Cloud-Based Grasp Analysis and Planning for Toleranced Parts Using Parallelized Monte Carlo Sampling

Ben Kehoe, *Member, IEEE,* Deepak Warrier, Sachin Patil, *Member, IEEE,* and Ken Goldberg, *Fellow, IEEE*

***Abstract*—This paper explores how Cloud Computing can facilitate grasp analysis and planning with shape uncertainty by estimating lower bounds on achieving force closure. We consider the most common robot gripper: a pair of thin parallel jaws, and a conservative class of push-grasps allowing slip that can enhance part alignment for parts that can be modeled as extruded polygons. The grasp analysis algorithm takes as input a set of candidate grasps and perturbations of a nominal part shape. We define a grasp quality metric based on a lower bound on the probability of achieving force closure. The highly-parallelizable algorithm adaptively reduces the number of grasp evaluations and improves the lower bound by including slip. We develop a procedure for finding the effect of increasing tolerance in vertices on grasp quality, which allows part tolerances to be bounded to ensure minimum grasp quality levels. We find that ignoring shape uncertainty would result in a 42% drop in grasp quality versus our algorithm. We report computation times with local and Cloud-based execution and perform a sensitivity analysis on algorithm parameters. Our adaptive extension reduces grasp evaluations by 91.5% while maintaining 92.6% of grasp quality. We test a Cloud-based implementation on varying numbers of nodes, obtaining a maximum of 445× speedup with 500 nodes, suggesting our algorithm scales well with increasing parallelism. Further information including code and data is available at: http://automation.berkeley.edu/cloud-based-grasping**

***Note to Practitioners*—In manufacturing, small variations in part shape are inevitable; the range of allowable variations can be described using geometric or statistical tolerancing. This paper addresses the challenge of grasping parts with a parallel-jaw gripper where the true part shape is modeled with statistical tolerancing. We use statistical sampling to compute the grasp position and orientation that optimizes the probability of achieving a stable grasp. The computation requires many samples but can be parallelized and performed efficiently using Cloud computing. We consider a class of objects that can be modeled as extrusions of planar polygons and present algorithms and experiments using PiCloud, a commercial cloud computing platform, confirming that the approach scales well. In future work, the approach will be generalized to 3D parts with curved surfaces.**

***Index Terms*—Cloud Automation, Cloud Robotics, grasping, Cloud Computing, Monte Carlo sampling**

I. Introduction

AUTOMATION focuses on quality and reliability of processes in repetitive tasks. We present an approach to reliable grasp analysis and planning based on highly-parallelizable Monte Carlo sampling that enables cloud-based execution.

Manuscript received Tuesday 18th March, 2014.

Authors are with the University of California, Berkeley, CA, USA, (e-mail: {benk, sachinpatil, goldberg}@berkeley.edu).

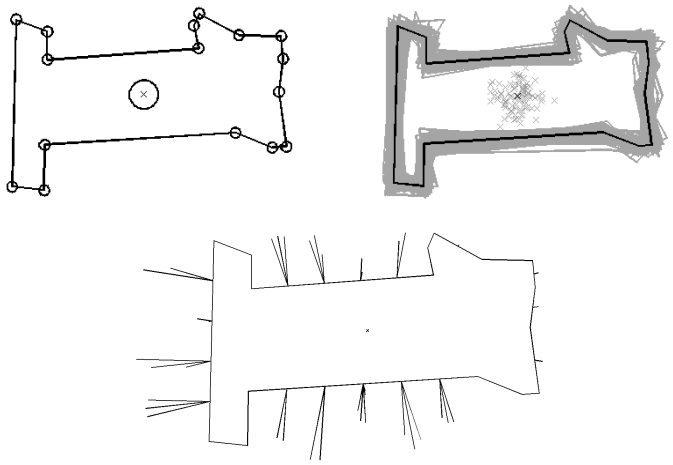

Fig. 1. Part tolerance model and example results. On the upper left, circles with a radius of one standard deviation of an isotropic Gaussian distribution are drawn around each vertex and the center of mass. On the upper right, the nominal part is plotted over 100 sampled perturbations (shown in gray). The lower center is a sample "whisker diagram", which is used to show algorithm results. Each line segment represents a candidate grasp, and indicates its contact point on the part. The line segment indicates the direction of approach for the grasp, and is orthogonal to the gripper jaw. The length corresponds to the lower bound on the probability of a stable grasp.

A fundamental challenge, even with perfect recognition, is variation in part shape, because of manufacturing constraints, and variation mechanics, because of limits on sensing during grasping.

The need to determine robust grasps is especially important in Automation, where the cost of failure can be high, but is offset by the ability to perform extended analysis offline. This situation is ideal for Cloud Computing, where vast computing power is available but high latency impairs real-time operation.

This paper describes a method that leverages Cloud Computing to analyze grasps on 2D polygonal parts with shape tolerances. We take a conservative approach: we use a statistical sample of part shape perturbations to find the value of a quality metric that estimates a lower bound on the probability of force closure for a class of grasps called *conservative-slip push grasps*, which can be rapidly evaluated without simulation. We then combine the results of the retained candidate grasps, weighting their success on a given part perturbation by the probability of that perturbation, to estimate a lower bound on the probability of achieving force closure.

We provide a grasp planning algorithm that uniformly samples from our simplified grasp configuration space on a

simplified version of the part shape. We improve the grasp planning by adaptively reducing the candidate grasp set after testing a small number of part perturbations, reducing the overall number of grasp evaluations.

We explore properties of the algorithm by performing a sensitivity analysis on the parameters of the algorithm, determining the effect of these parameters on grasp quality; by evaluating the adaptive grasp reduction; and by developing a procedure for finding tolerance bounds based on a quality threshold. We evaluate the scalability of the algorithm in Cloud-based parallel execution in Section VI.

This journal paper is an expanded and updated presentation of research first described in a series of papers at ICRA [27] and CASE [28], and includes a Cloud implementation and results.

II. Related Work

In "Algorithmic Automation" [20], abstractions can allow the functionality of automation to be designed independent of the underlying implementation and can provide the foundation for formal specification and analysis, algorithmic design, consistency checking and optimization. Algorithmic Automation thus facilitates integrity, reliability, interoperability, and maintainability and upgrading of automation.

Several studies use contact sensors to improve grasp quality in the presence of uncertain part geometry [18] [24]. However, many robotic grippers do not have contact sensing capability. Sensing is often implicitly assumed to be present, such as when pinch grasps are required, since the part must not be moved by contact with the gripper [11] [29] [42] [43].

Studies have explored properties of polygonal parts for grasping [9] [10] [14], but focus on point grasps, which ignores the complex interaction created by a gripper of nonzero width, as is the case with parallel-jaw grippers.

Push manipulation of parts has been extensively investigated by Mason [32] and others [4] [31]. Performing pushing operations with a gripper to reduce pose uncertainty has been demonstrated by Dogar and Srinivasa [16]. However, these methods, again, do not take into account part shape tolerance.

Similarly, many recent studies in robotic grasping focus on improving grasps on known parts [39] [40] [41] that do not take into account tolerances. The work in robotic grasping that addresses tolerance largely focuses on part pose [6] [16] [36]. Methods for sensorless part orientation [7] [21] [45] can also be used in the presence of uncertain part pose. However, these methods do not take into account tolerances for the geometry of the part.

While networked automation has a long history [22], only recently has research focused on networked robots sharing information to accomplish tasks widely separated in time and space [33] [44]. The introduction of Cloud Computing can allow computation to be offloaded from robots [5], as well as development of databases that allow robots to reuse previous computations in later tasks [13]. Grasping could benefit from this effort, since grasps computed for a part can be applied to similar parts encountered later [12] [19] [23]. This allows the construction of grasp databases that can be shared and referenced by multiple robots [23] [30].

An explicit part tolerance model for grasping was proposed by Christopoulos and Schrater [11] that approximates the part boundary with splines but does not account for motion induced by contact from the gripper. Models exist for tolerance [8] [25] that use worst-case bounds rather than probability distributions. Other work defines topological tolerance models but does not apply it to grasping [38].

III. Problem Statement

We consider a parallel-jaw gripper, gripping a part from above. We assume that we have a conservative estimate of the coefficient of friction between the gripper and the part, denoted μ.

We assume that the part can be modeled as an extruded polygon to be gripped on its edges, resting on a planar work surface, and that the part has an estimated nominal center of mass, which may not be at the centroid. The gripper–part interaction is assumed to be quasistatic, such that the inertia of the part is negligible [35].

A. Part Tolerance Model

Part shape tolerances are modeled as independent Gaussian distributions on each vertex and center of mass, centered on their nominal values, as shown in Fig. 1. The variance of the distributions is an input, denoted Σ, which may be dictated by manufacturing constraints. One advantage of using probability distributions is that we can use a Monte Carlo approach to evaluate the effect of higher tolerances on candidate grasps.

We denote the space of possible parts as $\mathbb{S}_0$, the space of possible perturbations of a shape $S \in \mathbb{S}_0$ as $\mathbb{S}(S)$. Note that for any $S \in \mathbb{S}_0$, $\mathbb{S}(S) \subseteq \mathbb{S}_0$. We further denote the space of all (part, part perturbation) tuples as $\widehat{\mathbb{S}} = \{(S_0, S) \,|\, S_0 \in \mathbb{S}_0, S \in \mathbb{S}(S_0)\}$.

The input to the algorithm is a list of edges defining a non-intersecting polygon, denoted S_0, and the variance Σ of the Gaussian tolerance distributions for the vertices and center of mass.

B. Contact Configuration Space

The contact a gripper jaw makes with a part is defined by the ordered pair $c = (p, \phi)$, where p is the *contact point*, a point along the one-dimensional boundary of the part, and $\phi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ is the *approach angle*. The gripper jaw extends perpendicularly from the contact point. For $\phi \in [-\frac{\pi}{2}, 0]$, the contact point is the right edge of the gripper jaw, otherwise it is the left edge. The *approach line* is the line through $\hat{p}$ along ϕ. We denote the space of all contact points as $\mathbb{C}$. Examples of contact configurations can be seen in Fig. 2.

We denote sets of similar contact configurations as $T_{q,\psi} \subseteq \mathbb{C}$, where a configuration $c = (p, \phi)$ is in $T_{q,\psi}$ if $\phi = \psi$ and p lies on the line through q perpendicular to the approach line. We denote the similar contact configuration set that contains a given contact configuration c as $T(c)$. We denote the set of all similar contact configuration sets as $\mathbb{T}$. These sets become useful as conservative-slip pushes result in many initially-dissimilar grasps joining similar configuration sets.

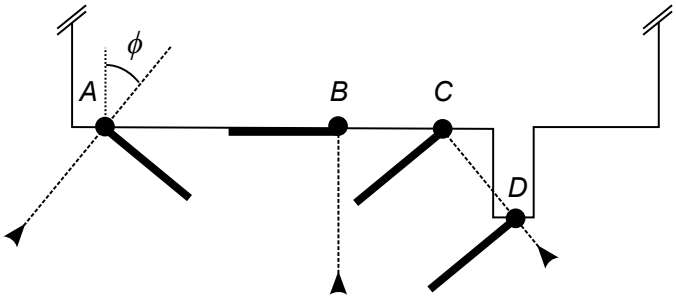

Fig. 2. Contact configurations. The contact points are indicated by circles. By convention, references to left and right are relative to the approach line in the direction from $\hat{p}_i$ into the part, and positive ϕ is clockwise. By definition, if $\phi_j > 0$, the gripper jaw's right edge must be on the approach line, and if $\phi_j < 0$, the gripper jaw's left edge must be on the approach line. If $\phi_j = 0$, we define the approach line to be the gripper jaw's right edge. Configuration A shows an approach angle of $-40°$, which implies a gripper to the right of the approach line. Configuration B shows an approach angle of $0°$, which by convention has a gripper to the left of the approach line. Configurations C and D show an approach angle of $40°$, which implies a gripper to the left of the approach line. Additionally, if configuration C was a nominal contact configuration g, the actual contact configuration g' would be configuration D.

C. Candidate Grasp Configuration Space

The grasp configuration space is defined by a starting position and orientation of the first gripper jaw, and a direction of motion from this position. We assume that orientation of the gripper jaw face is perpendicular to the direction of motion.

We reduce the configuration space from three dimensions to two using *nominal contact configurations* to eliminate some of the redundancies in grasp configurations. A grasp is defined by a contact configuration $g \in \mathbb{C}$ on a nominal part S, as if the gripper jaw moved in along the approach direction from infinity.

As shown by configuration C in Fig. 2, the actual contact configuration $g' \in \mathbb{C}$ for a grasp g may not be the nominal contact configuration. We define a function

$$f_C : \mathbb{C} \times \widehat{\mathbb{S}} \to \mathbb{C}$$

that takes a grasp (in the form of a nominal part and nominal contact configuration) and a perturbation of the nominal part and produces the contact configuration for that grasp on the perturbation.

D. Conservative-Slip Push Grasps with Force Closure

We consider a class of push grasps that enhance part alignment, *conservative-slip push grasps with force closure*. We define this as grasps in which the gripper pushes the part without slipping until it rotates into alignment with the first gripper jaw (a *zero-slip push*), or slips but is guaranteed to enter a zero-slip push (a *conservative-slip push*) and then completes force closure with the second gripper jaw, as seen in Fig. 3. Under this conservative definition, we include slip of the second gripper jaw under limited conditions described in Section IV-A3.

We define the following notation:

$$f_\alpha : \mathbb{S}(S) \times \mathbb{C} \to \mathcal{P}(\mathbb{C})$$
$$f_\beta : \mathbb{S}(S) \times \mathbb{T} \to \mathbb{C} \times \mathbb{C}$$
$$f_\gamma : \mathbb{S}(S) \times \mathbb{C} \to \{0, 1\}$$

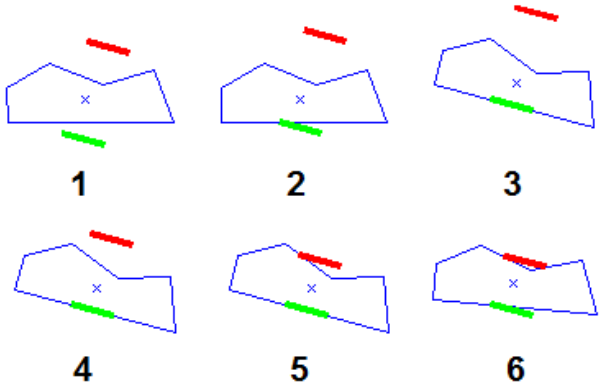

Fig. 3. Snapshots of the execution of a conservative-slip push grasp. The green jaw makes the first contact, and once a stable push is established in frame 3, the red jaw closes. After making contact in frame 5, the part rotates into slip closure in frame 6.

where $\mathcal{P}$ is the power set,
f_α is a function that, for a given part perturbation and contact configuration, determines the set of possible conservative-slip push contact configurations that could result, or the empty set if a conservative-slip push is not possible,
f_β is a function that, given a similar contact configuration set T, returns two disjoint sets T_0 and T_1 such that $T_0 \cup T_1 = T$ and T_0 contains all contact configurations in T that do not achieve force closure; thus T_1 contains all the contact configurations in T that do achieve force closure, and
f_γ is a function determining grasp success; it is the composition of f_α and f_β:

$$f_\gamma(S, c) = \begin{cases} 1 & \text{if } c' \in T_1 \text{ for } (T_0, T_1) = f_\beta(S, T(c')) \\ & \quad \forall c' \in f_\alpha(S, c) \\ 0 & \text{otherwise} \end{cases}$$

E. Quality Measure

We define a quality measure $Q(g, S; \Sigma, \theta)$ as a lower bound on the probability that grasp g on part S will result in force closure based on the tolerance parameter Σ and parameter vector θ.

$$Q(g, S; \Sigma, \theta) = \int_{\mathbb{S}(S)} p(s; \Sigma) f_\gamma(g, s)\, ds \tag{1}$$

The output of the grasp analysis algorithm is

$$\mathcal{Q} = \{Q(g, S; \Sigma, \theta) \,|\, g \in G\} \tag{2}$$

where $G \subseteq \mathbb{C}$ is the set of candidate grasps for part S.

The best grasp and Q-value are:

$$g^* = \arg\max_{g \in G} Q(g, S; \Sigma, \theta) \tag{3}$$
$$Q^*(S, \theta) = Q(g^*, S; \Sigma, \theta) \tag{4}$$

The adaptive version of our algorithm may reduce the value of Q^* relative to the non-adaptive version. The value of Q^* as found by the non-adaptive algorithm is denoted Q^*_{max}, and the normalized value of Q^* for the adaptive version is $\widehat{Q}^* = Q^*/Q^*_{max}$.

IV. Algorithm

The optimization in Eq. 3 is difficult to solve. The problem is nonconvex over G, and f_γ is discontinuous with no simple closed form available. This means the integrand of Eq. 1 cannot be solved for directly. Our approach is to use Monte Carlo integration for Eq. 1 and to use a discrete set of grasps for G.

Our grasp analysis algorithm, shown in Algorithm 1, calculates the quality metric for a set of grasps and part perturbations, evaluating Eq. 1 over multiple grasps simultaneously. For each part perturbation, the candidate grasps are evaluated to estimate if they result in conservative-slip pushes (see Section IV-A1). The successful conservative-slip pushes are grouped into sets of similar configurations (see Section IV-A2), and conservative conditions for force closure are evaluated. Finally, the overall probability of achieving force closure for each candidate grasp is estimated.

Algorithm 1: Grasp Analysis Algorithm. Highlighted line numbers indicate parallelizable steps.

Input: candidate grasp set G_1, part perturbations $S_1, S_2, \ldots, S_M \in \mathbb{S}(S_0)$;
1 **for** *Part perturbation set* $\mathcal{S}_m = \mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_M$ **do**
2 **for** *Part* $S_k = S_1, S_2, \ldots, S_l \in \mathcal{S}_m$ **do**
3 **for** *Candidate grasp* $g_{ij} \in G_m$ **do**
4 Determine actual contact configuration $g' = f_C(g_{ij}, S_k)$;
5 Estimate if g_{ij} results in conservative-slip push of S_k, finding push configurations $\mathcal{C}_{ij} = f_\alpha(S_k, g')$;
  end
6 For all push configurations $\mathcal{C}$, collect similar push configurations $\mathcal{T}$;
7 **for** *Similar configuration set* $T_{q,\psi} \in \mathcal{T}$ **do**
8 Estimate regions of force closure success on S_k, finding $(T_{q,\psi,0}, T_{q,\psi,1}) = f_\beta(S_k, T_{q,\psi})$;
9 **for** *Contact configuration* $c_{ij,S_k} \in T_{q,\psi}$ **do**
10 Predict force closure success $s_{ijk} \in \{0, 1\}$ of g_{ij} for S_k as $c_{ij,S_k} \in T_{q,\psi,1}$;
  end
  end
  end
11 **for** *Candidate grasp* $g_{ij} \in G_m$ **do**
12 Compute intermediate grasp quality $Q_m(g_{ij}, S_0; \Sigma, \theta)$;
  end
12 Produce grasp set G_{m+1} by removing low-quality grasps from G_m according to parameter R;
  end
13 **for** *Candidate grasp* $g_{ij} \in G_1$ **do**
14 Compute grasp quality $Q(g_{ij}, S_0; \Sigma, \theta)$;
  end

Our grasp planning algorithm, shown in Algorithm 2, uses the analysis algorithm on a part using a Monte Carlo method: it generates a set of candidate grasps, and creates part perturbations drawn from the distribution. These grasps and perturbations are passed to the analysis algorithm.

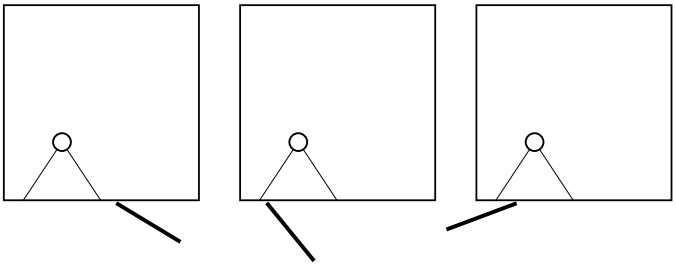

Fig. 4. Push failures. In the left example, the gripper contacts outside the friction cone, pushing away from the center of mass. In the center example, the contact is inside the friction cone, but the direction of pushing is to the wrong side of the center of mass. In the right example, the contact is inside the friction cone, but the push will rotate the object away from alignment.

The analysis algorithm uses a single parameter, the grasp elimination criterion R, which is used in adaptively reducing the candidate grasp set. The planning algorithm also uses several additional parameters, denoted as the vector $\theta = [d_C, \rho, \Phi, \mathcal{M}, R]$. The part tolerances for the vertices and center of mass described in Section III-A are also parameters. Three parameters are used for generation of candidate grasps. A filtering parameter d_C and a configuration density parameter ρ are used to determine the set of candidate grasp positions, and the set of candidate grasp orientations is a third parameter, denoted Φ. The algorithm iteratively tests part perturbations; the number of iterations and part perturbations in each iteration is set by the parameter $\mathcal{M}$, where $M = |\mathcal{M}|$ is the number of iterations, and $\mathcal{M}_i$ is the number of perturbations tested in iteration i. The total number of part perturbations is $N = \sum_i \mathcal{M}_i$. The final parameter is the grasp elimination criterion for the grasp analysis. We describe these parameters and each step of our algorithms below.

A. Evaluating Part Perturbations

For each part perturbation in a part perturbation set, the candidate grasps are evaluated to estimate whether they achieve conservative-slip push grasps with force closure.

1) Conservative-Slip Push Conditions: The algorithm uses geometric properties of the part to determine all candidate grasps resulting in conservative-slip pushes aligned with a part edge for a given gripper width.

The conditions for success of a zero-slip push are as follows: the part purely rotates about the contact point without slipping, the part rotates towards stability with the gripper jaw (that is, the edge rotates toward alignment with the gripper), and once the gripper has two points of contact, the center of mass must be between these points. This means that either the gripper jaw can align with the initially-contacted edge or that the gripper jaw contacts another edge or a convex vertex. Examples of push failures can be seen in Fig. 4.

For a conservative-slip push, the gripper must be guaranteed to align with the initially-contacted edge. Unlike a zero-slip push, the exact motion of the contact point is not known, so any possible contact with another edge or convex vertex cannot be guaranteed to occur in any particular configuration.

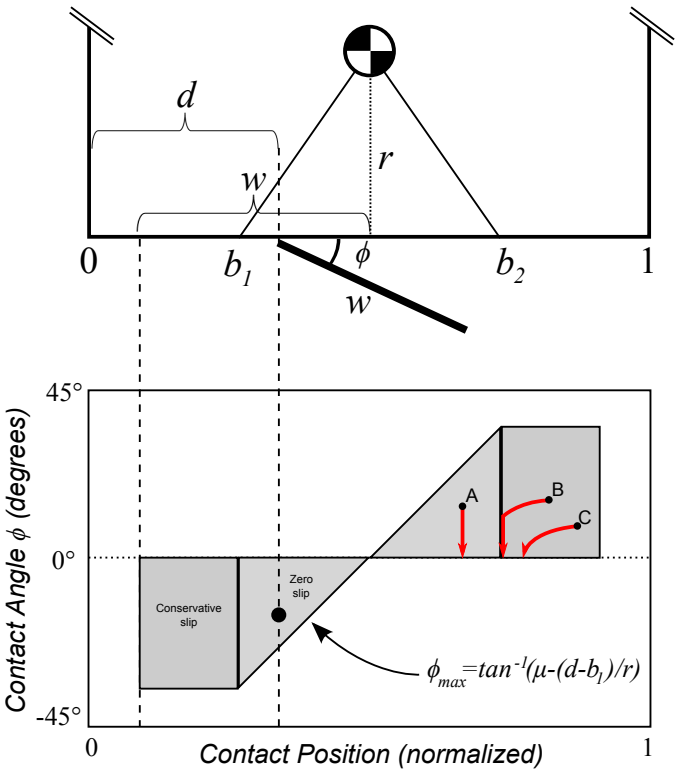

Fig. 5. Configuration space for fast analysis. The upper half of the figure shows a gripper of width w contacting the part at position d with (negative) contact angle $\phi = 30°$, inverse friction cone bounds b_1 and b_2 and perpendicular distance r from the center of mass. Contact with this edge of the part results in the configuration space shown below it; the shaded area is the region where a conservative-slip push occurs. The red lines in the lower region show the configuration-space path for a zero-slip push (A) and possible paths for two conservative-slip pushes (B and C) from initial contact at the points shown. Conservative-slip paths are not predicted specifically, but cannot increase in contact angle or move away from the center of mass. If the path intersects the zero-slip region, it follows a zero-slip path, shown by path B.

As shown by Mason [32], the motion of a part pushed at a given contact point is determined by the friction cone and the direction of pushing. The resulting constraint on candidate grasps is shown in Fig. 5. In the conservative-slip regions, the motion of the gripper is guaranteed to be towards a 0 angle and the center point along the edge. Therefore, the configuration of the gripper as it slips must stay in the region or enter the zero-slip region, in which case a zero-slip push occurs. If the gripper becomes aligned without entering the zero-slip region, the gripper is guaranteed to cover the center of mass, so a successful push occurs. Because the slip analysis does not predict the exact aligned position of the gripper, the force closure tests for a slip push must succeed over all possible aligned positions of the gripper.

2) Collecting Similar Conservative-Slip Push Configurations: Before evaluating force closure on the candidate grasps that result in conservative-slip pushes, the conservative-slip push configurations for those candidate grasps are collected into sets of similar configurations. A similar configuration set often contains all the conservative-slip pushes for some edge of the part. Because our estimation of force closure for all positions on an edge can be determined analytically, the estimated closure success of all elements of a similar configuration set can be evaluated simultaneously, as shown below.

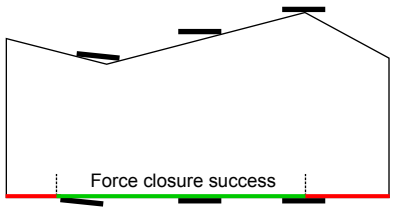

Fig. 6. Force closure modes. Slip closure is shown by the gripper pair on the right; friction closure is shown by the gripper pair in the middle; and convex vertex closure is shown on the right.

3) Conditions for Force Closure: Force closure on a part is achieved when the line between the contact points on each side lies inside the friction cones of both contact points [34]. If there are multiple contact points on a side, there need be only one successful contact point for force closure.

In our algorithm, force closure is considered to be achieved under three conditions, shown in Fig. 6. First, if the second gripper jaw contacts an edge and the contact direction is within the friction cone, the gripper completes force closure. Second, if the second gripper jaw contacts a convex vertex, and this convex vertex is opposite a section of the first gripper jaw that contacts the part, force closure is successful. The third condition involves slip of the second gripper jaw. If the second gripper jaw can slip along the edge it contacts and come into contact with an adjacent edge, and this configuration produces valid force closure, the gripper is considered successful. While this condition is restrictive, it can be determined for ranges of gripper contact points, whereas more general slip conditions require each grasp to be tested individually. This allows our conservative-slip push test, which returns a range of possible aligned positions, to have force closure estimated efficiently for the entire range.

B. Lower Bound on Probability of Achieving Force Closure

Once the candidate grasp conditions have been evaluated for all part perturbations, the lower bound on the probability of achieving force closure for that candidate grasp is estimated using a weighted percentage, where the estimated success or failure on a part perturbation is weighted by the probability of that situation occurring.

C. Adaptive Candidate Grasp Removal

Adaptive grasp candidate removal was added to the algorithm after the observation that the best grasps were already part of the top candidate grasps after only a few part perturbations had been tested, although their final Q-values were not predictable from their Q-values earlier in the analysis. Therefore, the adaptive procedure was developed to remove unpromising grasps, while still testing the promising grasps to refine their Q-values.

After all the part perturbations in a part perturbation set are tested, candidate grasps with low Q-values are removed from further testing. The criterion for removing a grasp is the

parameter R. The number of grasps eliminated at step m is $R\,|G_m|$, giving the total grasp evaluations as

$$\eta = \sum_{m=1}^{M} |G_m| \mathcal{M}_m \tag{5}$$

where $|G_m| = R\,|G_{m-1}|$ for $m = 2, \ldots, M$.

The algorithm checks the minimum Q-value of the top $(1-R)\,|G_m|$ candidate grasps, Q_{min}. If the set of candidate grasps $\{g \,|\, Q_g \geq Q_{min}\}$ is bigger than $(1-R)\,|G_m|$, ties between the lowest-Q grasps are broken randomly. The elimination criterion balances maximizing grasp elimination for faster execution with preventing the elimination of grasps that may eventually prove to have high Q-values. Other elimination criteria are possible; ties could be included rather than broken randomly, or all grippers above a certain fraction of the current best Q-value could be retained. However, these criteria do not guarantee a fixed number of grasp evaluations. We denote the number of grasp evaluations in the adaptive algorithm normalized to the number of evaluations in the non-adaptive algorithm as $\hat{\eta} = \frac{\eta}{N|G_1|}$.

Algorithm 2: Grasp Planning Algorithm. Highlighted line numbers indicate parallelizable steps.

1 Filter S_0 into S_C;
2 Determine nominal contact points $\hat{P}$ on S_0 using S_C;
3 Create candidate grasp set G_1 from $\hat{P}$ and Φ;
4 Create part perturbations $S_1, S_2, \ldots, S_N$ of S_0;
5 Compute quality of candidate grasps $\mathcal{Q}$ using Algorithm 1;

D. Grasp Planning

The grasp planning algorithm shown in Algorithm 2 uses two additional steps to generate candidate grasps and part perturbations, which are then analyzed using our grasp analysis algorithm.

1) Generating Candidate Grasps: The grasp planning algorithm generates an initial candidate grasp set

$$G_1 = \{g_{ij} = (\hat{p}_i, \phi_j) \,|\, \hat{p}_i \in \hat{P}, \phi_j \in \Phi\} \tag{6}$$

While each $(\hat{p}, \phi)$ pair could be independently generated, we use a fixed set of ϕ values as a parameter, and apply them to a generated set of $\hat{p}$ values, using the method in [27], which takes as parameters a configuration density ρ, a set of approach angles Φ, and a filtering parameter d_C.

We use a scale-invariant parameter to determine the number of $\hat{p}$ values (i.e., $|\hat{P}|$) for the part, *sample density*, denoted ρ. For each edge, a set of $\hat{p}$ values is generated, linearly spaced with the number of points equal to $\rho \times \frac{\text{length of edge}}{\text{mean edge length}}$. To reduce the effect of complexity on ρ, this is computed on a filtered shape S_C.

The filtered shape S_C is generated using an extension of the Ramer–Douglas–Peucker (RDP) algorithm [17] [37]. The RDP algorithm smooths a polyline using a distance parameter (here, d_C) that defines the maximum distance a removed vertex can be from the resulting new edge. In our extension to polygons, every pair of adjacent vertices are tested by removing the edge between the vertices, smoothing the resulting polyline, and forming a new polygon with fewer edges by reconnecting the two vertices. The filtered polygon with the fewest edges is selected.

2) Sampling Part Perturbations: Before testing the candidate grasps, part perturbations are created by sampling from the distributions of each vertex and the center of mass. The number N of part perturbations is determined by a parameter to the algorithm, $\mathcal{M}$. The part perturbations are collected into part perturbation sets $\mathcal{S}_1, \ldots, \mathcal{S}_M$, where $|\mathcal{S}_i| = \mathcal{M}_i$. In Section V-D we determine that using 100 part perturbations provides reliable results. We explore values of $\mathcal{M}$ in Section V-F.

V. Experiments

To test the algorithm in simulation, a set of images of brackets were found on Google Image Search, and manually contoured by tracing a polygon over the image. The shapes produced by this method are shown as Parts A through I in Fig. 7, along with three simpler, manually-created parts. A comparison with the approach of ignoring uncertainty is presented in Section V-B. We evaluated a large number of parameter combinations, which is detailed in Section V-C. In Section VI, we report results from testing a Cloud-based implementation of the algorithm.

Except for where noted, tests used vertex variance of 0.2 times the maximum shape radius (measured from the centroid to the vertices), a center of mass variance of 0.7 times the maximum shape radius, a gripper width 25% of the maximum shape diameter (measured between vertices), and a coefficient of friction of 0.7. The tests were run on an four core 3.40 GHz machine with 16 GB of RAM, using MATLAB R2013a, and on PiCloud, a cloud computing provider.

A. Analysis of Parts

For one parameter combination, the full results for two parts are shown in Figures 8 and 9, and best grasp for each of the shapes are shown in Fig. 7. The parameters for these figures were $d_C = 0$, $\rho = 1.5$, and $|\Phi| = 5$.

We observed that the algorithm did not choose edges close to the center of mass when only zero-slip pushes were allowed. While this result can seem counterintuitive, grasps close to the center of mass are less robust under our assumptions because an edge close to the center of mass has a smaller region in which zero-slip pushes can be achieved. A perturbation in the center of mass will move this region, invalidating a large number of zero-slip pushes originally in the region. This effect was observed on Part D. The maximum Q-value for a zero-slip push on the two horizontal edges of Part D is 43.6, but with slip, the maximum is 94.2. The maximum Q-value for a zero-slip push on the vertical left edge is 90.3. The best grasp on Part J is on an edge close to the center of mass because the edges on either end of the part are too angled to each other for reliable force closure.

Part B, shown in Fig. 9, demonstrates the effect of requiring a conservative-slip push. Grasps on the edges marked α and

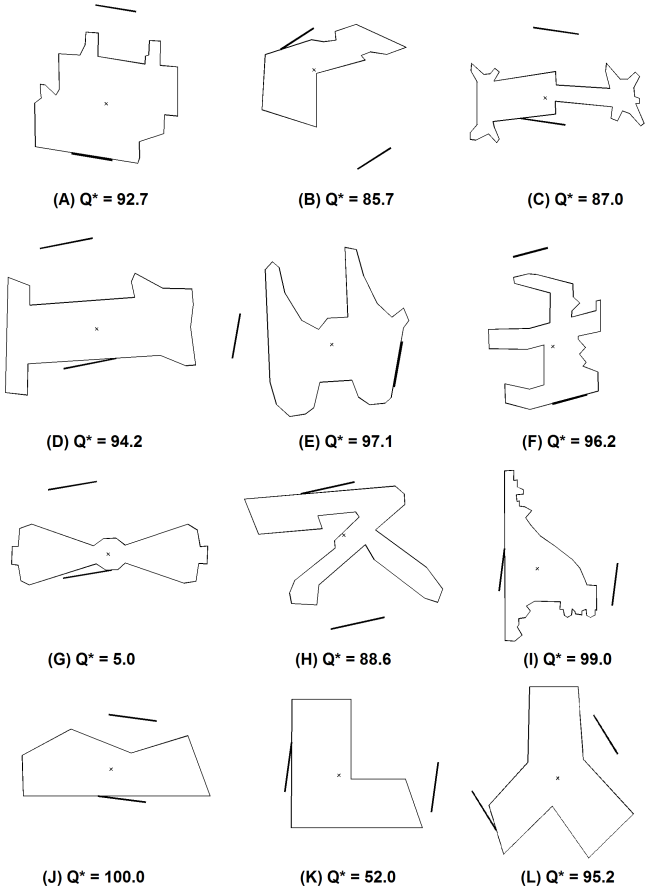

Fig. 7. The test set of brackets. The g^* grasps for parameters $d_C = 0$, $\rho = 1.5$, and $|\Phi| = 5$ are depicted. The grasps are indicated with the pushing jaw in contact with the part, and the closing jaw opposite it away from the part. "Whisker diagrams" showing detailed results for Parts A and B can be seen in Figures 8 and 9, respectively. Part G has a problematic shape for the algorithm. The size of the gripper prevents it from contacting the edges very near the center of mass. The long, straight edges are outside the inverse friction cone of the center of mass, meaning an contact on them will slip. The ends of the part are narrow and consist of several different edges, which, under perturbation, can prevent conservative-slip pushes or force closure from being achieved. Part K is also problematic. As shown in Table I, a 100% successful grasp is possible, but is only found with a very dense grasp set. With the grasp set used here, the best grasp only has a Q-value of 52.0. This is because the shape of part means grippers that are in good position relative to the center of mass are in areas that are very sensitive for force closure.

β only have very low Q values, because most of each edge is outside the inverse friction cone from the center of mass, meaning any contact will result in slip. The large angle between edges γ and α causes successful pushes on edge γ to fail to achieve our conservative force closure conditions. However, force closure can be achieved against the vertex labeled τ, and this is reflected in the high Q value of some grasps on edge γ.

Parts F and H show how the differences in the shape can have a large effect on the quality of grasps, given equal uncertainty. Part F has a very high quality grasp that contacts a flat edge and closes against a small edge with a convex corner. The best grasp on Part H has the same properties, but a much lower Q value. The difference between the parts is that the first edge contacted by the gripper is further from the center of mass on Part F, which as mentioned above can be problematic, and that the uncertainty in the opposite edges on Part H can cause the gripper to contact edges that are more angled.

Part G has a problematic shape for the algorithm. The size of the gripper prevents it from contacting the edges very near the center of mass. The long, straight edges are outside the inverse friction cone of the center of mass, meaning an contact on them will slip. The ends of the part are narrow and consist of several different edges, which, under perturbation, can prevent conservative-slip pushes or force closure from being achieved.

Part K is also problematic. As shown in Table I, a 100% successful grasp is possible, but is only found with a very dense grasp set. With the grasp set used for Fig. 7, the best grasp only has a Q-value of 52.0. This is because the shape of part means grippers that are in good position relative to the center of mass are in areas that are very sensitive for force closure.

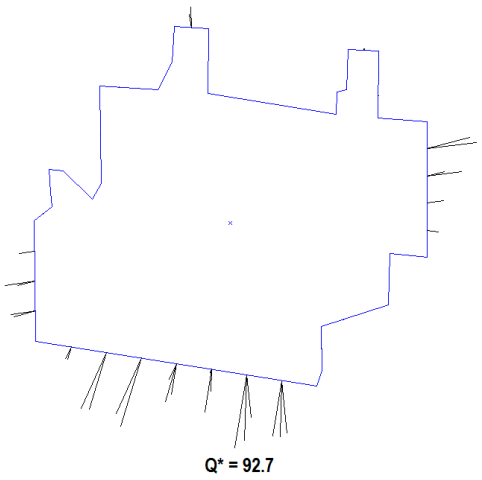

Fig. 8. "Whisker diagram" showing algorithm results for Part A, using $d_C = 0$, $\rho = 1.5$, and $|\Phi| = 5$. Each line segment represents a candidate grasp, and indicates its nominal contact point on the part. The line segment indicates the approach lines for the grasp, and is orthogonal to the gripper jaw. The length indicating the Q-value relative to other segments. The approach line with the highest Q-value (i.e., the longest line segment) is labeled, and the jaw positions for this grasp is illustrated in Fig. 7.

B. Comparison with Ignoring Shape Uncertainty

We compared our results to a first-order grasp planner ignoring shape uncertainty, which ran the algorithm simply on the nominal part, without considering perturbations. Generally, many candidate grasps are predicted to achieve force closure on the nominal part. However, when subject to uncertainty, many of these grasps become considerably less desirable. For comparison, we ran 84 tests using various parameter values (described in Section V-C), and for each run, the candidate grasps predicted to achieve force closure on the nominal part were tracked and their final quality compared. On average, only 4% of these candidate grasps were also the best grasps after 100 iterations of the algorithm. After 100 iterations, the average Q-value of these candidate grasps was only 58% of the value of Q^*.

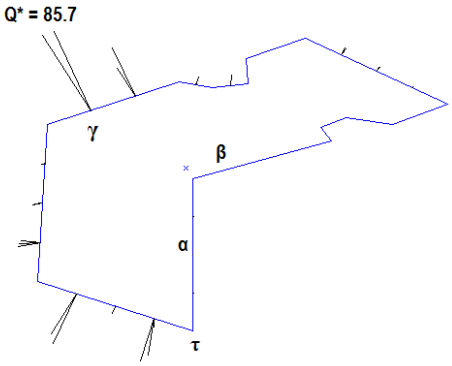

Fig. 9. "Whisker diagram" showing algorithm results for Part B, using $d_C = 0$, $\rho = 1.5$, and $|\Phi| = 5$. The labels are used in Section V-A to illustrate various aspects of the results. Each line segment represents a candidate grasp, and indicates its nominal contact point on the part. The line segment indicates the approach lines for the grasp, and is orthogonal to the gripper jaw. The length indicating the Q-value relative to other segments. The approach line with the highest Q-value (i.e., the longest line segment) is labeled, and the jaw positions for this grasp is illustrated in Fig. 7.

C. Sensitivity Analysis

We performed a sensitivity analysis on the parameters for the candidate grasp generation step in the algorithm, which are the maximum distance for the filtering step d_C, sample density ρ, and the approach angle set Φ.

The number of nominal contact points is critical to maximizing the value of Q^* grasps. For a given edge in contact with the first gripper jaw, force closure depends on the opposite edges, which define regions where closure is or is not achieved. With increasing part complexity, the regions become smaller and more numerous, and edges must be covered more densely with contact points to ensure that the regions in which force closure is achieved are found.

To evaluate combinations of values for the parameters, the parameter space was gridded and tested. For filtering, the scale-invariant measure used was fraction of maximum part radius. Increasing values were used until it was judged that large features of the test parts were being filtered out. For the approach angles, a wide, dense range of approach angles were tested, 15 linearly spaced directions from $-45°$ to $45°$, inclusive, along with a high value of points per mean edge and no filtering. For all parts tested, the maximum magnitude was never above $13°$. We subsequently chose $\pm 15°$ as the range bounds. For sample density (ρ), the value was increased until no further gain in Q^* was seen, and this was used as an upper bound.

The parameter grid included four filtering distances, three sets of approach angles, and seven values for points per mean edge. The filtering distances were 0 (i.e., no filtering other than combining collinear edges), 0.03, 0.06, and 0.09. For approach angles, three sets of linearly spaced points between $-15°$ and $15°$, inclusive, were used, with 5, 9, and 15 points, respectively. Only odd values were chosen such that $0°$ would be included. For sample density, the following seven values were used: 1.5, 3, 5, 7.5, 10, 15, and 20.

The results from the gridded parameter space illustrated the trade-off between Q^* and runtime; selected results can be seen in Table I. Additionally, the discontinuous nature of force closure on polygonal parts was apparent: holding other parameters constant, increasing the sample density sometimes decreased Q^*, when a small region of an edge had the highest probability, and was alternately hit or missed by the spacing of the nominal contact points.

Part	Q^*	runtime (s)	d_C	ρ	$\|\Phi\|$
A	86.0	47.6	0.03	1.5	5
A	91.3	102.9	0.03	5	5
A	92.3	569.4	0	20	9
A	92.7	48.8	0	1.5	5
B	65.9	19.4	0.09	1.5	5
B	80.6	32.7	0.03	1.5	5
B	80.6	38.3	0.09	3	5
B	85.7	34.8	0	1.5	5
C	76.6	40.4	0.06	1.5	5
C	85.0	31.5	0.09	1.5	5
C	89.1	110.8	0.09	7.5	5
C	90.0	156.5	0	5	5
D	93.2	27.3	0.09	1.5	5
D	97.3	40.9	0.09	3	5
D	98.2	75.8	0.03	7.5	5
D	98.3	219.6	0.06	15	9
E	88.4	28.1	0.09	1.5	5
E	95.6	42.4	0.06	1.5	5
E	98.1	47.4	0.03	1.5	5
E	100.0	71.4	0	3	5
F	94.9	83.0	0	3	5
F	97.3	37.2	0.09	1.5	5
F	98.2	55.2	0.03	1.5	5
F	99.0	281.5	0.03	7.5	9
G	5.2	15.5	0.09	1.5	5
G	7.8	16.8	0.06	1.5	5
G	9.5	32.5	0.09	3	5
G	12.2	61.7	0	3	5
G	12.4	98.8	0.06	5	9
G	15.2	102.2	0.09	7.5	9
G	17.1	184.6	0.06	15	9
H	77.8	30.3	0.09	1.5	5
H	93.2	78.5	0.03	5	5
H	94.0	236.7	0.06	15	9
I	81.1	35.9	0.09	1.5	5
I	98.0	57.5	0.06	1.5	5
I	99.0	118.4	0	1.5	5
I	99.0	1550.4	0	20	9
J	100.0	14.7	0	1.5	5
J	100.0	125.3	0.03	20	9
K	52.0	12.6	0	1.5	5
K	72.2	12.1	0.09	1.5	5
K	83.6	21.9	0	3	5
K	99.0	37.0	0.09	7.5	5
K	99.5	99.9	0.06	15	9
K	100.0	123.4	0	20	9
L	90.6	16.6	0.09	1.5	5
L	90.9	35.5	0	5	5
L	91.0	146.2	0	20	9
L	95.2	17.0	0	1.5	5

TABLE I

SELECTED RESULTS FROM THE SENSITIVITY ANALYSIS FOR PARTS IN FIG. 7, SHOWING PART NAME, VALUE OF Q^*, RUNTIME USING MATLAB R2013A ON AN FOUR CORE 3.40 GHZ COMPUTER WITH 16 GB OF RAM, FILTERING PARAMETER d_C, SAMPLE DENSITY ρ, AND NUMBER OF APPROACH ANGLES $|\Phi|$, USING $\mu = 0.7$.

D. Number of Part Perturbations

Reducing the set of part perturbations reduces the runtime of the algorithm, but runs the risk of individual samples having a large effect on the result. We investigated the effect of this trade-off by generating 500 part perturbations and running the

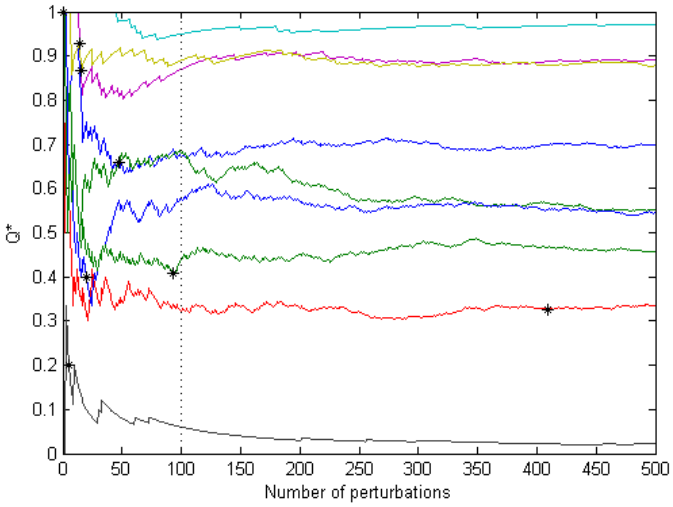

Fig. 10. Q^* vs. number of part perturbations evaluated for parts A-I in the test set. The point at which g^* stops changing is marked with an asterisk.

algorithm on each sequential subset of 1 to 500 perturbations. The value of Q^* over this range can be seen in Fig. 10. In the first few iterations, there are some candidate grasps that are predicted to achieve force closure for all part perturbations tested so far, so the maximum probability is at 1. By the point where 100 part perturbations had been processed, the maximum probability was always within 5% of its value at 500 perturbations. Additionally, g^* stopped changing before 100 perturbations for all but one part (that is, the best grasp was identified early). This suggests a convergence heuristic: once the Q^* stops changing by more than 5% after testing a new part perturbation, perhaps measured over a moving window, the best grasp has likely been found and the algorithm can terminate.

E. Tolerancing

To test the effectiveness of our algorithm for estimating part tolerance bounds, we developed a procedure to find tolerance limits that allow a grasp to stay above a given Q threshold. Because the variance of different aspects of the part may affect a grasp to a greater or lesser degree, the variances for the parts were split into two groups: the variance of the vertices for the initial contact edge, and the variance of the remaining vertices. The vertices for the initial contact edge along with the center of mass determine the success of the stable push, while other vertices determine the success of closure.

To test this tolerance bounding procedure and the effect of variance on closure, three simple parts were created with different features. These parts are shown in Fig. 11. Part A, a simple rectangle, tested closure on a flat edge. Part B introduced a single convex vertex instead of a flat edge, to test closure against a vertex. Part C used a set of three vertices to test the effect of complex edges on closure. A fourth part, Part D, was created to test the effect of variance on the initial push. It is a thin rectangle, with the edge to be tested close to the center of mass, creating a smaller valid region more sensitive to higher tolerances. The best grasp on the highlighted edge

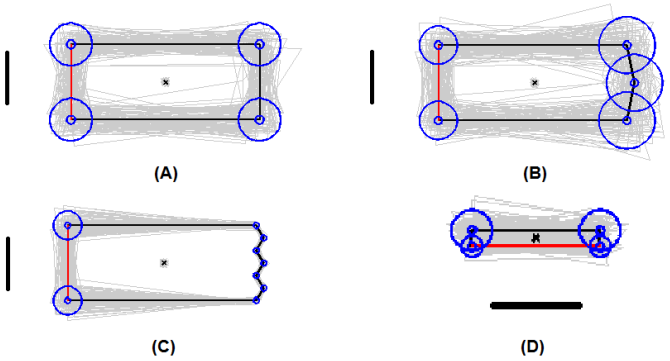

Fig. 11. Tolerancing results for selected parts. The best grasps on the highlighted edge were found with small tolerances shown as the smaller circles around the vertices with radius two standard deviations (95% confidence interval). The gripper width used for all parts is shown next to the part. Tests were performed as described in Section V-E using $d_C = 0$, $\rho = 4$, and $|\Phi| = 5$, and for the indicated tests from that section, the tolerance for each vertex and center of mass is shown along with 100 perturbations of each part. Parts A and B are shown with tolerances that give comparable $\widehat{Q}^*$ (64.5 and 66.9, respectively), and suggest that friction closure is more sensitive to increased tolerances. Part D suggests that, relative to Part A, narrow parts have greater sensitivity to near-edge tolerances.

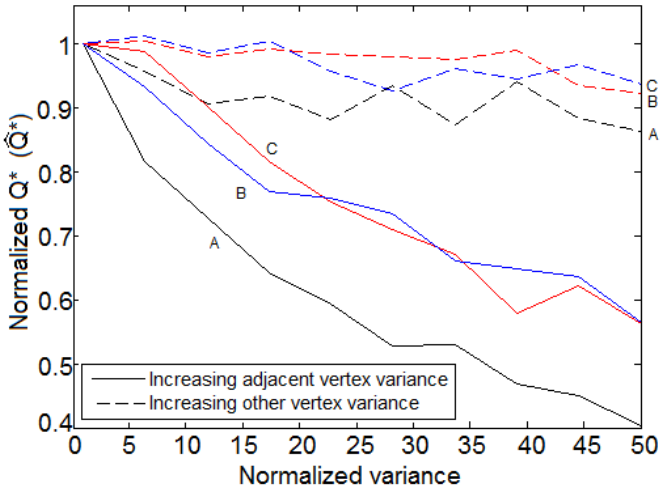

Fig. 12. Effect of increasing tolerances on quality. Tolerance is shown as vertex variance normalized to the initial variance. Each set of three lines show the results for Parts A, B, and C. The solid lines show the average $\widehat{Q}^*$ for increasing near-edge vertex variance, keeping other variance constant. The dashed lines show the average $\widehat{Q}^*$ for increasing values of the non-near-edge vertex variance, keeping the near-edge variance constant.

shown in Fig. 11 was found, and this grasp was tested under increasing variance for the near-edge vertices and the other vertices. The center of mass was fixed to the centroid of each perturbation.

The results for Parts A, B, and C suggest that Q-values are significantly more sensitive to near-edge variance. As shown in Fig. 12, as the variance of near-edge vertices increases while the remaining variances are kept constant, the value of Q^* reduces significantly. Keeping the near-edge variance constant while increasing the others had a smaller effect on Q^*, staying within 14% of its initial value.

While the response of these parts was similar when considering the relative change of Q^*, the absolute value showed

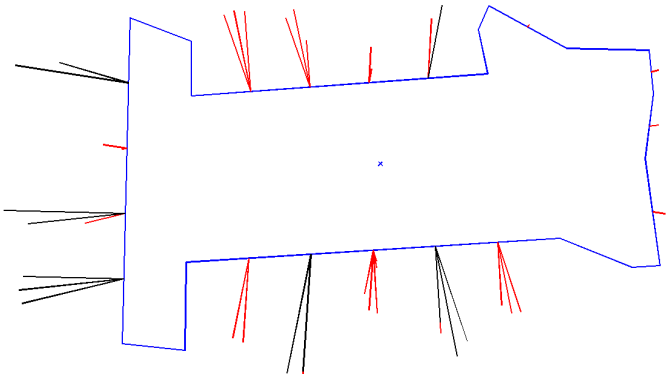

Fig. 13. Candidate grasps eliminated by the adaptive candidate grasp removal. Eliminated grasps are marked in red. The parameters for this test were $d_C = 0$, $\rho = 1.5$, and $|\Phi| = 5$, $R = 0.9$, and $M_1 = 19$.

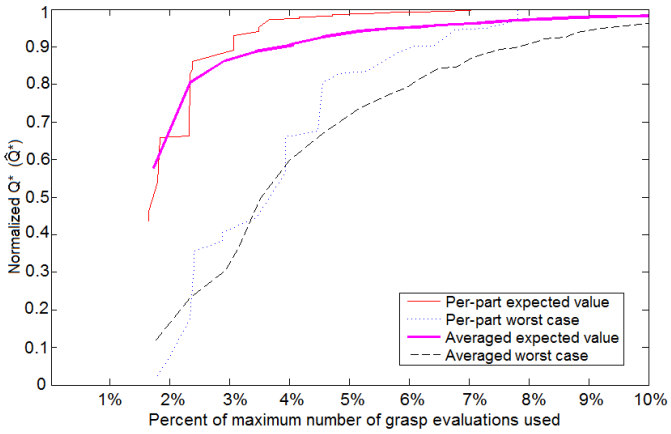

Fig. 14. Tradeoff between execution time and grasp quality, showing $\widehat{Q}^*$ vs. percent of grasp candidate evaluations performed ($100 \times \hat{\eta}$) for multiple test parts and adaptive parameters. The graph is truncated at 10% on the x axis because all per-part expected and worst-case values after this have a $\widehat{Q}^*$ of 1. A value of 1 on the y axis indicates the overall best gripper was still found by the adaptive algorithm. The Pareto curve of average expected $\widehat{Q}^*$ over all parts tested is shown as a solid magenta line, and the Pareto curve of worst case is shown as a dashed black line. The red and blue lines show the lower bound of the Pareto curves for per-part expected and worst-case values, respectively.

differences between the parts. The minimum Q^* for Part A was 28.2, for Part B, 46.3, and for Part C, 43.9. Part A had lower Q^*-values because it used only friction closure. Large movements of the vertices can cause the angle between the near edge and the far edge to exceed frictional limits. Closure against a convex vertex is more robust to variance, since such closure does not depend on an angle with the gripper, and if it becomes concave, slip closure may allow force closure.

Part D retained $Q^* = 100$ for tests with high tolerances in the opposite vertices and center of mass, but low tolerances in the adjacent vertices.

We found that the initial contact edge vertices required lower variances, suggesting that success of the stable push was the component of the grasp most sensitive to higher tolerances. In designing a part, tolerance specifications could be defined using the results of this maximum allowable variance test.

F. Adaptive Sampling

The adaptive removal procedure introduced two new parameters, so we tested these parameters to determine their effect on the algorithm's performance.

The adaptive grasp candidate removal step involves a tradeoff between low execution time and high-quality grasps. In particular, if a fixed number of grasp evaluations are allowed, then the larger the initial part perturbation set, the more aggressive the grasp candidate removal step must be. To explore this tradeoff, we tested the adaptive grasp candidate removal step by varying the parameters for both initial part perturbation set size and the grasp elimination criterion. We used a single grasp reduction step (that is, $|\mathcal{M}| = 2$) to do initial testing; our tests using more steps are described at the end of this section.

First, we ran the non-adaptive algorithm (i.e., $|\mathcal{M}| = 1$) on the dataset of parts from [27]; each of the twelve parts was tested using twenty separately generated perturbation sets (giving a total of 120 part/perturbation set combinations), using $d_C = 0.06$, $\rho = 6$, and $|\Phi| = 5$, and $N = 70$. The value of Q^* for each test was thus the maximum Q^* that could be found by the adaptive algorithm (i.e., it was Q^*_{max}). Then, for each initial perturbation set size $M_1 = 1, \ldots, 70$ all possible distinct values of the adaptive elimination threshold were found. For each test, the unique Q-values at the M_1-th iteration were found, and the values of the elimination threshold that would select those Q-values were found. Then, for each initial perturbation set size, all of the distinct values of the adaptive elimination threshold R from all of the tests were combined into a set, and for each threshold value (which was determined from a single part/perturbation set), the outcome of the adaptive algorithm on all of the 120 part/perturbation sets using that threshold value was analyzed.

To analyze the outcome of the adaptive algorithm on a part/perturbation set, we used data from the already-run non-adaptive test. At the given M_1-th iteration, the grasp reduction step was simulated from the Q-values calculated previously. However, because the grasp elimination criterion randomly breaks ties, it couldn't be used directly. Instead, the worst-case and expected values were found. The worst value was found by retaining the tied candidate grasps with the lowest final Q-values. The expected value was calculated as the sum over all combinations of the maximum final Q-value in that combination weighted by the likelihood of occurrence of the combination.

The result of this analysis is shown in Fig. 14. The adaptive sampling was able to aggressively reduce the candidate grasp set without reducing $\widehat{Q}^*$. Considering the best parameter values for each part individually, the results suggest a very low number of perturbations must be tested to find high quality grasps. Above $\hat{\eta} = 0.031$ (that is, 3.1% of the possible grasp evaluations are performed), the expected value of Q^* was within 10% of the maximum, and the worst case values reached the maximum by $\hat{\eta} = 0.08$. Averaging $\widehat{Q}^*$ across all parts for each parameter combination, the performance reduces slightly: the expected value of $\widehat{Q}^*$ does not reach the maximum until $\hat{\eta} = 0.277$, and the worst case did not reach

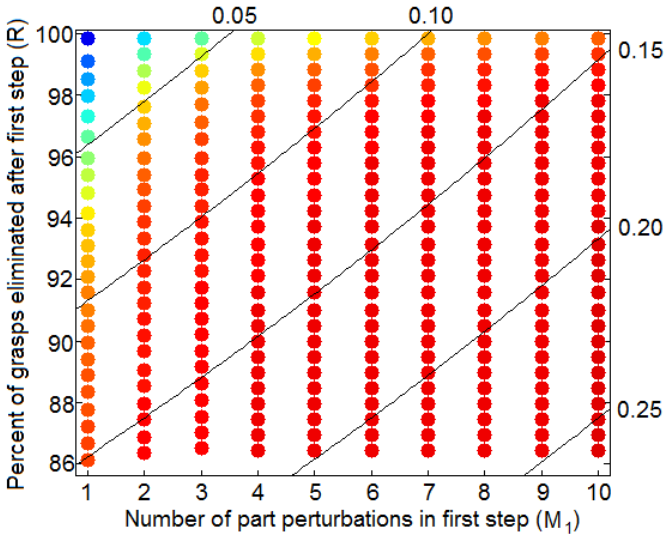

Fig. 15. Tradeoff in worst-case quality (color) and execution time (lines) over parameter combinations. The color of each dot indicates the average worst-case $\widehat{Q}^*$ for the parameter values. For example, the point in the upper left represents $M_1 = 1$ and 99.85% of grasps eliminated (i.e., $R = 0.0015$), meaning after one part perturbation is tested, one grasp is selected from the successful grasps on that perturbation, and tested on the remainder of the perturbations. This point has a $\widehat{Q}^*$ of 0.577. Contours of $\hat{\eta}$ between 0.05 and 0.25 are shown. The parameter values at any point along a contour require the same number of grasp evaluations.

maximum until $\hat{\eta} = 0.285$. However, for $\hat{\eta} \geq 0.08$ (when the per-part worst case $\widehat{Q}^*$ reaches 1), the best expected value of $\widehat{Q}^*$ averaged over all parts was 0.978, and the worst case was 0.926.

This analysis would allow a designer to choose the best adaptive parameters satisfying design constraints, either reducing the number of evaluations given a minimum worst case or expected value, or maximizing worst case or expected value given a maximum number of evaluations.

Fig. 14 does not indicate what parameter values produce the displayed Pareto curves. Fig. 15 shows the average worst-case value of $\widehat{Q}^*$ over all tests for parameter ranges $M_1 \in [1, 10]$ and $R \in [0.85, 1]$. The contours of $\hat{\eta}$ are shown for several values between 0.05 and 0.25. Given a low limit on grasp evaluations, this analysis allows the best parameter combination satisfying the constraint to be found.

Good grasps are identified after testing a small number of part perturbations, as shown in our previous work. This allows the adaptive grasp elimination step to cull unpromising grasp candidates, and use the remaining part perturbations to refine the Q-value of the good grasp candidates. We experimented with using more than one iteration of grasp candidate removal, but the extra reduction in number of grasp candidates was of minimal benefit.

VI. Cloud Computing Experiments

We tested the scalability of our algorithm in the Cloud using PiCloud, a platform that automates high performance computing through Cloud-based computation using Amazon EC2 [2]. PiCloud allows for an executable, along with its environment, to be replicated across any number of nodes in the Cloud and run in parallel.

Our Cloud-based implementation is modeled on MapReduce [15]. It uses a set of nodes, all started in parallel, to each process a portion of the part perturbation set for the non-adaptive algorithm. This corresponds to parallelizing Step 2 of Algorithm 1, as well as parallelizing the actual perturbation sampling itself. The results from each node were collected and combined to produce the algorithm results.

In our tests, the executable was a compiled MATLAB script, with input parameters specifying the part, algorithm parameters, and number of perturbations to test. The executable was placed in an environment with the appropriate part data, and this environment was replicated across all the nodes in the test. PiCloud would then start all the nodes and run the executable on each one. The executable outputs a results file; all the results files were then collated locally to produce the overall output of the algorithm. The PiCloud nodes used "c2" cores, which have 800 MB of memory and 2.5 "compute units" [3] as defined by Amazon for EC2. A compute unit provides "the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor" [1].

A. Measuring Running Time

PiCloud provides the total running time of each node, which includes the time to start up the node itself as well as the time to start the MATLAB script. We also measured the running time of the algorithm within MATLAB (on each node). However, PiCloud does not have any mechanism for including node-identifying information with the results files, we could not match a MATLAB running time to a specific PiCloud node running time. Therefore, we have two sets of timing information, one of just the algorithm and one including the overhead of Cloud-based execution. Because of the inability to match this information on a per-node basis, they can only be compared in aggregate.

The total running time of the algorithm when running in parallel is the longest time taken by any node, which is called *synchronous parallelism*. This is because they are all started at the same time, but the algorithm is only finished once all nodes have returned their data. We explore an alternative to this in Section VI-F.

We measured speedup in three values: the average MATLAB runtime and the overall PiCloud and MATLAB runtimes (that is, the maximum over nodes in each test).

B. Test Runs

Using three configurations, we ran two sets of tests, one to test the run times over all parts, and one to test the variability of run times on a single part.

Configuration 1 was $d_C = 0$, $\rho = 1.5$, and $|\Phi| = 5$. Configuration 2 was $d_C = 0.03$, $\rho = 7.5$, and $|\Phi| = 9$. Configuration 3 was $d_C = 0.09$, $\rho = 20$, and $|\Phi| = 15$. We chose Configurations 1 and 3 as corners of the parameter grid used in Section V-C, and Configuration 2 as midway between them. Thus Configuration 1 has a very sparse candidate grasp set, and Configuration 3 has a very dense candidate grasp set.

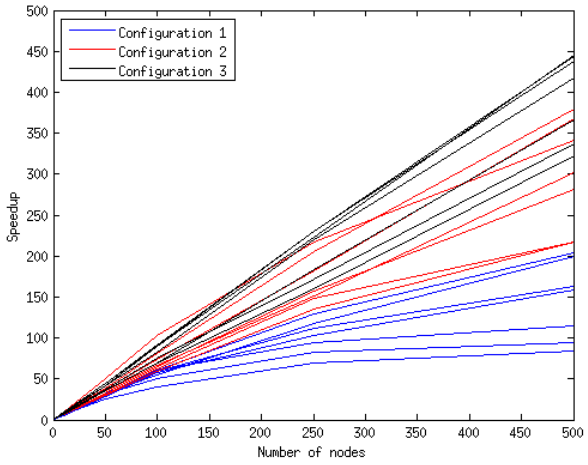

Fig. 16. Average MATLAB runtime speedup vs. number of nodes. Each part is plotted as a separate line. The highest speedup is 445×, for Part G with Configuration 3.

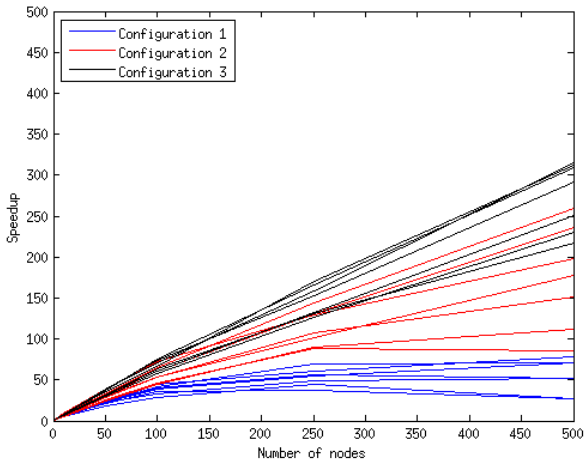

Fig. 17. Overall (i.e., worst case) MATLAB runtime speedup vs. number of nodes. Each part is plotted as a separate line. The highest speedup is 315×, for Part H with Configuration 3. The speedups are less than for the average times because with increasing numbers of nodes, the probability increases of a node taking significantly longer than average, increasing the overall (i.e., worst case) running time.

C. Speedup

The speedup for the average node MATLAB running time for all parts is shown in Fig. 16. The plot shows that the denser the configuration, the closer the speedup is to being completely linear. The best speedups achieved for 500 nodes were 445× for Part G and 444× for Part L, both with Configuration 3. The best speedup for the much sparser Configuration 1, however, were 204× for Part G and 199× for Part H. These are lower because of overhead in the algorithm; the average running time for Configuration 3 was 9.4 times longer than for Configuration 1.

The overall running time is dependent on the maximum node running time, rather than the average, i.e., it is the worst-case running time. The speedups for overall MATLAB running

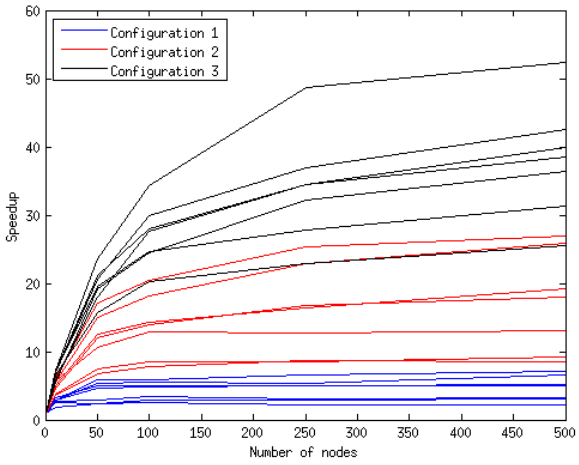

Fig. 18. Overall (i.e., worst case) PiCloud runtime speedup vs. number of nodes. Each part is plotted as a separate line. The highest speedup is 52×, for Part H with Configuration 3. In addition to lower speedups due to the probability of an outlier that increases the overall (i.e., worst case) runtime increasing with increasing numbers of nodes, the overhead of starting a PiCloud node is large relative to the algorithm running time. We estimate this overhead in Section VI-D.

time are shown in Fig. 17. The best speedups, 315× for Part H and 312× for Part G, are much less than for the average running time. By increasing the number of nodes, the average running time goes down, but the probability of one node taking much longer than the average (and thus driving up the overall running time) goes up. We discuss a strategy to reduce this effect in Section VI-F.

The speedups for overall PiCloud running time are shown in Fig. 18. The best speedup, 52×, is an order of magnitude lower than the MATLAB speedups. The reason for this is that the PiCloud node startup time is a large overhead, so even as the MATLAB runtimes reduce, the overall time taken including the PiCloud node overhead does not decrease as much.

D. Overhead estimation

Since we could not match PiCloud node running times to the MATLAB running times, we looked at the aggregate over each test. For each test, we subtracted the average MATLAB runtime from the average node runtime. Then, we took the minimum value over all tests, since this is a lower bound on the running time. We found this value to be 41.6 seconds. However, it ranged up to 133 seconds; this variability is a fundamental characteristic of on-demand computing. It could be ameliorated by using more expensive reserved Cloud computing infrastructure.

E. Variability

We ran a test to determine the variability of running times. We used Part A, the same three configurations and five node numbers used above, and ran each combination of configuration and number of nodes five times. The results are shown in Table II. We found that the average node runtimes had a larger variance as the number of nodes increased. For 10 nodes, the

	Configuration 1		Configuration 2		Configuration 3	
Number of nodes	PiCloud runtime (s)	MATLAB runtime (s)	PiCloud runtime (s)	MATLAB runtime (s)	PiCloud runtime (s)	MATLAB runtime (s)
10	139.4 ± 9.2	72.6 ± 3.7	1022.2 ± 75.1	944.2 ± 66.5	1109.1 ± 106.6	1023.9 ± 96.1
25	98.1 ± 21.7	29.5 ± 5.4	496.3 ± 77.9	407.8 ± 60.5	197.7 ± 26.6	140.1 ± 19.7
50	74.8 ± 18.0	14.2 ± 2.1	291.5 ± 56.6	208.9 ± 38.1	289.3 ± 39.5	218.3 ± 30.1
100	71.3 ± 18.7	8.0 ± 1.5	195.2 ± 30.5	107.2 ± 15.6	182.3 ± 29.8	106.2 ± 17.7
250	62.8 ± 12.8	4.1 ± 0.8	135.3 ± 18.9	44.1 ± 5.7	118.2 ± 18.7	42.8 ± 6.7
500	60.3 ± 11.5	2.6 ± 0.5	105.7 ± 20.9	20.0 ± 3.8	90.9 ± 19.9	18.8 ± 6.2

TABLE II

RESULTS FOR VARIATION IN RUNNING TIMES FOR CLOUD-BASED IMPLEMENTATION. THE TESTS WERE RUN ON PART A WITH 500 PART PERTURBATIONS DIVIDED EVENLY OVER THE NODES. CONFIGURATION 1 IS $d_C = 0$, $\rho = 1.5$, AND $|\Phi| = 5$; CONFIGURATION 2 IS $d_C = 0.09$, $\rho = 20$, AND $|\Phi| = 15$; AND CONFIGURATION 3 IS $d_C = 0.09$, $\rho = 20$, AND $|\Phi| = 15$. EACH TEST WAS RUN FIVE TIMES, AND THE AVERAGE NODE RUNTIMES FOR BOTH PICLOUD AND MATLAB ARE REPORTED HERE.

variance was 7.9% of the mean, but this increased up to 20.3% for 500 nodes. This is likely due to the variance in processing time for individual perturbations, which are averaged out over larger sets of perturbations when using fewer nodes.

F. Asynchronous Parallelism

The overall runtime of the algorithm is the maximum node runtime, since all nodes are started in parallel at the same time, but the algorithm is only finished once the results from all nodes are in. With more nodes, even though the average node runtime may drop considerably, it is more likely for a node to be further from the mean, driving up the overall runtime relative to the average. With this factor considered, we expected speedups for the overall runtime to be less than the speedups for the average node runtime.

This method of parallelism is called *synchronous parallelism*. To improve the overall running time relative to the average node runtime, we considered the practice of only waiting for a subset of nodes to finish, called *asynchronous parallelism*. A usual approach is to set a time limit, and only nodes that finish within the time limit are used. However, for this algorithm, the running time is not well-known in advance. Therefore, we considered starting a set of nodes to process 500 part perturbations, but only waiting for enough nodes to finish to obtain 100 perturbations (i.e., $\frac{1}{5}$ of the nodes). We tested this approach using 10, 50, 100, 250, and 500 nodes.

This reduces the vulnerability of runtime to outliers. We found that on average, this method would have reduced the overall PiCloud runtime on average 1.43×. The speedup ranged between 1.04× to 2.00×. It is possible that there is a correlation between the results of the algorithm and the PiCloud node runtime; in this case, taking the earliest nodes to complete may produce biased results. In future work, we will explore this possibility.

VII. CONCLUSION

We have presented an approach for quickly analyzing conservative-slip push grasps on planar parts by finding the value of a quality metric that estimates a lower bound on the probability of force closure. This sampling-based algorithm is well-suited for cloud-based execution as shown in Section VI. We investigated the number of perturbations needed to reliably evaluate the quality of a grasp, and the effect of increasing tolerance on grasp quality. We have also presented an adaptive elimination procedure to remove low-quality grasps after a number of part perturbations have been tested. The adaptive elimination step reduces grasp evaluations by 91.5% while maintaining 92.6% of grasp quality. We reported results from a Cloud-based implementation, obtaining a maximum of 445× speedup with 500 nodes, suggesting our algorithm scales well with increasing parallelism.

In future work we will extend our algorithm to other kinds of uncertainty. This includes part pose, in which the part location may not be known precisely but an estimate is available, robot mechanics, where the robot may have imprecise actuators. We will take advantage of other opportunities for parallelism in our algorithm, further scaling it in the Cloud. Our sampling-based approach is a flexible one, and we will extend it to grasping in 3D, to non-polygonal parts with curved surfaces, and to multi-DoF grippers.

A. Belief Space-based Optimization

In Section V-E we demonstrated a procedure for finding the maximum tolerance that would allow for a given desired probability of success. In an automation setting, however, these two quantities may each have an associated cost; tighter tolerances incur higher manufacturing costs, and a lower probability of success means more grasping failures, also incurring costs. Because these two quantities are at odds with each other, we can formulate an optimization problem to determine the best balance given the costs.

To create this optimization problem we change the tolerance Σ from a parameter to a variable. The quality of a grasp g on a part S, $Q(g, S, \Sigma; \theta)$, is then a distribution defined by the tolerance Σ. The technique of optimizing over distributions with the variance in the state is termed a *belief space* optimization [26].

We define two costs, the scalar *failure cost* c_F and the *tolerance cost* matrix C_Σ. The optimal grasp and tolerance are then found as follows:

$$g^*, \Sigma^* = \arg\min_{g \in G, \Sigma} c_F(1 - Q(g, S, \Sigma; \theta)) + C_\Sigma \Sigma$$

This optimization problem is difficult to solve using the techniques presented in this paper; if the value of Q is determined using Monte Carlo methods, calculating the gradient would involve repeated evaluation of the function. While the expansion of cloud computing infrastructure and capability in the future may allow sampling-based techniques to calculate the gradient in a timely manner, in future work we will explore methods to more efficiently solve this optimization problem.

B. Code and Data Availability

Further information including code and data is available at: http://automation.berkeley.edu/cloud-based-grasping

ACKNOWLEDGMENTS

This work has been supported in part by funding from Google, Cisco, the UC Berkeley Center for Information Technology in the Interest of Society (CITRIS) and by the U.S. National Science Foundation under Award IIS-1227536: Multilateral Manipulation by Human-Robot Collaborative Systems. We thank Dmitry Berenson, Melissa Goldstein, Frank Ong, and Edward Lee, who all participated in this research, and James Kuffner and Frank van der Stappen for valuable discussions.

REFERENCES

[1] "Amazon EC2 FAQs," http://aws.amazon.com/ec2/faqs/\#What_is_an_EC2_Compute_Unit_and_why_did_you_introduce_it.
[2] "PiCloud," http://picloud.com/.
[3] "PiCloud | Pricing," http://picloud.com/pricing/.
[4] S. Akella and M. Mason, "Posing polygonal objects in the plane by pushing," in *IEEE International Conference on Robotics and Automation*. IEEE Comput. Soc. Press, pp. 2255–2262.
[5] R. Arumugam, V. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. Kong, A. Kumar, K. Meng, and G. Kit, in *IEEE International Conference on Robotics and Automation*, pp. 3084–3089.
[6] D. Berenson, S. S. Srinivasa, and J. J. Kuffner, "Addressing pose uncertainty in manipulation planning using Task Space Regions," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1419–1425, Oct.
[7] R. C. Brost, "Automatic Grasp Planning in the Presence of Uncertainty," *The International Journal of Robotics Research*, no. 1, pp. 3–17, Feb.
[8] J. Chen, K. Goldberg, M. H. Overmars, D. Halperin, K. F. Böhringer, and Y. Zhuang, "Computing tolerance parameters for fixturing and feeding," *Assembly Automation*, no. 2, pp. 163–172.
[9] J.-S. Cheong, H. J. Haverkort, and A. F. van der Stappen, "Computing All Immobilizing Grasps of a Simple Polygon with Few Contacts," *Algorithmica*, no. 2, pp. 117–136, Dec.
[10] J.-S. Cheong, H. Kruger, and A. F. van der Stappen, "Output-Sensitive Computation of Force-Closure Grasps of a Semi-Algebraic Object," *IEEE Transactions on Automation Science and Engineering*, no. 3, pp. 495–505, Jul.
[11] V. N. Christopoulos and P. R. Schrater, "Handling Shape and Contact Location Uncertainty in Grasping Two-Dimensional Planar Objects," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1557–1563.
[12] M. Ciocarlie, K. Hsiao, E. Jones, S. Chitta, R. Rusu, and I. Sucan, "Towards reliable grasping and manipulation in household environments," in *Intl. Symposium on Experimental Robotics*, pp. 1–12.
[13] M. Ciocarlie, C. Pantofaru, K. Hsiao, G. Bradski, P. Brook, and E. Dreyfuss, "A Side of Data With My Robot," *IEEE Robotics & Automation Magazine*, no. 2, pp. 44–57, Jun.
[14] J. Cornelia and R. Suarez, "Efficient Determination of Four-Point Form-Closure Optimal Constraints of Polygonal Objects," *IEEE Transactions on Automation Science and Engineering*, no. 1, pp. 121–130, Jan.
[15] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Communications of the ACM*, no. 1, p. 107.
[16] M. R. Dogar and S. S. Srinivasa, "Push-grasping with dexterous hands: Mechanics and a method," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct., pp. 2123–2130.
[17] D. H. Douglas and T. K. Peucker, "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, no. 2, pp. 112–122, Oct.
[18] J. Felip and A. Morales, "Robust sensor-based grasp primitive for a three-finger robot hand," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct., pp. 1811–1816.
[19] J. Glover, D. Rus, and N. Roy, "Probabilistic Models of Object Geometry for Grasp Planning," in *Robotics: Science and Systems*.
[20] K. Goldberg, "Algorithmic Automation," http://goldberg.berkeley.edu/algorithmic-automation/.
[21] ——, "Orienting polygonal parts without sensors," *Algorithmica*, no. 2-4, pp. 201–225, Oct.
[22] K. Goldberg and B. Kehoe, "Cloud Robotics and Automation: A Survey of Related Work," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2013-5, 2013. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-5.html
[23] C. Goldfeder and P. K. Allen, "Data-Driven Grasping," *Autonomous Robots*, no. 1, pp. 1–20, Apr.
[24] K. Hsiao, L. P. Kaelbling, and T. Lozano-Perez, "Grasping POMDPs," in *IEEE International Conference on Robotics and Automation*. IEEE, Apr., pp. 4685–4692.
[25] L. Joskowicz, Y. Ostrovsky-Berman, and Y. Myers, "Efficient representation and computation of geometric uncertainty: The linear parametric model," *Precision Engineering*, no. 1, pp. 2–6, Jan.
[26] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra", "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.
[27] B. Kehoe, D. Berenson, and K. Goldberg, "Toward Cloud-Based Grasping with Uncertainty in Shape: Estimating Lower Bounds on Achieving Force Closure with Zero-Slip Push Grasps," in *IEEE International Conference on Robotics and Automation*. IEEE, 2012, to appear, available at http://goldberg.berkeley.edu/pubs/kehoe-cloud-icra-2012-final.pdf.
[28] ——, "Estimating Part Tolerance Bounds Based on Adaptive Cloud-Based Grasp Planning with Slip," in *IEEE International Conference on Automation Science and Engineering*. IEEE.
[29] E. Klingbeil, D. Rao, B. Carpenter, V. Ganapathi, A. Ng, and O. Khatib, "Grasping with Application to an Autonomous Checkout Robot," in *IEEE International Conference on Robotics and Automation*.
[30] J. J. Kuffner, "Cloud-Enabled Robots," in *IEEE-RAS International Conference on Humanoid Robotics*, Nashville, TN.
[31] K. Lynch, "The mechanics of fine manipulation by pushing," in *IEEE International Conference on Robotics and Automation*. IEEE Comput. Soc. Press, pp. 2269–2276.
[32] M. Mason, "Manipulator grasping and pushing operations," Massachusetts Inst. of Tech., Cambridge (USA). Artificial Intelligence Lab., Tech. Rep.
[33] G. McKee, "What is Networked Robotics?" *Informatics in Control Automation and Robotics*, pp. 35–45.
[34] V.-D. Nguyen, "Constructing stable force-closure grasps," in *ACM Fall Joint Computer Conference*. IEEE Computer Society Press, pp. 129–137.
[35] M. Peshkin and A. Sanderson, "Planning robotic manipulation strategies for sliding objects," in *IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers, pp. 696–701.
[36] R. Platt, L. Kaelbling, T. Lozano-Perez, and R. Tedrake, "Simultaneous Localization and Grasping as a Belief Space Control Problem," in *International Symposium on Robotics Research*, pp. 1–16.
[37] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," *Computer Graphics and Image Processing*, no. 3, pp. 244–256, Nov.
[38] A. A. G. Requicha, "Toward a Theory of Geometric Tolerancing," *The International Journal of Robotics Research*, no. 4, pp. 45–60, Dec.
[39] A. Rodriguez, M. Mason, and S. Ferry, "From Caging to Grasping," in *Robotics: Science and Systems*, Los Angeles, CA, USA.
[40] C. Rosales, L. Ros, J. M. Porta, and R. Suarez, "Synthesizing Grasp Configurations with Specified Contact Regions," *The International Journal of Robotics Research*, Jul.

[41] J. D. Schulman, K. Goldberg, and P. Abbeel, "Grasping and Fixturing as Submodular Coverage Problems," in *International Symposium on Robotics Research*, pp. 1–12.
[42] G. Smith, E. Lee, K. Goldberg, K. Bohringer, and J. Craig, "Computing parallel-jaw grips," *IEEE International Conference on Robotics and Automation*, pp. 1897–1903.
[43] C.-P. Tung and A. Kak, "Fast construction of force-closure grasps," *IEEE Transactions on Robotics and Automation*, no. 4, pp. 615–626, Jun.
[44] M. Waibel, "RoboEarth: A World Wide Web for Robots. Automaton Blog, IEEE Spectrum."
[45] T. Zhang, L. Cheung, and K. Goldberg, "Shape Tolerance for Robot Gripper Jaws," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1782–1787.

Ben Kehoe received his B.A. in Physics and Mathematics from Hamline University in 2006. He is now a Ph.D. student in the ME department at University of California, Berkeley. His research interests include cloud robotics, medical robotics, controls, and grasping.

Deepak Warrier is a second year undergraduate in the Electric Engineering and Computer Science department at University of California, Berkeley. His reseach interests include parallel programming, machine learning, and developing artificial intelligence algorithms.

Sachin Patil received his B.Tech degree in Computer Science and Engineering from the Indian Institute of Technology, Bombay, in 2006 and his Ph.D. degree in Computer Science from the University of North Carolina at Chapel Hill, NC in 2012. He is now a postdoctoral researcher in the EECS department at University of California, Berkeley. His research interests include motion planning, cloud robotics, and medical robotics.

Ken Goldberg is Professor of Industrial Engineering and Operations Research at UC Berkeley, with appointments in Electrical Engineering, Computer Science, Art Practice, and the School of Information. He was appointed Editor-in-Chief of the IEEE Transactions on Automation Science and Engineering (T-ASE) in 2011 and served two terms (2006–2009) as Vice-President of Technical Activities for the IEEE Robotics and Automation Society. Goldberg earned his PhD in Computer Science from Carnegie Mellon University in 1990. Goldberg is Founding Co-Chair of the IEEE Technical Committee on Networked Robots and Founding Chair of the (T-ASE) Advisory Board. Goldberg has published over 200 refereed papers and awarded eight US patents, the NSF Presidential Faculty Fellowship (1995), the Joseph Engelberger Award (2000), the IEEE Major Educational Innovation Award (2001) and in 2005 was named IEEE Fellow.