

Learning of 2D Grasping Strategies from Box-Based 3D Object Approximations

Sebastian Geidenstam, Kai Huebner, Daniel Banksell and Danica Kragic

Computer Vision & Active Perception Lab

KTH – Royal Institute of Technology, Stockholm, Sweden

Email: {sebbeg,khubner,banksell,danik}@kth.se

Abstract—In this paper, we bridge and extend the approaches of 3D shape approximation and 2D grasping strategies. We begin by applying a shape decomposition to an object, i.e. its extracted 3D point data, using a flexible hierarchy of minimum volume bounding boxes. From this representation, we use the projections of points onto each of the valid faces as a basis for finding planar grasps. These grasp hypotheses are evaluated using a set of 2D and 3D heuristic quality measures. Finally on this set of quality measures, we use a neural network to learn good grasps and the relevance of each quality measure for a good grasp. We test and evaluate the algorithm in the GraspIt! simulator.

I. INTRODUCTION

In the field of intelligent grasping and manipulation, a robot may recognize an object first and then reference an internal object model. For unknown objects, however, it needs to evaluate from data it can collect on the spot. How to grasp a novel object is an ongoing field of research. Difficulties in this area include (i) the high dimensionality of the problem, (ii) incomplete information about the environment and the objects to be grasped, and also (iii) generalizable measures of quality for a planned grasp.

Since contacts and forces of the fingers on an object’s surface make up a grasp, it is very important to have good information both about the hand and the object to be grasped. Both hand and object constraints together with the constraints for the task to be performed need to be considered [1]. Though there is interesting work on producing grasp hypotheses from 2D image features only, e.g. [2, 3], most techniques rely on 3D data. Due to the complexity of the task, much work has been done for simplifications of 3D shape, such as planar [4] or 3D-contour-based [5] representations. Other approaches involve modelling an object perfectly, i.e. known a-priori, or with high-level shape primitives, such as the use of grasp pre-shapes or Eigengrasps [6, 7, 8]. One work that uses high-level shape primitives, and is similar to ours in terms of learning, but by using an SVM approach, is [9]. Another approach to learning from 2D grasp qualities, using neural networks and genetic algorithms, is presented in [10].

This paper builds on the work of Huebner *et al.* [11, 12], which uses a hierarchy of minimum volume bounding boxes to approximate an object from a set of 3D points delivered by an arbitrary 3D sensor, e.g. laser scanners or stereo camera setups. Grasping is then done by approaching each face of a box until contact, backing up, and then grasping the object. What this work lacked however, was a way to explicitly place the fingers

of the hand and to choose the best configuration of the hand. Learning to predict successful grasps was done only with raw data from the projections of points inside a box onto the face to be grasped. Secondly, our work makes use of an algorithm for finding and predicting the success of a grasp, but for planar objects, as proposed by Morales *et al.* [4, 13]. The approach uses 2D image analysis to find contact point configurations that are valid given specific kinematic hand constraints. From the geometrical properties of an object, it then calculates a set of quality measures that can later be used for learning to predict the success of found grasp hypotheses. The limitations of this work lie mainly in the fact that really ‘planar’ objects and representations are discussed in which information about 3D shape is discarded.

In this paper, we bridge and extend these two methods to enable 2D grasping strategies for 3D object representations.

II. 3D BOX APPROXIMATION

We will shortly revisit the pre-computation of approximating a 3D point cloud by a constellation of minimum volume bounding boxes (MVBBs). The fit-and-split approach starts with fitting a root bounding box and estimating a best split by using the 2D projections of the enclosed points to each of the box surfaces. Depending on a volume gain parameter t , two child boxes might be produced and then be tested for splitting. To provide an insight to this algorithm as a base for the experiments in this paper, the two core algorithms have been sketched in Fig. 1. For more details and examples, we refer to Huebner *et al.* [11]. However, it is important to note that in that work (i) 2D projections have been used to estimate a split and (ii) only edge-parallel planar splits have been tested.

From these constraints, three main problems were evident relating to the original split estimation. These problems are outlined as follows.

1) *Splitting of non-convex regions, e.g. u-shapes:* As shown in [11], the presented algorithm will not do any splitting in case of u-shaped 2D projections. This is due to the fact that it uses upper bounds and area minimization, which are constant in such cases. This means that a split does not result in a substantial change in the area of a region. A solution for this problem remains a challenge [11], especially when sparse and noisy data is provided. For 3D data from real vision systems or laser scanners, such distortions are unavoidable, in part because of occlusion or sensor inaccuracies. Thus,

Algorithm II.1: BOXAPPROXIMATE($points^{3D}$)

```

 $box \leftarrow findBoundingBox(points^{3D})$ 
 $faces \leftarrow nonOppositeFaces(box)$ 
 $(p, q) \leftarrow split(FINDBESTSPLIT(faces, points^{3D}))$ 
if ( $percentualVolume(p + q, box) < t$ )
  then  $\begin{cases} BOXAPPROXIMATE(p) \\ BOXAPPROXIMATE(q) \end{cases}$ 
  else return ( $box$ )

```

Algorithm II.2: FINDBESTSPLIT($faces, points^{3D}$)

```

for  $i \leftarrow 1$  to 3
   $p^{2D} \leftarrow project(points^{3D}, faces[i])$ 
  for  $x \leftarrow 1$  to  $width(faces[i])$ 
    do  $\begin{cases} (p1, p2) \leftarrow verticalSplit(p^{2D}, x) \\ a1 \leftarrow boundArea^{2D}(p1) \\ a2 \leftarrow boundArea^{2D}(p2) \\ \text{if } (a1 + a2 < minArea) \\ \text{then } \begin{cases} minArea \leftarrow (a1 + a2) \\ bestSplit \leftarrow (i, x) \end{cases} \end{cases}$ 
  for  $y \leftarrow 1$  to  $height(faces[i])$ 
    do  $\begin{cases} (p1, p2) \leftarrow horizontalSplit(p^{2D}, y) \\ a1 \leftarrow boundArea^{2D}(p1) \\ a2 \leftarrow boundArea^{2D}(p2) \\ \text{if } (a1 + a2 < minArea) \\ \text{then } \begin{cases} minArea \leftarrow (a1 + a2) \\ bestSplit \leftarrow (i, y) \end{cases} \end{cases}$ 
return ( $bestSplit$ )

```

Fig. 1. Pseudocode (original algorithm): a point set and its bounding box, respectively, are recursively split (II.1). A good split was estimated through analysis of 2D splits of the projected points onto each of the box faces (II.2).

how to distinguish between a real non-convex object region and just incompleteness of the data becomes a critical issue. The models used in [11] were ideal models, extracted from simulated 3D mesh data. As it is our aim to evaluate our algorithm also on real sensory data, we can not generally assume such ideal conditions.

2) *Splitting along non-edge-parallel directions:* The minimum volume box fitting approach naturally fits extensions of the shape into corners of a box, as this keeps the box smaller. The handle of a cup, for example, will fit best diagonally into one of the box corners. However, such diagonal structures in particular can rarely be cut parallel to one of the box edges as proposed in the previous algorithm.

3) *Sensitivity to noise:* The box decomposition’s robustness showed the splitting to be very sensitive to noise. This is not a main issue in terms of single box or face grasping in general, since any constellation of boxes will produce grasp hypotheses. However, if one would like to take into account and learn from a whole constellation of boxes, then robustness and repeatability are necessary.

A. Improved Split Algorithm using 2D Convex Hulls

For the experiments presented in this paper, we have therefore implemented a new algorithm based on convex hulls. The new algorithm replaces II.2, solving the above mentioned issues, and in addition producing much more confident splitting

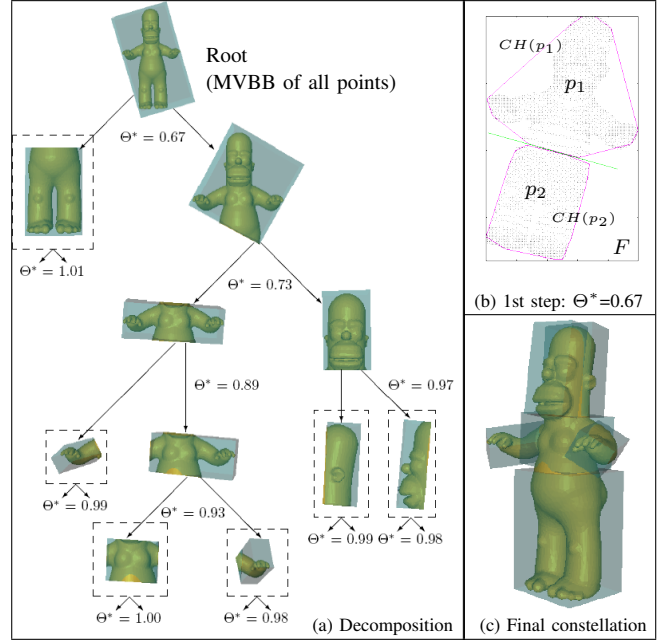


Fig. 2. (a) Example of a decomposition hierarchy, using a gain parameter of $t=0.98$. With $\Theta^* < t$, a valid cut is detected, as presented for the first step in (b). Otherwise, the box is a leaf box (dashed), i.e. a part of the final constellation which is plotted in (c).

results. For efficiently computing convex hulls on a set of 2D points p , like our projections (see Fig. 2), we use a monotone chain algorithm [14]. Starting from the convex hull $CH(p)$ of the whole projection set p , we select those segments of the hull that exceed a given threshold in length. We thereby assume that those segments either span a non-convex region of the outer contour of the data, or that they represent a very straight edge. On these segments, we interpolate a number of sample points. Between each pair of points on each pair of segments, we simulate a cut that splits the point set p into two subsets p_1 and p_2 . The two segment points that minimize,

$$\Theta = [A(CH(p_1)) + A(CH(p_2))]/F, \quad (1)$$

where A is the area function for a convex hull and F the overall rectangular area of the face (see Fig. 2b), define our best split. An example of such a decomposition tree produced with the new hull algorithm is presented in Fig. 2.


















B. Evaluation

To be able to make a large scale test of the box decompositions stability, an algorithm was developed that estimates if two box decompositions are similar or not. First, the algorithm summarizes the total volume V_i of all boxes which a decomposition i is composed of. Secondly, it calculates the Euclidean distances between the centers of all pairs of leaf boxes and summarizes them as D_i . In order to determine if two compositions i, j are similar, the differences in overall volume and distance measures between the decompositions are simply compared with empirically found thresholds:

$$\text{if } |D_i - D_j| \leq 0.1 \wedge |V_i - V_j| \leq 0.9 \text{ then } similar(i, j). \quad (2)$$

TABLE I

PERCENTAGE OF DECOMPOSITIONS WITH SIMILAR COUNTERPARTS. 19 NOISE LEVELS AND 14 LEVELS OF POINT REMOVAL WERE USED.

3D																	
	Bunny	Car	Cup	Duck	Goblet	Goose	Heart	Homer	Horse	Human	Mug	PaperCup	Pen	Pillow	Radio	Squirrel	ToyDog
Old	78,62	100,00	77,78	61,59	21,21	24,28	100,00	55,07	53,99	19,93	80,43	100,00	91,67	91,67	84,62	59,42	27,54
New	94,00	100,00	97,78	91,67	22,77	46,00	100,00	66,67	30,33	58,00	62,67	74,24	92,00	100,00	100,00	75,00	28,33

To test the stability of the box decomposition algorithms, we simulated 17 different object models and modified them: 19 different levels of close proximity noise and 14 different levels of point removal were used for the experiment to let modified point clouds emerge from each original object point cloud. Both algorithms were then executed ($t=0.9$) on each of those point clouds before comparing the resulting box decompositions of each unmodified with its modified models. The results presented in Table I show that the previous algorithm is quite sensitive to noise. However, simpler objects like Car or Pen gave very good results. This is mainly because they all produced only one box due to their compact shape. On the other hand, more complex models like the toydog or the human model gave quite poor results. We note that the bound-based algorithm tends to produce a single large box enveloping the whole object also in such cases. This raises the similarity rate significantly, but is not preferred in our application.

The new hull-based algorithm produces much better approximation for the objects, very few single-box decompositions, and a significantly better similarity rate. The models that produced single-box decompositions with the bound-based algorithm produce worse values in some cases. This is caused by better approximations with multiple boxes that are more sensitive to the comparison than a single-box-to-single-box comparison. Since we prefer multi-box decompositions which give better object approximation, this is a good improvement, while the new algorithm is considerably less affected by noise.

The old and the new techniques are also compared to each other in Fig. 3 according to robustness to the change of the gain parameter t (for t , see II.1 and Fig. 2), e.g. the duck model decomposition repeatedly shows the same constellation. Another visible effect is that the decompositions seem more intuitive, e.g. in case of the cup handle.

III. 2D GRASP HYPOTHESES

In this paper, we are concerned with finding 2D grasps for 3D objects. Thus, we need to find a suitable grasping strategy based on the above mentioned box decomposition. We base our grasping hypotheses on the faces of the final box decomposition. The set of hypotheses is further reduced by including geometrical heuristics on which faces are valid in terms of visibility, reachability, and more [12]. For each leaf box in the hierarchy, the points enveloped by it are projected onto the valid faces of the box and stored in a grayscale image. The distance of the closest point to each pixel cell onto which it is projected is stored as a grayscale value between 0 and

255, where 1 is the depth of the box and 255 means zero depth (see Fig. 4a). This provides us with 2.5D representations of the object parts. The decomposition captures symmetries of objects quite well, resulting in faces and thus projections that are often perpendicular to the axes of most variance. This yields suitable information about approach directions of planar grasps and a good dissection of the object. In short, for each of the projections attained, grasps will be planned similarly to a top-view on a planar object. Thus grasp points on the contour of the projection images need to be found.

For grasp hypotheses from 2D contours, we will use an algorithm that is closely related to the work of Morales [4]. This algorithm involves a four step procedure for finding a number of grasp hypotheses, followed by a fifth to disqualify unfeasible grasps and selecting the best of the hypotheses.

A. Finding Good Regions to Grasp

We use the notion of grasp region, as defined in [4] and assume that a good region for grasping is a region that is as straight as possible. The fact that studies have shown that slightly concave curvature may be better suited [15] is left as a possible extension to the work. For this task a combination of the Canny edge detector and the k -angular bending algorithm [16] was used. First, the projection images described above are preprocessed by erosion and dilation steps. By removing pixels with fewer than 2 neighbours the number of outliers in the image is reduced. Expanding each remaining pixel (a projected point from 3D) to its neighbouring 8 pixels, gaps caused by sparse 3D point information are filled. Without these steps internal contours will be found that do not actually exist in the object and many grasp regions will be invalid. By using

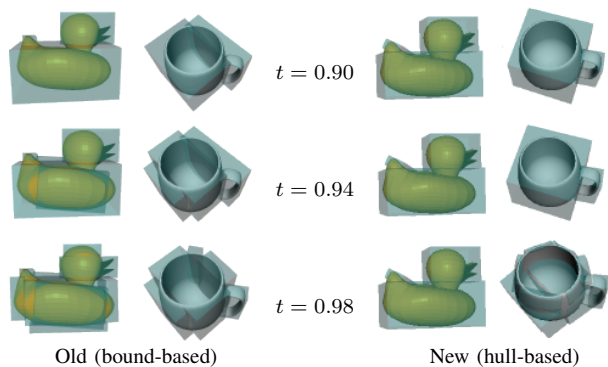


Fig. 3. Results of the box approximation for two models. Compared to the results produced with the bound-based algorithm [11] to the left, new hull-based constellations (right) stay more robust despite of different decomposition granularities (described by gain thresholds t in each row).

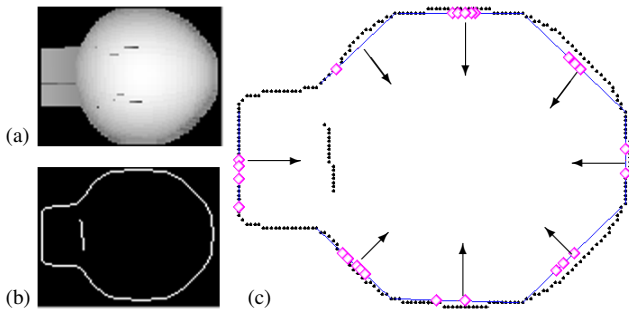


Fig. 4. (a) Projection image of the Duck model's head box (from above). (b) Canny edges image. Size of the Gaussian, lower and higher thresholds are automatically chosen by the Matlab edge algorithm. (c) A set of grasp regions (contour and 2D normal vectors) and grasp points (diamonds). The regions are found with $\sigma = 2.5$, curvature threshold $t_\kappa = 0.4$ (max. angle in radians), accumulated curvature threshold $T_\kappa = 4$, minimum length $l_{min} = 20\text{mm}$ (similar to Barrett finger width), and maximum length $l_{max} = 50\text{mm}$.

the edge detector with comparably high smoothing, see Fig. 4 for an example, we extract edges in the image.¹ These edges correspond to inner and outer contours of the projected object part as well as places where depth is rapidly changing. We assume that for these edges, grasps can be executed similarly to planar objects.

With the discrete edge points detected, these are ordered in a list following the contour. From the contour we will extract regions that satisfy four main conditions:

- 1) The curvature in any point of the region should be low,
- 2) the total accumulated curvature of a region should not be too high,
- 3) a minimum length of a region should be achieved, in order to reduce the number of hypotheses and reduce the effect of positioning errors,
- 4) a maximum length of a region should not be exceeded, in order to break long straight parts of the contour into several regions such that two fingers can be placed at the same side of an object such as a cube.

The curvature part is handled by the k -angular bending algorithm [16], that considers k neighbours in each direction of a point to determine its curvature by calculating the angles to these neighbours. Let C be the ordered list of points on the contour, $c_i = (x_i, y_i)$ the i^{th} point in this list, $\vec{a}_{i,k} = c_{i+k} - c_i$ and $\vec{b}_{i,k} = c_{i-k} - c_i$. The angle between these vectors is then calculated as,

$$\kappa_i = \arccos(\vec{a}_{ki} \cdot -\vec{b}_{ki}). \quad (3)$$

Convolving κ with a Gaussian provides smooth curvature values at any point along the contour. This removes remaining noise in the image so that a pixelated straight diagonal will not be discarded because the angle between each pixel and the next is too high. This enables us to assign a threshold on the local curvature, i.e. a region is only chosen considering that no point in it has a curvature value above the threshold.

An additional requirement for a region on the contour to be accepted is that the accumulated curvature of a region is

not higher than a chosen threshold. This condition is checked by summing up all κ -values for the region and comparing to the threshold. This takes care of problems with low constant curvature such as for a circle. Without the use of accumulated curvature, the circle would be regarded to have either no feasible regions to grasp or one region going straight through the center. With accumulated curvature, the circle will be broken into several regions. This also applies to other shapes with regions of low curvature with the same sign. Each region is approximated with the line connecting the endpoints of that region. Thus, each region is only represented by these two points and the inwards pointing normal, see Fig. 4.

The two remaining conditions for a region to be considered are the minimum and maximum length of a region. A minimum length is needed in order to account for positioning errors and to have a value close to the finger width. A maximum length is needed so that the representation does not become too simplified. If for a simple object like a cube no maximum length of regions was set, its projection images would only be represented by four regions that would be very hard to combine into a working grasp. By dividing the regions such that none are larger than the assigned maximum length produces more regions and therefore enables the possibility to place two fingers on one single side of the square, for example. Lower maximum length gives more regions and thus higher number of hypotheses, which means more possibilities. One should be cautious, however, since computation time increases rapidly with the number of regions.

B. Determining Finger Positions on the Regions

For each possible triplet (in the case of a 3-finger hand such as the Barrett hand [17] that is used) of regions, two criteria must be met. The normals must positively span the plane and finger placement must be such that all the friction cones of these fingers intersect. In this paper we will assume Coulomb friction and point contacts. By considering the union of all friction cones of one region and looking at the intersection of such 'combined friction cones', one can determine if the intersections of all three regions are empty and the hypothesis discarded, or non-empty and considered. This becomes a geometrical problem for each triplet and can be solved with standard linear programming methods. In the case of non-empty intersections, the centroid of the intersection area is calculated and projected back to the regions. These points will be used for finger positions, as discussed in [4].

C. Determining Hand Configuration

From the finger positions and the Barrett hand kinematics one can test if there is a configuration that can reach the selected points. By varying the angle of the thumb² to the surface and searching for those angles that correspond to configuration of the hand that can reach all three grasp points, one can find hypotheses for grasps. The angle of the thumb is varied on the interval $(-\arctan \mu, \arctan \mu)$ in 100 steps,

¹We use the Matlab standard function *edge* with parameter 'Canny' and a value for $\sigma = 2.5$ and automatically calculated threshold.

²Note that the thumb of the Barrett hand does not allow rotation. Thus, the angle of the thumb to the object is closely connected to the hand orientation.

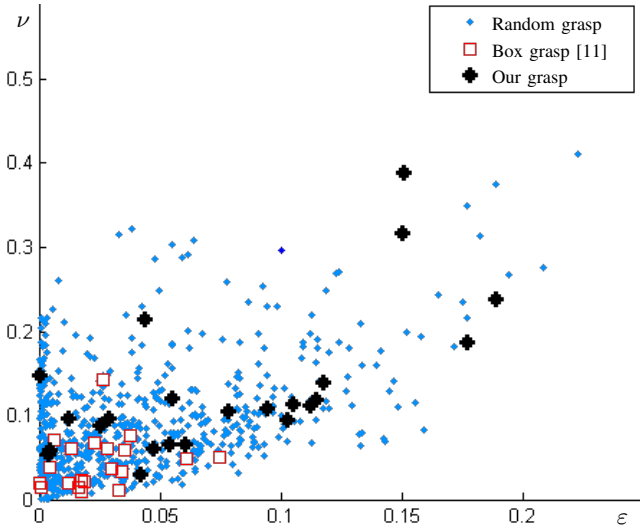


Fig. 5. Plot of 2000 random grasps, hypotheses found with the method from [11], and hypotheses found with our new approach, on the Bunny model. On the axes are the two built-in quality measures from GraspIt!. Only one grasp per finger positioning is chosen randomly and plotted. As can be seen, the best hypotheses are close to the best of the 2000 random grasps, suggesting that with only these 23 hypotheses one could get one or more good grasp.

where μ is the same friction coefficient used for calculating the friction cones in the previous section. Those configurations that satisfy the conditions are stored as grasp hypotheses. For a more detailed description, see [13]. A plot of the quality for grasp hypotheses found for the Bunny compared to 2000 random grasps is shown in Fig. 5.

D. Determining the Quality of a Grasp

From the algorithm presented one can generate a number of grasp hypotheses. However, we still need to determine which grasps are more likely to be successful. This is done by different measures of quality. Firstly however one needs to discard grasp hypotheses that are not reachable. This means that all grasp hypotheses outside of the physical reach of the hand will be discarded. This includes those grasps where one part of the object is in the way for grasping another part of the object. One example for the duck appears when a top grasp is attempted, but with finger positionings on the body of the duck: the head would be occluding the body, thus this grasp is discarded even before attempting it.

Many of the quantitative quality measures are the same as the ones developed by Morales *et al.* [4, 13], and will thus only be mentioned by name. There is one important difference, however: the empirical normalization constants used by Morales *et al.* will not be used here, as an artificial neural network will be used to determine the weights of each measure instead.

The measures derived from Morales are the following:

- | | |
|------------------------------|--------------------------|
| q_1 : Grasp Triangle Size, | q_5 : Finger Spread, |
| q_2 : Point Arrangement, | q_6 : Focus Deviation, |
| q_3 : Force Line, | q_7 : 2D Force Focus. |
| q_4 : Finger Extension, | |

These measures however are developed for planar objects. To adapt to non-planar objects to be grasped in our case we add two extra quality measures. These are:

1) *Finger Depth Difference*: The projection image contains information about the depth of the shape. Thus, it is possible to compare the selected grasp point depths d_i for each finger i with the linear approximations of the real finger extensions $g(e_i)$ by

$$q_8 = (g(e_1) - d_1)^2 + (g(e_2) - d_2)^2 + (g(e_3) - d_3)^2, \quad (4)$$

where $g(\cdot)$ is the linear depth approximation function.³ This measure depicts how close to the desired grasp points the grasp is likely to be. Note that this measure is the one that explicitly takes into account the 2.5D information provided by the box approximation and projection steps from Section II.

2) *3D Force Focus*: Ideally, one would like to measure the distance from the force focus in three dimensions to the actual center of gravity. This is, however, not possible since the information about the object is incomplete and the representations of grasps are only in two dimensions. We provide a rough approximation of this quality by using the center of the root box in the decomposition (containing all points in the point cloud) and the mean of the calculated finger positions in three dimensions,

$$q_9 = \|\bar{p}_{finger} - p_{rootCenter}\|. \quad (5)$$

IV. EVALUATION

The evaluation of the algorithm has mainly been made with data from simulation. This object data consists of 42 different 3D models, consisting of 14 different objects in 3 different scales to provide more data to train on. First, each model was decomposed with the box decomposition algorithm, using a gain threshold $t=0.90$. Over all models, this resulted in 570 projections from leaf boxes. 5951 grasp triplets were finally found from those projections and used as the data set for evaluation of grasps. Different types of results for the used models and decompositions are presented in Fig. 6. In a next step, the presented quality measures were computed, and grasp success measures extracted by simulating the grasps in GraspIt! [18]. The correlation between these quality measures and success measures is going to be learned by a neural network. We also explore how different network architectures affect the overall result.

A. Measure for Success

We want to produce a set of grasp hypotheses where the outcome is known in order to supervise the training. Since to do this with a real robot would be both time-consuming and costly in order to get enough data to train on, a more time and cost-efficient simulation option was used. By simulating the grasp hypotheses found for different objects, and by measuring the success for these in the simulator a set of input / output

³For the Barrett hand: $g(e) = 0.953 * e + 128.8$, empirically found. Using this linear approximation causes little loss in precision compared to calculating the actual inverse kinematics for the hand.

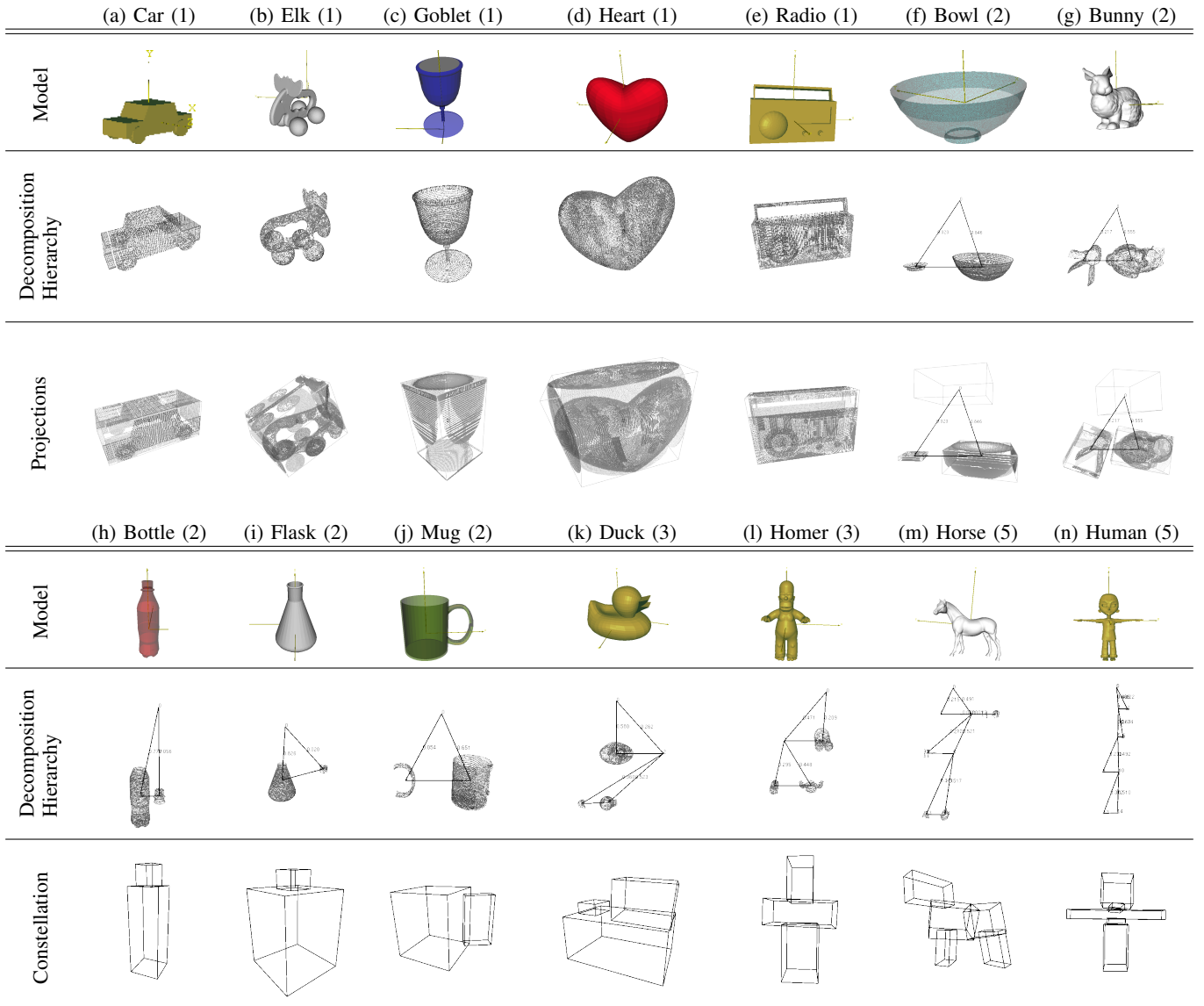


Fig. 6. The 14 models used in the experiment in order of complexity. The number of boxes resulting from the final decomposition hierarchies (2nd and 5th row) is assigned to each model in brackets. Since (a)-(e) are very compact and only the root box is included, the 3rd row visualizes examples of box face projections that will be used for the experiments. For the more complex models (h)-(n), the 6th row depicts the final box constellations. We refrained from showing constellations for (a)-(g), since they would only show 1-2 boxes, as also from showing projections for (h)-(n), since they would be hard to recognize.

pairs was found. The GraspIt! simulator [18] provides two different measures of success, introduced in [19].

The first measure, denoted ν here, measures the volume of the intersection of friction cones from the finger contacts. The second measure, called ε here, is a measure of the radius of the largest sphere that can fit in this space. For the purpose of learning it is more suitable to use only one measure since it is easier to learn a one-dimensional function than a two-dimensional. However, in order to utilize all of the information provided and since there is no firm consensus on which measure is best [20], we use a combination of the two:

$$s_i = (\nu_i/\nu_{max})^2 + (\varepsilon_i/\varepsilon_{max})^2, \quad (6)$$

where s_i is the success of grasp i , ν_{max} and ε_{max} are the maximum ν and ε values found for the current object and all hypotheses, respectively.

Other measures that could be used would be the division of grasps into two classes, namely successful and unsuccessful, or to train the system using only one of the above measures. Since potentially this could be a waste of useful grasp quality information, the above combination measure was chosen.

Given that the net is used to grasp an unknown object in the final application, the system will continue learning from newly observed hypotheses. However, since we have the possibility to initially gather vast amounts of simulation data to use for training, one additional example will not impact the prediction results noticeably. Combining this with the possibility to retrain an eager learner when the system is offline makes the advantages of a lazy learner, like k NN, diminish. Therefore, we used a feed-forward neural network to implement a supervised eager learning approach. The training algorithm used was the Levenberg-Marquardt backpropagation

algorithm included in the Matlab Neural Networks Toolbox. It is a fast training algorithm with good generalization properties, which is needed for predicting unknown objects.

B. Leave-One-Object-Out Validation

For evaluation, we apply a leave-one-object-out validation. This method validates by picking out one of the 14 objects that is not the test object, while all grasp hypotheses that belong to this object will be used as part of the validation set. There are both advantages and disadvantages to this approach. Considering that the prediction error for an unknown validation object is at a minimum, it would be intuitive to assume that the prediction error for an unknown test object would also be minimal. However, these two objects can be very different, both in complexity and suitable grasps, more than the ones in the training set and the test set. Another drawback with this technique is that the size of the validation set is different for each object used for validation. An advantage to using this approach is that it seldom overfits and thus stops the training when generalization still performs well.

C. Network Architecture

We used a network architecture with 9 input nodes, from the 9 quality measures, and 1 output node corresponding to the success measure s . From here, we still must decide how many hidden layers and how many hidden nodes in each layer to use. To be able to decide what is a good architecture and what is not we need to measure the overall success of the network. This measure should incorporate the s -measure for a grasp, but being as independent of an object as possible, e.g. an easy object to grasp will give better s -measures, but should (with the same prediction success) give the same success of the net, S_{net} . We want to rank the grasp hypotheses and only use the ones highest ranked. This is reflected in the network success measure by using only the top-ranked 10% hypotheses and comparing them with the lowest-ranked 10%:

$$S_{net} = \frac{(s_{high} - s_{low})}{0.1n} = \frac{(\sum_{i=1}^{0.1n} r_i - \sum_{i=0.9n}^n r_i)}{0.1n}, \quad (7)$$

where r_i is the ranked list of hypotheses, the hypothesis with highest predicted success being at index 1, and n is the total number of hypotheses.

To test the effect of adding hidden layers, three different setups were used: the first had only one hidden layer with 10 hidden nodes, the second had two hidden layers each with 10 hidden nodes, and the third had three hidden layers each with 10 hidden nodes. There was no improvement of prediction success for more than one hidden layer. The time for training, however, increased dramatically.

Extensive testing was performed in order to choose the number of nodes in the one hidden layer. This testing was done with the leave-one-object-out validation described above. Tests were made with 1 to 30 hidden nodes. After studying the performance results depicted in Fig. 7, the optimal number of hidden nodes was chosen to be 8. Using this architecture, no training phase in our experiment required more than 100 epochs.

D. Learning to Grasp Unknown Objects

In order to learn to grasp unknown objects, the leave-one-object-out validation method was applied again. Each time the network is trained one object is left out of the training data to be used for testing. This unknown object will be used to determine how well the algorithm has performed. In order to get a reliable result these tests were run 10 times. As such, one can conclude that the method for grasp synthesis, the quality measures and the learning approach used can indeed find and rank a set of grasps for an unknown object. Fig. 8 shows distribution of predicted success measures for each model. Some of the high-ranked grasps are presented in Fig. 9. These were encountered after separately performing a training on all other models in the set (except for one validation object). Note that the input for the overall approach is only a 3D point cloud representation that could also be delivered from real sensor input. The approach therefore does neither need training on every possible object model, nor does it rely on connected surface structure, like triangle meshes.

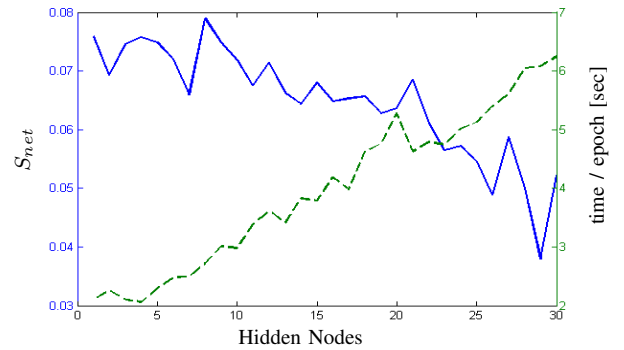


Fig. 7. Success (solid) and training time (dashed) for different number of hidden nodes with the leave-one-object-out validation, averaged over 10 runs. The maximum success value of $S_{net} = 0.078$ was detected at 8 nodes.

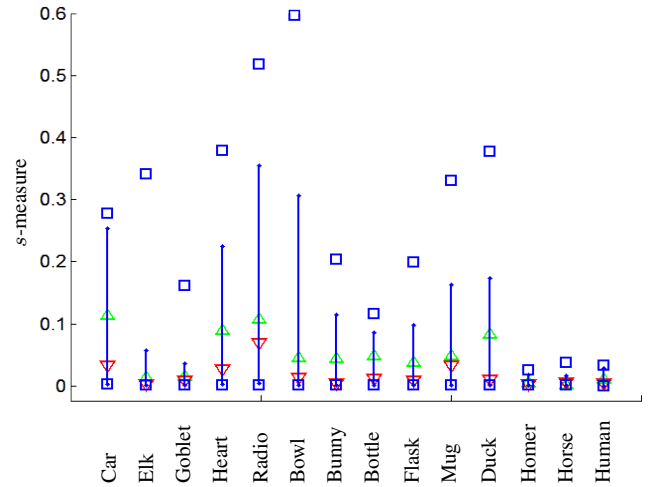


Fig. 8. Prediction results for the 14 models used (see Fig. 6). Upwards pointing triangles represent mean of the best 10% grasps, downwards the worst. Lines correspond to the possible span of predictions, with a perfect prediction of the best in the top and a perfect prediction of the worst in the bottom. The squares represent the best and the worst grasp for each object.

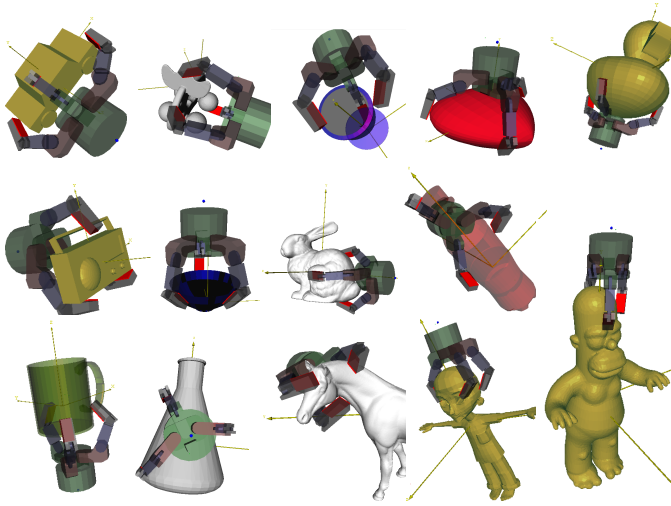


Fig. 9. Visualization of some high-ranked predicted grasps for all models.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we presented an approach for grasping 3D objects by using 2D grasping strategies and heuristics. We applied and extended approaches from each of these domains, namely the 3D box approximation [11] and the 2D grasping quality measures [4]. We showed that, given a point cloud of 3D data, an optimized version of box approximation produced repeatable decompositions in addition to resolving the issues encountered in our previous algorithm. This will contribute to further connected applications based on box constellation representations, e.g. learning from and grasping on whole constellations instead of just single boxes. Learning might also include a classification of the enveloped point cloud of each box and object part as another shape primitive, i.e. cylinders or spheres. Another classification will be approached by learning from the box constellation itself. Not only similarities between constellations could be used, e.g. all ‘duck’-like box decompositions afford similar types of grasps, but also finger positioning on more than one face will be enabled.

From a 2.5D representation such as the ones used here, one can produce a set of feasible grasp hypotheses. For these hypotheses one can evaluate a set of physically intuitive quality measures for a 3D object and use them for learning to predict success. It is important to note that representation, synthesis and evaluation are three independent parts and do not need the other parts to be present. The only requirement for a representation is that it has to contain information not only about the position in image space for a point, but also the depth. The grasp synthesis algorithm works independently of the other two and only needs the contour and the kinematics of the hand used. For the last step, most of the quality measures are extendible to all hands with the same or a higher degree of freedom than the Barrett hand used here. This can be done either by the use of virtual fingers, or by an extension of the measures themselves to include sums and differences for more than three fingers. A continuation of the work could include an extension of the quality measures to better take into account

3D shape. With the use of more flexible hands the complete inverse kinematics could be used for finding reachable points in 3D space. A natural extension to the learning part is to include not only data from simulation, but to continue learning from real-world objects. By retraining the network with the increased data set, the evaluation would get more precise and be a useful learning system.

ACKNOWLEDGMENT

This work was supported by EU through PACO-PLUS, IST-FP6-IP-027657.

REFERENCES

- [1] B.-H. Kim, B.-J. Yi, S.-R. Oh, and I. H. Suh, “Non-Dimensionalized Performance Indices based Optimal Grasping for Multi-Fingered Hands,” *Mechatronics*, vol. 14, pp. 255–280, 2004.
- [2] A. Saxena, J. Driemeyer, and A. Y. Ng, “Robotic Grasping of Novel Objects using Vision,” *International Journal of Robotics Research*, vol. 27, no. 2, pp. 157–173, 2008.
- [3] G. M. Bone and E. Y. Du, “Multi-Metric Comparison of Optimal 2D Grasp Planning Algorithms,” in *IEEE International Conference on Robotics & Automation*, 2001, pp. 3061–3066.
- [4] A. Morales, P. J. Sanz, A. P. del Pobil, and A. Fagg, “Vision-based three-finger grasp synthesis constrained by hand geometry,” *Robotics and Autonomous Systems*, vol. 54, pp. 496–512, 2006.
- [5] D. Aarno *et al.*, “Early Reactive Grasping with Second Order 3D Feature Relations,” in *From Features to Actions*, 2007, pp. 319–325.
- [6] 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.
- [7] M. Ciocarlie, C. Goldfeder, and P. Allen, “Dexterous Grasping via Eigengrasps: A Low-Dimensional Approach to a High-Complexity Problem,” in *Sensing and Adapting to the Real World*, 2007.
- [8] 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.
- [9] R. Pelossof, A. Miller, P. Allen, and T. Jebara, “An SVM Learning Approach to Robotic Grasping,” in *IEEE International Conference on Robotics and Automation*, 2004, pp. 3512–3518.
- [10] A. Chella, H. Dindo, F. Matraxia, and R. Pirrone, “Real-Time Visual Grasp Synthesis Using Genetic Algorithms and Neural Networks,” in *AI*IA 2007: Artificial Intelligence and Human-Oriented Computing*, 2007, pp. 567–578.
- [11] K. Huebner, S. Ruthotto, and D. Kragic, “Minimum Volume Bounding Box Decomposition for Shape Approximation in Robot Grasping,” in *IEEE Int. Conf. on Robotics and Automation*, 2008, pp. 1628–1633.
- [12] K. Huebner and D. Kragic, “Selection of Robot Pre-Grasps using Box-Based Shape Approximation,” in *IEEE International Conference on Intelligent Robots and Systems*, 2008, pp. 1765–1770.
- [13] A. Morales, “Learning to Predict Grasp Reliability with a Multifinger Robot Hand by using Visual Features,” Ph.D. dissertation, Department of Computer and Engineering Science, Universitat Jaume I, 2004.
- [14] A. M. Andrew, “Another Efficient Algorithm for Convex Hulls in Two Dimensions,” *Information Processing Letters*, vol. 9, pp. 216–219, 1979.
- [15] D. Montana, “The Condition for Contact Grasp Stability,” in *IEEE Int. Conference on Robotics and Automation*, 1991, pp. 412–417.
- [16] A. Rosenfeld and E. Johnston, “Angle Detection on Digital Curves,” in *IEEE Transactions on Computers*, vol. C-22, 1973, pp. 875–878.
- [17] W. T. Townsend, “The BarrettHand Grasper – Programmably Flexible Part Handling and Assembly,” *Industrial Robot: An Int. Journal*, vol. 27, no. 3, pp. 181–188, 2000.
- [18] A. T. Miller and P. K. Allen, “Graspi! A Versatile Simulator for Robotic Grasping,” *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.
- [19] C. Ferrari and J. Canny, “Planning Optimal Grasps,” in *IEEE International Conference on Robotics and Automation*, 1992, pp. 2290–2295.
- [20] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, “The Columbia Grasp Database,” in *IEEE International Conference on Robotics and Automation*, 2009.