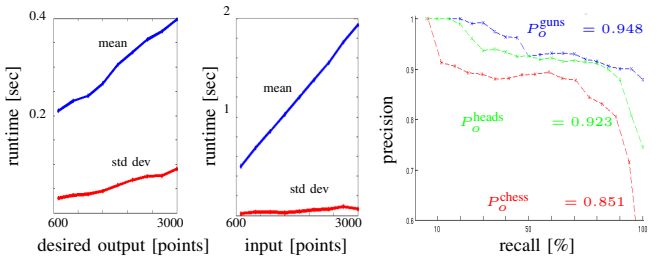


Fig. 3. Performance of point cloud filter (Left) and mesh building (Center). Right: Precision-recall diagram of three classes (guns, heads, chess) from the PSB. In this example, we use $N_{voxels} = 256$, $O_{Zernike} = 25$.

The experiments revealed a good behavior of the Zernike orders, obtaining an average P_o^C value around 0.9. The results suggest that complex shapes are better retrieved when using detailed voxelizations, $N_{voxels} = 256$, while simple objects got better results with simpler voxelizations, e.g. $N_{voxels} \in \{48, 64\}$. Nevertheless, the loss in P_o^C for making the opposite choice is less than 0.05. The optimal range of Zernike order was observed as $O_{Zernike} \in \{10, 11, 12\}$. Since it is thus not possible to get the optimum for both simple and complex shapes at the same time, and detailed voxelizations and high Zernike orders demand more calculations, it was decided to use $N_{voxels} = 64$ and $O_{Zernike} = 12$ (49 coefficients). The time spent to perform both processes with this configuration is 0.5 seconds.

D. Segmentation

The segmentation is the slowest process in our system. Though the mesh is calculated before-hand and core extraction is performed only once, the decomposition is repeated several times to find a variety of segmentations and choose the best one. We consider that testing 10 different scales of the core is sufficient to get a good segmentation in most of the cases. Obviously, including more scales could lead to a better result, but at the cost of computation time.

The lower the number of points to be handled, the faster the segmentation. This puts more emphasis on the cloud decimation described in II-B, limiting the time spent not only for meshing, but even more for segmentation. We found that the average number of segmented parts over the PSB in two test cases (a: $<$2000 points, b: $\geq$2000 points) is similar (a: 2.61, b: 3.34). Nevertheless, the convexity gain (the merit value corresponding to the difference between the convexity measures (2) of the original model and its decomposition) is much higher in the case of large point clouds (a: 0.12, b: 0.26). We associate this observation to the fact that simple objects are represented by lower number of points in the PSB database. Thus, the improvement reached through the segmentation in these models is lower. However, there is a clear difference in computation time (a: 2.93s, b: 6.38s).

E. Grasp Stability Estimation

Four main definitions of grasp quality are implemented according to [14]. Out of those we take a measure of grasp quality (eps_{L1}) that implies we are using a pessimistic criterion and considering one unit force distributed over all grasp points. We assume the friction coefficient μ to be 1, corresponding to a rubber coated hand grasping an object made of metal, glass, plastic or wood.

The percentage of force-closure among all 400 grasps found on each object is typically in the range of 75-80%. The average time for generating a triplet, evaluating its grasp quality and configure the Barrett Hand is 1, 2.8 and 0.2 milliseconds, respectively. This results in 1.6 seconds in the worst case, when we evaluate all candidate triplets. The eps_{L1} value of the best grasp is typically within the range of 0.12 to 0.16. Our acceptance threshold is set to 0.15.

VI. Using Experience in an Object-Grasp Database

An important aspect of our our work is to enable decisions if some process on the object data is reasonable or not. For example, if the object is convex, there is no need for segmentation. Our approach is based on a database in which each entry for an object O is composed of three fields:

- $D(O)$, a description of an object O; we will use its Zernike descriptor (Sec. V-C), the volume of its mesh, its main axes and center-of-mass (Sec. IV-B).
- $T(O)$, a triplet (Sec. IV-A) leading to a grasp on O.
- $Q(O)$, a measure of grasp quality (Sec. IV-A and V-E) connected to the triplet $T(O)$.

Given an object description $D(O)$, the system can search for similar models in the database using the shape descriptor. If a similar model O^* is found, the stored triplet $T(O^*)$ can be adapted to the new object size and orientation. Otherwise, a collection of triplets $\{T(O)\}$ will be created according to the morphology of O and the first $T(O)$ with quality $Q(O)$ exceeding an acceptance threshold will be selected.

Finally, a corresponding entry will be added into the local database. Optionally, the configuration of the robotic (Barrett) hand is computed to align the fingertips with the triplet. As the number of grasp requests grows, the local database becomes more complete and the likelihood of finding similar models stored increases. Since re-using a triplet takes less time than searching for force-closure-feasible triplets on the object surface, the average response time is reduced. We note that our database is hand-independent, since we do not store hand configurations, but triplets of contact points. The time spent to find a similar object in the database and adapt its triplet is negligible (< 0.1 seconds).

In Tab. I, we show the average runtime of the major processes, considering 3 basic cases: (1) known objects, (2) unknown, simple objects, and (3) unknown, complex objects.

TABLE I
Average runtime of the different subtasks on PSB models.

Input model:	[sec]	case (1)	case (2)	case (3)
Denoising & Interp-Decim.	2.95+N	•	•	•
Meshing	1.3	•	•	•
Voxel. & Zernike	0.45	•	•	•
Segmentation	6.4			•
Triplet Search	1.6		•	•
Total [sec]		4.7+N	6.3+N	12.7+N

with $N = 10^{-6}$ * number of points [sec].

The input cloud is firstly randomly decimated to 10000 points (if it exceeds this size) in order to limit N. The table does not include processes with runtime $\ll 0.1$ seconds: search in the database, adding entries or reuse of a triplet. As it can be seen, there is a big difference in timing between cases (1) and (2) with respect to case (3).

VII. Discussion and Conclusion

We have proposed a grasping framework capable of dealing with known and unknown objects considering the acquisition of a good 3D point cloud, the choice of an appropriate 3D shape representation and the management of the experience as the key aspects. The creation and update of a grasp database in order to gain experience is an important part of the system. The idea of building a grasp database is not new: the Columbia Grasp Database (CGDB) [6] is a most recent and attractive repository of grasps over a set of 3D models. Nevertheless, there are three main difference between the CGDB and the grasp database presented here:

Extension: While the CGDB covers a set of 7,256 models (the 1,814 models from the Princeton Shape Benchmark [23] cloned at four different scales), the database proposed here is constructed according to local experience. Only the models that have been processed and grasped by the system will be included in the database. Obviously, when searching for similar models, this results in a much shorter response time, noticeable saving in memory space, and links to strategies of active learning (and active forgetting).

Specific nature: When an object is grasped once, it is likely that the same object or a similar one will be requested to be grasped again (e.g. a book). The first time a *specific* grasp will be generated for that specific object. The next times this object will be re-grasped using the same grasp again, and those objects which are similar will be grasped by adapting the original triplet. In the worst case, if an adapted grasp does not work on a new object, a new *specific* grasp will be generated for that object and a corresponding new entry will be added to the database.

Independence of object size: The grasp is not directly reused, the original triplet is *adapted* to the size (and to the orientation) of the new object instead. Next, the grasp is generated from the adapted triplet.

Regarding future work, there are several ideas to be explored. In this paper, we took into account the constraints given by the hand and the objects, but left out the constraints given by the task (e.g. hand-over, pouring, tool use); this could be included in the grasp hypothesis search. Secondly, in the presented framework we considered that the objects were pre-segmented from the scene. Our current work in stereo based segmentation will be integrated with the system for better point-cloud generation. Then, a path planning algorithm could be added to avoid collisions with the obstacles when approaching the object.

Acknowledgments

This work was supported by EU IST-FP6-IP-027657 PACO-PLUS, EU IST-FP7-IP GRASP and Swedish Foundation for Strategic Research (SSF).

References

[1] J. Bohg and D. Kragic, "Learning Grasping Points with Shape Context," *Robotics and Autonomous Systems*, vol. 58, pp. 362–377, 2010.

[2] C. Goldfeder, P. K. Allen, C. Lackner, and R. Pelossof, "Grasp Planning Via Decomposition Trees," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 4679–4684.

[3] K. Huebner, S. Ruthotto, and D. Kragic, "Minimum Volume Bounding Box Decomposition for Shape Approximation in Robot Grasping," in *IEEE Int'l Conf. on Robotics and Automation*, 2008, pp. 1628–1633.

[4] L. Chevalier, F. Jaillet, and A. Baskurt, "Segmentation and Superquadric Modeling of 3D Objects," *Journal of Winter School of Computer Graphics, WSCG'03*, 2003.

[5] K. Huebner, K. Welke, M. Przybylski, N. Vahrenkamp, T. Asfour, D. Kragic, and R. Dillmann, "Grasping Known Objects with Humanoid Robots: A Box-based Approach," in *International Conference on Advanced Robotics*, 2009.

[6] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, "The Columbia Grasp Database," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 3343–3349.

[7] M. Novotni and R. Klein, "3D Zernike Descriptors for Content Based Shape Retrieval," in *ACM Symposium on Solid and Physical Modeling*, 2003, pp. 216–225.

[8] R. Diankov, S. Srinivasa, D. Ferguson, and J. Kuffner, "Manipulation Planning with Caging Grasps," in *IEEE International Conference on Humanoid Robots*, 2008, pp. 285–292.

[9] S. El-Khoury and A. Sahbani, "On Computing Robust N-Finger Force-Closure Grasps of 3D Objects," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 2480–2486.

[10] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic Grasp Planning Using Shape Primitives," in *IEEE International Conference on Robotics and Automation*, 2003, pp. 1824–1829.

[11] S. Geidenstam, K. Huebner, D. Banksell, and D. Kragic, "Learning of 2D Grasping Strategies from Box-based 3D Object Approximations," in *Proceedings of Robotics: Science and Systems*, 2009, pp. 9–16.

[12] M. Richtsfeld and M. Zillich, "Grasping Unknown Objects Based on 2 1/2 D Range Data," in *IEEE International Conference on Automation Science and Engineering*, 2008, pp. 691–696.

[13] E. Lopez-Damian, "Grasp Planning for Object Manipulation by an Autonomous Robot," Ph.D. dissertation, Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS, 2006.

[14] A. Miller and P. Allen, "Graspit! A Versatile Simulator for Robotic Grasping," *Robotics and Automation*, vol. 11 (4), pp. 110–122, 2004.

[15] S. Ary, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions," *Journal of the ACM*, vol. 45, no. 6, pp. 891–923, 1998.

[16] L. Giaccari, "MyRobustCrust – Surface Reconstruction from Scattered Point Clouds," **[URL]** http://www.mathworks.com/matlabcentral/fileexchange/22185, January 2010.

[17] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The Quickhull Algorithm for Convex Hulls," in *ACM Transactions on Mathematical Software*, vol. 22, 1995, pp. 469–483.

[18] R. Liu and H. Zhang, "Segmentation of 3D Meshes through Spectral Clustering," in *Computer Graphics and Applications, 12th Pacific Conference*, 2004, pp. 298–305.

[19] M. Attene, B. Falcidieno, and M. Spagnuolo, "Hierarchical Mesh Segmentation based on Fitting Primitives," *The Visual Computer*, vol. 22, no. 3, pp. 181–193, 2006.

[20] C. Borst, M. Fischer, and G. Hirzinger, "A Fast and Robust Grasp Planner for Arbitrary 3D objects," in *IEEE International Conference on Robotics and Automation*, vol. 3, 1999, pp. 1890–1896.

[21] ——, "Grasping the Dice by Dicing the Grasp," in *IEEE/RSJ Int'l Conference on Intelligent Robots and Systems*, 2003, pp. 3692–3697.

[22] W. T. Townsend, "The BarrettHand Grasper – Programmably Flexible Part Handling and Assembly," *Industrial Robot: An International Journal*, vol. 27, no. 3, pp. 181–188, 2000.

[23] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The Princeton Shape Benchmark," in *International Conference on Shape Modeling and Applications*, 2004, pp. 167–178.

[24] Z. Huang, "Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values," *Data Mining and Knowledge Discovery*, vol. 2, no. 3, pp. 283–304, 1998.

[25] D. Zhang and G. Lu, "Content-Based Shape Retrieval Using Different Shape Descriptors: A Comparative Study," in *IEEE International Conference on Multimedia and Expo*, 2001, pp. 1139–1142.