UNIVERSITY OF CALIFORNIA, SAN DIEGO

Learning to Perceive: A Developmental Robotics Approach to Vision and Object Interaction

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Cognitive Science

by

Walter A. Talbott

Committee in charge:

Virginia de Sa, Chair
Gedeon Deák
Javier Movellan
Ayse Saygin
Terrence Sejnowski
Mohan Trivedi

2015

The dissertation of Walter A. Talbott is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2015

DEDICATION

To patience, understanding, and those who have displayed these qualities beyond reasonable expectation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Thanks are primarily due to my advisor Javier Movellan. He has set an admirable example of how to approach difficult research topics, clearly explain difficult concepts, and have the tenacity to pursue a project in the face of multiple setbacks. I have learned a great deal from him, and this thesis would not have been possible without his help, encouragement, and patience.

I'd like to thank Crane Huang for listening to my ideas, even the bad ones, and helping me figure out which to focus on. Tingfan Wu put in a great deal of work getting Diego the robot to be usable, and helped get me up to speed with how to use it. This thesis would have taken twice as long without his work. I'd also like to thank my other labmates at the Machine Perception Lab: Nick Butko, Paul Ruvolo, Jacob Whitehill, Josh Susskind, Deborah Forster, Mohsen Malmir, and others, who have given helpful feedback and helped make the lab an exciting place to be.

Thank you to my committee, for taking the time to listen to me and give constructive feedback.

I'd also like to thank Carson Dance for her support and patience.

Finally, I'd like to thank Diego, who is one of the most stubborn robots I know.

Chapter 2, in full, is a reprint of the material as it appears in the Proceedings of the International Conference on Development and Learning and Epigenetic Robotics. Talbott, W and Movellan, J, 2012. The dissertation/thesis author was the primary investigator and author of this paper

Chapter 3, in full, is a reprint of the material as it appears in the Proceedings of the International Conference on Development and Learning and Epigenetic Robotics. Talbott, W, Huang, H, and Movellan, J, 2012. The dissertation/thesis author was the primary investigator and author of this paper.

Chapter 4, in part, is a reprint of the material as it was presented to the Affor-

dances in Vision for Cognitive Robotics Workshop, Talbott, W and Movellan J, 2014. Chapter 4 is currently being prepared for submission for publication of the material. The dissertation/thesis author was the primary investigator and author of this paper.

Chapter 5, is currently being prepared for submission for publication of the material. The dissertation/thesis author was the primary investigator and author of this paper.

VITA

2004	B. S. in Symbolic Systems, Stanford University
2005	M.Sc. with Distinction in Informatics, University of Edinburgh
2008	P.D.Eng. in User-System Interaction, Eindhoven University of Technology
2015	Ph.D. in Cognitive Science, UC San Diego

PUBLICATIONS

W. Talbott, I. Fasel, J. Molina, V. de Sa, and J. Movellan, "Coordinating Touch and Vision to Learn What Objects Look Like," in *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, L. Carlson, C. Hölscher, and T. Shipley, Eds. Cognitive Science Society, 2011, pp. 562–567

W. Talbott, H. Huang, and J. Movellan, "Continuous-time infomax models of oculomotor control," in *Proceedings of the International Conference of Development and Learning and Epigenetic Robotics*, 2012

W. Talbott and J. Movellan, "An expected motion information model of salience for active cameras," in *Proceedings of the International Conference of Development and Learning and Epigenetic Robotics*, 2012, **Best Paper Award**

W. Talbott, T. Wu, and J. Movellan, "Estimating dynamic properties of objects from appearance," in *Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference on*, Aug 2013, pp. 1–6

W. Talbott and J. Movellan, "A computational framework for visual perception of inertial affordances," in *Affordances in Vision for Cognitive Robotics Workshop at RSS*, 2014

ABSTRACT OF THE DISSERTATION

Learning to Perceive: A Developmental Robotics Approach to Vision and Object Interaction

by

Walter A. Talbott

Doctor of Philosophy in Cognitive Science

University of California, San Diego, 2015

Professor Virginia de Sa, Chair

A robot is a true blank slate, awash in sensory information inextricable from its own actions. As such, it can be a powerful tool for investigating the space of problems that an infant, or whatever innate machinery was granted to the infant by evolution, must solve. The key observation is that the environment in which an infant develops is the same as that in which a robot might exist. A robot may have a different view on that environment, through different types or qualities of sensors. A robot may have different capabilities for interacting with the environment, for example by having wheels instead of legs. But, to act autonomously and coherently in the world, like an infant

learns to do, a robot must somehow make sense of its sensory information. The goal of this thesis, broadly, is to enable the pneumatic humanoid robot, Diego, to actively perceive the world. Specifically, three problems are addressed. How can a robot: (1) identify interesting information in its sensory input, (2) direct its sensors to best acquire meaningful information, (3) learn to generalize its experience to interact with novel objects? To help answer these questions, this thesis presents: (1) a model of salience that is suitable for active cameras, (2) a model of eye movements based on optimal control, and (3) a framework and robotic implementation for visual perception of the inertial properties of objects.

Chapter 1

Introduction

William James poetically described the perceptual challenge faced by newborns, writing, "The baby, assailed by eyes, ears, nose, skin, and entrails at once, feels it all as one great blooming, buzzing confusion." His suggestion is that the infant must interpret multiple continuous streams of sensory information to generate a meaningful perception of the statistical regularities present in the world. Compounding the confusion, the actions of the newborn affect the sensations it receives. A strict interpretation of James' ideas, of the environment as undifferentiated chaos presented to a blank-slate infant, may not aptly describe human infants, whose brains and bodies have been shaped by millions of years of evolutionary pressures. However, James' "buzzing confusion" is particularly suited to describing the problem faced by scientists working with artificial agents, such as robots. Wire several sensors together, throw in a few actuators, turn the power on, and nothing will happen. A robot is a true blank slate, awash in sensory information inextricable from its own actions. As such, it can be a powerful tool for understanding the space of problems that an infant, or whatever innate machinery was granted to the infant by evolution, must solve.

The key observation is that the environment in which an infant develops is the

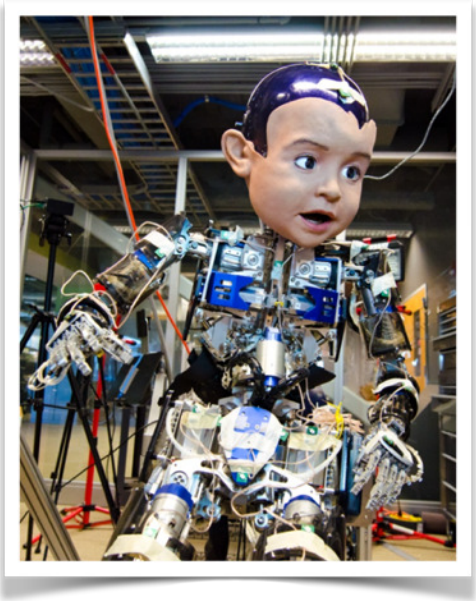


Figure 1.1: Diego, the pneumatic humanoid robot

same as that in which a robot might exist. A robot may have a different view on that environment, through different types or qualities of sensors. A robot may have different capabilities for interacting with the environment, for example by having wheels instead of legs. But, to act autonomously and coherently in the world, like an infant learns to do, a robot must somehow make sense of the confusion.

Diego, the robot pictured in Figure 1.1, is a pneumatic humanoid robot that is the motivation for this thesis. Diego was designed with actuators that approximate the control problem faced by the human body. Stiff actuators, like the ones found in many current robots, allow independent control of each joint, at the cost of increased energy required to overcome the stiffness and move the joints. In contrast, the human body has joints that are compliant (non-stiff), and pneumatic actuators are compliant as well. Compliant actuators mechanically couple the different joints. In other words, moving one limb creates significant forces in other limbs. Control signals sent to one joint must therefore take into account the movement and forces across the entire body. This makes

the control problem more difficult with pneumatic actuators, in a way that is similar to how human muscles do.

This thesis is a step on the path to enable Diego to perceive and act in the world. Although there are many interrelated aspects of the broad problem of learning to perceive, in this thesis we examine the following questions:

- **How can a robot identify meaningful information in its sensory stream?**
- **How can a robot direct its sensors to best extract meaningful information?**
- **How can a robot learn to interact with objects it has never seen before?**

To address the first question, Chapter 2 presents a model of visual salience that is sensitive to the motions generated by an active robotic camera. This model is able to explain human fixation data on still images, which we take as a measure for what is interesting in the visual input. In addition, the model explains why, in cameras that do not have very high temporal resolution, a policy like alternating saccades and fixations makes sense in order to best extract information from the visual input.

Chapter 3 deals with directing sensors to extract meaningful information. The chapter uses optimal control methods to explain the motor behavior of sensors, specifically eyes, in the case where the task is to gather information about the environment. We show that the model can explain saccade velocity profiles, as well as hand-eye coordination, and smooth pursuit.

Chapters 4 and 5 describe the replication, in simulation and with Diego, of an infant experiment in which object interaction is learned and generalized to novel objects. Stiff actuators do not require internal models, since the control problem is made easier by the design of the robot. This ease comes at a price of higher energy use. For compliant actuators, like those in Diego and like human muscles, we find that internal models of the body and object-body dynamics during manipulation, improve the performance of a

controller. We show that model-based controllers implemented on Diego display similar behavior to the infants, and are susceptible to being tricked by surprising relationships between visual and inertial properties of objects in the same way infants are.

Chapter 2

An Expected Motion Information Model of Salience for Active Cameras

Faced with the challenge of learning about its environment, an important first step for a robot is deciding how to direct its sensors to extract meaningful information. Computational models of visual salience have been developed to predict where humans tend to look in visual scenes, and thus they may provide useful heuristics for orienting robotic cameras. However, as we show here, current models of visual salience exhibit some important problems when applied to active robotic cameras. Here, we describe a new model of visual salience, named EMI, specifically designed to work on robotic cameras. The intuition behind this model is that, regardless of the task at hand, it is critical for robot cameras to keep track of motion, be it caused by camera movement or by world movement. Thus, it is reasonable for cameras to focus on image regions that are expected to provide high information about future motion, and to do so in a way that blurs the image as little as possible. We show that EMI predicts human fixations at least as well as current models of visual salience. In addition, we show that EMI overcomes the limitations that current visual salience models have when applied to robotic, active

cameras.

2.1 Introduction

Consider the night-driving scene pictured in Figure 2.1(a). Driving requires keeping track of one's vehicle's motion as well as the motion of other vehicles. Thus, it is important for the driver to look at regions of the scene expected to provide high information about motion. As we show in this paper, it is possible to compute the expected number of bits that each region of an arbitrary scene will provide about future motion. Figure 2.1(b) shows, for example, the top 15% of the pixels in the night-driving scene expected to provide the most information about future motion. To focus excessively on other image regions while driving may have catastrophic consequences.

Humans have a wide range of tasks and goals while acting in the world, but many of these tasks share a common need to keep track of how things move. It is thus reasonable to expect that the saccade distributions produced by people would show an aversion to image regions expected to provide little or no information about motion. The study of eye saccade distributions in free viewing conditions has recently become of interest in the computer vision, computational neuroscience, and machine learning literature [6][7][8][9][10][11][12][13]. Much of the previous work explains saccade distributions as the result of a bottom-up, task-free process known as visual salience. Typically these models ignore the fact that the observers have active visual sensors operating continuously. From the point of view of the observer, a static picture is in fact constantly changing due to the fact that the eyes, head, and body of the observer are not static.

Here, we propose that saccade distributions in free viewing conditions are the residue left by a distribution of saccade policies that reflect a wide range of tasks and

(a) (b)

Figure 2.1: (a) A typical night driving scene in San Diego. (b) Image regions expected to provide the most motion information.

goals. While it is difficult to infer what those tasks and goals are, most of them share a common need to keep track of motion. As such, human saccade distributions should have less density in image regions expected to provide little information about motion. Rather than seeing saccade distributions as a result of a task-free, bottom-up process, here we see them as the result of integrating over a wide range of task-driven policies. From this point of view, distinctions such as bottom-up and top-down salience are unnecessary and of little scientific value.

We show that Expected Motion Information (EMI) explains the saccade distributions obtained when people freely view static images as well as or better than previous models. Contrary to many previous models, EMI predicts that avoidance of low EMI image regions will be observed in a wide range of intelligent systems, natural and robotic. The tasks of different systems may be different, the statistics of the world they are exposed to may be different, but they all share a common need to avoid regions expected to provide little information about motion.

We further show that, consistent with other emerging views of their shortcomings [14], models of visual salience have important limitations when implemented in robot cameras. The problem addressed here is that these models find motion highly salient and thus, when used to control a camera, result in policies that maximize camera motion.

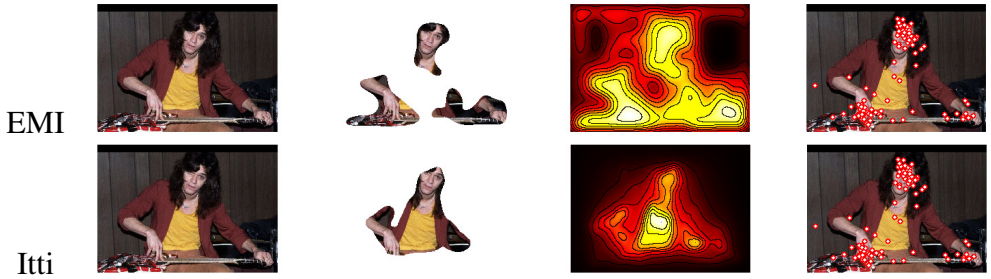


Figure 2.2: Results in an example image. Top Row: EMI model. Bottom Row: Itti's model. From Left to Right: Original Image. Top 20% most salient pixels. Itti's model salience and EMI maps. Human Saccade Distribution.

Contrary to this, EMI predicts a trade-off between focusing on new locations and the process of moving to a new location. As such, it predicts the emergence of saccade-like policies that minimize time spent in transition, rapidly moving visual sensors from location to location and staying in that location for a period of time. This work may provide clues on how to make robots that learn to scan the world on their own starting with low level principles. Policies like the ones predicted by current visual salience models, which continuously move a camera to maximize motion input, may not be conducive to the development of object detectors. Policies like the ones predicted by EMI, which rapidly saccade between image regions, may provide an easier path towards learning object detectors in an unsupervised manner.

In the next section, we present the EMI model and show that it is a reasonable contender for explaining human saccade distribution in free viewing conditions. We then detail the problems that standard visual salience algorithms face when coupled to robotic cameras, and how EMI solves these problems.

2.2 Expected Motion Information: Static Case

In this section, we examine an observer that is presented with a fixed image U and has to estimate how much motion information it expects to obtain prior to seeing a future image V. The formal model we use is a simplified version of G-Flow [15], originally developed for tracking objects in computer vision applications. Let $f : \mathbb{R}^2 \to \mathbb{R}^c$ be an image that maps pixel locations $x \in \mathbb{R}^2$ onto image values $f(x) \in \mathbb{R}^c$. Each of the c elements of $f(x)$ can be seen as an image channel (e.g., a color channel). For a fixed pixel location $x \in \mathbb{R}^2$, let $u = f(x) \in R^c$. Let $\Theta \in \mathbb{R}^2$ be a random variable representing an image displacement. Let $V \in \mathbb{R}^c$ be defined as follows

$$V = f(x+\Theta)+W \tag{2.1}$$

where $W \in \mathbb{R}^c$ is a zero mean Gaussian variable with covariance matrix σ_w. Given u and a random sample v of V our task is to infer the value of the displacement Θ that generated image v from image u. Note for a fixed value θ of Θ, we get

$$p(\theta \,|\, u,v) \propto p(u)p(\theta \,|\, u)p(v \,|\, f,\theta) \propto p(\theta)p(v \,|\, u,\theta) \tag{2.2}$$

where

$$p(v \,|\, u,\theta) = \phi(v \,|\, f(x+\theta),\sigma_w) \tag{2.3}$$

and $\phi(\cdot \,|\, \mu,\sigma)$ is a multivariate Gaussian density function with mean μ and covariance matrix σ. For small θ

$$f(x+\theta) \approx f(x)+J\theta \tag{2.4}$$

where the Jacobian $J \in \mathbb{R}^c \times \mathbb{R}^2$ is defined as follows

$$J = \frac{\partial f(x)}{\partial x'} = \begin{pmatrix} \frac{\partial f_1(x)}{\partial x_1} & \frac{\partial f_1(x)}{\partial x_2} \\ \vdots & \vdots \\ \frac{\partial f_c(x)}{\partial x_1} & \frac{\partial f_c(x)}{\partial x_2} \end{pmatrix} = \begin{pmatrix} J_1 \\ \vdots \\ J_c \end{pmatrix} \tag{2.5}$$

and

$$J_i = \frac{\partial f_i(x)}{\partial x'} = (\frac{\partial f_i(x)}{\partial x_1}, \frac{\partial f_i(x)}{\partial x_2}) \tag{2.6}$$

Thus, for small θ

$$p(v \,|\, u, \theta) = \phi(d \,|\, J\theta, \sigma_w) \tag{2.7}$$

where

$$d = v - u \tag{2.8}$$

We let the prior distribution of θ be Gaussian with mean μ_θ and covariance matrix σ_θ. We let this prior be such that only small values of θ are probable, thus making the linear approximation of f valid with high probability. In this case,

$$\begin{aligned} &\log p(\theta \,|\, u, v, \eta_w) = \\ &-\frac{1}{2}(\theta - \mu_\theta)'\eta_\theta(\theta - \mu_\theta) - \frac{\eta_w}{2}(d - J\,\theta)'(d - J\,\theta) + c \end{aligned} \tag{2.9}$$

where $\eta_\theta = \sigma_\theta^{-1}$ is the prior precision matrix and $\sigma_w = I/\eta_w$. We note that the log of the posterior distribution has only linear and quadratic terms on θ and therefore it has to be Gaussian. The expected value and covariance matrix of the posterior distribution can be

easily derived

$$\mathrm{E}[\Theta \mid u, v] = (\eta_\theta + J'\eta_w J)^{-1}(\eta_\theta \mu_\theta + J'\eta_w d) \tag{2.10}$$

$$\mathrm{Cov}(\Theta \mid u, v) = \left(\eta_\theta + J'\eta_w J\right)^{-1} \tag{2.11}$$

Note $J'J$ and $J'd$ are sums over image channels

$$J'J = \sum_{i=1}^{c} J_i' J_i \tag{2.12}$$

$$J'd = \sum_{i=1}^{c} J_i' d_i \tag{2.13}$$

A key measure used by the EMI model is the information that we expect the future image V to provide about the Θ. This is given by the following mutual information function

$$I(\Theta, V \mid u) = H(\Theta \mid u) - H(\Theta \mid V, u) = H(\Theta) - H(\Theta \mid V, u) \tag{2.14}$$

$H(\Theta)$ represents the current motion uncertainty (entropy). Since we assume a Gaussian prior we have

$$H(\Theta) = \frac{1}{2}\log\left|\sigma_\theta\right| + c \tag{2.15}$$

where c is a constant we may ignore. The term $H(\Theta \mid V, u)$ represents the uncertainty about motion we expect to have after observing the next image V. Note this is an expected

uncertainty and thus it requires integrating over all possible future images

$$H(\Theta \mid V,u) = \int p(v) H(\Theta \mid v,u) dv \tag{2.16}$$

where

$$\begin{aligned} H(\Theta \mid u,v) &= \frac{1}{2} \log \left| \text{cov } (\Theta \mid u,v) \right| + c \\ &= -\frac{1}{2} \log \left| \eta_\theta + J' \eta_w J \right| + c \end{aligned} \tag{2.17}$$

where c is a constant. Note the posterior entropy is independent of v. Thus

$$\begin{aligned} H(\Theta \mid V,u) &= \int p(v \mid u) H(\theta \mid u,v)\, dv \\ &= -\frac{1}{2} \log \left| \eta_\theta + J' \eta_w J \right| + c \end{aligned} \tag{2.18}$$

An essential aspect of this equation is that the uncertainty about motion $H(\Theta, \mid V,u)$ that we expect to have after we observe the future image V can be computed analytically prior to observing that image. This allows using the current image to position the eye (or robot camera) on locations of the image that are expected to provide high motion information in the future.

2.2.1 Optimizing Expected Motion Information

In the previous section, we computed the expected motion information for a fixed pixel location x. In general, we want to observe multiple pixels, in which case we work with a collection of variables $\{\Theta_x, u_x, V_x\}$ indexed by the location x. A saccade positions the visual sensor at a given location providing access to a set of pixels $\mathcal{S}$. We can measure the expected motion information that the saccade will provide by summing expected

information over the pixels it gives access to

$$M(\mathcal{S},v)=\sum_{x\in S} I(\Theta_x \,|\, V_x,u_x)=\sum_{x\in\mathcal{S}} = H(\Theta_x)-H(\Theta_x \,|\, V_x,u_x) \tag{2.19}$$

where

$$H(\Theta_x)=-\frac{1}{2}\log|\eta_\theta(x)|+c \tag{2.20}$$

is the current uncertainty about the future motion at pixel x and

$$H(\Theta_x \,|\, V_x,u_x)=-\frac{1}{2}\log\left|\eta_\theta(x)+J_x'\eta_w J_x\right|+c \tag{2.21}$$

is the uncertainty we expect to have after we observe the next image V. Note $H(\Theta_x \,|\, V_x,u_x)$ can be seen as a measure of the quality of the sensor available at pixel location x. A large value of $H(\Theta_x \,|\, V_x,u_x)$ indicates a bad sensor and a small value indicates a good sensor, i.e. an image location expected to provide high information about future motion. Thus, the expected future motion information $M(\mathcal{S},v)$ provided by a saccade $\mathcal{S}$ to a current image u integrates two factors: On one hand, it likes locations where there is currently high uncertainty about motion, i.e., $H(\Theta_x)$. On the other hand, it likes image locations expected to reduce motion uncertainty, i.e., regions with low $H(\Theta_x \,|\, ,V_x,u_x)$.

2.3 Predicting Saccade Distributions

We compared the performance of EMI and two popular models of visual salience: Itti's model [7] and the SUN model [10]. Itti's model defines highly salient regions as regions with higher response from a neural network then neighboring regions. SUN models free viewing of images as a search task, where the likelihood of the observed

image given the target object is uniform. Under this assumption, the information provided by an image region about the target object is inversely proportional to the logarithm of its probability in naturally occurring images.The implementation of the Itti model and the SUN model were downloaded from the Web sites of their developers. Our implementation of EMI will also be made available on the Web.

We tested the models on a subset of 429 images from the NUSEF dataset [16]. The dataset contains probability distributions of fixations made by 75 subjects in free-viewing conditions. Performance was measured in terms of the accuracy of the model in a 2 Alternative Forced Choice Task (2AFC): On each trial, we randomly select two pixels: one pixel fixated by at least one human and one pixel fixated by no human. The goal was to discriminate which pixel was fixated by humans using only the salience values given by the model. A 2AFC of 50 indicates that the salience value provides no information. A 2AFC value of 80 indicates 80 % accuracy on the task. An advantage of the 2AFC statistic (also known as the area under the ROC) is that it is independent of the relative frequency of the two categories, and that it is invariant to monotonic transformation of the salience values.

The performance of the SUN model averaged across the 429 images was 77.3 %. The performance of Itti's model was 81.17 % and the performance of the EMI model was 81.77 %. The difference between the EMI model and the SUN model was large and statistically significant $t(428) = 18.46, p < 0.01$. The difference between the EMI model and Itti's model was small but also statistically significant, $t(428) = 2.48, p < 0.05$.

Thus, it appears that expected motion information describes fixation distributions at least as well as existing models of visual salience. Figure 2.2 shows an example of the predictions made by EMI and by Itti's model.

2.4 Expected Motion Information: The Dynamic Case

In previous sections, we analyzed an observer frozen in time who sees an image U at time t and has to decide where to saccade next. Here, we extend the model to analyze the continuous time dynamics of an observer operating with an active camera. The model presented here can be seen as a combination of the I-POMDP of eye movements [17] and the G-Flow model of object tracking [15]. While I-POMDP focuses on gathering information about the location of a target object, here we focus on the problem of gathering information about local motion.

We model the image formation process as a set of linear stochastic differential equations. Let $Y_t \in \mathbb{R}^n$ be a collection of point objects whose appearance and world coordinates may change over time, e.g.,

$$dY_t = dB_t^y \tag{2.22}$$

where B^y is a Brownian process. We think of images as 2D clouds of these point objects. Let $S_t \in \mathbb{R}^2$ represent the location of the camera in world coordinates. This location can be controlled via motor actions A_t

$$dS_t = A_t dt + dB_t^s \tag{2.23}$$

The local displacement vector $\Theta_t \in \mathbb{R}^n$ represents the current location of the different point objects in the image plane

$$d\Theta_t = dS_t + dB_t^\theta \tag{2.24}$$

Here, dS_t models motion due to controlled camera displacements and dB_t models motion due to other uncontrolled processes. Neither Θ_t, nor Y_t are directly observable. Instead,

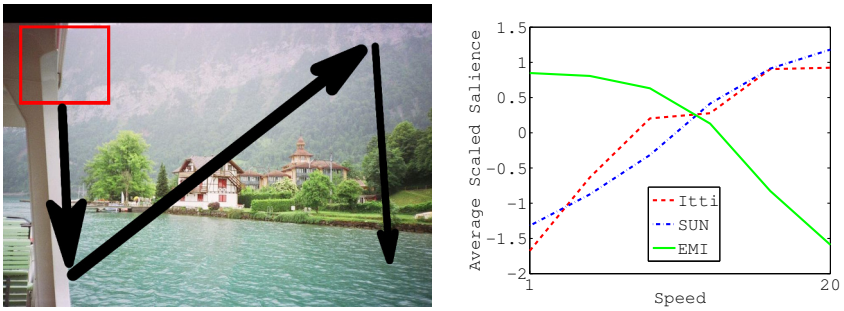
the observer has access to portions of the image $U_t \in \mathbb{R}^m$

$$dU_t = \frac{1}{\tau}(f(Y_t, \Theta_t) - U_t)dt + dB_t^u \tag{2.25}$$

where $f(Y_t, \Theta_t)$ is simply a permutation of Y_t according to the current location Θ_t of the point objects. Note U_t, will be a weighted average of the image Y_t transformed by the local shifts Θ_t and the previous images $\{U_s : s < t\}$. Thus, a camera with a long time constant (slow shutter) τ will produce images with high motion blur. The expected motion information $\mathcal{M}_t(S_t, U_t)$ is given by the EMI formula (2.19) applied to the set of pixels of U_t visible at that time. Which pixels are visible depends on the current location S_t of the camera. From the point of view of the EMI model, good policies move the camera so as to gather large amount of information about the entire local motion vector $d\Theta_t$. For locations $x \in \mathbb{R}_2$ that are not currently seen by the camera, the uncertainty of the motion increases due to the effect of dB_t^θ. For locations seen by the camera, uncertainty will increase due to the passage of time, but it will also decrease due to the sensory U_t provided by the camera. The amount by which it is expected to decrease is controlled by the EMI equation (2.19). This is superficially related to the model of Najemnik and Geisler (NG)[18] in that it is based on expected information gain. However, NG is a model of visual search in a very specific task (e.g., search for a known pattern in a pink Gaussian noise background). As such, its goal and scope are quite different from EMI.

2.4.1 Optimal Inference and Control

Optimal inference under a model similar to this used for a computer vision task was analyzed in [15]. There, it is shown that inference can be handled with standard Kalman filters with time dependent observation parameters. The main difference added here is the use of camera motion blur.



(a) (b)

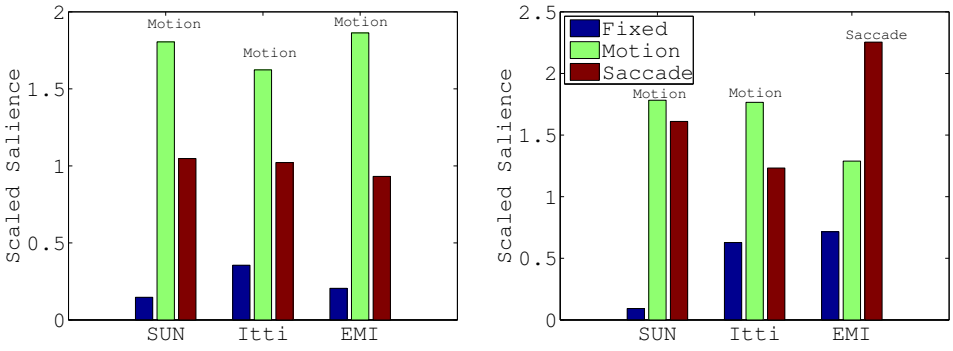
Figure 2.3: Salience ratings for motion. (a) Movies are generated by sliding a box over an image in a predefined path at different speeds, with motion-induced blur added as described in the text. (b) Normalized mean salience from Itti and SUN model across the frames of 20 videos plotted against speed of the video.

While the model is very simple, it already captures important trade-offs faced when designing a camera controller. If the camera does not move, the uncertainty about motion in the unseen portions of the scene will increase rapidly. On the other hand, if the camera moves, motion blur will decrease the expected motion information provided by the observed images.

In contrast, current visual salience models treat motion as a desirable (salient) property of visual sequences. This is problematic both for developing robot cameras that learn on their own how to maximize visual information gathering, and for modeling how humans move their eyes. In the previous sections, we saw that EMI predicts well *where* humans move their eyes. Here, we show that previous models of salience cannot predict *how* humans move their eyes, whereas, at least qualitatively EMI, does not have such a problem.

2.5 Preference for Motion

To explore the problems with current models of salience, we set up a test case by constructing movies that scan a scene with varying speed and motion blur parameters.



high temporal resolution (no blur) (a)

standard temporal resolution (blur) (b)

Figure 2.4: Salience in high- and low-temporal-resolution cameras. Normalized salience estimates from SUN, Itti & Koch, and EMI for the three policies, Fixed, Random, and Saccade-like, in two conditions: (a) a high temporal resolution, non-blurring camera, and (b) a standard temporal resolution camera. Notice that EMI rates the motion policy more salient (higher quality policy) when there is no blur, but unlike the other measures, prefers a saccade-like policy when camera movements generate blur.

As backgrounds for these movies, we selected images from the NUSEF database [16]. Figure 2.3(a) shows a schematic for how these movies were generated: a small field of view was moved along the background image following a fixed path at a standard speed. This generated a video sequence Y. The camera representation U of the input sequence was determined by the following model

$$dU_t = \frac{1}{\tau}(Y_{\alpha t} - X_t) \tag{2.26}$$

Thus, α determines the speed at which the camera moves through the standard path. For $\alpha = 1$, it moves at the standard speed, for $\alpha = 2$ it moves at twice the standard speed. The parameter τ represents the time constant of the camera sensor. Large time constants mean that the sensor blurs more through time. We used 20 different images as backgrounds, and videos were generated for $\tau \in \{1, 2\}$ and $\alpha \in \{1, 4/3, 2, 4, 10, 20\}$. We examined

the average visual salience of these movies as computed by the Itti & Koch model [7], the SUN [8] model and EMI (see Figure 2.3(b)). As the speed of the movie increases, the previous salience algorithms estimate higher salience. Such unconstrained preference for motion is problematic. Any robotic implementation that attempts to maximize these measures of salience would result in a robot that moves as much as possible in order to increase how interesting it thinks its environment is. The problem is that these approaches do not capture the fact that the reliability of the sensors decreases during motion, and thus the amount of useful motion information extracted from the world is actually lower when the camera moves very fast. As Figure 2.3(b) shows, EMI does not like high camera motion, due to the fact that it produces temporal blur, which degrades the visual information about motion.

Some previous work in robotics has attempted to design cameras that learn appropriate compensatory movements to stabilize the image in a camera during self motion [19]. Other work has dealt with the problem by requiring the head to remain fixed [20] or only estimating salience after a movement has been completed [21]. These approaches, however, do not explain why it is important to be wary of motion. The next section provides an experimental argument that the EMI model deals with self-produced motion in an intelligent way. EMI also predicts when saccadic camera movements, as opposed to continuous camera motion, are desirable.

2.6 Control Policies for Active Cameras

To examine the type of exploratory behavior favored by SUN, Itti's model, and EMI, we evaluated the salience of a set of movies generated by three different camera control policies. Extrapolating from these estimates, we can get a qualitative sense for what types of eye movement policies the different algorithms would prefer. Here, a policy

defines the path of a movie frame across a background image, similar to the process outlined in Figure 2.3(a), subject to the camera model from equation (2.26).

Three policies were considered. Under policy F, the camera remains fixed in a constant location for the duration of the movie. Under policy M, the camera moves rapidly and continuously around the image. Although the path changes direction, it does not pause in any location. The third policy, S, mimics a series of saccades by moving first to the left of the image, pausing there, and then moving to the right of the image.

The three policies, F, M, and S, were evaluated with two different sensory conditions. The first condition used a camera model with very good temporal resolution, i.e., very small time constant τ. The second condition used a camera with temporal resolution approximating that of the human eye.

The scaled salience estimates from the three algorithms for the movies associated with policies F, R, and S are shown in Figure 2.4. Figure 2.4(a) shows the results with a simulated high-shutter-speed (no motion blur) camera. In this case, the three salience algorithms have the same preferences. They all prefer the high motion (M) policy. This makes sense: since the camera sensor has a very small time constant, one loses no information by moving the camera quickly. However, things change dramatically when operating with a more realistic camera that produces motion blur, see Figure 2.4(b). In this case, SUN and Itti's model still prefer the high, continuous camera motion. EMI, on the other hand, prefers the saccadic policy (S), presumably because it minimizes the loss of information due to motion blur during saccadic flight.

2.7 Conclusions

Recent years have seen great advances in the development of computational model of visual salience. While these models predict well where humans will look in

a scene, they have problems predicting how they will do so. As such, they are not well suited for robotic cameras that learn to develop their own motor policies to maximize visual sensory information. Here, we have proposed EMI as a model of salience which explains saccade distributions at least as well as current models of visual salience while solving their limitations for robotic applications. These limitations include a preference for motion. This preference, when used to guide an active camera, would lead to policies that move the camera without stopping. When we use EMI as a measure of the quality of a movement policy, we see that it prefers ballistic behaviors that minimize time in motion (when the cameras are blurry). In future work, we will apply the EMI measure of salience to allow the Machine Perception Lab's humanoid robot, Diego-san, to learn how to move its cameras during explorations of novel objects.

Acknowledgments

This research was supported by NSF IIS 0968573 and NSF IIS 0808767.

Chapter 2, in full, is a reprint of the material as it appears in the Proceedings of the International Conference on Development and Learning and Epigenetic Robotics. Talbott, W and Movellan, J., 2012. The dissertation/thesis author was the primary investigator and author of this paper

Chapter 3

Infomax models of oculomotor control

From a Bayesian point of view, learning is simply the process of making inferences about the world based on incoming data. The efficiency of this learning is determined by the quality of the information provided by the sensors. Thus, a critical part of learning is the existence of a sensory-motor system designed to maximize the information required to achieve goals. Here we show that a wide range of primate eye movement phenomena can be elegantly explained from the point of view of infomax control. The proposed approach describes the velocity profiles of saccadic eye movements as well as previously existing models. In addition, the infomax approach explains phenomena that are beyond the scope of previous models: non-saccadic eye movements, and the difference in end point and velocity profiles observed in saccade-to-target and reach-to-target tasks. The results suggest that the occulomotor control system evolved to be a very efficient real time learning machine.

3.1 Introduction

From a Bayesian point of view, learning is simply the process of making inferences about the world based on incoming data. The efficiency of this learning is

determined by the ability of the sensory-motor control system to maximize the information needed to achieve goals (infomax control).

Humans make over 150,000 saccades per day, spending about 2 hours in saccadic flight, during which useful vision is very poor. It is well known that the velocity profiles of primate saccadic eye movements are quite stereotyped and adhere to consistent relationships between amplitude, duration, and peak velocity. These relationships have been called the "main sequence" [22].

Recent models of oculomotor control have been successful at describing saccade velocity profiles using optimal control principles. Typically, these models focus on the relationship between motor commands and forces applied to the eyes, and postulate that the goal of the oculomotor system is to drive the eye to target locations as quickly and accurately as possible. Some models postulate that the eyes minimize the expected deviation from a target end point [23, 24]. Other models postulate that eye movements minimize the time required to reach the target point [25], which turns out to be mathematically equivalent. These models ignore the sensory properties of the eyes and assume that the goal of oculomotor control is to reach target points. How these targets are selected is beyond the scope of the models. A recent class of models has focused on explaining how the target points are selected using information maximization principles [26, 27]. Up to now these models have focused on the sensory properties of the eyes (e.g., the fall-off of sensitivity as a function of eccentricity) and have ignored their mechanical properties. Here, we show that by jointly examining the sensory and mechanical properties of the eyes it is possible to explain a range of new phenomena that were beyond the scope of the previous models. The approach shows that a wide range of primate eye movement phenomena reveal that the primate oculomotor system evolved to be a very efficient real-time learning machine.

3.2 Infomax Model

To model oculomotor control, we first need to have a description of the system of the eye. We follow [23] in using a state-space model with signal dependent noise to describe the eye. We call the state of the eye at time t as X_t, and describe its changes through time with

$$dX_t = \underbrace{AX_t dt}_{\textbf{drift}} + \underbrace{BU_t dt}_{\textbf{control}} + \underbrace{(C + U_t)dB_t}_{\textbf{noise}} \tag{3.1}$$

$$X_t = \begin{bmatrix} X_{e,t} \\ \dot{X}_{e,t} \end{bmatrix} \tag{3.2}$$

Where the matrix A represents the passive dynamics of the system, B describes the effects of the control inputs U_t on the state, and C describes the effect of the Brownian motion B_t on the state. X_t is the state matrix, which contains the eye position $X_{e,t}$ and velocity $\dot{X}_{e,t}$ Notice that the noise scales with the size of the control input, which gives rise to a tradeoff between controlling the system and being certain of the system's state. The values in the A and B matrices were found from human saccades in [28], and like [23] we use these values. These values were retrieved from horizontal saccades, and we model saccades similarly in only one dimension.

Although previous models have assumed that the endpoint of a saccade, which we will call Z, is known exactly, here we introduce target uncertainty by treating Z as a random variable. Especially if this target is presented in the periphery, subjects will be unsure of the target's location due to sensory uncertainty. Here we model the belief of the target's location as a Gaussian distribution. Additionally, we assume target has its

own dynamics described by

$$dZ_t = \eta_t dt + n dV_t \tag{3.3}$$

where η_t is the model of the target's velocity, and dV_t is Brownian motion, with magnitude determined by n. We assume the model of the target dynamics is known. The model could potentially be learned or estimated from the observations of the target, but we do not address this issue here. Even though the dynamics are known, Z is not, so we need a model for how the eye learns about the location of the target.

Similar to [29], who use a POMDP framework to examine hand-eye coordination, we model the observations Y that the eye collects about the target. These observations change through time as

$$dY_t = (X_{e,t} - Z_t) g(X_{e,t}, Z_t, \dot{Z}_t) dt + dW_t \tag{3.4}$$

If the observations were noiseless and accurate, they would give the eccentricity of the target with respect to the location of the eye. However, the observations are contaminated by noise, dW_t, and the signal-to-noise ratio (SNR) is defined by the visual acuity function g. We choose the following form of the visual acuity function

$$g(X_t, Z_t, \dot{Z}_t) = e^{-\left[\underbrace{\frac{1}{\rho}((X_{e,t} - Z_t)\beta)^\rho}_{\textbf{eccentricity}} + \underbrace{\frac{1}{2}(\dot{X}_{e,t} - \dot{Z}_t)^2\gamma}_{\textbf{velocity}}\right]} \tag{3.5}$$

where ρ and β define the shape and width of the falloff in SNR due to the target's eccentricity, and γ defines the width of the falloff in SNR due to the relative velocity of the eye. For computational simplicity, we restrict ρ to be an even number so we can

avoid using an absolute value on $(X_{e,t} - Z_t)$. For example, if we assume the velocity term is zero, and $\rho = 4$, Figure 3.1 shows a schematic of how the SNR would decrease as the eccentricity (on the x-axis) diverges from zero in either direction.

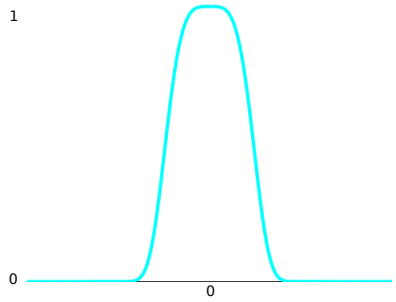


Figure 3.1: Schematic figure of how the SNR decreases as the eccentricity (x-axis) differs increasingly from zero.

A similar shape would hold for the velocity term of the visual acuity function as well. The velocity term models the cost of moving the eyes rapidly. In humans, saccadic suppression masks high-frequency visual information during fast eye movements. Similarly, in the model, a high velocity of the eye with respect to the target reduces the SNR of the observations. This sets up another tradeoff. If the eye is far from a stationary target, the controller must decide whether it is better to make slow movements that generate more reliable observations, or fast, unreliable movements to decrease the eccentricity of the target. The optimal action will depend on the values of the parameters ρ, β, and γ, and the relative cost of action and uncertainty. Taken together, (3.1), (3.3), (3.4), and (3.5) describe the system to be controlled. What remains is to find a control policy that can generate the actions U_t from times 0 to final time horizon T such that the system is driven to a desired state.

3.2.1 Learning the control policy

To find the optimal policy, we first need to define an objective function. Here we use a quadratic objective function, and use iterative Linear Quadratic Gaussian (iLQG) [30] control, which will require dealing with both the partial observability and the non-linearity from (3.5). In this paper, we will model eye movements in three tasks (target-directed saccades, smooth pursuit, and eye-hand coordination). Objective function in (3.8) will be modified based on task goals.

For the moment, we will focus on target-directed saccades. In this task, we use data collected from horizontal saccades in humans [31]. The participants were asked to saccade from a central fixation point to a flashed target at different amplitudes. Because the task involves positioning the eyes as close as possible to the target, we start with a term for minimizing the squared error between eye and target. Additionally, we include a term to model the cost of action. Let the cost function take the form

$$(X-Z)^2+U^2Q \tag{3.6}$$

where Q is a scalar that captures the tradeoff in cost related to being far from the target point and making actions.

The expression in (3.6) will be the cost function if the target location Z is known. However, because the location Z is unknown, we cannot use this cost directly. Instead, we need to take the expected value of (3.6), which leads to

$$(X-\hat{Z})^2+\sigma_Z+U^2Q \tag{3.7}$$

where $E[Z]=\hat{Z}$ and σ_Z is the variance of the estimate of the target location. With (3.7), we have a quadratic cost function. Notice that even in a situation where the task is to

move to a specified location, there is still pressure to find a solution that maximizes the information about the target location.

Figure 3.2: Finite horizon time segments. The two segments of the finite horizon used for control. First, from time 0 to time T, there is no penalty for the distance from the eye to the target. Second, during fixation (from time T to time $T+F$), the penalty on the eye position is enforced. The penalty on the magnitude of the action U is enforced for the entire horizon.

In this target-directed task, the entire eye movement in one trial includes first a saccade and then a short fixation period at the target. To apply the cost function (3.7) to the entire movement, we also separate the movement control into these two segments, as shown in Figure 3.2. The first segment, which we will call the saccade, only contains penalties on the actions. The second segment, from time T to time $T+F$, which we call the fixation, also includes the penalty on the state of the eye. With this, the complete minimization objective becomes

$$\int_T^{T+F} ((X_t - \hat{Z})^2 + \sigma_Z)dt + \int_0^{T+F} U_t^2 Q dt \tag{3.8}$$

Following [30], we include $\hat{Z}$ and σ_Z in the state X, and plan according to the belief state. Using an extended Kalman-Bucy filter, we can find the dynamics of $\hat{Z}$ and

σ_Z. The extended Kalman-Bucy filter equations are as follows

$$d\hat{Z}_t = \eta dt + K_t dI_t \tag{3.9}$$

$$dI_t = (dY_t - f(X_t, \hat{Z}_t, \dot{\hat{Z}}_t))dt \tag{3.10}$$

$$K_t = \sigma_{Z,t} \frac{\partial}{\partial \hat{Z}_t} f(X_t, \hat{Z}_t, \dot{\hat{Z}}_t) \tag{3.11}$$

$$d\sigma_{Z,t} = -K_t^2 dt + n_t^2 dt \tag{3.12}$$

where we've defined

$$f(X_t, \hat{Z}_t, \dot{\hat{Z}}_t) = (X_{e,t} - \hat{Z}_t) g(X_t, \hat{Z}_t, \dot{\hat{Z}}_t) \tag{3.13}$$

and g is as in (3.5). Using the product rule, we can find

$$\frac{\partial}{\partial \hat{Z}_t} f(X_t, \hat{Z}_t, \dot{\hat{Z}}_t) = g(X_t, \hat{Z}_t, \dot{\hat{Z}}_t)(1 - \beta(X_{e,t} - \hat{Z}_t)^\rho) \tag{3.14}$$

Using the above equations, we can incorporate the observation process Y with the estimate of the target location and give the dynamics of $\hat{Z}$ and σ_Z in relation to time.

The final step is to augment the A and B matrices from (3.1) to include the dynamics of $\hat{Z}$ and σ_Z in relation to changes in the other state variables. We can find the necessary terms in the augmented matrices by taking the partial derivatives of (3.12) with respect to $X_{e,t}$, $\dot{X}_{e,t}$, $\hat{Z}$, and σ_Z. This will allow us to linearize the dynamics of the system around a given state or sequence of states.

With the linearized dynamics, the belief state $\hat{Z}$, and the quadratic cost function, we can now solve for a control policy using iLQG. The policy learned by iLQG is a closed-loop policy. This means the optimal action at a given time can depend on the state rather than just the time; in other words, the optimal policy can react to changes in the

environment. This feature of the policy is interesting, and differs from previous models, and its implications are discussed in more detail in Section 3.5. Once the control policy has been learned, it can be applied to a noise-free simulation to give an the expected trajectory of the eyes for a set of parameters.

3.3 Evaluation Methods

To evaluate the suitability of the infomax model for describing oculomotor control, we looked at three different eye movement paradigms. The first is in describing the velocity profiles of horizontal saccades, as was described earlier. Second, we examined the qualitative suitability of our model for predicting smooth pursuit in amenable situations in simulation. We also examined a task where the eye played a supportive role to the hand, which had to reach a target.

3.3.1 Saccades

To learn the parameters of the system that describes the eye's movement and observations, we perform a pattern search to minimize the root squared error between the velocity profiles of 5, 10, 20, 30, 40, and 50 degree saccades generated by the optimal controller under the fixed set of parameters and the behavioral data from [31]. We allow the saccade duration T to change with each amplitude, but all other parameters are held constant across amplitudes.

To compare the infomax model to other models, we use a cross-validation paradigm, where each amplitude saccade is held out in turn. The parameters for each model are set from the remaining amplitudes, and an optimal movement for the held-out amplitude is generated with the learned parameters. Then, the error is calculated, and averaged across the amplitudes.

3.3.2 Smooth pursuit

To model eye movements other than saccade, we need only make minimal changes to the objective function (3.8). Rather than considering the task where the eyes are required to move to a particular location (as was the case in our model of the saccade task from [31]), here we only consider the goal of minimizing the variance of the estimate of the target location. As such, the objective comes closer to pure information maximization, and is defined as

$$\int_{T}^{T+F} \sigma_Z dt + \int_{0}^{T+F} U_t^2 Q dt. \tag{3.15}$$

Although we have tried the following experiments with an objective function closer to (3.8) with similar results, it is more compelling to show that even without a strict penalty on the location of the eyes, we can generate qualitatively similar behavior to smooth pursuit, so we will focus on this case.

3.3.3 Hand-eye coordination

We model eye movements in a rapid reaching task (data collected and described by [32]) in two conditions: Eye+hand, in which the reward is given based on the endpoint of hand movement; Eyes only, in which the reward is given based on the endpoint of saccadic eye movement. In the experiment, subjects were instructed to reach the target at a distance of 20 cm (25 degree visual angle) from the starting location either with hand (Eyes+hand) or eye (Eyes only) movement at a time window of 600 ms. For the former, subjects can freely move their eyes and thus eye movements only serve to guide hand movements. For the latter, subjects' reward will be based on the endpoints of eye movements and thus eye movements contribute directly to the task goal. We used Eyelink1000 system to track eye movement and Phasespace motion capture system to

record hand movement in the experiment.

We model the hand as a point mass, and used the dynamics for the hand as described in [25].

For Eyes+hand condition, without constraint on eye movement to the target, the objective function for eye movement is

$$\int_{T}^{T+F} ((X_{h,t} - \hat{Z})^2 + \sigma_Z)dt + \int_{0}^{T+F} U_t^2 Q dt \tag{3.16}$$

where $X_{h,t}$ is the position of the hand at time t.

For Eyes only condition, with the task goal of moving eyes to the target, similar as the target-directed saccade task, the objective function for eye movement is (3.8).

3.4 Results

3.4.1 Predictions of optimal saccades for static targets

Figure 3.3 compares the infomax model prediction of saccade velocity profiles over a range of amplitudes. Optimal velocity profiles (Fig 3.3b) captured the important shape features shown in behavioral data (Fig. 3.3a) (i.e. symmetric for low amplitudes and asymmetric/left-skewed peak for high amplitudes). The optimal positions (Fig. 3.3c) also show similar trajectories as in the observed behavior.

In Figure 3.4, we compare infomax with previous models (Internal Model from [33]; Minimum Variance Model from [24]). RSE comparison (Fig. 3.4c) suggests there is no significant difference (summed over all velocity profiles between behavior and model predictions) between those 3 models ($p>0.1$).

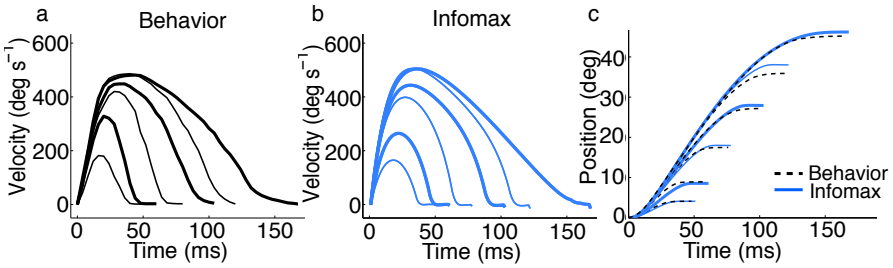


Figure 3.3: Comparison of behavioral result and infomax predictions. a. Observed velocity profiles of horizontal saccades when the target is at 5°, 10°, 20°, 30°, 40° and 50° ([31]). b. Optimal saccadic velocity profiles for corresponding amplitudes shown in a. c. Optimal eye positions (solid blue line) and observed eye positions (dashed black line) for the amplitudes shown in a.

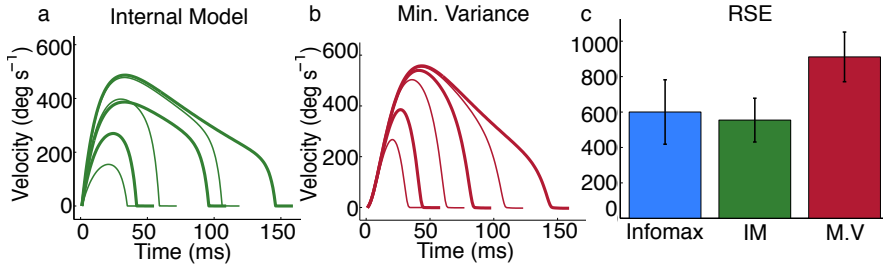


Figure 3.4: Comparison of infomax and previous models. a. Internal Model from [33]. b. Minimum Variance Model from [24]. c. Mean RSE over all the amplitudes, error bars show the standard error of the mean.

3.4.2 Prediction of saccadic and smooth pursuit eye movement for moving targets

Figure 3.5a and 3.5b show infomax prediction for eye movements when the target moves at 20 deg/s with no location difference between initial fixation and the onset of the moving target. Eye velocity trace in Fig. 3.5a suggests the eye will increase velocity rapidly and continuously until reaches the target velocity ($\sim$ 40 ms after target onset), and then track the target using pursuit eye movements. Eye position trace in Fig. 3.5b shows eye positions closely match target locations.

Figure 3.5c and 3.5d shows model prediction of eye movement when the moving target is initially located 5 deg to the right of fixation, and then moves to the right at

10 deg/s. Eye velocity trace in Fig. 3.5c shows the eye will first make a quick catch-up saccade-like movement to the target (∼150 ms after target onset) and then track the moving target at a matching speed. The eye position trace in Fig. 3.5d suggests the eye will reach the target at the end of the first quick movement and then track the target position.

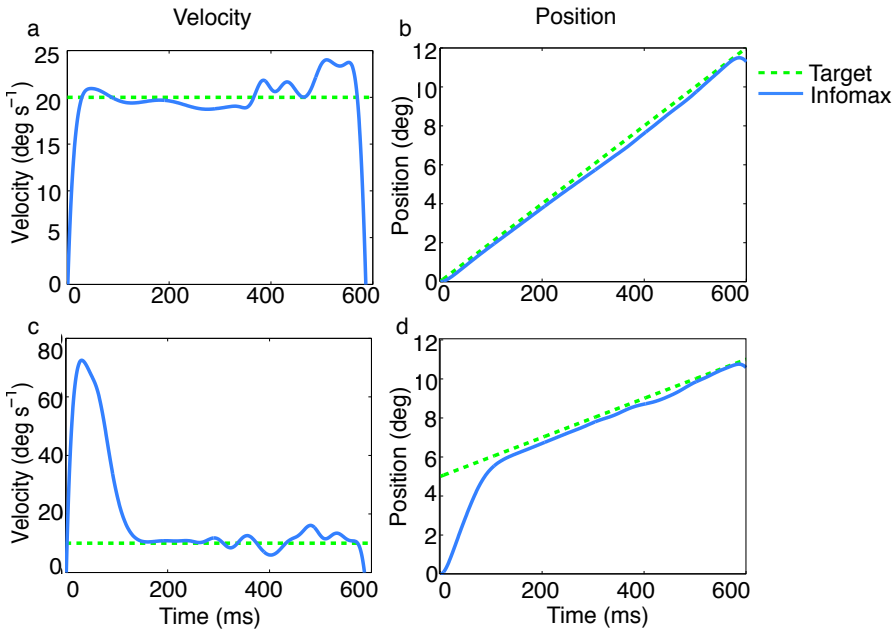


Figure 3.5: Representative eye movement responses to moving targets. Top row: a. Eye velocity trace (solid blue line) and b. corresponding eye position and target position in response to a zero-offset target moving rightward at 20 deg/s (dashed green line). Bottom row: c. Eye velocity trace (solid blue line) and corresponding eye position and target position in response to a moving target initially located at 5 deg in the right visual field and then moves rightward at 10 deg/s (dashed green line).

3.4.3 Prediction of eye movement in Hand-eye coordination

Figure 3.6a shows model predictions of eye movement in hand-eye combination (solid blue) and eyes only (solid red) conditions. Comparing with behavioral data (dashed blue, dashed red lines) observed in the experiment (Figure 3.6b), eye movement endpoints in the task show that subjects undershoot target with saccadic eye movements when the

task goal is to reach the target with the hand (top panel in Figure 3.6b). However, the undershooting disappeared when the task goal was to fixate the target with eye movements (bottom panel in Figure 3.6b). Infomax predictions (Fig. 3.6a) of optimal eye positions for the hand-eye condition (solid blue) and eye only (solid red) are consistent with this observation from the behavioral data (dashed blue and red).

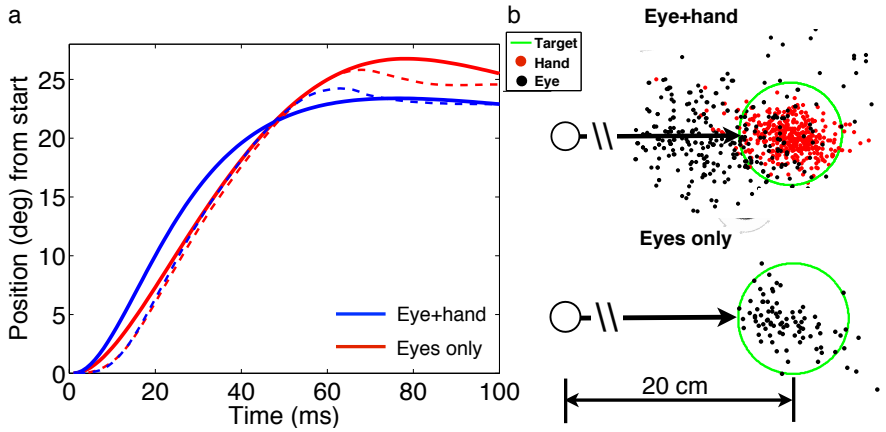


Figure 3.6: Eye movements in the reaching task. a. Comparison between infomax prediction of optimal eye positions (solid lines) and behavioral data (dashed lines). b. Eye movement endpoints in Eye+hand condition (top panel) and in Eyes only condition (bottom panel). Green circle is the target; red dots are hand endpoints in Eye+hand condition; black dots are eye endpoints.

3.5 Discussion

We showed that saccadic eye movements emerge as the solution to an information maximization problem with sensors that have a limited field of view and limited temporal resolution. The information maximization principle explains the velocity profiles observed in saccadic eye movements as well as previous principles, including minimum end-point variance [23] and minimum time [25]. More importantly, information maximization explains eye movement phenomena that were beyond the scope of previous models. We showed that, for moving targets, infomax generates both saccades and

smooth pursuit eye movements. When target onset location is close to the initial eye fixation (foveal), infomax predicts smooth pursuit eye movement which closely tracks target positions. When the target appears in a peripheral location, infomax predicts a catch-up saccade followed up by smooth pursuit eye movement. Qualitatively, this behavior was observed in empirical studies [34, 35]. While previous models [36] explained smooth pursuit from the point of view of minimizing tracking errors, here we explain both saccadic movements and smooth pursuit from the point of view of maximizing information about the location of a target.

We designed an experiment in which target tracking and information maximization make different predictions. Subjects were instructed to reach a target with their hands (Hand condition) or with their eyes (Eye condition). Subjects were rewarded based on the endpoints of hand movements or eye movements, respectively. For the Eye condition, subjects made eye movements as predicted both by the infomax approach and by the target tracking approach. However, for the Hand condition, eye movements undershot the target by $\sim$2.5 degrees. This result was predicted by the information maximization approach but contradicted the target tracking models. According to the infomax model, the reason why people undershoot in the Hand condition but not in the Eye condition is that moving the eyes close to the target does not improve the accuracy of the hand motion.

It should be noted that the infomax model generates closed-loop control policy for the eyes. At first glance, this might seem an undesirable feature since saccades are widely believed to be open-loop, ballistic movements. However, due to the limited temporal bandwidth of our eye model, when the eyes move quickly they provide very little visual information, and thus virtually operate in open loop mode. The decision to move slowly in closed loop mode, or quickly in open loop mode, can be seen as the result of optimizing a common information maximization principle.

3.6 Conclusions

From a Bayesian point of view, learning is equivalent to making inferences based on the information gathered by the sensors. Efficient learners are thus those that control their sensors so as to maximize the expected value of information. Here, we showed that a wide range of properties of the oculomotor system, including the velocity profiles of saccades, the transition between smooth pursuit and saccadic movements, and eye hand coordination in reaching tasks can be explained from the point of view of information maximization. In summary we showed that, by considering that the occulomotor system has evolved to be a very efficient real-time learning machine, one can make sense of a wide range of phenomena that were previously addressed using different principles or that were beyond the scope of previous models.

It should be noted that our work is agnostic with respect to brain implementation issues. For example, while we show that saccadic movements and smooth pursuit movements serve a common goal (information maximization) it is perfectly plausible for the two forms of movements be controlled by different brain systems. It is also possible that, as recently suggested [35, 37], saccades and pursuit are two outcomes of a single sensorimotor system. Regardless, our work suggest that the brain systems involved in oculomotor control have evolved to serve a common computational principle: efficient, real-time learning.

3.7 Acknowledgments

This research was supported by NSF IIS 0968573 and NSF IIS 0808767.

Chapter 3, in full, is a reprint of the material as it appears in the Proceedings of the International Conference on Development and Learning and Epigenetic Robotics. Talbott, W, Huang, H, and Movellan, J, 2012. The dissertation/thesis author was the

primary investigator and author of this paper.

Chapter 4

Visual Perception of Inertial Affordances: Computer Simulation

We present a Model Predictive Control approach to visual affordances. Under this framework, visual information is used to estimate the inertial parameters of internal models of physical objects. Model Predictive Control algorithms then generate and update motor control policies based on the existing internal models. Finally, the torques and accelerations observed while applying the control policies provide a training signal to refine the mapping between visual features and inertial model parameters. We show that the proposed approach models the results of existing behavioral experiments, suggests and makes predictions for new experiments, and is amenable for implementation in humanoid robots.

4.1 Introduction

Consider the two objects in Figure 4.1. Most people have not encountered these objects before, yet they have strong intuitions about which object is better for mashing

potatoes and about how to grasp it for doing so. This is an example of a visual affordance, our intuition of how to interact with a novel object based on visual inspection. There is experimental evidence that infants already use visual affordances when interacting with objects. In a classic experiment, [38] showed that by 9 months of age infants estimated inertial properties of objects based on visual information. Rods of different sizes were given to infants at arm's length, so that when the experimenter released the object the infant's arm either dropped, raised, or remained stable. Infants soon learned to predict the weight of the rod from its length, as shown by the fact that the initial arm-drop decreased as new rods were presented. After this learning had occurred, infants were given a hollow decoy rod that broke the learned relationship between length and weight. The infants' arms lifted up significantly, indicating that the infants had learned a visual affordance: prior experience taught them to modulate the forces to be applied to a new object in response to the visual perception of that object.

In this paper, we explore a computational approach for how visual affordances may be developed and applied in the context of motor planning and motor control. We focus on perception of the inertial properties of objects (e.g., weight, center of mass, moment of inertia). We propose an approach that can model the phenomena from [38], suggest and predict the results of novel experiments, and be implemented in physical robots.

The approach we propose has three main components:

1. A visual system that predicts inertial properties of objects based on their visual features.

2. A proprioceptive system that estimates inertial properties of objects from observed joint torques and resulting accelerations. The proprioceptive system is used to update the predictions of the visual system.

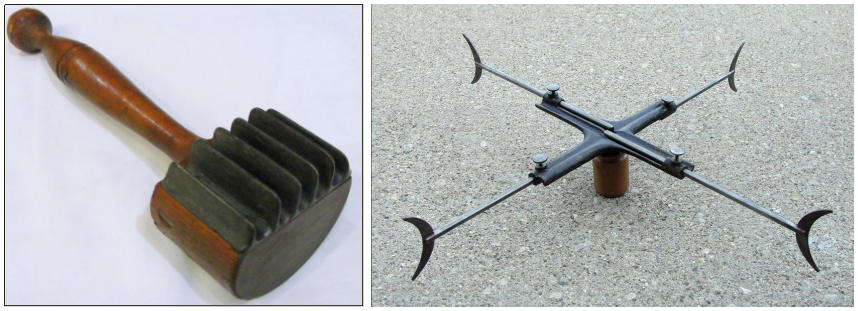


Figure 4.1: Two uncommon objects.

3. A model predictive controller that can both run internal simulations of object dynamics for anticipatory choices, and control behavior in the physical world.

4.1.1 Prior Work

[39] presented one of the pioneering approaches to robotic affordance learning. In their approach, a robot poked objects and observed whether or not they rolled. This allowed the robot to learn to predict whether new objects will roll based on visual information alone. Other approaches focused on learning to choose from a discrete set of actions based on the results of these actions on visually similar objects [40] [41] [42] [43]. For example, in [40], a robot performs predefined movements with color-coded tools to move a puck on a flat surface. The results of each movement are recorded, and the previous results are used to select behaviors to move the puck to desired locations. One limitation of these approaches is that they do not generalize past the actions in their pre-defined sets.

[44] argues for an approach similar to ours: rather than classifying a discrete set of affordances directly from visual features, the approach first makes inferences about intermediate object properties, such as material or size. Then, the presence or absence of a set of affordances (pushable, liftable, *etc.*) is classified using the intermediate features.

Our approach differs from this previous proposal in that, instead of predicting specific affordance labels, we learn to map visual features into internal models of objects. These internal models are then used within a Model Predictive Control (MPC) approach to generate control policies for a wide range of tasks and situations.

There is evidence that humans use internal models of their own bodies and of external objects in order to formulate motor control plans [45]. There is also evidence that the cerebellum plays an important role in the formation adaptation and real time use of these internal models. For example [46] had subjects in an fMRI scanner learn to track a target with the cursor of a computer mouse with a rotated coordinate frame. The cerebellar activations showed one pattern that was proportional to the error between the mouse cursor and a tracking target, and another pattern that showed increased activation even after the error decreased to baseline levels. They propose that this activation is evidence of an internal model being learned and remaining active when the task is performed.

In the next sections, we formalize the problem we are trying to solve, specify the proposed approach, and run computational experiments to explore how the approach behaves in different conditions of interest.

4.2 Problem Formalization

The goal of our approach is to explain phenomena like the one illustrated in Figure 4.1, in which people can make predictions for how to choose and use novel objects for a wide range of tasks. We formalize this problem from the point of view of MPC: we aim to develop closed loop control policies for articulated bodies (robots) that use rigid objects (e.g., tools) to achieve goals. These control policies are developed by using by using optimal control methods applied to internal models of the world. Our goal is

for the visual appearance of the objects to modulate the resulting control policies in an intelligent manner, *i.e.*, a manner that is optimal with respect to a well defined function.

4.2.1 Robot Dynamics

The dynamics of a rigid articulated body, such as a robot, follow the standard equation of motion

$$M(\theta_t)\ddot{\theta}_t = \tau_t + N(\theta_t, \dot{\theta}_t) \tag{4.1}$$

where θ_t is the vector of joint angles at time t, $\dot{\theta}_t$ the angular velocities, $\ddot{\theta}_t$ the angular accelerations, $M(\theta_t)$ is the moment of inertia matrix of the entire articulated body, τ_t the vector of torques applied by the rotational joint actuators, and $N(\theta_t, \dot{\theta}_t)$ is the vector of gravitational, friction and Coriolis/Centripetal forces.

We model a stably grasped object as a change to the robot itself. An object attached to the robot by a grasp changes the robot's geometry, inertial properties and equation of motion. Here, we use the parameter λ to represent the inertial properties of the grasped object. The equation of motion for the resulting robot-object system is

$$M(\theta_t, \lambda)\ddot{\theta}_t = \tau_t + N(\theta_t, \dot{\theta}_t, \lambda) \tag{4.2}$$

4.2.2 Control Policy

Formally, a control policy c is a mapping between robot states (angles, and angular velocities) and actions (torques applied to each joint), i.e.,

$$\tau_t = c(\theta_t, \dot{\theta}_t) \tag{4.3}$$

Goals are formulated in terms of a scalar function that captures the expected reward resulting from using a control policy over a finite period of time $[0, T]$

$$\rho(c) = \int_0^T E[R_t|c]dt \tag{4.4}$$

where R_t is the reward rate and T is the terminal time. The reward rate R_t is simply a function that expresses the desirability of achieving a robot's state at a particular point in time t. For example, R_t could be the Euclidean distance, at time t, between a target location and the tip of a robotic finger.

The problem is to find a control policy $\hat{c}$ that maximizes $\rho(c)$ subject to the dynamics in (4.2), where the inertial properties λ of the object are not completely known. Moreover we want for the control policy to be modulated by the visual appearance of the object, in a principled manner. In the case where the robot can use multiple tools to achieve a task, the robot can generate a policy for using each tool. Then, it can choose the tool with the highest expected reward.

4.3 Proposed Approach

We formalize the lack of knowledge of the object's inertial properties by using a probability distribution, $p(\lambda_t \mid v_t, s_t)$, over inertial parameters of the object (weight, center of mass, moment of inertia). Here λ_t represents the inertial properties of the object observed at time t. v_t represents the visual information observed while manipulating objects up to time t, and s_t is the proprioceptive information (joint torques and angular accelerations) obtained while manipulating objects up to time t. This probability distribution is the mathematical expression for the visual perception and learning of inertial affordances.

Suppose at time t the robot is presented a new object. Prior experience observing

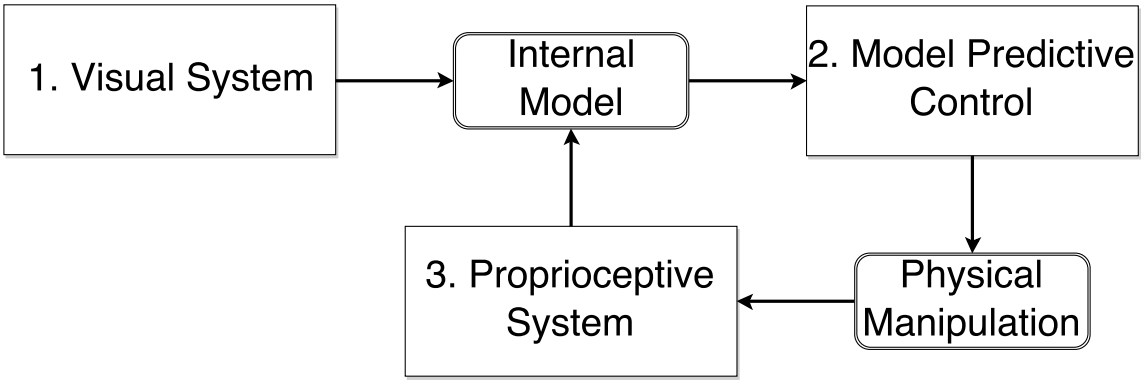


Figure 4.2: Diagram of the proposed approach. The numbered components correspond to the components described in the text.

and manipulating objects (v_t, s_t) allows the robot to predict probable inertial parameters for the current object: $p(\lambda_t \mid v_t, s_t)$. Based on this prediction, the robot formulates a control policy to achieve a goal using the object. The control policy is implemented, and as a result of it the robot grasps the object and manipulates it. During the manipulation, the robot keeps track of the joint torques and resulting joint accelerations. This new data is used to update the probability distribution of inertial parameters given visual features.

Figure 4.2 shows the different components of the proposed approach: (1) A visual system generates a probability distribution of object inertial properties based on their visual properties (*e.g.* color, texture, shape). Given a new image, the system generates a probability distribution over object geometry and density. This is then used to compute a probability distribution over probable inertial object properties (weight, center of mass, moment of inertia) given their observable visual features. (2) A Model-Predictive Control system that generates control policies to achieve goals given the current probability distribution over object inertial parameters. The control system uses an MPC approach, i.e., it utilizes internal models with the known inertial parameters of the robot, plus the available distribution of inertial object parameters to formulate control policies. These control policies are then applied in the physical world. The results are

used to update the internal model. (3) A proprioceptive system that infers the posterior distribution over the inertial properties of objects given: a prior distribution over inertial object parameters, observed torques applied to the robot joints, and the resulting angular accelerations observed on the joints. This is used as a teaching signal to update the probability distribution generated by the visual system.

4.3.1 Estimating Material Density from Experience

A key aspect of our approach is the ability to use prior experience seeing and manipulating objects to generate reasonable control policies for novel objects and tasks. This is achieved by estimating the density of the parts which comprise objects. Basically when a new object is observed, we form a 3D model of the object as a collection of 3D ellipsoids. Algorithms for inferrring 3D ellipsoids from 2D images are well known in the computer vision community [47]. This geometric model, in combination with the proprioceptive information (joint torques and angular acceleratiors) observed while maniputating the object provides enough information to estimate the density of the different object parts. Once the densities of the object parts are known (via proprioceptive data), the mapping between the visual appearance of parts (e.g., color and texture) and material densities can then be learned using standard machine learning methods. This allows to readily generate internal dynamical models of new objects. In this section we show that once the 3D geometry of the object is known, the density of its parts can be estimated from the proprioceptive information (joint torques and angular accelerations) using a simple linear regression model.

As said before we model rigid objects as a set of n uniform-material ellipsoids, each with visually-estimated geometry parameterized by center μ_i and axis lengths a_i,b_i, and c_i. While manipulating the object the robot's propioceptive system provides information about the torques applied to each joint and the resulting accelerations.

Let τ_t be the vector of joint torques observed at time t. Assuming an inverse dynamics model of the robot is available, we can decompose τ_t into two components, a component due to the dynamics of the robot itself, and a component caused by the grasped object

$$\tau_t = \tau_t^r + \tau_t^o \tag{4.5}$$

It is well known that the articulated bodies dynamics equation (4.2) is linear with respect to the inertial parameters of the system (see Appendix A). Thus we can write the dynamics in the following linear form

$$\tau_t^o = K_t \gamma \tag{4.6}$$

where, K_t is a known matrix function of the robot-plus-object kinematics and joint accelerations, and γ is a vector containing the unknown the inertial parameters of the object.

We next show that γ is itself a linear function of the vector of object densities δ, which we aim to estimate. The 10 inertial parameters of the object, $\gamma = (m, mx, Vec_6[I])'$, are:

- the object's mass, m
- the object's center of mass times its mass, mx
- the object's inertial matrix, I, in the object's frame, with origin o as reference. $Vec_6[I]$ extracts the 6 elements from the upper triangle of the symmetric matrix I.

First, we write I in a form linear on the masses m_i of each component. Using the

parallel axis theorem, we can write the total inertia of the object as

$$I = \sum_i I_i - m_i [d_i]_\times^2 \tag{4.7}$$

where d_i is the vector from the origin of the object's frame, o, to μ_i, m_i is the mass of ellipsoid i, I_i is the inertia of ellipsoid i around its center of mass, and $[d_i]_\times$ is the skew-symmetric cross-product matrix constructed from the vector d_i. Since we know the geometry of ellispoid i, and the form of the inertial matrix for ellipsoids, we can write

$$I_i = \begin{bmatrix} \frac{1}{3}(b_i^2 + c_i^2) & 0 & 0 \\ 0 & \frac{1}{3}(a_i^2 + c_i^2) & 0 \\ 0 & 0 & \frac{1}{3}(a_i^2 + b_i^2) \end{bmatrix} m_i \tag{4.8}$$

$$= G_i m_i \tag{4.9}$$

and

$$I = \sum_i I_i - m_i [d_i]_\times^2 \tag{4.10}$$

$$= \sum_i (G_i - [d_i]_\times^2) m_i \tag{4.11}$$

Also,

$$x = \frac{\sum_i m_i d_i}{m} \tag{4.12}$$

$$m = \sum_i m_i \tag{4.13}$$

We want to estimate

$$\delta = \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_r \end{bmatrix} \tag{4.14}$$

where δ_i is the density of material i. The color of each component indicates which of a set of r materials makes it up. This information gives an $n \times r$ matrix A where element A_{ij} is v_i if object component i is made of material j, and 0 otherwise. Here, v_i is the volume of component i, which is known, and allows estimation of the density of materials rather than the mass of the components, since

$$\begin{bmatrix} m_1 \\ \vdots \\ m_n \end{bmatrix} = A \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_r \end{bmatrix} \tag{4.15}$$

Combining (4.11), (4.12), and (4.15), we can write

$$\gamma = HA\delta \tag{4.16}$$

where

$$H = \begin{bmatrix} \mathbf{1}_n \\ d_1 \dots d_n \\ \text{Vec}_6[G_1 - [d_1]_\times^2] \dots \text{Vec}_6[G_n - [d_n]_\times^2] \end{bmatrix} \tag{4.17}$$

and $\mathbf{1}_n$ is a row vector of n ones. Finally, we have

$$\tau_t^o = K_t HA\delta \tag{4.18}$$

Here the torque vector τ_t^o, and the K_t, H, A matrices are known. Thus we can infer the desired vector of object densities using simple linear regression methods. In practice, we use Bayesian linear regression methods to update, after each trial, a Gaussian estimate of the density of a set of materials. Here, K is a stacked matrix containing K_t for all observed timesteps in a trial, and similarly τ^o is a stacked vector of all τ_t^o. For a prior belief $\mathcal{N}(\mu_\delta, \sigma_\delta)$ and our data K and τ_t^o, we get the parameters of the posterior distribution:

$$\bar{\sigma}_\delta = (\sigma_\delta^{-1} + \eta(KHA)'(KHA))^{-1}$$
$$\bar{\mu}_\delta = \bar{\sigma}_\delta(\sigma_\delta^{-1}\mu_\delta + \eta(KHA)'\tau^o) \tag{4.19}$$

This approach contains a parameter, η, for the variance of the observation noise. In our case, we can think of the inverse of this parameter, roughly, as a learning rate.

4.3.2 Finding the Control Policy

Once we have internal models of the objects, we use optimal control methods, applied to those models, to generate optimal policies. In this paper we generate Minimum Angular Acceleration policies for articulated bodies [48]. These policies have the advantage of generating human-like trajectories, adhering to a well-defined optimality framework, and being computationally efficient. In order to handle the uncertainty over the inertial parameters of grasped objects, we sample from the available distribution of parameter values, and compute the optimal control policy given the sampled parameter. The control policy is applied to the physical world and the results are used by the proprioceptive system to improve the probability distribution of inertial parameters from the visual component.

4.4 Computer Simulations

We run 4 computer simulations of the proposed approach. The goal of the simulations is to gain insights about how the approach behaves before we implement it in a physical robot. The first simulation focuses on the Mounoud and Bower (1974) study previously described [38]. The second simulation describes an additional experiment suggested by the approach and specifies the predictions made by the model. The third and fourth simulations show that, under the proposed approach, visual affordances can generalize to novel objects and tasks.

For the simulations, we use a 7 degree-of-freedom model of the human arm. The first joint (shoulder) has 3 degrees of freedom, the second joint (elbow) has 2 degrees of freedom and the third joint (wrist) has 2 degrees of freedom. The links are simulated as ellipsoids with the density of ice. Gravitational forces use the Earth surface standard. The simulator is implemented in Matlab using the Gaussian mechanics approach to articulated bodies [49]. The equations of motion are integrated using a the Euler method. Our implementation was validated using the Matlab Robotics Toolbox [50].

4.4.1 Simulation I: Modeling Mounoud and Bower's 1974 study

[38] showed that infants older than 9 months use visual information to adapt their motor behavior towards novel objects. Here we simulate their experiment using the following procedure: On each trial, an ellipsoidal object of a given length and material is presented to the robot arm. The robot reaches out horizontally and is given the object (the object is attached to the hand). The desired behavior is to hold the object fixed at the same height where it was given.

The first 4 objects, which we call the training objects, are all of the same density, but varied in length. The last object, which we call the decoy, has a much lower density

but is length-matched to one of the training objects. The training objects are presented in order from shortest to longest. The color and axis lengths of the ellipsoids are used as visual features. There are well known approaches from the computer vision literature to estimate geometric properties of 3D ellipsoidal objects from 2D projections, *e.g.* [47]. Here, we assume that these or other algorithms were used to estimate these 3D geometric properties.

On each trial, the proprioceptive system uses the observed torques and accelerations measured while holding the object to estimate the inertial properties of the object. This estimate is used to update the probability distribution of object density given its observed color. On the next trial, a new object is presented. The most probable density given the observed visual features is used to estimate the object's inertial parameters, λ. This estimate is used by the controller to generate a policy to keep this new object as level as possible. Figure 4.3 shows the magnitude of the drop in arm height for each of the first 4 objects. As observed in the Mounoud and Bower experiment, we found a decrease in the magnitude of arm drop between trials. This is due to the fact that on each trial, the model improves its estimates of the objects' weights based on the observed visual properties and the inferred density (see Figure 4.3).

Figure 4.4 shows the height of the held object through time. As observed in the [38] experiment, by trial 4 the object is held near the target level, but the arm lifts the decoy object much higher than the target level. This result confirms that the computational approach outlined here can reproduce the results of the Mounoud and Bower experiment.

4.4.2 Simulation II: Center of Mass

The Monoud and Bower experiment was designed to test visual perception of one inertial property of objects (their weight). Here, we explore a generalization of their original experiment designed to test visual perception of a different inertial property

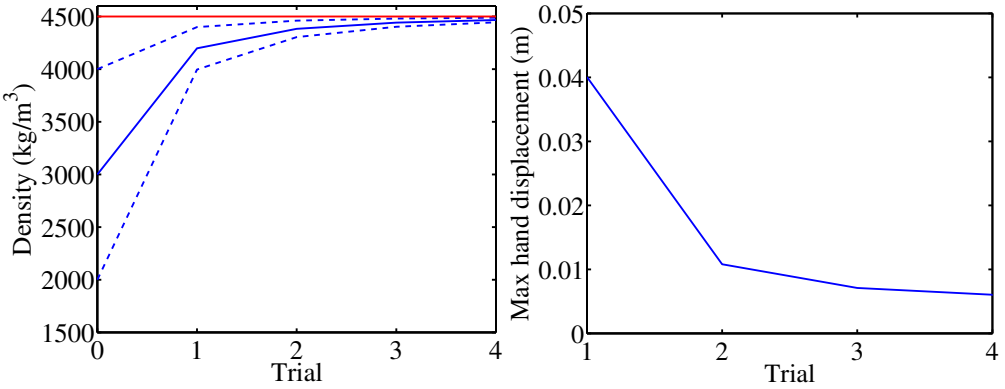


Figure 4.3: Learning to estimate material density. (Left) Estimated distribution, $\mathcal{N}(\mu, \sigma)$, of material density after each trial. Solid line shows μ. Dashed lines show $\pm\sigma$. Trial 0 shows the initial belief. Horizontal line shows true density. (Right) Magnitude of arm drop for each trial. The drop is reduced from trial to trial, even though the mass of the objects increases.

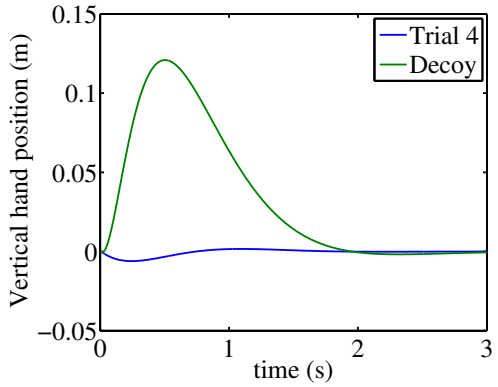


Figure 4.4: Response to decoy object. The vertical position of the object over 3 seconds for trial 4 and the decoy object. The decoy object, which is much less dense than its appearance suggests, is mistakenly lifted above the desired height, which is at 0m.

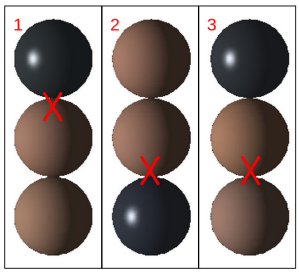


Figure 4.5: Objects used for Simulation II. The three objects used for Simulation II. Object center of mass is indicated by an X. Object 3 is the decoy object: it has the appearance of object 1, but the inertial properties of object 2. All objects have the same mass.

(center of mass). We propose a novel experiment and run computer simulations to specify the predictions made by our model. For this experiment, we used 3 different objects, each of which is made of 3 aligned ellipsoidal components (see Figure 4.5). The 3 objects weigh the same, however they have different centers of mass. The first object has two components made of wood (low density) near the hand, and one component made of steel (high density) farthest from the hand. The second object is the reverse, i.e., the steel is near the hand, and the wood is farther from the hand. The third object, the decoy in this experiment, has the appearance of the first object, but the density distribution of the second. Because its density and appearance are inconsistent with the learned affordance, we expect behavior similar to the decoy response from the first experiment.

As in Simulation I, we present each object to the robot when the arm is horizontal, and observe the drop or lift relative to the initial height. The training objects, objects 1 and 2, are presented 3 times each, and the decoy object is presented once. The density estimate of each material is updated after each trial. Figure 4.6 shows the estimates of the material densities on a trial by trial basis.

Figure 4.7 shows how the robot arm behaves in response to the decoy objects. The results are similar to Simulation I, where an exaggerated lift of the decoy object is observed.

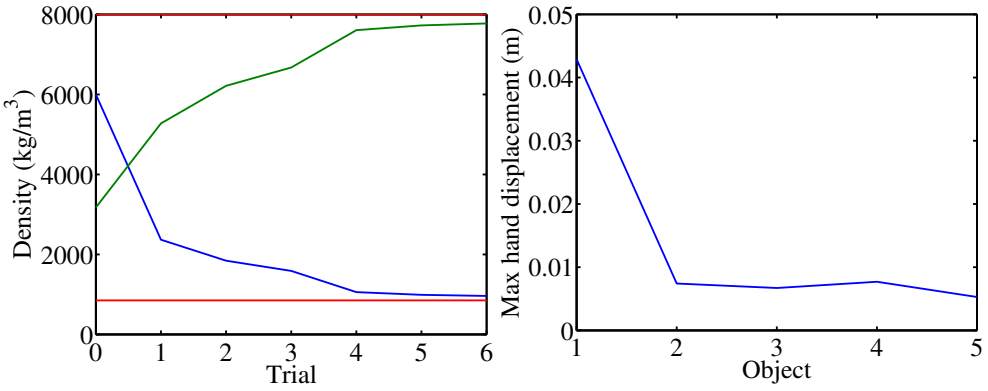


Figure 4.6: Estimates of material density. (Left) Estimates of each material's density after each trial. Trial 0 shows the initial belief. Horizontal lines shows true density. Although neither material is given to the robot in isolation, it is able to estimate the density of both materials. (Right) Magnitude of arm drop for each trial.

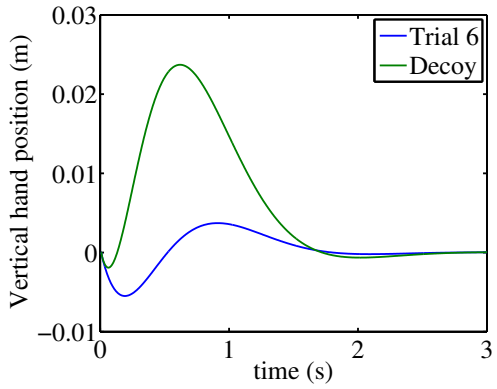


Figure 4.7: Response to decoy object. The vertical position of the object over 3 seconds for trial 6 and the decoy object. The decoy object is mistakenly lifted above the desired height, which is at 0m.

Figure 4.8: The novel object used for simulations III and IV. The object is made of the same materials as the object from simulation II, but in a different shape.

The exaggerated response to the decoy, although smaller than when the decoy's weight is overestimated as in Simulation I, is still observed even though the training and decoy objects have the same weight. We also see that dynamic properties of a compound object can be estimated from the visual features of the object, and used to plan behavior. The inertial properties of objects can give the robot important information about how an object will behave in particular situations. The final two simulations explore how this information can help the robot generalize its experience to novel tasks with a novel object.

4.4.3 Simulation III: Choosing a Grip for Hammering

In this simulation, the robot's goal is to efficiently hit a nail with sufficient force using the object in Figure 4.8, which is novel to the robot. The robot never interacts with this object directly, but rather uses its experience with similar objects to predict how to use the novel object. This is analogous to the human intuition about how to use the objects in Figure 4.1 despite never interacting with them. Specifically, the robot chooses how to use the object by choosing between grasp locations, each of which results in a different hammering behavior.

The robot makes choices between grasps based on running its estimate of the object's inertial properties through its MPC component. Although the framework allows comparison of multiple grasp locations, we focus on comparing two: one grasp, g_s, holds

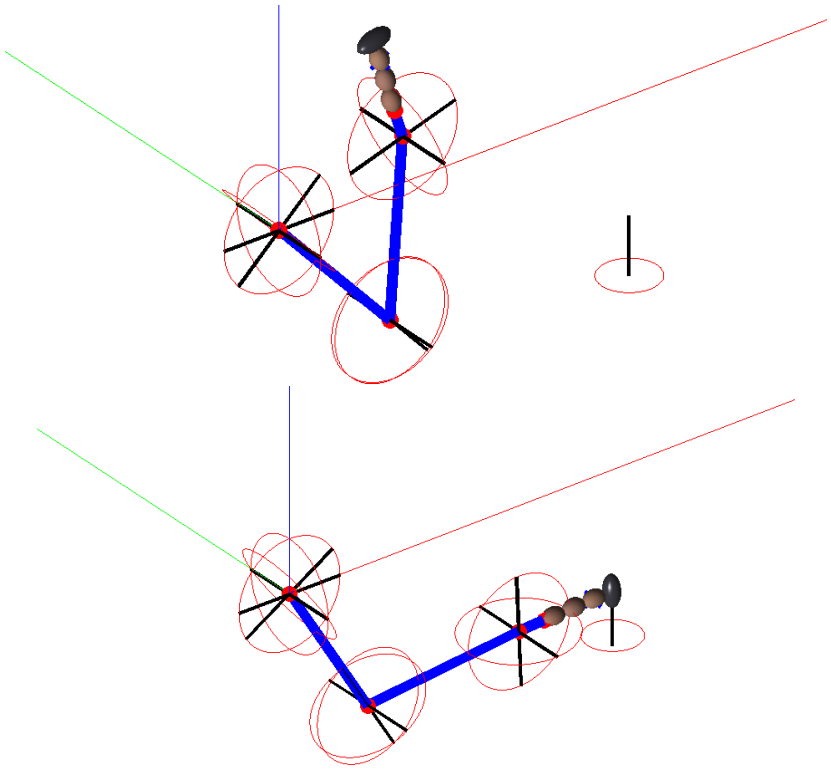


Figure 4.9: Illustration of the hammering behavior. The top panel shows the start point. The bottom panel shows the hammer striking the simulated nail.

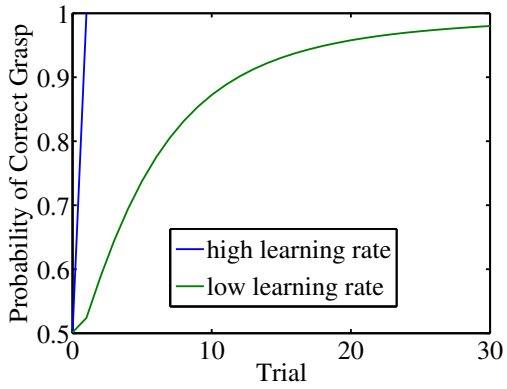


Figure 4.10: Choosing the correct grasp for hammering. The probability of choosing the correct grasp for hammering with a novel test object efficiently as a function of amount of experience with different objects. Shown for both a low and a high learning rate.

the hammer by the steel end, and the other, g_w, holds the wooden end.

A trial follows these steps:

1. The robot is given a training object (not the decoy) from Figure 4.5 and must hold it as level as possible.

2. The robot's belief about material density is updated.

3. The robot compares grasps for hammering with the novel object (Figure 4.8). The robot does the comparison using internal model simulations. It doews not perform the hammering task or otherwise interact physically with this novel object.

The MPC component of the proposed framework generates hammering control policies (Figure 4.9) for each grasp using the inertial model of the object generated from the estimated material density. These control policies give rewards for each grasp, which we will call r_w and r_s for grasps g_w and g_s respectively. These rewards are inversely proportional to the squared torque required to generate the desired force. For this task, the optimal choice is g_w, which is to hold the wooden end of the object. Gravity acting on the higher-density, steel end of the hammer generates more downward torque when

that steel end is distant from the hand than when it is more proximal. When g_s is selected, the robot must itself generate the additional torques that gravity provides in g_w, which costs energy. Because the task is to hit the nail with high force, choosing g_w results in a net energy savings over g_s.

We model the robot choice response using a softmax function, popular in the response choice literature [51]. Figure 4.10 shows the probability of a correct choice, p, as a function of trial number for different values of the learning rate η^{-1} from (4.19). The probability p is calculated as the softmax function applied to the rewards:

$$p = \frac{\exp(\alpha r_w)}{\exp(\alpha r_w) + \exp(\alpha r_s)} \tag{4.20}$$

where α is a constant that accounts for the scale of the rewards. This figure highlights that sufficient information is contained in the robot's interactions with the training objects to learn the correct behavior on a novel object in a single trial, but that the framework can also arrive at the correct answer when it learns more slowly.

4.4.4 Simulation IV: Choosing a Grip for Tapping

In this final simulation we require the robot to use a novel object (Figure 4.8) for a novel task. Rather than hammering efficiently by generating a large force at a small cost, the goal is to tap a point with the object by producing a small force at a small cost. This simulation compares the same two grasps as Simulation III, grasping the steel end g_s, and grasping the wooden end g_w. Gravity acting on the higher-density, steel end of the hammer again generates more downward torque when that steel end is distant from the hand than when it is more proximal. But, in this simulation, the goal is to tap the point with a small force. So, the additional torques from gravity when grasping the wooden end must be counteracted by the robot, which consumes energy. The optimal choice is g_s.

A trial follows the same steps as in Simulation III:

1. The robot is given a training object (not the decoy) from Figure 4.5 and must hold it as level as possible.

2. The robot's belief about material density is updated.

3. The robot compares grasps for tapping with the novel object (Figure 4.8). The robot does not perform the tapping task or otherwise interact with this novel object.

Figure 4.11 shows the probability of choosing the correct grasp as a function of number of trials for different values of the learning rate η^{-1} from (4.19). Note that the initial belief of material density was chosen to give a 0.5 probability in Simulation III, and the same initial belief is used here. Because the task is different, the initial belief results in a different probability of being correct. But again, we see that the robot can learn to predict which grasp is best as soon as after one trial with the training objects.

Together, simulations III and IV highlight the ability of the framework to use prior experience with similar objects to generalize competently to novel objects, and to using those objects for novel tasks. This ability comes from the underlying estimation of the inertial parameters of the novel objects, which are used as a compact representation of how the object will react to forces that the robot can apply.

4.4.5 Sensitivity Analysis

Since our goal is to implement this framework in a physical robot, we would like to estimate how sensitive the framework is to noise. Here, we examine the effect of adding zero-mean Gaussian noise to three signals: the joint accelerations and torques from the proprioceptive system, and the estimated geometry of the object components.

We are interested in how the results from the simulations hold up subject to increasing levels of noise. For this analysis, we focus on simulation I. Specifically, we

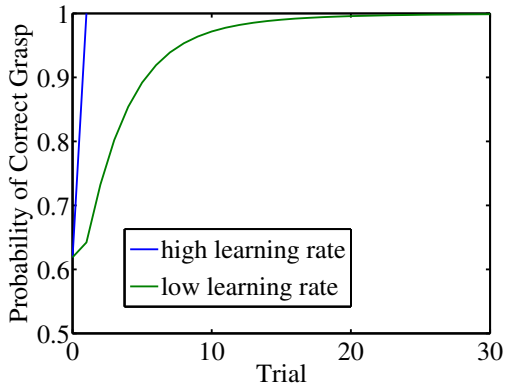


Figure 4.11: Choosing the correct grasp for tapping. The probability of choosing the correct grasp for tapping with a novel test object efficiently as a function of amount of experience with different objects. Shown for both a low and a high learning rate.

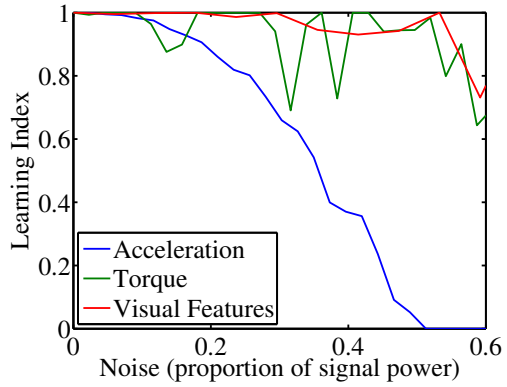


Figure 4.12: Learning effect sensitivity analysis. An index of whether the behavioral result is reproduced as a function of the amount of noise in the joint accelerations, torques, and object geometry. The index is the sum of displacement reduction between non-decoy trials, and the displacement increase for the decoy trial, scaled to between 0 (no evidence of learning) and 1 (evidence equal to or greater than in the noiseless case).

repeat the simulation with different levels of noise, and examine whether the indicator of learning is present: the displacement of the arm is reduced from trial to trial. Also, we see whether the displacement is again increased when the decoy object is presented.

The acceleration signal is important for the proprioceptive system that generates estimates of the density of a held object because it is used to compute K_t in (4.6). This matrix depends non-linearly on the accelerations. Figure 4.12 shows the effect of different levels of acceleration noise on the evidence of learning. The Learning Index plotted in Figure 4.12 is calculated by adding the reduction in displacement from the consistent objects to the increase in displacement from the decoy object. This index is scaled between 0 and 1, where 0 means no evidence for learning is observed, and 1 means evidence equal to or greater than the noiseless case is observed. This index increases when both the consistent objects are learned (reducing the displacement between trials), and when the decoy object is mistakenly lifted (increasing the displacement. When the index is 0, the learning is not observed. The figure shows that, for the acceleration signal at low levels of noise, the effect observed in simulation I is preserved. As the noise increases, however, the effect decreases. When the noise power is roughly half of the acceleration signal power, the effect disappears.

When noise is added to the torques and the visual estimates of the object's axes lengths, we observe a lower impact on the effect of learning. Figure 4.12 shows that for all noise levels investigated, the effect was still present.

4.5 Conclusion

We proposed a computational framework for visual perception of inertial affordances. The approach combines three modules: (1) a visual system that predicts inertial properties from visual information. (2) a proprioceptive system that uses observed forces

and accelerations to teach the visual system. (3) A Model Predictive Controller that uses internal models to generate control policies.

The proposed approach is designed to reproduce humans' ability to use visual information to plan how to use novel objects in novel ways. Our approach is based on the use of internal models of our own bodies and of external tools. Model-based approaches have become popular for solving complex tasks in humanoid robots [52]. In addition, there is mounting evidence that internal models are part of the machinery for motor control used by the brain [45].

Our simulations show that the proposed approach can replicate behaviors observed in human experiments, and make predictions for new experiments. The approach can also be implemented in robots to use visually perceived affordances in a manner similar to the way humans do.

Acknowledgments

This research was supported by NSF IIS 0968573 and NSF IIS 0808767.

Chapter 5, in part, is a reprint of the material as it was presented to the Affordances in Vision for Cognitive Robotics Workshop, Talbott, W and Movellan J, 2014. Chapter 4 is currently being prepared for submission for publication of the material. The dissertation/thesis author was the primary investigator and author of this paper.

Chapter 5

Visual Perception of Inertial Affordances: Physical Robot

In the previous chapter we proposed a computational framework for learning to perceive the inertial properties of objects. We used computer models to illustrate how the approach explains the Mounoud and Bower [38] weight conservation experiment and predicts results for new possible experiments. These simulations were a first step toward reproducing the experiment on a robot.

In this chapter we implement the proposed approach in a complex humanoid robot named Diego. To do so, we have to modify some aspects of the computational model.

Table 5.1 describes the major differences between the computer simulations in the previous chapter and the physical robot implementation described in this chapter. There are three differences, but they ultimately derive from one key problem: The parametric physics model (PPM) used in the simulations does not match well enough with the data collected from the actual robot. Although the sensitivity analysis on the simulations suggested the approach was resistant to random, unsystematic noise, we find that the data

collected from the robot was systematically different from what was predicted from the physics model. This is likely due to physical phenomena that are difficult to capture with a parametric physics model, like complex friction forces, and forces caused by cables and other artifacts that are part of the robot.

An inverse dynamics model is a method for estimating the joint torques τ that cause desired joint accelerations, $\ddot{\theta}^d$. For a given joint position θ and joint velocity $\dot{\theta}$. In the PPM, the torques can be derived as follows:

$$\tau = M(\theta)\ddot{\theta} + N(\theta, \dot{\theta}) \tag{5.1}$$

where $M(\theta)$ is the inertial matrix, and $N(\theta, \dot{\theta})$ are the gravitational, friction, and Coriolis forces. A PPM can learn estimates for the functions M and N from data, and can therefore be used to compute the torques required to produce a desired acceleration. We collected a dataset $\mathcal{D} = \left\{\theta_t, \dot{\theta}_t, \ddot{\theta}_t, \tau_t\right\}$ from the physical robot, where t indexes the sample time. Then, we used the PPM to estimate $\hat{\tau}_t$, given θ_t, $\dot{\theta}_t$, and $\ddot{\theta}_t$. The correlation between model predictions and observed torques was quite high (r=0.91), but the model is not able to control the robot accurately.

Deep neural networks (DNN) are a machine learning framework for learning arbitrary functions [53]. They consist of layers of simple models of neurons, that take weighted inputs from previous layers, and produce output for the next layer by applying a non-linear activation function to the sum of the weighted inputs. Their components are simple, and they can be trained by stochastic gradient descent, but they have a large representational capacity. Recently, DNN approaches have proven beneficial for various problems (for a review, see [54]). Here, we explore the application of DNN to learning and inverse dynamics model directly from recorded data. When we use a DNN approach instead of PPM, we obtain a higher correlation (r=0.97) between the DNN estimate and

Table 5.1: Differences between the simulated and robotic approaches

	Simulation	Robot
Dynamics Model	Analytical physics	Deep neural network (DNN)
Object learning	Estimate material density	Learn new DNN dynamics model
Object control modification	Geometry estimated from vision plus estimated material density gives analytical physics model	Visual arbitrator network weights output of multiple DNN dynamics models

the recorded data that is good enough to control the physical robot accurately, and to reproduce the Mounoud and Bower experiment introduced in the previous chapter.

This chapter is organized as follows. In Section 5.1, we introduce the robot, Diego, that the experiments are conducted on. Next, we introduce the methods we use to control Diego to follow trajectories: PID control, computed torque based on a parametric physics model, and DNN-based Computed Torque. Finally, we explain the Visual Computed Torque controller, which is how the framework accommodates controlling different objects based on their visual features. In Section 5.3 we present the experiments we conducted to examine the DNN model, and to replicate the Mounoud and Bower experiment from the previous chapter. Section 5.4 discusses the results of the experiments.

5.1 Diego

Diego, in Figure 5.1, is a pneumatic humanoid robot that has 38 potentiometers for measuring joint angles, and 88 controllable valves for actuating the pneumatic cylinders. As described in [55], pneumatic actuators are an attractive choice because of their high force-to-friction ratio, their low cost, their compliance, and their similarity to muscle in terms of their dynamic time constants. However, these advantages come at the cost of increased control complexity. The complexity arises mainly from two facts:

1. Stiff actuators, like the ones found in many current robots, allow independent control of each joint, at the cost of increased energy required to overcome the stiffness and move the joints. In contrast, the human body has joints that are compliant (non-stiff), and pneumatic actuators are compliant as well. Compliant actuators mechanically couple the different joints. In other words, moving one limb creates significant forces in other limbs. Control signals sent to one joint must therefore take into account the movement and forces across the entire body.

2. Pneumatic actuators behave like non-linear series-elastic actuators. In practice, this means that rather than directly controlling the forces that apply to the different joints, like in standard robots, we effectively control non-linear springs that indirectly control the joints.

The following section describes how, in the face of the difficulties, we control Diego's joints to follow trajectories. It first introduces some characteristics of the robot. Then, it discusses the two different modeling approaches outlined in Table 5.1.

Each of Diego's joints is instrumented with 3 sensors: a potentiometer for measuring the joint angle, and two pressure sensors for measuring the pressure in each of the chambers of the pneumatic cylinder that actuates the joint. The joint angle $\theta_{j,t}$ of joint j at time t is related to the potentiometer reading $q_{j,t}$ by the transformation $q_{j,t} = \gamma_j \theta_{j,t} + \beta_j$ where γ_j and β_j are calibration parameters. The 2 pressure sensors give readings $p^e_{j,t}$ and $p^f_{j,t}$ in Volts, where $p^e_{j,t}$ is the pressure reading from the extensor chamber (Chamber 1 in Figure 5.2) that extends the cylinder's rod, and $p^f_{j,t}$ is the reading from the flexion chamber (Chamber 2) that retracts the rod. A pressure reading $p^c_{j,t}$ in chamber c (in Volts), is related to the measured pressure (in Pascals) $\pi^c_{j,t} = \nu^c_j \pi^c_{j,t} + \eta^c_j$ where ν^c_j and η^c_j are calibration parameters.

The linear extension of the rod of the pneumatic actuator drives the joints of the robot. Each chamber of the actuator is connected to two constant pressure sources, atmo-

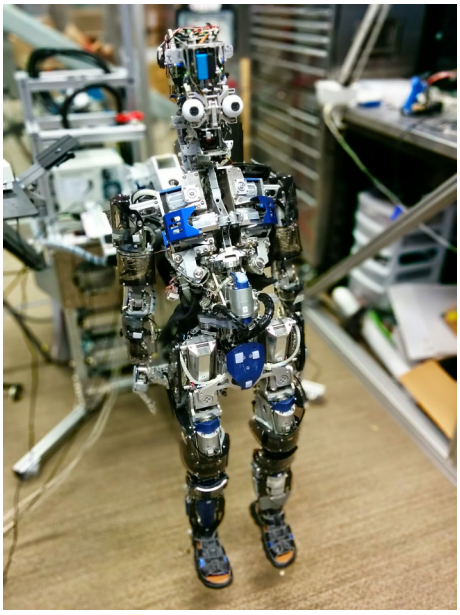


Figure 5.1: Diego, the pneumatic humanoid robot

sphere and compressor pressure, by a valve that determines the areas of the connection ports to each source. The voltage accepted by each valve is between 0 and 10V, and this signal controls the areas of the two ports (one to compressor and one to atmosphere). The area of the ports of each chamber is a function of the voltage that, in the ideal case, looks like Figure 5.3. In this figure, we see that with a control signal of 0V, the chamber is open maximally to the atmosphere source, and minimally to the compressor source. At 5V, both ports are minimally open. At 10V, the port to the compressor is maximally open, and the port to the atmosphere remains minimally open.

For one model of a cylinder, the force $F_{j,t}$ generated by actuator j is related to the difference in pressures between the two chambers as follows, where the time index on the force, pressure, and sensor reading has been dropped for brevity:

$$F_j = a_j(\pi_j^e - \pi_j^f) \tag{5.2}$$

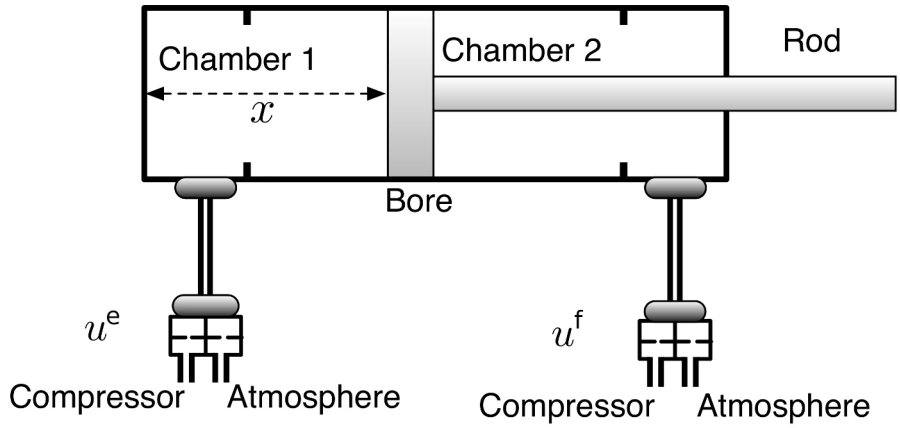


Figure 5.2: Cylinder Model

where a_j is the area of the bore. The torque τ_j generated around the joint is related to F_j by the moment arm $J_{jj}(\theta_j)$ so that $\tau_j = J_{jj}(\theta_j)F_j$. The moment arm is also known as the Jacobian of the joint angle with respect to the rod extension. Note that the torque produced at a given joint angle changes only if the difference of pressures in the chamber change. So, we control the torque around each joint by controlling the ratio of compressor/atmosphere port areas at each valve.

Each joint j has two associated control signals $u^e_{j,t}$ and $u^f_{j,t}$, one for each valve controlling the opposing chambers in the cylinder. These voltage signals range from 0 to 10 Volts. To reduce the degrees of freedom the controller must deal with, we use a single control signal $u_{j,t}$, which we then translate to the two voltages as

$$\begin{aligned} u^e_{j,t} &= 5 + u_{j,t} \\ u^f_{j,t} &= 5 - u_{j,t} \end{aligned} \tag{5.3}$$

As Figure 5.4 shows, if the chambers are at equilibrium pressure, and the control signal $u_t = 0$, the ports to both sources in each chamber will be almost closed (except for a

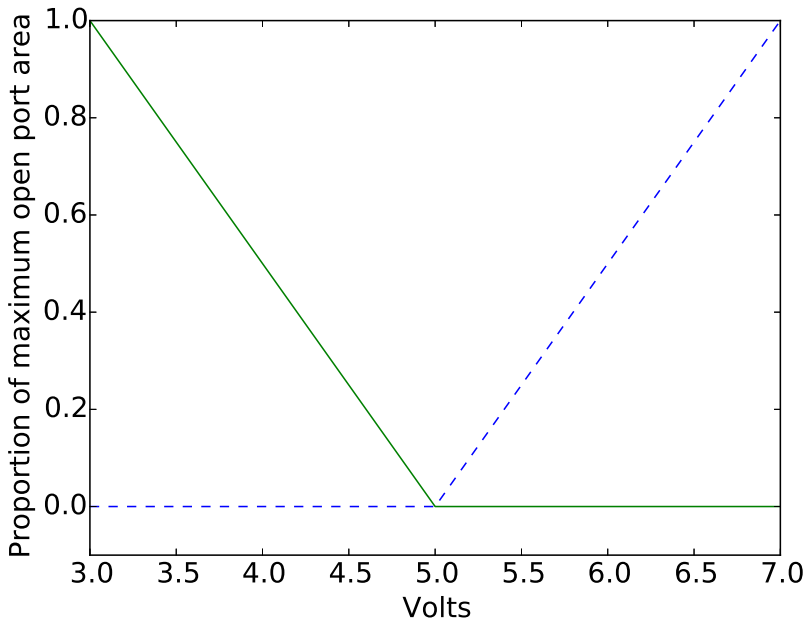


Figure 5.3: Port area as a function of input voltage. Shows the proportion each port is open to a single chamber as a function of voltage, where 1 is fully open, and 0 is fully closed. The solid green line is the port to the atmospheric pressure, and the dashed blue line is the port to the compressor.

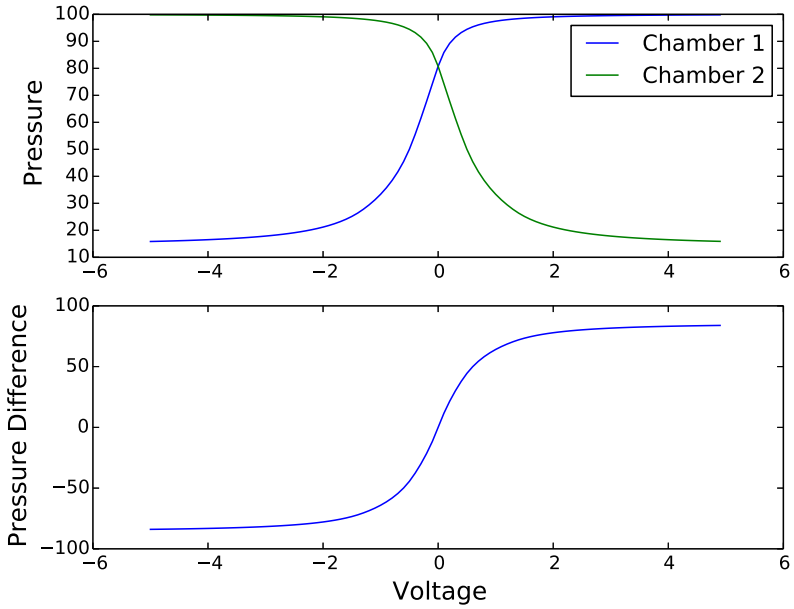


Figure 5.4: Equilibrium pressure as a function of control signal. The x-axis of both the top and bottom graph is the control signal $u_{j,t}$. (Top) Equilibrium pressure in each chamber of the cylinder using independent commands defined in (5.3). (Bottom) Equilibrium pressure difference between the two chambers as a function of $u_{j,t}$

small leakage), which means the pressures will stay near their current value. As u_t is raised higher than 0, the extension chamber will have its ports open more to the compressor than atmosphere, and so will increase in equilibrium pressure. Additionally, the flexion chamber will have its ports more open to atmosphere than compressor, and so will decrease in equilibrium pressure. The resulting pressure difference will create an extension force. The opposite, a flexion force, will be generated when u_t is lowered below 0.

Next, we describe the methods used to control Diego's joints to follow a desired trajectory [55]. A schematic of the method is shown in Figure 5.5. The first controller specifies a desired pressure difference based on potentiometer readings. This desired pressure difference is then given to another controller that issues the commands to the

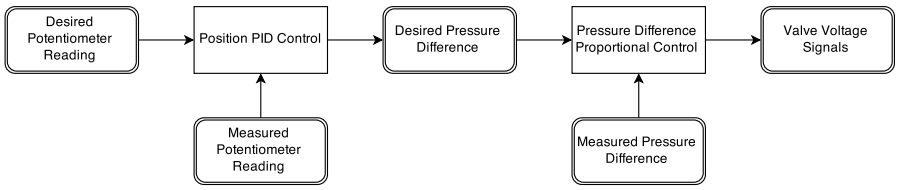


Figure 5.5: Schematic of the controller of Diego's joints

valves to change the pressures. The controllers used for this approach are described in the following sections.

5.2 Controlling Diego: Trajectory Tracking

A trajectory is a sequence of potentiometer readings $q^d_{j,t}$, in Volts, indexed by the time t and joint j. The goal of trajectory tracking is to send motor commands $u_{j,t}$ to a robot to generate a sequence of potentiometer readings $q_{j,t}$ that match, as closely as possible, the desired readings $q^d_{j,t}$. Since the potentiometer readings are linearly related to the angle of the joints, following a trajectory of potentiometer readings corresponds to a trajectory of joint angles.

Figure 5.5 shows the controller we use on Diego's joints. The controller takes a desired trajectory as input, and outputs a sequence of voltage signals to the pneumatic actuator valves, using a pair of controllers. The first is a proportional-integral-derivative (PID) controller that takes the desired potentiometer reading and the observed potentiometer reading as input, and outputs a desired pressure difference to the second controller. The second controller is a proportional controller that takes the output of the first controller and the measured difference of pressure sensors as input, and outputs voltage signal $u_{j,t}$ to the valves. Below we describe these controllers in more detail. In doing so, for notational simplicity, we drop the joint index j, noting that the description applies to each joint.

5.2.1 Introduction to PID control

One of the most popular methods for controlling a system is proportional control. Take the example of controlling the position of one of Diego's joints. Define the error $\varepsilon_t = q_t^d - q_t$ between the desired and measured potentiometer readings. When applied to Diego, u_t is a voltage sent to the actuator valves, whose relationship to the equilibrium force generated by the actuator is shown in Figure 5.4. A proportional (P) controller, defines u_t as

$$u_t = k_p \varepsilon_t \tag{5.4}$$

where k_p is a tunable parameter known as the proportional gain. Note this control law generates forces such that forces will be generated that reduce the error. If $q_t < q_t^d$, u_t will increase the pressure difference between the chambers. The resulting positive force will increase the potentiometer reading. One way to think of a P controller is as a spring attached between the current measurement and the desired measurement. If the spring constant k_p is too large, the spring will produce large forces in response to small disturbances, and lead to oscillation (or instability). If the spring constant is small, the generated forces may be insufficient to reduce the error and track the desired trajectory.

Proportional controllers are simple, but subject to oscillations and steady-state errors. Steady-state error can be illustrated with a simple example. Consider a system that attempts to maintain the vertical position of a mass at position x^d with an actuator that uses proportional control to apply an upward force $f = k_p \varepsilon$ (where the error $\varepsilon = (x^d - x)$). Gravity generates a downward force g on the mass. In the steady state, when the mass is not moving, both forces acting on the mass will be equal. Since we know $f = k_p \varepsilon = g > 0$, we see that the steady-state error must be non-zero in this system. We also see that one way to reduce the steady state error is to increase k_p. However, as k_p increases, the forces

generated in response to small disturbances also increases, which can lead to stiffness and instability.

One way to mitigate the oscillatory and steady-state error effects is to use a proportional-integral-derivative (PID) controller that defines the control signal as

$$u_t = k_p \varepsilon_t + k_d \dot{\varepsilon}_t + k_i \int_0^t \varepsilon_s ds \tag{5.5}$$

where k_d and k_i are tunable parameters. From before, we think of k_p as a spring attached to the desired measurement. The k_d parameter can be thought of as a viscous friction acting on that spring, which can dampen oscillations. The k_i parameter helps offset steady-state errors. As the error integral increases, so does the amount of force applied to the system from the integral term.

We next describe the two-level controller used to control Diego (schematic in Figure 5.5).

Position Controller

Target trajectories are specified in potentiometer readings q_t^d. The position controller we use in Diego (Figure 5.5) takes these desired readings as input, and generates an error between the desired and the measured potentiometer voltage: $\varepsilon_t = q_t^d - q_t$. We let $\hat{d}_t$ be the output of a PID controller that operates on the potentiometer error signal:

$$\hat{d}_t = k_p \varepsilon_t + k_d \dot{\varepsilon}_t + k_i \int_0^t \varepsilon_s ds \tag{5.6}$$

The output of this controller, $\hat{d}_t$, is a desired pressure difference between the two chambers of the actuator. Recall that the difference in pressure specifies the torque at the joint. Thus, the controller attempts to generate forces that reduce the error. However, the desired pressure difference must be translated into a voltage command to the valves, so

we use a separate controller to do so.

Pressure Difference Controller

The position controller specifies the difference in pressure that the pneumatic actuator should achieve. We use $\hat{d}_t$ as the desired pressure difference, and set the error signal $\psi_t = \hat{d}_t - d_t$ where d_t is the observed difference in pressure readings (in Volts) between the two chambers of the actuator, $d_t = p_t^e - p_t^f$.

For the pressure difference controller, we set $u_t = k_u \psi_t$. The full controller (from Figure 5.5) for the extensor chamber of an actuator can be written:

$$\begin{aligned} u_t^e &= 5 + u_t \\ &= 5 + k_u \psi_t \\ &= 5 + k_u(\hat{d}_t - d_t) \\ &= 5 + k_u(k_p \varepsilon_t + k_d \dot{\varepsilon}_t + k_i \int_0^t \varepsilon_s ds - d_t) \end{aligned} \tag{5.7}$$

and similarly for the opposing chamber voltage u_t^f.

$$\begin{aligned} u_t^f &= 5 - u_t \\ &= 5 - k_u(k_p \varepsilon_t + k_d \dot{\varepsilon}_t + k_i \int_0^t \varepsilon_s ds - d_t) \end{aligned} \tag{5.8}$$

5.2.2 Computed Torque Control

PID control treats gravitational, inertial and friction forces as error. As a consequence it typically requires the use of very large values of the spring parameter k_p, which in turns results in high stiffness and high energy consumption. Consider for example an arm held out parallel to the ground, holding a weight. Gravity produces a large torque that needs to be compensated by using large gain values in the controller. When the

arm is lowered and perpendicular to the ground, gravity produces no torque on the joint, however the large gain needed in the controller will make the joint very stiff. Stiffness can in turn result in large energy consumption as well as potentially dangerous behavior when unexpected disturbances occur. The integral term can help reduce the error, but can also lead to dangerous behavior. If the cause of the steady-state error is removed, like when the arm is lowered, the integral term will still generate forces to compensate for the now-absent forces. One potential solution is to compensate for known disturbances, like gravity and friction, in tandem with a PID controller. This approach is generally known as computed torque (CT) [56]. For our purposes, we will use inverse dynamics compensation control, henceforward referred to as simply CT control.

CT control uses an inverse dynamics model to map from the current joint angle vector θ, velocity $\dot{\theta}$, and desired acceleration $\ddot{\theta}^d$ to a torque τ_t^c that should be applied to compensate for external forces and achieve the desired acceleration. We consider two such functions, one that is derived from the PPM, and one that is learned directly from recorded data using a DNN. The physics approach was used in the previous chapter, for the simulations, and the DNN approach is used for the robot implementation.

Computed torque with parameterized physics models

The PPM approach is based on the robot dynamics explained in Appendix A. It is presented here to contrast it with the DNN approach. The articulated body dynamics equation can be written.

$$\tau_t = M(\theta_t)\ddot{\theta}_t + N(\theta, \dot{\theta}) \tag{5.9}$$

where τ_t is the vector of torques due to the robot's actuators at time t, $M(\theta_t)$ is the inertial matrix, and $N(\theta, \dot{\theta})$ are the gravitational, friction, and Coriolis forces.

The type of CT control examined here uses the inverse dynamics function to define the computed torque $f(\theta, \dot{\theta}, \ddot{\theta}^d) = \tau_t^c$ using (5.9) as follows [57]:

$$\tau_t^c = M(\theta_t)\ddot{\theta}_t^d + N(\theta, \dot{\theta}) \tag{5.10}$$

Given τ_t^c, the next step is to identify the pressure readings $\hat{p}_t^e$ and $\hat{p}_t^f$, in Volts, that produce the torque desired for each joint. There are an infinite number of solutions, so we introduce a constraint by choosing a constant sum of the pressure readings $s = \hat{p}_t^e + \hat{p}_t^f$. From (5.2), we relate pressure readings and torques

$$\tau_{j,t}^c = J_{jj}(\theta_j)\left(a(p_t^e - p_t^f)\right) \tag{5.11}$$

where $J_{jj}(\theta)$ is the moment arm of joint j. Since the moment arm and the area a are known, we get the desired pressure reading difference $d^c = \hat{p}^e - \hat{p}^f$ that can be added to the output of the positional PID controller, to specify

$$\hat{d}_t = d^c + k_p\varepsilon(t) + k_i\int_0^t \varepsilon(s)ds + k_d\frac{d}{dt}\varepsilon \tag{5.12}$$

This controller replaces the PID controller in (5.6), and sends $\hat{d}_t$ to the pressure difference P controller. This approach is also called the PIDF approach in [55].

The parameters of the PPM can be identified from recorded observations of joint torques, positions, velocities, and accelerations. Appendix A describes how the inverse dynamics equation is a linear function of the inertial parameters, and describes how these parameters can be identified.

DNN Computed Torque

Using the PPM requires explicitly modeling friction and other phenomena like joint limits and self-contact forces, which is notoriously difficult. In simulated experiments, the physical model matches the simulated physics exactly, but this is not the case in the real world. Rather than trying to augment the physical model to better represent the behavior of the real world, another approach is to use a function approximator to learn a direct mapping from the joint angle measurements to the pressure difference measurements that produce the observed acceleration. Here we use a deep neural network. Using the Torch framework [58], we construct a DNN similar to the one depicted in Figure 5.6.

The inputs to the model at time t are the potentiometer readings, $q_{t-b:t+f}$ in a window from time $t-b$ to time $t+f$, where b and f are tunable parameters, and the sampling rate is 100Hz. We tried multiple values for b and f, and found the best performance for $b = f = 2$. The DNN is trained to approximate the inverse dynamics, which is a function mapping the joint angles, joint angle velocities, and accelerations to the torques that cause the accelerations. The potentiometers at the joints of Diego can only measure the position of the joint. Typically, these readings are converted into velocities and accelerations by finite differencing. This process introduces noise, and so temporally smoothed versions of the readings are used. Here, we do away with this complexity. Since differentiation is a linear operator, we allow the network to learn a filter over the consecutive sensor readings themselves. Note that the input here is the reading from the potentiometer at each joint, q, rather than the joint angle, θ. In the PPM approach, the known calibration parameters γ and β convert the voltage from the sensors to joint angles as $\theta = \frac{q-\beta}{\gamma}$, but the neural network uses the sensor readings directly.

The target output of the neural network at time t is a 38-dimensional vector p, where each entry $p_j = p_j^e - p_j^f$ is the difference in pressure sensor readings, in Volts,

Table 5.2: Size of layers used for inverse model neural network. The first three layers are replicated for each of the 38 joints. Values are those found to have best validation error.

Layer	1	2	3	4	5	6	7
#neurons	5	10	5	190	240	70	38

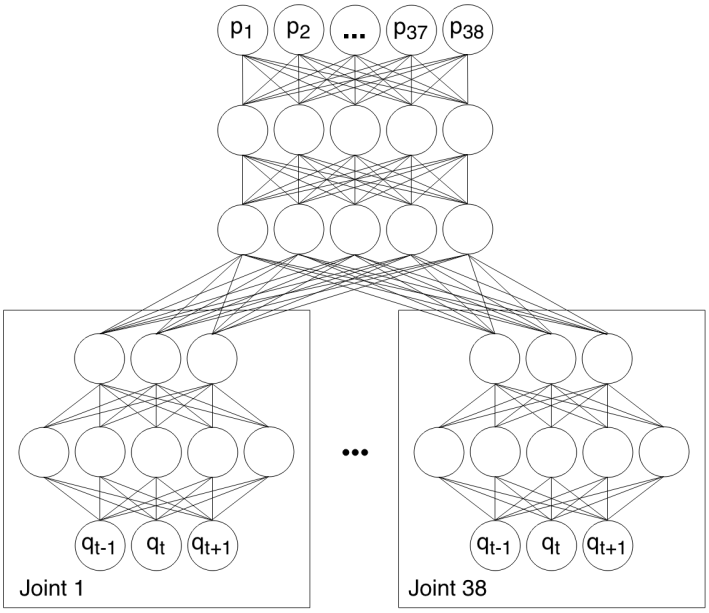


Figure 5.6: Diagram of the network used to estimate the inverse dynamics compensation. Exact number of layers and nodes is shown in Table 5.2

between the two chambers (extension p_j^e, and flexion p_j^f) in the pneumatic actuator for joint j. Figure 5.6 shows the architecture of the net used to map from potentiometer sensor readings to observed pressure sensor differences. First, the potentiometer readings are separated by joint, and processed through 3 fully-connected layers of neurons with hyperbolic tangent (tanh) activation functions. Then, the top layers of these joint-separated branches are combined into another set of fully-connected layers that take information from all joints into account. Different architectures, activation functions, and sizes of layers were examined, and Table 5.2 shows the size of the layers for the network that was found to have the best validation error.

Once the neural network is trained, its predicted p vector are used to generate a desired pressure difference control signal $\hat{d}_{j,t}$ to send to the pressure difference P controller of joint j:

$$\hat{d}_{j,t} = p_j + k_p \varepsilon_j(t) + k_i \int_0^t \varepsilon_j(s)ds + k_d \frac{d}{dt}\varepsilon_j \tag{5.13}$$

Visual Deep Learning Computed Torque Control (VCT)

As Table 5.1 indicates, the approach from the previous chapter uses visual features to estimate the analytical physics model of the robot+object system, and uses this model to generate torques to match the desired acceleration. But, as previously mentioned, the PPM approach is not accurate enough, so we must include visual features into the deep learning approach. To do so, we change the architecture of the deep network to that reflected in Figure 5.7.

In this architecture, there are n DNN CT expert networks. Each of these networks is trained with data recorded while Diego held a different object. These experts each output their estimate for the target p vector. For expert e this output is $\hat{p}_e$. There is also a gating network, whose n-dimensional output, w weights the output of each expert to give the final estimate $\hat{p} = \sum_e w_e \hat{p}_e$. The input to the gating network, v, is a visual signal. This could be a convolutional neural network that takes pixels as input. Here, we use features derived from the RGBD images taken from a depth camera. Using the Point Cloud Library [59], we extract the length and radius of the red cylindrical object Diego interacts with, and use these as the visual features.

We will use the acronym VDNN when referring to VCT implemented with the DNN inverse models and arbitrator network, and VPPM when referring to VCT with the PPM inverse model.

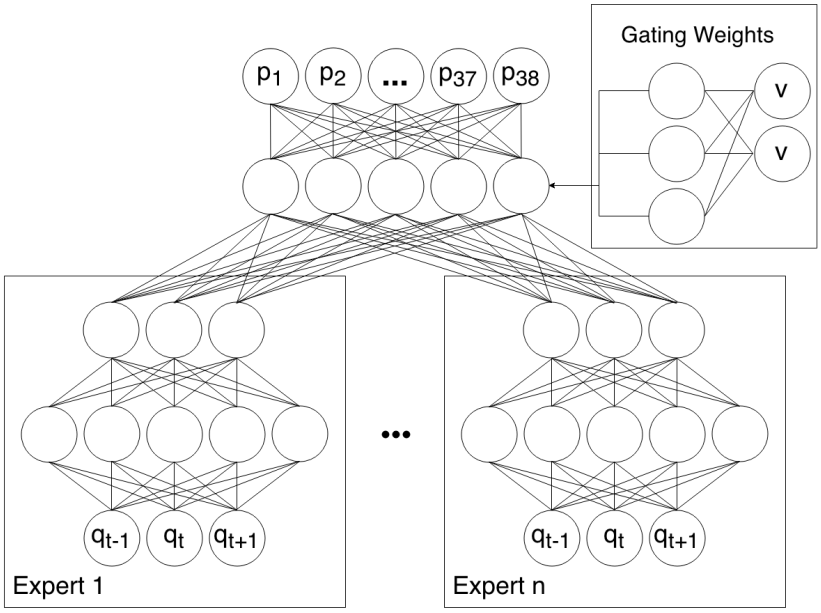


Figure 5.7: Diagram of the network used for visual computed torque.

5.2.3 Controller Tuning

To allow Diego to track a target trajectory efficiently, the gains k_p, k_i, and k_d for both the PID alone and the VDNN controller must be set. Since these parameters can dramatically change the performance of the controller, we automate the process of setting the parameters to facilitate comparison between the two.

To find gains that follow the desired trajectory described above, we perform a grid search to minimize the tracking error:

$$\rho = \min_{s} \alpha s + \frac{1}{T-s} \sum_{t=s}^{T} \left[q_t - q^d_{t-s} \right]^2 \tag{5.14}$$

where q_t is the potentiometer reading, in Volts, at time t in the observed trajectory, q^d_t is the desired joint angle at time t, and α is a penalty for delay, which we set to 0.01 for our experiments. We include the delay penalty because we observe that when required to follow the trajectory exactly, the robot produces juddering movements. Instead, we want

Figure 5.8: The set of rods used in the experiment. The rods of the same length are a rod of the same density as the other lengths, and a decoy rod that has lower density. The rods are coated with a red plastic to help Diego grasp them more securely.

a controller that gives a good, smooth fit to the trajectory even if there is a small offset in time.

Each value on the grid is repeated once with each of the first three objects, but never with the decoy object, or the largest object. The trajectory was the same as we use for the Mounoud and Bower experiment: the arm raises, grabs an object, and holds steady. The grid search examines each joint in isolation, and for each joint the grids are adapted by hand until the results show a minimum value that was not at the border of the grid. The best parameter values for each joint are combined to give the final, whole-arm PID gains. Once these gains are found, we fix them for conducting the experiments described in Section 5.3. The parameters are the same for both the PID and the VDNN controllers.

5.2.4 Visual Features

To implement a VCT controller, we need to confirm that Diego can successfully distinguish the objects used in the experiments based on visual features. These objects are shown in Figure 5.8.

Using a Microsoft Kinect depth camera and the Point Cloud Library [59], each

Table 5.3: Object radius and length mean (standard deviation) in cm

	Estimated Radius	Estimated Length	Actual Radius	Actual Length
Object 1	2.61 (0.37)	14.1 (1.3)	2.5	13.5
Object 2	2.59 (0.40)	19.1 (0.9)	2.5	18.5
Object 3	2.62 (0.31)	23.8 (2.2)	2.5	23
Decoy	2.58 (0.34)	23.2 (1.4)	2.5	23
Object 4	2.54 (0.48)	27.5 (1.8)	2.5	27

Figure 5.9: Example RGBD image for extracting rod radius and length. Pixels colored in green represent the estimate of which points belong to the object

rod's radius and length is estimated. Figure 5.9 shows an example frame gathered from the depth camera. Pixels colored in bright green represent the estimate of which points belong to the object. The radius and length are estimated as the mean of the estimates from 60 such frames. For all objects, these estimates were within 1cm of the actual values as measured by hand. Table 5.3 shows the mean and standard deviation for each of the objects' radius and length estimates, as well as the actual measurements.

5.3 Experiments

5.3.1 Experiment 1: Computed torque tracking performance

First, we compare the performance of PID control alone and the DNN CT approach on a trajectory tracking task. We examine the tracking error of each on a recorded trajectory while Diego's left hand grasped the largest object in Figure 5.8. Because the PID controller does not have a model of the system, we expect the changing orientations of the joints with respect to gravity will create disturbances that the PID controller will not consistently reject.

Consider the elbow joint, for example. When the arm is oriented such that gravity is perpendicular to the axis around which the elbow rotates, the PID controller must generate larger forces to move against the effect of gravity. But, when the elbow is oriented such that gravity is parallel to the joint axis, gravity has no effect. In this case, smaller forces are required to move the arm. However, since the PID defines its control signal based only on positional error, it cannot differentiate between these two scenarios.

If the DNN model has learned the inverse dynamics well enough, the DNN CT approach can generate compensatory forces that make the output of its associated PID controller independent of the joint axis orientation with respect to gravity. Here, we measure this effect by examining the trajectory tracking error. We hypothesize that the DNN CT approach will have a lower tracking error than the PID approach.

Methods

To specify the target trajectory, during a 120 second period, Diego's joints are moved by hand while the potentiometer readings are recorded. These readings are used as the desired positions to be tracked.

The DNN CT controller here is learned using data while grasping the same object

Table 5.4: Trajectory tracking errors

Controller	Mean	Standard Deviation
DNN CT	0.035	.011
PID	0.066	0.002

held during tracking. The structure of the network is shown in Figure 5.6 and Table 5.2. Because the DNN CT model corresponds exactly to the object being held, visual features are not used in this experiment. Both of the controllers use the same PID parameters, which were set following the procedure described in Section 5.2.3.

We use each controller to track the target trajectory 10 times. On each trial, we measure the tracking error as defined in (5.14). A t-test is performed to examine whether there is a difference between the tracking error for the two controllers.

Results

Table 5.4 shows the mean and standard deviation of the tracking error for each controller type. The mean tracking error (5.14) for the DNN CT approach is 0.035. The mean for PID is higher, 0.066, and this difference is significant ($p < 0.00005$, $t=10.5$, $df=9$).

Figure 5.10 shows an image captured during trajectory tracking for the PID controller on the left, and with DNN CT on the right. The desired position of the robot at the time was with the object near Diego's mouth, and with the object held upright. The PID controller is unable to bring the object to the mouth, but the DNN CT controller can.

Figure 5.11 shows the potentiometer readings through time for the forearm joint for one trial of each controller. From Section 5.1 we know that these potentiometer readings are linearly related to the joint angles. For the forearm specifically, increased potentiometer readings mean an increased joint angle. When the arm is hanging down, the increased joint angle means the thumb is rotated toward the body. Lower potentiometer

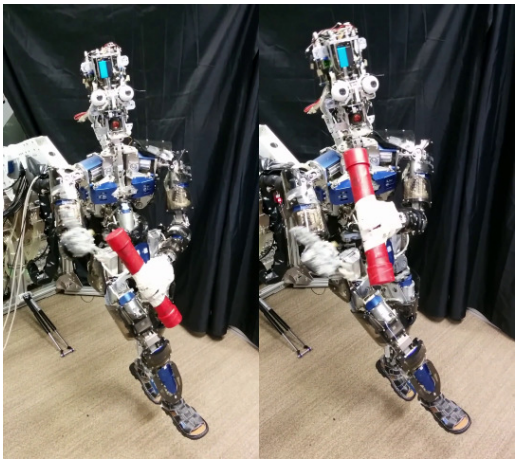


Figure 5.10: Comparison of PID and CT control. An image from the same time step while following a trajectory that involves moving the object near Diego's mouth and keeping the object upright. PID control alone is on the left, and DNN CT control (using the same PID parameters) is on the right. Note that the object is held higher with the DNN CT control, and the orientation of the object is more upright.

readings mean rotation away from the body in that position. The desired trajectory is the solid black line, the dashed blue line is the PID controller, and the solid red line is the DNN CT controller. Other joints look similar, in that the DNN CT is more stable and follows the desired trajectory more closely.

From this experiment, we can see that the DNN CT approach allows lower error trajectory tracking than PID alone. This also suggests that the inverse dynamics model can be learned well enough by the DNN to improve the control of Diego's joints. Next, we examine whether visual features of an object can be used to weight the output of several DNN models trained on different objects.

5.3.2 Modeling Mounoud and Bower's 1974 study

In the previous chapter (section 4.4.1), we show via computer simulations that our proposed framework for visual affordances can replicate the results of the Mounoud and Bower experiment. Here, we examine whether the framework can replicate the results in

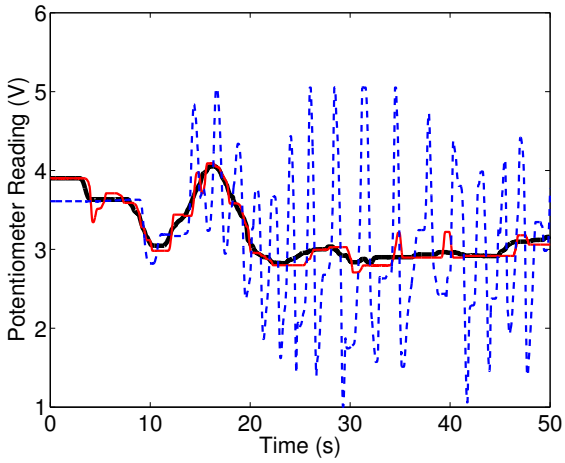


Figure 5.11: Potentiometer readings during trajectory tracking task for the forearm joint. The desired trajectory is the solid black line, the dashed blue line is the PID controller, and the solid red line is the DNN CT controller.

Table 5.5: Object length (cm) and mass (g)

Object	Length (cm)	Mass (g)
1	13.5	310
2	18.5	445
3	23	550
Decoy	23	150
4	27	650

an actual robot, Diego.

The following experiments are performed using a set of five rods (Figure 5.8) of varied length and weight. Of those five, four have the same density, and the fifth is a decoy; it has the same length as the middle rod, but is considerably lighter. We label the rods from the shortest (Object 1), to the longest (Object 4), and label the rod with lower density matched in length to Object 3 the decoy. Table 5.5 describes the characteristics of the rods.

We are interested in reproducing the following three experiments from the Mounoud and Bower paper:

1. Learn to generate correct anticipatory forces for grabbing an object based on

experience holding it. (Section 5.3.3)

2. Generalize forces to objects not previously explored. (Section 5.3.4)

3. Show an error on the decoy object based on its surprising mass-length relationship. (Section 5.3.5)

For each trial of the following experiments, Diego tracks a target trajectory intended to mimic the infants grasping of the rods presented by the experimenter. The left arm is raised to horizontal, the left hand closes around the object presented by the experimenter, and the goal is for Diego to keep the arm at the same position as before the object was grasped.

The following experiments examine two controllers:

1. VCT using DNN CT experts, (VDNN henceforward)

2. PID control

5.3.3 Experiment 2: Anticipatory Forces

One feature of the Mounoud and Bower experiment is that the infants were able, after experience with a rod, to generate anticipatory forces that reduce the error observed in arm position after the object is grasped. In this experiment, we examine whether the VDNN controller can generate anticipatory forces while grasping Object 3. If the VDNN controller is able to generate anticipatory forces for holding the object level, we expect the error to be lower when the VDNN controller is used than when the PID controller alone (here used as experimental control) is used to grasp the object. If this hypothesis is correct, it would indicate that using a VDNN model to make predictions about what forces need to be applied can allow better control of a robot at the instant of contact with an object, rather than relying on reactions to a changing situation.

Methods

Pilot trials revealed that the elbow shows the largest movement when a heavy object is grasped. The elbow is oriented such that gravity is perpendicular to the elbow's joint axis. Thus, when an object is grasped, the gravitational forces are converted to torques around the elbow's axis, and these torques generate movement if they are not compensated by a controller.

To simplify analysis, we focus on the potentiometer reading at the elbow. When the elbow is fully extended, the potentiometer reading is low. As the elbow flexes, the potentiometer reading increases. The dependent measure δ is the elbow potentiometer reading that gives the largest deviation from the desired trajectory of the elbow:

$$\delta = \underset{q_t}{\operatorname{argmax}} |q^d - q_t| \tag{5.15}$$

where q_t is the recorded potentiometer reading of the elbow, q^d is the target potentiometer reading, and t is restricted to the 0.5 second period directly after the object is grasped. During this period, q_d is constant, and is set to 1.4.

A trial with the VDNN controller consists of the following steps:

1. Extract visual features from the object.
2. Using the visual features as input to the arbitrator network, generate control signals from the VDNN while attempting to track the target trajectory.
3. Measure the δ of the trial.

A trial with the PID controller tracks the same target trajectory, but with no visual modulation.

Fifteen trials are run with each controller, and a t-test is performed to examine if there is a difference in δ between the VDNN and PID controllers.

Table 5.6: Anticipatory forces: δ mean and standard deviation

Controller	Mean	Standard Deviation
VDNN	1.22	0.028
PID	0.82	0.029

Results

Table 5.6 shows the dependent measure δ (5.15) for the VDNN and PID controllers. A two-sample t-test shows that the means are significantly different ($p<1x10^{-16}$, t=53.8, df=14).

Figure 5.17 shows the potentiometer recordings from the elbow as well as the desired trajectory. The plot shows the arm rising between 6 and 9 seconds, grasping the object just after 9 seconds (the sharp drop in both traces), and trying to hold steady until the end of the plot. Many of the arm joints beside the elbow show a similar, if smaller, effect.

Figure 5.12 shows the arm position after the object is grasped with the PID controller alone. Figure 5.13 shows the same point in the trajectory for the VDNN controller. These are sample frames taken from two trials of the experiment.

These results indicate that the VDNN controller allows the robot to grasp the object and stay closer to the desired trajectory than the PID controller alone does.

5.3.4 Experiment 3: Generalizing to Novel Objects

Mounoud and Bower showed that, after experience with one rod, the error on the next rod presented was reduced compared to if the infant was presented that rod first. This suggests that infants are generalizing the visual-proprioceptive relationship they are learning while exploring one of the objects. Here, we investigate whether our approach can reproduce this phenomenon, using an object with which the VDNN was not trained.

Figure 5.12: Response of the PID controller to grasping a heavy object. The left image shows the position of the arm at the start of the grasp, which is the desired location. The image on the right shows the object at its farthest displacement from the desired trajectory.

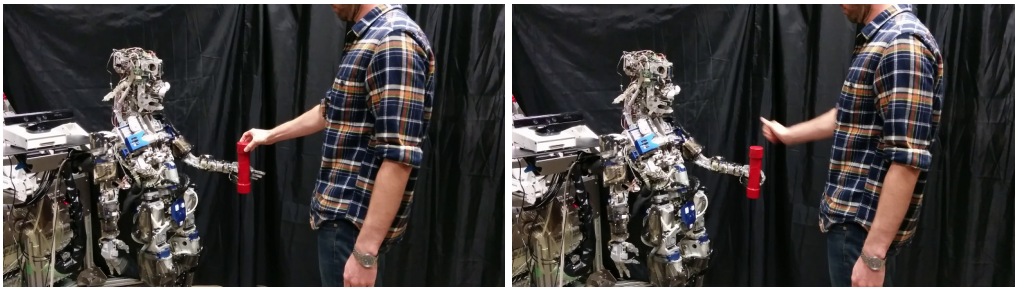


Figure 5.13: Response of the VDNN controller to grasping the same object from Figure 5.12. The left image shows the position of the arm at the start of the grasp, which is the desired location. The image on the right shows the object at its farthest displacement from the desired trajectory.

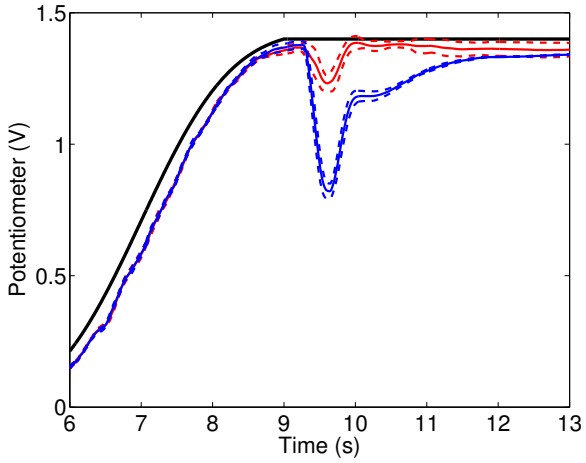


Figure 5.14: PID and VDNN comparison for heavy object. Potentiometer measurements at the elbow around the time the object is grasped (just after 9 seconds). The thick black line shows the desired trajectory. The solid red line shows the mean value for the VDNN controller trials. The solid blue line shows the mean value for the PID controller trials. The dashed lines show the standard deviation.

In this experiment, we examine whether the VDNN controller can produce anticipatory forces for an object that it has never seen before. Since the visual and inertial features of Object 4 are consistent with the three objects on which the VDNN was trained, we hypothesize that the VDNN will be able to extrapolate to the larger Object 4 and produce a smaller error than a PID alone.

Methods

For this experiment, the target trajectory is the same as the previous experiment: raise arm to horizontal, grasp object, remain level. The object grasped in this experiment is the largest object in the set, Object 4. Object 4 was not shown to the VDNN controller during training. It is larger and heavier than all other objects used in the experiments, but has the same density as the other non-decoy objects. We compare the δ of the VDNN on object 4 with that of a PID controller on object 4.

The dependent measure δ is given in (5.15).

Table 5.7: Generalizing to novel objects: δ mean and standard deviation

Controller	Mean	Standard Deviation
VDNN	1.19	0.022
PID	0.70	0.025

Fifteen trials are run with each controller, and a t-test is performed to examine if there is a difference in δ between the VDNN and PID controllers.

Results

Table 5.7 shows the mean and standard deviation of the dependent measure δ for both the VDNN and PID controllers. A t-test reveals a significant difference between the controllers ($p < 1x10^{-17}$,t=56.0,df=14). This difference indicates that the VDNN is able to produce reasonable forces for an object it has never seen before. Since the relationship between the visual features (like the length) and the inertial properties of Object 4 are similar to the objects with which the VDNN was trained, it is able to scale the forces appropriately to extrapolate.

Figure 5.17 shows the distribution of potentiometer readings for the different conditions across trials. Note that the PID controller has a very large standard deviation near the end of the plot. This is because several trials were unstable. Since the PID parameters were not tuned on this object, the controller was not able to stably compensate for some errors and produced high-velocity oscillations.

Figure 5.15 shows the arm position after the object is grasped with the PID controller alone. Figure 5.16 shows the arm position at the same point in the trajectory, but where the VDNN controller is used.

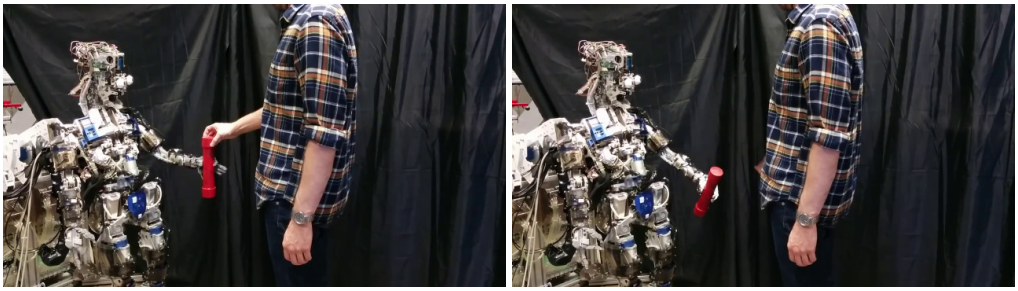


Figure 5.15: Response of the PID controller to grasping a novel object, heavier than the objects in a test set. The left image shows the position of the arm at the start of the grasp, which is the desired location. The image on the right shows the object at its farthest displacement from the desired trajectory.

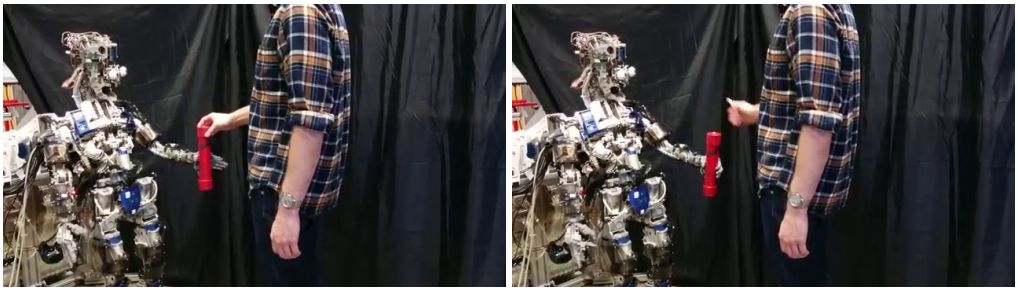


Figure 5.16: Response of the VDNN controller to grasping the same object from Figure 5.15. The left image shows the position of the arm at the start of the grasp, which is the desired location. The image on the right shows the object at its farthest displacement from the desired trajectory.

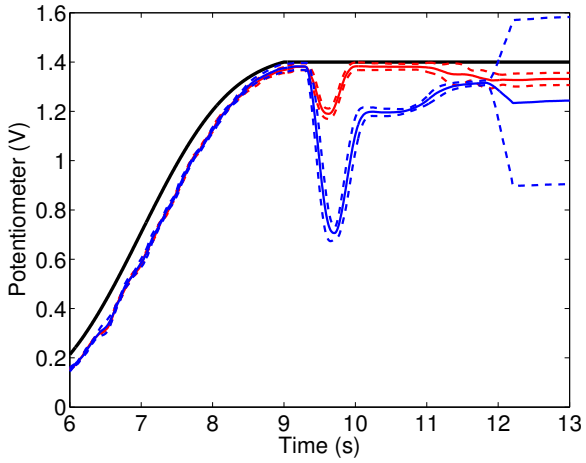


Figure 5.17: PID and VDNN comparison for a novel object, heavier than the objects in the test set. Potentiometer measurements at the elbow around the time the object is grasped (just after 9 seconds). The thick black line shows the desired trajectory. The solid red line shows the mean of the VDNN controller trials. The solid blue line shows the mean of the PID controller trials. The dashed lines show the standard deviation

5.3.5 Experiment 4: Response to Decoy

In the Mounoud and Bower experiment, infants mistakenly lift a decoy object. Since the object looks the same as one they have previously grasped, they lift with more force than is required for the lighter decoy. Here we investigate whether the decoy object, that does not match the learned visual-inertial relationship, produces the same mistaken lift in our framework as it does in infants. This experiment explores the response of the VDNN controller to the decoy object.

The anticipatory forces generated by the VDNN in the previous experiments allow more accurate trajectory tracking than a PID alone. The VDNN is able to compensate for the additional forces due to gravity starting from the instant when the objects were grasped. The VDNN represents a framework for learning to interact with different objects in the environment, capitalizing on regularities between visual and inertial properties of objects. Since the VDNN is learning this visual-inertial relationship, we expect that, when we alter this relationship, the VDNN will make an error by lifting the decoy

object. The Mounoud and Bower experiment shows that infants make an error when the visual-inertial relationship is broken.

If the VDNN replicates the phenomenon observed in infants, we expect the elbow to lift higher than the target trajectory with the VDNN controller when the decoy object is presented. Since there is no visual modulation in the PID controller, we expect the PID controller to show a small drop at the elbow commensurate with the smaller weight of the decoy object compared to the other objects used in the experiment.

Methods

For this experiment, the target trajectory is the same as the previous experiment: raise arm to horizontal, grasp object, remain level. The object grasped in this experiment is the decoy, which was never shown to the VDNN during training.

Fifteen trials are run with each controller, and a t-test is performed to examine if there is an difference in δ (5.15) between the VDNN and PID controllers. Additionally, a one-sample t-test is performed to examine if the VDNN controller generates a lift compared to the desired trajectory.

Results

Table 5.8 shows the dependent measure δ (5.15) for the VDNN and PID controllers. A t-test reveals a difference between the two controllers ($p<1x10^{-15}$, t=41.2, df=14). A one-sample t-test reveals that the VDNN has mean different from the desired trajectory, which has a value of 1.40 ($p<1x10^{-13}$, t=28.8, df=14). Figure 5.20 shows the potentiometer reading from the elbow as the object is grasped. Notice that the VDNN controller overshoots the desired trajectory. The VDNN specifies CT forces based on the visual features of the object it interacts with. Because the decoy object and Object 3 have very similar visual features (see Table 5.3), and the VDNN was trained to generate forces

Table 5.8: Response to decoy: δ mean and standard deviation

Controller	Mean	Standard Deviation
VDNN	1.49	0.012
PID	1.30	0.014

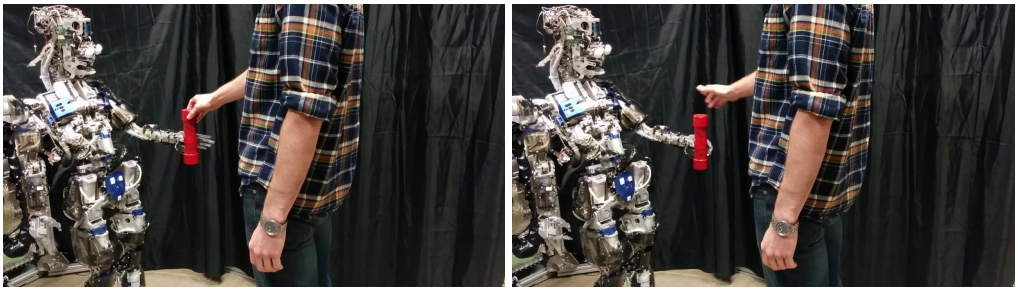


Figure 5.18: Response of the PID controller to grasping the decoy object. The left image shows the position of the arm at the start of the grasp, which is the desired location. The image on the right shows the object at its farthest displacement from the desired trajectory.

for the heavy Object 3, the forces it estimates for the decoy object are larger than what the mass of the decoy require. Thus, the elbow rises to produce the overshoot we see in Figure 5.20.

Figure 5.18 shows a frame after the decoy object has been grasped by the PID controller. Figure 5.19 shows the same frame but when the VDNN controller grasps the object. Notice the angle of the forearm with respect to the elbow is sharper in the VDNN case, and that the object is higher.

5.4 Discussion

We explored whether our framework for visual affordances can reproduce the following phenomena in a complex humanoid robot: (1) learning to generate correct anticipatory forces to hold objects closer to level, (2) generalizing experience with objects to novel but consistent objects, and (3) being susceptible to overcompensating for a decoy

Figure 5.19: Response of the VDNN controller to grasping the decoy object. The left image shows the position of the arm at the start of the grasp, which is the desired location. The image on the right shows the object at its farthest displacement from the desired trajectory.

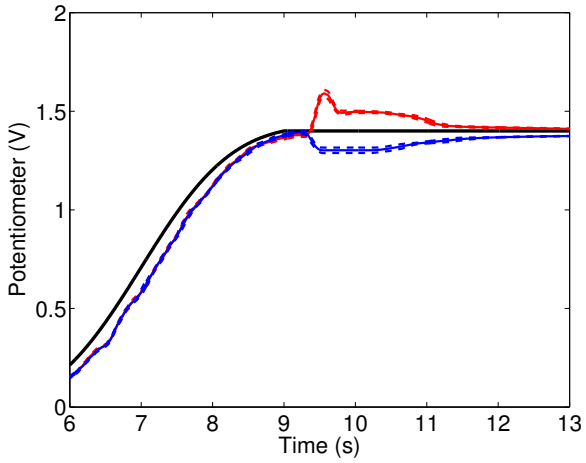


Figure 5.20: PID and VDNN comparison for decoy object. Potentiometer measurements at the elbow around the time the object is grasped (just after 9 seconds). The thick black line shows the desired trajectory. The solid red line shows the VDNN controller. The dashed blue line shows the PID controller

object. These phenomena have been observed in human infants, and suggest that humans use internal models to plan motor commands to interact with objects in the world. In the previous chapter, we suggested a framework for how these models might be learned and used, which is based on a parameterized articulated-body physics model (PPM). Here, we found that parametric physics models are not accurate enough at predicting the inverse dynamics for a complex robot like Diego. By making some changes to the approach, namely using a deep neural network (DNN) as an inverse dynamics model, we were able to improve the accuracy of the predictions of the torques required to generate desired accelerations. This required a change to how anticipatory forces are generated for objects based on visual features. Rather than using the articulated body physics and estimates of density, the new approach uses an arbitrator network to weight the output of several DNN expert inverse model networks. These changes have several implications, including:

- **Generalization to other areas of joint space.** Because the PPM is a physics model, it describes motions across the entire range of the robot joint angles (and also different grasp configurations of an object). In contrast, the DNN approach, since it is learned directly from experience, requires experience in the area of joint space where testing will be done. Interestingly, there seems to be some evidence that learned dynamic models in humans do not generalize well to different areas of joint space [60]. In their study, subjects learn to generate anticipatory forces while trying to hold an object in the same location while rotating it. Given practice when the object is at a particular orientation with respect to the hand, subjects generate only about half the anticipatory forces when the same object is in a different orientation with respect to the hand. This suggests that there may not be a global representation of the hand/object dynamics for all possible orientations of the hand and object, as would be expected from the parameterized physics model.

- **Generalization to other object shapes.** The PPM approach depends on the visually extracted geometry of an object to create an inverse dynamics model. Because this can generalize to any geometry, this approach is well suited for estimating dynamic models of objects with novel shape. In simulations, this approach works well. But, if the model is not powerful enough to capture the dynamics of an actual robot, this approach will not succeed. The DNN approach, in contrast, learns directly from experience. Because the DNN approach is a general function approximator, given enough data and computational power, it will be able to model the dynamics of the objects it has learned. In a recent result, a DNN was taught to play Atari directly from visual input [61]. The approach here is similar in that control is learned directly from visual input and experience.

- **Prior knowledge requirements.** The cost of the greater generalization of the PPM is that there is a great deal of knowledge that must be supplied by a human to the robot. The DNN model learns from experience, and so does not require knowledge of, for example, the robot's geometry or sensor calibrations. In a sense, the PPM has been given a shortcut: physics prescribes a good, succinct representation of an object's dynamics through its density and geometry. The DNN model, given enough representational power and data, could potentially learn a similar representation implicitly. Additionally, in the case where the PPM itself is not representative enough to explain the data, the extra effort required to learn the DNN model pays off in better accuracy on data similar to what has been learned.

The flexibility of the function-approximating deep neural network that makes up the DNN approach means it can adapt to different situations and learn from experience. But this flexibility comes at a cost of reduced generalization. However there is evidence that humans do not generalize as well as would be predicted from a PPM.

5.5 Conclusion

We implement a computational model of inertial affordances on a physical robot. The implementation is inspired by the model described in the Chapter 4. However, in the process of implementing the approach, we made changes to the original model. Namely, the parametric physical model was replaced by a deep neural network that learns the robot dynamics from sensor readings.

Changing the underlying dynamic model to the DNN also requires a change to the way we generalize control to new objects. In the previous chapter, this is done analytically using a physics model. In this chapter, we use an approach composed of arbitrator network that utilizes visual features to predict the robot dynamics when the robot grasps a new object.

When tested on the physical robot Diego, the neural network approach produced more accurate control (compared to a PID controller) on objects that had a density similar to previously experienced objects. When the density was different than previously experienced, the approach showed behavior qualitatively similar to the behavior shown by infants.

As robotic applications expand, and robots become more ubiquitous, they will need to interact with the objects that humans interact with. One way to allow this interaction is to engineer robots with high-friction joints and powerful actuators. Robots designed this way are able to move objects easily, but at the cost of higher stiffness and energy use. Not only is this undesirable given a limited battery, but from a safety standpoint. Any human in the way of a robot like this would risk serious injury. Another approach is to use compliant robots, like Diego. Compliant robots require less energy, and are not as stiff as typical robots. This confers safety and energy use advantages, but at the expense of increased control complexity. The framework we explore in this chapter

is one method for making control of these robots easier. The framework learns how to interact with objects based on experience. Such an approach lays the groundwork for altricial robots that, through interaction with their environments, can build competence autonomously. Rather than requiring human intervention to solve a novel problem, a developmental robot may discover the solution itself. The work presented in this thesis is one step on the long road toward that goal.

Acknowledgments

This research was supported by NSF IIS 0968573 and NSF IIS 0808767.

Appendix A

Linear System Identification

In this appendix, we show how the dynamics equations for articulated bodies are linear on the inertial parameters. These parameters are:

- m_i: the mass of link i
- $m_i b_i$: the mass times the center of mass
- I_i: the inertial matrix of link i
- η_i: the coefficient of viscous friction at joint i

For a more thorough treatment, see [[49]], from which this appendix is distilled. Articulated bodies are a series of rigid links connected by joints. Here, we focus on revolute joints with 1 degree of freedom (dof), but the similar results hold for prismatic joints. Multiple-dof joints can be modeled as a series of 1-dof joints with zero-length links between them.

A.1 Notation

- f_k: the frame of reference of link k. Its origin is at the parent joint for that link. The world coordinate frame is f_0.

- l_{k-1}: The f_{k-1} coordinates of the origin of f_k. The f_{k-1} coordinates of joint k. The f_{k-1} coordinates of the parent joint for link k.

- y_{k-1}: The f_0 (world) coordinates of the origin of f_k. The f_0 coordinates of joint k. The f_0 (world) coordinates of the parent joint of link k.

- b_k: The f_k coordinates of the center of mass of link k.

- x_k: The f_0 (world) coordinates of the center of mass of link k.

- u_k: The f_{k-1} coordinates of the axis of rotation for joint k.

- θ_k: The angle of rotation for joint k. The vector of all joint angles is θ.

- r_k: The rotation of f_k with respect to f_{k-1}. Thus the columns of r_k have the f_{k-1} coordinates of the f_k unit axes.

$$r_k = e^{[u_k \theta_k]_\times} \tag{A.1}$$

  We define $r_0 = I_3$, the 3×3 unit matrix.

- s_k: The rotation of f_k with respect to the world f_0. Thus s_k has the world coordinates of the f_k unit axes

$$s_k = r_{1:k} = r_1 \cdots r_k \tag{A.2}$$

- v_k: The f_0 (world) coordinates of the axis of rotation for joint k.

$$v_k = s_{k-1} u_k \tag{A.3}$$

- J_k: The Jacobian of the center of mass of link k

$$J_k = \frac{\partial x_k}{\partial \theta} \tag{A.4}$$

- H_k: The Jacobian of the origin of f_{k+1}, which we have defined as y_k.

$$H_k = \frac{\partial y_k}{\partial \theta} \tag{A.5}$$

- $[\cdot]_\times$: The skew-symmetric cross-product matrix constructed from a vector

A.2 Dynamics

We begin with the dynamics equation from (4.1)

$$M(\theta_t)\ddot{\theta}_t = \tau_t + N(\theta_t, \dot{\theta}_t) \tag{A.6}$$

and expand N to explicitly represent the Coriolis, gravitational, and viscous friction torques

$$M(\theta_t)\ddot{\theta}_t + C\dot{\theta} = \tau_t + \tau_t^g + \tau_t^v \tag{A.7}$$

where τ_t is the torque due to the robot's motors at time t. The terms M and C are the sums of inertia and Coriolis terms from each link, $M = \sum M_i$ and $C = \sum C_i$. We can also write

M_i and C_i in terms of the Jacobian, J_i, of the center of mass of link i, and its temporal derivative, $\dot{J}_i = \frac{d}{dt}J_i$

$$M_i = m_i J_i' J_i \tag{A.8}$$

$$C_i = J_i' \dot{J}_i \tag{A.9}$$

We can write J_i as

$$H_0 = \mathbf{0} \tag{A.10}$$

$$J_i = H_{i-1} - s_i [b_i]_\times s_i' w_i \tag{A.11}$$

where $w_i = (v_1, v_2, \cdots, v_i, 0, \cdots, 0)$.

We can also write

$$\dot{J}_i = \dot{H}_{i-1} + [\dot{x}_i - \dot{y}_{i-1}]_\times' w_i + [x_i - y_{i-1}]_\times' \dot{w}_i \tag{A.12}$$

where

$$\dot{H}_0 = \mathbf{0} \tag{A.13}$$

$$\dot{H}_i = \dot{H}_{i-1} + [\dot{y}_i - \dot{y}_{i-1}]_\times' w_i + [y_i - y_{i-1}]_\times' \dot{w}_i \tag{A.14}$$

$$\dot{x}_i - \dot{y}_{i-1} = (J_i - H_{i-1})\dot{\theta} = [x_i - y_{i-1}]_\times' w_i \dot{\theta} \tag{A.15}$$

$$\dot{y}_i - \dot{y}_{i-1} = (H_i - H_{i-1})\dot{\theta} = [y_i - y_{i-1}]_\times' w_i \dot{\theta} \tag{A.16}$$

and

$$\dot{w}_i = (\dot{v}_1, \cdots \dot{v}_i, 0, \cdots, 0) \tag{A.17}$$

$$\dot{v}_j = \sum_{k=1}^{n} \dot{\theta}_k \frac{\partial v_j}{\partial \theta_k} \tag{A.18}$$

.

Using these definitions of J_i and $\dot{J}_i$, we examine each of the terms in the dynamics equation (A.7) and write them in a form linear in the inertial parameters.

A.3 Inertial Forces

We first write (A.8) linear on the inertial parameters m_i,b_i, and I_i.

$$\begin{aligned} M_i\ddot{\theta} =& m_i \underbrace{H'_{i-1} H_{i-1} \ddot{\theta}}_{a_i^{[1]}} \\ & - \underbrace{H'_{i-1} s_i}_{p_i^{[1]}} m_i [b_i]_\times \underbrace{s'_i w_i \ddot{\theta}}_{p_i^{[2]}} \\ & + \underbrace{w'_i s_i}_{p_i^{[3]}} (m_i [b_i]_\times) \underbrace{s'_i H_{i-1} \ddot{\theta}}_{p_i^{[4]}} \\ & + \underbrace{w'_i s_i}_{p_i^{[5]}} I_i \underbrace{s'_i w_i \ddot{\theta}}_{p_i^{[6]}} \end{aligned} \tag{A.19}$$

Then we have

$$\begin{aligned} M_i\ddot{\theta} =& a_i^{[1]} m_i \\ & - p_i^{[1]} m_i [b_i]_\times p_i^{[2]} \\ & + p_i^{[3]} m_i [b_i]_\times p_i^{[4]} \\ & + p_i^{[5]} I_i p_i^{[6]} \end{aligned} \tag{A.20}$$

Next we use the facts that $\text{Vec}[a] = a$ if a is a vector, and:

$$\text{Vec}[abc] = (c' \otimes a)\text{Vec}[b] \tag{A.21}$$

so that

$$\text{Vec}[p_i^{[1]} [m_i b_i]_\times p_i^{[2]}] = ((p_i^{[2]})' \otimes p_i^{[1]})\text{Vec}[[m_i b_i]_\times] \tag{A.22}$$

$$\text{Vec}[p_i^{[3]} m_i [b_i]_\times p_i^{[4]}] = ((p_i^{[4]})' \otimes p_i^{[3]})\text{Vec}[[m_i b_i]_\times] \tag{A.23}$$

$$\text{Vec}[p_i^{[5]} I_i p_i^{[6]}] = ((p_i^{[6]})' \otimes p_i^{[5]})\text{Vec}[I_i] \tag{A.24}$$

Moreover

$$\text{Vec}[[m_i b_i]_\times] = D_3 b_i m_i \tag{A.25}$$

where

$$D_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{A.26}$$

and

$$\text{Vec}[I_i] = D_6 \text{Vec}_6[I_i] \tag{A.27}$$

where

$$D_6 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.28}$$

and and $\text{Vec}_6[I]$ extracts the 6 unique element in the upper triangle of I

$$\text{Vec}_6[I] = [I_{1,1}, I_{1,2}, I_{1,3}, I_{2,2}, I_{2,3}, I_{3,3}]' \tag{A.29}$$

Thus

$$M_i\ddot{\theta} = a_i^{[1]} m_i + a_i^{[2]} m_i b_i + a_i^{[3]} \text{Vec}_6[I_i] \tag{A.30}$$

where

$$a_i^{[1]} = H'_{i-1} H_{i-1} \ddot{\theta} \tag{A.31}$$

$$a_i^{[2]} = \left(-(p_i^{[2]})' \otimes p_i^{[1]} + (p_i^{[4]})' \otimes p_i^{[3]} \right) D_3 \tag{A.32}$$

$$a_i^{[3]} = \left((p_i^{[6]})' \otimes p_i^{[5]} \right) D_6 \tag{A.33}$$

A.4 Coriolis Forces

We can expand (A.9) as

$$\begin{aligned} C_i\dot{\theta} =& m_i \underbrace{H'_{i-1}\dot{H}_{i-1}\dot{\theta}}_{a_i^{[4]}} + \underbrace{H'_{i-1}[w_i\dot{\theta}]^2_\times s_i}_{q_i^{[0]}} m_i b_i \\ & - \underbrace{H'_{i-1}s_i}_{q_i^{[1]}} m_i [b_i]_\times \underbrace{s'_i \dot{w}_i \dot{\theta}}_{q_i^{[2]}} \\ & + \underbrace{w'_i s_i}_{q_i^{[3]}} m_i [b_i]_\times \underbrace{s'_i \dot{H}_{i-1}\dot{\theta}}_{q_i^{[4]}} \\ & + \underbrace{w'_i [w_i\dot{\theta}]_\times s_i}_{q_i^{[5]}} I_i \underbrace{s'_i w_i \dot{\theta}}_{q_i^{[6]}} \\ & + \underbrace{w'_i s_i}_{q_i^{[7]}} I_i \underbrace{s'_i \dot{w}_i \dot{\theta}}_{q_i^{[8]}} \end{aligned} \tag{A.34}$$

Thus

$$\begin{aligned} C_i\dot{\theta} = & a_i^{[4]} m_i \\ & + q_i^{[0]} m_i b_i \\ & - q_i^{[1]} m_i [b_i]_\times q_i^{[2]} + q_i^{[3]} m_i [b_i]_\times q_i^{[4]} \\ & + q_i^{[5]} I_i q_i^{[6]} + q_i^{[7]} I_i q_i^{[8]} \end{aligned} \tag{A.35}$$

Using the properties of Vec$[\cdot]$, this can be written

$$C_i\dot{\theta} = a_i^{[4]} m_i + a_i^{[5]} b_i m_i + a_i^{[6]} \mathrm{Vec}_6[I_i] \tag{A.36}$$

where

$$a_i^{[4]} = H_i' H_i \dot{\theta} \tag{A.37}$$

$$a_i^{[5]} = q_i^{[0]} + \Big(-(q_i^{[2]})' \otimes q_i^{[1]} + (q_i^{[4]})' \otimes q_i^{[3]} \Big) D_3 \tag{A.38}$$

$$a_i^{[6]} = \Big((q_i^{[6]})' \otimes q_i^{[5]} + (q_i^{[8]})' \otimes q_i^{[7]} \Big) D_6 \tag{A.39}$$

A.5 Gravity

The derivation of the torque due to gravity uses the fact that, for an articulated body, when a force f is applied to a point p (in world coordinates) on the articulated body, the vector of torques τ generated at each joint is computed as

$$\tau = \left(\frac{\partial p}{\partial \theta'} \right)' f \tag{A.40}$$

which is the transpose of the Jacobian of p times the force. Since gravity acts at the center of mass of a link, and the force is gm_i, the torque generated by gravity from each

link is $tau_i^g = J_i' g m_i$. Using this and the definition of J_i, we can write

$$\tau^g = \sum_i \tau_i^g \tag{A.41}$$

$$\begin{aligned}
\tau_i^g &= m_i \left(H_{i-1}' + w_i' s_i [b_i]_\times s_i' \right) g \\
&= m_i \underbrace{H_{i-1}' g}_{a^{[7]}} + \underbrace{w_i' s_i}_{h^{[1]}} (m_i [b_i]_\times) \underbrace{s_i' g}_{h^{[2]}} \\
&= a^{[7]} m_i + h^{[1]} m_i [b_i]_\times' h^{[2]} \\
&= a_i^{[7]} m_i + \left((h^{[2]})' \otimes h^{[1]} \right) [m_i b_i]_\times \\
&= a_i^{[7]} m_i + \left((h^{[2]})' \otimes h^{[1]} \right) D_3 \mathrm{Vec}[m_i b_i] \\
&= a^{[7]} m_i + a^{[8]} m_i b_i
\end{aligned} \tag{A.42}$$

Thus

$$\tau_i^g = a_i^{[7]} m_i + a_i^{[8]} b_i m_i \tag{A.43}$$

where

$$a_i^{[7]} = H_{i-1}' g \tag{A.44}$$

$$a_i^{[8]} = \left((h^{[2]})' \otimes h^{[1]} \right) D3 \tag{A.45}$$

A.6 Viscous Friction

The viscous friction on each joint is linear on eta_i.

$$\tau^v = \sum_i \tau_i^v \tag{A.46}$$

$$\tau_i^v = -\dot{\theta}_i \eta_i \tag{A.47}$$

A.7 Equation of Motion

Using the definitions above, we can now express (A.7) as follows

$$\begin{aligned}\tau = &\sum_i (a_i^{[1]} + a_i^{[4]} - a_i^{[7]}) m_i \\ &+ \sum_i (a_i^{[2]} + a_i^{[5]} - a_i^{[8]}) b_i m_i \\ &+ \sum_i (a_i^{[3]} + a_i^{[6]}) \mathrm{Vec}_6[I_i] \\ &+ \mathrm{diag}[\dot{\theta}]\eta\end{aligned} \tag{A.48}$$

In matrix form and making the dependence on time explicit

$$\tau_t = c_t \gamma \tag{A.49}$$

$$\begin{aligned}c_t = \Big(&a_1^{[1]} + a_1^{[4]} - a_1^{[7]}, \ldots, a_n^{[1]} + a_n^{[4]} - a_n^{[7]}, \\ &a_1^{[2]} + a_1^{[5]} - a_1^{[8]}, \ldots, a_n^{[2]} + a_n^{[5]} - a_n^{[8]}, \\ &a_1^{[3]} + a_1^{[6]}, \ldots, a_n^{[3]} + a_n^{[6]}, \mathrm{diag}[\dot{\theta}] \Big),\end{aligned} \tag{A.50}$$

and

$$\gamma = \begin{pmatrix} m_1 \\ \vdots \\ m_n \\ m_1 b_1 \\ \vdots \\ m_n b_n \\ \mathrm{Vec}_6[I_1] \\ \vdots \\ \mathrm{Vec}_6[I_n] \\ \eta_1 \\ \vdots \\ \eta_n \end{pmatrix} \tag{A.51}$$

Appendix B

Isolating Unknown Parameters

If the inertial parameters of some of the links are known, we can modify the terms in (A.49) so that only the unknown parameters are estimated. This is the method used to estimate the inertial parameters of objects in (4.6) given an inverse model of the robot.

To estimate only some links, first separate the terms in c_t and γ into those that are known and those that are unknown.

$$\tau_t = c_t^k \gamma^k + c_t^u \gamma^u \tag{B.1}$$

If there are n total links, k known links, and u unknown links, then

- τ_t is an $n \times 1$ vector
- γ is an $11n \times 1$ vector
- γ^k is an $11k \times 1$ vector
- γ^u is an $11u \times 1$ vector
- c_t is an $n \times 11n$ matrix
- c_t^k is an $n \times 11k$ matrix

- c_t^u is an $n \times 11u$ matrix

  Since $\tau^k = c_t^k \gamma^k$ is known, we are left with

$$\tau_t - \tau^k = c_t^u \gamma^u \tag{B.2}$$

which is in the same form as (A.49). From this, the unknown parameters can be estimated. Note that τ^k can be estimated with any inverse model, not necessarily one that requires knowledge of the inertial parameters.

Appendix C

Estimating Density

In this appendix, we show that γ, the inertial parameters of the links, is itself a linear function of the vector of object densities δ, which we aim to estimate. The 10 inertial parameters of the object, $\gamma = (m, mx, Vec_6[I])'$, are:

- the object's mass, m
- the object's center of mass times its mass, mx
- the object's inertial matrix, I, in the object's frame, with origin o as reference. $Vec_6[I]$ extracts the 6 elements from the upper triangle of the symmetric matrix I.

First, we write I in a form linear on the masses m_i of each component. Using the parallel axis theorem, we can write the total inertia of the object as

$$I = \sum_i I_i - m_i [d_i]_\times^2 \tag{C.1}$$

where d_i is the vector from the origin of the object's frame, o, to μ_i, m_i is the mass of ellipsoid i, I_i is the inertia of ellipsoid i around its center of mass, and $[d_i]_\times$ is the skew-symmetric cross-product matrix constructed from the vector d_i. Since we know the

geometry of ellispoid i, and the form of the inertial matrix for ellipsoids, we can write

$$I_i = \begin{bmatrix} \frac{1}{3}(b_i^2 + c_i^2) & 0 & 0 \\ 0 & \frac{1}{3}(a_i^2 + c_i^2) & 0 \\ 0 & 0 & \frac{1}{3}(a_i^2 + b_i^2) \end{bmatrix} m_i \tag{C.2}$$

$$= G_i m_i \tag{C.3}$$

and

$$I = \sum_i I_i - m_i [d_i]_\times^2 \tag{C.4}$$

$$= \sum_i (G_i - [d_i]_\times^2) m_i \tag{C.5}$$

Also,

$$x = \frac{\sum_i m_i d_i}{m} \tag{C.6}$$

$$m = \sum_i m_i \tag{C.7}$$

We want to estimate

$$\delta = \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_r \end{bmatrix} \tag{C.8}$$

where δ_i is the density of material i. The color of each component indicates which of a set of r materials makes it up. This information gives an $n \times r$ matrix A where element A_{ij} is v_i if object component i is made of material j, and 0 otherwise. Here, v_i is the volume of component i, which is known, and allows estimation of the density of materials rather

than the mass of the components, since

$$\begin{bmatrix} m_1 \\ \vdots \\ m_n \end{bmatrix} = A \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_r \end{bmatrix} \tag{C.9}$$

Combining (C.5), (C.6), and (C.9), we can write

$$\gamma = HA\delta \tag{C.10}$$

where

$$H = \begin{bmatrix} \mathbf{1}_n \\ d_1 \dots d_n \\ \text{Vec}_6[G_1 - [d_1]_\times^2] \dots \text{Vec}_6[G_n - [d_n]_\times^2] \end{bmatrix} \tag{C.11}$$

and $\mathbf{1}_n$ is a row vector of n ones. Finally, we have

$$\tau_t^o = K_t HA\delta \tag{C.12}$$

Here the torque vector τ_t^o, and the K_t, H, A matrices are known. Thus we can infer the desired vector of object densities using simple linear regression methods. In practice, we use Bayesian linear regression methods to update, after each trial, a Gaussian estimate of the density of a set of materials. Here, K is a stacked matrix containing K_t for all observed timesteps in a trial, and similarly τ^o is a stacked vector of all τ_t^o. For a prior belief

$\mathcal{N}(\mu_\delta, \sigma_\delta)$ and our data K and τ_t^o, we get the parameters of the posterior distribution:

$$\begin{aligned} \bar{\sigma}_\delta &= (\sigma_\delta^{-1} + \eta(KHA)'(KHA))^{-1} \\ \bar{\mu}_\delta &= \bar{\sigma}_\delta(\sigma_\delta^{-1}\mu_\delta + \eta(KHA)'\tau^o) \end{aligned} \tag{C.13}$$

Bibliography

[1] W. Talbott, I. Fasel, J. Molina, V. de Sa, and J. Movellan, "Coordinating Touch and Vision to Learn What Objects Look Like," in *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, L. Carlson, C. Hölscher, and T. Shipley, Eds. Cognitive Science Society, 2011, pp. 562–567.

[2] W. Talbott, H. Huang, and J. Movellan, "Continuous-time infomax models of oculomotor control," in *Proceedings of the International Conference of Development and Learning and Epigenetic Robotics*, 2012.

[3] W. Talbott and J. Movellan, "An expected motion information model of salience for active cameras," in *Proceedings of the International Conference of Development and Learning and Epigenetic Robotics*, 2012, **Best Paper Award**.

[4] W. Talbott, T. Wu, and J. Movellan, "Estimating dynamic properties of objects from appearance," in *Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference on*, Aug 2013, pp. 1–6.

[5] W. Talbott and J. Movellan, "A computational framework for visual perception of inertial affordances," in *Affordances in Vision for Cognitive Robotics Workshop at RSS*, 2014.

[6] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, Nov 1998.

[7] L. Itti and C. Koch, "Computational modelling of visual attention," 2001.

[8] L. Zhang, M. H. Tong, T. K. Marks, H. Shan, and G. W. Cottrell, "Sun: A bayesian framework for saliency using natural statistics," *Journal of Vision*, vol. 8, no. 7, 2008.

[9] A. Torralba, "Contextual modulation of target saliency," in *In Advances in Neural Information Processing Systems*. MIT Press, 2002, pp. 1303–1310.

[10] D. Gao, V. Mahadevan, and N. Vasconcelos, "The discriminant center-surround hypothesis for bottom-up saliency," in *Advances in Neural Information Processing Systems 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. Cambridge, MA: MIT Press, 2008, pp. 497–504.

[11] N. Bruce and J. Tsotsos, "Saliency based on information maximization," in *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf, and J. Platt, Eds. Cambridge, MA: MIT Press, 2006, pp. 155–162.

[12] W. Kienzle, F. A. Wichmann, B. Schölkopf, and M. O. Franz, "A nonparametric approach to bottom-up visual saliency," in *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. Platt, and T. Hoffman, Eds. Cambridge, MA: MIT Press, 2007, pp. 689–696.

[13] L. W. Renninger, J. M. Coughlan, P. Verghese, and J. Malik, "An information maximization model of eye movements," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. Cambridge, MA: MIT Press, 2005, pp. 1121–1128.

[14] B. W. Tatler, M. M. Hayhoe, M. F. Land, and D. H. Ballard, "Eye guidance in natural vision: Reinterpreting salience," *Journal of Vision*, vol. 11, no. 5, 2011.

[15] T. Marks, J. R. Hershey, and J. R. Movellan, "Tracking motion, deformation, and texture using conditionally gaussian processes," *Pattern Analysis and Machine Intelligence*, vol. 32, no. 2, pp. 348–363, February 2010.

[16] S. Ramanathan, H. Katti, N. Sebe, M. Kankanhalli, and T.-S. Chua, "An eye fixation database for saliency detection in images," in *ECCV 2010*, Crete, Greece, 2010.

[17] N. J. Butko and J. R. Movellan, "I-POMDP: An infomax model of eye movement," *Proceedings of the 2008 IEEE International Conference on Development and Learning (ICDL)*, 2008.

[18] J. Najemnik and W. Geisler, "Optimal eye movement strategies in visual search," *Nature*, vol. 434, no. 7031, pp. 387–391, 2005.

[19] F. Panerai, G. Metta, and G. Sandini, "Learning vor-like stabilization reflexes in robots," in *in 8th European Symposium on Artificial Neural Networks*, 2000.

[20] M. Hikita, S. Fuke, M. Ogino, T. Minato, and M. Asada, "Visual attention by saliency leads cross-modal body representation," in *Development and Learning, 2008. ICDL 2008. 7th IEEE International Conference on*, aug. 2008, pp. 157 –162.

[21] N. Butko, L. Zhang, G. Cottrell, and J. Movellan, "Visual saliency model for robot cameras," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, may 2008, pp. 2398 –2403.

[22] C. M. Harris and D. Wolpert, "The main sequence of saccades optimizes speed-accuracy trade-off," *Biological cybernetics*, vol. 95, no. 21–29, 2006.

[23] C. M. Harris and D. M. Wolpert, "Signal-dependent noise determines motor planning," *Nature*, vol. 394, no. 6695, pp. 780–784, Aug. 1998.

[24] R. J. Van Beers, "Saccadic eye movements minimize the consequences of motor noise," *PLoS One*, vol. 3, no. 4, p. e2070, 2008.

[25] H. Tanaka, J. W. Krakauer, and N. Qian, "An optimization principle for determining movement duration," *Journal of neurophysiology*, vol. 95, no. 6, pp. 3875–3886, 2006.

[26] J. Najemnik and W. S. Geisler, "Optimal eye movement strategies in visual search," *Nature*, vol. 434, no. 7031, pp. 387–391, Mar. 2005.

[27] N. J. Butko and J. R. Movellan, "Infomax control of eye movements," *Autonomous Mental Development, IEEE Transactions on*, vol. 2, no. 2, pp. 91–107, 2010.

[28] D. Robinson, "Models of the saccadic eye movement control system." *Kybernetik*, vol. 14, no. 2, pp. 71–83–, Dec. 1973.

[29] T. Erez and E. Todorov, "Inverse dynamics optimal control for domains with contacts," 2011.

[30] E. Todorov and W. Li, "A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *American Control Conference, 2005. Proceedings of the 2005*, june 2005, pp. 300 – 306 vol. 1.

[31] H. Collewijn, C. Erkelens, and R. Steinman, "Binocular co-ordination of human horizontal saccadic eye movements." *J Physiol*, vol. 404, pp. 157–82–, 1988.

[32] H. Huang, M. Plank, S. Gepshtein, and H. Poizner, "Target predictability and eye-hand coordination in a rapid reaching task," *Journal of Vision*, vol. 12, no. 9, p. 411, 2012.

[33] H. Chen-Harris, W. M. Joiner, V. Ethier, D. S. Zee, and R. Shadmehr, "Adaptive Control of Saccades via Internal Feedback," *The Journal of Neuroscience*, vol. 28, no. 11, pp. 2804–2813, 2008.

[34] D. A. Robinson, J. L. Gordon, and S. E. Gordon, "A model of the smooth pursuit eye movement system," *Biological Cybernetics*, vol. 55, pp. 43–57, 1986, 10.1007/BF00363977.

[35] C. J. Erkelens, "Coordination of smooth pursuit and saccades," *Vision Research*, vol. 46, no. 1–2, pp. 163–170, 2006.

[36] S. Grossberg, K. Srihasam, and D. Bullock, "Neural dynamics of saccadic and smooth pursuit eye movement coordination during visual tracking of unpredictably moving targets," *Neural Netw.*, vol. 27, pp. 1–20, Mar. 2012.

[37] J.-J. Orban de Xivry and P. Lefevre, "Saccades and pursuit: two outcomes of a single sensorimotor process," *The Journal of Physiology*, vol. 584, no. 1, pp. 11–23, 2007.

[38] P. Mounoud and T. Bower, "Conservation of weight in infants," *Cognition*, vol. 3, no. 1, pp. 29 – 40, 1974.

[39] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini, "Learning about objects through action - initial steps towards artificial cognition," in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 3, sept. 2003, pp. 3140 – 3145 vol.3.

[40] A. Stoytchev, "Learning the affordances of tools using a behavior-grounded approach," in *Towards Affordance-Based Robot Control*, ser. Lecture Notes in Computer Science, E. Rome, J. Hertzberg, and G. Dorffner, Eds. Springer Berlin / Heidelberg, 2008, vol. 4760, pp. 140–158.

[41] J. Sinapov and A. Stoytchev, "Learning and generalization of behavior-grounded tool affordances," in *Development and Learning, 2007. ICDL 2007. IEEE 6th International Conference on*, july 2007, pp. 19 –24.

[42] R. Jain and T. Inamura, "Learning of tool affordances for autonomous tool manipulation," in *System Integration (SII), 2011 IEEE/SICE International Symposium on*, dec. 2011, pp. 814 –819.

[43] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning object affordances: From sensory–motor coordination to imitation," *IEEE Journal of Robotics*, vol. 24, no. 1, pp. 15–26, 2008.

[44] T. Hermans, J. M. Rehg, and A. Bobick, "Affordance prediction via learned object attributes," in *International Conference on Robotics and Automation Workshop on Semantic Perception, Mapping, and Exploration*. Citeseer, 2011.

[45] G. Pezzulo, M. Candidi, H. Dindo, and L. Barca, "Action simulation in the human brain: Twelve questions," *New Ideas in Psychology*, 2013.

[46] H. Imamizu, S. Miyauchi, T. Tamada, Y. Sasaki, R. Takino, B. PuÈtz, T. Yoshioka, and M. Kawato, "Human cerebellar activity reflecting an acquired internal model of a new tool," *Nature*, vol. 403, no. 6766, pp. 192–195, 2000.

[47] G. Cross and A. Zisserman, "Quadric reconstruction from dual-space geometry," in *International Conference on Computer Vision*, Jan 1998, pp. 25–31.

[48] J. Movellan, "Minimum angular acceleration control of articulated body dynamics," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, Oct 2012, pp. 5006–5011.

[49] J. R. Movellan, *Physics for Robotics and Animation: A Gaussian Approach*. MPLab, UCSD, 2011.

[50] P. Corke, "A robotics toolbox for matlab," *IEEE Robotics and Automation Magazine*, vol. 3, no. 1, pp. 24–32, 1996.

[51] J. I. Yellot, "The relationship between luce's choice axiom, thustone theory of comparative judgement, and the double exponential distribution." *Journal of Mathematical Psychology*, vol. 15, pp. 109–144, 1977.

[52] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, and E. Todorov, "An integrated system for real-time model predictive control of humanoid robots," *International Conference on Humanoid Robots*, 2013.

[53] S. Haykin, *Neural networks : a comprehensive foundation*. Upper Saddle River, N.J: Prentice Hall, 1999.

[54] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015, published online 2014; based on TR arXiv:1404.7828 [cs.NE].

[55] E. Todorov, C. Hu, A. Simpkins, and J. Movellan, "Identification and control of a pneumatic robot," in *Biomedical Robotics and Biomechatronics (BioRob), 2010 3rd IEEE RAS and EMBS International Conference on*. IEEE, 2010, pp. 373–380.

[56] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[57] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Computed torque control with nonparametric regression models," in *American Control Conference, 2008*, 2008, pp. 212–217.

[58] "Torch, a scientific computing framework," torch.ch.

[59] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[60] J. N. Ingram, I. S. Howard, J. R. Flanagan, and D. M. Wolpert, "Multiple grasp-specific representations of tool dynamics mediate skillful manipulation," *Current Biology*, vol. 20, no. 7, pp. 618 – 623, 2010.

[61] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.