Generic Fitted Shapes (GFS): Volumetric Object Segmentation in Service Robotics

Tiberiu T. Cocias*, Florin Moldoveanu, Sorin M. Grigorescu

Department of Automation, Transilvania University of Brasov, Mihai Viteazu 5, 500174, Brasov, Romania.

Abstract

In this paper, a simultaneous 3D volumetric segmentation and reconstruction method, based on the so-called *Generic Fitted Shapes* (GFS) is proposed. The aim of this work is to cope with the lack of volumetric information encountered in visually controlled mobile manipulation systems equipped with stereo or RGB-D cameras. Instead of using primitive volumes, such as cuboids or cylinders, for approximating objects in point clouds, their volumetric structure has been estimated based on fitted generic shapes. The proposed GFSs can capture the shapes of a broad range of object classes without the need of large a-priori shape databases. The fitting algorithm, which aims at determining the particular geometry of each object of interest, is based on a modified version of the *active contours* approach extended to the 3D Cartesian space. The proposed volumetric segmentation system produces comprehensive closed object surfaces which can be further used in mobile manipulation scenarios. Within the experimental setup, the proposed technique has been evaluated against two state-of-the-art methods, namely superquadrics and 3D Object Retrieval (3DOR) engines.

Keywords: Active contours, 3D segmentation, 3D reconstruction, Robot vision systems, RGB-D sensors

1. Introduction

In the last decades, the number of service robotics systems centered on human environments has drastically increased, together the introduction of novel sensors and actuators that push the boundaries of visual perception and mobile manipulation further [1]. Such applications span from common all-day-living assistance platforms [2] to care-giving robots deployed in hospitals and homes [3]. The main goal of a service robot operating in such environments is to autonomously perform an action in order to assist a human person in achieving his/her goal [4]. Among such tasks, autonomous object grasping in mobile manipulation [5] is one of the most researched areas within the robotics community, with imaging and computer vision being a common source of data for performing and improving grasping capabilities [6]. The success or failure of these procedures are directly dependent of the precision through which the imaged objects are reconstructed into the virtual environment of the robot [7].

Reconstructing and segmenting real-world scenes in service robotics scenarios is not a trivial task, especially when the robot perceives the environment from only one perspective. Considering the geometrical complexity of the scene and the uncertainty introduced by the single perspective, the achievement of mobile manipulation tasks is difficult to obtain. Given the challenge of handling complex objects, a robot must deal with both the reconstruction precision, as well as with the computation of the safest and most reliable grasp configuration [8]. The usage of a series of predefined shape models can provide structural information which further enables the estimation of the object of interest's volume. However, such an approach cannot model the particularities of each object to be grasped, thus leading to low grasping configurations.

*Corresponding author. Tel./Fax: +40-268-418-836.

Email addresses: `tiberiu.cocias@unitbv.ro` (Tiberiu T. Cocias), `moldof@unitbv.ro` (Florin Moldoveanu), `s.grigorescu@unitbv.ro` (Sorin M. Grigorescu)

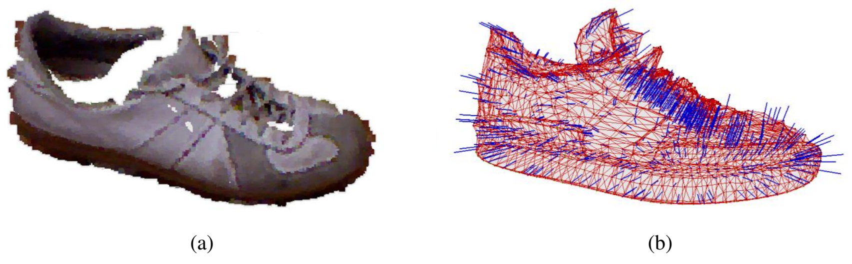


Figure 1: 3D object segmentation via GFS estimation. (a) Object points cluster. (b) Estimated GFS.

In order to cope with such challenges, in this paper, we introduce the so-called *Generic Fitted Shapes* (GFS) concept with the goal to model the full volumetric structure of objects present in mobile manipulation scenarios. A GFS model addresses 3D segmentation by the usage of a small amount of 3D models which are deformed in order to capture the particularities of each object. The deformation principle is derived from the concept of *active contours* [9], well-established in the 2D image segmentation area. Namely, a GFS structure is designed to capture the particular shape of an object by minimizing a set of *internal* and *external* 3D contour energies [9].

The challenge which we tackle lies in the area of 3D segmentation and reconstruction, which implies the partitioning of a certain n-dimensional space (2D digital image, 3D point clouds etc.) into a subset of meaningful regions representing objects of interest and the background of the scene. While for 2D images the most used characteristics are color or texture, for 3D point clouds, the most relevant attributes describing the scene are the surface's geometry, point normals, surface mesh or point density.

The problem of segmenting an object S from a scene point cloud can be regarded as a simultaneous segmentation and reconstruction of its 3D shape, given a sparse *Point Distribution Model* (PDM). An example of PDM segmentation, where the object is approximated through a GFS, is illustrated in Fig. 1.

1.1. Related Work

With the advent of new structured light sensors, such as the MS Kinect®, the 3D segmentation area received a large attention from the computer vision and robotics community [10, 11]. A series of implicit segmentation methods, entitled *superquadrics* aim at generating a 3D model $S(b)$, defined by a series of parameters b, for approximating the volume of the given surface [12]. As opposed to the work in [12], where a single superquadric is used to approximate the shape of an object, in [13], multiple superquadrics are merged with the purpose of enhancing the surface estimation precision. The obtained superquadric structure is a shape, describing the segmented object, which can be further use for grasping or mobile manipulation. Since the object is approximated by 3D geometric primitives (e.g. spheres, cubes, toroids, etc.), the final representation is rough. In [14], the authors calculated a refined object model by dividing the initial primitive into more meaningful subregions. In each subregion, a separate superquadric has been fitted, increasing thus the overall estimation accuracy of the model. The approach can be successfully used for simple regular objects like boxes, simple mugs, or cylindrical bottles. For more complex shapes, the algorithm fails to properly segment the object, delivering in the end over-segmented areas.

In [15], a series of kitchen objects (e.g. plates and mugs) are reconstructed using a similar implicit model formulation as for the superquadrics case. Again, simple geometric primitives (e.g. cylinders, cuboids or spheres) are use to estimate the structure of the objects, thus leading to rough representations.

3D deformable shapes have been extensively treated in the work of Terzopoulos [16], where the authors introduce a 3D deformable balloon primitive model with its shape driven by a series of internal forces (e.g. elastic properties). The introduced dynamical model does not need any a-priori knowledge about the object which must be estimated and, in the same time, no *position and orientation* (pose) normalization is required. The major drawback of this approach is that in the regions where no point information is available (occluded regions), the approximated 3D contour is described by a rounded balloon surface.

In comparison to the above presented methods, the so-called *explicit* approaches make use of a series of predefined models (e.g. particular primitives or models) to better approximate a certain surface [17]. The main idea here is to fit into the point clouds complex 3D shapes which best describe its structure. In this case, an appropriate shape model is searched within a database containing multiple predefined models. The similarity between the scene object and the objects from the database can be established using a *3D Object Retrieval* (3DOR) search engine [18]. The database model, with the lowest convergence error (highest similarity), is further annotated to the scene. The major difference between a 3DOR system and the proposed GFS method lies in the object fitting approach. If, for the case of the 3DOR technique, the selected shape from the database remains static, that is, no adaptation with respect to the nature of the scene is performed on it, in the case of the GFS, the selected shape if automatically deformed in such a way that it captures the particularities of the imaged object.

Other approaches, which exclude the need of active sensors, make use of voxel-based 3D reconstruction techniques from multiple calibrated cameras to obtain an approximated 3D representation of a focused object. In [19], the authors ensures the consistency of the projected reconstruction with the original images by making explicit use of the finite size footprint of a voxel when projecting it into the image plane. The major disadvantage, considering the image based 3D reconstruction techniques, is that multiple views are needed to obtain a full model of the object of interest [20], which, for the case of a service robot, are difficult to obtain.

1.2. Structure and Main Contributions

The main contributions of the paper may be summarized as follows:

- the introduction of the GFS technique based on a 3D active contour formulation; the deformation of the shape model is performed with respect to energies calculated directly in the 3D Cartesian space;
- the usage of a GFS as an initial contour within the active contours framework, thus improving the computation time and avoiding erroneous approximations;
- usage of the GFS approach for building full 3D volumetric models of objects of interest in the context of mobile manipulation.

The rest of the paper is organized as follows. Section 2, describes the 3D perception apparatus used in imaging a robotic scene. In Section 3, the GFS model is detailed, followed by the segmentation approach given in Section 4. In the end, before conclusions and outlook, experimental results are presented in Section 5.

2. 3D Scene Perception Apparatus

The main motivation for fully segmenting the 3D volumetric shapes of objects is to obtain an exhaustive 3D representation which can be reliable used for any further mobile manipulation task. The block diagram of the proposed scene analysis system is presented in Fig. 2. Starting from the PDM of an object and by the similarity transform applied to a correspondent generic shape, a rough 3D segmentation of the object's structure is obtained.

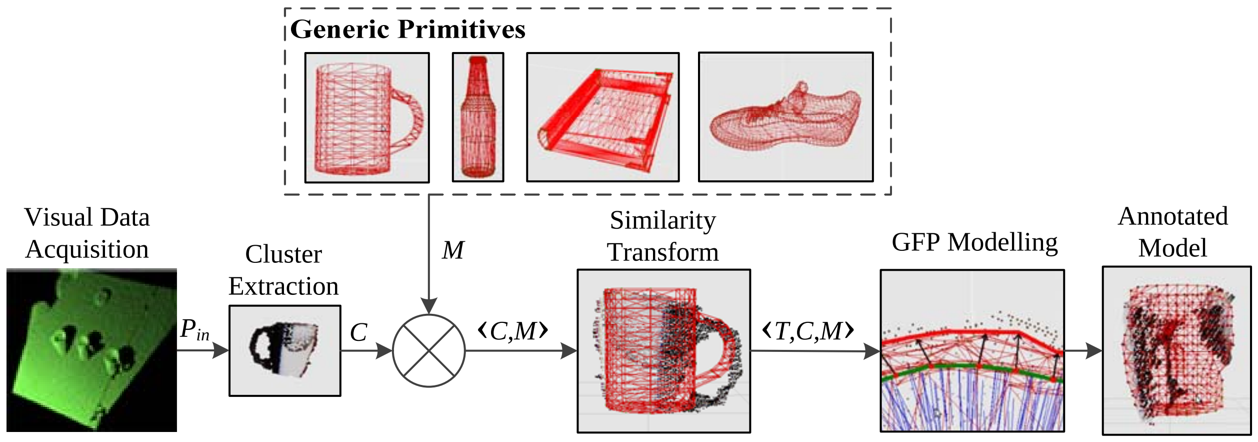


Figure 2: Block diagram of the proposed 3D volumetric estimation system.

The process in Fig. 2 is initialized through tabletop segmentation [21], which calculates the initial reference models, or clusters, $C = [c_0, c_1, \dots, c_k, \dots, c_n]$ on which the GFS will be fitted. For the remainder of the paper, the explanations will be given for a single object cluster c_k. Although, in our work, we have used a standard tabletop segmentation algorithm, the proposed GFS approach can be applied on any given point distribution model.

The tabletop segmentation separates object clusters from a point cloud P_{in} by segmenting out the supporting plane on which the objects reside. Points are grouped together into a cluster if the Euclidean distance between them is smaller than $0.02m$. The output of this operation is represented by the vector of object clusters C. On each of $c_k \in C$ a GFS model will be fitted, as described in Section 4.

As opposed to the approach in [21], which concentrated on fitting fixed CAD models onto objects present on a table, in our work we have tried to extract full shapes of objects present on any given flat surface. As stated before, the cluster extraction method may be replaced with any algorithm which can extract PDMs of objects to be approximated using the GFS system. Also, our main camera used in the experimental sessions, consisted of a MS Kinect® structured light sensor, thus avoiding the calibration between the Time-of-Flight (ToF) sensor and the stereo camera. Finally, the novelty of the approach in [21] lies in the realization of a CAD model matching algorithm between the imaged objects and the CAD models from a database, while the strong advantage of the GFSs is on the precise estimation of the object's shape, given an object cluster from the point cloud data.

Once a cluster has been extracted, the corresponding shape class which will be fitted on it is selected according to a 2D image based classification procedure. The boosting classifier deplyed for this task is based on 6 types of Haar-like features extracted from rectangular patches, resulting in a number of 192 feature values normalized to the interval $[-1, 1]$. As additional feature, the object's color, calculated from the HSV (*Hue*, *Saturation*, *Value*) plane of the image, is added as an extra feature vector.

3. GFS Model Definition

A generic shape is considered to be an a-priori mean shape describing a range of objects, independent of their complexity in terms of 3D structure. The shape modeling challenge is addressed in this paper by deforming a general shape, or GFS, in order to obtain the particularities of a cluster c_k. The GFS is constructed in such a manner that it resembles many similar objects, depicting in this way an universal model for a certain class of objects (e.g. different types of bottles can be roughly approximated by a common model), as stated in the *Generalized Procrustes Analysis* [22]. The core idea of the Procrustes analysis is to estimate a mean shape from a set of samples by aligning them with the purposed of removing the

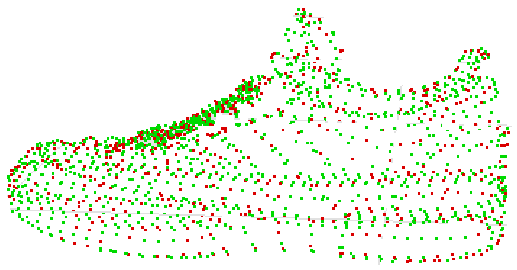


Figure 3: Assignment of points into control (red) and regular (green) points.

translation, rotation and scaling. In our work, we have used the Procrustes analysis for calculating the reference models which will be fitted onto the segmented clusters.

Depending on the actual geometric complexity of an object, the number of 3D points describing the shape model varies, thus having a direct influence on the computation time. In order to exploit the PDM of the shape, we have divided the points of a GFS into two classes:

- *regular points* (p_r), which make up the majority of the points in the GFS model and
- *control points* (p_c), which drive the modeling process according the active contours laws.

The point distribution model of a sample shape is illustrated in Fig. 3. Given an explicit 3D model representation, the classification of the points making up the GFS can be done either manual, or automatic. In the manual procedure, a human manually marks the control points using a special tool. This manual point type definition is considered to be highly accurate, although the procedure is time consuming.

On the other side, the automatic point type assignment is governed by a set of rules, derived from the 2D active contours concept [9], which automatically determines if a point in the GFS is a control point or not. A point of a GFS is considered to be a control point if:

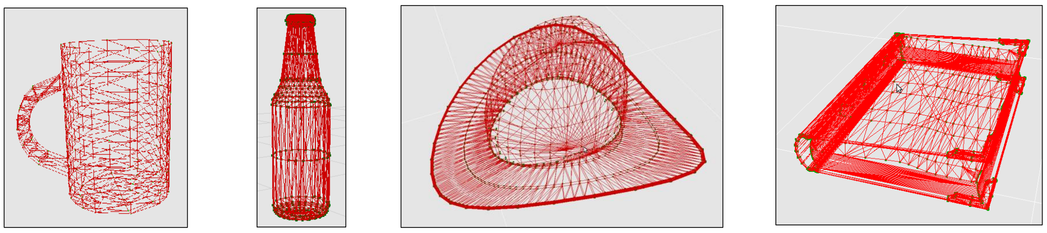
1. it marks the center of a region, or is located on sharp corners of a boundary;
2. is located on curvature extremes;
3. it represents the farthest point in the PDM, calculated by iteratively rotating the model around all its three axes;
4. the point is situated at an equal distance around a boundary between two control points obeying rule one.

The points which do not obey the above rules are implicit considered to be regular points.

The GFS model M of a class of objects is a data structure composed of a vector P for storing the Euclidean 3D coordinates of all the features describing the generic model, a vector Y containing the point types of P, a matrix F for storing the mesh triangulation indexes and, in the end, a structure for additional information regarding a series of global characteristics, namely, length L, height H, width W, scale factor s and the local rotation R and translation t:

$$M(P,Y,F,A) = \begin{cases} P = (p_0, \dots p_n), & \\ Y = (y_0, \dots y_n), & y_i \in \{p_r, p_c\} \\ F = (f_0, \dots f_m), & \text{facet } f = \{p_i, p_j, p_k\};\ \text{where } i,j,k = 0, \dots, n;\ \ i \neq j \neq k \\ A = \{L, H, W, R, t, s\}. & \end{cases} \tag{1}$$

Given a cluster c_k and a shape model M, the fitted, or modeled, GFS S can be defined as:



(a) (b) (c) (d)

Figure 4: Generic shapes. (a) Mug, (b) Bottle, (c) Hat, (d) Book.

$$S : c_k \times M \to P_{in}, \tag{2}$$

where the particular shape S is obtained according to the modeling principles presented in Section 4.

Through the usage of the GFSs, large databases used for object retrieval can be reduced to a small number of shapes. For example, the Princeton shape benchmark [23] has been reduced from 1814 particular objects to a number of 142 shapes by applying the generalized Procrustes analysis and the GFS rules. For each object class, a mean shape has been generated [24], resulting in PDMs such as the ones from Fig. 4. As can be seen, the models are stored in a local coordinate system attached to each PDM. In the followings, the alignment of the shapes to the coordinate system of the segmented cluster c_k will be described.

3.1. GFS model alignment

In order to properly transfer the particularities of a sensed object to its GFS shape, both point distribution models, that is, of the GFS and of the segmented object cluster, need to be registered to the same coordinate system. The alignment process starts by calculating a common reference frame, whose origin is considered to be the centroid m_c of c_k. Further, the GFS shape is registered by translating (t), rotating (R) and scaling (s) a model to the location of the object cluster using the *similarity transform*:

$$S \doteq ||c_k - s \cdot R(M - t)||^2. \tag{3}$$

The scale factor s between the two models is determined by approximating each object with a circumscribed sphere, as depicted in Fig. 5. Firstly, the centroid $m_c(x,y,z)$ of each PDM is computed, followed by the definition of a small sphere centered on each m_c. By constantly increasing the radius of the initial sphere, the smallest sphere which includes inside all the PDM's points is determined. This procedure is applied for the GFS, as well as for the segmented object. The ratio between the radius of the two optimal cropping spheres (r_c and r_{GFS}) is referred to as the scale factor s which adjust the size of the GFS to the size of cluster c_k.

The translation $t = (t_x, t_y, t_z)$ between the GFS and c_k is determined with respect to the two centroids:

$$t = m_c(c_k) - m_c(M), \tag{4}$$

where $m_c(c_k)$ and $m_c(M)$ are the centroids of the segmented and shape's object clusters, respectivelly.

Given the scaled and translated model S, the next challenge is to determine its rotation R relative to the object cluster c_k. This is achieved by rotating the GFS around all its three axes and finding the smallest Euclidean distance d between its points an the closest c_k points:

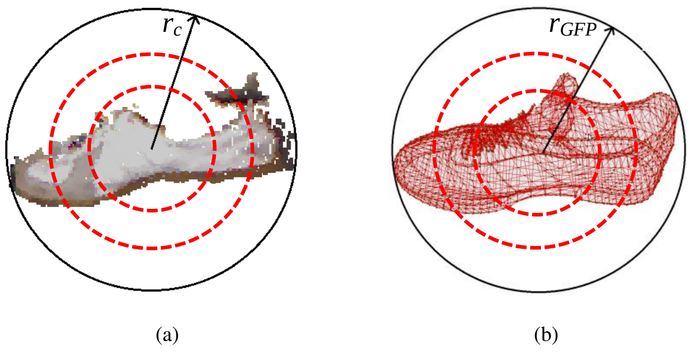


Figure 5: The optimal sphere surrounding an object cluster (a) and the GFS shape model (b).

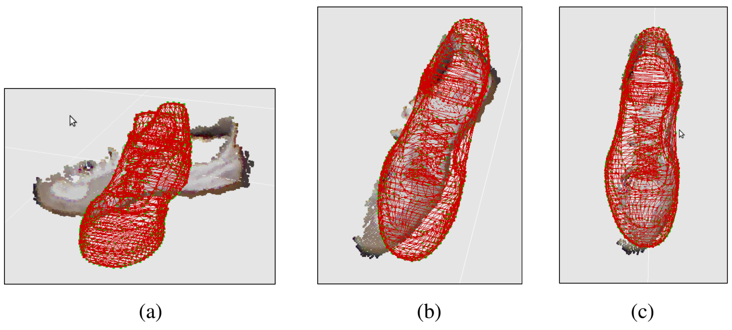


Figure 6: Registration of the GFS onto the object cluster c_k. (a) Translated and scaled GFS model. (b) Euclidean distance based rough orientation estimation. (c) Fine alignment using ICP.

$$d = \underset{d}{\operatorname{argmax}} \sum_{i=0}^{n} \sqrt{(x_i - x_s)^2 + (y_i - y_s)^2 + (z_i - z_s)^2}, \tag{5}$$

where, n is the number of points describing the segmented cluster and x_s, y_s and z_s are the nearest coordinates of the shapes points with respect to the i-th cluster point. For computation efficiency, S is rotated with a $10°$ increment. The rotation around the x, y and z axes which has the best overlapping score, is considered to be the relative rotation of the shape to the scene. This procedure will determine only a coarse orientation R_{coarse} of S relative to c_k.

The final alignment is performed using an *Iterative Closest Point* (ICP) [25] algorithm, which calculates the more accurate orientation R_{fine}. Using only ICP for determining the rotation is not enough since the method converges only if the two input shapes are pre-arranged, that is, the two PDMs are closely positioned in space, given the R_{coarse} transform. The ICP convergence time is dependent on the similarity between the two shapes, as well as on the number of points describing the two PDMs. In our experiments, since the similarity between the GFS and the object cluster is high, the computation time for the ICP varied from 0.17 to $2.79sec$. In the end, the full rotation of S is obtained as a combination of the two coarse and fine orientation matrices:

$$R = R_{coarse} \cdot R_{fine}. \tag{6}$$

A complete similarity transform example is illustrated in Fig. 6. As it will be further explained, the transformed model is used as the input shape for the fitting process described in the next section.

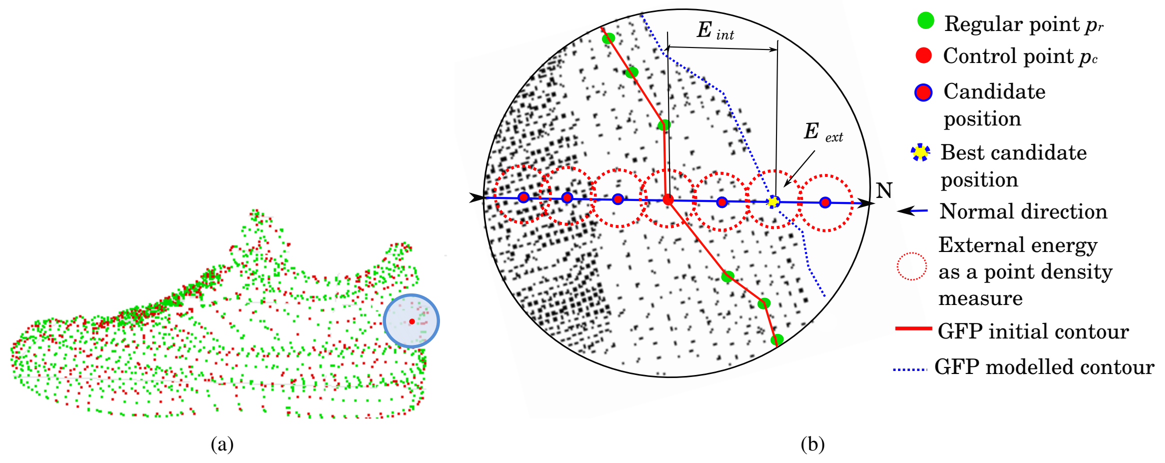


Figure 7: Deformation of a GFS. (a) PDM model of the GFS shape. (b) Locating the best candidate position for a control point along the normal direction.

4. Volumetric Segmentation and Reconstruction via GFS

The purpose of the volumetric segmentation process is to refine the GFS model in order to capture the local geometry of each segmented cluster. Since usually robotic systems image a scene from only one perspective, an important objective of the GFS is to fill in the missing 3D information. As an example, a zoomed neighborhood region of an object is presented in Fig. 7. The modeling procedure will be applied only for the control points p_c. The regular points p_r will be relocated relative to newly determined position of the control points and with respect to the Euclidean distance $d = p_r - p_c$.

Moving a control point p_c towards the real border of the imaged object is not a trivial task. A common technique for automatically determining the position of a contour point, usually applied to the 2D image domain, is the active contours principle, better known as *Snakes* [26]. In the initial formulation, a snake is a 2D curve (e.g. circle) which moves trough the image domain driven by a set of energies attracted by a particular feature in the image, such as intensity transitions.

If snakes are well established algorithms with many applications in 2D computer vision, their usage for estimating object shapes directly in the 3D space has been scarcely investigated. In our GFS framework, the active contours principle is applied for the minimization of a functional of energy $\varepsilon(c_k, S)$, between the cluster c_k and the model shape S positioned using the similarity transform from Eq. 3:

$$\varepsilon(c_k, S) = \underset{\varepsilon}{\operatorname{argmin}} \sum_{1}^{N} (E_{int} - E_{ext}), \tag{7}$$

where $E_{int} \in [0, 1]$ is a so-called internal energy used for constraining the deformation of S such that the integrity of the shape is kept and $E_{ext} \in [0, 1]$ is the external energy which drives the control points to the best candidate position according to the scene point density information. N represents the number of points in S. The objective of the minimization in Eq. 7 is to incrementally sculpt the initial contour, given by the shape model M and the similarity transform into the final fitted shape S which best describes the 3D structural properties of the cluster c_k.

4.1. E_{ext} computation

The goal of the E_{ext} energy is to determine the best position for the control points p_c, given the imaged cluster c_k. In the 2D formulation, each point from the active contour is free to move towards a particular location using a grid schema given by the 2D image domain. The GFS, due to its 3D shape, is more difficult to control because of the extra degree of

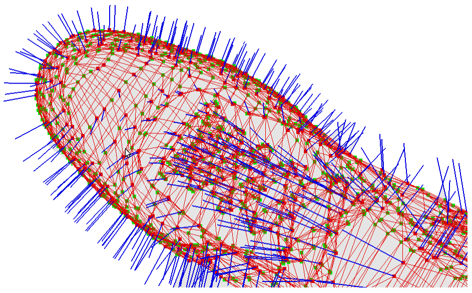


Figure 8: GFS model (red) together with its normal (blue lines) distributions.

freedom introduced by the third dimension. While for a 2D image there are 8 possible moving directions for a contour point, for 3D the number of candidate directions reaches the value of 26.

Also, if in 2D the neighboring relationships between the points are simpler, that is, a neighboring point is simply the next image pixel, in the 3D Cartesian space the neighbors have to be searched. As energy features in 3D, equivalent to the intensity change, we have considered the point cloud's density within a given sphere q, as illustrated in Fig. 7(b). q is relocated in an iteratively manner for the purpose of finding candidate positions for the control points. Hence, if the vicinity of a moving control point encountered a high density of points, then a probable object surface has been found. The amount of neighbors laying in q is determined using the *kdtree* algorithm.

To avoid searching along all 26 possible directions of a control point, the usage of the GFS's control points *normals* $N(p_c)$ is proposed, as illustrated in Fig. 7(b). A set of GFS point normals are presented in Fig. 8. From a total of 26 candidate direction, the point cloud's density search problem has been reduced to only two direction, along the control point's normal. The sphere q is thus iteratively translated along the normal direction until a point density has been encountered. The intersection between this density and the sphere is considered to be the candidate position of the control point. During each iteration, the internal energies ensuring continuity and smoothness are also computed in order to determine if new point location affects the global structure of the GFS. Along the normal, the best candidate is determined using the following set of rules:

- if the control candidate point p_c is already positioned on a high density region, move p_c along the normal and find the first closest point with the number of nearest neighbors nn closest to 0, $nn \rightarrow 0$,
- if p_c has $nn = 0$, search along the normal direction for a density zone $nn > 0$. If no density is found, then the control point is consider to be in an occluded part of the object.

Fig. 7(b) describes the modeling of a control points obeying rule number 2. Thus, the best candidate position is obtained when the searching algorithm finds the edge of the object.

In order to fit the functional of energy formulation in Eq. 7, E_{ext} is obtained as the normalized values of the number of density points in q along the normal direction $N(p_c)$:

$$E_{ext} : q \times N(p_c) \rightarrow c_k. \tag{8}$$

4.2. Determining E_{int}

During each modeling iteration, after the position of the control points has been determined, the regular points p_r are moved according to the positions of p_c. For simplicity, a linear Euclidean distance based deformation law has been proposed. Depending on the sensed and classified object, more complex deformation laws can be developed. After the

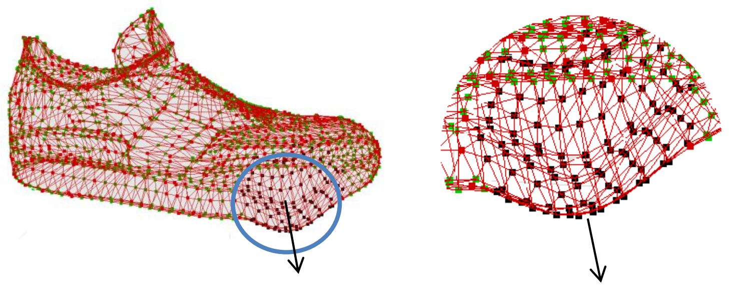

(a) (b)

Figure 9: Linear deformation of points belonging to the GFS (a) arround a local neighborhood (b).

new position of a control point has been determined, all its neighbors lying in a spherical vicinity, with the radius equal to the Euclidean distance between the initial and the candidate control point position, will be adjusted as:

$$p[i+1] = p[i] \cdot \left(1 + \frac{d_c}{d_{max}}\right), \quad p \in \{p_c, p_r\}, \tag{9}$$

where $p[i+1]$ and $p[i]$ are the new $[i+1]$ and the old $[i]$ coordinates of the point lying inside the affected area of radius d_{max}, respectively. d_c is the Euclidean distance between p and the respective control point. d_{max} represents the Euclidean distance between the new and the old position of the control point itself. Fig. 9 shows the behavior of the linear modeling principle. To model the points at a certain vicinity, only the initial position of the shape control point and its best candidate position is required. The Euclidean distance between these points represent the maximum affected area, described as a sphere. All the neighbors outside the sphere will remain unchanged. Inside the sphere, the points are modified according to Eq. 9. The behavior of the contour is similar to the pulling of a piece of cloth in a certain direction.

Following the linear deformation law, the regular points situated closer to the affected control point will suffer a larger translation, whereas for the points laying at the border of the affected area, the deformation will be lower, as illustrated in Fig. 9(b).

The internal energy E_{int} is given by the following linear combination:

$$E_{int} = \alpha \cdot E_{cont} + \beta \cdot E_{curv}, \tag{10}$$

where $\alpha \in [0,1]$ and $\beta \in [0,1]$ are internal, heuristically determined, weight factors, E_{cont} represents the energy which ensures that the GFS surface is continuous, such that the newly rearranged points will not produce large gaps, and E_{curv} is the energy responsible for the surface's smoothness.

E_{cont} and E_{curv} are used exclusively to constrain the movement of the points and, in the same time, to keep the model as compact and intuitive as possible. E_{cont} is determined as the first derivative of a given point p_i inside the contour, represented in Fig. 7 by the distance $d = p_c[i+1] - p_c[i]$ between the current location i and the candidate position $[i+1]$:

$$E_{cont} = ||p_c[i+1] - p_c[i]||^2. \tag{11}$$

The second energy, E_{curv} is computed based on the second derivative of the same modeled point. The derivative computation involves knowledge regarding the neighboring contour points, stored in the M model.

$$E_{curv} = ||p_c[i-1] - 2 \cdot p_c[i] + p_c[i+1]||^2. \tag{12}$$

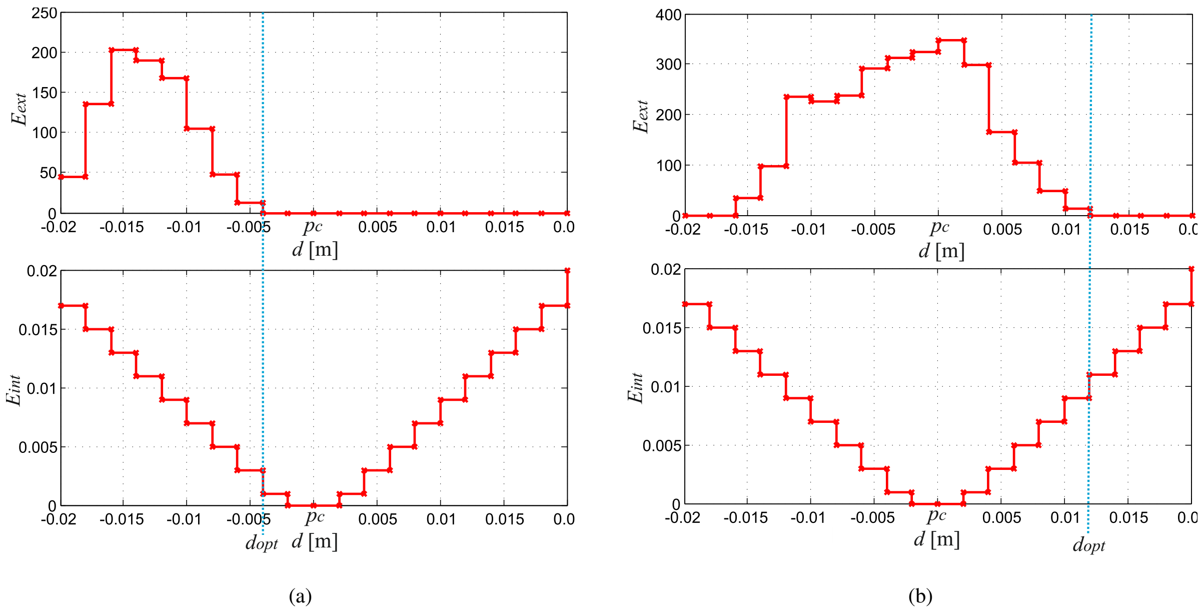


Figure 10: Evolution of the $\varepsilon(c_k, S)$ energy while searching for the candidate position of a control point. Rule one (b) and two (a) from Section 4.1.

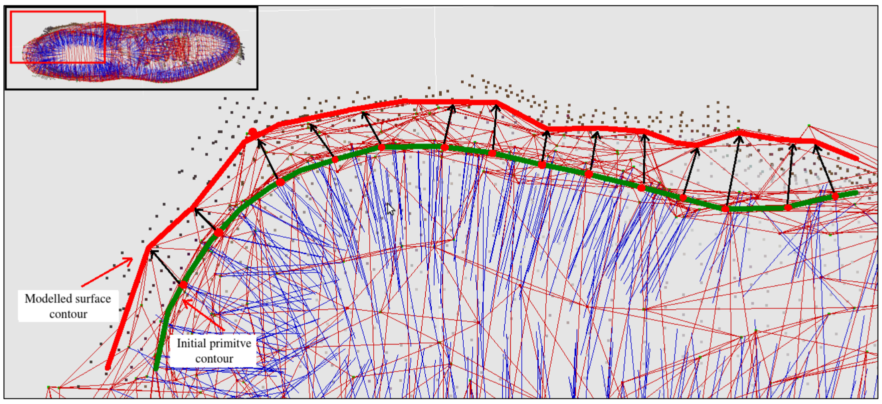


Figure 11: Modeled GFS of a shoe. The difference between the initial (green) and final (red) GFS contour is illustrated with respect to the position of the input point cloud.

Given the above formula, the α factor is a contour elasticity measure. For larger values of α, the E_{int} will grow larger as the contour tends to reach its final form. In the same way, β is responsible for the smoothness of the contour. Again, E_{int} will grow larger if the contour is curved. The evolution of the functional energy $\varepsilon(c_k, S)$ is illustrated in Fig. 10. The modeling of a certain GFS area can be seen in Fig. 11, while the pseudo-code of the proposed method is described in algorithm 1:

By using the proposed GFS system, a substantial computational enhancement can be achieved, along with an optimal 3D model of the imaged object, as put forward in the following experimental results section.

5. Performance Evaluation

5.1. GFSs in Service Robotics

Considering object grasping as one of the main task in service robotics scenarios, the challenge is to determine the optimal grasp configuration for a given 3D segmented object, where a minimal safe grasp is considered to be one in which the object is grasped using at least three pressure points [8, 27]. In our experiments, we have calculated possible grasp configurations using *GraspIt!* [7], a tool which provides an interactive environment where grasping points for any given

```
Data: Object cluster $c_k$;
      Shape model $M$.
Result: modeled GFS volume $S$.
Calculate the initial pose of $S$ using the similarity transform: $S \doteq ||c_k - s \cdot R(M - t)||^2$;
Compute the initial GFS energy $\varepsilon(c_k, S)$;
Set weight factors $\alpha$ and $\beta$;
Initialize the iterations counter $i = 0$.
while $\varepsilon \neq min(\varepsilon)$ do
    Move control points along their normal direction: $p_c[i+1] : p_c[i] \times N(p_c) \rightarrow \Re^3$;
    Relocate all points $\{p_c, p_r\}$ in $S$;
    $\varepsilon[i] = 0$.
    foreach $p \in S$ do
        Measure $E_{ext}$; Compute $E_{curv}$ and $E_{cont}$ for current $p_c$;
        Compute $\varepsilon[i] = \varepsilon[i] + \alpha \cdot E_{cont} + \beta \cdot E_{curv} - E_{ext}$;
    end
    if $\varepsilon[i-1] \leq \varepsilon[i]$ then
        Optimal GFS $S$ obtained;
        Exit.
    end
    $i = i + 1$;
end
```

Algorithm 1: Pseudocode of the GFS modeling approach.

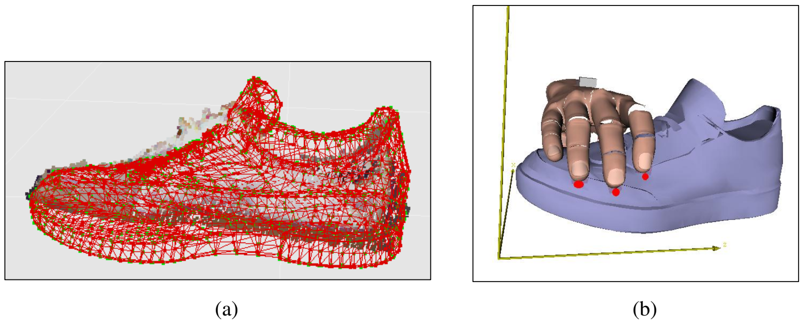


Figure 12: Object grasping simulation. (a) Input GFS shape object. (b) Optimal grasp configuration.

gripper or human hand can be easily determined. Also, instant feedback with respect to the grasp quality can be analyzed for each configuration of the manipulator, along with the projections of the 6-dimensional space of forces and torques that can be applied by the grasp. Hence, the configuration having the grasping points on a modeled surface or near to it gets to be voted as the best grasp configuration, as seen in the example from Fig. 12. Due to the single perspective of the object, at least one of the pressure points lays on the occluded part of the estimated GFS[1].

A couple of grasping configurations extracted from the GraspIt simulator are illustrated in Fig. 13 for the case of a modeled mug and a shoe. The autonomous grasping experiments were performed using a virtual model of a Barreta® gripper. All the configurations have been automatically determined within the simulator, each of them delivering a proper *epsilon* ε grasp quality measure [7], as visible from the values given in Fig. 13. The ε quality refers to the minimum relative magnitude of the outside disturbances that could destroy the grasp. Namely, a grasp would be less stable if it has a smaller epsilon quality. As stated in [28], grasps with epsilon quality $\varepsilon > 0.07$ tend to be more robust in uncertain object perturbations. In our experiments, we modeled the materials of the mug and shoe's GFSs as glass and rubber, respectively. Both of them were considered to have a $0.2Kg$ mass. The friction coefficient between the fingers of the hand

[1]Please see the videos accompanying this paper.

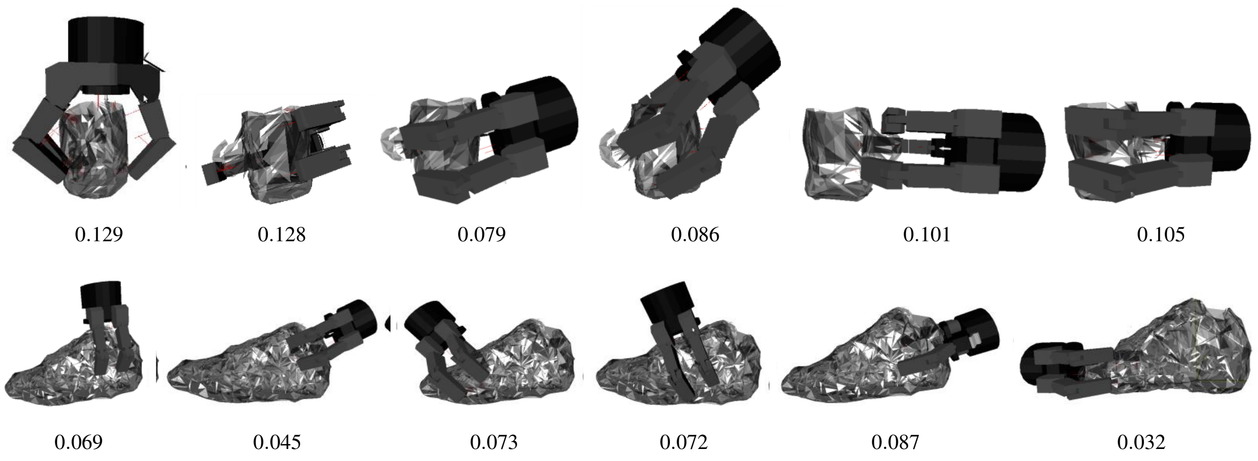


Figure 13: **Different grasp configurations and their ε quality measure for the estimated GFS models of a mug (top row) and shoe (bottom row).**

and the objects, as well as the gravitational constant, were set to 1.0. Although the grasping quality values might be lower than the ones calculated for objects estimated using primitive shapes such as cylinders or cuboids, it should be noticed that the GFS technique delivers 3D shapes very close to the real structures of the imaged objects. It can be thus stated that the value of the computed grasp quality is directly related to the real shape of the objects present in the scene.

5.2. Experimental Setup

For evaluation purposes, a MS Kinect® structured-light device was used to acquire visual data in indoor environments, together with a benchmark database [23] for generating the GFS models. During the tests, all objects were placed on flat surfaces for detection and segmentation[2].

In order to validate the proposed approach, a total of 12 tests were performed on different types of household objects. With the help of the generalized procrustes analysis [22], the size of the models database has been reduced from an initial number of 1814 objects to 142 GFSs.

The point types of the models, that is regular and control points, have been determined using the automated process described in Section 3. The average computation time was approx. $8min$ for each model, varying based on the complexity of the shape and the number of 3D points used to represent it. The $8min$ computation time represents the time required to calculate the regular and control points of the GFS before storing it into the GFSs models database. This is an off-line process, presented in Section 3, and should not be confused with the actual on-line volumetric segmentation and reconstruction time described next.

5.3. Performance Metrics

The evaluation of the GFS modeling procedure, as well as of the methods against which it has been compared, has been made with respect to the Euclidean distance between the modeled shape and the input point cloud. Since the main goal of the proposed approach is to create a particular representation of a GFS best resembling the imaged object, the distance between these two shapes can be considered to be a similarity measure. Thus, by summing the distances between each point from the cluster object and the nearest GFS point, the following fitting likelihood metric is obtained:

[2]The source code of the GFS approach is released under an open-source license at `http://rovis.unitbv.ro/rovis-machine-vision-system/`. Please ask the authors permission for downloading.

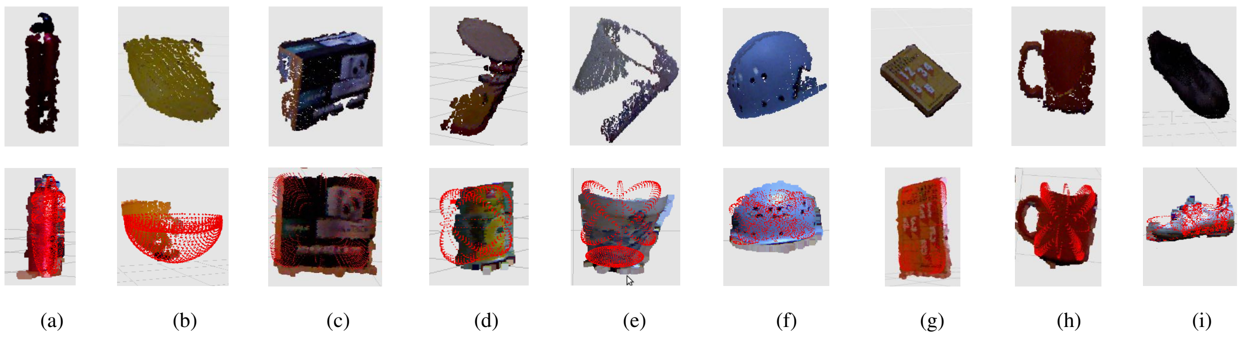


Figure 14: Objects clusters and obtained superquadric models. (a) Bottle. (b) Bowl. (c) Box. (d) Can. (e) Flower pot. (f) Helmet. (g) Book. (h) Mug. (i) Shoe.

$$f_d(c_k, S) = \frac{1}{N} \sum_{i=0}^{n} \frac{1}{1 + \underset{f_d}{\operatorname{argmin}} ||c_{k_i} - nn_i(S)||^2 \cdot \gamma}, \tag{13}$$

where $f_d \in [0,1]$ is the fitting error and N the number of points in the GFS. c_k and S are the PDMs of the clustered object and of the modeled GFS, respectively. c_{k_i} is the closest scene point to a GFS point $nn_i(S)$, while γ represents a scale factor. The better the GFS modeling is, the lower the value of f_d is. A perfect fit corresponds to a modeled shape that perfectly fits the imaged scene. The f_d, determined using a kdtree structure, is calculated for distances smaller that $3mm$.

Along with the f_d measure, the computation time has also been taken into consideration. The proposed algorithm has been tested on a laptop computer, equipped with a 2.6GHz dual-core CPU and 2GB of RAM memory.

To demonstrate the efficiency of the presented method, a comparative analysis has been performed against the *superquadric estimation* method and the *3D object retrieval* technique. Both methods, along with their advantages and drawbacks, are highlighted in the next subsections.

5.3.1. Superquadric estimation vs. GFS

An alternative approach to the 3D segmentation objective is to approximate the rigid body's shape with an implicit geometric model, entitled *superquadric*. A superquadric $S(b)$ is a parametrized geometric shape obtained by the spherical product between two curves modeled by a series of parameters b [12]. It approximates geometrical models starting from simple cubes or cylinders and ending with complex shapes such as toroids or hyperboloids. Because of the roughness of the obtained volume, it is not desirable to use only one superquadric for segmenting an object. Multiple fused superquadics can produces a more precise and finer shape [14].

By constantly changing the 11 superquadric parameters (3 for rotation, 3 for translation and 5 for modeling) which define the superquadric shape and by evaluating the newly obtained form, an optimal best fit model can be obtained. The approach is considered fast since only few parameters are actually controlling the superquadric shape. Nevertheless, complex objects tend to increase the overall segmentation time since a higher number of superquadric models need to be estimated. The accuracy of the segmented object is dependent on the number of superquadric shapes used. Thus, a compromise between computation time and accuracy must exist. In other words, for simple objects (e.g. cuboid like objects), the superquadric approach is fast and reliable. One advantage of the superquadric 3D segmentation method is that it does not need pose normalization or any a-priori knowledge about the segmented object. Shapes estimated using both methods are illustrated in Fig. 14 and 15, together with their quantitative results given in Table 1.

One advantage of the superquadric approach is that it uses a greater number of points to represent the final volume. As opposite to this property, the GFS segmentation technique seeks to model shapes represented by a reduced number of

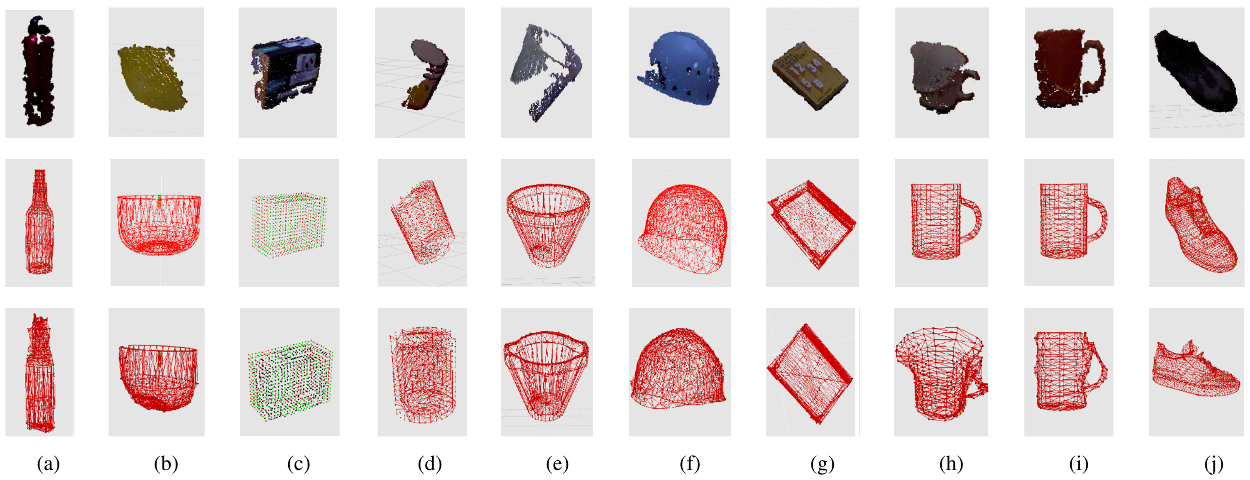
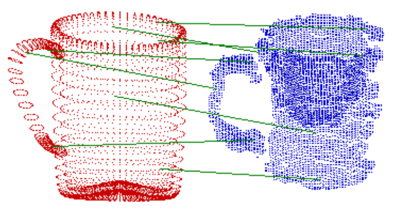


Figure 15: Object clusters (top row), initial shape models (middle row) and modeled GFSs (lower row).

Table 1: Comparative results between Superquadric based volume estimation and the GFS technique.

Object	Nr. of superquadrics used to represent the volume	Nr. of points describing the volume		f_d		Computation time [sec]	
		Superquadric	GFS	Superquadric	GFS	Superquadric	GFS
Bottle	3	10944	558	0.073371	0.037132	5.8	3.02
Bowl	2	8208	799	0.094871	0.011962	5.46	3.41
Box	1	2736	1232	0.064358	0.044763	3.9	4.32
Can	1	3754	546	0.099987	0.033166	3.8	2.97
Flower pot	2	8917	529	0.073548	0.017785	7.6	3.37
Helmet	3	11324	1002	0.063191	0.011354	5	5.2
Book	1	2895	1284	0.039124	0.030215	4.7	4.68
Mug	1	3102	410	0.058250	0.029193	3.5	2.56
Shoe	4	16416	1842	0.060907	0.030155	10.7	4.96


(a)

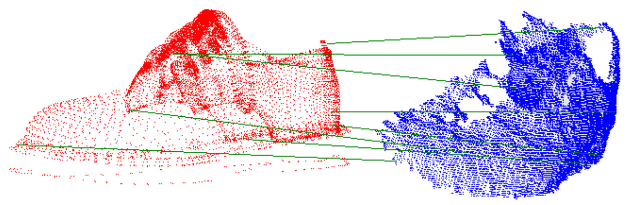

(b)

Figure 16: 3DOR based point correspondences (green lines) between the scene clusters c_k and the most similar models from the database.

Table 2: Comparative results between the 3DOR engine and the GFS technique.

Object	Nr. of 3DOR correspondences	f_d		Computation time [sec]	
		3DOR	GFS	3DOR	GFS
Bottle	23	0.062543	0.037132	124	7.23
Bowl	48	0.051513	0.011962	97.2	6.74
Box	19	0.056728	0.044763	175.4	8.21
Can	33	0.077719	0.033166	131	5.24
Flower pot	28	0.052494	0.017785	89	8.69
Helmet	14	0.055895	0.011354	101	10.47
Book	19	0.042879	0.030215	69	6.5
Mug	103	0.043365	0.029193	136	4.18
Shoe	365	0.060907	0.030155	201	6.56

control points in order to keep the algorithm's computation time attractive. However, the superquadric fitting likelihood is lower in comparison the GFS technique since the superquadric points cannot approximate complex shapes accurately.

5.3.2. 3DOR vs. GFS

One other approach in determining the 3D shape of an object, without actually modeling it, can be addressed by the usage of a large database (DB) with 3D models. Such a DB should be large enough to encompass the shape variability of objects belonging to the same class. The main challenge here is on finding within the DB the most similar shape describing the object of interest. Such a problem is usually approached using a *3D Object Retrieval* (3DOR) engine. The 3DOR method locally evaluates small regions between the models and the query input cloud, without a need for recognizing the whole object. If the extracted DB model is a perfect replica of the real object, then the fitting accuracy will be close to 1.0.

However, the processing time is one of the major drawbacks of a 3DOR engine, where the highest amount of time is spent on establishing correspondence points between the scene and the objects in the DB. Considering the benchmark DB from [23], a search operation for a given object over the 1814 models lasted 5220 sec (87 min).

The final computation time is also influenced by the type of descriptor used in searching for point correspondences. In our implementation, for representing the surfaces of the models, a SHOT (*Signature of Histograms of OrienTations*) descriptor has been used [29]. Considering the particular case of a shoe, the final computation time was of 24.36 min. The correspondence matching, illustrated in Fig. 16, has been performed with the help of the L2-Norm and correspondence correlation coefficient [30]. As shown in Table 2, the GFS approach is around ten times faster than the 3DOR method. In terms of accuracy, the 3DOR based segmentation approach is superior to the GFS only for cases in which the optimal database model resembles the sensed object very accurately (see Table 2).

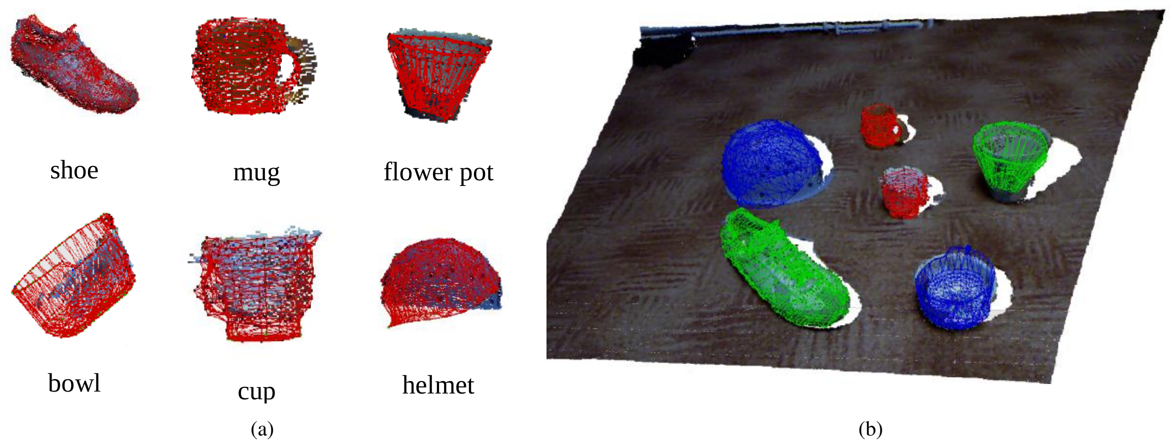


Figure 17: Segmenting multiple objects via the GFS approach. (a) Input object clusters and the modeled GFSs. (b) Whole scene reconstruction.

5.3.3. *Modeling multiple GFSs*

Usually, in real-world applications, the challenge is to detect and segment the whole scene, together with the objects involved in the mobile manipulation scenarios, as shown in Fig.17. In our implementation, we have sequentially reconstructed the 3D shapes of objects present in a test scene.

The Euclidean cluster extraction method is performed once, at the beginning of the algorithm. The priority of the objects in the pipeline is given by the relative distance between the objects and the robot. Thus, the closest scene object to the robot will be the first segmented model. Considering the case of multiple identical objects placed in the same scene (e.g. 3 or 4 cups on a table), the first modeled GFS cup can be used as an initial GFS for the second cup, enhancing thus the convergence of the second model. In Fig.17, a complex scenario consisting of multiple segmented objects is presented.

5.4. *Discussion*

Numerical results with respect to the performance of the proposed GFS approach applied on the considered test objects are given in Table 3. The number of modeled points is directly dependent on the input cloud information, that is, they are moved only if valid 3D data exist along the directions of the GFS's normals. The sparse nature of the input object cluster c_k can be seen in the first row of Fig. 15. The higher the number of points representing c_k, the better the GFS approximation is. In Fig. 18, the likelihood fitting error f_d is illustrated for the three evaluated methods, that is GFS, superquadrics and 3DOR engine. As can be seen, its value is lower for the case of the GFS approach.

As a comparison to the sparse clouds from Fig. 15, the same modeling principle has been applied to dense clusters describing the objects of interest. These point clouds have been calculated from multiple views, acquired through the movement of the sensor around the objects and fusing the segmented objects points using an *Iterative Closest Point* (ICP) algorithm, similar to the one proposed in [31]. In such an approach, the only part unregistered part of the object is its bottom, as seen in Fig. 19. Although such a dense object point cloud is more appropriate for the GFS modeling technique, its acquisition and ICP computation time makes it difficult for practical usage. Usually, in mobile manipulation systems, the scene is image from only one perspective, thus obtaining only around 40% of the points making up the object of interest, as seen in Fig. 15. The fitting errors for both single view and ICP based multiple views reconstructed objects are given in Table 4.

Table 3: Experimental results for the GFS modeling algorithm applied on different objects.

Object	Total nr. of GFS points	Object size (r_c) [mm]	Points modeled	f_d	Modeling time [sec]	Total processing time [sec]
Bottle	558	160	165	0.037132	3.02	7.23
Bowl	799	82	572	0.011962	3.41	6.74
Box	1232	134	483	0.044763	4.32	8.21
Can	546	90	176	0.033166	2.97	5.24
Flower pot	529	104	336	0.017785	3.37	8.69
Helmet	1002	140	515	0.011354	5.2	10.47
Book	1284	138	466	0.030215	4.68	6.5
Mug	410	72	185	0.029193	2.56	4.18
Cup	410	62	198	0.013508	2.19	3.93
Shoe	1824	158	993	0.030155	4.96	6.56

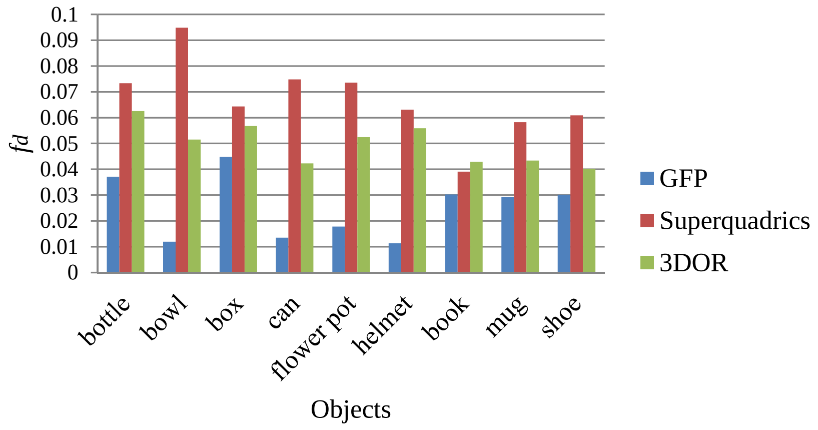


Figure 18: Fitting error f_d for GFSs, superquadrics and the 3DOR engine.

Also, there is a direct connection between the form of the initial, un-modeled shape, and the actual structure of the c_k cluster. If an un-modeled GFS resembles very close the perceived surface, then, the majority of its points lay from start almost near to the positions of the points in c_k. In consequence, fewer iterations are needed to drive the structure of the GFS. Fewer iterations imply rapid convergence and thus a lower modeling time. Considering again the case of the bowl, the final model has converged in 22 iterations. In the original formulation of the active contours principle, the initial shape was considered to be a sphere. In the sphere case, the final convergence is obtained after 37 iterations instead of 22. The smaller amount of iterations needed to converge to the final model are the direct consequence of using a GFS model instead of using a sphere as an initial contour of the segmented object. The iteration number is also influenced by the initial GFS alignment. A rough alignment forces the control points to search for their final positions with respect to the farthest allowable distance, thus increasing in the number of iterations needed to converge, and implicit increasing the computation time. Fig. 20 shows the evolution of the energy function 7 with respect to the number of iterations needed

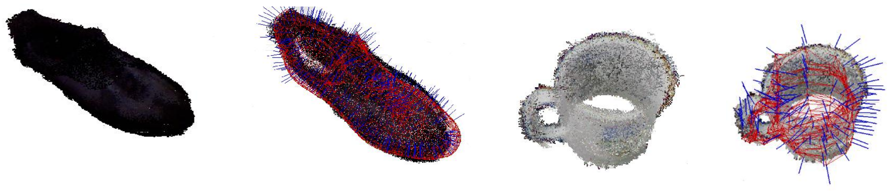


Figure 19: GFSs on dense object clusters. (a, c) Dense PDMs. (b, d) Modeled shapes.

Table 4: GFS modeling results for objects described by dense point clouds.

Object	f_d (single view)	f_d (multiple views)	Modeling time [sec]
Bottle	0.037132	0.025689	4.98
Bowl	0.011962	0.010289	5.04
Box	0.044763	0.035897	7.24
Can	0.033166	0.030289	4.92
Flower pot	0.017785	0.009851	7.68
Helmet	0.011354	0.008799	8.56
Book	0.030215	0.028932	6.36
Mug	0.029193	0.015792	4.20
Cup	0.013508	0.010325	4.13
Shoe	0.030155	0.028934	6.78

to reach the final contour of the object using different shapes (GFS and sphere) as the initial contour. The number of iterations for the objects used in our experiments can be seen in Fig. 21.

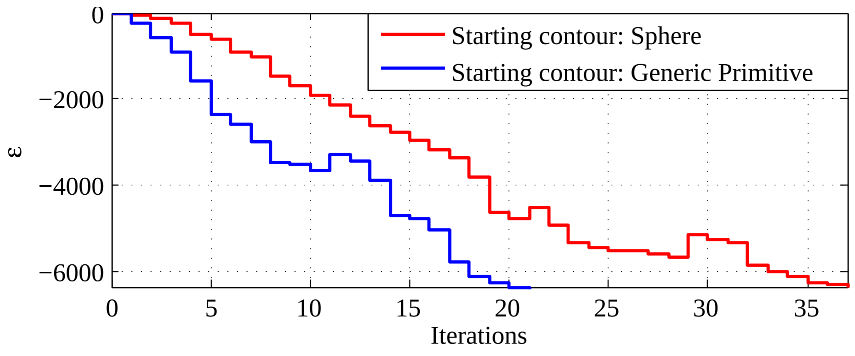


Figure 20: Number of iterations required for modeling an object using the GFS approach and two different initial shape contours.

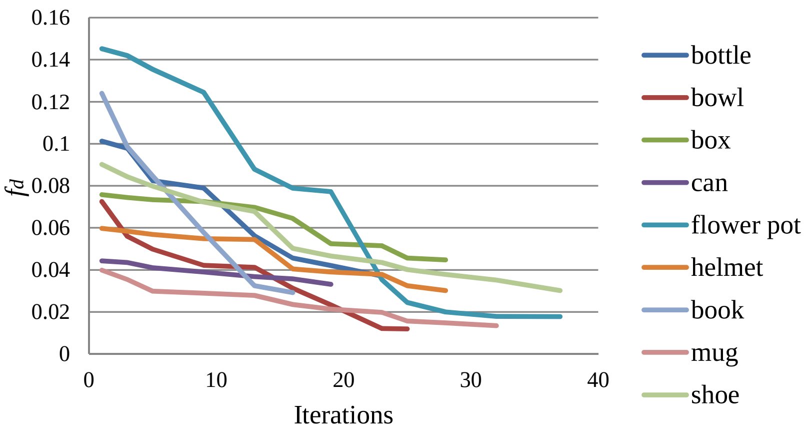


Figure 21: Number of iterations required for modeling different objects.

From the processing time point of view, the GFS method overcomes closely the superqadric approach and from far the 3DOR technique. Indeed, for simple and regular shapes, the superquadrics are faster but not as accurate as the GFSs. One of the most computational expensive operations in our current version of the algorithm is the estimation of the shape's rotation, prior to the actual modeling of the GFS. However, the high computation time is due to the sequential implementation of the method. As further work, the parallelization of the procedure through *Graphical Processing Units* (GPUs), or *Field Programmable Gate Arrays* (FPGAs) has been taken into consideration. As a comparison between the two parallel computation methods, the GPU one seems more attractive from the implementation point of view, since standard computers and smartphones nowadays are equipped with such devices. Also, only the heavy computational functions can be moved to the parallel device, without interfering with the pure sequential parts of the algorithm.

6. Conclusion and future work

In this paper, a 3D object segmentation method based on *Generic Fitted Shapes* (GFS) has been proposed in the context of robotic mobile manipulation. The main objective of the algorithm is to segment the full 3D volume of objects of interest encountered in service robotics scenarios. Because of the comprehensive formulation, the GFS model can be used for grasping, object tracking and/or obstacles avoidance. As future work, the authors plan to enhance the approach with the purpose of estimating the volumetric shapes of deformable objects and to improve the computation time through parallel computational devices such as GPUs, or FPGAs.

Acknowledgement

This paper is supported by the Sectoral Operational Program Human Resources Development (SOP HRD), financed from the European Social Fund and by the Romanian Government under the projects POSDRU/107/1.5/S/76945 and POSDRU/89/1.5/S/59323.

References

[1] O. M. Mozos, Z. C. Marton, and M. Beetz, “Furniture Models Learned from the WWW – Using Web Catalogs to Locate and Categorize Unknown Furniture Pieces in 3D Laser Scans,” *Robotics & Automation Magazine*, vol. 18, no. 2, pp. 22–32, June 2011.

[2] J. Bohren, R. B. Rusu, G. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Moesenlechner, W. Meeussen, and S. Holzer, “Towards Autonomous Robotic Butlers: Lessons Learned with the PR2,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Shanghai, China, 2011, pp. 5568–5575.

[3] S. M. Grigorescu, T. Lüth, C. Fragkopoulos, M. Cyriacks, and A. Gräser, “A BCI-Controlled Robotic Assistant for Quadriplegic People in Domestic and Professional Life,” *Robotica*, vol. 30, no. 3, pp. 419–431, May 2012.

[4] Z. C. Marton, D. Pangercic, N. Blodow, and M. Beetz, “Combined 2D-3D Categorization and Classification for Multimodal Perception Systems,” *Int. Journal of Robotics Research*, vol. 30, no. 11, pp. 1378–1402, Sept. 2011.

[5] J. Dias, A. Sahbani, L. Seneviratne, S. Zeghloul, and J. Zhang, “Special issue on Autonomous Grasping,” *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 323–325, 2012.

[6] D. R. Faria, R. Martins, J. Lobo, and J. Dias, “Extracting Data from Human Manipulation of Objects Towards Improving Autonomous Robotic Grasping,” *Robotics and Autonomous Systems, Special issue on Autonomous Grasping*, vol. 60, no. 3, pp. 396–410, 2012.

[7] A. Miller and P. K. Allen, “Graspit!: A Versatile Simulator for Robotic Grasping,” *IEEE Robotics and Automation Magazine*, vol. 11, no. 4, pp. 110–122, Dec. 2004.

[8] A. Morales, P. J. Sanz, A. P. D. Pobil, and A. H. Fagg, “Vision-based Three-finger Grasp Synthesis Constrained by Hand Geometry.” *Robotics and Autonomous Systems*, vol. 54, no. 6, pp. 496–512, June 2006.

[9] T. F. Cootes, C. J. Taylor, D. Cooper, and J.Graham, "Active Shape Models: Their Training and Application," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38–59, 1995.

[10] G. Wang, Z. Houkes, G. Ji, B. Zheng, and X. Li, "An Estimation-Based Approach for Range Image Segmentation: on the Reliability of Primitive Extraction," *Pattern Recognition*, vol. 36, no. 1, pp. 157–169, 2003.

[11] A. Adán, S. Salamanca, and P. Merchán, "Cultural Heritage: A Hybrid Human-Computer Approach for Recovering Incomplete Cultural Heritage Pieces," *Computer and Graphics*, vol. 36, no. 1, pp. 1–15, Feb. 2012.

[12] A. Barr, "Superquadrics and Angle-Preserving Transformations," *IEEE Computer Graphics and Applications*, vol. 1, pp. 11–23, 1981.

[13] G. Biegelbauer and M. Vincze, "Efficient 3D Object Detection by Fitting Superquadrics to Range Image Data for Robot's Object Manipulation." in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Rome, Italy, 2007, pp. 1086–1091.

[14] T. T. Cocias, S. M. Grigorescu, and F. Moldoveanu, "Multiple-Superquadrics Based Object Surface Estimation for Grasping in Service Robotics," in *Proc. of the 13th Int. Conf. on Optimization of Electrical and Electronic Equipment*, Brasov, Romania, 2012, pp. 1471–1477.

[15] Z.-C. Marton, D. Pangercic, N. Blodow, J. Kleinehellefort, and M. Beetz, "General 3D Modelling of Novel Objects from a Single View," in *Proc. of the 2010 IEEE/RSJ int. Conf. on Intelligent Robots and Systems*, Taipei, Taiwan, Oct. 2010, pp. 3700–3705.

[16] T. McInerney and D. Terzopoulos, "A Finite Element Model for 3D Shape Reconstruction and Nonrigid Motion Tracking," in *Proc. of the 4th Int. Conf on Computer Vision*, Berlin, Germany, May 1993, pp. 518–523.

[17] R. Durikovic, K. Kaneda, and H. Yamashita, "Reconstructing a 3-D Structure with Multiple Deformable Solid Primitives," *Computer and Graphics*, vol. 21, no. 5, pp. 611–624, Sep. 1997.

[18] J. W. Tangelder and R. C. Veltkamp, "A Survey of Content Based 3D Shape Retrieval Methods," *Multimedia Tools and Applications*, vol. 39, no. 3, pp. 441–471, Sep. 2008.

[19] E. Steinbach, B. Girod, P. Eisert, and A. Betz, "3-D Object Reconstruction Using Spatially Extended Voxels and Multi-Hypothesis Voxel Coloring," in *Proc. of the Int. Conf on Pattern Recognition*, ser. ICPR '00. Washington, DC, USA: IEEE Computer Society, 2000, pp. 774–777.

[20] G. Slabaugh, T. Malzbender, B. Culbertson, and R. Schafer, "A Survey of Methods for Volumetric Scene Reconstruction from Photographs," in *Proc. of the 2001 Eurographics Conf. on Volume Graphics*, New York, USA, 2001, pp. 81–101.

[21] U. Klank, D. Pangercic, R. B. Rusu, and M. Beetz, "Real-time CAD Model Matching for Mobile Manipulation and Grasping," in *Proc. ot the 9th IEEE-RAS Int. Conf. on Humanoid Robots*, Paris, France, Dec. 2009, pp. 290–296.

[22] I. L. Dryden and K. Mardia, *Statistical shape analysis*, ser. Wiley series in probability and statistics: Probability and statistics. Chichester [u.a.]: J. Wiley, 1998.

[23] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The Princeton Shape Benchmark," in *Proc. of the 2004 Int. Conf. on Shape Modeling*, Washington, DC, USA, June 2004, pp. 167–178.

[24] R. H. Davies, C. J. Twining, and C. J. Taylor, *Statistical Models of Shape - Optimisation and Evaluation*. London, UK: Springer Publishing Company, Incorporated, 2008.

[25] Z. Zhang, "Iterative Point Matching for Registration of Free-form Curves and Surfaces," *Int. Journal on Computer Vision*, vol. 13, no. 2, pp. 119–152, Oct. 1994.

[26] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models," *Int. Journal on Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[27] M. Prats, P. J. Sanz, and A. P. D. Pobil, "Task Planning for Intelligent Robot Manipulation," in *Proc. of the 25th Int. Conf. on Artificial Intelligence and Applications*, Innsbruck, Austria, 2007, pp. 214–219.

[28] H. Dang and P. K. Allen, "Learning grasp stability," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, RiverCentre, Saint Paul, Minnesota, USA, May 14-18 2012, pp. 2392–2397.

[29] F. Tombari, S. Salti, and L. Di Stefano, "Unique Signatures of Histograms for Local Surface Description," in *Proc. of the 11th European Conf. on Computer Vision*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, vol. 6313, pp. 356–369.

[30] J. L. Rodgers and A. W. Nicewander, "Thirteen Ways to Look at the Correlation Coefficient," *The American Statistician*, vol. 42, no. 1, pp. 59–66, 1988.

[31] M. Krainin, P. Henry, X. Ren, and D. Fox, "Manipulator and Object Tracking for In-hand 3D Object Modeling," *Int. Journal of Robotics Research*, vol. 30, no. 11, pp. 1311–1327, Sept. 2011.