Bayesian Grasp Planning

Kaijen Hsiao[†], Matei Ciocarlie[†], and Peter Brook[†*]

Abstract— **We present a Bayesian framework for grasp planning that takes into account uncertainty in object shape or pose, as well as robot motion error. When trying to grasp objects based on noisy sensor data, a common problem is errors in perception, which cause the geometry or pose of the object to be uncertain. For each hypothesis about the geometry or pose of the object to be grasped, different sets of grasps can be planned. Of the resulting grasps, some are likely to work only if particular hypotheses are true, but some may work on most or even all hypotheses. Likewise, some grasps are easily broken by small errors in robot motion, but other grasps are robust to such errors. Our probabilistic framework takes into account all of these factors while trying to estimate the overall probability of success of each grasp, allowing us to select grasps that are robust to incorrect object recognition as well as motion error due to imperfect robot calibration. We demonstrate our framework while using the PR2 robot to grasp common household objects.**

I. Introduction and Related Work

Robots trying to operate in human environments have to contend with many types of uncertainty. In particular, grasping and manipulation tasks can be affected by both perception uncertainty (such as incomplete data due to occlusions or incorrect object segmentation and recognition) and execution errors (such as imperfect trajectory following during arm motion). Grasp planning algorithms aim to increase reliability in unknown scenarios by producing grasps able to resist a wide range of disturbances, for example, using well-established quality metrics based on either the Grasp or Task Wrench Space [9], [4]. However, this type of analysis does not handle potential errors that, as the ones mentioned above, affect the grasp planning and execution process itself.

The uncertainty inherent in unstructured environments also means that different perception and analysis algorithms can provide different interpretations of the same scene. For example, an ideal recognition algorithm would always identify the correct object with 100% certainty. However, available methods commonly provide a list of possible results, each associated with a numerical measure of confidence (with units specific to the algorithm itself). Primitive fitters provide a measure of how shape primitives (such as spheres, boxes, cylinders, etc.) fit a given object. Different grasp planning algorithms, using different data representations, will have their own views of how an object can be grasped.

By combining this data in a probabilistic framework, we can take advantage of multiple sources of information, and produce algorithms better suited for handling uncertainty. In this paper, we show how to use the results from multiple object detectors and multiple grasp planning approaches, potentially running on different types of input data, in a way that is agnostic to the inner workings of each planning algorithm. We also describe a method of using pre-computed grasp information to replace expensive run-time computations when estimating how likely a grasp is to succeed, which is applicable to planners that can run off-line on a database on known object models.

In addition to the uncertainty associated with sensor data, grasp execution is also affected by potential calibration errors between the sensors and the end-effector, as well as imperfect following of a desired trajectory by the arm itself. This type of error can be mitigated, for example, through extensive calibration or controller tuning, but rarely eliminated altogether. We attempt to handle this case by extending the probability of success of each grasp to also take into account the range of possible outcomes for the respective execution commands.

II. Related Work

There is a long history of work that deals with the problem of planning robot motions and manipulation tasks under uncertainty, starting with preimage backchaining [14]. More recently, Berenson *et al.* [2] used Task Space Regions (TSRs) to generate manipulator trajectories that satisfy the requirements of a task despite robot pose uncertainty. Hsiao *et al.* [12] used a belief-based representation of object pose along with forward search through belief space to robustly execute specific grasps of known object shapes. Saxena *et al.* [16] used a probabilistic classifier to generate grasps and to predict their probability of success given features of an image and a point cloud of the object. Glover *et al.* used a probabilistic representation of 2-D shape to recognize and complete object outlines in an image for grasping. Finally, Balasubramanian *et al.* [1] examined how to improve the robustness of grasps using grasp measures derived from human-guided grasp demonstrations.

A database-driven grasp planning approach, including grasp evaluation across multiple objects of similar shapes, was recently discussed by Goldfeder *et al.* [10]. De Granville *et. al.* [7] also used mixtures of Gaussians over grasp position and orientation to represent functionally different grasp affordances, based on a database of human-demonstrated grasps; Detry *et al.* [8] represented grasp densities using a nonparametric kernel representation over grasp position and orientation, refined through robot grasp experiments.

This paper builds off of [5], which also examined grasp planning under object shape uncertainty and motion error, but which combined object recognition and grasp evaluation

[†]Willow Garage Inc., Menlo Park, CA. Email: {hsiao, matei, pbrook}@willowgarage.com

[*]University of Washington, Seattle, WA.

results in a more ad-hoc way that was limited to the particular recognizers and evaluators used; this paper addresses these limitations and makes the framework general enough to be used with arbitrary sets of object recognition and grasp evaluation algorithms.

III. Object Representations and Probabilistic Framework

Our general approach can be summarized by Figure 1. Consider the problem of a robot attempting to execute a grasp based on a perceived sensor image of a target object, shown in the top half of 1(a). The sensor image may be, for instance, a stereo camera point cloud, as in the bottom half of Figure 1(a). The robot has some set of object detectors that detect possible objects, (c) and (d), with associated confidence levels. The raw, segmented point cloud as well as the object meshes associated with the detected objects can all be used to plan grasps, as in (d, f, and h). Notice that due to (self-) occlusions, noise, and other imperfections of the point cloud, the original object (in this case a wine glass) is barely recognizable even to a human operator. A naive planning algorithm using only the first object detection result (the tennis ball can), or only the raw point cloud, has a significant chance of failure. Ideally, we would like to select a grasp that is likely to work on all possible object representations, weighing the importance of each representation and the likelihood of success of each grasp appropriately according to how much we believe each object representation to be correct, and at the same time, how confident we are that each grasp will work if its chosen representation is actually correct.

In order to select grasps in a principled way, we first create a set of possible grasps to choose from. From there, we can estimate the probability of success of each grasp, based on observations from various object detectors that tell us something about the identity of the object, and from various grasp evaluators that tell us something about how likely a grasp is to succeed, taking into account expected levels of robot motion error.

A. Probabilistic Model of Grasping

Here we present our probabilistic model for the success of a single grasp, g, which represents the 6D pose of the robot hand when grasping. We refer to grasp g as having a grasp success variable S, which has possible values successful (s) or unsuccessful (f).

We assume that we have a mutually-exclusive set of possible object representations for the object we are trying to grasp, O. These representations can be specific object geometries (e.g., meshes, point clouds, combinations of primitives), object poses, combinations of object geometries and poses, distinct object classes, or any other mutually-exclusive set of object representations that a set of grasp evaluators can be found to evaluate grasps on. This set can be generated for a particular situation from object detection results, or it can be fixed beforehand.

We also assume that we have some set of object detection results, D, to tell us something about how likely each object representation is. These could be results from vision-based object detectors, or they could be results from detectors based on other object characteristics such as object weight, texture, or even likelihood of being in a particular room.

Finally, we assume that we have a set of grasp evaluation results, E, that each tell us something about how likely g is to work on each of the possible object representations.

We wish to estimate $P(s|E, D)$, or the probability that the grasp g will succeed ($S = s$) based on our grasp evaluations and object detection results. The actual true object representation is unknown. However, we can take all of them into account:

$$P(s|E, D) = \sum_{o \in O} P(o|E, D)P(s|E, D, o) \tag{1}$$

Intuitively, we expect that $P(s|E, D, o)$ should not depend on D once given o, the actual object representation. Also, $P(o|E, D)$ should not depend on E, since for the types of grasp evaluations we are likely to obtain, a computed value for how well a grasp works on a hypothetical object representation should not affect which object representation we think is actually in front of our robot. Thus:

$$P(s|E, D) = \sum_{o \in O} P(o|D)p(s|E, o) \tag{2}$$

Using Bayes' rule:

$$P(s|E, D) = \sum_{o \in O} \frac{P(D|o)P(o)}{P(D)} \frac{P(E|s, o)P(s|o)}{P(E|o)} \tag{3}$$

$1/P(D)$ is a normalizing constant (α) that we can determine by computing both $P(s|E, D)$ and $P(f|E, D)$ and normalizing so that they sum to 1; but the other terms must be computed based on our data and models.

B. Bayesian Network Model

The Bayesian network models shown in Figure 2 compactly represent the independence assumptions that we just stated; it contains two pieces, each containing the object representation as a node, because of the constraint that $P(o|E, D)$ should not depend on E. These models as drawn add the additional independence assumptions that all object detections and grasp evaluations are independent of each other; this need not be the case, but does make the models much easier to deal with. The result of all of these independence assumptions is that we effectively compute the probabilities of each object representation in our set using a Naive Bayes model, and then use the resulting representation probabilities to compute the probability of grasp success.

In this instantiation of the model, we have $n + 1$ possible object representations, $\boldsymbol{O} \in o_1 ... o_n, o_{nd}$, where o_{nd} refers to none of the other object representations being correct.

We have a set of r observed object detection results, $D := d_1 ... d_r$, and a set of m observed grasp evaluation results, $E := e_1 ... e_m$ for the particular sensor scene and grasp. We also assume here that we have appropriate priors and

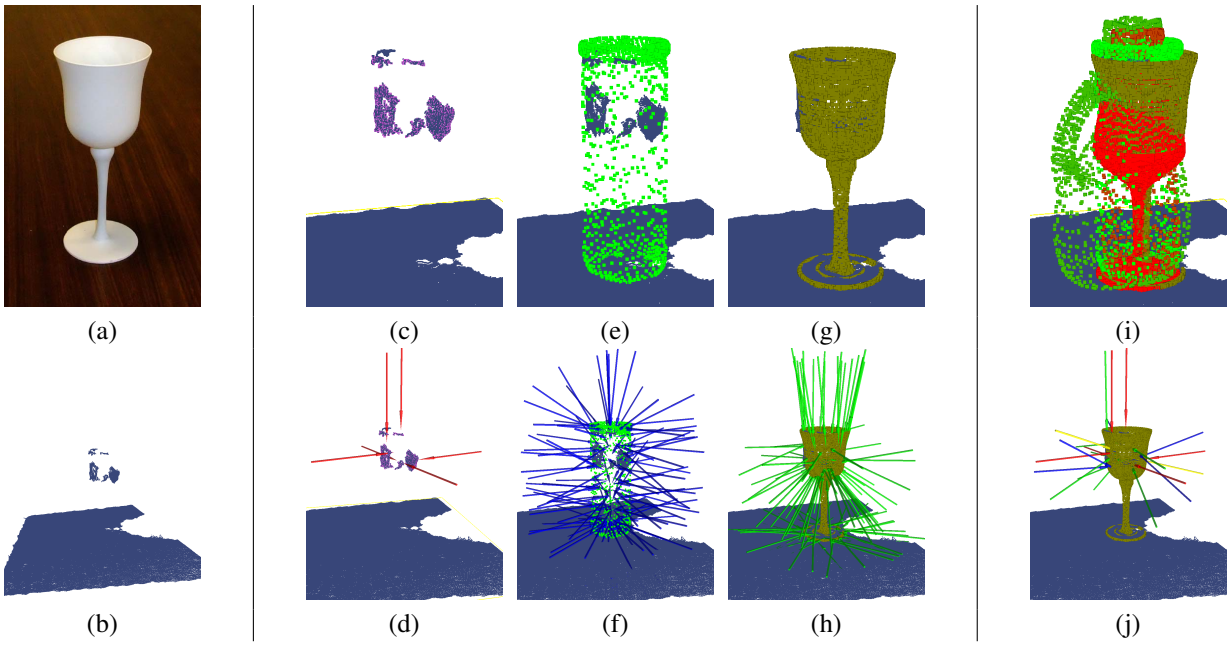

Fig. 1: Challenges in sensing for manipulation. **(a)** An object to be grasped, as well the recorded point cloud **(b)**. **(c)-(h)** Different object representations for grasping, and grasps planned using each representation. These include grasps planned based only on the segmented point cluster, and results of object recognizers, one incorrect shown in (e) and one correct shown in (g), along with the grasps planned using their associated object meshes in (d-h). **(i, j)** All detected object representations superimposed, along with a set of grasps that combines information from all of them.

conditional distributions to compute the joint distributions for any grasp g.

Based on the independence assumptions contained in our Bayes net model,

$$P(s|D,E) = \alpha \sum_{o_i} P(o_i)[\prod_{d_j} P(d_j|o_i)]P(s|o_i)[\prod_{e_k} \frac{P(e_k|s,o_i)}{P(e_k|o_i)}] \quad (4)$$

where $1 \leq i \leq n+1$, $1 \leq j \leq r$, and $1 \leq k \leq m$. $P(f|D,E)$ can be computed in a similar fashion, substituting f for s. $P(o_i)$ is a prior over the possible identities of the object, $P(d_j|o_i)$ is the conditional probability of having observed object detection result d_j given that the object representation o_i is actually correct, $P(s|o_i)$ is a prior on how often we would expect a grasp such as g to be successful on object o_i, and $P(e_k|s,o_i)$ is the conditional probability of having observed grasp evaluation result e_k given that the grasp of object o is of the successful variety.

If the two pieces of Bayes net were combined into one, the term $P(e_k|o_i)$ would go away; dividing by this term removes the influence of the computed grasp evaluations on the likelihoods of the different object representations. It can be computed as follows:

$$P(e_k|o_i) = P(e_k|o_i,s)P(s|o_i) + P(e_k|o_i,f)P(f|o_i) \quad (5)$$

All of the above values can be (ideally) estimated from data on previous grasps and object detection results.

By combining different object detectors and grasp evaluators in this manner, we allow them to reinforce each other's results, mitigating the risk of grasp failure due to incorrect scene interpretation. This encourages the selection of a grasp that would work well on the entire family of possible object representations. The goal is to avoid grasping particular features that belong to just one object representation, unless the associated recognition algorithm is extremely confident in its result.

Furthermore, using both success and failure conditional probabilites ensures that evaluators contribute to the final grasp success estimate with weights commensurate with their discriminative power; a weak evaluator can still add information without overly swaying the resulting grasp success probability.

In order for a set of object detectors, grasp planners, and grasp evaluators to be used in this framework, the following requirements have to be met:

- each object detector must be able to give an estimate of its confidence in detecting at least one object representation $o_i \in \boldsymbol{O}$, and must also be able to estimate (ideally based on actual detection data) conditional probabilities for how likely it is that it would state that level of confidence given that each possible object representation is correct in turn ($P(d_i|O)$);
- at least one grasp planner must be able to propose a set of potential grasps, in order to form a pool of grasps g to be evaluated;
- each grasp evaluator must be able to test a possible grasp, *i.e.*, using at least one representation $o \in \boldsymbol{O}$, must be able to generate a numerical evaluation of how well that grasp will do given that the object representation is

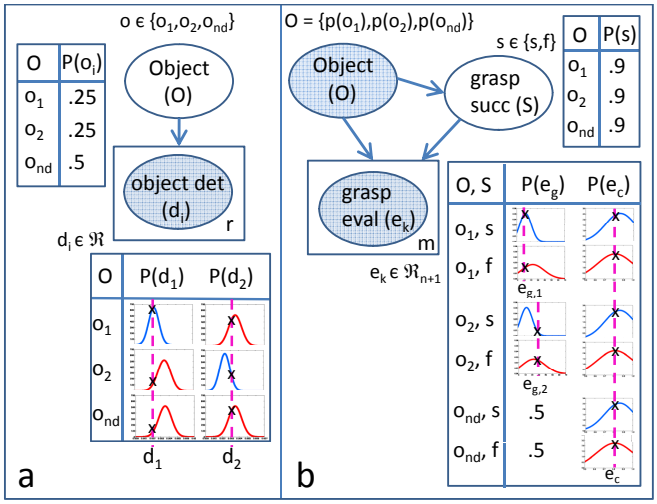

Fig. 2: The models used to estimate the probability of grasp success (S = s) for a single candidate grasp g, given r different results from object detectors (d_j, $1 \leq j \leq r$) and m different grasp evaluators (e_k, $1 \leq k \leq m$). a) Object representation probabilities are estimated from the various object detection results using a Naive Bayes model. b) Bayesian network model used to estimate grasp success probability given various grasp evaluations and the object representation probabilities.

actually correct, and must be able to estimate (ideally based on actual grasp data) how likely it is that each grasp evaluation is associated with a successful grasp, as well as how likely it is that the same evaluation is associated with an unsuccessful grasp ($P(e_k|O, S)$).

Ideally, each object detector would have separate conditional probability distributions for all possible object representations, but as a simplifying assumption can model all incorrect outcomes as being similar. Likewise, ideally, each grasp evaluator would have object-specific probability distributions for successful and unsuccessful grasps, but can model various objects as behaving similarly.

Figure 2 shows an example situation where there are two mutually exclusive object representations being considered (n=2), o_1 and o_2; o_{nd} refers to the case in which the object is neither o_1 nor o_2. In this example, there are r=2 object detectors. d_1 is the result from a detector that only detects o_1, and d_2 from a detector of only o_2; each detector models the conditional probability distribution over d_i for detecting its object ($o = o_i$) and for detecting some other object ($o \neq o_i$) as Gaussians. Grasp evaluator e_g has different evaluations for objects o_1 and o_2 but no opinion about the case where $O = o_{nd}$ (and thus contains uniform probabilities of .5 for that case), while evaluator e_c is based only on raw sensor data and is thus agnostic to object representation. As with the object detectors, the conditional probabilities of observing the particular grasp evaluation values are modeled as Gaussians, with one for successful grasps and one for unsuccessful grasps for each grasp evaluator.

C. Robustness to Execution Errors

To account for possible errors in grasp execution (due to imperfect robot calibration or trajectory following), we can consider a range of possible outcomes for a commanded grasp when computing $P(E|o_i, s)$ and $P(E|o_i, f)$ for a grasp g. Due to errors in execution, we assume that while attempting to execute a grasp g, we will actually end up at some perturbed grasp g_p. If we have a model for likely it is that we will end up at g_p for a given commanded grasp g on object o ($P(g_p|g, o)$), we can compute $P(E|s, o_i)$ as follows:

$$P(E|o, s) = \int_{g_p} P(E_p|o, s_p)P(g_p|g, o) \qquad (6)$$

where E_p is the set of grasp evaluation values for grasp g_p on the object representation o, and s_p refers to g_p being a successful grasp. In practice, the integral is likely to be difficult to compute, so we can instead sample the space of possible g_p and change the integral to a sum over samples, normalizing so that the sum of the sample probabilities ($P(g_p|g)$) is 1. $P(E|o, f)$ can be computed in a similar manner, substituting f_p for s_p and f for s.

IV. Implementation

In this section, we describe our specific implementation of the framework presented above. Currently, our implementation has object detectors for each object in our database, each of which detects only the one object. Objects that are detected with a reasonable probability are added to the list of possible object representations, along with the non-database (nd) object case. Because our object detector is quite good at matching the pose of the object model to the point cloud (in 2-D, which is all that is required for our test situations due to the limitations of the detector, as detailed shortly), our representations have only one pose per object model detected; if this were not the case, we could add additional representations to our set containing different poses of the same object model.

Our implementation uses two grasp generators: GraspIt! provides grasps that were pre-generated and stored based on the mesh models for object representations in our object database, and a point cluster grasp planner generates grasps for any situation, whether the object is recognized as potentially being in the database or not. All of the grasps generated by either type of generator are evaluated (on all available object representations) using our two grasp evaluators, which have both precise versions and data-driven regression versions. The precise versions happen to be the same evaluation functions used by GraspIt! and by the point cluster grasp planner for evaluating grasps during planning; the data-driven regression versions evaluate grasps much more quickly, based on sets of known good grasps, as outlined in V.

The hardware used for implementation is the PR2 personal robot. The features of the PR2 most relevant for this study include two 7-DOF arms, allowing multiple ways of achieving a desired grasp pose, and a narrow-field-of-view

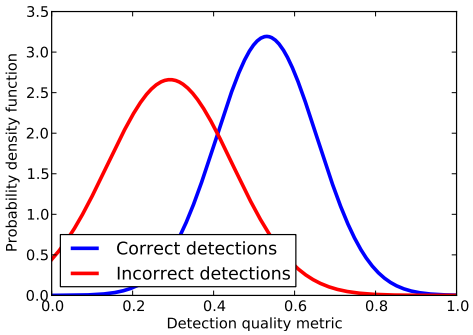

Fig. 3: Distributions of the detection quality metric for successful detections (the correct object was detected) and unsuccessful detections (an incorrect object was detected).

stereo camera equipped with a texture projector, providing dense point clouds of the robot's workspace (as illustrated in Fig. 1). Each arm is equipped with a parallel jaw gripper, which is the end-effector used for the implementation of the methods presented in this study.

A. *Object Recognition*

Object recognition is an extremely active field of research in its own right; the question of which algorithm (or even which type of sensor data) is currently best suited for grasping applications is beyond the scope of this study. In order to provide an implementation of our framework, we used a simple matching algorithm to seed the database-driven component. Once an object point cluster is segmented, an iterative technique similar to ICP [3] attempts to match it against each 3D model in our database.

The current matching algorithm only operates in 2 dimensions, and thus can only recognize objects sitting upright on a tabletop, either rotationally symmetrical or sitting in a known orientation. Furthermore, it only evaluates how well the points in a cluster match a given model, without reasoning about negative (or the absence of) sensed data. As a result, small point clusters will be considered excellent matches for large database objects, even if they only explain a small part of the mesh. We believe that such limitations underscore the importance of robust grasp planning. One of the strengths of the framework presented here is also its ability to integrate data from multiple sources. As more general and more reliable object recognition algorithms become available, they should directly benefit this approach to grasp planning.

One aspect which requires further attention is the output of a recognition algorithm, or its detection *quality metric*. In general, different recognition algorithms use different internal quality metrics; our approach returns the quality of the match between the point cluster and a 3D model as the average distance between the points in the cluster and their closest counterpart on the mesh. In order to convert this raw distance score into estimated conditional probabilitites of scores for correct and incorrect detection, we applied this algorithm to a set of 892 point clouds from 44 objects.

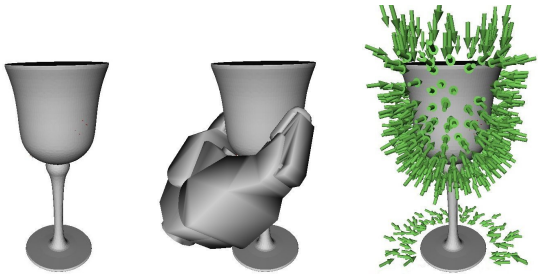

Fig. 4: Database-driven grasp planning. Left: object model; Middle: example grasp; Right: complete set of computed grasps.

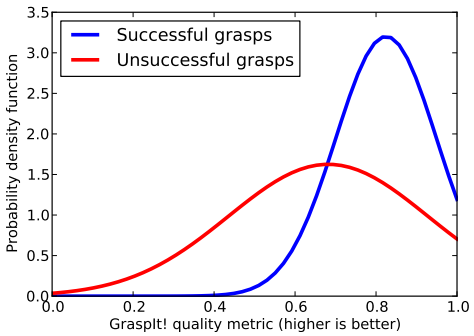

Fig. 5: Distributions of GraspIt!'s grasp quality metric for successful and unsuccessful grasps.

Detection scores are modeled as being generated from one of two Gaussians, one for correct and one for incorrect detections; the two class-conditional distributions are shown in Figure 3.

B. *Database-driven Planning*

For computing and evaluating grasps on 3D meshes contained in our database of models, we used the publicly available *GraspIt!* [15] simulator. To evaluate the quality of a given grasp, and to plan new grasps for an object, we used the energy metric and simulated annealing search described in [6]. An example is shown in Fig. 4.

Testing a novel grasp on a known 3D model can be performed explicitly, inside the simulator. However, this test, performed at run-time, is computationally expensive (on the order of 100ms per grasp). The alternative is to use the regression method presented in [5], where the set of known grasps G_r can be precomputed off-line. Fig. 4 also shows a complete set of reference grasps pre-computed for a given object.

As in the case of object recognition, in order to combine multiple grasp planners under a single framework, we must make the conversion from individual grasp quality metrics to measures that can be used in the Bayes net. In this case, we need to estimate $P(e_i|o,s)$ and $P(e_i|o,f)$, or the probability that the grasp quality metric is associated with a successful or unsuccessful grasp for a particular object. Again, to provide an analytical model for this conversion, we used a data-driven

approach, where 490 grasps generated using *GraspIt!* were executed on the real robot. Fig. 5 shows the Gaussians used to model both conditional probabilities.

C. Adding Robustness to Execution Errors

Next we would like to add robustness to errors in grasp execution to our estimates of $P(e|o,s)$ and $P(e|o,f)$ for each grasp, as in Equation 6). As mentioned earlier, the full integral would be very difficult to compute, so we instead evaluate the quality metric e both at g and at a set of u grasps $g_p \in G_p(g)$ that we refer to as the *perturbations* of g, or grasps that are other possible outcomes when the command for executing g is sent to the robot.

In general, e becomes a vector of grasp quality values, e_p, $1 \leq p \leq u$, instead of a single quality value. If we make the assumption that the grasps in $G_p(g)$ are the only possible grasp outcomes, estimates for $P(e|o,s)$ and $P(e|o,f)$ for a grasp g can be computed as follows:

$$P(e|o,s) = \sum_{g_p \in G_p(g)} P(e_p|s_p,o)P(g_p|g) \tag{7}$$

$P(g_p|g)$, the probability of actually executing grasp g_p when g is intended (normalized to sum to 1), depends on the calibration of the robot; we model it as being independent of o, even though this is not generally the case.

In this study, we built a simple error model by independently sampling one perturbation each along the positive and negative directions of the X, Y and Z axes for the position component of a grasp g. Each of the 6 perturbations was assigned a probability $P(g_p|g) = 1/9$, with $P(g|g) = 3/9$ (probability of correct grasp execution) completing the model. This model is equivalent to performing an unscented transformation over (X,Y,Z) with a center point weight $W^{(0)} = 1/3$ and standard deviation $\sigma = 0.47$cm [13].

V. Data-driven Regression

In this section, we describe a method for evaluating a grasp g on a given object representation o indirectly, based on how well it matches other grasps that are available for o. If we assume that $g_l \in G_o$ form a set of grasps for which the grasp quality metric e_l is known for each grasp g_l, we would like to evaluate g based on how well it matches the grasps in G_o.

This particular regression function is tailored for grasp databases that contain only positive examples (good grasps), rather than storing all bad grasps that may have been evaluated at some point. Thus we assume that regions with no grasps encode negative information; that is, a grasp in an empty region is likely to fail, and so estimated grasp quality should drop off sharply away from stored grasps (assuming that e_l is scaled to be between 0 and 1, with 1 being a good grasp and 0 being a bad grasp).

Recall that, in our implementation, the definition g of a grasp encodes the position and orientation of the gripper, which we will denote as p_g and q_g, respectively. Intuitively, two grasps are similar if they define two gripper poses that

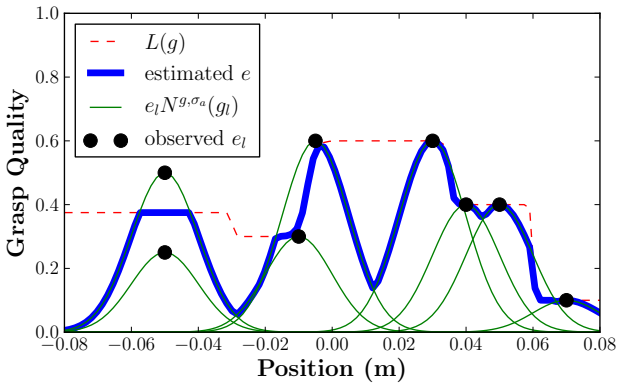

Fig. 6: Illustration of the data-driven regression function. The black circles represent known grasps from G_o, with the green bell curves showing the dropoff functions centered at each grasp. The Locally Weighted Linear Regression function $L(g)$ is shown by the dotted red line, and the resulting quality estimation function is shown by the thick blue line.

are close to each other in a Euclidian sense. We define the following distance function between two grasps g and g_r:

$$K(g,g_r;\sigma) = G(||p_g - p_{g_r}||;0,\sigma_p) * G(\theta_{q_g,q_{g_r}};0,\sigma_q) \tag{8}$$

where $G(x;\mu,\sigma_g^2)$ is a Gaussian with mean μ and variance σ_g^2, σ has position standard deviation component σ_p and orientation standard deviation component σ_q, and $\theta_{q_g,q_{g_r}}$ is the smallest rotational angle between the quaternions q_g and q_{g_r}. Note that this distance metric assumes that the position and orientation of the grasp are independent.

We compute a Locally Weighted Linear Regression function $L(g)$ with a Gaussian kernel over the quality metrics for the grasps in G_o as follows:

$$L(g) = \frac{\sum_{g_l \in G_o} e_l K(g,g_l;\sigma_a)}{\sum_{g_l \in G_o} K(g,g_l;\sigma_a)} \tag{9}$$

$L(g)$ essentially takes on the quality value of the nearest grasp, except where two grasps are within the range of the smoothing bandwidth σ_a, which in our implementation has values of $[0.0025m, 0.025rad]$.

We then estimate the quality e of a new grasp g as:

$$e = \min \begin{pmatrix} L(g), \\ e_l \max_{g_l}(K(g,g_l;\sigma_b)) \end{pmatrix} \tag{10}$$

Here, $K(g,g_l;\sigma_b)$ is employed as a smooth dropoff function, which lowers the estimated quality away from positive data points. Thus σ_b, which we will refer to as the *grasp bandwidth*, is a measure of the size of a region around itself that a given grasp informs, which in our implementation has values of $[0.01m, 0.001rad]$.

A. Cluster-based Planning

The second grasp generator and evaluator used in this paper relies solely on the observed point cloud of an object. As a result, it is applicable in a wide range of scenarios, as

Fig. 7: The set of test objects used in this study.

it does not depend on other components such as a model database, recognition algorithm, or primitive fitter.

We used the point-cloud-based grasp evaluation and planning algorithms presented in detail in [11]. This planner uses a set of heuristics, such as hand alignment with the cluster bounding box, size of the bounding box compared to gripper aperture, and number of observed points that fit inside the gripper, to assess the quality of grasp, and also to populate a list of grasp candidates. Because the point cluster is the same regardless of the object identity, $P(e|o,s)$ is the same for all o.

The conditional probability distributions used in our model are modeled using a bimodal distribution: a significant fraction of grasps have a quality score of 0, which makes the overall distribution non-Gaussian, so instead we split both success and failure conditional distributions into a mode with scores at 0 and a Gaussian mode covering the rest of the space: the Gaussian part of $P(e|s)$ has a mean of 0.68 and a variance of 0.26, and the Gaussian part of $P(e|f)$ has a mean of 0.74 and a variance of 0.22; 32% of successful grasps have a quality score of 0, as opposed to 73% of unsuccessful grasps. These parameters were computed from 3647 grasps planned on actual point cloud data, evaluated by the cluster evaluator, and then tested using GraspIt!. A grasp with a greater than 80% probability of success according to the GraspIt! energy metric was deemed to be successful.

VI. Experimental Results and Analysis

We now have implementations available for all the components of the framework of Eq. 4, with the additional option of using the regression-based evaluator outlined in V. Our test set, shown in Fig. 7, consisted of 25 objects common in human environments, such as dinnerware and containers of various shapes. The models of these objects were all part of the database used for object detection. The database also contained 45 additional models that, while not part of our real-life test set, were included in the detection results and used by the database-driven grasp planning component.

In order to analyze robustness to detection errors, we also performed tests where each object was temporarily removed from our model database, to simulate grasping non-database objects while retaining knowledge of actual object pose and shape for ground-truth evaluation. We refer to this condition as *novel-object* testing. We note that for objects belonging to a class well-represented in our set (such as cylinders), a similar but non-identical object was often available.

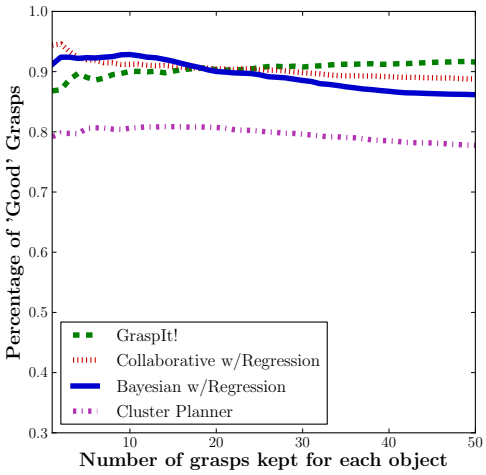

Fig. 8: Comparison of the top grasps returned by the Bayesian grasp planner described in this paper; the collaborative grasp planner described in [5], and the individual grasp evaluators (GraspIt!-based and cluster-based) used in both probabilistic frameworks, for 250 scans of objects contained in the object database.

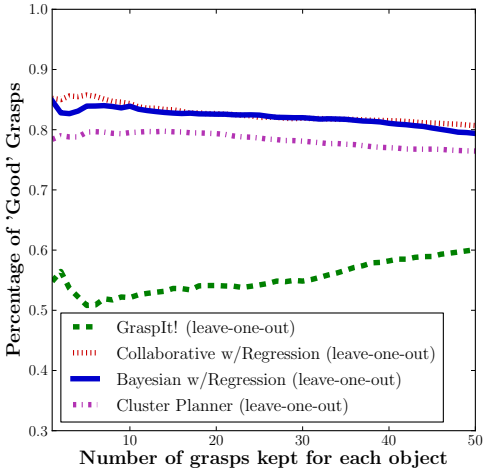

Fig. 9: Comparison of the same set of planners, for the same 250 scans of objects, but with each object in question temporarily removed from the object database (*novel-object* testing).

We tested the results of the Bayesian grasp planning framework described in this paper in simulation, using *GraspIt!* to evaluate the planned grasps. All the tests were run on real-life point clouds of the objects acquired with the PR2 robot's stereo cameras. For the simulated tests we used recorded real-life sensor data as input, manually annotated with ground truth information for object identity and pose.

A typical test of the grasp planner on one of the objects in our set used a recorded point cloud of the object sitting alone on a table as input to the planner, then used the simulation engine to test all the returned grasps. Each grasp

was evaluated in simulation on the ground-truth object model and location, and its quality metric and ground-truth success probability were evaluated as described in Sec. IV-B. The ground-truth success probability was then compared against the confidence estimated by the planner. It is important to note that, while the planner could only see a single point cloud of the object (similar to run-time conditions on the robot), the final ground-truth testing of the planned grasps in simulation was performed against a complete object model placed at the correct location in the world.

As mentioned earlier, this paper builds upon [5], in which a similar framework was presented that we call here the collaborative grasp planner; the implementation presented for that framework used exactly the same object detectors, grasp planners, and grasp evaluators described here. The method by which their results were combined, however, was somewhat ad-hoc and limited to exactly that set of detectors, planners, and evaluators.

Because the results in [5] showed that the regression-based GraspIt! evaluator generates roughly equivalent results to doing more expensive, precise evaluation by testing proposed grasps in GraspIt! directly, we used the regression-based GraspIt! evaluator in all of our tests. The cluster-based evaluator, on the other hand, computes its grasp evaluations quickly enough to use the precise version.

Figures 8 and 9 show results comparing the top 50 grasps returned by the Bayesian grasp planner, the collaborative grasp planner, the cluster-based planner, and a GraspIt!-based planner that returns grasps that are estimated to succeed more than 90% of the time for the top object detection result. The x-axis shows the number of grasps (x=2 means only the top two grasps returned for each of the 250 scans), and the y-axis shows the fraction of those grasps with a greater than 80% probability of success according to the GraspIt! energy metric when computed on the ground-truth object geometry.

Figure 8 shows results for all planners when the actual object in the scan is contained in the object database; the detection is correct the vast majority of the time, and so the GraspIt!-based planner does quite well. The cluster-based planner has no concept of object models, and only has a partial scan to go on, and thus its grasps succeed less often. The Bayesian and collaborative planners using inputs from both perform approximately as well as the GraspIt!-based planner; the top grasp tends to be slightly better because they are somewhat more resistant to the few object mis-detections, but the later grasps are somewhat worse because the cluster planner does not trust perfectly valid grasps of occluded parts of the objects.

Figure 9 shows results for all planners in the novel-object case; the object is not contained in the database, as is often the case with objects encountered in real life, and thus the GraspIt!-based planner does quite poorly; the cluster-based planner has the same performance as before. The Bayesian and collaborative planners perform better than either of their inputs, since they can make use of grasps from similar-looking objects in the database without also selecting grasps that are not supported by the partial object scan.

As you can see, the Bayesian and collaborative planners have very similar performance; the Bayesian framework, however, is more general, allowing the inclusion of arbitrary numbers of object detectors, grasp planners, and grasp evaluators, and also combines them in a purely data-driven fashion, as opposed to a hand-tuned, ad-hoc fashion. Because the simulation results for both planners are so similar, we did not carry out additional experiments comparing the Bayesian planner to the collaborative planner on the actual PR2 robot, expecting them to be nearly identical. In [5], we compared the performance of the collaborative planner with a naive planner that uses the GraspIt!-based planner when the recognition results were above a threshold, and the cluster-based planner when the results were below; the collaborative planner had the most significant impact in the case of novel-object testing (improving from 72% to 88% success rate on 25 objects), while also showing a smaller improvement in the case of database objects (from 84% to 88%).

VII. Conclusions and Future Work

We have presented a probabilistic approach to grasp planning based on the observation that real-life sensed data of an object is often incomplete and noisy, allowing for multiple interpretations of a scene. The different ways to interpret object data suggest different approaches to the grasp planning problem. Our framework attempts to combine these approaches and look for consensus among them, thus choosing final solutions (grasps) that are robust to errors in both perception and grasp execution.

In particular, the implementation that we have presented uses grasp suggestions from all available grasp planners, then uses a Bayes net model to estimate the probability of success of each grasp, combining information from multiple object recognition algorithms, as well as multiple grasp evaluation algorithms.

In our implementation, we employ both an object-model-based grasp planner/evaluator and a point-cloud-based grasp planner/evaluator, which, when used in our Bayes net framework, allows the overall system to handle situations ranging from completely unknown objects to tentatively recognized models to confidently recognized models in a consistent and principled way.

References

[1] R. Balasubramanian, L. Xu, P. Brook, J. Smith, and Matsuoka Y. Human-guided grasp measures improve grasp robustness on physical robot. In *ICRA*, 2010.

[2] D. Berenson, S.S. Srinivasa, and J.J. Kuffner. Addressing pose uncertainty in manipulation planning using task space regions. In *Intl. Conf. on Intelligent Robots and Systems*, 2009.

[3] P. J. Besl and M. I. Warren. A method for registration of 3-d shapes. *IEEE Trans. on Pattern Analysis*, 14(2):239–256, 1992.

[4] C. Borst, M. Fischer, and G. Hirzinger. Grasp planning: How to choose a suitable task wrench space. In *IEEE Intl. Conf. on Robotics and Automation*, pages 319–325, 2004.

[5] P. Brook, M. Ciocarlie, and K. Hsiao. Collaborative grasp planning with multiple object representations. In *ICRA*, 2011.

[6] M. Ciocarlie and P. Allen. Hand posture subspaces for dexterous robotic grasping. *Intl. Journal of Rob. Research*, 28:851–867, 2009.

[7] C. de Granville, J. Southerland, and A.H. Fagg. Learning grasp affordances through human demonstration. In *Intl. Conf. on Development and Learning*, 2006.

[8] R. Detry, E. Baseski, M. Popovic, Y. Touati, N. Krueger, O. Kroemer, J. Peters, and J. Piater. Learning object-specific grasp affordance densities. In *Intl. Conf. on Development and Learning (ICDL)*, 2009.
[9] C. Ferrari and J. Canny. Planning optimal grasps. In *IEEE Intl. Conf. on Robotics and Automation*, pages 2290–2295, 1992.
[10] C. Goldfeder, M. Ciocarlie, J. Peretzman, H. Dang, and P. Allen. Data-driven grasping with partial sensor data. In *IROS*, 2009.
[11] K. Hsiao, S. Chitta, M. Ciocarlie, and E. G. Jones. Contact-reactive grasping of objects with partial shape information. In *Workshop on Mobile Manipulation, ICRA*, 2010.
[12] K. Hsiao, L.P. Kaelbling, and T. Lozano-Pérez. Task-driven tactile exploration. In *RSS*, 2010.
[13] S.J. Julier and J.K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401 – 422, mar. 2004.
[14] T. Lozano-Perez, M.T. Mason, and R.H. Taylor. Automatic synthesis of fine-motion strategies for robots. *IJRR*, 1984.
[15] Andrew Miller and Peter K. Allen. GraspIt!: a versatile simulator for robotic grasping. *IEEE Rob. and Autom. Mag.*, 11(4), 2004.
[16] A. Saxena, L. Wong, and A.Y. Ng. Learning grasp strategies with partial shape information. In *AAAI*, 2008.