Learning a Real Time Grasping Strategy

Bidan Huang[2,1], Sahar El-Khoury[1], Miao Li[1], Joanna J. Bryson[2] and Aude Billard[1]

***Abstract*— Real time planning strategy is crucial for robots working in dynamic environments. In particular, robot grasping tasks require quick reactions in many applications such as human-robot interaction. In this paper, we propose an approach for grasp learning that enables robots to plan new grasps rapidly according to the object's position and orientation. This is achieved by taking a three-step approach. In the first step, we compute a variety of stable grasps for a given object. In the second step, we propose a strategy that learns a probability distribution of grasps based on the computed grasps. In the third step, we use the model to quickly generate grasps. We have tested the statistical method on the 9 degrees of freedom hand of the iCub humanoid robot and the 4 degrees of freedom Barrett hand. The average computation time for generating one grasp is less than 10 milliseconds. The experiments were run in Matlab on a machine with 2.8GHz processor.**

I. Introduction

Given an object and a multi-fingered robotic hand, generating a set of contacts on the object's surface which ensure grasp stability while being feasible for the hand kinematics is a common problem in grasp synthesis. Over the last few decades, robot grasping has been a popular topic and numerous approaches for grasp planning have been proposed [21]. Most of these approaches adopt iterative methods, which are usually able to find a solution within a finite number of iterations and the average computation time is usually in the scale of a few to tens of seconds. However the number of iterations required grows quadratically with the size of the problem and this creates an uncertainty of the time for the robot to plan a grasp. The upper bound of the computation time is barely analyzed in the literature.

Moving from the traditional engineering environment into a human dominated environment necessitates a fast grasp planning strategy to respond in real time. For example, when reaching out to grasp an object, a robust grasping strategy must be able to adapt rapidly to external perturbations that can modify the initial object position and orientation relative to the robot hand. In the case of catching a flying object [12], the robot has only a few milliseconds to plan a grasp before the object touches the floor.

Another application is receiving objects handed over by humans with a robot hand (Figure 1). In many circumstance the object must be grabbed quickly: one such example is when the object is heavy or hot; other examples involve time-pressing situations, e.g. in surgery a robot assistant must react sufficiently quickly to doctors handing back implements to ensure smooth running of the surgery.

[1]Learning Algorithms and Systems Laboratory (LASA), Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland

[2]Intelligent Systems Group (IS), Computer Science Department, University of Bath, UK

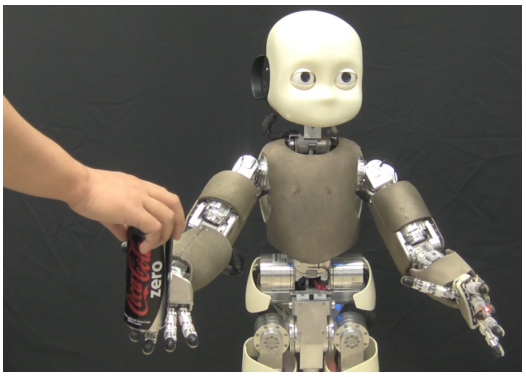


Fig. 1: A human hands a can to an iCub

Besides human-robot interaction, real time planning for the pick-and-place task in the industrial environment may also be necessary: spare parts could be randomly placed on the conveyer belt. The conveyer belt runs constantly at a high pace and leaves no time for the robot to stop its action and replan. The robot must therefore respond swiftly to avoid incurring delays in production. Given the limited computational power available on computers embedded in the robot, a computationally expensive algorithm would result in a prohibitively long decision time, leading to task failure in the above scenarios.

Because of the complexity of the problem, real time grasp planning has not been extensively studied. To tackle this problem, we propose a closed-form solution which requires at most three steps to compute a new grasp, and hence guarantee a short computation time and the uncertainty is reduced to the largest extent. This paper presents the proposed method and is organized as follows. Section 2 reviews the related work in grasp planning. Section 3 presents the proposed method: section 3A illustrates how we generate training data; section 3B explains the choice of the model and how it is built; section 3C details the real time strategy of generating new grasps from the model. Section 4 shows the experimental results, followed by the discussion in Section 5.

II. Related Work

In the robot grasping literature, the most extensively used mechanism for guaranteeing grasp stability is the force-closure criterion [16]. A grasp is said to achieve force-closure when the fingers can apply appropriate forces on an object to produce wrenches in any direction [22]. Optimal force-closure grasp synthesis is a technique based on this concept for optimizing the grasping performance by finding the contact point locations. Some approaches optimize an objective

function according to a pre-defined quality criterion [27], [28] in the grasp configuration space. These approaches do not take into account the kinematics of the hand, which is difficult to achieve. To bridge this gap, in our previous work [11] we proposed a one shot grasp synthesis approach that formulates and solves the problem as a constraint-based optimization. However, multi-finger grasps usually involve a large number of degrees of freedom. As a result, massive computing time is required to search an optimal grasp in the grasp space, considering the huge amount of possible hand configurations.

Instead of directly searching the high dimensional configuration space of robotic hands, this space can be reduced by generating a set of grasp starting positions, hand preshapes [15] or eigengrasps [2] that can then be tested on the object model. Such approaches reduce the dimensionality of the hand configuration space, but doing so implies a corresponding reduction in the accessible hand postures.

Other approaches avoid the complexity of computing kinematical constraints guaranteeing stable grasps by applying machine learning techniques to the problem. In imitation learning, some researchers use datagloves for human demonstration. The human hand configuration is then mapped to an artificial hand workspace and the joint angles [6], [8], or hand preshapes [13], [17], [26] are learnt. Some other researchers use stereoscopy to track the hand when a demonstrator is performing a grasp [10] or to match the hand shape to a database of grasp images [20]. These learning based approaches succeed in taking into account the hand kinematics and generate hand preshapes that are compatible with the object features. However they focus on different problems, such as telemanipulation [8] and human hand tracking [10], rather than real time unattended grasping. Compared to those methods which concentrate on generating a list of grasps for one object [13], [17], [24], [26], our method takes one step further: we learn a model from the list and use the model to quickly generate new grasps.

None of the previous work just mentioned addresses real-time planning. The reported computation time varies from 0.1 seconds to a few minutes. Recently, there have been some attempts to tackle the problem with real time solutions. Richtsfeld et al. [19] use a laser scanner to detect cylindrical shapes and plan grasps. This method is limited to cylindrical objects. Kanehiro et al. [9] use approximation models of the friction cone and roughly estimate the force closure criterion. However, this approximation may limit their solutions. In the planning step, they use random sampling techniques to generate grasping postures and loop through the samples to find a grasp satisfying all the kinematic constraints. The reported computation time varies from 10*sec* to 25*sec* including path planning of the arm using a 2GHz core. Daoud et al. [4] employ a genetic algorithm optimization approach to provide an initial grasp before online manipulation. This evolutionary approach relies on several iterations of optimization before reaching the solution. The reported time is 12.61*sec* for a spherical object with a 2.2GHz core. The latter two methods, due to their iterative approaches, do not guarantee fast computation in all cases. In contrast, with our closed-form solution the computation time is bounded within a few milliseconds.

III. Methodology

Traditional manipulation planning strategy usually involves inverse kinematics and optimization, which are computationally expensive. We avoid using these by adopting a learning approach. Our method starts by generating a training dataset of stable grasps. A *Gaussian Mixture Model* (GMM) [3] is learned from the data, and the target pose is predicted via *Gaussian Mixture Regression* (GMR). Hence there is no inverse kinematics computation nor iterative optimization in our method. Generally speaking, our approach is to:

1) Generate a set of stable grasping demonstrations for a given object and a robot hand (subsection A).
2) Build a statistical model for the training dataset offline (subsection B).
3) Use the model to quickly generate a new grasp, given a starting object-hand configuration (subsection C).

A. Grasp generation given the hand kinematics

Two robot platforms available in our lab are chosen to perform the grasping tasks: the iCub and the Barrett hand. The iCub has an anthropomorphic hand with 9 degrees of freedom: 3 in the thumb, 2 in the index, 2 in the middle finger, 1 in the ring and little fingers and 1 for the adduction/abduction movement. The Barrett hand is an industrial grasper with 3 fingers and 4 degrees of freedom: 1 for each finger and 1 for the separation between the second and the third finger. These two platforms differ drastically in the range of motion for each finger and provide very different grasp demonstrations.

There are numerous possible ways to grasp one object depending on the task's needs [11]. To encapsulate all the possible ways, a large amount of training data is needed. To collect this amount of data on a real robot is time consuming. Therefore, instead of using a real robot, we generate training data by synthesis. Two different approaches are used here: optimization and simulation.

1) Optimization: As the official iCub simulator does not provide a grasping module, we use the algorithm proposed in our recent work [11] to generate grasp training data. The iCub hand is modeled in 8 dimensions in this algorithm and the thumb, index and middle finger are taken into account. This approach formulates the problem as a constraint-based minimization for a set of hand configuration parameters (hand position $\boldsymbol{h}$, hand orientation $\boldsymbol{o}$ and finger joints $\boldsymbol{\theta}$). These parameters are subjected to a number of constraints to satisfy the following criteria:

1) The grasp is kinematically feasible for the robot hand;
2) The grasp is a force-closure grasp;
3) The robot hand is not penetrating the object;
4) The robot fingertips contact the object surface;
5) The force provided by the robot hand is able to raise the object.

The iCub's finger joints can only apply a limited amount of torque. The less joint torque required, the easier it is for the iCub to lift the object. For this reason, we choose the objective function to be the minimum joint torque required to balance the gravity wrench, formulated as:

$$J(\boldsymbol{h},\boldsymbol{o},\boldsymbol{\theta}) = \|\sum_{i,j}\tau_i^j\| \tag{1}$$

where τ_i^j is the ith joint torque of the jth fingers under the force feasibility constraints:

$$\tau_i^j \in [\bar{\tau}_i^j, \hat{\tau}_i^j] \tag{2}$$

where $\bar{\tau}_i^j$ and $\hat{\tau}_i^j$ are the lower and upper boundaries of τ_i^j. Minimizing this cost function is equivalent to minimizing the energy required in the joint space in order to accomplish the grasping task.

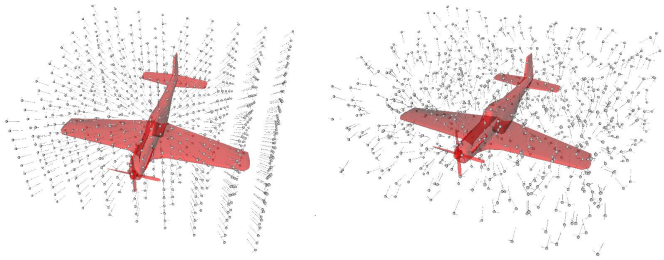
The optimization is solved by the Interior Point OPTimizer (IPOPT) method proposed by Wächter and Biegler [25], written in the AMPL Model Language for Mathematical Programming. To generate a variety of grasps, we exploit the fact that the IPOPT solver converges to local solutions. We provide the solver with a large number of initial conditions, varying from 1000 to 2000. From these initial conditions, which are located in different areas of the space, the IPOPT converges to their corresponding local optima. By this means 500 to 1000 optimized grasps for an object can be obtained. They will be used as the training data in the next phase. The average computation time for the IPOPT to converge to one solution is $2.65sec$, with a standard deviation of $1.82sec$. As additional information, the quality Q of each optimized grasp is calculated in the form described in [18]:

$$Q = \|\frac{1}{3}\sum_j \boldsymbol{c}^j\| \tag{3}$$

where $\boldsymbol{c}^j$ is the contact point (i.e. fingertip) position of the jth finger. Though it is not included in the optimization, the quality is used in the comparison between the training set and the result set shown in Section 4.

The algorithm above can generate a variety of high quality force-closure grasps for a given robot hand kinematic structure and a simple shaped object modeled by a superquadric. Since IPOPT is a continuous optimization solver, generating grasps on complex objects requires a continuous implicit representation of the whole object model. Representing complex objects as an assembly of superquadrics induces a discontinuity in this model preventing IPOPT from converging to a feasible solution. An implicit object representation for grasp generation using optimization will be addressed in our future work. This paper will only focus on grasps generated, for the iCub hand, on simple shaped objects such as a cylinder and cuboid.

2) Simulation: As the Barrett hand is modeled in the widely used simulator GraspIt! [14], we use simulation to generate its data. GraspIt! is designed for grasp analysis and it provides a library of robots and object models. Its quality measurement module computes the grasp quality according to all the contacts between the hand and the object, in the form described by Ferrari and Canny [7]. A grasp planning module for primitive shapes, i.e cylinder, sphere, cuboid and cone, is available, allowing users to easily generate grasps [15]. To sample grasps for objects with complex shapes, we alter the module and generate grasps as follows.



(a) Initial distribution (b) Final distribution

Fig. 2: An illustration of part of the grasp position lattice of an airplane model. Each grey dot in the lattice represents one robot hand position. The long arrows at each dot represent the hand normal directions and the short arrows represent the fix finger directions. The hand normals are initialized by pointing toward the center of the object, as shown in (a). Small random variance is then added to each grasp later to even the distribution and the final distribution is shown in (b).

Firstly a robot hand position "lattice" is generated. Each vertex in the lattice represents one robot hand position, where the hand will be placed to grasp the object (Figure 2). The object is located in the center of the lattice surrounded by the grasping positions. All palm normals are initially pointing to the center of the object. Random finger separation angles are assigned to each point to form a list of grasp configurations for testing. According to the object size, 1000 to 20000[1] testing grasps can be generated to ensure that the entire object is surrounded by the lattice and the farthest point to grasp the object is included. The density of the hand position lattice depends on the object shape. Objects with sharp edges, where the normals on the surface change sharply, should have a higher lattice density compared to those with smooth surfaces.

In the final step before testing, small random perturbations are added to each grasp so that the testing points are evenly and continuously distributed in all dimensions. To test these grasps, the hand is first placed at each position on the test list with the desired posture (hand orientations and finger joints). Next, the fingers clutch around the object until contacts or joint limits prevent further motion. We then use the quality measurement module to compute the quality of each grasp. The non-zero quality grasps, i.e. force-closure grasps, are recorded and used as training data. Note that not all the testing grasps result in feasible grasps. Points causing collisions are removed from the list and only the force-closure grasps are kept as the training data. The average generating rate for the feasible grasps is roughly one per five seconds.

The Barrett hand has one joint in each finger. These three joints can only rotate in one direction and how much they rotate is determined by the object surface, given the hand position, orientation and the separation angle. Therefore we drop this redundant information and model a Barrett hand

[1]Note that the number of samples for the Barrett hand is an order of magnitude larger than for the iCub because the target object shapes for the Barrett hand are more complex.

grasp only with the hand position, hand orientation and the finger separation angle. The robot kinematics is programmed into the simulator and all simulated robot movement is feasible.

In the above two methods, the size of the generated training data varies from 500 to 1600 (Table I). Each training dataset is split into 5 groups for the 5-fold cross validation in the later step.

B. Model learning

The second phase of the approach is to build a model Ω for the grasp demonstrations. A *Gaussian Mixture Model* (GMM) is used here to get a probabilistic encoding of the joint distribution $p(\boldsymbol{h},\boldsymbol{o},\theta \mid \Omega)$. We choose to use GMM because of its ability to effectively extrapolate the missing data, as has been exploited in many applications [1], [23]. It also has the advantage of capturing the non-linearity of the space, as well as determining how likely a point in the input space is under the model. The ability to estimate the likelihood of an input query point is crucial: an inference far away from the region covered by the training data can be unreliable, resulting potentially in an infeasible grasp. With GMM we are able to make sure that each input query point is located in or projected to a reliable region (this is explained in the next phase).

Given a set of sample grasps represented by the hand position $\boldsymbol{h}$, orientation $\boldsymbol{o}$ and the finger configuration $\boldsymbol{\theta}$, we model the distribution with a GMM as a sum of K Gaussian components:

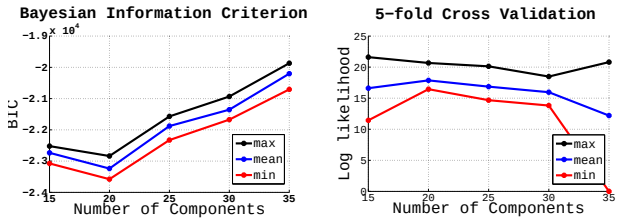
$$P(\boldsymbol{h},\boldsymbol{o},\boldsymbol{\theta}|\Omega) = \sum_{k=1}^{K} p_k p(\boldsymbol{h},\boldsymbol{o},\boldsymbol{\theta}|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k) \tag{4}$$

where k is the number of Gaussian components, p_k the prior of the Gaussian component and the $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ the corresponding mean and covariance as:

$$\boldsymbol{\mu}_k = \begin{pmatrix} \boldsymbol{\mu}_{h,k} \\ \boldsymbol{\mu}_{o,k} \\ \boldsymbol{\mu}_{\theta,k} \end{pmatrix} \quad \boldsymbol{\Sigma}_k = \begin{pmatrix} \boldsymbol{\Sigma}_{hh,k} & \boldsymbol{\Sigma}_{ho,k} & \boldsymbol{\Sigma}_{h\theta,k} \\ \boldsymbol{\Sigma}_{oh,k} & \boldsymbol{\Sigma}_{oo,k} & \boldsymbol{\Sigma}_{o\theta,k} \\ \boldsymbol{\Sigma}_{\theta h,k} & \boldsymbol{\Sigma}_{\theta o,k} & \boldsymbol{\Sigma}_{\theta\theta,k} \end{pmatrix} \tag{5}$$

A GMM approach requires that the data space is locally convex. For a complex object shape, however, the grasp space of hand configuration — coupled with the finger joint space and constrained by the geometry of the object surface — may be a non-smooth manifold. In both of the data generation methods described above, we evenly distribute the testing points so as to reduce the possibility of missing small good grasp regions. By these means we obtain most of the possible grasps for the object and approximate a locally convex data distribution, which is suitable for a GMM.

Before training we 1) convert all data into the object reference frame and 2) normalize the data so that all dimensions have a zero mean and a unit variance. Initialized by the K-means, the *Expectation-Maximization algorithm* (EM) [5] is used to find the value of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ that maximizes the probability of the training data under the GMM. The number of Gaussian K is selected by the *Bayesian Information Criterion* (BIC) and verified by 5-fold cross validation to make sure the model is not overfitting (Figure 3).



(a) BIC (b) 5-fold cross validation

Fig. 3: The *Bayesian Information Criterion* and 5-fold cross validation test results of the training dataset of the Barrett hand and a joystick shaped object. For each number of Gaussians, the test is run 5 times. After empirical testing, the number of Gaussians is chosen to be 20. The corresponding experiment are shown in Section 4.

C. Grasp Planning

With the learned GMM model of the grasping demonstrations, we plan a feasible grasp given a current hand configuration $\boldsymbol{q}=\{\boldsymbol{h},\boldsymbol{o}\}$. As discussed above, we first need to determine whether the $\boldsymbol{q}$ is a valid query point. To do this we define a membership function $f(\boldsymbol{q})$ as:

$$f(\boldsymbol{q}) = \sum_{k=1}^{K} \bar{\boldsymbol{N}}(\boldsymbol{q};\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k) \tag{6}$$

where $\bar{\boldsymbol{N}}$ is the normal distribution with the output being normalized between 0 and 1:

$$\bar{\boldsymbol{N}}(\boldsymbol{q};\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k) = exp\left(-\frac{1}{2}(\boldsymbol{q}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{q}-\boldsymbol{\mu}_k)\right) \tag{7}$$

We consider a point to belong to the model if its Mahalanobis distance to any Gaussian component is less than a given threshold σ. In our experiments, we find that within 1 standard deviation the success rate of finding a feasible grasp is constantly high. For example in the Barrett hand and the model plane grasping task, the rate of producing a feasible and stable grasp within 1 standard deviation is 85% (Table I) while it is 64% within 3 standard deviations. On the other hand, it is possible that GMM encapsulates two different clusters of data within a single Gaussian, leaving the mean of the Gaussian at an infeasible point. This means getting closer to the means does not ensure a higher success rate. Taking this trade-off into account, we choose 1 standard deviation as our threshold, which gives us a cutoff criterion $\eta = exp(-\frac{1}{2}\sigma^2)$. If the membership function of a point has a higher value than η, we consider this point as a valid query point. Note that the finger configuration $\boldsymbol{\theta}$ is not part of this input query point as $\boldsymbol{\theta}$ will be inferred by GMR later.

This membership function differs from the marginal likelihood $p(\boldsymbol{h},\boldsymbol{o})$ in two aspects. Firstly, it gives each Gaussian component the same weight, regardless of their priors p_k. As the prior of each Gaussian is proportional to the number of data points that are explained by this Gaussian, using this information in our selection may bias our choice toward the "most popular" grasps, yielding less variety in the results. Secondly, $\bar{\boldsymbol{N}}$ is a normalized function bounded between 0 and 1. This ensures the points with the same Mahalanobis distance from a Gaussian will have the same membership value, regardless of the size of the covariance [23].

In the case that $\boldsymbol{q}$ is not a valid query point, we need to project it to a new point $\boldsymbol{q}^*$ that has a membership function value higher than η. Here we use a closed-form solution by considering each individual Gaussian component. We first compare the Mahalanobis distances between the query point $\boldsymbol{q}$ and each Gaussian to find the nearest Gaussian component. The projection point $\boldsymbol{q}^*$ is found by projecting $\boldsymbol{q}$ to this nearest component (Figure 4). In the Mahalanobis space the Gaussian is in a uniform shape. As a result, the projection point $\boldsymbol{q}^*$ lays on the direction from the $\boldsymbol{q}$ to the center of the Gaussian. Therefore the projection point $\boldsymbol{q}_k^*$ of the k^{th} Gaussian can be written as:

$$\boldsymbol{q}_k^* = \boldsymbol{q} + \alpha_k(\boldsymbol{q} - \boldsymbol{\mu}_k) \tag{8}$$

where α_k is a scalar. With $\sigma = 1$ and the equation

$$\bar{\boldsymbol{N}}_k(\boldsymbol{q}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = exp(-\frac{1}{2}\sigma^2) \tag{9}$$

we have the equation to easily compute $\boldsymbol{q}_k^*$:

$$-\frac{1}{2}(\boldsymbol{q}_k^* - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{q}_k^* - \boldsymbol{\mu}_k) = -\frac{1}{2} \cdot 1^2 \tag{10}$$

Once the projection point $\boldsymbol{q}^*$ is found, the *Gaussian Mixture Regression* (GMR) is used to predict a feasible finger configuration $\boldsymbol{\theta}^*$ for it. First we define:

$$\boldsymbol{\mu}_{q,k} = \begin{pmatrix} \boldsymbol{\mu}_{h,k} \\ \boldsymbol{\mu}_{o,k} \end{pmatrix} \qquad \boldsymbol{\Sigma}_{qq,k} = \begin{pmatrix} \boldsymbol{\Sigma}_{hh,k} & \boldsymbol{\Sigma}_{ho,k} \\ \boldsymbol{\Sigma}_{oh,k} & \boldsymbol{\Sigma}_{oo,k} \end{pmatrix} \tag{11}$$

and GMR then uses:

$$\hat{\boldsymbol{\mu}}_{\theta,k} = \boldsymbol{\mu}_{\theta,k} + \boldsymbol{\Sigma}_{\theta q,k}(\boldsymbol{\Sigma}_{qq,k})^{-1}(\boldsymbol{q} - \boldsymbol{\mu}_{q,k}) \tag{12}$$

$$\hat{\boldsymbol{\Sigma}}_{\theta\theta,k} = \boldsymbol{\Sigma}_{\theta\theta,k} - \boldsymbol{\Sigma}_{\theta q,k}(\boldsymbol{\Sigma}_{qq,k})^{-1}\boldsymbol{\Sigma}_{q\theta,k} \tag{13}$$

Finally, all the K components are taken into account and the target finger configuration $\boldsymbol{\theta}^*$ is predicted as the mean $\hat{\boldsymbol{\mu}}_\theta$ with the covariance $\hat{\boldsymbol{\Sigma}}_{\theta\theta}$ according to:

$$\hat{\boldsymbol{\mu}}_\theta = \sum_{k=1}^{K} \beta_k(\boldsymbol{q}^*)\hat{\boldsymbol{\mu}}_{\theta,k} \tag{14}$$

$$\hat{\boldsymbol{\Sigma}}_{\theta\theta} = \sum_{k=1}^{K} \beta_k(\boldsymbol{q}^*)^2\hat{\boldsymbol{\Sigma}}_{\theta\theta,k} \tag{15}$$

where

$$\beta_k(\boldsymbol{q}^*) = \frac{p_k p(\boldsymbol{q}^*|\boldsymbol{\mu}_{q,k}, \boldsymbol{\Sigma}_{qq,k})}{\sum_{k=1}^{K} p_k p(\boldsymbol{q}^*|\boldsymbol{\mu}_{q,k}, \boldsymbol{\Sigma}_{qq,k})} \tag{16}$$

Due to the probabilistic nature of the GMR, the inferred result $\boldsymbol{\theta}^*$ is not a unique value but a mean value with variance. Though this mean does not guarantee a feasible solution, it provides a good estimation of a feasible one. The performance of this method is discussed in the next section.

IV. Experimental Results

This section presents a few results of our method (Figure 4[2], 5[3], 7). As mentioned above, grasps of the iCub hand are described in 14 dimensions: hand position (3D), hand orientation (3D in Euler angle), finger joint angles (8D), and grasps of the Barrett hand are described in 8 dimensions: hand position (3D), hand orientation (4D in axis-angle representation), finger separation angle (1D). Six different objects are presented here: cylinder, cuboid, ashtray, shoe, joystick and airplane model. For each object, three different initial postures and their final grasps are shown. Figure 4 shows the results of iCub grasping a cylinder, and the corresponding projections from the initial query points to the model. The results of the cylinder and cuboid show that a variety of grasps can be obtained for simple shapes to satisfy different task requirements. The ashtray, airplane model and joystick shapes are chosen from the GraspIt! object library, showing the method indeed works with complex shapes.

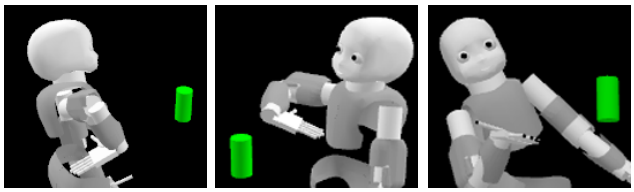
To test the computation time we generated 3000 random initial query points for each grasping task. The initial query points are placed at different distances away from the object surface, varying between $3cm$ to $50cm$, and the hand orientation is random. The initial finger configuration is not taken into account in finding the feasible region and hence it is set to the robot hand starting values. The computation time and experimental details are shown in Table I.

Table I also shows the success rate of generated grasps with the iCub and the Barrett hands. A grasp is considered to be successful if it satisfies the force-closure criterion, is feasible for the hand kinematics and is not in collision with the object (see Section 3A). When executing the obtained grasp, the fingers will continue to clutch until contact is made; if they contact the object surface before reaching the expected finger configuration, they will halt to avoid penetration. All the results shown in Figure 4, 5, 7 are good grasps.

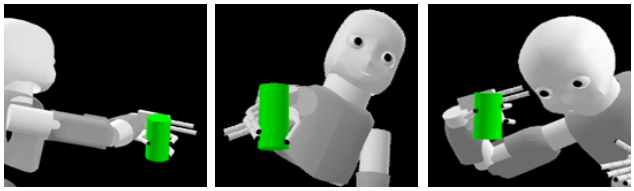
As can be seen from Table I, the success rate depends on the dimensions of the grasp space and the surface geometry of the target objects. Grasps in lower degrees of freedom (the Barrett hand) have higher success rates than those in higher degrees of freedom (the iCub hand). This suggests that the higher dimension grasp space is more complex than the lower dimension grasp space and needs more data to represent the full complexity. On the other hand, objects with smooth surfaces have a success rate around 90%. Objects with a couple of prominences have success rates over 85% as the configuration space of grasping is discontinuous. In the Barrett hand and airplane model task, the failed grasps are

[2]In some figures the wrist may seem to rotate over 360 degrees to reach the final grasps from the initial pose. This is because the path planning of the arm is not taken into account in our approach. In terms of the hand orientation solely, a much smaller rotation is needed to go from the initial pose to the final grasp.

[3]The small penetrations and gaps between the fingers and the object are caused by two factors, (1) that the width of the fingers are not taken into account in the optimization and (2) the variance of the results. A supplemental implementation will be applied on the real scenario to handle the variances.



(a) Initial pose 1 (b) Initial pose 2 (c) Initial pose 3



(d) Final grasp 1 (e) Final grasp 2 (f) Final grasp 3

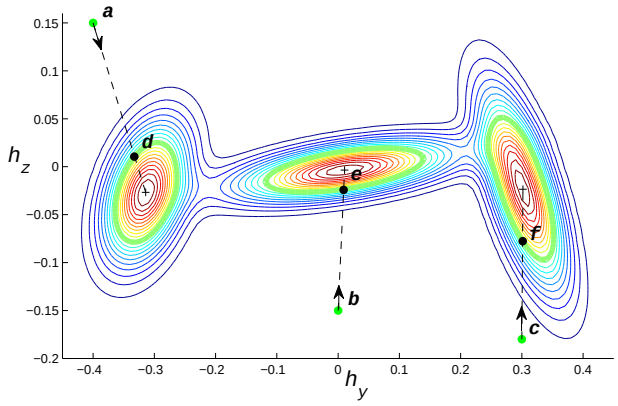
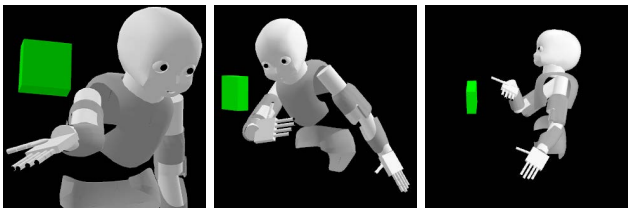
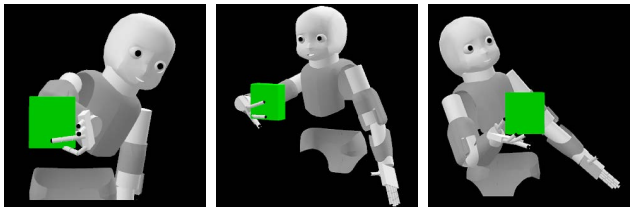


Fig. 4: Two-dimensional illustration of the learned model. h_y and h_z correspond to the hand position along the y and z axis of the object reference frame. a, b and c are the initial query points, while d, e and f are their corresponding computed grasps. Green dots correspond to initial query inputs $\boldsymbol{q}$, black dots to found feasible query inputs $\boldsymbol{q}^*$, contours to parts of the space with constant likelihood, and the thick green contour to threshold value $\eta = \exp(-\frac{1}{2}\sigma^2)$ of each Gaussian, where $\sigma = 1$ standard deviations. The initial finger joint angles in a,b,c are all set to zero. After each feasible query point is found, GMR is used to predict the finger configuration to get the final grasp d,e,f.



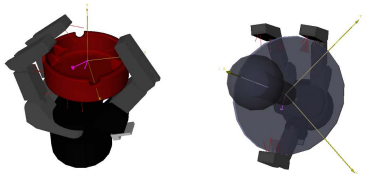
(a) Initial pose 1 (b) Initial pose 2 (c) Initial pose 3



(d) Final grasp 1 (e) Final grasp 2 (f) Final grasp 3

Fig. 5: Examples of iCub hand grasping of a cuboid. The first row (a,b,c) shows the initial postures and the second row (d,e,f) shows the corresponding final grasps.

(a) Best grasp found. Quality is 0.16. (b) Best grasp found. Quality is 0.19.



(c) Neighbor grasp of (a). Quality is 0.027. (d) Neighbor grasp of (b). Quality is 0.03.

Fig. 6: (a) The best grasp found for the Barret hand and the ashtray. (b) The best grasp found for the Barret hand and the joystick. (c) The nearest grasp of (a) in the training set. Note the gap between the finger and the object. (d) The nearest grasp of (b) in the training set.

concentrated on two places: the thin edges of the wings and the propeller. Grasping these places requires high accuracy and more training data on these parts would be needed.

To compare with the training data, we compute the grasp quality of the results with the same metrics we used in data generation. The mean of the grasp quality of the training set and the result set are similar, though the result set has slightly higher value in most of cases. We are able to find some grasps of higher quality than all grasps in the training set (Figure 6). This shows that GMM is able to model and generalize the high dimensional grasp space, especially for objects with smooth surfaces.

V. Discussion

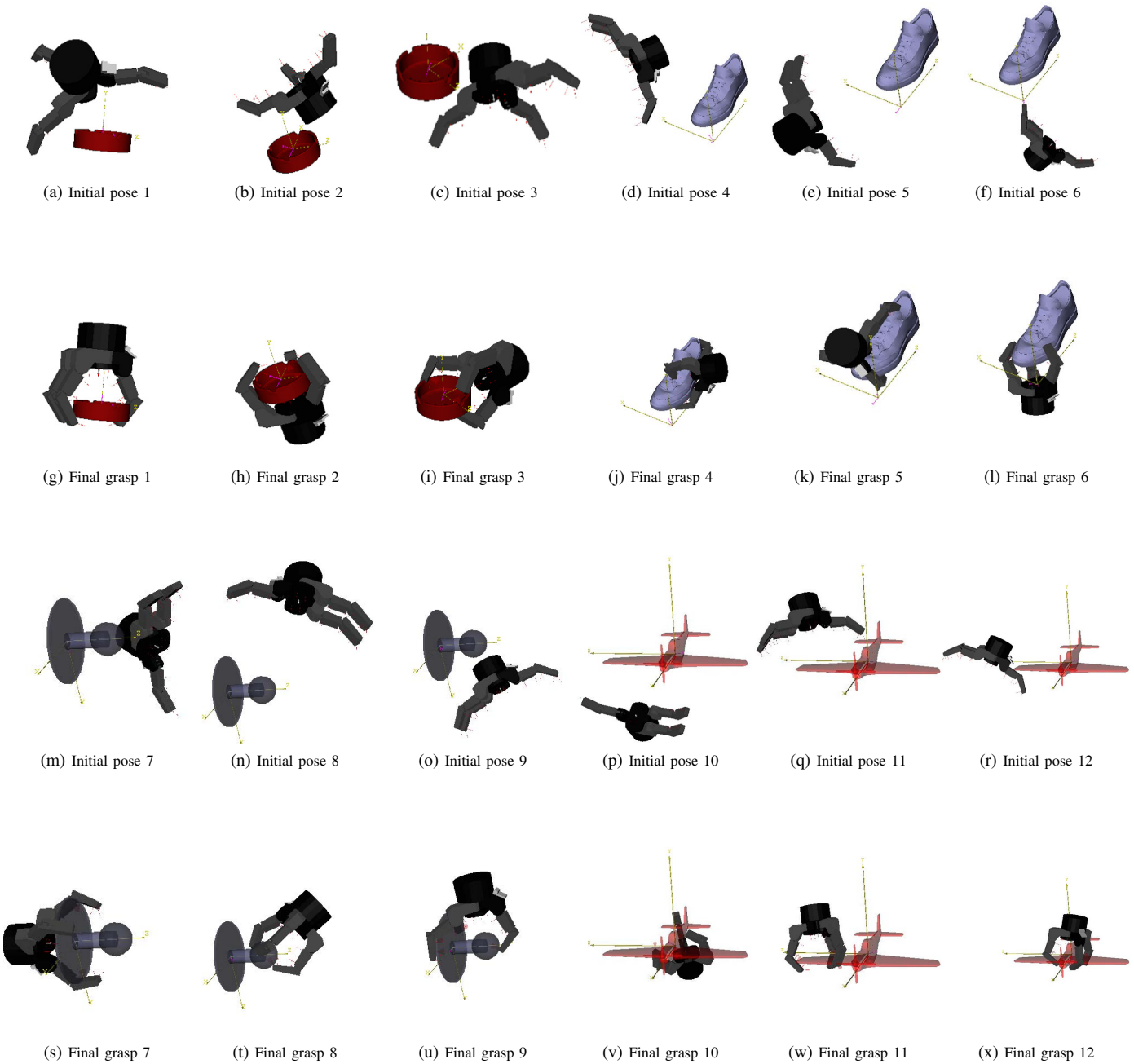
We presented a method for computing grasps in real time. This is demonstrated on two very different robot platforms: the iCub hand and the Barrett hand. The result shows that the method can capture the versatility of grasps that are typical of grasps performed by an industrial gripper, and those that can be performed by a humanoid hand. With the computation time in the millisecond scale, this method would enable the robot to react quickly in robot-human interaction, such as picking up heavy objects from a person's hand, as well as adopting to fast perturbations in a dynamic environment.

We achieve this goal by using a closed-form solution. A GMM model is learned from the grasping demonstration generated offline for a given object. During the online execution no iterative method is used, we only need to solve a few equations with basic arithmetic operations. Hence the computation time is significantly shorter than the conventional optimization methods. Though we need to pre-train the robot to grasp different objects, in many scenarios such as surgery assistance, robots and humans must work with a predefined set of objects. This allows us to build the grasping model for each object beforehand.

To find the closest Gaussian component we used the Mahalanobis distance rather than the Euclidean distance. The

TABLE I: Average computation time for generating new grasps for the iCub hand and the Barrett hand.

Robot/Object	Number of training data	Average Grasp Quality(train)	Number of Gaussians	Force-Closure Grasp Found	Average Grasp Quality(result)	Mean of Computation Time(*msec*)	Variance (*msec*)
iCub/Cylinder	621	0.0965	40	90%	0.1008	9.1	0.0001
iCub/Cuboid	532	0.1317	40	89%	0.1224	9.4	0.0007
Barrett/Ashtray	1560	0.0975	15	100%	0.1644	4.3	0.0001
Barrett/Shoe	629	0.0019	25	99%	0.0034	6.9	0.0001
Barrett/Joystick	1500	0.0061	20	98%	0.0064	5.9	0.0002
Barrett/Plane	1374	0.0002	55	85%	0.0003	15.1	0.0003



(a) Initial pose 1 (b) Initial pose 2 (c) Initial pose 3 (d) Initial pose 4 (e) Initial pose 5 (f) Initial pose 6

(g) Final grasp 1 (h) Final grasp 2 (i) Final grasp 3 (j) Final grasp 4 (k) Final grasp 5 (l) Final grasp 6

(m) Initial pose 7 (n) Initial pose 8 (o) Initial pose 9 (p) Initial pose 10 (q) Initial pose 11 (r) Initial pose 12

(s) Final grasp 7 (t) Final grasp 8 (u) Final grasp 9 (v) Final grasp 10 (w) Final grasp 11 (x) Final grasp 12

Fig. 7: Examples of Barrett hand grasping different objects (ashtray, shoe, joystick, airplane model). The first and third rows (a,b,c,d,e,f and m,n,o,p,q,r) show the initial postures and the second and forth rows (g,h,i,j,k,l and s,t,u,v,w,x) show the corresponding final grasps.

advantage of this is that it takes into account the correlations among each dimension of the hand configuration. In a space of different types of measurements, i.e. length and angle, Mahalanobis space is a better representation than the Euclidean space. Indeed, humans do not always use the Euclidean distance to select their grasps. We may move our hand further than needed to grasp an object, in order to avoid flipping our hand to another orientation.

Our approach provides a good estimation of a stable and feasible grasp for the given object and robot hand. To model the actual contact points between the robot hand and the object is difficult in real time because of the high dimension of the solution space and the non-linearity of the kinematic constraints. In our method, instead of computing the actual contact point position, we compute the most likely solution using a GMM. Though a certain amount of accuracy is traded off to achieve the real time goal, the overall performance is satisfying. In our experiments, over 90 percent of the testing points find good grasps within a few milliseconds. This method is most efficient for objects with a smooth surface. For complex objects this method can achieve a high success rate of over 85%. When grasping the parts requiring high precision, additional feedback from visual or tactile sensors is needed for further refinement of the grasp.

In contrast to the common approach of learning from human demonstrations, these grasps are generated solely according to the mechanics of the robot hand. Some resulting grasps are markedly different from human grasps, especially for the Barrett hand which is very different from the human hand. Our method may therefore out perform human demonstrations in some contexts by better exploiting differences between human and robot "physiology".

VI. Acknowledgments

The research leading to these results has received funding from the European Community Seventh Framework Program FP7=2007-2013 - Challenge 2 - Cognitive Systems, Interaction, Robotics - under grant agreement no [231500]-[ROBOSKIN], and grant agreement no [288533] [ROBOHOW] and by the Swiss National Science Foundation through the NCCR in Robotics.

References

[1] S. Calinon, F. Guenter, and A. Billard. On learning, representing, and generalizing a task in a humanoid robot. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 37(2):286–298, 2007.

[2] M.T. Ciocarlie and P.K. Allen. Hand posture subspaces for dexterous robotic grasping. *The International Journal of Robotics Research*, 28(7):851–867, 2009.

[3] D.A. Cohn, Z. Ghahramani, and M.I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[4] N. Daoud, J.P. Gazeau, S. Zeghloul, and M. Arsicault. A fast grasp synthesis method for online manipulation. *Robotics and Autonomous Systems*, 2011.

[5] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[6] S. Ekvall and D. Kragic. Learning and evaluation of the approach vector for automatic grasp generation and planning. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4715–4720. IEEE, 2007.

[7] C. Ferrari and J. Canny. Planning optimal grasps. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 2290–2295. IEEE, 1992.

[8] M. Fischer, P. van der Smagt, and G. Hirzinger. Learning techniques in a dataglove based telemanipulation system for the dlr hand. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1603–1608. IEEE, 1998.

[9] K. Harada, K. Kaneko, and F. Kanehiro. Fast grasp planning for hand/arm systems based on convex model. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1162–1168. IEEE, 2008.

[10] M. Hueser, T. Baier, and J. Zhang. Learning of demonstrated grasping skills by stereoscopic tracking of human head configuration. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2795–2800. IEEE, 2006.

[11] S. El Khoury, M. Li, and Aude Billard. Bridging the gap: One shot grasp synthesis apporach. In *Intelligent Robots and Systems, IEEE/RSJ International Conference on*. IEEE, 2012.

[12] S. Kim and A. Billard. Estimating the non-linear dynamics of free-flying objects. *Robotics and Autonomous Systems*, 2012.

[13] F. Kyota, T. Watabe, S. Saito, and M. Nakajima. Detection and evaluation of grasping positions for autonomous agents. In *Cyberworlds, 2005. International Conference on*, pages 8–pp. IEEE, 2005.

[14] A.T. Miller and P.K. Allen. Graspit! a versatile simulator for robotic grasping. *Robotics & Automation Magazine, IEEE*, 11(4):110–122, 2004.

[15] A.T. Miller, S. Knoop, H.I. Christensen, and P.K. Allen. Automatic grasp planning using shape primitives. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1824–1829. IEEE, 2003.

[16] V.D. Nguyen. Constructing stable grasps in 3D. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 234–239. IEEE, 1987.

[17] R. Pelossof, A. Miller, P. Allen, and T. Jebara. An SVM learning approach to robotic grasping. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3512–3518. IEEE, 2004.

[18] J. Ponce, S. Sullivan, A. Sudsang, J.D. Boissonnat, and J.P. Merlet. On computing four-finger equilibrium and force-closure grasps of polyhedral objects. *The International Journal of Robotics Research*, 16(1):11, 1997.

[19] Mario Richtsfeld, Wolfgang Ponweiser, and Markus Vincze. Real time grasping of freely placed cylindrical objects. In *ICINCO-RA (1)'08*, pages 165–170, 2008.

[20] J. Romero, H. Kjellstrm, and D. Kragic. Human-to-robot mapping of grasps. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, WS on Grasp and Task Learning by Imitation*, 2008.

[21] A. Sahbani, S. El-Khoury, and P. Bidaud. An overview of 3D object grasp synthesis algorithms. *Robotics and Autonomous Systems*, 2011.

[22] J.K. Salisbury Jr. *Kinematic and force analysis of articulated hands*. John Wiley & Sons, Inc., 1985.

[23] E.L. Sauser, B.D. Argall, G. Metta, and A.G. Billard. Iterative learning of grasp adaptation through human corrections. *Robotics and Autonomous Systems*, 2011.

[24] J.P. Saut and D. Sidobre. Efficient models for grasp planning with a multi-fingered hand. *Robotics and Autonomous Systems*, 2011.

[25] A. Wächter and L.T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[26] L. Ying, J.L. Fu, and N.S. Pollard. Data-driven grasp synthesis using shape matching and task-based pruning. *Visualization and Computer Graphics, IEEE Transactions on*, 13(4):732–747, 2007.

[27] X. Zhu and H. Ding. Planning force-closure grasps on 3-D objects. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 2, pages 1258–1263. IEEE, 2004.

[28] X. Zhu and J. Wang. Synthesis of force-closure grasps on 3-D objects based on the Q distance. *Robotics and Automation, IEEE Transactions on*, 19(4):669–679, 2003.