Effective Multifingered Grasp Synthesis[1]

Jefferson A. Coelho Jr. and Roderic A. Grupen

Laboratory for Perceptual Robotics
Departament of Computer Science
University of Massachusetts, Amherst MA 01003

Technical Report 94-12
February 18, 1994

Abstract

This report discusses the issues involved in developing a grasp controller, within the framework of control composition, and introduces control pre-imaging. Pre-imaging is a design technique for augmenting the performance of a baseline controller through the opportune activation of metalevel control actions. Such metalevel control actions are activated whenever the current system state is associated to past unsuccessful patterns of interaction between the existing predictable, stable controller and the environment. The technique is illustrated by the implementation of a grasp controller capable of generating grasp configurations through successive, local refinements, given the position and normal of each contact. Our objective is to demonstrate the value of formally designed control modules to the design and implementation of effective controllers for complex control tasks.

[1]This work is supported in part by NSF CDA-8922572, IRI-9116297, IRI-9208920, and CNPq 202107/90.6

Contents

1 Introduction

Since the early days of robotics, one of its long-term goals has been to create efficient and flexible mechanisms to expand the spectrum of what can accomplished by both persons and machines. Autonomous navigation, robotic walking/exploration, automated manufacturing, and dextrous manipulation are examples of problem domains committed to this original goal. Typically, such problem domains are highly complex: multiple modes of sensory information are usually available, and the associated control problems are high-dimensional, non-linear, and plagued by uncertainties in sensor data and actuation.

In recent years, interest in behavior-based controllers has grown in the robotics community, mainly because they were perceived as having the potential to overcome the complexity associated with the control of autonomous, sensor-intensive, redundant systems operating in unstructured environments. In the behavior-based control framework, the overall state space is decomposed into less complex subspaces, and a behavior is assigned to each subspace. Each behavior addresses a particular aspect of the overall control task, resulting in improved run-time performance and robustness [5].

However, decomposing the control task into several tractable control subtasks is just half of the solution: a control strategy for a broad range of tasks and perceptual contexts must be built from the constituent controllers. Building control strategies from a given basis or repertoire of controllers is the control composition problem.

In this work, we study the control composition problem, in the context of grasp synthesis for dextrous hands. Section 2 presents a learning intensive approach to the grasp synthesis problem, illustrating the issues involved in developing a general purpose *grasp controller*. Section 3 formalizes the composition problem and describes two controllers that form a basis for construction of effective grasp controllers. Two distinct context-dependent control composition schemes are presented to demonstrate pertinent issues. Section 4 presents the pre-imaging (PI) technique (the central contribution of this thesis), designed to augment the performance of composite controllers by exploiting the control characteristics of the constituent regulators. Section 5 describes the implementation of the grasp controller as a component controller within a system capable of performing autonomous, collision-free reaching and grasping of objects. Section 6 concludes this article and discusses issues to be explored in the future. Appendix A shows how the composition problem can be cast as an optimal control problem and how to accomplish time-optimal grasp synthesis, using the same grasp controllers mentioned earlier. Appendix B characterizes formally the knowledge-based grasp controller introduced in Section 3, and proves its completeness for certain grasp tasks and classes of objects.

2 Learning Primitive Grasping Actions

To grasp an object involves a sequence of complex actions, from object identification up to manipulator/finger positioning. In this work, to grasp an object means to position contacts on its surface so as to realize a desired wrench task. Grasp synthesis denotes the process of designing a contact configuration capable of delivering the desired wrenches to the object. The grasp synthesis process can be executed either off-line or on-line: in the off-line approach, the final contact positions are computed before the grasp process starts, and there is no concern about *how* to generate the final contact configuration, or which intermediate steps should be taken in order to get to the final configuration.

This work concerns on-line grasp synthesis: the objective is the development of a feedback control mechanism for incremental grasp formation. Closed loop grasp synthesis can accommodate geometric uncertainty during grasp, as small errors in sensing and actuation can be suppressed by the predictable performance of convergent controllers.

This section presents an implementation of a grasp controller, in which the control policy for atomic grasp actions is learned over a number of trials. This initial approach illustrates the issues involved in developing a general purpose, multifingered grasp controller.

2.1 Related Work

The grasp planning or grasp synthesis problem has been studied extensively. Most published work concerns off-line grasp synthesis, and three general approaches have been used, namely (1) geometric force closure grasp synthesis [12, 13, 29], (2) optimization-based grasp synthesis [6, 17, 18], and (3) taxonomy-based grasp synthesis [11, 28].

The first two approaches aim at finding the best grasp configuration under a certain grasp metric, either (1) purely based on object geometry or (2) based on specific aspects of grasp configurations. Taxonomy-based techniques are primarily concerned with searching in a taxonomy tree for the best manipulator configuration given the task and object geometry. Some approaches restrict the number of contacts according to the object dimensionality (2 contacts in 2D and 3D [6], 2 contacts in 2D [12], at least 3 contacts in 3D and 2 contacts in 2D [18]), others constrain the object representation (polygons or polyhedra are assumed in [13, 29], and smooth/parametric models are required in [6, 12, 17]).

None of the approaches above yields a grasp controller. Grasps are preceded by off-line computation of the best grasp configuration, and the algorithmic complexity can be very high [12] or the computation may involve exhaustive search over the object geometry [13, 29]. Because the approach in [18] only requires local information (contact position and normal), the authors claim that real-time implementation is possible. However, the authors report inadequate grasps due to local minima, and residual moments may be exerted on the object.

With the exception of [18], no other approach allows for incremental models of the object. A complete model (either in terms of its real geometry or in terms of a smooth model) is required before grasp synthesis starts, and it is not clear how incremental object models could be accommodated in the framework of these approaches.

2.2 Learning Primitive Grasping Actions

In its first implementation, the grasp controller executes atomic actions[2] selected according to a control policy expressed as a map relating state to actions. The underlying control policy is learned

[2]Atomic actions denote elementary control actions requiring little or no computation.

over a number of trials using Q-learning [31]. The Q-learning procedure incrementally computes the expected reward of selecting each action, for each state; the corresponding "greedy" control policy is simultaneously derived.

For the sake of discussion, a simple 2D grasp controller was constructed, based on three atomic actions: each individual contact could either move a fixed angular distance clock- or counterclockwise, or remain in its current position. The remainder of this section details the Q-learning procedure setup to compute the expected reward of each state-action pair. Our goal is to establish a benchmark control performance and learning rate to which we may compare subsequent alternative control designs.

2.2.1 Learning Problem Setup

State Representation: In this first implementation, state information is essentially geometrical, based on the angular position of all contacts involved in the grasp. Specifically, the object surface is divided into 12 bins, corresponding to angular sectors of 30^0 degrees. Assuming four-fingered grasps (or four contacts), the grasp state is completely specified by a 4-tuple containing the bin index of each contact. This grasp state representation is unique for a given object, provided that object's geometry and orientation are known. Considering three possible actions for each contact, a total of 3^4 actions is possible from each state. So, even for this coarse state codification, there is a total of $12^4 3^4 = 1,679,616$ entries in the state-action table.

Trial Description: In each trial, four contacts are randomly placed in four different bins over the object surface. Each contact selects one of the three possible atomic actions, in each control step. Therefore, it is possible to generate any contact configuration within the limit of at most 10 control steps, no matter the initial configuration. Accordingly, each trial terminates upon convergence or after at most 10 control steps, whatever comes first.

Convergence: The grasp synthesis process converges as soon as a satisfactory grasp configuration is generated. In this work, all grasp configurations are scored by the same metric, described in Section 3.5, where the grasp metric and related issues are fully discussed.

Training: Training proceeded for 500 epochs; each epoch consisted of 90 grasp trials over the same object – the irregular triangle, shown on Figure 2(a). A total of 45,000 grasp trials were attempted during training.

Update equation:

$$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha(R + \gamma \max_{a \in A} Q(s_{t+1})),$$

where $\alpha = 0.02$ and $\gamma = 0.9$. After each control step, the Q-table is updated using the update equation above with $R = 0$, and the resulting grasp configuration is tested for convergence. Once the trial is finished, the grasp solution is evaluated as either failure or as success, and the Q-table is updated with $R = -10$ (punishment) or $R = 10$ (reward), respectively.

Action selection: The next action is chosen randomly, according to a Boltzmann probability distribution over the Q-values associated with each action. The probability that action a^i will be selected, given that the system is at state s at time t is

$$Pr(a_{t+1} = a^i) = \frac{e^{\frac{Q(s,a^i)}{T}}}{\sum_{j=1}^{n} e^{\frac{Q(s,a^j)}{T}}}.$$

The temperature T varied in the empirically determined range of $[20.0, 0.005]$.

2.3 Results

Figure 1 is the plot of average failure rate as a function of the number of training epochs, for the irregular triangle. This plot was smoothed by averaging the raw data over a 10-epoch wide sliding window.

After the training stage, the performance of the resulting controller was assessed over 200 grasp trials, for the irregular triangle in its original orientation and for a 30^0-rotated version, depicted in Figure 2(b). The goal of this comparison was to evaluate the invariance of the control policy with respect to orientation. As before, each trial would proceed up to 10 steps or until convergence. No failures were observed for the irregular triangle, but the controller failed to generate a satisfactory grasp solution in 29.0% of the trials for the rotated irregular triangle. For the irregular triangle, 99 different grasp configurations were generated (all evaluated as satisfactory), while for the rotated triangle 81 distinct grasp configurations (55 evaluated as satisfactory) were generated. Figure 2 shows the four grasp configurations most frequently generated for the irregular triangle and the rotated irregular triangle object. The frequency each configuration was generated over 200 grasp trials is also indicated, for both satisfactory and unsatisfactory grasp configurations.

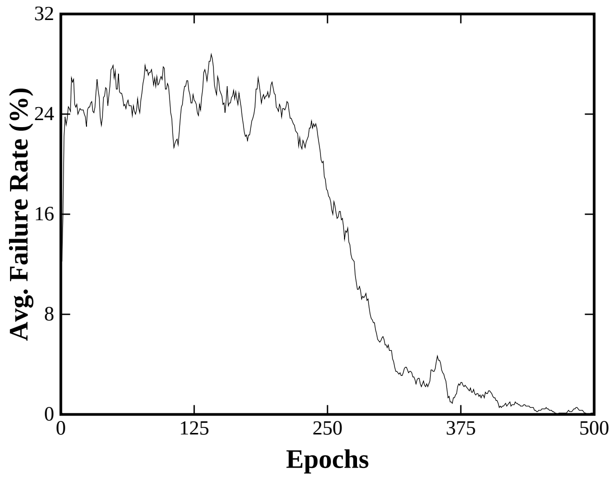


Figure 1: Average failure rate during training. Each epoch consists of 90 grasp trials over the irregular triangle object (oriented at 0^0). The plot was smoothed by averaging raw data over a 10-epoch wide sliding window.

2.4 Discussion

As formulated above, the problem of synthesizing grasp configurations from a given initial configuration is no different than finding the solution of a maze from an arbitrary initial position. In the case of the irregular triangle, the Q-learning method learned how to select an appropriate control sequence given the current state. When the same control policy was applied to a different object, some past action sequences were still successful; this explains the successes observed for the rotated irregular triangle object.

The average failure rate observed for the rotated irregular triangle (29.0%), indicates that the control policy derived by the learning scheme described is orientation-dependent. Furthermore, it is likely

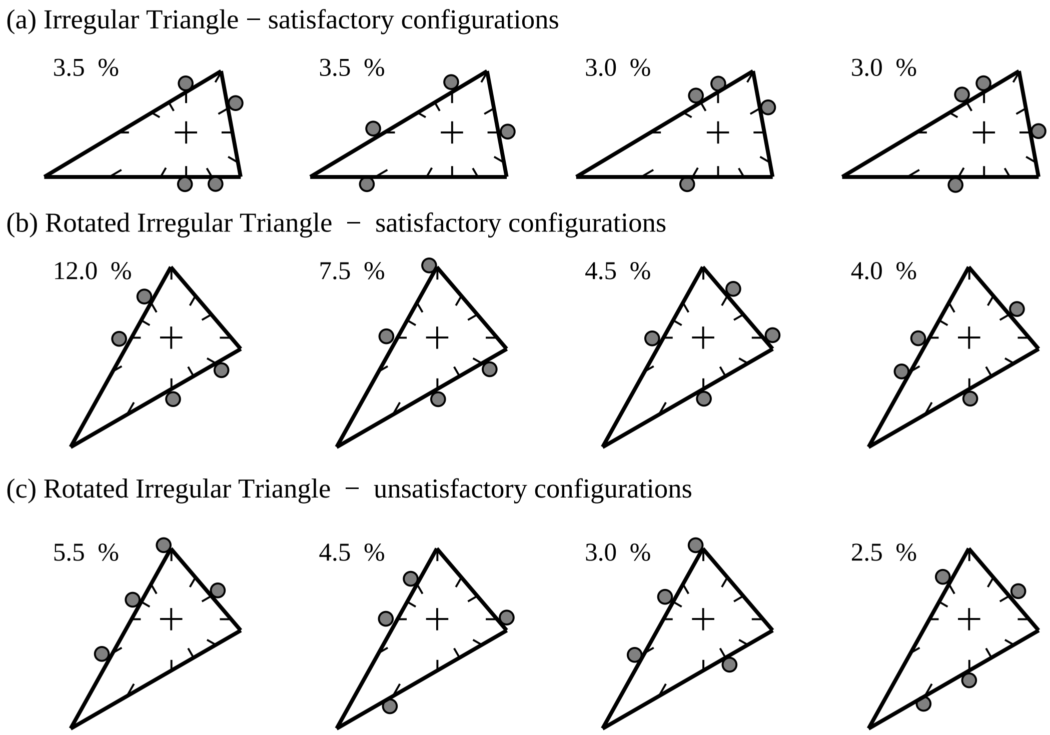


Figure 2: The four configurations most frequently generated for (a) irregular triangle, (b) the rotated irregular triangle (satisfactory configurations) and (c) unsatisfactory configurations. The number immediately on the left of each configuration reflects the frequency it was generated.

that controller performance will degrade as the geometric dissimilarity between a given object and the irregular triangle object grows.

Dependence on object geometry and orientation is related to the state representation employed. Controllers based on geometric representations do not respond well to geometric uncertainty and do not generalize over objects as well as force-domain derived representations.

State representation also directly affects the acquisition of a control policy through experience. Grasp state representation directly affects the size of the search space, and therefore, the convergence rate. Also, each grasp state must be uniquely described by the adopted representation: the presence of indistinguishable ("hidden") states may hinder convergence or even make the learning process inviable.

The two problems just discussed – how to represent the state of a grasp and how to derive a control policy based on the chosen representation scheme – should not be treated independently. Grasp states can be uniquely represented in several different ways, but only representations allowing for the derivation of effective grasp control policies should qualify as potential solutions; finding an adequate state representation is an important part of problem solution.

2.5 Conclusion

Our implementation of an atomic action-based grasp controller lacks a context independent state representation capable of encoding a rich class of correct grasp solutions. The work that follows demonstrates that these deficiencies can be addressed by replacing atomic actions by controllers. Section 3.6 discusses the value of adopting a control-based state representation, in terms of conferring context independence to the composite controller. Moreover, the idea of using control-based representations can be further expanded, by associating to each state its relevance to the final control objectives. This is the basis for the pre-imaging (PI) technique, to be described in Section 4.

3 Control Composition: Formalization and Application

3.1 Behavior-Based Architectures

Complex, autonomous, sensor-based control tasks (e.g., dextrous manipulation, robotic exploration) usually involve redundant sensory information and actuation, i.e., there is more than one way of generating state information and of accomplishing a particular task. Also, solutions must frequently satisfy arbitrary constraints (kinematic workspace limitations, obstacle avoidance, etc.), increasing further the complexity of the control task.

In order to cope with control complexity researchers have proposed behavior-based control architectures, where the state space S is decomposed into subspaces S_i, and behaviors β_i are assigned to the S_i to address subtasks relevant to the task domain. The control task domain must be *decomposed* into subtasks and the individual behaviors must be *composed* to form a composite controller capable of solving a family of specific control tasks.

3.1.1 Related Work

Behaviors have been defined [5] as control modules designed to execute particular control actions when triggered by a specific sensor data pattern. Little (if any) interpretation of sensory data is performed; control actions are computed by fixed procedures, closely linked to sensory information. In general, behaviors use local or partially global models of the environment, and some implementations advocate against the use of any explicit model.

In terms of behavior composition, three approaches sum up the alternatives tried so far: (i) handcrafted behavior composition [1, 5, 21, 26], (ii) potential function-based composition[25], or (iii) composition learning [24]. Robot navigation is the problem domain for most of the systems [8, 21, 24, 26]. In [1, 25] a sequence of actions is sought for an abstract, simulated problem. For most implementations [1, 5, 21, 26], the composition strategy is determined off-line, hardwiring sensor data to control actions.

The advantages of using behaviors are: (1) information acquisition becomes more directed to specific features of the environment, alleviating the burden of interpreting sensory inputs and keeping complete models of the environment, and (2) control actuation over the system is simplified, due to the straightforward mapping from sensory information to control actions. As a corollary of (1) and (2), the overall control complexity is reduced, making tractable rather complex control tasks. Furthermore, by encapsulating low and intermediate level expertise about the problem domain, behaviors constitute a new abstraction level on top of the original level formed by atomic actions.

3.1.2 Issues on Control Composition

Although regarded as successful in its problem domain, the behavior-based paradigm leaves unanswered important issues, chiefly derived from the lack of methodology in composing appropriate solutions from a repertoire of controllers:

1. Composition complexity: The human designer is an active element in the synthesis process of behavior-based controllers: ultimately, if the "emergent" behavior is not right for some reason, a human designer is called to recraft the composition strategy, or to patch the behavioral repertoire. Indiscriminate proliferation of behavior may re-introduce the planning complexity that control decomposition sought to avoid, since control composition is itself exponential in nature.

Furthermore, the control characteristics of the resulting controller are not known in general. Because of this, the results of adding an extra control layer on top of an existing controller are not predictable, in terms of control stability and performance.

2. Generalization capabilities: Handcrafted compositions tune intricate behavior-based architectures on the basis of observed performance in a particular environment. This procedure compiles robot, environment, and task into the emergent sensorimotor strategy. As such, it can not be expected to generalize well over tasks, robots, or sensors.

In order to improve on the existing control architectures, we propose that constituent controllers should be formally designed from a *declarative* control specification (as opposed to procedural behaviors), and implemented as asymptotically stable controllers, amenable to composition into a predictable composite controller.

3.1.3 Procedural and Declarative Behaviors

Procedural behaviors are characterized as device-, task-dependent control actions, triggered by direct sensor data. In contrast, declarative control specifications express more abstract objectives, independent of sensors, mechanisms, the environment, or the task. Grasp stability, or more generally, imparting a force to the world through a mechanical contact, is such an objective. This goal can be expressed independent of the number of fingers, hands and arms employed, independent (largely) of the object geometry and task. It is important to note that the control *specification* is declarative, not the derived controller. The controller marshals kinematic and sensory resources as required by the task in the current context.

3.1.4 Formal Design and Predictability

Ideally, a formal design methodology would (1) establish a minimum set of control characteristics each constituent behavior must possess, and (2) establish a composition scheme capable of preserving those control properties, yielding predictable yet adaptable composite controllers. The next section focuses on efforts to synthesize provable correct controllers.

3.2 Provable Correct Controllers

Many researchers have explored the issue of how to synthesize provable correct controllers, attempting to determine an effective set of requirements or properties the controller must possess to be considered correct. Navigation functions [20] and harmonic functions [9] have been used as to generate sufficient control surfaces.

3.2.1 Navigation Functions

In [20], Koditschek and Rimon describe how to construct a correct navigation function, for a point robot moving amidst spherical obstacles within a bounded workspace. Of special interest are the conditions a map $\phi : \mathcal{S} \rightarrow [0, 1]$ must met to be considered a navigation function:

1. ϕ must be continuous and analytic on its domain;
2. ϕ must be polar, i.e., must have an unique minimum on its domain;

3. ϕ must be Morse on its domain — this implies that the Hessian of ϕ must be non-singular in each of ϕ's critical points (points where the gradient of ϕ goes to zero);
4. ϕ has its maximal value (uniformly) exactly on the boundary of its domain;
5. The gradient of ϕ is bounded over the entire domain.

The gradient of a navigation function ϕ can be proven to yield a sufficient controller, under the conditions above. Although effectively solving the robot navigation problem for some theoretic domains, the prerequisites for control sufficiency presented in [20] are difficult to apply in practice. First, it is assumed complete knowledge about the environment, what is often not a realistic assumption. Provably correct navigation functions can be constructed, up to world model precision, albeit at a high computational complexity. Second, the sequence of transformations required to complete the control surface can be very sensitive to relatively small changes in the environmental geometry.

3.2.2 Harmonic Functions

Connolly and Grupen [9] illustrate how harmonic functions can be used as basis for developing sufficient controllers. Harmonic functions are linearly superimposable and yield complete navigation functions, generating smooth, continuous trajectories in c-space. Furthermore, this approach may be applied to general, incomplete geometry, and can be recomputed incrementally.

3.3 Building Controllers Through Composition

In the control composition framework, elements from a finite set of control modules are composed into a composite controller. The Hessian matrix can be used to characterize the individual controllers and to analyze the resulting composite controllers. The Hessian matrix of a multivariable control function, F, defined on the domain, Θ, is defined as $\partial^2 F/\partial\Theta^2$, or in matrix notation

$$\partial^2 F/\partial\Theta^2 = \begin{bmatrix} \frac{\partial^2 F}{\partial\theta_1^2} & \frac{\partial^2 F}{\partial\theta_1\partial\theta_2} & \cdots & \frac{\partial^2 F}{\partial\theta_1\partial\theta_n} \\ & \vdots & \vdots & \\ \frac{\partial^2 F}{\partial\theta_n\partial\theta_1} & \frac{\partial^2 F}{\partial\theta_n\partial\theta_2} & \cdots & \frac{\partial^2 F}{\partial\theta_n^2} \end{bmatrix}. \quad (1)$$

Three classes of controllers can be distinguished by examining properties of the Hessian of F:

Convex controller: If the Hessian matrix is positive definite over the domain Θ, then the resulting controller is convex over Θ. The control function will possess exactly one minimum.

Harmonic controller: If the trace of the Hessian matrix is identically zero, then the control function is a solution to the Laplace's equation, or a harmonic function. As mentioned earlier, harmonic functions can be used to construct complete controllers.

Sub-harmonic controller: If the trace of the Hessian matrix (or equivalently, the Laplacian of the control function F) is non-positive,

$$trace(\frac{\partial^2 F}{\partial\Theta^2}) = \frac{\partial^2 F}{\partial\theta_0^2} + \frac{\partial^2 F}{\partial\theta_1^2} + \cdots + \frac{\partial^2 F}{\partial\theta_n^2} \leq 0$$

then this implies that local maxima may be present on control surface. Because regions around these local maxima are inaccessible to all states at a lower potential, the controller may no longer be complete.

All these classes of controllers (convex, harmonic, and sub-harmonic) are closed under linear superposition: given two control functions F_1 and F_2 belonging to class C, their linear superposition $F' = a_1F_1 + a_2F_2$ will also belong to class C, provided both a_1 and a_2 (called the superposition coefficients) are non-negative, but not simultaneously zero.

3.3.1 Control Properties of the Composite Controller

While convexity guarantees convergence and monotonic error decrease for individual controllers, neither of these properties are necessarily inherited by the composite controller, because of the dynamic nature of the composition strategy. In the current framework, the superposition coefficients are a function of system state; limit cycles may arise in particular situations, preventing convergence. Therefore, composition may corrupt some of the "good" properties the individual controllers possess, if not properly addressed. Each composition strategy covered in this work deals with this issue to a different extent: knowledge-based composition (Section 3.6.1) uses a steep, heuristic switching boundary positioned to yield provably correct grasp geometries under certain conditions (Appendix B). Learning-derived composition (Section 3.6.2) derives a strategy to avoid undesirable states over a number of trials. Finally the pre-imaging controller manages the control activation policy to optimize the performance of a particular control composition strategy.

3.4 A Basis for Grasp Control

Grasp synthesis presumes the existence of a grasp metric which is being maximized during the synthesis process. Grasp metrics usually emphasize one or more aspects of the resulting grasp configuration: capacity to resisting external disturbances [17], distance between contacts [6], ratio between resulting wrenches and applied forces, or minimization of total forces applied to the object [13], grasp stability [18], and others.

This section describes two controllers, respectively the force and moment closure controllers, that later will be composed into a grasp controller. Both controllers, originally introduced by Grupen et. al. [15, 14], are fixed-time, computationally inexpensive controllers of complexity $O(n)$ on the number of contacts. They are based on simple wrench domain models of the object-manipulator interaction and on a simplified model of the grasping task, wherein:

- Contacts are assumed to be frictionless point contacts, and contact forces have unit magnitude;
- Object geometry is scaled such that the maximum moment is also unitary; the solution contact geometry is an optimization based on shape, rather than dimension.

Both controllers are designed around the stable grasp sufficiency metric, that reflects to which extent the wrench task is attained by the current contact configuration.

3.4.1 Stable Grasp Sufficiency Metric

In scoring a grasp configuration, it is frequently useful to analyze the associated grasp matrix (also known as the grasp Jacobian). This matrix describes the transformation from a set of forces applied by contacts on the object surface to a set of object frame wrenches. The grasp matrix W is defined as $W = [W_1\ W_2\ \ldots W_n]$, where W_i is a six dimensional vector of wrenches applied at the i^{th} contact position.

One of the basic requirements that precedes most tasks is the construction of a *stable grasp.* A grasp configuration is considered stable if it can resist arbitrary perturbation wrenches. The null space of the grasp matrix defines the basis for the perturbation wrenches that can be resisted by the given contact configuration. If the null space is six dimensional, contact wrenches can be scaled up to suppress any arbitrary external disturbance, while keeping the desired net wrench on the object.

From the previous discussion, it's clear that a pre-condition for stable grasps is the construction of a null space within the grasp matrix. Grasp configurations that meet this pre-condition are referred to as *null grasps.* One possible strategy for null grasp synthesis would be (1) to check for the null grasp condition. If it is met, the synthesis process is over. If the current contact configuration doesn't yield a null grasp, then (2) move the contacts towards a configuration yielding null grasp. However, the procedure for computing the null space of a matrix doesn't allow for differentiability and therefore cannot be used as a control surface. The same shortcomings plague other grasp metrics: they are computationally expensive and frequently involve optimization procedures or matrix analysis techniques that don't offer directional information on which a *grasp controller* can be based.

To overcome those shortcomings, a sufficiency metric was devised [15], based on the residual wrench vector $\vec{\rho}$

$$\begin{aligned} \epsilon &= \vec{\rho}^T\vec{\rho} \\ &= \left(\vec{t} - \frac{1}{n}\sum_{i=1}^{n}\hat{\omega}_i\right)^T \left(\vec{t} - \frac{1}{n}\sum_{i=1}^{n}\hat{\omega}_i\right), \end{aligned} \tag{2}$$

where $\vec{\rho}$ expresses the net wrench over $1 \leq j \leq n$ contacts, $\vec{t}$ is an optional wrench closure *bias*, and $\vec{\omega}_i$ is the wrench vector resulting from the i^{th} interaction force. The elements $t_j \in [-1, 1]$ of $\vec{t}$ and the elements $w_{ij} \in [-1, 1]$ of $\vec{w}_i$ are qualitative in the sense that they do not reflect engineering units of force and torque, but express the relative ability of a contact configuration to transmit forces and torques through the object's surface. Minima in ϵ correspond to linear combinations of these qualitative wrenches which map into the null space of the grasp matrix.

By moving the contacts along the direction that minimizes ϵ, the grasp controller approaches the desired null grasp configuration. One plausible approach is to compute the gradient of ϵ with respect to the contact coordinates θ_i and move the contacts accordingly. Assuming that the function $\hat{\omega}_i(\theta_i)$ is differentiable, one can compute $\frac{\partial\epsilon}{\partial\theta_i}$:

$$\begin{aligned} \frac{\partial\epsilon}{\partial\theta_i} &= 2\left(\vec{t} - \frac{1}{n}\sum_i\hat{\omega}_i\right)^T \frac{\partial\left(\vec{t} - \frac{1}{n}\sum_i\hat{\omega}_i\right)}{\partial\theta_i} \\ &= -2\left(\vec{t} - \frac{1}{n}\sum_i\hat{\omega}_i\right)^T \frac{\partial\omega_i}{\partial\theta_i}\frac{1}{n} \\ &= -\frac{2}{n}\vec{\rho}^T G_i \\ &= -\frac{2}{n}G_i^T\vec{\rho}, \end{aligned} \tag{3}$$

where G_i is the wrench gradient with respect to the contact coordinate θ_i. In vectorial notation

$$\begin{aligned} \frac{\partial\epsilon}{\partial\vec{\theta}} &= -\frac{2}{n}\begin{bmatrix} G_1^T \\ G_2^T \\ \vdots \\ G_n^T \end{bmatrix}\vec{\rho} \\ &= -\frac{2}{n}G^T\vec{\rho}. \end{aligned} \tag{4}$$

Equation 4 evaluates to $\vec{0}$ in one of two conditions: (a) $\vec{\rho} = \vec{0}$ and (b) when $G^T\vec{\rho} = \vec{0}$. The first condition is the desired convergence criterion and results in a grasp geometry that can be *scaled* to span the task. The second condition is equivalent to a local minimum of ϵ. This class of minima results when a local tangent to the wrench surface, G, is orthogonal to the residual vector, $\vec{\rho}$. Both grasp controllers are based on Equation 4: what distinguishes them is exactly how $\hat{\omega}_i$ varies with the contact coordinate, θ_i.

3.4.2 Force Closure Controller

The force closure (FC) controller regulates the position of unit normal forces on the surface of a Gaussian sphere. If θ and ϕ are the angular coordinates on the Gaussian sphere, then the corresponding wrench domain model of the object $\vec{W}(\theta,\phi) = [F_x\ F_y\ F_z\ M_x\ M_y\ M_z]$ becomes[3].

$$\begin{aligned} w_1 &= f_x = -cos(\theta)cos(\phi) & w_4 &= m_x = 0 \\ w_2 &= f_y = -sin(\theta)cos(\phi) & w_5 &= m_y = 0 \\ w_3 &= f_z = -sin(\phi) & w_6 &= m_z = 0. \end{aligned}$$

The sufficiency metric for the FC controller yields:

$$\epsilon_{FC} = \left[t_{fx} + \frac{1}{n}\sum_{i=1}^{n} cos(\theta_i)cos(\phi_i)\right]^2 + \left[t_{fy} + \frac{1}{n}\sum_{i=1}^{n} sin(\theta_i)cos(\phi_i)\right]^2 + \left[t_{fz} + \frac{1}{n}\sum_{i=1}^{n} sin(\phi_i)\right]^2 \quad (5)$$

The gradient of Equation 5 with respect to θ_k yields:

$$\frac{\partial \epsilon_{FC}}{\partial \theta_k} = (A_{\theta k}^2 + B_{\theta k}^2)^{1/2} sin(\theta_k + \psi_k) \quad (6)$$

where $A_{\theta k}$, $B_{\theta k}$, and ψ_k are constants independent of θ_k:

$$\begin{aligned} A_{\theta k} &= \frac{2}{n}\left(t_{fy} + \frac{1}{n}\sum_{i\neq k}(sin(\theta_i)cos(\phi_i))\right) cos(\phi_k) \\ B_{\theta k} &= -\frac{2}{n}\left(t_{fx} + \frac{1}{n}\sum_{i\neq k}(cos(\theta_i)cos(\phi_i))\right) cos(\phi_k) \\ \psi_k &= tan^{-1}(A_{\theta k}/B_{\theta k}) \end{aligned}$$

Equation 6 demonstrates that the sufficiency metric employing the force closure model generates a unimodal error function. Figure 3(a) shows the plot of both the force closure metric (Equation 5, dashed curve) and the plot for the sufficiency metric (Equation 2, solid curve), for the square shape, 2 contacts. Those plots are obtained by fixing the black contact at the origin $\theta = 0$, and moving the white contact counterclockwise over the object surface. Figure 3(b) shows the complete object model; in the same figure, the sufficiency metric's local minima are marked over the object surface with triangles, and the white contact is positioned over the metric's global minimum. Figure 3(c) shows the corresponding constant curvature unit Gaussian sphere, and its global (and only) minimum.

[3]Note that using this assumption for regular polygons reduces the control of grasp sufficiency (wrench closure) to a force closure optimization. This is due to the fact that the moment diagrams for regular polygons exhibit characteristic frequencies of $n\frac{cycle}{2\pi}$ and are therefore ignored in the 1^{st} harmonic approximation (the unit Gaussian sphere). Thus the designation "force closure".

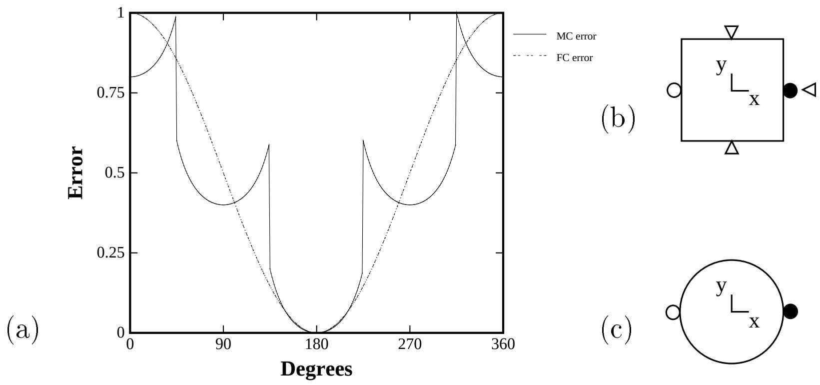


Figure 3: (a) Sufficiency metric error plot, for the complete object model (solid curve) and for the smoothed object model (dashed curve). (b) Square object being grasped by two contacts; the black contact stays fixed at the origin, while the white one moves around object. The triangles mark the position of the sufficiency metric local minima on the object surface. (c) FC object model. In this case, the global minima for both metrics coincided on $\theta = 180^o$.

Figure 3 illustrates how the FC controller may contribute to grasp synthesis: it is a global approximation of every manifold geometry with no spurious minima. For regular polygons, this heuristic control surface selects combinations of faces that minimize the force closure residual.

To allow for higher performance and predictable composition, a convex approximation of the FC metric is constructed that preserves the equilibrium point in the original control surface (a similar quadratic controller for ϕ_k can be derived by differentiating Equation 5 with respect to ϕ_k). The gradient on which the controller will be based on is then computed over the quadratic approximation function:

$$\left.\frac{\partial \epsilon}{\partial \theta}\right|_{\theta_k}^{FC} = 2\frac{\epsilon_{FC} - \epsilon^*_{FC}}{(\theta_k - \theta^*_k)}, \tag{7}$$

where ϵ^*_{FC} is the force closure error for $\theta_k = \theta^*_k$ and θ^*_k is the angle θ in which contact k minimizes ϵ_{FC}:

$$\theta^*_k = -tan^{-1}\left(\frac{A_{\theta k}}{B_{\theta k}}\right). \tag{8}$$

3.4.3 Moment Closure Controller

Reasoning in terms of surface curvature, a scaled version of the FC controller can capture the local shape of the sufficiency metric for all positive and negative curvatures: the scaling constant is proportional to $\frac{1}{r}$, where r is the radius of curvature. However, as $r \to \infty$ the FC gradient vanishes, since forces remain constant on planar surface facets. The moment closure (MC) controller minimizes the moment residual by adjusting each contact position on the local surface facet. We use the perpendicular of a plane, (r_0, θ_0, ϕ_0), to parameterize the contact plane. Figure 4 illustrates the parameterization and the geometry from which the wrench gradient is derived.

The forces transmitted through any planar face are constant at all contact positions on that face. The moments applied to the object, $\vec{m} = \vec{r} \times \vec{f}$, vary linearly with surface coordinate and pass through

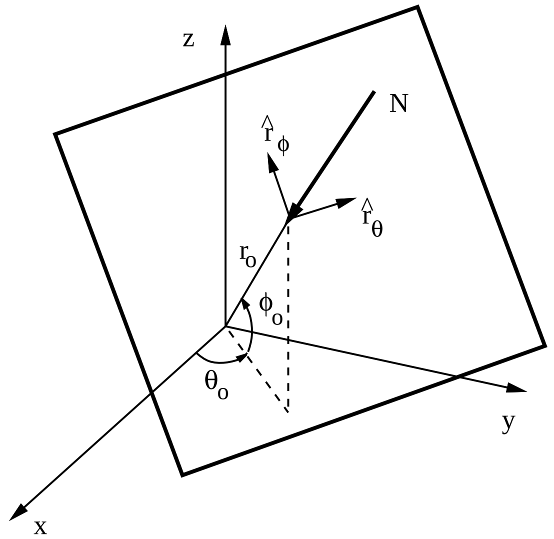


Figure 4: The Moment Closure Model derived from a Parametric Object Plane

zero where the perpendicular passes through the plane. The resulting wrench model is:

$$\begin{aligned} w_1 &= f_x = -cos(\theta_0)cos(\phi_0) & w_4 &= m_x = -r_\theta sin(\phi_0)cos(\theta_0) + r_\phi sin(\theta_0) \\ w_2 &= f_y = -sin(\theta_0)cos(\phi_0) & w_5 &= m_y = -r_\theta sin(\phi_0)sin(\theta_0) - r_\phi cos(\theta_0) \\ w_3 &= f_z = -sin(\phi_0) & w_6 &= m_z = r_\theta cos(\phi_0), \end{aligned} \tag{9}$$

where $(\hat{r_\theta}, \hat{r_\phi})$ are the surface coordinates of the contact in the plane. Note that this approximation yields constant contact forces which implies that control is derived from the moment diagrams exclusively; thus the designation "moment closure". The moment closure sufficiency metric can be written as follows:

$$\begin{aligned} \epsilon_{MC} &= \sum_i \epsilon_{MC}^{w_i}, \\ \epsilon_{MC}^{w_i} &= \left[t_{w_i} - \frac{1}{n} \sum_i w_i \right]^2 \end{aligned} \tag{10}$$

The controller based on the gradient of ϵ_{MC} is locally convex in surface coordinates, but not in polar coordinates. We developed a convex controller in polar coordinates using a quadratic controller with the same equilibrium state as the original convex metric, similarly to the convex FC controller. The planar surface coordinate where the gradient of ϵ_{MC} with respect to r_θ vanishes is given by

$$\begin{aligned} r_{\theta k}^* = &- \left[nt_{mx} - \sum_{i \neq k} m_{xi} \right] (sin(\phi_0)cos(\theta_0))_k - \left[nt_{my} - \sum_{i \neq k} m_{yi} \right] (sin(\phi_0)sin(\theta_0))_k \\ &+ \left[nt_{mz} - \sum_{i \neq k} m_{zi} \right] (cos(\phi_0))_k \end{aligned}$$

The corresponding contact coordinate of this minimum is:

$$\theta_k^* = tan^{-1}\left[\frac{r_{\theta k}^*}{r_0}\right] \tag{11}$$

so that,

$$\left.\frac{\partial \epsilon}{\partial \theta}\right|_{\theta k}^{MC} = 2\frac{\epsilon_{MC} - \epsilon_{MC}^*}{(\theta_k - \theta_k^*)}. \tag{12}$$

A similar result for the MC controller in the $\hat{\phi}$ direction can be derived by differentiating Equation 11 with respect to ϕ_k.

3.4.4 Common Characteristics

The FC controller is based on contact positions exclusively. The contacts navigate the surface of the continuous Gaussian sphere, and the resulting controller is globally convex. The MC controller considers both contact position and normal; the resulting control surface is piecewise convex. Each controller encodes a model of how local contact geometry transforms contact forces into contact wrenches, but no explicit model of the object itself is ever required – only local sensory information. Moreover, it is possible to assign an independent asynchronous controller to each contact. As a consequence, the complexity of the composite controller grows linearly with the number of contacts.

3.5 Grasp Evaluation Metric

In order to establish a benchmark with which to evaluate controller performance, a more complete metric is derived. The objective is to ensure an independent and more realistic evaluation of the grasp configurations produced by a particular control composition policy. This grasp evaluation metric considers frictional forces, and permits variable magnitude contact forces. A natural approach is to measure the quality of the null space associated with the given grasp configuration using, for instance, its rank as the quality measure. However, qualitatively different grasp configurations can look deceptively similar in terms of rank of null space, due in part to the fact that rank does not capture unisense wrench constraints and because rank only guarantees marginal (or infinitesimal) ability to suppress perturbations. To discriminate among grasp configurations with the same rank, the magnitude of both normal and tangential forces can be used as the distinguishing feature.

Basing grasp stability on friction forces is frequently undesirable, because friction is hard to model and uncertain: slip may occur as contacts move from one region over the object surface to another. In addition, all tangential forces require a complementary normal force and are therefore less efficient. To evaluate a grasp configuration, one needs to evaluate the resulting distribution of normal and tangential forces in the grasp. Many researchers have solved this problem using linear or quadratic programming; the implementation described here uses linear programming.

For each contact, the normal force N and the associated tangential (friction) force T are constrained so as:

- Normal forces must be compressive forces, with norm $\| N \|_2 \geq 1$;
- $\| T \|_2 \leq \| N \|_2$ (Coulomb friction coefficient $\mu \leq 1$).

Besides the natural constraints to the grasping task expressed above, null grasp constraints must also be satisfied; for n contacts:

$$\hat{t} - \frac{1}{n}\sum_{i=1}^{n} \vec{w}_i = 0$$

The objective function to be minimized is the sum of the norms of all forces involved in the grasp:

$$z = \sum_{i=1}^{n} \| N_i \|_2 + \| T_i \|_2$$

The set of linear constraints may not be satisfiable for a given contact geometry. If they can be satisfied, the final normalized grasp score is defined as

$$S_G = \frac{z}{n} - 1. \tag{13}$$

High values of S_G denote that either the system is relying heavily on tangential forces to stabilize the grasp (in which case the normal forces would also be large), or the contact forces are not balanced among the fingers, resulting in excessive normal forces in some fingers.

Grasp configurations will be evaluated as either "good" or "bad" based on the metric S_G. The decision of where to position the cutoff line between "good" and "bad" configurations is completely arbitrary. Figure 5 shows several grasp configurations and the corresponding scores, as computed by Equation 3.5. In this work, grasps with $S_G \leq 0.4$ are considered to be "good", or successful.

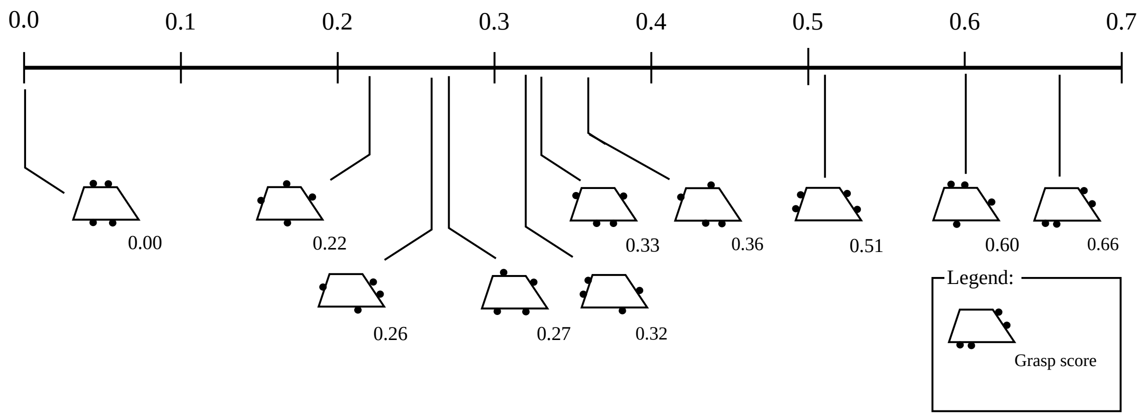


Figure 5: Distinct grasp configurations and respective scores S_G.

3.6 Control Composition

The force closure (FC) and moment closure (MC) controllers were composed into a grasp controller through knowledge-based and learning-derived composition strategies, as follows.

3.6.1 Knowledge-Based Heuristic Composition

Knowledge-based composition schemes assume that control composition can be encoded by a collection of simple activation rules. Such rules partition the state space into several regions, and different sets of controllers are activated simultaneously within different regions. Human experts are responsible for determining the activation rules, or shaping/profiling the decision surfaces.

In the grasp synthesis domain, a sensible heuristic rule is to address force residuals first, using a globally convex controller to take the system to the neighborhood of a grasp solution where the resulting forces over object are close to desired, and then refine contact positions to address moment errors, through the use of a locally convex controller. In terms of the local wrench domain models, the switching from the FC to the MC controller implies switching from a constant curvature model of the object surface (FC controller) to a model assuming zero local curvature (MC controller) – a planar facet. The final contact gradient after composition is

$$\nabla\epsilon = \alpha_{FC}\frac{\partial\epsilon_{FC}}{\partial\theta} + \alpha_{MC}\frac{\partial\epsilon_{MC}}{\partial\theta}.$$

Implementation: In the Luce's rule[4] formulation, the activaction coefficients α_{FC} and α_{MC} are determined based on the relative significance of each controller to the task, measured in terms of the current error ϵ. In our implementation, the coefficients α_{FC} and α_{MC} were computed by

$$\begin{aligned}\alpha_{FC} &= \frac{\exp(k_{FC}\epsilon_{FC}+b)}{\exp(k_{FC}\epsilon_{FC}+b)+\exp(k_{MC}\epsilon_{MC}-b)}\\ \alpha_{MC} &= 1-\alpha_{FC},\end{aligned}$$

where $k_{FC} = 4,000,\ b = -10$, and $k_{MC} = 0$.

The constants on the equations above are selected so as to give priority to the FC controller in the initial phase of the grasp. The MC controller is activated after ϵ_{FC} decreases below a preset threshold.

Knowledge-based heuristic composition can efficiently encode domain expertise, it is computationally inexpensive, doesn't require memory, and may have general applicability, depending on the quality of the heuristic rules. Moreover, this scheme yields convergence/completeness results for some classes of objects (see Appendix B).

3.6.2 Learning-Derived Composition

Learning methods have been extensively used in control. In [3], Barto presents an overview of connectionist methods applied to control of dynamical systems. Anticipatory control and system identification (forward, inverse, and differential models) are typical applications that benefit from the estimation and association capabilities offered by learning methods. Likewise, learning methods can associate past control actions with controller performance over time, effectively building a control policy or composition scheme that can be proved to be optimal under certain circumstances.

More sophisticated approaches to learning involve some kind of estimation of system performance, given the current state, and using such estimation to refine future control policies:

- In [4] Barto, Sutton, and Anderson describe an estimator of future system performance that is incrementally built through a learning process. After a number of trials, the estimator is able to correlate the present state to the expected final reinforcement. Using the expected reinforcement value, the control policy is continuously refined, from random action selection at first, to optimal action selection after convergence. Notice that this approach deals with the

[4]Luce's rule was originally formulated in the Game Theory domain to mediate competition between rival hypothesis. Given the subjective evidence for each hypothesis, Luce's rule determines hypothesis utility based on a set of parameters that weigh the relative importance among all hypotheses.

temporal credit assignment problem, where reinforcement is to be associated with past actions. Usually, the reinforcement signal is only available after a sequence of control actions, and a fraction of credit or blame is to be assigned to each action.

· In [22], Liu and Asada construct a forward model of the system that maps the current state and action vectors to an estimation of system performance. This estimator is built incrementally, using on-line data provided by the system. Once an accurate estimation is available, the approximate mapping from actions to performance index is used to refine the current control strategy, and overall system performance improves over time.

· In [30], Salganicoff and Bajcsy use Projection Pursuit Regression (PPR) to approximate the probability distribution of reinforcement in the state space, augmented by the action vector. The authors claim that the required number of training instances scales better with the state space dimensionality for the PPR technique than for most multivariate function approximation methods. Training consists of presenting patterns $< s, a, r >$, respectively state, action, and reinforcement. Once a good approximation is available, the associative function is queried with a partial pattern, consisting only of the state information, and a range of actions that maximizes the reinforcement is returned.

Storing and Retrieving Associations: An issue as important as how to associate reinforcement and actions is the issue of how to store and retrieve such associations. Neural networks are a natural choice, because of their capacity of smooth function approximation. However, they are not the only choice. In [30], a 2^k-tree is used to store the mapping from system state to expected reinforcement. In this scheme, the granularity of the representation is variable, depending on the local characteristics of the function being represented. Moore [27] has also used such scheme for its efficiency in representing multidimensional state spaces. In another attempt to overcome the dimensionality issue, Atkeson [2] proposed the use of locally weighted regression in memory-based robot learning. For each query, a local model is built based on the closest points in the state space: similar experiences influence the choice of the next action more than dissimilar ones. No model persists in memory: there is no global model and local models are formed specifically to answer a query.

Learning-derived composition techniques are very powerful and have wide applicability. Through learning, controllers can be constructed by composing atomic actions (Section 2) or sophisticated control algorithms. In our implementation, the policy on which controller to activate next is derived using Q-learning over a number of trials. No previous information is given about the utility of each controller; the composition scheme is completely random at first. By exploring the whole state-action space and rewarding "good" solutions/penalizing solutions corresponding to failures, this procedure incrementally computes the expected reward of taking each action for all states, effectively computing the activation policy that maximizes future rewards.

State Representation: In the first grasp controller implementation, grasp state is based on object geometry (see Section 2), not allowing for the encoding of generalizable control policies. Both knowledge-based and Q-learning implementations use a state representation based on the error metrics associated with the regulators: $S =< \epsilon_{FC}, \epsilon_{MC} >$, where ϵ_{FC} and ϵ_{MC} are computed respectively by Equations 5 and 11. It is our belief that this representation scheme offers better generalization than the pure geometric scheme, because it is based on general measures of task accomplishment: the controllers' error metrics.

The standard Q-learning implementation requires discretization of both state and action spaces. The range of each state variable ϵ_{FC} and ϵ_{MC} was discretized into 30 bins, resulting in a state space with 900 states. A logarithmic scale was used to help distinguishing states corresponding to low values of ϵ_{FC} and ϵ_{MC}; given that for most grasp configurations of interest the values are in the range $[0, 1]$, state S is encoded as:

$$\begin{aligned} S &= < \epsilon'_{FC}, \epsilon'_{MC} >, \\ \epsilon'_{FC} &= 14.5 \log(\max(0.01, \epsilon_{FC})/\ 0.01) \\ \epsilon'_{MC} &= 14.5 \log(\max(0.01, \epsilon_{MC})/\ 0.01) \end{aligned} \tag{14}$$

Thus, if $\epsilon_{FC} \leq 0.01$, then $\epsilon'_{FC} = 0$; if $\epsilon_{FC} = 1.0$, then $\epsilon'_{FC} = 29$; similarly for ϵ_{MC} and ϵ'_{MC}.

Action Space: In terms of action discretization, we decided to restrict the activation policy to a "bang-bang" type activation profile. Since the system composes 2 controllers, the combined gradient can be expressed as

$$\nabla\epsilon = \alpha_{FC}\frac{\partial\epsilon_{FC}}{\partial\theta} + \alpha_{MC}\frac{\partial\epsilon_{MC}}{\partial\theta},$$

where α_{FC} and α_{MC} are either 0 or 1 (but not simultaneously 0).

Trial Description: In each trial, four contacts are randomly placed on the object surface. The composite controller executes up to convergence, or for 300 control steps, whichever comes first. Convergence is achieved if every contact has not moved more than 0.0005 radians during the last control step. The limit on the number of steps was set to avoid possible instabilities on the control activation policy during learning. After convergence or 300 control steps, the resulting grasp configuration is scored using Equation 3.5.

Training: The system was trained over 500 epochs; each epoch consisted of 90 grasp trials on the 9 objects shown on Figure 6. The objects were presented in a random, fixed order for training. Ten grasp trials were executed for each object in the training sequence. A total of 45,000 grasp trials were attempted (5,000 trials per object).

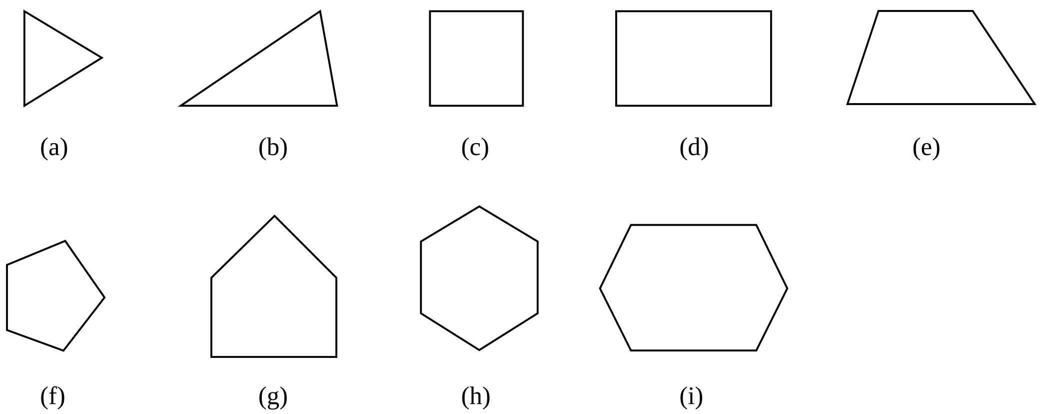


Figure 6: Objects used in our experiments: (a) Triangle, (b) Irregular Triangle, (c) Square, (d) Rectangle, (e) Trapezoid, (f) Pentagon, (g) Irregular Pentagon, (h) Hexagon, and (i) Irregular Hexagon.

Update equation:

$$Q(s_t, a_t) = (1-\alpha)Q(s_t, a_t) + \alpha(R + \gamma \max_{a\in A} Q(s_{t+1})),$$

where $\alpha = 0.02$ and $\gamma = 0.9$. After each control step, the Q-table was updated, using the update equation above with $R = 0$. Once the trial is finished, the Q-table is updated with $R = -10$ or $R = 10$, depending how the final grasp configuration was classified (respectively, as a failure or as a success).

Action selection: Actions are selected stochastically, according to a Boltzmann probability distribution over the Q-values associated with each action. The temperature varied in the range $[20.0, 0.005]$.

3.6.3 Results

Figure 7 is the plot of average failure rate as a function of the number of training epochs for the Q-learning implementation. This plot was smoothed by averaging the raw data over a 10-epoch wide sliding window.

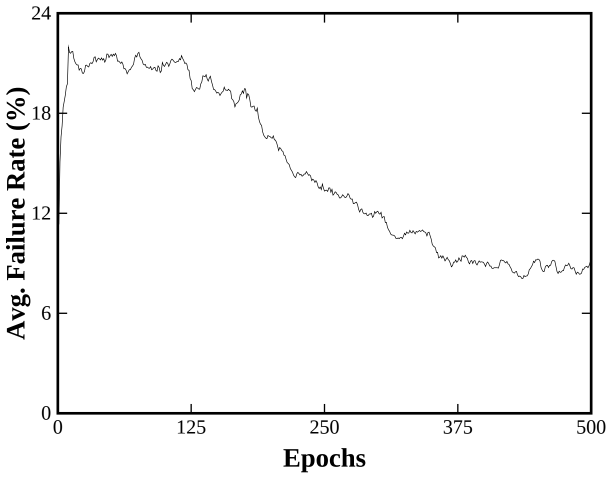


Figure 7: Average failure rate during training. Each epoch consists of 90 grasp trials, or 10 grasp trials per object. The plot was smoothed by averaging raw data over a 10-epoch wide sliding window.

The performance of both the Q-learning and the knowledge-based controllers was sampled over 200 grasp trials for all objects in the training set. As before, each trial would proceed until convergence or up to 300 control steps. Table 1 shows the average failure rate for both controllers.

Table 1: Average failure rate per object, for grasp controllers derived using Luce's rule and Q-learning.

	Failure Rate (%)	
Object	Knowledge-based composition	Q-learning composition
Triangle	17.00	14.50
Square	20.50	9.00
Pentagon	0.00	0.00
Hexagon	0.00	0.00
Irreg.Triangle	20.50	22.00
Rectangle	12.00	3.50
Trapezoid	2.00	2.50
Irreg.Pentagon	18.50	23.00
Irreg.Hexagon	0.20	0.50
Average	10.06	8.33

3.6.4 Discussion

Table 1 illustrates that in terms of on-line performance, neither implementation has a distinct advantage. An important result one can draw from Table 1 is the existence of a single, fixed composition policy applicable to all objects in the test set. It supports the notion of the declarative nature of the control specification and the suitability of the adopted state representation across the complete set of objects.

The Q-learning based controller has slightly better overall performance. Nevertheless, it is important to remember that the choice of constants used in the knowledge-based composition did not involve any exhaustive search over all possible switching functions. No memory is required for the knowledge-based grasp controller, that can be characterized as a computationally inexpensive, efficient grasping engine.

The Q-learning implementation explored the action space during training for all states and objects, and in principle it should yield an optimal strategy. Yet, some failures were observed; those failures can be attributed to:

1. Incomplete control repertoire: control composition can be viewed as function approximation based on a finite control basis. It is hard to guarantee that the current control repertoire is complete, or capable of generating solutions for any possible context.

2. Switching instabilities: limit cycles may arise from coupling effects between constituent controllers, negatively affecting overall performance.

3. Suboptimal composition functions can degrade system performance. Learning algorithms are powerful tools in deriving provably optimal control policies, but some of its working assumptions (e.g., completely observable states) may not hold for the current problem. Moreover, our goal is to develop a control strategy that generalizes well to varying object geometries, while Q-learning aims at reducing the average grasp error over a finite set of objects. Some performance deficit results in specific grasps when robustness is sought over a broad context range.

Regardless of the learning algorithm chosen, issues related to finite training sets, finite memory, or inherent limitations of the representation scheme chosen (neural nets, state-action tables) can affect performance. Therefore, the best approach is to substitute the current learning problem by an easier one, provided the final objective – control composition – is still accomplished.

4 Extending Control Composition through Pre-Imaging

Pre-imaging aims at enhancing an existing control policy through the opportune activation of metalevel control actions. Such metalevel control actions are activated whenever the current system state is associated to unsuccessful patterns of interaction between the existing predictable, asymptotically stable controller and the environment. Pre-imaging supports continual monitoring of the evolving system, and reactive compensation for deficient solutions.

The use of predictable control actions to reduce problem complexity bears resemblance to Lozano-Pérez [23] approach to the automatic synthesis of fine-motion strategies, in which the final goal was propagated backwards through the actions of the underlying generalized impedance controller, for a number of discrete steps. In each pre-imaging step, the set of goal states increased until it includes the start state. When this happens, there exists a sequence of control actions that achieves the goal. The approach organizes the fine motion plan around the predictable behavior of the generalized damper control module. Notice however that in their work no estimation is involved because complete knowledge about the geometry and uncertainties is assumed; also, the associated planning procedure is completely off-line.

4.1 The Pre-Imaging Concept

By assumption, the existing controller (or baseline controller) is predictable and stable; its success pre-image region is defined as the union of all states ultimately leading to successful solutions, according to the criterion established in Section 3.5. More formally, consider X as the state space over which the baseline controller C is defined. Given state $x \in X$, define $y = S_C(x)$ as the successor state of x due to the action of controller C. Let $r(x,t) = \{y : y = S_C^t(x)\}$ represent the set of reachable states from the initial state $x \in X$ after time t. The set of reachable states $R(t)$ where the system can be encountered at time t, assuming continuous control action since $t = 0$, can now be defined as the closure of $r(x,t)$ for all states $x \in X$. Clearly, $R(0) = X$: at time $t = 0$, the controller C has not acted upon the system, and any state $x \in X$ can be the initial state. With the passage of time, the equality between the sets is transformed into a containment relation $R(t) \subseteq X$. As $t \to \infty$, the system state converges to one attractor state $a_i = R(t_\infty)$, $a_i \in A$, where A is the set of system attractors. The attractor a_i may correspond to a satisfactory solution, in which case it is labeled as a success attractor, and all intermediate states en route to it are considered to be part of attractor a_i's success pre-image. Similarly, every state en route to a failure attractor a_j belongs to a_j's failure pre-image region.

Once a map relating state to failure expectation is available, it can be used to enhance the performance of the baseline controller: whenever failure is predicted, an alternative control action is to be applied to the system to revert failure expectation. This control action (referred to as the pre-image control action) complements the baseline controller, acting as a metalevel controller and can be viewed as a primitive mechanism for skill refinement.

4.2 Failure/Success Prediction and Control Composition

The map relating system state to failure expectation can be incrementally constructed over a number of trials with the use of standard supervised learning algorithms. Its construction encompasses learning to recognize those regions of the state space in which the baseline controller provides adequate action sequencing. Such learning problem is considerably easier than building a composition policy from scratch, as done by the Q-learning procedure (Section 3.6.2). Note that pre-imaging avoids the problem of searching in the universe of possible control policies (of exponential size on the number of control modules to compose); instead, pre-imaging constructs a success/failure prediction function. Rather than assigning credit for trial success to individual actions in each state (as the Q-learning procedure does), pre-imaging assigns success expectaction to regions of the state space.

Intuitively, learning to classify a given state as belonging to the success pre-image region is not a hard problem, because of (1) the topology of the region, and because (2) exact classification is not required. First, the convergent, asymptotically stable baseline control policy partitions the state space into domains of attraction corresponding to success and failure attractors, grouping together states in both success and failure pre-images; very few states will belong to the boundary between failure and success regions. Such grouping or clustering constrains the topology of the map to be learned and facilitates learning.

The second simplifying aspect in control pre-imaging is its robustness with respect to erroneous success/failure predictions. As long as no attractor state corresponding to a failure is classified as belonging to the success pre-image region, the system will eventually trigger the pre-image control action, which will presumably bring the system to the success pre-image region.

4.3 PI-Based Grasp Controller

Figure 8 depicts the block diagram for the grasp controller used in our experiments. The baseline controller used in the implementation was the knowledge-based controller, described in Section 3.6.1.

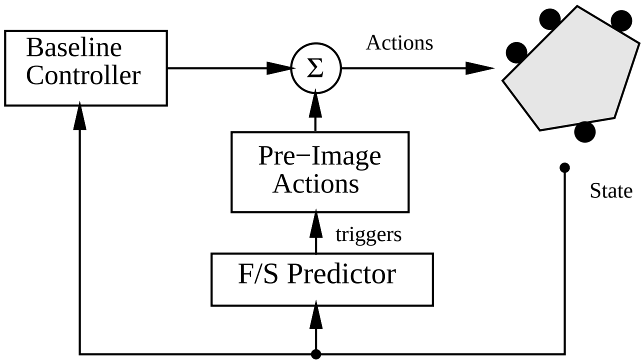


Figure 8: Grasp controller.

Failure/Success Predictor: The F/S predictor module computes success expectation, given the current system state. The Failure/Success (F/S) predictor was implemented as a neural network with two units for input of state information, three hidden units (one hidden layer) and one output unit (success expectation). Both the hidden and the output units were logistic units; the squashing function employed was the bipolar sigmoid function. Standard backpropagation was used to update the weights.

The F/S predictor was trained over 10 epochs, using the same scheme described for the Q-learning based grasp controller: each epoch consisted of 90 grasp trials, on 9 different objects. A total of 900 grasp trials were attempted, or 100 trials for each object.

In the beginning of each trial, four contacts were positioned randomly on the object's surface. From this point on, the controller would proceed to convergence or until the 300^{th} control step, whichever event comes first. Once the trial was over, the grasp solution was labeled as either failure or as success, and this label was attached to all states visited in that particular trial. At the end of each epoch, a fraction[5] of all visited states and respective labels was presented in random order as training instances to the F/S predictor module.

Pre-Image Actions: In the event of failure prediction, some corrective action must be taken. Conceptually, there are two alternatives: (1) corrective actions can be designed for the specific problem, or (2) corrective actions can be computed using the predictor module itself. The predictor module effectively maps state to expected reinforcement; if implemented by a neural network, the forward model constructed can be used to compute how expected reinforcement varies with respect to state variables. This technique has been used before [19, 22], to refine the performance of an existing controller.

Our implementation employs the first option: once failure is predicted, the pre-image action adds a zero mean, gaussian noise signal, bounded to $[-0.05, 0.05]$ radians (standard deviation $\sigma = 0.0167$), to the current control gradient. We shall see that this small perturbation works efficiently to divert the system towards states upstream of successful grasp configurations.

4.4 Results

Figure 9 is the plot of average failure rate as a function of the number of training epochs, for the PI-based grasp controller (plot (a)) and the knowledge-based controller (plot(b), averaged over 500 epochs). For comparison purposes, the corresponding plot for the Q-learning based controller and is also shown (plot (c)). In all cases, an epoch consists of 90 grasp trials, or 10 grasp trials for each object in the object set. The F/S predictor module is used following the 10^{th} epoch to activate the pre-image control action.

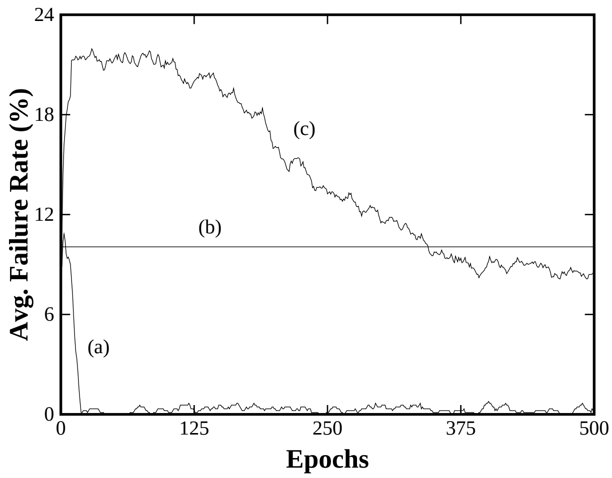


Figure 9: Average failure rate, for (a) PI-based controller, (b) knowledge-based controller, and (c) Q-learning based controller. Each epoch consists of 90 grasp trials, or 10 grasp trials per object. Plots are smoothed by averaging raw data over a 10-epoch wide sliding window.

Figure 10(a) displays a typical path in state space, for the knowledge-based controller during the four contact grasp of a square object. Starting from the contact configuration shown in (c), the controller drives the system from the initial state in the right upper corner (corresponding to high

[5]A number of training instances corresponding to success states was discarded such that both success and failure states would have roughly the same representation on the set of training instances.

force and moment closure errors) to a failure attractor (marked by an "X"). Figure 10(b) shows the corresponding path for the PI-based controller. Departing from the same initial configuration, the controller manages to converge to a success attractor (marked with "O"). Notice that object geometry affects the relation between ϵ_{FC} and ϵ_{MC}, affecting both the shape of the path in state space and the position and number of success and failure attractors.

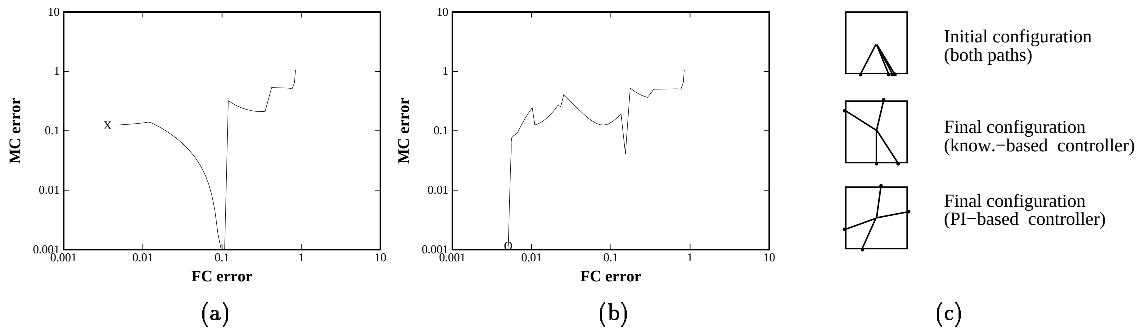


Figure 10: Paths for (a) knowledge-based controller (converging to failure attractor), and (b) for the PI-based controller (converging to success attractor); (c) initial and final configurations, for both paths.

Figure 11(a) shows the output of the F/S predictor module over the state space (higher values denote greater failure expectation). As shown, failure expectation is high for great values of ϵ_{MC}. Figure 11(b) shows the relative frequency each state in the state space is visited, for the knowledge-based controller attempting a four contact grasp of a square object. The two peaks correspond to the success and failure attractors; not surprisingly, the failure attractor is in the failure pre-image region. The final grasp configuration for both the failure and success attractors are also shown. Figure 11(c) shows the corresponding plot for the PI-based controller. Notice that only the peak corresponding to satisfactory solutions is present – no unsatisfactory solution was generated.

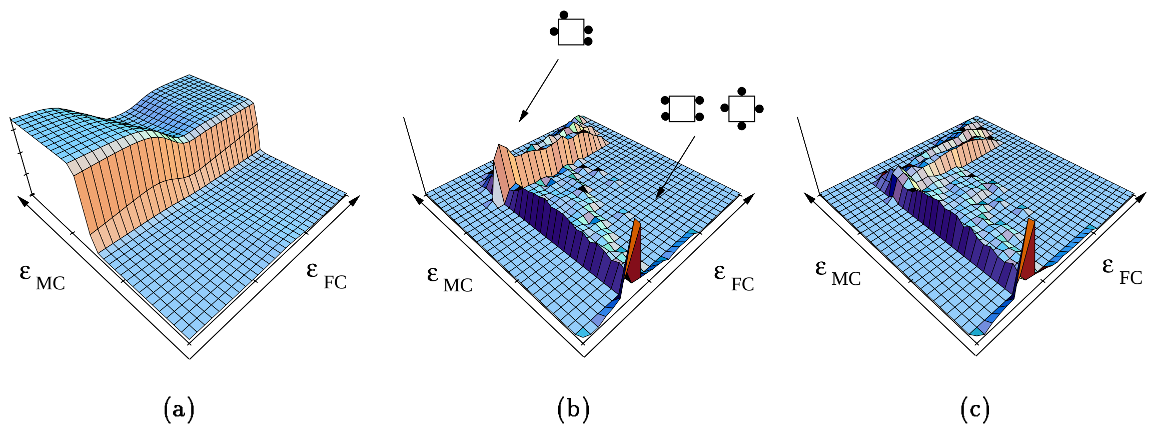


Figure 11: (a) Failure expectation, as a function of state. Relative frequency each state is visited for (b) knowledge-based controller and (c) PI-based controller. The grasp configurations corresponding to the failure and the success attractors are also shown in (b).

After the training stage, the performance of the knowledge-based, Q-learning, and PI-based controller

Table 2: Average failure rate per object, for different implementations and experiments.

Failure Rate (%)				
Object	Knowledge-Based Controller (a)	QL (b)	PI-Based Controller (c)	(d)
Triangle	17.00	14.50	0.00	1.00
Square	20.50	9.00	0.00	0.00
Pentagon	0.00	0.00	0.00	0.00
Hexagon	0.00	0.00	0.00	0.00
Irreg.Triangle	20.50	22.00	2.50	2.50
Rectangle	12.00	3.50	0.00	0.00
Trapezoid	2.00	2.50	0.50	0.00
Irreg.Pentagon	18.50	23.00	0.00	0.50
Irreg.Hexagon	0.20	0.50	0.00	0.00
Average	10.06	8.33	0.33	0.44

was assessed for the four contact null grasp task of planar objects. Several experiments are summarized in Table 2.

Knowledge-Based and Q-Learning Based Controller: For comparison, the results obtained for the knowledge-based controller and the Q-learning based controller in Section 3.6.3 are repeated in columns (a) and (b) of Table 2.

The F/S predictor module was trained under two different conditions: for the first condition, it was trained for a total of 900 grasps, or 100 grasp trials per object. All objects in the object set took part on the training stage. For the second condition, a total of 500 trials were performed on 5 objects (triangle, square, trapezoid, irregular pentagon, and irregular hexagon).

For both conditions, the resulting grasp controllers were subsequently tested over 200 grasp trials for each object in the object set. The corresponding average percent failure rates are listed under columns (c) and (d), respectively for training over all objects and training over 5 objects. In these experiments, each grasp trial would proceed for at most 300 control steps, or until convergence.

Grasp Configurations: Besides comparing failures rates, it is important to check the grasp configurations generated by each controller. Figure 12 displays all grasp configurations generated by different control composition schemes, namely knowledge-based (baseline controller), Q-learning derived composition, and the PI-based composition. For each grasp configuration, the corresponding grasp score (Equation 3.5) and the percentage of trials in which the configuration was generated is reported.

4.5 Discussion

The results in Table 2 show that the PI-based grasp controller performs remarkably well under the experimental conditions. All objects in the set can be grasped with fewer than 300 control steps, with exception of the irregular triangle and the trapezoid. Also, training over a reduced object set does not significantly affect system performance.

Control pre-imaging is efficient in terms of number of learning trials per object until convergence: it required 100 trials per object, compared to 5,000 for the Q-learning implementation. Such efficiency increase can be credited to the use of prior system expertise as a starting point, even if such

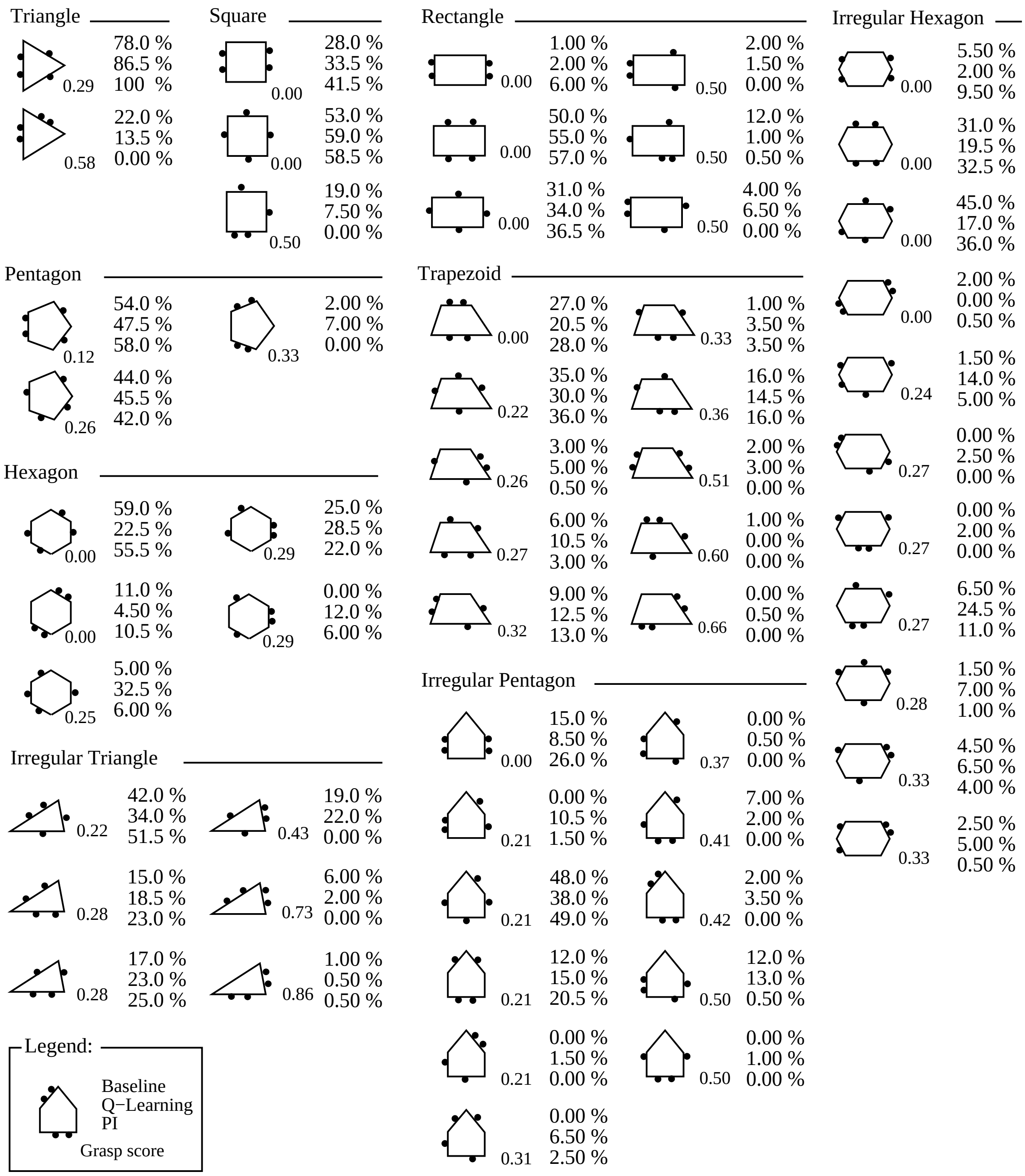


Figure 12: Configurations generated by different composition schemes. The number immediately on the right of the object is the corresponding grasp score; the percents to the right reflect how many times the particular grasp configuration was generated using the baseline (knowledge-based) composition, Q-learning derived composition, and composition through control pre-imaging, respectively from top to bottom.

existing system expertise is far from perfect. According to Figure 9, the performance delivered by the knowledge-based controller (in terms of average failure rate) is only matched by the Q-learning implementation after approximately 300 epochs. Also, it is important to emphasize that control pre-imaging involves a different learning problem – it identifies successful patterns of interaction between the existing controller and the object, rather than building a control policy from ground zero.

The results in Table 2(d) indicate that a very good composition policy can be derived even if a subset of the objects is used to train the F/S predictor module; the resulting composition policy has general applicability over this set of convex objects.

One can evaluate grasp controllers by computing their average score over all objects:

$$\overline{S_G} = \frac{1}{n} \sum_{1}^{n} \sum_{1}^{m} f \; S_G,$$

where n is the number of objects in the training set, m is the number of different grasp configurations for each object, f is the frequency the grasp configuration was generated, and S_G the corresponding grasp score. For the objects in the training set, the knowledge-based controller has a score of $\overline{S_G} = 0.172$; for the Q-learning controller, $\overline{S_G} = 0.189$. The PI controller overall score is $\overline{S_G} = 0.138$. Therefore, the knowledge-based controller produces grasp configurations with better scores in average than the Q-learning controller; however, the Q-learning controller fails less than the knowledge-based controller, according to our success/fail criteria.

Notice that pre-imaging was designed to generate successful configurations, not necessarily the optimal ones. Therefore, the overall score reflects the decision to label every grasp configuration with $S_G \leq 0.4$ as a success. It is plausible that by lowering the threshold (e.g., $S_G \leq 0.3$) a better overall score $\overline{S_G}$ could be obtained, at a cost of extended training and/or a more sophisticated F/S predictor module.

The results in Section 2 indicate that learning a control function using atomic actions as the building blocks may be implausible for our objectives; the results in this section suggest that control pre-imaging can provide an efficient alternative to learning from scratch a composition function for each individual controller. The unique aspect of our work is the synergy between control and learning: we have shown how learning augments an existing controller, and how the its control actions can be exploited to speed up the learning process.

5 Experimental Implementation

The grasp controller was used within the context of a pick-and-place task, where several control modules were integrated into a system capable of performing autonomous, collision-free reaching and grasping of wood blocks placed over a table at arbitrary positions.

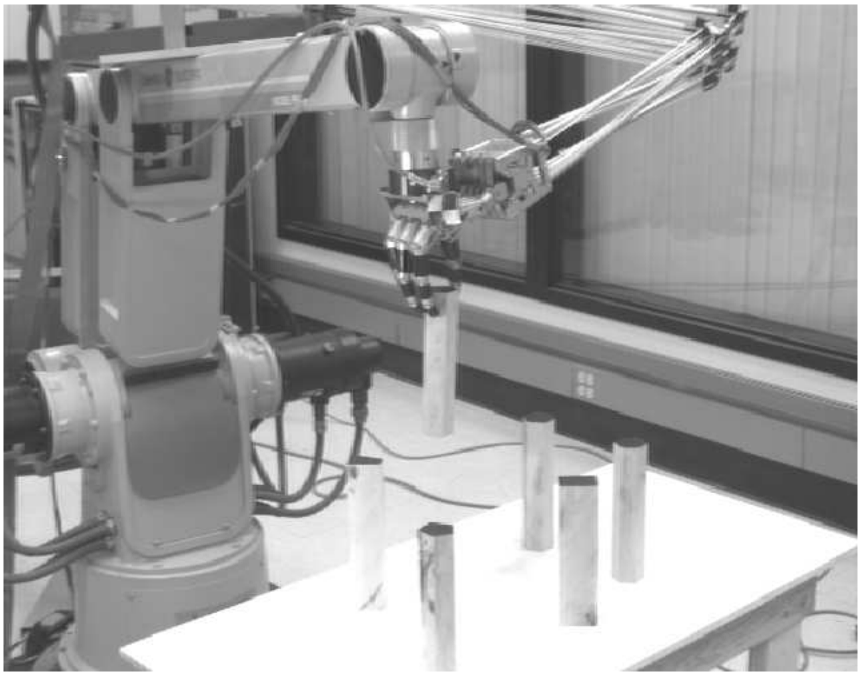


Figure 13: Experimental setup, showing the GE P50 robot arm and the Utah/MIT hand carrying an object with a top grasp.

5.1 Task Description

The task consisted of grasping one wood block, positioned among several others over a table, lifting and placing it at another point on the table. Both the object and the delivery point are specified by the user by clicking a mouse on the overhead image of the cluttered table. This is the only user intervention in the system that is otherwise completely autonomous.

Both the FC and MC controllers were developed based on the assumption that normal and position for each contact were available, ideally through tactile feedback. In this experiment, no tactile feedback was available; vision was used to derive contact normal and position information. Figure 13 shows the experimental setup. Grasps are performed by a Utah/MIT hand, attached to a GE P50 robot arm.

Objects: The wood blocks were approximately 20 cm in height, and with maximum distance between edges of 5 cm (for the rectangle). All objects were prisms with polygonal bases ranging from triangles to hexagons, in their regular and irregular versions.

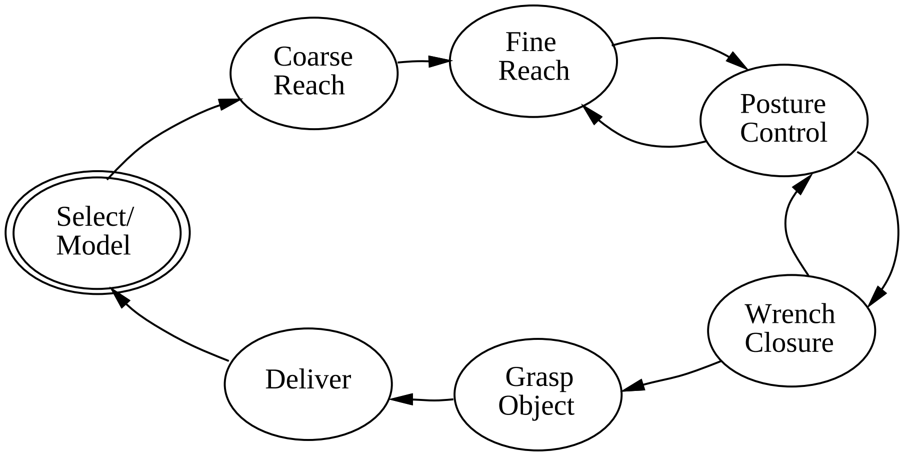


Figure 14: Finite state machine for controller sequencing.

5.2 Control Architecture

Several control modules were assembled together for this task, and these control modules were sequenced through a finite state machine, shown in Figure 14.[6] A complete description of every module is beyond the scope of this document: in what follows, we give a brief account of each control module:

Select/Model: In the beginning of the task cycle, the total number of objects and their approximate position on the table is determined by analyzing the overhead image of the table and objects. The user starts the cycle by selecting the object to be grasped; this selection is done directly on the overhead image, by clicking on the object image. The Hough edge detection algorithm is then used to extract a 2D object contour model, which is extruded in the z direction. The object contour is modelled as a polygon, and the vision algorithm provides with the object centroid, number of edges and overall shape.

Coarse Reach: Harmonic path computation [10] is used to compute a path from the current arm configuration to a configuration where the hand is close to the object. All objects are mapped as obstacles on a four-dimensional c-space, as well as the table and the robot itself. Harmonic functions are formal control surfaces with no local minima that permit multiple goals and that avoid all observed obstacles. It can be used to navigate the hand/arm through a cluttered environment as well as to navigate the hand/arm/object in this environment. The path computed determines whether a top or lateral grasp will be executed; the user does not specify the final approach strategy.

Fine Reach: Focus shifts from macroscopic collision avoidance to controlling the contact configuration between the hand and the object. Due to c-space discretization, the coarse reach will not necessarily converge with the hand in a position in which the reachability criterion is satisfied for all fingers. The fine reach module is another harmonic motion controller that postures the hand so as to minimize the distance between the fingertips and points projected over the object surface.

Posture Control: Typically, the fine reach module will position the fingers in a bad kinematic configuration. The posture control module (described in [16]) reconfigures the arm such as to maximize the hand manipulability, while maintaining the current fingertip positions. The combined effect of transitioning back and forth between the fine reach and posture control module is to approach the hand to the object, while keeping the fingers in a good kinematic posture.

[6] The finite state machine integrates the research effort of many individuals in the Laboratory for Perceptual Robotics at UMASS, namely C. Connolly, M. Huber, R. Grupen, K. Souccar, and the author.

Wrench Closure: Once all fingers can reach the object, the grasp controller is activated until convergence, or until the the reachability criterion is no longer satisfied. In the latter case, the system transitions back to the posture control module, adjusting the arm posture. In the wrench closure module, the reachability criterion was extended to mark as unreachable configurations in which finger links penetrates the object. After a number of transitions between both modules, the system reaches a convergent grasp configuration, in which the object is within reach of all fingers, and the arm is positioned such as to maximize the hand manipulability. The system transitions then to the grasp object module.

Grasp Object: This module drives the fingers from their approach configuration (fingers completely extended) to the virtual finger configuration synthesized by the grasp controller.

Deliver: At this point, the user is prompted to enter a delivery point to which the object should be moved. Again, this selection is done directly on the overhead image. The path is then computed by the same harmonic path planning algorithm used in the coarse reach module. Once the delivery is finished, the hand releases the object and the system returns to the Select/Model module, waiting for the next user command.

5.3 Experimental Results

It is extremely hard to characterize the performance of one of the component control modules (in our case, the grasp controller) in isolation. Overall, system performance was very robust, even in the presence of woes that plague every real world implementation: imprecise object models, discretization errors, calibration drifts, and kinematic model imprecision. Figure 15 shows a top grasp of an object with pentagonal base, and a lateral grasp of an object with rectangular base; both grasps are shown from different perspectives.

Over the course of a number of experiments, no structural problem (i.e., deficits in control partitioning) was detected. Failures would result after a sequence of mishaps, typically starting with a bad object model, followed by an imprecise grasp action, that ultimately would cause the object to wobble and fall after being delivered. Most of the time, imperfect models would not prevent the system from executing a secure grasp and delivery.

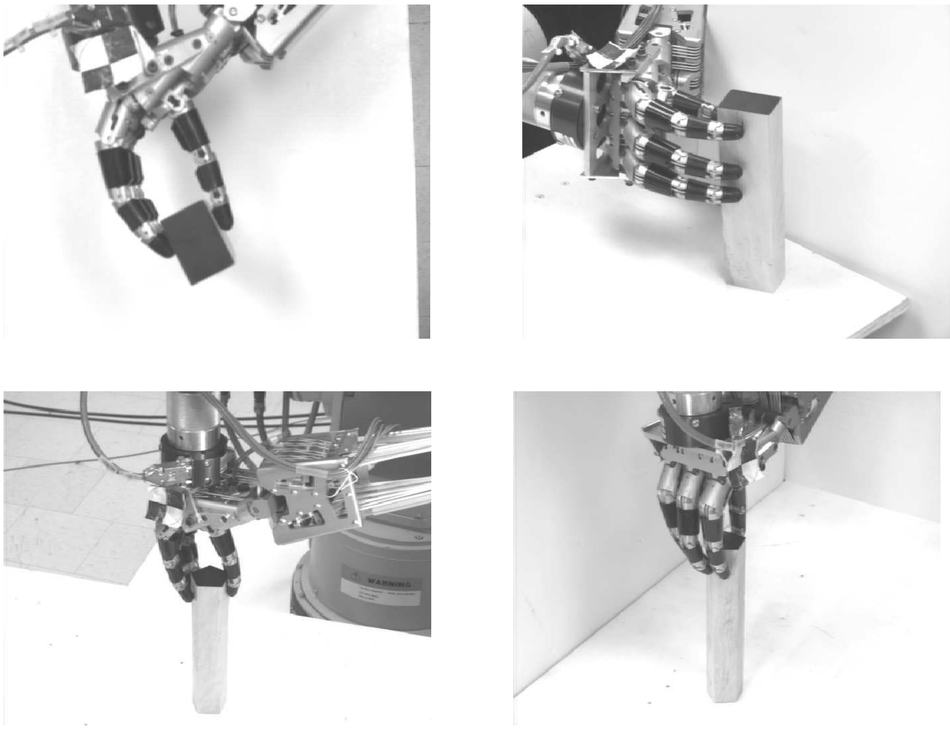


Figure 15: Grasps synthesized by system. Top row shows two perspectives of the lateral grasp of a rectangular prism, while bottom row shows the top grasp of a pentagonal prism.

6 Conclusion and Future Work

A very efficient grasp controller for general 3D shapes was tested empirically on convex objects. It is able to synthesize grasp configurations through successive, locally optimal configuration refinements, given position and normal of each contact. The grasp controller itself consists of a set of independent controllers (one for each contact) of complexity $O(n)$ on the number of contacts. Despite its inexpensive implementation and minimal requirements in terms of sensory information, it manages to generate stable grasp configurations for all objects in the training set within a bounded number of iterations.

The existence of formally derived, highly competent grasp controllers is what distinguishes our approach from other grasp synthesis approaches. These controllers provide for an operational state space representation for grasping, structure and simplify control composition, and ultimately enhance generalization across the problem domain. Furthermore, as demonstrated in Appendix A, these controllers make possible the use of optimal control techniques in problem domains traditionally out of their scope (e.g., grasp synthesis). For the first time, optimal control techniques were demonstrated in the grasp synthesis problem, establishing a connection to a large body of literature and methods, yet to be fully reviewed.

This whole thesis was developed to address deficits in composition-based control architectures, providing context sensitivity and incrementally extending controller competency. By identifying regions of the state space in which the grasp controller is prone to fail, the PI technique links the state space and controller, simplifying the partitioning of the state space and the composition problem. It also provides a primitive skill acquisition mechanism for incremental incorporation of new controllers to existing control structures. The method is efficient in terms of memory and number of training instances required. Overall, the results obtained suggest that predictable, efficient control structures for complex systems can be constructed from formally designed control modules.

Future Work: There is more to grasp synthesis than placing the contacts at the right position on the object's surface. The proposed contact configuration may be rendered infeasible by manipulator kinematics, the required forces over the object for stable grasp may not be deliverable by the manipulator, and there are also issues related to dynamic characteristics of the grasp. A nice way to verify how our approach scales with the number of controllers would be to augment the composite controller with an extra controller to address the manipulator prerogatives in the grasp synthesis process.

Another important extension is to build a grasp controller for both convex and concave objects. It appears possible to extend the current framework by developing a grasp controller that incorporates local curvature information about the object surface. Curvature can be estimated locally either by vision algorithms or contact estimators, but these possibilities are still to be explored.

A Control Composition as an Optimal Control Problem

All alternatives to the control composition problem discussed in this paper assumed that only partial information about the object is available: the moment closure controller just requires position and normal of each contact, and the force closure controller requires only contact positions. Although this is probably the most accurate scenario for control of real-world complex systems, one can do better if complete or very good models about the problem is available, using optimal control techniques.

In this appendix, the composition problem is solved optimally, with respect to time. The objective is to demonstrate once again the utility of the grasp controllers discussed previously – because such controllers are available, the optimal control problem can be formulated. Also, portraying the grasping of an object as an optimal control problem serves to illustrate yet another solution strategy to the grasp synthesis problem.

A.1 Problem Description

In order to keep the presentation clear, a small and somewhat constrained version of the grasp synthesis problem was chosen to illustrate our point. There is no impediment to using the technique discussed below to the unrestricted grasping problem, involving grasps with n moving contacts. For the general control composition problem, the only requirement is differentiability of the metric being optimized and the constituent controllers. See [7] for a introduction to Optimal Control Theory.

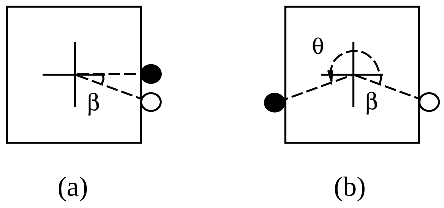


Figure 16: (a) Initial configuration (b) Final configuration.

The problem consists in finding out how to change the activation of both the force and moment closure controllers, such as to minimize the number of control steps required to take the system from the initial to the final configuration (see Figure 16). For simplicity, we will consider that just one contact moves over the object surface, while the other stays fixed at its initial position. Also, we will assume that the activation for the moment closure controller is constrained to be $(1-\alpha)$, where $\alpha \in [0,1]$ is the activation for the force closure controller.

A.2 Problem Formulation

Maximize

$$V = \int_0^T -\epsilon_{WC}(\theta)\, dt$$

Subject to

$$
\begin{aligned}
\dot{\theta}(t) &= -k\left(\alpha \frac{d\,\epsilon_{FC}(\theta)}{d\,\theta} + (1-\alpha)\frac{d\,\epsilon_{WC}(\theta)}{d\,\theta}\right) \\
\theta(0) &= 0.0, \quad \theta(T)\ unknown \\
\alpha &\in [0,1]
\end{aligned}
$$

In the formulation above, ϵ_{WC} is the wrench closure error (Equation 11). If the task is to achieve null grasp of a square with two contacts then

$$\epsilon_{WC}(\theta) = E + \frac{(2\tan\theta - 1)^2}{16},$$

where E is a constant corresponding to the force error incurred in positioning the moving contact in the various faces of the square. For the face corresponding to $\theta = 0$, $E = 1$; for faces corresponding to $\theta = \pi/2$ and $\theta = 3\pi/2$, $E = 0.5$, and for the face corresponding to $\theta = \pi$, $E = 0$.

Similarly, ϵ_{FC} is the force closure error, expressed as

$$\epsilon_{FC}(\theta) = 0.5 * (1 + \cos(\theta - \beta)),$$

being β the angle for the fixed contact. In the development that follows, the first and second derivatives of ϵ_{WC} and ϵ_{FC} with respect to θ will be required:

$$
\begin{aligned}
\frac{d\,\epsilon_{WC}}{d\,\theta} &= \frac{2\tan\theta - 1}{4\cos^2\theta} \\
\frac{d^2\epsilon_{WC}}{d\,\theta^2} &= \frac{8 + 2(\tan\theta - 1)\sin 2\theta}{16\cos^4\theta} \\
\frac{d\,\epsilon_{FC}}{d\,\theta} &= -0.5\sin(\theta - \beta) \\
\frac{d^2\epsilon_{FC}}{d\,\theta^2} &= -0.5\cos(\theta - \beta)
\end{aligned}
$$

The Hamiltonian function for the problem can now be expressed as

$$H(t,\theta,\alpha,\lambda) = -\epsilon_{WC(\theta)} + \lambda(t)\frac{d\,\theta}{d\,t}.$$

The costate variable λ is an auxiliary variable required by the method [7]. The Maximum Principle conditions are

$$H(t,\theta,\alpha^*,\lambda) \geq H(t,\theta,\alpha,\lambda)\ \ \forall t \in [0,T] \tag{15}$$

$$
\begin{aligned}
\dot{\theta}(t) &= \frac{\partial H}{\partial \lambda} \\
&= -k\left(\alpha \frac{d\,\epsilon_{FC}(\theta)}{d\,\theta} + (1-\alpha)\frac{d\,\epsilon_{WC}(\theta)}{d\,\theta}\right) &(16)\\
\dot{\lambda}(t) &= -\frac{\partial H}{\partial \theta} = \frac{d\,\epsilon_{WC}}{d\,\theta} \\
&+ \lambda(\alpha \frac{d^2\epsilon_{FC}}{d\,\theta^2} + (1-\alpha)\frac{d^2\epsilon_{WC}}{d\,\theta^2}) &(17)\\
\lambda(T) &= 0
\end{aligned}
$$

Equation 15 means that the resulting Hamiltonian H is to be maximized over time, and $\alpha^*(t)$ is the activation policy that maximizes it. Notice that selecting α such that $\frac{\partial H}{\partial \alpha} = 0$ does not cover all cases, because of the constrained range of values α can assume — it might be the case that α^* corresponds to one value in the limits of the closed interval $[0, 1]$, for example. And this is the case for the problem we are interested in, because $\frac{\partial H}{\partial \alpha}$ does not depend on α:

$$\frac{\partial H}{\partial \alpha} = -\lambda\left(\frac{d\,\epsilon_{FC}}{d\,\theta} - \frac{d\,\epsilon_{WC}}{d\,\theta}\right).$$

Therefore, the time optimal policy for this particular problem implies switching α from 0 to 1 and back. This optimal pattern can be easily computed by integrating Equations 16 and 17 above over time:

1. Initialize λ;
2. Compute $\frac{\partial H}{\partial \alpha}$; if $\frac{\partial H}{\partial \alpha} > 0$, select $\alpha = 1$; otherwise, make $\alpha = 0$;
3. Compute $\theta_{t+1} = \dot{\theta}_t \Delta t$;
4. Compute $\lambda_{t+1} = \dot{\lambda}_t \Delta t$;
5. Check for convergence on θ; return to (ii) above if not converged.

A.3 Results

In all results, we used $\lambda(0) = 5$, and $\Delta t = 0.01$. The method is not sensitive to the values chosen for these parameters; an ample range of values can be used. However, selecting a low initial value for λ may result in solutions corresponding to local minima. The fixed contact is stationed at $\beta = -0.46365$ radians.

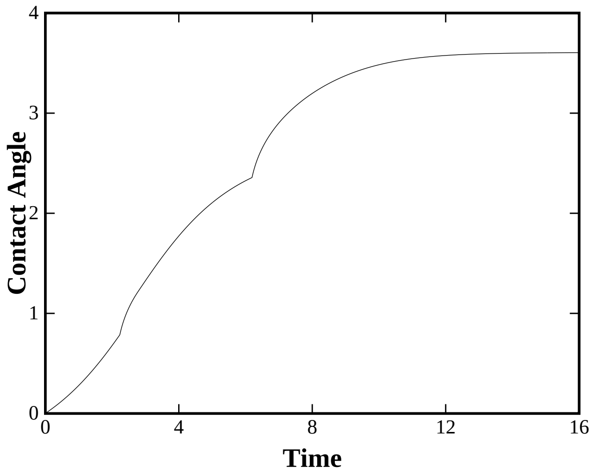


Figure 17: Evolution of θ with time.

Figure 17 shows how θ varies with time. Starting from $\theta(0) = 0.0$, the contact progresses steadily to the optimal solution $\theta(T) = 3.6052$ radians. Figure 18 shows the optimal activation pattern for α as a function of time. Figure 18 also displays the corresponding areas on the object surface where the force closure controller was selected; for the white areas, $\alpha = 1$. Gray areas correspond to where in the object the moment closure controller was selected ($\alpha = 0$).

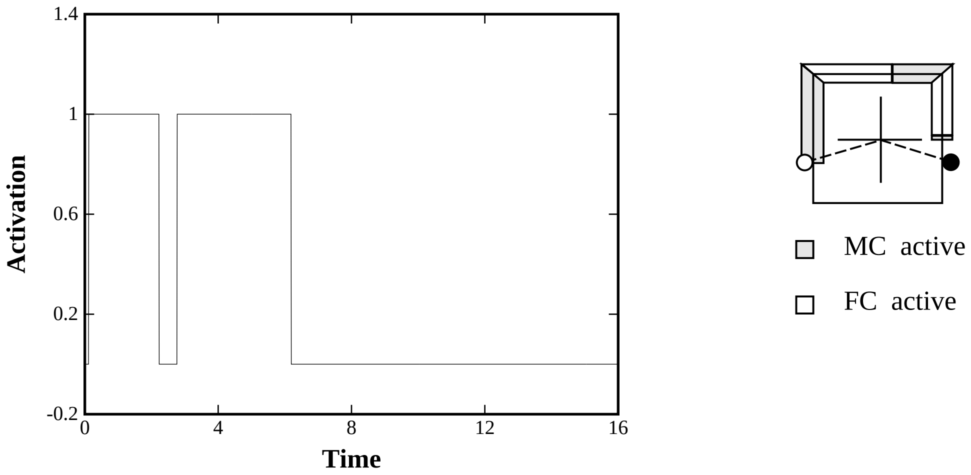


Figure 18: Time optimal activation pattern during the grasp. $\alpha = 1$ corresponds to full activation of the force closure controller, while $\alpha = 0$ corresponds to full activation of the moment closure controller. In the inset, gray areas correspond to where in the object's surface $\alpha = 0$; white areas mark the areas where $\alpha = 1$.

A.4 Discussion

Notice that, as long as the metric chosen to be optimized doesn't depend on the intermediate activations $\alpha(t)$, the time optimal activation policy will always be a *bang-bang* type solution, no matter the object being grasped.

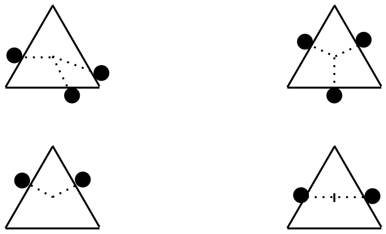
Another interesting aspect is the fact that this approach also yields an effective grasp controller: just by evaluating $\frac{\partial H}{\partial \alpha}$ one can decide on the next activation α. What remains to be determined is how robust is this controller with respect to object geometries, and uncertainties on actuation and sensory information. Because this approach essentially involves integration over time, it is not clear how to revise the control history incrementally: presumably, a object model is being continuously or periodically updated, and sensory data may yield discontinuities in the state information. How to accommodate such changes in the control scheme is yet to be determined.

B Knowledge-based Controller Characterization

The Luce's rule formulation (see Section 3.6.1) used in the implementation of the knowledge-based controller is an approximation to the following discrete composition rule

$$
\begin{array}{l}
\texttt{if } (\epsilon_{FC} > \tau) \\
\qquad \alpha_{FC} \texttt{ = 1.0} \\
\qquad \alpha_{MC} \texttt{ = 0.0} \\
\texttt{else} \\
\qquad \alpha_{FC} \texttt{ = 0.0} \\
\qquad \alpha_{MC} \texttt{ = 1.0,}
\end{array}
$$

where τ is a threshold set to an arbitrarily small value. The decision of setting τ to a small value reflects a domain-derived heuristic, namely to give precedence to FC controller over the MC controller. As $\tau \to 0$, the only convergent solutions allowed are those corresponding to grasp configurations in which both ϵ_{FC} and ϵ_{MC} are minimized by the current contact configuration. Notice that the decision rule adopted constrains the set of possible solutions. Figure 19(a) shows grasp solutions that are not generated due to the heuristic nature of the composition rule adopted; Figure 19(b) shows the equivalent configurations generated instead.



(a) (b)

Figure 19: (a) Grasp solutions excluded due to the composition rule adopted. (b) Corresponding solutions generated by the knowledge-based controller.

Formally, the universe of possible solutions corresponds to the solutions of the following system of non-linear equations:

$$
\left\{
\begin{array}{lcl}
t_{fx} + \frac{1}{n}\sum_{i=1}^{n} cos(\theta_i)cos(\phi_i) & = & 0 \\
t_{fy} + \frac{1}{n}\sum_{i=1}^{n} sin(\theta_i)cos(\phi_i) & = & 0 \\
t_{fz} + \frac{1}{n}\sum_{i=1}^{n} sin(\phi_i) & = & 0 \\
t_{mx} - \frac{1}{n}\sum_{i=1}^{n} m_{xi} & = & 0 \\
t_{my} - \frac{1}{n}\sum_{i=1}^{n} m_{yi} & = & 0 \\
t_{mz} - \frac{1}{n}\sum_{i=1}^{n} m_{zi} & = & 0
\end{array}
\right. \tag{18}
$$

The number of contacts n, task vector $\vec{t}$, and the particular object geometry determine the number of solutions, ranging from no solution to infinite solutions. In most situations, the system exhibits a great deal of redundancy, admitting an infinite number of solutions. However, it is hard to show the existence of solutions without duly specification of the task vector, the class of objects, and the number of contacts n.

B.1 Convergence Properties

Assuming $\vec{t} = 0$ and the class of regular polygons, it is possible to show that knowledge-based controller always converges to the optimal solution, for 2 contacts.

Theorem B.1 *For the class of regular polygons and $n = 2$, the knowledge-based controller will always converge to a solution that globally minimizes ϵ_{MC}, as expressed in Equation 11.*

Proof: The proof consists in showing that (a) the system initially converges to a FC solution, and then (b) it converges to a MC solution while keeping $\epsilon_{FC} = 0$. The generated solution minimizes globally Equation 11 because no moment residuals are present, while the force residuals of two-fingered grasps cannot be eliminated for polygons with an odd number of edges.

Part (a) follows from the properties of the modified FC controller, that is convex and unimodal over the whole domain by construction (as expressed by Equation 7); part (b) is proved in Lemma .1.

Lemma .1 *Following convergence to a FC solution, the knowledge-based controller will always converge to a MC solution while keeping $\epsilon_{FC} = 0$.*

Proof: For the conditions stated in Theorem B.1, Equations 18 are reduced to

$$\left\{ \begin{array}{lcl} \sum_{i=0}^{1} cos(\theta_i) & = & 0 \\ \sum_{i=0}^{1} sin(\theta_i) & = & 0 \\ & & \\ \sum_{i=0}^{1} m_{zi} & = & 0. \end{array} \right. \tag{19}$$

The first two equations are satisfied for $\theta_1 = \theta_0 + \pi$, as can be readily verified.

For convenience, the edges of the regular polygon are numbered from 0 to $n-1$, in the counterclockwise direction. Without loss of generality, it is assumed that contact 0 is positioned in edge 0, following convergence of the FC controller. Given that $\theta_1 = \theta_0 + \pi$, contact 1 is positioned in edge $\lfloor \frac{n(\theta_0+\pi)}{2\pi} \rfloor$. The respective moments for contacts 0 and 1 are

$$\left\{ \begin{array}{lcl} m_{z0} & = & r\tan(\theta_0 - \theta_{n0}) \\ m_{z1} & = & r\tan(\theta_1 - \theta_{n1}), \end{array} \right.$$

where θ_{n0} and θ_{n1} are the normal angles for the convergent edges of contacts 0 and 1. From Figure 20, it is easy to see that $\theta_{n0} = \frac{\pi}{n}$, and $\theta_{n1} = \frac{\pi}{n} + \frac{2\pi}{n} \lfloor \frac{n(\theta_0+\pi)}{2\pi} \rfloor$.

In order to show convergence to a MC solution, it is sufficient to evoke the convexity of the MC controller, and prove that there exist a solution θ_0^* such that $\theta_0^* \in (0, \dots, 2\pi/n)$ (i.e., the MC controller will drive the contacts to a stable configuration without ever causing the contacts to change edges). Such a solution is achieved when

$$(\theta_0^*,\, \theta_1^*) = \left(\frac{\theta_{n1}}{2} + \frac{\pi}{2n} - \frac{\pi}{2},\, \frac{\theta_{n1}}{2} + \frac{\pi}{2n} + \frac{\pi}{2} \right) \tag{20}$$

After verifying that (θ_0^*, θ_1^*) is a solution for Equations 19, one must show that $\theta_0^* \in (0, \dots, 2\pi/n)$.

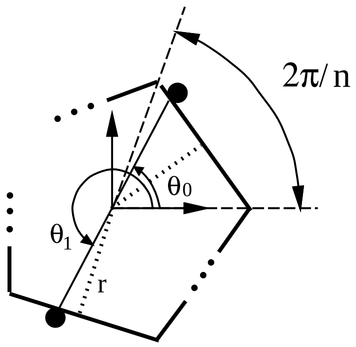


Figure 20: Generic regular polygon.

Lemma .2 $\theta_0^* \in (0, \ldots, 2\pi/n)$.

Proof: Observe that $\theta_{n1} \in [\pi, \ldots, \frac{2\pi}{n}\lceil\frac{n}{2}\rceil + \frac{\pi}{n}],\ \forall\ \theta_0 \in [0, \ldots, 2\pi/n],\ n \geq 3$. Substituting the lower bound of θ_{n1} in Equation 20:

$$\begin{aligned} \theta_0^* &= \frac{\theta_{n1}}{2} + \frac{\pi}{2n} - \frac{\pi}{2} \geq \frac{\pi}{2} + \frac{\pi}{2n} - \frac{\pi}{2} \\ \theta_0^* &\geq \frac{\pi}{2n} > 0. \end{aligned}$$

Similarly, replacing θ_{n1} by its upper bound yields an upper bound for θ_0^* :

$$\begin{aligned} \theta_0^* &= \frac{\theta_{n1}}{2} + \frac{\pi}{2n} - \frac{\pi}{2} \leq \frac{\pi}{n}\lceil\frac{n}{2}\rceil + \frac{\pi}{2n} + \frac{\pi}{2n} - \frac{\pi}{2} \\ &= \frac{\pi}{n}\lceil\frac{n}{2}\rceil + \frac{\pi}{n} - \frac{\pi}{2}. \end{aligned}$$

For n even:

$$\begin{aligned} \theta_0^* &\leq \frac{\pi}{n}\lceil\frac{n}{2}\rceil + \frac{\pi}{n} - \frac{\pi}{2} = \frac{\pi}{n}\frac{n}{2} + \frac{\pi}{n} - \frac{\pi}{2} \\ \theta_0^* &\leq \frac{\pi}{n}. \end{aligned}$$

For n odd:

$$\begin{aligned} \theta_0^* &\leq \frac{\pi}{n}\lceil\frac{n}{2}\rceil + \frac{\pi}{n} - \frac{\pi}{2} = \frac{\pi}{n}\left(\lfloor\frac{n}{2}\rfloor + 1\right) + \frac{\pi}{n} - \frac{\pi}{2} \\ &= \frac{2\pi}{n} + \frac{\pi}{n}\lfloor\frac{n}{2}\rfloor - \frac{\pi}{2} = \frac{2\pi}{n} + \frac{\pi}{n}\frac{(n-1)}{2} - \frac{\pi}{2} \\ \theta_0^* &\leq \frac{2\pi}{n} - \frac{\pi}{2n} < \frac{2\pi}{n}\ \square. \end{aligned}$$

To conclude the proof of Lemma .1 and Theorem B.1, it remains to show that the actions of the MC controller preserve $\epsilon_{FC} = 0$, as contacts move from (θ_0, θ_1) to (θ_0^*, θ_1^*). It suffices to show that the MC controller moves both contacts by the same amount, therefore preserving the relation $\theta_1 = \theta_0 + \pi$.

Lemma .3 *Both contacts in any two-fingered grasp configuration over a regular polygon have the same MC gradient, for the null grasp task.*

Proof: Equation 12 computes the MC gradient for both contacts. The difference $(\epsilon_{MC} - \epsilon^*_{MC})$ between the error associated with the current grasp configuration and the error due to force residuals is the same for both contacts. Thus, the proof resides in showing $(\theta_0 - \theta^*_0) = (\theta_1 - \theta^*_1)$.

As it turns out, $\theta^*_0 = \theta_1 + \sum_{i=0}^{1} \theta_{ni}$, and $\theta^*_1 = \theta_0 + \sum_{i=0}^{1} \theta_{ni}$; therefore, $(\theta_0 - \theta^*_0) = (\theta_1 - \theta^*_1) = \sum_{i=0}^{1}(\theta_i - \theta_{ni})$ $\Box$.

Notice that Lemma .3 does not hold for the non-convex version of the MC controller; however, a modified version of the non-convex controller, in which both contacts are moved according to the minimum MC gradient of all contacts, would still preserve the relation $\theta_1 = \theta_0 + \pi$ and the validity of Theorem B.1.

The approach above can be extended to prove the completeness of the knowledge-based controller for three contact grasps of regular polygons. However, as the number of contacts grows, the family of force closure solutions expands, increasing the complexity of completeness proofs. No counter example to demonstrate that the knowledge-based controller is not complete has been found, for the null grasp task involving a variable number of contacts; a completeness proof for an arbitrary number of contacts remains as future work.

References

[1] Agre, P., and Chapman, D. What are plans for? In *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back* (Cambridge, MA, 1991), The MIT Press, pp. 17–34.

[2] Atkenson, C. Using locally weighted regression for robot learning. In *Proc. 1991 IEEE Int. Conf. Robotics Automat.* (Sacramento, CA, May 1991), vol. 2, pp. 958–963.

[3] Barto, A. Connectionist learning for control: An overview. Tech. Rep. COINS Technical Report 89-89, COINS Department, University of Massachusetts, Sept. 1989.

[4] Barto, A., Sutton, R., and Anderson, C. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Syst., Man, Cybern. SMC-13*, 5 (Sept./Oct. 1983), 835–846.

[5] Brooks, R. A robust layered control system for a mobile robot. *IEEE J. Robotic Automat. 2*, 1 (Mar. 1986), 14–23.

[6] Chen, I., and Burdick, J. Finding antipodal point grasps on irregularly shaped objects. In *Proc. 1992 IEEE Int. Conf. Robotics Automat.* (Nice, FRANCE, May 1992), vol. 3, pp. 2278–2283.

[7] Chiang, A. *Elements of Dynamic Optimization.* McGraw-Hill, Inc., New York, NY, 1992.

[8] Clark, R., Arkin, R., and Ram, A. Learning momentum: On-line performance enhancement for reactive systems. In *Proc. 1992 IEEE Int. Conf. Robotics Automat.* (Nice, FRANCE, May 1992), vol. 1, pp. 111–116.

[9] Connolly, C., and Grupen, R. Harmonic control. In *Proc. 1992 Int. Symp. Intelligent Control* (Glasgow, SCOTLAND, Aug. 1992), pp. 503–506.

[10] Connolly, C., and Grupen, R. The applications of harmonic functions to robotics. *Journal of Robotic Systems 10*, 7 (Oct. 1993), 931–946.

[11] Cutkosky, M. On grasp choice, grasp models, and the design of hands for manufacturing tasks. *IEEE Trans. Robotics Automat. 5*, 3 (June 1989).

[12] Faverjon, B., and Ponce, J. On computing two-finger force-closure graps of curved 2d objects. In *Proc. 1991 IEEE Int. Conf. Robotics Automat.* (Sacramento, CA, May 1991), vol. 1, pp. 424–429.

[13] Ferrari, C., and Canny, J. Planning optimal grasps. In *Proc. 1992 IEEE Int. Conf. Robotics Automat.* (Nice, FRANCE, May 1992), vol. 3, pp. 2290–2295.

[14] Grupen, R., and Coelho Jr, J. Sensor-based contact geometry optimization for multifingered robot hands. In *Proc. of Int. Workshop on Mechatronical Computer Systems for Perception and Action* (Halmstad, SWEDEN, June 1993), pp. 147–156.

[15] Grupen, R., Coelho Jr, J., and Souccar, K. On-line grasp estimator: A partioned state space approach. Tech. Rep. COINS Technical Report 92-75, COINS Department, University of Massachusetts, Oct. 1992.

[16] Grupen, R., and Souccar, K. Manipulability-based spatial isotropy: A kinematic reflex. In *Proc. of Int. Workshop Mechatronical Comp. Systems for Perception and Action* (Halmstad, SWEDEN, June 1993), pp. 157–163.

[17] Guo, G., Gruver, W., and Jin, K. Grasp planning for multifingered robot hands. In *Proc. 1992 IEEE Int. Conf. Robotics Automat.* (Nice, FRANCE, May 1992), vol. 3, pp. 2284–2289.

[18] Jameson, J., and Leifer, L. Automatic grasping: An optimization approach. *IEEE Trans. Syst., Man, Cybern. SMC-17*, 5 (Sept./Oct. 1987).

[19] Jordan, M., and Rumelhart, D. Internal world models and supervised learning. In *Machine Learning: Proc. 8^{th} Int. Workshop* (1991), L. Birnhaum and G. Collins, Eds.

[20] Koditschek, D., and Rimon, E. Robot navigation functions on manifolds with boundary. *Advances in Applied Mathematics 11*, 4 (1990), 412–442.

[21] Kweon, I., Kuno, Y., Watanabe, M., and Onoguchi, K. Behavior-based mobile robot using active sensor fusion. In *Proc. 1992 IEEE Int. Conf. Robotics Automat.* (Nice, FRANCE, May 1992), vol. 2, pp. 1675–1682.

[22] Liu, S., and Asada, H. Teaching and learning of deburring robots using neural networks. In *Proc. 1993 IEEE Int. Conf. Robotics Automat.* (Atlanta, GA, May 1993), vol. 3, pp. 339–345.

[23] Lozano-Pérez, T., Mason, M., and Taylor, R. Automatic synthesis of fine-motion strategies for robots. *The Int. J. Robotics Res. 3*, 1 (Spring 1984).

[24] Maes, P. Learning to coordinate behaviors. In *Proc. of 8^{th} National Conf. Artificial Intelligence* (Boston, MA, 1990), pp. 796–802.

[25] Maes, P. Situated agents can have goals. In *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back* (Cambridge, MA, 1991), The MIT Press, pp. 49–70.

[26] Mataric, M. Minimizing complexity in controlling a mobile robot population. In *Proc. 1992 IEEE Int. Conf. Robotics Automat.* (Nice, FRANCE, May 1992), vol. 1, pp. 830–835.

[27] Moore, A. Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces. In *Proc. 8^{th} Int. Workshop Machine Learning* (1991), Morgan Kauffman.

[28] Nguyen, T., and Stephanou, H. A topological algorithm for continuous grasp planning. In *Proc. 1990 IEEE Int. Conf. Robotics Automat.* (Cincinnati, OH, May 1990), vol. 1, pp. 670–675.

[29] Nguyen, V. Constructing stable grasps. *The Int. J. Robotics Res. 8*, 1 (1989), 26–37.

[30] Salganicoff, M., and Bajcsy, R. Robotic sensorimotor learning in continuous domains. In *Proc. 1992 IEEE Int. Conf. Robotics Automat.* (Nice, FRANCE, May 1992), vol. 3, pp. 2045–2050.

[31] Watkins, C. J. C. H. *Learning from Delayed Rewards.* PhD thesis, Cambridge University, Cambridge, England, 1989.