EMPIRICAL EVALUATION OF INFORMATION FOR ROBOTIC MANIPULATION TASKS

By

JANE MULLIGAN

B.C.S.H. Acadia University 1984

M.Sc. (Computer Science) University of British Columbia 1988

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES
(DEPARTMENT OF COMPUTER SCIENCE)

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

August 1996

© I.J. Mulligan, 1996

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Computer Science

The University of British Columbia
Vancouver, Canada

Date August 27, 1996

DE-6 (2/88)

Abstract

Rigorous analysis and evaluation of real implemented robotic systems for intelligent tasks are rarely performed. Such systems are often extremely complicated, depending not only on 'interesting' theoretical parameters and models, but on many assumptions and constants which may be set almost arbitrarily. The information required to achieve good task performance includes all of these acknowledged and unacknowledged parameters. In order to evaluate and compare systems which employ diverse components and methods, we need a framework and criteria by which to measure them. We demonstrate techniques for evaluating a system's performance as a function of the parameters and information it uses and compare different task implementations on this basis.

We view all task implementations as particular parameterizations of the task goals they represent. Some parameters belong to fixed, pre-measured models; others are data collected or measured online by the system. Comparing systems then becomes the comparison of these parameterizations. There are three key questions we need to ask when determining task parameterizations: What do we need to measure (observability)? How well must we discriminate between measured values (precision)? and How accurately must our measurements reflect the true world state (accuracy)? We present a performance based framework for empirical evaluation and comparison of task information based on these three basic notions of *observability*, *precision* (discrimination) and *accuracy*. Factorial experiments give us a starting point for determining which measurements, and their interactions, define the task subspace. Sensitivity analysis determines optimal parameter values and the effect of uncertainty and variations in measurements on task performance. Finally a cost/performance metric offers a quantification of the task complexity with respect to the parameters, and performance based precision and accuracy requirements determined in the previous steps.

The experiments presented to demonstrate this methodology are based on a part orienting task implemented in two very different ways. One system uses a 'sensorless' model-based push-orienting method, the other uses a real-time stereo vision system to

localize objects in the workspace for sensor-driven part orienting. The parameters used to represent manipulated parts for sensing versus model-based manipulation are similar, though not identical sets, and encode the information critical to the task. Through detailed experiments we establish the statistically significant parameters and parameter interactions for each system, and apply sensitivity analysis to set optimal parameter values and explore the nature of interactions. Finally, the cost/performance metric gives us a means of counting the computation and sensing costs to achieve the observed system error rates. This type of analysis is a necessary step to understanding integrated intelligent systems. It reveals aspects of system implementations which cannot easily be predicted in advance, and gives a clear picture of the information required and the strengths and weaknesses of the system.

Table of Contents

Abstract ii

Table of Contents iv

List of Tables viii

List of Figures ix

Acknowledgment xi

1 Exploring Information for Robotic Tasks 1

1.1 Information for Robotic Tasks 2

1.1.1 Sensing for Action 5

1.1.2 Parameterization, Representation and Features 7

1.2 Detailed Analysis of Two Information Modalities 8

1.2.1 Part Orienting 8

1.2.2 Framework for Performance-based Evaluation of Information . . . 9

1.3 Thesis Plan 13

2 Information for Robotic Tasks 14

2.1 Measurement and Information 14

2.2 Task Level Parameterizations 16

2.3 State Spaces, Observability and Information for Tasks 18

2.3.1 Task Based Perception 20

2.3.2 Features and Information 25

2.4 Evaluating Task Performance 28

2.4.1 Complexity of Robotic Tasks 28

2.4.2 Part Orienting 30

2.5 Summary 32

3 Evaluating Task Parameterizations **33**

3.1 Identifying the Task Subspace 34

3.1.1 Design of Experiments 35

3.2 Precision and Accuracy 43

3.2.1 Sensitivity Analysis 43

3.2.2 Measuring Cost versus Performance 45

3.3 Comparing Implementations 46

3.3.1 Part Orienting Systems 47

3.4 Framework Summary 48

4 Push Orienting **50**

4.1 Apparatus 50

4.2 The Robot 51

4.2.1 Straight Line Interpolated Motion 52

4.3 Push Planner 52

4.4 Robust Push Plans 55

5 Vision-based Part Orienting **59**

5.1 Introduction 59

5.2 Stereo Vision 61

5.2.1 Calibration 61

5.2.2 Standard Transformations 62

5.2.3 Edge-based Correlation 64

5.2.4 DATACUBE Pipeline Implementation 64

5.3 Part Localization 74

5.3.1 Right Angle Part Localization 75

5.3.2 "G" Localization 78

5.4 Summary 80

6 Experimental Results **81**

6.1 System Task Parameterizations 81

6.1.1 Test Parts 85

6.2 Methods 87

6.2.1 The $\frac{1}{8}$ and $\frac{1}{4}$ Fractional Factorial Designs 87

6.2.2 Simulators 89

6.3 Push Planner — Parameters and Outcomes 90

6.3.1 Triangle Push Planner Results 93

6.3.2 "L" Push Planner Results 95

6.3.3 "G" Push Planner Results 101

6.4 Vision and Manipulation — Parameters and Outcomes 104

6.4.1 Triangle Vision-based Orienter Results 106

6.4.2 "L" Vision-based Orienter Results 109

6.4.3 "G" Vision-based Orienter Results 114

7 Analysis of Results **119**

7.1 Optimization of Information for Tasks 119

7.2 Cost/performance Analysis 122

7.2.1 The Formalism 122

7.2.2 Vision Orienting 124

7.2.3 Push Orienting 128

7.2.4 Summary 131

8 Conclusion **132**

8.1 Information for Tasks 132

8.1.1 A Methodology for Comparing and Evaluating Task Implementations 132

8.2 Comparison of Task Implementations 133

8.3 Future Work 134

Bibliography **135**

Appendices **141**

A Treatments and Outcomes - Push Planner **141**

A.1 Triangle 141

A.1.1 Success Rate 141

A.1.2 Planning Time 145

A.2 "L" Part 148

A.2.1 Success Rate 148

A.2.2 Planning Time 152

A.3 "G" Part 155

A.3.1 Success Rate 155

A.3.2 Planning Time 159

B Treatments and Outcomes - Vision Orienter **162**

B.1 Triangle 162

B.1.1 Success Rate 162

B.1.2 Extraction Time 164

B.2 "L" Part 166

B.2.1 Success Rate 166

B.2.2 Extraction Time 168

B.3 "G" Part 170

B.3.1 Success Rate 170

B.3.2 Extraction Time 172

List of Tables

3.1 ANOVA Table 38
3.2 Factorial coefficients 39
3.3 ANOVA for Factorial Experiments 42

6.1 Push Planner Parameters 82
6.2 Vision System Parameters 84
6.3 Push parameter levels 90
6.4 Triangle push planner success ANOVA 91
6.5 Triangle push planner time ANOVA 92
6.6 "L" push planner success ANOVA 97
6.7 "L" push planner time ANOVA 98
6.8 "G" push planner success ANOVA 101
6.9 "G" push planner time ANOVA 102
6.10 Vision parameter levels 106
6.11 Triangle vision success ANOVA 107
6.12 Triangle vision time ANOVA 108
6.13 "L" vision success ANOVA 110
6.14 "L" vision time ANOVA 111
6.15 "G" vision success ANOVA 115
6.16 "G" vision time ANOVA 116

List of Figures

1.1 Visual Part Orienter 4
1.2 Fence part feeder 9

2.1 Basic dynamic system diagram 18
2.2 Indiscriminable Sets Induced by Measurement 20

3.1 Task black box 33
3.2 Computing BC interaction 40

4.1 Equipment. 51
4.2 Configuration Maps 53
4.3 Alignment error 56

5.1 Stereo Geometry 62
5.2 MaxVideo 200 Components 65
5.3 Continuous Datacube Pipes 67
5.4 Left Camera 70
5.5 Right Camera 70
5.6 Windowing histogram 71
5.7 Epipolar Approximation 74
5.8 Theta Map 76
5.9 Lines reprojected into image 77
5.10 Data and Line Fits 78
5.11 "G" Vision Model 79

6.1 Triangular part 85
6.2 L-shaped part 86
6.3 G-shaped part 86

6.4 Triangle Sensitivity Surface in D and L 94
6.5 Triangle Search for L with D=26 95
6.6 Triangle push plan success re: part variation 96
6.7 Extended "L" Sensitivity Surface in D and L 99
6.8 "L" push plan success re: part variation 100
6.9 "G" Sensitivity Surface in D and L 103
6.10 Extended "G" Sensitivity Surface in D and L 104
6.11 "G" push plan success re: part variation 105
6.12 Rho discretization versus success 109
6.13 Rho discretization versus time 110
6.14 Number of disparities versus success 112
6.15 Number of disparities versus time 113
6.16 Rho discretization for selected Disparities 113
6.17 Rho discretization for selected disparities versus time 114
6.18 Gap/discretization for "G' 117
6.19 Optimizing gap/discretization for "G' 118

7.1 Histogram of vision system position error for "G" 126
7.2 Histogram of vision system orientation error for "G" 127
7.3 Histogram of push plan orientation error for "G" 130

Acknowledgement

I would like to thank Greg Grudic for his constant support and encouragement throughout my PhD, without his help it would have been impossible. Thanks also to Alex Kean who always had insightful comments and encouraging words to contribute. My gratitude to my supervisor Alan Mackworth for supporting me and smoothing the way through the latter, dark days of my degree. Thanks to Peter Lawrence, who got me into robotics in the first place, and to the rest of my supervisory committee who stuck with me even when my ideas seemed far fetched or ill defined. Finally thanks to the many friends whose discussion, suggestions and commiseration helped me through the process.

Chapter 1

Exploring Information for Robotic Tasks

Rigorous analysis and evaluation of real implemented robotic systems for intelligent tasks are rarely performed. Such systems are often extremely complicated, depending not only on 'interesting' theoretical parameters and models, but on many assumptions and constants which may be set almost arbitrarily. The information required to achieve good task performance includes all of these acknowledged and unacknowledged parameters. For any task goal there are many potential robotic systems which could execute a plan to achieve it. These would employ a variety of actuators, information sources and computations. We would like to design task solutions which are optimal, but in order to evaluate and compare systems which employ diverse components and methods, we need a framework and criteria by which to measure them. In the absence of clear and detailed specifications for complex robotic tasks, the process of determining what information is required will inevitably be a cyclical and experimental process. This thesis demonstrates an empirical, performance-based methodology for analysing and comparing the information used in implemented robotic manipulation tasks.

In the quest to build intelligent agents, we often study particular subsystems such as reasoning, sensing or motion planning without reference to how to integrate them for the particular tasks we wish to achieve. Often the result is that the agent's subsystems are particularly well tuned to perform in ways which are not relevant to the task at hand. This may be the result of having the elusive goal of building intelligence, making the target tasks ever more ambitious. The goals of intelligent agents are often complicated and difficult to specify and therefore the idea of determining the sensors or 'instrumentation' which will satisfy their requirements is equally vague. The framework provided in this thesis gives specific empirical techniques for analyzing which measurements are critical and what precision and accuracy are required to achieve the desired level of task

performance. This allows us to optimize our systems with respect to the task at hand.

1.1 Information for Robotic Tasks

In recent years researchers have begun to look at the problem of determining what information is required for robotic tasks. Donald [18] has proposed the idea of information invariants to formally address this question. He describes sensory systems as components embedded in the world with some level of calibration or co-designation. These systems can be permuted and compared to system specifications and each other. He proposes that a measure of information invariants may imply a measure of hardness or difficulty and hence a notion of lower bounds in robotics. Others have proposed "RISC" robotics which would use simple, minimal sensors and actuators to achieve complex goals [14]. On the sensor side there is a new press for metric vision, which would apply more rigorous evaluations to the performance of high level vision systems.

Current practice with respect to sensor system design for intelligent agents, however, is often to attach a sophisticated existing system to the robot and hope it delivers the desired results. For example, given the most sophisticated vision system available, if such a system cannot return relevant information, it is not useful for furthering the task. Obviously sensors, and the time and computation required to extract relevant data from them, as well as the space and computation required to evaluate internal models, all have a significant cost. Given the scarcity of resources, we need to examine exactly what information is required for a particular task in a particular environment. Ideally, we could identify the optimal or necessary and sufficient information for a robotic system to execute a task with acceptable reliability. Realistically, we can begin by studying the systems we have and analysing how the information they use, in the form of models or sensor readings, affects their performance.

For any agent, organic or electro-mechanical, the world consists only of what it can perceive or infer from its perceptions. We often think of robots as acting in the world we perceive, but in fact they live and act in the world of their own perceptions as generated by their suite of sensors. For example robots such as Brooks's Atilla are often touted

as acting sensibly in a complex world, but their world consists only of "obstacle" or "not obstacle", warm moving object or not, and thus their projection of the world is not really very complex. These robots are successful only so long as this subspace, and their reactions to it, allow them to succeed in their tasks, in the environment where they live. What is really interesting in reactive robots and in nature, is how well such projections of the environment satisfy the execution of task goals.

Models and sensors used by robotic systems are only valuable to the extent that they contribute to the performance of the robot's task. We need to answer questions such as "Is the subspace of the world observable by the robot the same as the subspace of the task?" and "Can we transform the task into this subspace?" In fact systems from planning to reactive control are computations on simplified models and they all work as long as the model and computation are adequate to produce or approximate the desired results.

One of the defining characteristics of tasks for intelligent agents is their complexity. They are often composed of numerous subgoals all conditioned on preceding events and the current world state. There are many ways to represent a robotic task. Probably the most common descriptions are the symbolic models of STRIPS-like planners, configuration space (C-SPACE) approaches, or methods based on the principles of control systems. Those in the reactive robotics camp have claimed that simple control loops are sufficient to generate intelligent robot behaviours [9]. Advocates of 'sensorless' manipulation claim that all you need is a good world model [42]. Broadly speaking any representation must account for the information (sensed or modelled), actuators and computations (match, search etc.) available to the system and the requirements of its task. There is as yet no accepted formal description of complicated intelligent tasks, particularly in light of the variety of actuators, sensors and computations available. In particular the difficulty and characteristics of obtaining sensor data are not accounted for, except perhaps by a simple uncertainty region. Our current best tool for comparing and evaluating such systems is thus to compare and evaluate their performance.

We take our notion of information from the field of Measurement Science [64]. Mea-

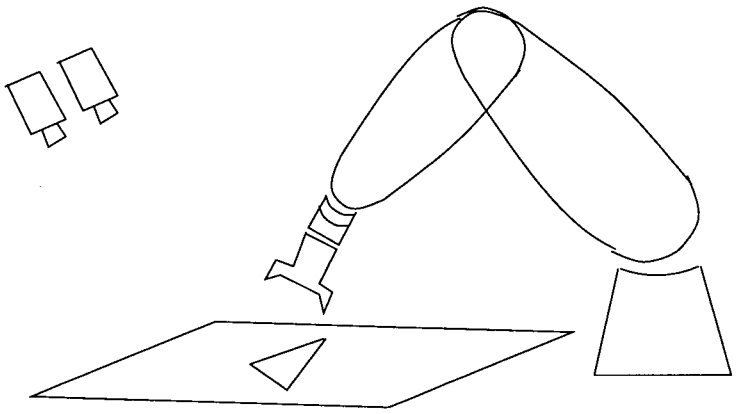


Figure 1.1: A possible stereo vision based part orienter.

surement involves extracting information from objects or events in the physical world. The raw numbers recorded by the measurement system are called *data*; they are converted to information via calibration. Calibration compares data with some agreed measurement standard to assign them common meaning or symbolism. For our purposes then, *information* is the representation of physical phenomena via standardized symbolism. We will view sensors as measurement instruments, models are stored information which at some point was obtained by a measurement system. *Knowledge*, in turn, is some interpretation of a collection of information.

Let us consider, for example, the simplified part manipulation system illustrated in figure 1.1, consisting of a robot manipulator arm and a pair of stereo cameras. The task goal is to pick up the part and place it in a known position and orientation. Disparities extracted from a stereo vision system are only data until we apply a calibrated camera model to extract depth points from them. They then become position information which, via model fitting or other localization technique, becomes the knowledge of part position required by the task. Calibration may be more or less precise depending on the number of (world, image) points used to generate the camera model, and the degree to which those points span the workspace. For this example the part orienting task is parameterized by estimated part position $\hat{p}$ and the calibrated camera model C, which refers data to the robots coordinate frame, as well as the aperture of the gripper g which determines

how much error in $\hat{p}$ can be tolerated. If the aperture is large in comparison to the part dimension, estimation of $\hat{p}$ can be less accurate while still allowing a successful grasp. The performance or success rate S of our orienting system is a function parameterized by these quantities, thus $S = f(\hat{p}, C, g)$.

Clearly this example is an extreme simplification in comparison to real robotic systems. Indeed, any real system will be far more complex than any model we can build of it. One of the goals of the framework presented in this thesis is to be able to analyse even very complicated systems, and determine how their many parameters act and interact to affect system performance.

1.1.1 Sensing for Action

Overall there are three main requirements for sensing for robotic tasks: *observability* — we must know "what-to-sense" to make progress toward the goal; *precision* — our sensors must be able to detect the desired parameters at the required discrimination; and *accuracy* — our information must be calibrated or reconciled to some common frame such as the workspace of the actuators. Clearly for our vision-based part orienter example, the part position must be observable in order for the arm to manipulate it, position values must be returned with discrimination on the order of a centimetre, and the accuracy must be within a centimetre or two of the true part position.

We conjecture that there is information inherent for a robotic problem; in other words, to successfully solve a robotic problem certain information is required to specific precision in order to adequately compute the result. Which information sources provide this fundamental information is a separate *design issue*.

Observability

What events in the world need to be sensed, modelled or measured to achieve a particular task goal? In modern linear state space control the concept of observability refers to the ability to identify any state of the system, based on some finite sequence of future sensor readings [25]. This admits specific tests of the matrix formulation of the system to

determine whether all states are observable. Of course for many systems we have no tidy set of differential equations, but clearly these systems must be able to distinguish task critical states. The idea of the robot being limited by its sensors and models has however, been addressed by work on task directed sensing [24], planning in recognizable sets [21, 43] and landmark-based navigation [17, 34, 35]. In this context the system parameters which must be observable, and hence are essential to task execution, suggest a minimal information set.

A related idea in Measurement Science is problem boundary or the limit of significant outside influences to the measurement system. No specific techniques are offered to determine this boundary except for cyclical refinement of the system model.

Precision

Not only what the agent sees but how clearly it sees is important to achieving a task. Obviously a system such as ROBODOC which is intended to perform hip replacements [32], operates under much more stringent standards of precision than a system in a less critical task. Precision has three aspects: *discriminability* is the resolution of changes in a measured quantity which can be observed; *repeatability* is the difference between measured values of the same world value in the short term; and *reproducibility* is the difference between measured values of the same world value in the long term. We will assume that tasks have an inherent precision and complexity. Generally a task requires certain physical differences between the goal state and the world state to be resolved to a level within some specified tolerance. Speed is also critical for sensing for action and is frequently traded off with other performance criteria such as discrimination.

Accuracy

The work described in this thesis began with an examination of hand-eye coordination for visually guided manipulation. In many instances such coordination takes the form of precise calibration between the camera coordinate frame and the manipulator workspace. We have also seen that in order for measured data to actually become meaningful infor-

mation it must be referred to some common standard by means of calibration. Whether achieved by an adaptive learning scheme or by rigorous fitting of world values to sensor values, relating sensor readings to the real world is critical to the agents ability to act in that world.

1.1.2 Parameterization, Representation and Features

Complex 'intelligent' robotic systems are often a patchwork of sensors and actuators spread across several architectures. If we want to view real systems in a unified way for purposes of comparison and evaluation, we need a framework in which to describe and analyse them.

The study of physical systems typically cycles through three steps: determining minimal model parameterizations; predicting observations using the laws applying to the forward model; and solving for precise model parameters using observations and the inverse problem. For robotic systems the idea of minimal models is a relatively recent, but important step. Viewing these systems as parameterizations with features, thresholds, constants, and constraints as the task level parameters of interest, gives us a unifying representation common to such diverse fields as economics, statistics, control, and inverse theory. It allows the use of well established methods for analysis and comparison. In existing systems task level parameters are often referred to as "features". In the terminology of measurement such features will usually be measurements of type indirect (composed from known relationships between other measurements) or combined (found by solving systems of equations formed from measurements) [52].

For robotic systems there remains much to be done in terms of identifying such issues as how sensing, modelling, actuation, and computation are used and interchanged for particular tasks. Many implementations are possible for any task. If we compare them we may learn something fundamental about the task itself. For industrial applications it may be sufficient to create a framework where we can look at the costs and benefits of various task implementations and recommend the one which is optimal under specified measures. As a step toward these goals we present specific experimental methods for

evaluating and comparing real task implementations, since a better understanding of the systems we have, should give us insight into how such systems can be made optimal.

1.2 Detailed Analysis of Two Information Modalities

There is currently no standard empirical framework for evaluating and comparing robotic task implementations. Donald has introduced the premise that we can reduce or compare resources of one task implementation to another to study information invariants [18]. On a more practical level we will demonstrate detailed experimental analyses of implementations of the same robotic task. The task we have chosen is the part orienting task popular in the literature on sensorless manipulation. Logically, if, as claimed, this work has successfully moved task information entirely to the domain of models, it may give us insight into the information actually in use. A part can be oriented by a number of different devices. Perhaps because of the low dimensionality (resolving only the angle of the part) and hence limited state space of the task, part orienting has become the focus of considerable study.

1.2.1 Part Orienting

Part orienting is a problem encountered in automatic assembly. In order to assemble a product in a factory, parts must generally be presented to a work cell in a known position and orientation. The importance of this task in industry is attested to by the variety of devices devised to execute it.

A common industrial part manipulator is the vibratory bowl feeder. The bowl feeder contains a collection of randomly oriented parts which as a result of the vibration, move up a track on the interior of the bowl. The physical properties of the parts cause them to be oriented in one of a number of stable resting positions. At the top of the track parts encounter a series of features in the bowl wall and track which cause them to be either reoriented or rejected and dropped back into the bowl [13].

Another part orienting strategy formalized by Peshkin and Sanderson [51] is that of parts on a conveyor belt pushed past a series of specially designed fences of various

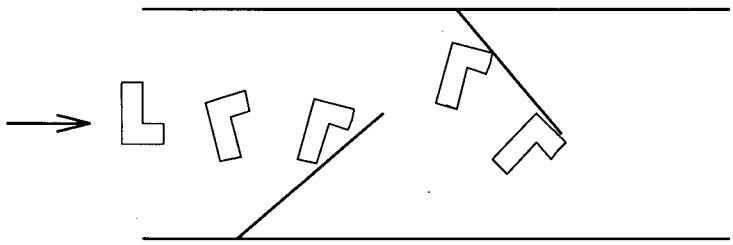


Figure 1.2: Parts travel along the belt and are "pushed" by the fences along the way to exit in a desired range of final orientations.

orientations and lengths (see figure 1.2). As parts pass the last fence they lie in a single desired orientation. The passive fences with moving conveyor act in a manner similar to an active pusher operating on a statically supported object.

Another, more general, manipulation system which is capable of part orienting is a robot arm with one or more cameras similar to our example system. A part is localized by the vision system so that the arm can directly orient it using a simple parameterized pick and place plan.

Other proposed techniques include 'squeeze grasps' where a sequence of oriented squeezes by a robot gripper can be used to establish a particular orientation [27, 10]

Obviously the task goal is the same for each of these proposed implementations, but the nature of the sensing, models and manipulation differ significantly. Clearly they all entail physical or sensory localization of the part, and some degree of manipulation.

1.2.2 Framework for Performance-based Evaluation of Information

Our hypothesis is that different task implementations can be described as different feature-based parameterizations of the same problem. Analysis and comparison of these parameterizations tells us something fundamental about the information being used. More specifically, for any proposed task solution it is invaluable to have a clear analysis of system parameters and how they interact with each other and with task success. We have implemented two part orienting systems: a push orienter and a vision-based manipulation system. There are a number of techniques available in the literature which allow us to analyse the requirements of parameter observability, precision and accuracy

which characterize task information. These include factorial experiments, sensitivity analysis and analyses which suggest cost/performance metrics.

Factorial Experiments

Factorial experiments [33] are applied to systems where a number of parameters or factors may interact to affect system outcome. They are designed to give the researcher information on the effects and interactions of all factors simultaneously while limiting the number of experimental trials required. "Taguchi's method" is a version of such designs advocated for improving manufacturing processes [55]. Generally k factors are tested at each of l levels yielding an l^k factorial experiment. The theory of experimental design also gives us analysis of variance techniques which allow us to determine whether effects observed as a result of setting parameters to various levels are statistically significant. Thus for a system which we hypothesize to have k possibly interacting parameters, we can run specifically designed experiments and use statistical tools to determine whether each factor or interaction has a significant effect on outcome. For us this approach helps answer the question of what to sense: we will devote our resources to sensing and optimizing those parameters with significant impact on task performance.

For our example system in figure 1.1, we could test all combinations of the three parameters $\hat{p}$, C, and g at each of two levels. Levels can be qualitative as well as quantitative, for example the levels of $\hat{p}$ might be two different techniques for extracting part position from image disparities in two cameras. Such a 2 level factorial design would allow us to analyse, at least at a coarse level, the significance of each factor and interaction with respect to the observed variance in S. We could then devote our resources to the most significant factors or combinations of factors. Gaining a sense of the importance of factor interactions is useful when tradeoffs occur in complex systems, as they are not always easy to observe without this sort of principled experiment.

Sensitivity Analysis

Sensitivity analysis [31] determines how a model varies with its parameters. Analytical or empirical methods are used to track system outcomes as parameters are modified. This leads us to more evidence for what information is critical to task performance. We apply sensitivity analysis to determine the precision and accuracy required to achieve desired system performance, and to reinforce our hypotheses about which system parameters are most significant.

Again referring to the example system, suppose we have determined that the technique designated the "high" level of $\hat{p}$ significantly outperforms the other method. Suppose also that we have observed that calibration and gripper aperture interact significantly. With this evidence we can simultaneously optimize gripper aperture as well as degree of calibration C with respect to the preferred extraction method, using sensitivity analysis techniques.

Cost/performance Measures

When proposing computer algorithms for task solutions it is usual to provide a statement of the computational cost or complexity of the method. We want to provide a similar statement of cost versus performance for the robotic task implementations we describe. A promising candidate is information-based complexity theory (ε-complexity) which is a part of analytical or continuous complexity theory; it attempts to determine the intrinsic difficulty of approximating continuous problems, given only expensive, partial, inaccurate information [49]. It determines the computational cost required to achieve a specified uncertainty level ε. This provides us with a convenient metric for calibration and accuracy characteristics on a sensori-computational system. Factorial experiments and sensitivity analysis tell us which parameters are of interest and what precision and accuracy are required; the notation and abstractions of ε-complexity can then be used to suggest the complexity of these requirements.

ε-complexity assigns the usual complexity values to the computational parts of an approximation system, but adds a constant c for every sensor evaluation required. For our

example the calibration data requires *a priori* sensor evaluations as well as computational model fitting, while the online estimation of $\hat{p}$ requires obtaining disparities as well as some search process to localize the part based on them. Empirically or analytically we can determine the error ε achieved by the system. If n_C is the number of sensor values for C and n_p the number for $\hat{p}$ then we add $n_p \cdot c + n_C \cdot c$ to the combinatoric complexity of the system to obtain the ε-complexity for the determined ε.

Evaluation

The above techniques form the basis of our framework for performance based analysis of robotic task information. We want to answer the questions "what should we sense?", "how precisely?" and "with what degree of accuracy?" in order to establish the three basic characteristics of task information. We conclude that intelligent robotic systems are parameterized by measurements which are used online or stored as models. Factorial experiments offer one way of determining what parameters must be observable. Sensitivity analysis can be used to verify the importance of particular parameters and determine the precision and accuracy required to achieve desired task performance. Finally ε-complexity allows us to quantify the difficulty of achieving these performance levels, and compare different implementations of the same task.

For the two real part orienting systems we have analysed in detail we began by determining a comprehensive parameter set for each system (11 parameters for the push planner, 9 for the vision and manipulation system). The methodology described above was applied based on these parameterizations for each of 3 parts (a triangle, an "L" shape and a "G" shape). Perhaps the most surprising results were the differences between the sets of parameters which proved significant for each part. Even for the same orienting system, the different parts demonstrated different sets of significant parameters and interactions. For the push planner the analysis clearly indicated the increasing complexity of the parts with respect to push orienting. For the vision system analysis turned up important interactions among model parameters, which sensitivity analysis then allowed us to explore specifically.

Clearly this methodology reveals aspects of task implementations which are not easy to predict in advance, as well as demonstrating a system's strengths and weaknesses. The result is a detailed picture of the information requirements of the particular part orienting task studied, and the way in which information is encoded in various parts of the implemented systems. Rigorous testing and analysis of intelligent robotic systems to determine key system parameters and their effect on performance, is critical to providing and understanding robust task solutions. It provides a necessary first step toward developing a quantitative science of robotics.

1.3 Thesis Plan

Chapter 2 reviews material on task directed sensing, system parameterizations, feature-based models, and system performance evaluation techniques.

Chapter 3 presents the basic evaluation methodology for exploring task information for a particular task parameterization, including factorial experiments, sensitivity analysis, information-based complexity.

Chapter 4 describes the equipment used for our empirical tests and the details of the push orienter. Chapter 5 describes the fast calibrated stereo system used for our vision-based part orienter.

Chapter 6 contains the empirical results of our system evaluations. Chapter 7 analyzes these observations and chapter 8 summarizes the conclusions.

Chapter 2

Information for Robotic Tasks

What characterizes the information used in robotic tasks? Clearly the information available to any system is limited to what can be measured, either *a priori* in the form of models, or online by its sensors. To clarify this notion of information from measurements, section 2.1 introduces definitions from measurement science. We then propose a standard view of intelligent robotic systems as parameterizations in a physical parameter space of models and data. In section 2.3 we introduce our basic premise that there are three aspects to information for robotic tasks: observability, precision and accuracy. Section 2.3.1 reviews work in task directed sensing including planning in the recognizable sets, sensor planning and information invariants. Frequently the measurements used in sensory and robotic systems are described as "features". Section 2.3.2 reviews the types of features used in biological, sensor and robotic systems. Finally, some formal methods for evaluating the information and performance of robotic systems are discussed.

2.1 Measurement and Information

When we refer to information for tasks we will be referring to information in the sense used in Measurement Science. To make our usage more precise we include the definitions below, following Sydenham *et al.*'s "Introduction to Measurement Science" [64].

Measurement only occurs when a measurement system interacts with the physical world. Properties of physical objects or events to be measured are assumed to have a 'true value' which can never actually be known because there is no perfect instrument to determine it.

Definition 2.1 measurement: *extracting information from objects or events in the physical world.*

A signal passes from the physical world through the measuring system and is recorded in a numerical form as 'data'.

Definition 2.2 calibration: *compares data with some agreed measurement standard to assign them common meaning or symbolism.*

Calibration allows data to be referred to standard units so that their meaning can be used and shared, effectively converting it to information.

Definition 2.3 information: *the representation of physical phenomena via standardized symbolism.*

'Knowledge' on the other hand is some interpretation of an information set. This notion of information is distinct from that used in the field of information theory. Here we are dealing with measured properties of the physical world compared to a standard unit; in information theory the negative entropy value called information is basically a quantification of the expected reduction in uncertainty obtained from an observed signal [54].

In designing a measurement system we must first determine the properties of interest or 'referents'. In order to implement such a system, referents must be instantiated by 'measurands' or properties that can actually be measured by existing equipment. For example we cannot directly measure the referent 'quality', but we can measure properties of an object which determine our perception of quality.

Measuring instruments have various properties used to describe the measurements they make.

Definition 2.4 discrimination: *the magnitude of changes in a measured quantity which can be observed by the instrument*

Definition 2.5 repeatability: *the difference between measured values of the same world value in the short term*

Definition 2.6 reproducibility: *the difference between measured values of the same world value over the long term*

These three properties determine the **precision** of the measurement device, although precision is not generally assigned some overall numeric value.

Definition 2.7 accuracy: *is the difference between the true value of a measured quantity and the measurement returned by the measuring device.*

Accuracy of an instrument is determined by comparison of the instrument's measurements with declared reference values, or calibration.

Measurements can further be classified by whether they are direct, indirect or combined [52]. Direct measurements are read directly from measuring instruments. Indirect measurements assume the quantity to be measured is a function of several arguments which in turn can be measured. Combined and simultaneous measurements are similar in that they determine the quantity of interest by solving a system of equations where the coefficients and terms are measured values. Simultaneous measurements usually involve quantities of different kinds, and the equations solved reflect relationships in nature. Combined measurements involve quantities of the same kind, and equations based on arbitrary relationships between them.

2.2 Task Level Parameterizations

Tarantola [66] describes the investigation of physical systems as a cyclical process roughly divided into three steps: parameterization, prediction via the forward problem, and system identification via the inverse problem. Parameterization attempts to describe the system with a minimal set of model parameters. In studying the forward problem, the rules or physical laws which govern the system are used to predict values for observable parameters or measurements. For the inverse problem a collection of measurements are used to determine the true model parameters. If we view investigation of robotic tasks in a similar light, the search for minimal fundamental descriptions is a relatively new phenomenon [14, 22, 18]. In the past individual robotic systems for various tasks have been developed in a relatively *ad hoc* manner, and little effort has been made to compare and evaluate what underlies them.

For physical systems, the model parameterization M, is not necessarily unique; it merely places a coordinate frame on the abstract underlying model space $\mathcal{N}$. Any point

$\vec{m} = \{m_1, ..m_n\}$ in M represents an instantiation of the parameters, or a model. The set of observable parameters chosen is not unique either. We can imagine a space of all possible sensor readings for all possible instruments, where a particular choice of sensors and measurements defines a data space D of observable parameters, $\vec{d} = \{d_1, ..., d_m\}$. If we extend the physical system to include the sensors used we can speak of the physical parameter space $P = M \times D$. In this thesis we view robotic tasks as characterized by this parameter space.

Although elements of such a parameterization are not made explicit for most intelligent robotic systems, this is a general and potentially useful way of describing them. Depending on the level of detail we want to model, parameters could range from low level system values such as camera aperture to high level constructs such as object position. In particular however, we are interested in the task level parameters which determine system performance as suggested by the work on task level learning by Aboaf *et al.* [2].

In the case of sensing for robotic systems we seek to infer some task related aspect of world state based on our model and observed sensor measurements.

$$\begin{aligned} f &: \vec{p} \mapsto \vec{m} \\ \vec{p} &\in P, \vec{m} \in M \end{aligned} \tag{2.1}$$

Our observation is that the task level parameters used by most implemented systems are generally referred to as features. Parameterizations in the form of feature based representations vary widely and we have little sense of minimality or optimality achieved by existing systems. Generally however, the operation f involves a search through M for the best explanation of $\vec{d}$. A task then can be described as a trajectory through the state/model space inferred from feature observations. This perspective is not new with respect to control theory, but it is a useful way to think of abstracted task level systems.

In the case of our example from chapter 1, a set of image disparities I and the calibrated camera model C are used to determine the best estimated part position $\hat{p}$ to explain the observed disparity values. In other words for $C, I \in P$, determine the best $\hat{p} \in M$, where M is the set of possible part states.

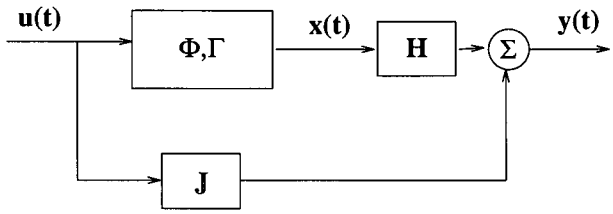


Figure 2.1: Basic dynamic system diagram.

2.3 State Spaces, Observability and Information for Tasks

When designing a robotic system to solve some task we have to decide what must be modelled, sensed or inferred to execute the task. Ideas for determining "what to sense" range from well known systems theory notions of observability of the state space, to high level planners which run wholly without reference to sensor observations. Obviously what the system can "see" limits what it can "do", and what tasks the system is required to perform should therefore drive what it is designed to perceive. In other words is the subspace generated by the agents perceptions adequate for the task?

The most precise notion of "what to sense" is that of observability from linear systems theory. It is defined for a system such as that illustrated in figure 2.1. In the discrete case the system is described by the equations:

$$x(n+1) = \Phi x(n) + \Gamma u(n) + \Gamma_1 w(n)$$
$$y(n) = Hx(n) + Ju$$

where $x(n)$ is the system state at time n, u is the control input and w is a disturbance. Φ is the system matrix, Γ is the control input matrix, Γ_1 is the plant noise matrix, H is the output matrix and J is the plant direct transmission matrix. The information required for control tasks is defined by the structure of the model which allows formal definitions for observability and reconstruction. The theory offers explicit tests to determine whether current system state is observable (and the dual property of whether the system is controllable), meaning it can be determined by measuring some future output.

Ignoring control and disturbances we can see that

$$\begin{aligned} y(0) &= Hx(0) \\ y(1) &= Hx(1) = H\Phi x(0) \\ y(2) &= Hx(2) = H\Phi x(1) = H\Phi^2 x(0) \\ &\vdots \\ y(n-1) &= H\Phi^{n-1}x(0) \end{aligned}$$

or

$$\begin{bmatrix} y(0) \\ \vdots \\ y(n-1) \end{bmatrix} = \begin{bmatrix} H \\ H\Phi \\ \vdots \\ H\Phi^{n-1} \end{bmatrix} x(0)$$

The matrix $O = \{H, H\Phi, ..., H\Phi^{n-1}\}$, is called the observability matrix. For the system to be observable, O must be nonsingular [25].

Reconstruction is the problem of predicting the state based on previous measurements [15]. Observability of system state captures the notion of being able to detect events relevant to the task, as well as referring them to the correct frame of reference as implied by identifiability of system state. We will use the term observability to suggest this idea of being able to identify critical system states from the measurements available.

Figure 2.2 illustrates the idea that no measurement system is infinitely accurate—some *set* of world points will be mapped to each measurement value. At a higher level, a mobile robot in an office environment may not be able to distinguish one office door, or corridor from another because they "look the same". These are sometimes referred to as perceptual equivalence classes [17], although generally real sensors will not induce a true partitioning on the space of world values. This idea of sets of world points indiscriminable to the system relates to the ideas of discrimination and accuracy in measurement. In general there is also a relationship between the dynamism of the task and environment and the complexity of the observable space required by the agent. This can be seen as a requirement for greater discrimination in time.

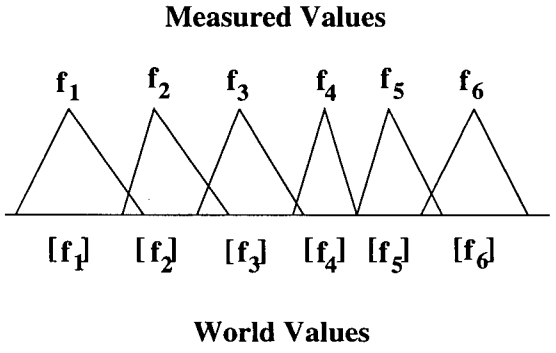


Figure 2.2: The process of measurement, and/or storing values digitally with finite precision, induces (possibly overlapping) indiscriminable sets $[f_i]$ of world points which are all mapped to the single measurement f_i.

A basic premise of this thesis is that information for robotic tasks has three fundamental characteristics: we must determine what to sense, or the observable parameters; and we must achieve a specific level of precision and a certain level of accuracy to achieve the desired level of system performance.

2.3.1 Task Based Perception

The issue of what to sense has become an interesting counterpoint to previous work in computational sensing which attempted to reconstruct the world from sensor values. As more scientists work with robots which have to act in a dynamic environment, the idea of task directed sensing as a carefully allocated resource becomes critical. This in turn brings us back to issues of minimality and optimality for robotic systems.

There are three key aspects to sensing for action: 1) determining what information or measurement is required for the robotic system to progress; 2) detectability— can your sensor get the required information at the required precision and required timeliness?; and 3) calibration— or reconciliation of reference frames for sensors and actuators. The first of these three relates to the notion of parameterizing the task model space, since deciding what-to-sense essentially determines what task related aspects of the world must be inferred to perform a task. Detectability affects the data space parameterization, and calibration establishes rules for f of eq. 2.1, which relate observations to world state.

The most common and successful integrated sensing and action systems are of course control systems. Unfortunately these successes are typically limited to finite, linear (or linearized) systems. For complex tasks it would be difficult or impossible to derive a useful dynamical model to which control theory might be applied. It is interesting to note that in control theory the system and environment are treated as a whole in the form of a single model composed of differential equations. Observability of system state indicates referring events to the correct frame of reference as implied by identifiability of system state.

Reactive systems have embraced the control model with the claim that it will be sufficient to combine simple control loops in complex environments, to achieve complex behaviour [9]. Current wisdom however suggests that more complex systems will require more retained state and higher level prediction and planning. When such representations are added to the reactive agent however, the notion of integrated sensing and action becomes less well defined. In early reactive systems it was not clear what drove selection of sensory processes, or how they were calibrated.

Firby points out that sensing is a scarce resource in robotic systems and should be applied judiciously to relevant aspects of the task at hand [24]. He notes that if you use sensing at run time to distinguish among several world states, you will have to have an alternative plan for each possible state, creating a combinatorial explosion. He sees actuators as acting on the world and sensors as acting on the world model. He does not offer specific rules for what task related observations may be pertinent; these decisions are left to the planner. He introduces the notion that sensed values become obsolete with time. Essentially his planner senses to fill in values of which it is not certain, in order to select which plans to apply.

Bajcsy's work on active perception [5] also pointed out the need to direct one's sensing effort toward a particular perceptual goal. Generally, active perception is the study of modelling and control for perception. Specifically it emphasizes intelligent data acquisition using feedback of errors and parameters from the current scene or percept. It requires a good understanding of the sensory devices, algorithms and the goal of perception.

Recently the idea of the virtual sensor has become popular. A virtual sensor describes what we would like to be able to observe during task execution, rather than the real sensor readings available [34, 17] and thus answers the question of what to sense. Virtual sensor readings are constructed from the available real sensor values. This idea has been characterized in more detail in Erdmann's action-based sensor design [21, 22]. Erdmann recommends determining what abstract information is required for the task, then implementing a physical sensor to provide it. His method for selecting the necessary abstract sensors, is to first design an action sequence to achieve the task, then define a progress measure for distance to completion. Progress regions in the state space are then defined as regions where particular actions make progress. The abstract sensor then observes the progress region containing the state of the system and hence recognizes the associated action which will make the system progress toward the goal. "Minimal information" is associated with the progress regions of the action plans which reach the goal quickest.

Sensor planning is the job of automatically determining sensor parameters given knowledge of the sensor, environment and the task at hand. The parameters in question can include internal instrument values as well as position and orientation, and even external properties such as the position and brightness of light sources. In most cases it deals with the question of designing sensors for detectability of the required task information. Tarabanis et al [65] describe the numerous constraints which must be considered to satisfy tasks in vision sensor planning. They describe three main approaches including those for recognition and localization of objects, scene reconstruction, and detection of specific object features. Model based recognition systems often use hypothesize-and-test techniques to search the discretized space of possible objects or poses, collecting more data on the scene as new sensor configurations are proposed and tried, until a stopping criterion is satisfied.

In scene reconstruction the scene model is built incrementally and sensor configurations are chosen according to criteria intended to fill in unknown parts of the scene until it is fully explored. One such scene reconstruction system is Whaite and Ferrie's autonomous exploration method [73]. It actively looks for viewpoints from which obser-

vations will maximize the accuracy of its internal model. The method seeks to improve its representation by making additional observations. It uses a gaze planning strategy which selects observations to reduce model uncertainty, in other words to improve regions where predictions are least reliable. This is a promising method for exploring an environment, but for more specific tasks what are the key uncertainties which must be reduced by sensing? Sensor planning for scene reconstruction seems mainly to deal with the problem of what to sense for a task requiring a dense world model, e.g. obstacle avoidance may require the most information along a planned path.

The sensor planning methods used for detecting features of objects deal mainly with designing sensors that provide the detectability required by the task. They require that the features be visible, in focus and projected at a given size in the image. Generally sensor configurations can be planned offline since objects are often assumed to arrive in known position. Various approaches are possible including generate-and-test search in a discretized space of sensor configurations, or synthesis methods where task constraints are solved analytically for valid parameter values. The large number of parameters necessary to fix optical and geometric configurations of sensors and illuminators, can cause these search problems to become huge, requiring various assumptions to make them viable.

The notion of planning in recognizable sets rather than reachable sets is an important one for task directed sensing. Obviously if our knowledge of the world comes from our sensors, our ability to plan in areas we cannot recognize or distinguish may be useless. This idea is echoed in the ideas of Perceptual Equivalence Classes which lump together parts of the world which "look the same" i.e. cannot be distinguished by the sensors available [17, 39] . These theoretical ideas are demonstrated by Mataric's robot which learns its world representation by accumulating sensor readings classed as right walls, left walls, corridors and irregular [43]. These features are used to construct landmarks in a graph representing the environment. Once constructed, the graph can be used to navigate in and recognize mapped regions of the world. Planning in recognizable sets versus the virtual sensor method offer dual approaches to task directed sensing, where one designs plans based on sensor driven representations, and the other designs sensors

based on plan requirements. This reflects the tradeoff between detectability and what-to-sense since the ability of our sensors limits what we sense. This tradeoff has yet to be studied or fully understood.

Landmark-based navigation is an interesting demonstration of task directed sensing, where relatively sparse "beacons" distinguish regions of the workspace allowing the robot to localize itself in some internal map. Lazanas and Latombe formalize robot motion planning with landmarks and propose an algorithm which is complete and of polynomial order [35]. They define landmarks as one or more physical features of the workspace which the robot can identify with perfect accuracy within some limited field of influence. This allows perfect control and sensing in the neighbourhood of landmarks. The method's obstacle-free environment and perfect landmark identification are strong assumptions. They make the interesting claim that engineering the environment by constructing landmarks allows tractability in planning. It is unclear what has been traded for planning tractability however, since we have no notion of the complexity of engineering or calibration of the environment. Evidently some of the planning and representational complexity has been encoded in the environment.

Information Invariants

Donald's work on information invariants attempts to formally address information for robotic tasks and touches on a number of issues in robotic task complexity [18]. Although still somewhat speculative, his goal is to provide a system for comparison of sensori-computational systems, transformation of one system into another, determining whether a system specification can be satisfied by a particular sensor system, and formal reduction of one system to another. He proposes that a measure of information invariants may imply a measure of hardness or difficulty and hence a notion of lower bounds in robotics.

He begins by discussing a manipulation task by two cooperating robots and describes an instance of communication being traded off with sensing. This rather anecdotal comparison prompts him to assign the sensor operations a value equal to the number of bits communicated. Similarly, following tasks require modifications in the path of the leader,

when the follower uses only sensory information versus communication. The added travel distance can also be compared to bits communicated. Comparing different parameterizations of the same task to examine fundamental task information is part of the motivation for the comparison framework presented in this thesis.

In Donald's framework sensor systems are represented by graphs with components such as transmitters and receivers forming the vertex set, and connections among them as edges. Sensors are said to be *situated* if their vertices are embedded by a mapping into a configuration space. The fixing of external state or calibration, is an important issue as seen in planning with landmarks. Donald attempts to quantify *calibration complexity* by the number of components of a sensor system whose relative positions are fixed. He proposes ordering sensor systems on the amount of information gleaned from calibration or external state.

The transformations and notation Donald uses are quite formal, but it is unclear that the embedding of a sensor graph in an environment is an adequate description of the power or information it represents. Calibration is certainty important, but there are more aspects to sensors than the relative location of components.

2.3.2 Features and Information

The term "feature" is pervasive in many areas of science and particularly in sensing and robotic manipulation, but it takes on somewhat different meanings in its various uses. Features always summarize and encode key physical properties for the problem at hand. Our premise is that features, as they exist in the literature, are measurements or combinations of measurements acting as task level parameters.

In biological systems we read of Treisman's preattentive features which are recorded preattentively in parallel and thus pop-out more noticeably than others [69]. Lederman and Klatzky [36] argue that touch (haptics) can be effectively used for perception and recognition, in the form of purposive hand movements. The notion of perception as part of an action system is echoed in work on 'perception-action systems' hypothesized to be holdovers of evolution. Jeannerod proposed visuomotor channels where a visual

stimulus activates a motor program [30]. In multimovement systems the value of sensory information varies with the motor task being performed and the sensory and motor modalities in use [1]. All of these suggest a close relationship between percept and action. Clearly the features extracted are those which have meaningful covariance with the need to act [38], and hence reflect the task-based parameterization we suggest.

In computational sensing features have more often been chosen to meet some intermediate goal of a general purpose sensor system than the outcome of an overall robotic system, although of course these are not exclusive. Early in the history of computational vision, pattern recognition classified images using a number of measurements made on the image or event, this vector of measurements or features defined a point in a *feature space.* Image patterns belonging to the same class were assumed to cluster in the same region of feature space, while distinct classes would be separated [45]. In the case of locating 3-D objects in 2-D images simple measurements tend to be dependent on lighting and on the object's pose, so classification without reconstructing object properties will not work well [29].

Selecting or constructing suitable features on which to perform classification is the key to pattern recognition. Features generally provide a reduction in dimensionality of the problem, and if they are well chosen the information which is discarded is irrelevant to the problem to be solved. Principal component analysis was used to transform the bases of the measurements into a space where combinations of features provided better discrimination among classes. Recently this notion has resurfaced in *modal* (as in mechanical modes of a system) correspondence and recognition techniques. Turk's eigenfaces [71] treat whole $N \times N$ images of human faces as huge vectors of length N^2 and basically use principal component analysis to transform images into points in a feature space. Another modal approach is Shapiro and Brady's feature correspondence method [59]. They determine correspondences between a pair of images while retaining a shape metric by computing the eigenvectors of symmetric matrices H_i, which contain the Gaussian weighted distances between features within each image i. Matrices of the eigenvectors for each image are compared to determine which features correspond.

Recently the notion of selecting features which act as indices into a set of models or hypothesized image events has gained popularity [62]. Ideally each feature would generate one (correct) hypothesis as to the contents of the image, much in the way feature vectors were meant to classify images. In our parameterization model this would make $f(\vec{p})$ an indexing operation which would give us a (hopefully small) set of possible task states $\vec{m}_i$.

From a more theoretical point of view Richards and Jepson present definitions for robust visual features which include both likelihood and context in robust statistical inference from images [53]. They base their arguments on the idea that good features reflect non-accidental configurations [37], which are typical of the world and yet informative, again preferring features which generate relatively few but likely hypotheses about the world state.

Feddema *et al.* describe selection criteria for image features to be used directly as the feedback signal in a resolved motion rate control scheme which allows an eye-in-hand robot to track a workpiece as it moves [23]. There are two aspects to feature selection: image recognition and control, so a set of quantitative measures of various extraction and control issues is proposed. Feature extraction measures include: uniqueness, noise resistance, expense and feature set completeness. Control measures include: observability, controllability and control's sensitivity to noise.

A similar system is described by Skaar *et al.* based on the premise that some tasks can be achieved by producing particular relationships among visual cues in camera space, thereby effecting the task in physical space [60]. The goal is to learn *camera space kinematics* such that a manipulator can be driven just as if real physical space kinematics were known. To this end the relationship between joint angles and visual cues must be established, methods of predicting the motion of cues in the image, and methods for driving the cues to required configurations are constructed.

Robotics literature has its own history of features, which includes some sensory events which index world state as with Eberman's contact features [20]. Contact features are essentially peaks in a temporal sequence of force measurements representing impacts with the environment. Given a model of our motion and where we expect to be, likely world

positions for the impact can be determined. Systems employing force sensors within control loops to generate passive behaviour such as compliance, can still be referred to as sensorless.

Kinematic features also play an important role in fine motion activities such as pushing [41]. Kinematic pairs define how objects can move relative to each other. The motion of a pushed object is also determined by the coefficient of friction between the object and the pusher, and the centre of mass of the object. All of these values parameterize the state transitions in the pushing system, and hence allow us to infer world state after a push is executed. Pushing operations underly much of the work on sensorless manipulation [27, 51].

Brost [11] and Caine [13] use a construct called a "contact superset" to enumerate all possible kinematic interactions between a pair of polygons in contact. Caine uses the contact superset with friction and support relations to design features for vibratory bowl feeders. These features are essentially shapes installed in the side of the bowl which cause correctly oriented parts to pass and incorrectly oriented parts to fall back into the bin.

What does the work described above tell us about features, and hence parameterizations, for robotic tasks? We know that in biological systems as well as many robotic systems the information and representations used depend on the task to be performed, whether that is navigation, manipulation or survival in a hostile environment. The features we require reduce the volume of information to be processed, while capturing something fundamental to acting in the environment. For the robot programmer we need to determine what features are useful for which types of tasks, and potentially what are the minimal set of features and minimal precision of measurement to consider to achieve this task.

2.4 Evaluating Task Performance

2.4.1 Complexity of Robotic Tasks

A number of researchers have made an effort to analyse the quintessential aspects of robotic tasks. From a computational complexity point of view several aspects of robotics

tasks have been explored. Natarajan and others [48, 61] have attempted to measure the complexity of assembly tasks and workpieces in *hands* or independent motions. Navigational issues are addressed in the context of graph searching, by determining which augmentations to standard automata such as pebbles, counters or additional automata, allow them to search mazes [6]. Others have studied the complexity of configuration space fine motion planning. Brafman *et al.* [8] have studied sound and complete termination conditions with uncertainty in control and sensing.

Andeen [3] proposes a "Science of Assembly" which measures the difficulty, complexity, uncertainty and variance of the task. Complexity is a generalized displacement determined by the number of robot primitives required to perform a task. Difficulty entails the weight and size of the parts and the forces required to mate them, and is measured by the time required to perform the task.

As previously described, for the job of designing and describing information sources, recent work by Donald [18] seeks to study *information invariants* or information necessary to perform a task, as a measure of task difficulty. He develops methods to compare and reduce one sensor system to another, when both generate the necessary information. Mackworth [40] proposes a set of *adequacy criteria for visual knowledge representation* which outlines some important requirements for sensor based information sources. There have also recently been calls to implement robotic tasks with only simple sensors and actuators rather than the vision systems and dexterous hands of the past [14], although the method for distributing complex tasks over simple sensor and actuator components appears to be lacking.

In a 1984 paper Sanderson proposed an information theoretic measure for assembly tasks he called *parts entropy* [57]. The measure is based on uncertainty in part position and orientation and reflects the view that assembly is the convergence of parts as they become more and more precisely oriented until they reach mating tolerances. The basic premise is that as sensing and manipulation actions are applied to the parts of an assembly, the set of possible states for each part is reduced. For some probability distribution on states, the entropy of the parts is correspondingly reduced. For assembly problems

involving several parts the joint entropy of the set of parts is used. The goal of the task is assumed to be to decrease the total entropy for the set. Sanderson proposes that the changing entropy for each step of the task be plotted against time to compare the entropy reduction of various task strategies. This idea, although neglected for some time has recently gained popularity [14], however part entropy speaks only to the precision of knowledge about a part's state. It does not relate information to success in achieving task goals.

2.4.2 Part Orienting

Given Donald's premise that we can reduce or compare resources of one task implementation to another to study information invariants, we will examine and compare successful implementations of a particular task. The task we have chosen is the part orienting task popular in the literature on sensorless manipulation. Logically, if, as claimed, this work has successfully moved task information entirely to the domain of models [42], it may give us insight into the information actually in use. A part can be oriented by a number of different devices. Perhaps because of the low dimensionality (resolving only the angle of the part) and hence limited state space of the task, part orienting has become the focus of considerable study.

Goldberg and Mason [27] describe what they call Bayesian Grasping, using a frictionless gripper. They develop a diameter function for each 2-D object, which plots the diameter of the object against its orientation relative to the gripper. When squeezed from any initial orientation the object will tend to rotate to a local minimum in its diameter function. The object orientations consistent with these minima will have some predictable probability after a squeeze has been executed. Obviously these squeeze operations reduce the uncertainty regarding object orientation, and a sequence of oriented squeezes can be used to place the object in a desired orientation.

Sequences of squeezes are computed by starting with an initial grasp for an unknown initial orientation, and searching a tree of possible sequences from the set of possible actions, until sufficient probability for some orientation is reached. The authors propose

ranking various plans based on cost functions such as distance from goal or time (steps) required.

Brost [10] also studied the squeeze grasp but addressed the stability of push operations over the operation space as well as including uncertainties in object position, friction and gripper orientation. He developed a sequence of "maps" indicating the behaviour of the part in various regions of the parameter space.

Another part orienting strategy formalized by Peshkin and Sanderson [51] is that of parts on a conveyor belt pushed past a series of specially designed fences of various orientations and lengths (see Chapter 1, figure 1.2). The passive fences with moving conveyor act in a manner similar to an active pusher operating on a statically supported object. As parts pass the last fence they lie in a single desired orientation.

A common industrial part manipulator is the vibratory bowl feeder. The bowl feeder contains a collection of randomly oriented parts which as a result of the vibration, move up a track on the interior of the bowl. The physical properties of the parts cause them to be oriented in one of a number of stable resting positions. At the top of the track parts encounter a series of "features" in the bowl wall and track which cause them to be either reoriented or rejected and dropped back into the bowl [13].

The interactions of these features with parts has been formalized in configuration space by Caine [13] in the form of *contact supersets*, which capture all interactions between the corner and surface features of the part and the bowl. By superimposing support and friction information on these configuration space constructs, Caine can determine bowl features which allow certain part orientations to pass by, while others are rejected.

Another, more general, manipulation system which is capable of part orienting is a robot arm with one or more cameras. A part is localized by the vision system so that the arm can directly orient it using simple parameterized pick and place plan.

Obviously the task goal is the same for each of these proposed implementations, but the nature of the sensing, models and manipulation differ significantly. Clearly they all entail physical or sensory localization of the part, and some degree of manipulation.

Donald's work suggests we can learn something fundamental about the task itself through comparing implementations. His reductions, however, tend to apply to sensors that return only a few bits of information. We need methods to make meaningful comparisons of disparate systems with potentially rich sensor information.

2.5 Summary

Information for robotic tasks takes the form of measurements stored as models or made online. Tasks are parameterized by the information or measurements available to the system. The three basic characteristics of information for robotic tasks are: which parameters must be observed, how precisely they must be measured, and how accurately they must be measured. This idea has been demonstrated by systems from control theory through sensor planning and task planning in the recognizable sets. Researchers have also attempted to capture the effects of measurement precision with descriptions of perceptual equivalence classes. In this chapter we have described the wide variety of features or task level measurements used in biology, sensing and robotics, and efforts by authors to characterize good or task related features.

We have described a few of the formal methods for describing robotic tasks including information invariants and parts entropy. None seems precisely suited to formalizing task information in the paradigm we have laid out in this chapter. One way to look for an appropriate model is to experiment with implemented tasks and find methods to explicate their information requirements. In the next chapter we will present such an empirical framework for discerning task information. The implemented task is the part orienting task we have described. The advantage of this simple manipulation task is that it can be implemented in very different ways. This gives us the opportunity to compare our evaluations of the information used in different implementations and thus examine the information required by the underlying or ideal task.

Chapter 3

Evaluating Task Parameterizations

Any task implementation is merely an approximation, since we do not know the ideal underlying task function, and because we construct implementations based on values sampled from the physical world. We want to identify the information space of a robotic task implementation in order to optimize the resources we devote to measurement and modelling. To achieve this we propose a framework where we first discover which parameters are statistically significant to task performance, and then determine the precision and accuracy required in those parameters to minimize cost while achieving task performance goals.

The view of robotic systems we have argued for in previous chapters is one where task implementations are parameterized by measurements. These measurements are either performed *a priori* and stored in the form of world or object models, or acquired as the task progresses. In what follows we will include other parameters which affect measurement quality, such as image window size in our stereo camera implementation, since it is the task information which interests us. They are parameters of the measurement process rather than measurements *per se*. Figure 3.1 depicts this general view of

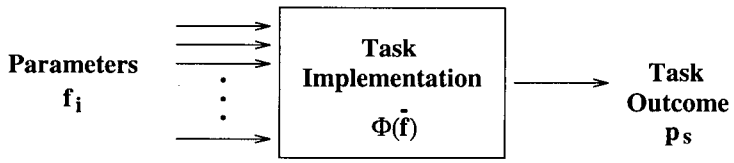


Figure 3.1: The details of the many components of a task implementation may be difficult and time consuming to analyse. We treat each task implementation as a black box into which we feed a number of measurements and other parameters to generate the performance level, or task outcome.

a complex task implementation treated as a black box driven by input parameters to produce task outcomes. This looks like a system identification problem but we are less interested in the details of the internal workings of the system than in how the quality of the parameters affect its performance. From an empiricist's point of view we have a number of tools to investigate such systems and it is just such an experimental exploration which we describe here.

The task outcomes we will use are actually binomial probabilities of success, estimated in the usual way, given an evaluation function S returning 1 for success and 0 for failure, for the system $\Phi(\vec{f})$. The probability of success ($\hat{p}_s$) over n trials is estimated as follows:

$$\hat{p}_s = \frac{\sum_i^n S(\Phi(\vec{f}), i)}{n}, \tag{3.1}$$

We will also, in some cases, use a 'distance to goal' value where it is illustrative of how the system is failing the task requirements. In many of our experiments we are also concerned with the standard deviation of our outcome, $\sigma_{\hat{p}}$, which is known to be [44, p.297]:

$$\sigma_{\hat{p}}^2 = \frac{pq}{n} = \frac{p(1-p)}{n},$$

(for the binomial distribution $q = 1 - p$). In general we will not know the true value of p, and so will estimate $\sigma_{\hat{p}}$ using

$$s_{\hat{p}}^2 = \frac{\hat{p}_s(1-\hat{p}_s)}{n}. \tag{3.2}$$

We have defined the three critical aspects of task information: observability, precision and calibration/accuracy. In the following sections we will introduce a framework combining various techniques to analyse and determine how these properties affect task performance, and therefore how we can tune our measurement processes to improve task performance.

3.1 Identifying the Task Subspace

When we refer to observability we mean the ability to observe or infer the system states of interest from available measurements. Of course we would like to find a minimal parameter set for any system, but without knowing the system model we cannot directly

determine the minimal parameterization. In fact intelligent robotic tasks are not often subjected to analyses aimed at minimization or optimization for a particular implementation. We must begin with some empirical exploration of the response of the system to variations in its parameters. A number of methods are presented in the following sections including factorial experiments, sensitivity analysis and ε-complexity . In the experiments in chapter 6 we will demonstrate the value of each of these approaches. As we will see all of these methods perform a type of sensitivity analysis, varying parameters under differing criteria and evaluating performance in a number of ways. Later in this chapter we propose an evaluation process, which is a hybrid of these methods and which is specifically designed for our stated goals.

3.1.1 Design of Experiments

Statistical methods for the design of experiments are intended to counteract variability in observations and reduce experimental error. Generally we have one or more 'factors', or independently controlled experimental variables, which can be set to various intensities or levels. Factors may be 'qualitative' such as different edgel grouping methods, or 'quantitative', such as the value of σ used by a particular edge detector. A particular combination of factor levels is referred to as a 'treatment'. The outcomes of the system under study are called 'responses' or observations [44].

Experiments can be structured in various ways determined by the selection of the objects which are to be measured under various experimental conditions or 'experimental units', and the way treatments are applied to them. A completely randomized design involves selecting independent random samples for comparison, from a set of p populations to which differing treatments have been applied. Noise created by nuisance variables not under our control can be nullified by making the experimental units as homogeneous as possible. Randomized block designs for experiments with p treatments, consist of b blocks each containing p experimental units to which the treatments are applied in a randomly permuted order. Other block designs include the Latin Square which allows blocking in two directions by requiring a $p \times p$ array of experimental units. Each row (or column)

has a distinct permutation of treatments from any other row (or column).

The general formulation for such experiments for p treatments and b blocks is

$$\begin{aligned} Y_{ij} &= \mu + \tau_i + \beta_j + e_{ij}, \\ i &= 1, 2, ...t,\ j = 1, 2, ...b. \end{aligned} \tag{3.3}$$

In other words the observation Y_{ij} is composed of its underlying mean μ plus the treatment effects τ_i plus the blocking effects β_j plus a normally distributed random error e_{ij}. The null hypothesis for such experiments is that all τ_i are zero and all β_j are zero, implying that the mean for all treatment populations is the same. We can then use analysis of variance techniques to accept or reject these hypotheses [56].

Analysis of Variance

The experimental designs we will use depend on the fundamental methods of analysis of variance (ANOVA). These are used to determine which are the important factors, and how they affect the system outcomes and interact with each other. The variance of a sample is the average of the squared deviations of the individual measurements about the sample mean $\bar{y}$ or

$$s^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^2$$

for the sample $y_1, y_2, ..., y_n$. Analysis of variance divides the total sum of squared deviations $Total\ SS$ into partitions attributed to the sum of squares for blocks (SSB), treatments (SST) and error (SSE).

$$\begin{aligned} Total\ SS &= SSB + SST + SSE \\ &= \textstyle\sum_{i=1}^{b}\sum_{j=1}^{p}(y_{ij} - \overline{y})^2 = \sum_{i=1}^{b}\sum_{j=1}^{p} y_{ij}^2 - CF \end{aligned}$$

$$CF = \frac{(\text{total of all observations})^2}{n} = \frac{\left(\sum_{i=1}^{b}\sum_{j=1}^{p}\sum_{k=1}^{n_{ij}} y_{ij}\right)^2}{n} = n\overline{y}^2, n = bp$$

$$SSB = p\textstyle\sum_{i=1}^{b}(\overline{B}_i - \overline{y})^2 = \frac{\sum_{i=1}^{b}\overline{B}_i^2}{p} - CF$$

$$SST = b\textstyle\sum_{j=1}^{p}(\overline{T}_j - \overline{y})^2 = \frac{\sum_{j=1}^{p}\overline{T}_j^2}{b} - CF$$

$$SSE = Total\ SS - SSB - SST$$

where CF is the correction factor for the mean, B_i and $\overline{B}_i$ are the total and average for block i, and T_j and $\overline{T}_j$ are the total and average for treatment j. If we assume, as above, that blocking and treatments have no effect on the response, then each part can be divided by an appropriate constant to generate an unbiased estimator s_e^2 of the variance of e the experimental error. These 'sums of squares' (SS) divided by their degrees of freedom (see table 3.1 for usual degrees of freedom (dof)) are referred to as 'mean squares' (MS), and are the estimates of σ_e^2, for the null hypothesis that the means of all block or all treatment populations are the same.

$$\begin{aligned} s_e^2 = MSE &= \tfrac{SSE}{n-b-p+1} \\ MST &= \tfrac{SST}{p-1} \\ MSB &= \tfrac{SSB}{b-1} \end{aligned} \tag{3.4}$$

We can think of this two way classification into blocks and treatments as a two factor experiment.

The F test is normally used to compare variances S_1 and S_2, where the F value is defined as

$$F = \frac{S_1^2}{S_2^2}.$$

Since MST and MSB are only estimators of σ_e^2, given that all blocks or treatments have the same mean, these estimators will grow large if this is not the case. Thus we can test whether the treatment or block means are all equal using the F test to reject unreasonably large values for type I error probability α. F_α values can be selected from standard tables according to the desired α and the numerator and denominator degrees of freedom.

$$F = \tfrac{MST}{MSE}, \quad H_0 : \mu_1 = \mu_1 = ... = \mu_p, \text{for } p \text{ treatments}$$

$$F = \tfrac{MSB}{MSE}, \quad H_0 : \mu_1 = \mu_1 = ... = \mu_b, \text{for } b \text{ blocks}$$

$$\text{reject } H_0 \text{ when } F > F_\alpha$$

Note that the F test with 1 dof in the numerator and n dof in the denominator is equivalent to the t test at n dof, although $F_\alpha = t_\alpha^2$ because of the structure of the test statistics.

Source	dof	Sum of Squares	Mean Squares	F
Blocks	$b-1$	SSB	$\frac{SSB}{b-1}$	$\frac{MSB}{MSE}$
Treatments	$p-1$	SST	$\frac{SST}{p-1}$	$\frac{MST}{MSE}$
Error	$n-b-p+1$	SSE	MSE	
Total	$n-1$	Total SS		

Table 3.1: Standard analysis of variance table.

ANOVA data are generally tabulated in the form illustrated in table 3.1. These methods are well known and thus their description here is only cursory. For more detailed explanations the reader is directed to any introductory Statistics text [44].

Factorial Experiments

Factorial experiments apply treatments in a structured manner to extract as much information about the effects and interactions of factors as possible. In the general case, for a complete factorial design we select some number of levels l_i for each of k factors and test all possible combinations of all levels of the various factors. Such an arrangement is called an $l_1 \times l_2 \times ... \times l_k$ factorial design. If all $l_1 = l_2 = ... = l_k = l$ then the design is referred to as an l^k symmetrical factorial experiment. Commonly 2^k experiments are used, where each of k factors are tested at two levels; these are the most easily explained and we shall describe them in what follows [7, 63].

We are interested in testing the effect of each factor individually, or its main effect, and the interactions among factors, or interaction effects. In 2^k experiments each of the $2^k - 1$ effects has one degree of freedom. These effects are associated with mutually orthogonal contrasts among the treatment responses. A 'contrast' is a combination of treatment yields, where the coefficients are ± 1 and sum to zero. The main effect of a factor is computed from a set of observed responses as follows

$$\text{Main Effect} = \begin{pmatrix}\text{average effect}\\ \text{while factor}\\ \text{is high}\end{pmatrix} - \begin{pmatrix}\text{average effect}\\ \text{while factor}\\ \text{is low}\end{pmatrix}$$

Various indexing schemes for determining how to combine factors to compute the desired contrasts exist. The usual method is to associate each treatment combination

response	1	c	b	bc	a	ac	ab	abc
A	-1	-1	-1	-1	1	1	1	1
B	-1	-1	1	1	-1	-1	1	1
C	-1	1	-1	1	-1	1	-1	1
AB	1	1	-1	-1	-1	-1	1	1
AC	1	-1	1	-1	-1	1	-1	1
BC	1	-1	-1	1	1	-1	-1	1
ABC	-1	1	1	-1	1	-1	-1	1

Table 3.2: Coefficients for computing effects for 2^3 factorial experiment.

with a binary string of length k where a zero in the ith position implies the ith factor is at its low level in the current treatment and a one implies it is high. The main effect for any factor i then, will be the average of all responses with a one in the ith index position minus the average of those with a zero in that position. For a 2^k experiment we have the set of trial indices formed by the k digit binary numbers from 0 to $2^k - 1$. All observations are used to compute all effects. We can tabulate ± 1 coefficients for these responses according to how they are combined to compute main and interaction effects. Coefficients for computing interaction effects can be generated by elementwise multiplication of the coefficients for the main effects of the factors involved in the interaction [16]. For example for a 2^3 experiment for factors A, B and C, we can construct the table of coefficients shown in table 3.2. The treatment combinations are frequently indicated by sequences such as a, ab, bc etc, where the lower case factor label appears only when the treatment includes the high level of that factor. If the factor is at its low level its label is absent. The 000 case where all factors are low is labelled 1. Upper case letters are used to label the main and interaction effects estimated from these experimental responses. Using table 3.2 we can compute the main effect for A

$$A = \frac{1}{4}(a + ac + ab + abc - bc - b - c - 1),$$

or for the interaction effect BC

$$BC = \frac{1}{4}(1 - c - b + bc + a - ac - ab + abc)$$

The expressions are multiplied by $\frac{1}{4}$ because the average while high (terms with positive

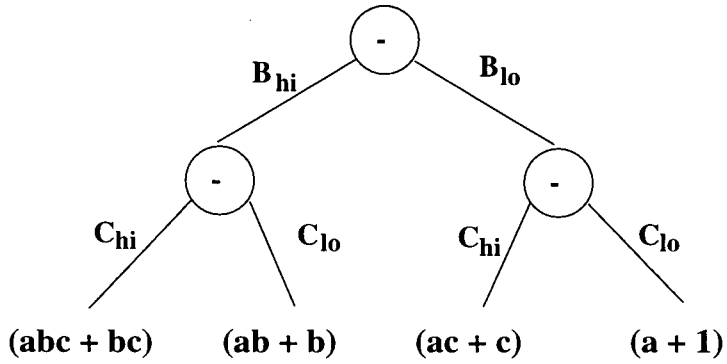


Figure 3.2: The interaction effect BC is found by computing the effect of C ($C_{hi} - C_{lo}$) while B is high, minus the effect of C while B is low.

coefficients) and average while low (terms with negative coefficients) each contain four terms.

For the interaction effect we can intuitively think of the effect of C while B is high minus the effect of C while B is low as illustrated in the tree structure in figure 3.2. Carrying through the signs at the nodes we can see the expression is the same as that above.

One mechanical way of generating the expression for the combination of observations for an effect X is to expand:

$$X = \frac{1}{2^{k-1}}(a \pm 1)(b \pm 1)(c \pm 1)...$$

where a '$-$' appears in a term if that factor is participating in the effect X and a '$+$' appears otherwise. Thus:

$$BC = \frac{1}{2^2}(a+1)(b-1)(c-1) = \frac{1}{4}(abc + bc - ac - c - ab - b + a + 1).$$

Again the expression is the same.

As with previously described experiments, we begin with the null hypothesis that the various factors and interactions between them have no effect on observed responses. Referring to equation 3.3, the treatments τ_i are the various treatments proscribed by the factor levels, for example A_{hi} versus A_{lo} is one partition of the observed samples. The sums of squares for the contrasts used to estimate effects, have a χ^2 distribution with 1 *dof*

(degrees of freedom are equal to one less than the number of factor levels, i.e. equal to the number of comparisons available) for the 2^k case, we therefore can construct an ANOVA table for our complete factorial example. Using the methods described previously we can determine whether our experiments provide sufficient evidence to reject the null hypothesis.

Instead of just two factors, blocks and treatments, we now have all the factors and interactions of interest resulting from our applied treatment conditions [56, 55]. Every 'effect' described above is actually the difference between average observations for two treatment levels spanning the entire set of experimental observations. Just as blocks and treatments are different groupings of the same samples, the observations for the various treatment levels are combined differently for each factor and interaction, according to the coefficients described above. As we saw in equation 3.4, $MST = \frac{SST}{t}$, where t is the number of degrees of freedom for treatment, thus in table 3.3 each effect has $MST = \frac{SST}{1} = SST$. Naturally, for larger experiments involving more factors we may not wish to consider all the factors and interactions in our ANOVA, particularly when we see they have low significance with respect to the F test. To 'pool' a factor we simply combine its sum of squares with SSE, adding its degree of freedom to the error degrees of freedom. Thus to pool a factor A, we update $MSE = (SSE + SSA)/(e_{dof} + a_{dof})$; of course we must update our F values accordingly since each has MSE as its denominator. In this way we can identify which parameters, and interactions among parameters, make significant contributions to system outcomes.

Fractional factorial experiments take advantage of the redundancy available in the set of observations to allow the experimenter to make fewer runs in the initial stages of his investigation. This is particularly important when the number of factors tested is large and hence the number of trials required by a complete design is huge. In this technique a subset of the factor/level combinations is used. Generally the selection of the subset is done such that higher order interactions are confounded or aliased to main effects or low order interactions, under the assumption that interactions among many factors are less likely to be significant [7, 16]. For example if we have 5 factors (A, B, C, D, E) the

Source		dof	Sum of Squares	Mean Squares	F
Blocks					
Replicates	r	r	SSR	$\frac{SSR}{r}$	$\frac{MSR}{MSE}$
Treatments					
A	1		SS_A		$\frac{SS_A}{MSE}$
B	1		SS_B		$\frac{SS_B}{MSE}$
C	1		SS_C		$\frac{SS_C}{MSE}$
AB	1	$2^3 - 1 = 7$	SS_{AB}		$\frac{SS_{AB}}{MSE}$
AC	1		SS_{AC}		$\frac{SS_{AC}}{MSE}$
BC	1		SS_{BC}		$\frac{SS_{BC}}{MSE}$
ABC	1		SS_{ABC}		$\frac{SS_{ABC}}{MSE}$
Error		$r(2^3 - 1) = 7r$	SSE	MSE	
Total		$r2^3 - 1 = 8r - 1$			

Table 3.3: An example of the ANOVA table constructed to analyse factor effects in a factorial experiment.

total combination effect $ABCDE$ is not likely to be significant compared to individual factors or smaller combinations. The interactions which are confounded in determining the fraction of trials, are referred to as the identity group of interactions. The resolution of a fractional factorial design is the number R of factors involved in the smallest interaction in this identity group [16].

For example for factors $ABCDE$ we can select an identity relationship

$$I = -ABC = -ADE = +BCDE$$

to generate a $\frac{2^3}{2^5} = \frac{1}{4}$ fractional factorial experiment. We can extend the idea of multiplying the signed coefficients of main effects to get interaction effects, to multiplying interactions (substituting 1 for any squared value) to determine confounding effects. For example in the identity above, the set of treatments selected for the experiment are those occurring with negative coefficients in the effect computation ABC and negative coefficients in ADE. Multiplying these two we get the positive coefficients from the interaction $BCDE$. Given this particular $\frac{1}{4}$ of the full set of treatments, any effect or interaction X cannot be distinguished from its generalized interaction (multiplication) with any member of the identity relationship. In other words if you eliminate some of the treatment

outcomes on which effect computations are based some effects will inevitably become indistinguishable. For any choice of the identity relationship we can generate the resulting confounding relationships via this process of computing generalized interactions with the identity [33].

Response Surfaces

Response surface methods attempt to model interactions among multiple inputs and multiple outputs. First a number of sample trials are made possibly using a factorial design. The local surface slope is estimated using a linear model, then steepest ascent (descent) is used to locate a maximal region. Second order techniques are then applied to model the maximal region. The method proceeds sequentially using information gained at each step to inform the next [7]. Response surfaces allow us to examine regions of interest in the space, since robotic tasks are localized only to regions where the robot happens to be. This type of optimization is helpful in setting the normal or preferred operating conditions for a system.

3.2 Precision and Accuracy

Having established the parameters or factors of interest via factorial experiments, we would now like to determine just how sensitive the system is to these values, and in particular what level of precision and calibration are required to achieve the desired system performance (success rate). Sensitivity analysis provides us with insight into how the precision and accuracy in system information affect performance, for the parameters of interest. As indicated in the definitions of section 2.1 precision entails notions of discrimination and repeatability, whereas accuracy is the difference between world and measured values as determined by calibration.

3.2.1 Sensitivity Analysis

Sensitivity analysis is a well known and widely used method for determining how a model varies with its parameters [31]. The most sensitive parameters are those with greatest

effect on the model's outcome and it is critical to identify them. It offers a quantitative means of determining the significance of some measurement used by the system, and hence gives a type of quantitative value of information. If an analytical description is not available, or the system is extremely complicated or lacks continuous derivatives, then empirical sensitivity methods are applied. This is the case with most of the complex robotic systems we are interested in.

The empirical approach to sensitivity analysis generally involves varying parameters and observing changes in system behavior. The direct observation method simply sets different parameter values for each run of the system. Numerous observations can be used to estimate partial derivatives about a Normal Operating Point (NOP), generating a "response surface" for the system relative to the parameters varied. Parameters with known or assumed dependencies could be varied simultaneously to explore their relationship.

We are interested in testing the system sensitivity to the precision and accuracy of its measurements. Let us assume we have a measurement function $N(f)$, and that we can vary its discrimination and observe the effects on system outcome. We can further vary its accuracy and repeatability. If we have in mind a desired level of system performance we can then determine the acceptable bounds on these precision and accuracy characteristics for the systems under analysis.

The sinusoidal variation of parameters is a more sophisticated empirical technique which addresses the combinatorial explosion in the number of required evaluations for systems with many parameters and many parameter interactions. The general principle is that if parameters are varied sinusoidally, at distinct frequencies, during a single run of the system, the sensitivity and interactions among many parameters can be determined all at once. The method requires first running the system at the NOP, then with sinusoidally modulated parameters. The power spectra of these two output sequences are used to form a spectral signal to noise ratio at each frequency [58]. Peaks in this set of ratios occur at the frequencies at which the different parameters were modulated as well as at frequencies related to products among parameters or interactions. Depending on the choice of modulation frequencies relative to system behaviour, the magnitude of these

response peaks helps estimate the importance of the parameter or interaction. Sinusoidal variation can provide a starting point for direct observation experiments.

3.2.2 Measuring Cost versus Performance

Sensitivity analysis gives us an estimate of the ε error acceptable to the system, in order to perform at required levels. Information-based complexity theory suggests some tools to determine the computational cost to achieve the specified uncertainty level ε. We will employ the notation and abstraction of ε-complexity in the form of a measure on the cost of our implemented systems, to achieve this required error level.

ε-complexity is specified for real, normed, linear spaces $\mathcal{F}$ and $\mathcal{G}$, so the $\{success, fail\}$ outcomes we have used until this point are not really appropriate. We will instead use outcome states, such as part location, for this analysis. The problem is posed for an ideal task implementation S which maps our input state parameters to goal locations G:

$$S : \mathcal{F} \rightarrow \mathcal{G}.$$

Again we have the measurement operator $N : \mathcal{F} \rightarrow \Re^n$ which gives us the values representing $f \in \mathcal{F}$, but because they are of finite dimension, $N(f)$ actually divides $\mathcal{F}$ into equivalence classes. We want to determine how the equivalence classes generated by the discrimination of the measurement process map to the outcome space. This is the radius of information, $r(N)$ which describes the effect of the information operator on the uncertainty in approximating S. If $[f]$ is the set of problem elements from $\mathcal{F}$, which N maps to the same equivalence class as f, then $r(N, f)$ is the radius of $S([f])$ in $\mathcal{G}$. $r(N)$ is defined as $\sup_{f \in \mathcal{F}}\{r(N, f)\}$, giving a lower bound on how well any approximation for S using N can hope to perform.

We can also use r to look at the value of information in an information theoretic sense related to the notion of mutual information, $V(N) = \log_2(\frac{r^{ave}(0)}{r^{ave}(N)})$. $r^{ave}(0)$ is the case where no information is available [49]. This can be compared to our preceding sensitivity analysis.

In the case of evaluating and comparing task implementations we have some implementation $\Phi : N(f) \rightarrow \mathcal{G}$ which approximates S. Algorithms can be evaluated under

various criteria for optimality. In this framework worst case error for Φ is defined as $e(\Phi, N) = \sup_{f \in \mathcal{F}}\{\|S(f) - \Phi(N(f))\|\}$. Φ is said to be optimal for S and N when $e(\Phi, N) = r(N)$. Average case analysis is also possible where a probability measure on $\mathcal{F}$ determines how samples are selected for the approximation. One now wants to minimize the expectation with respect to this probability measure $E(\|S(f) - \Phi(N(f))\|)$.

We can assign an ε-complexity value only if we look at the details of the computations involved, thus we will no longer view the systems as black boxes. A model of computation is required in a complexity framework to determine the available operations and their associated costs. For example Packel and Traub [49] use a model where infinitely precise real numbers are the elements, the information operations $N(f)$ have a dominant cost of c units and arithmetical and comparison operations in $\mathcal{G}$ are exact and have a 1 unit cost. Using such a model the computational cost for a particular Φ can be computed. For a specified error ε, the minimal worst case cost of computing an approximation for S is

$$comp(\varepsilon) = \inf_{e(\Phi,N) \leq \varepsilon} \sup_{f \in \mathcal{F}} \{cost(N, f) + cost(\Phi, N(f))\}.$$

A pair (N, Φ) which satisfy this complexity are optimal for ε.

The ε-complexity ideas of including the cost of information operations and explicitly accounting for inaccuracy and noise in measurements are very appealing for evaluating robotic systems. Like other complexity formalisms it gives us a single uniform counting scheme, but in this case for information as well as computation. Thus ε-complexity suggests a metric for comparing disparate robotic implementations of the same task, with respect to measurement accuracy and computation.

3.3 Comparing Implementations

The framework we present here allows us to analyse and tune the information used in implemented tasks, and with cost metrics similar to ε-complexity we can actually compare different implementations of the same task. This allows us to examine fundamental task information.

3.3.1 Part Orienting Systems

We will demonstrate our comparisons on two implemented part orienting systems. One is a sensorless push-orienter modelled on Peshkin's system [51]. The second is a stereo vision based system which localizes the part in three space, then manipulates it based on the coordinates provided by the vision system. Both systems are summarized briefly below and described in detail in chapters 4 and 5.

Push Orienting

Peshkin [51] describes a planning method for generating sequences of manipulations to transform a part from an unknown initial orientation to a desired final orientation. His work is based on the design of fence feeders, but can equally well apply to a robot arm holding a fence and applying similar pushing operations. We have implemented a version of the latter. The planner is based on a construct called the *configuration map*, a matrix which groups discretized initial orientations according to the final orientations which result after a push at a particular fence angle α. Final orientations are determined by the corner of the part contacting, then rotating to align with the fence. Rotation direction is determined by the friction between part and fence, and the centre of mass of the part [41]. For some discretization of the full range of fence angles, these maps are computed. Planning is achieved by a search of the tree of all push sequences with pruning.

Visual Part Orienting

A second system for part orienting uses a stereo pair of cameras, calibrated to a robot arm's workspace to localize the parts to be positioned. A high speed pipeline processor computes the zero crossings of $\nabla^2 G$ for regions of both images. The calibration mapping allows selection of these subwindows and a fixed range of disparities based on the robot's workspace region of interest. Calibration also allows computing the epipolar transformation used to warp the right image into the left for the correlation match algorithm. The resulting disparities are passed to a transputer network where edge pixels of similar

orientation and depth are grouped and fitted to a model. The major components or axes of the part will tend to give strong world edge responses, relative component positions matched to the edge positions give us an estimate of part position. Simple arm sequences parameterized by the computed location are executed to pick and place the part in a desired final location.

3.4 Framework Summary

The methods described in this chapter form a framework that can be applied in an ordered way to analyse and understand the information requirements for a task implementation. To begin, we hypothesize all of the parameters which could possibly affect the performance of our system. This may initially be a large number; for our part orienting task we begin with 11 parameters for the push system and 9 for the vision system. These will not be of equal importance to system performance and since we want a minimal parameter set, we begin with a two level factorial experiment to identify the most important parameters and interactions. With as many as 11 parameters we will in fact use a fractional factorial experiment, confounding interactions involving large numbers of parameters. A useful property of these designs is they not only tell us which parameters are significant, but which are insignificant. It can be very useful to eliminate from consideration a parameter to which the system is insensitive.

The next step is to ensure our systems are in their optimal operating range for these key parameters. To this end we begin to search in the range of treatment values or combinations which have shown promising outcomes in our initial experiments. By looking at the sensitivity of system performance to variations in the values, precision and accuracy of significant parameters, we can set these parameters appropriately. Sensitivity analysis also reinforces our experimentally determined parameter set—since the most sensitive parameters are generally the most critical.

Our evaluations of precision and accuracy allow us to determine an ε value for desired system performance. Finally cost/performance metrics allow us to quantify the difficulty of achieving this ε and therefore the required performance level.

The steps of this framework can be summarized as follows:

1. Hypothesize relevant state parameters.
2. Identify significant system state variables via experimentation, using factorial design.
3. Use sensitivity analysis and/or response surfaces to determine the precision and accuracy required in each parameter to achieve desired system performance, possibly hypothesizing other relevant parameters (return to 1).
4. Use cost/performance metrics to quantify precision and accuracy costs.

This type of empirical investigation gives a clear picture of the subspace of the task implementation and its precision and accuracy characteristics. On the basis of this analysis we can compare different complex task implementations.

Chapter 4

Push Orienting

In this chapter we describe the robotic equipment used and the first of our part orienting systems. Push planners compute sequences of fence angles used to orient parts, based on the physical properties of the part and fence and their interactions under quasi-static dynamics assumptions. The push planner we have implemented is based on the 1988 work of Peshkin and Sanderson [51, 50]. Push orienting was selected because, as a "sensorless" method it is in sharp contrast to the vision-based part orienter described in the next chapter. Very recently more direct methods for computing fence angles for fence feeders have been proposed [74], but the system described here, although slower, is nonetheless effective.

4.1 Apparatus

The equipment used to demonstrate sensorless and sensor-based manipulation strategies consists of a CRS A-460 6-dof robot arm with kinematics and servo computations performed by a pair of INMOS transputers. The Kinematics transputer computes forward and inverse kinematics for the robot. The Servo transputer communicates with robot controllers to generate motion. These processors are in turn connected to a larger network of four transputers which run straight line interpolated motion routines, and in the sensor-based case compute the location of the part to be manipulated. The vision system consists of a pair of Sony CCD cameras connected to the DIGICOLOR board of the DATACUBE pipeline image processing system. The image pair is passed from the DIGICOLOR to a MaxVideo-200 board to compute correlation stereo disparities, which are in turn passed across a British Aerospace data bridge to the same network of transputers. This configuration is illustrated in figure 4.1. Programs on the transputers

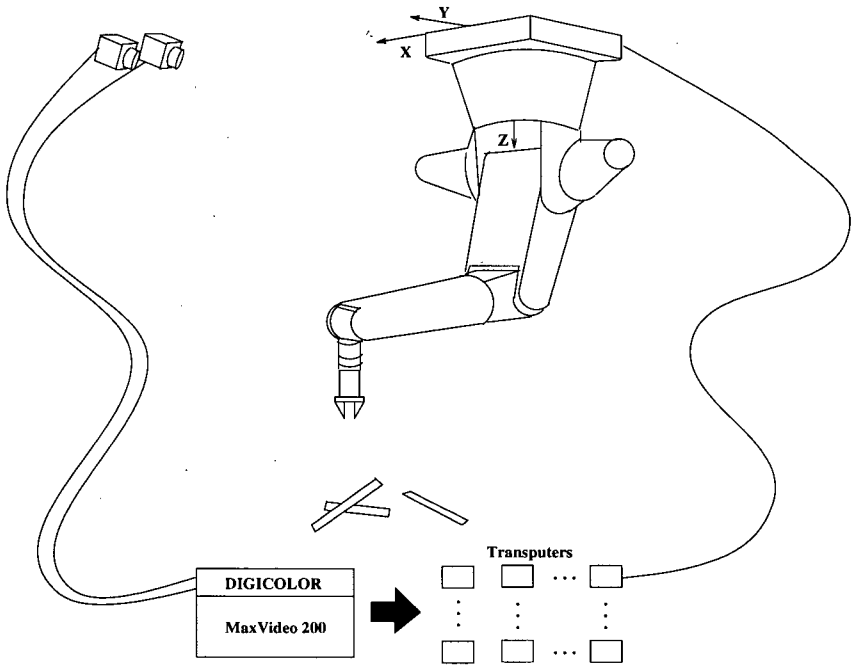


Figure 4.1: The controller and inverse kinematics for the CRS A460 robot reside on a pair of dedicated INMOS Transputers. The stereo pair of cameras are attached to the DATACUBE Pipeline Processor running a correlation stereo algorithm. The DATACUBE MaxVideo-200 dumps any image displayed to the DIGICOLOR board to one of the network of transputers via a British Aerospace transputer framestore. The transputer network completes segmentation and localization of parts and guides the robot in manipulating them.

and DATACUBE hardware are initiated from a SUN workstation. The entire system is calibrated by moving the arm through a cube of world positions and determining the image location of a circle on the robot gripper. The resulting set of world and image point pairs are used to calibrate both cameras to the robot's workspace.

4.2 The Robot

The CRS A-460 robot is PUMA-like and has 6 degrees of freedom. It is a standard light industrial manipulator produced by the CRS Plus company of Burlington, Ontario. The transputer-based inverse kinematics provided by CRS however, has a number of idiosyncrasies including rather ungraceful behaviour near singularities. Other robot limitations include wrist rotation limits of plus or minus 180 degrees, (rather than +/- 360) which

cause frequent wrist flips in trajectories which attempt extended straight line motion.

The robot has been suspended base uppermost from an aluminum frame upon which lights and cameras are mounted. The work surface is 105 cm below the robot base.

4.2.1 Straight Line Interpolated Motion

Routines for straight line interpolated motion are required for the pushing operations and were also used in generating vision-based pick and place motions. The inverse kinematics routines provided by CRS were used to generate trajectories, despite rather mysterious behaviour in regard to changes in arm configuration. For this reason trajectories are position interpolated rather than using velocities. The classic method for generating smooth motions picks knot points along the trajectory and computes joint positions at these locations [67, 26]. Trajectories in joint space are then approximated by a spline so that they can be sampled at a high rate by the controller. The spline values of mid points between knots are then compared to the desired world position determined by the world trajectory. If the approximated position is too far from the desired trajectory a new knot point is added and the spline is recomputed. Joint velocities were constrained to be zero at the start and end of the motion. Sudden wrist flips resulting from joint limits and not warned of by the inverse kinematics, appear as discontinuities in the joint trajectories, allowing the system to detect them, and reposition the wrist to make the straight line motion possible.

4.3 Push Planner

The implemented planner for push orienting parts is essentially a version of Peshkin and Sanderson's work on fence feeders, applied to a robot actively pushing the part with a fence [51]. The planner is based on a construct called the *configuration map* M_α, a matrix which groups discretized initial part orientations according to the final orientations which result after a push at a fence angle α. This idea is illustrated in figure 4.2. A part contacted by a pusher at a corner will rotate into a stable alignment with the fence. Thus a range of input orientations converge to a single (or in the discretized case a small range

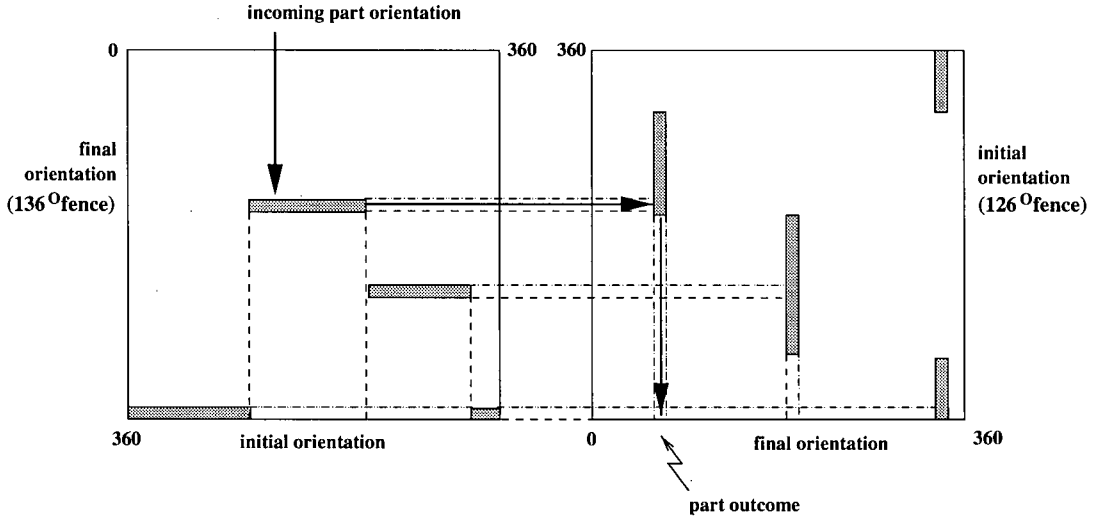


Figure 4.2: A sequence of two configuration maps reduce the possible outcome orientations to two.

of) outcome orientation as suggested in the figure. Which outcome occurs is the result of which corner feature is contacted, and of rotation direction which is determined by the friction between part and fence, and the position of the centre of mass of the part [41].

One of the parameters tested in our experiments is the discretization for fence angles. The available fence angles determine the possible outcome angles θ_f for each push, and hence the outcome bands F. Fence angles range over 180 deg at intervals determined by the discretization factor. Initial part orientations θ_i are tested at 1 deg subintervals of the 360 deg range to generate the configuration maps. A range of input orientations which maps to an outcome F_j is labelled I_j. These intervals will be referred to only by their indices in the representation below. Consider a sequence of two pushes, one at angle α the next at angle γ, written $M_\gamma M_\alpha$. As illustrated in figure 4.2 the outputs of M_α become the inputs of M_γ. This product is summarized by a code set ${}^{\gamma\alpha}C$

$$ {}^{\gamma\alpha}C_j = \{k|\ {}^{\gamma}I_j \bigcap {}^{\alpha}F_k\}. $$

In other words the outputs k of the first map become the inputs of the second map and are in turn mapped to output interval j. A particular code set C_j indicates which inputs k are mapped to outcome interval j. To determine the outcome of sequences of pushes,

products of pairwise code sets can be computed. To compute a product ${}^{\beta,\alpha,\gamma}C$ Peshkin and Sanderson use

$$ {}^{\beta,\alpha,\gamma}C_j = \bigcup_{k \in {}^{\beta,\gamma}C_j} {}^{\gamma,\alpha}C_k. $$

For example, the 2 push sequence illustrated in figure 4.2 for the angle bracket part gives rise to the code sets

$$ {}^{126,136}C_{180} = \{223\}, $$
$$ {}^{126,136}C_{323} = \{350\}, $$
$$ {}^{126,136}C_{53} = \{133\}. $$

To compose a third push at angle 178 deg we use the code set ${}^{178,126}C$

$$ {}^{178,126}C_{308} = \{53, 323\}, $$
$$ {}^{178,126}C_{91} = \{180\} $$

and compute the product ${}^{178,126,136}C$ as follows:

$$ {}^{178,126,136}C_j = \bigcup_{k \in {}^{178,126}C_j} {}^{126,136}C_k, $$
$$ {}^{178,126,136}C_{308} = \{133, 350\}, $$
$$ {}^{178,126,136}C_{91} = \{223\}. $$

Planning is achieved by considering the tree of all push sequences, generated by combining code sets. The push orienter alternates pushing from one side then the other, hence the outcome of a push is 180 deg offset when considering the next push to apply, essentially the viewpoint of the push changes. Code sets for odd and even pushes are computed and as the tree is constructed each layer alternates direction.

Peshkin proposes two pruning techniques. The first is based on the fact that if a solution exists from a node via some fence, it will also exist for a steeper fence with common code sets, thus the less steep arc can be pruned. The implemented planner generates code sets from steepest to shallowest angle, and compares code sets as they are generated, eliminating shallower fences with identical code sets. Second if a fence angle recurs in the search path leading to a node with more possible outcome orientations than its first occurrence, the branch can be pruned since any solution will be reached sooner from the earlier node.

Our implementation requires a file containing the coordinates of part corners (or the corners of the part's convex hull) centred at the part centre of mass, and the coefficient of

friction between the part and the fence. Depending on the part and the discretization in fence angles used the planner will generate numerous feasible push plans. In our experiments we look at various criteria for determining the "best" plan including optimization for minimum number of pushes, minimum push distance and overall "shallowness" of α's (to avoid the part sliding off the fence).

4.4 Robust Push Plans

Figure 4.3 illustrates a danger in depending on the rules of rotation direction when there is uncertainty in the locations of vertices, the centre of mass, or the coefficient of friction. Instead of the transition from clockwise to counter-clockwise rotation occurring at a known part orientation, it may occur anywhere in the range γ. A problem we have encountered with Peshkin and Sanderson's push planner is that it does not distinguish transitions which are "robust" in the sense that the planned transitions after the first push are not near these singular transition points. Brost has examined the inclusion of uncertainties in grasp planning [10], and has identified properties of stable squeeze grasps which are very similar to push operations.

To combat this problem we have added one-to-many mappings in our code sets to account for uncertainties. In the case of no uncertainty we can solve for the part angle θ_s where the transition from clockwise (cw) to counterclockwise (ccw) rotation occurs. The current contact feature $\mathbf{f_i} = \{f_{ix}, f_{iy}\}$ and the centre of mass $\mathbf{CM} = \{CM_x, CM_y\}$ will be rotated by the unknown θ_s so that the vector $\overline{\mathbf{f_i}CM}$ aligns with the mid (deciding)

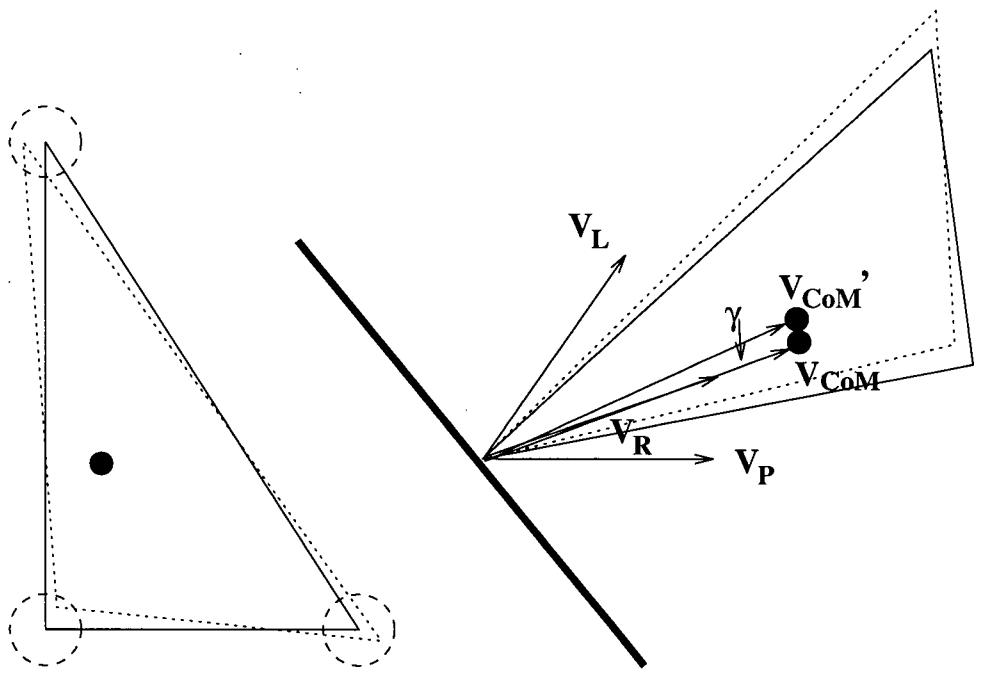


Figure 4.3: For the exact and perturbed parts on the left the push outcome will differ only in marginal cases, such as that illustrated on the right. For the part orientations θ from the point $V_R = V_{CoM}$, where the exact part reaches the transition from counterclockwise to clockwise rotation, through the range γ the perturbed part will continue to rotate counter-clockwise.

vector for the configuration. Let the angle of the mid vector be β,

$$\begin{aligned}
\mathbf{f_{i,\theta}} &= \{f_{ix}\cos\theta_s - f_{iy}\sin\theta_s, f_{ix}\sin\theta_s + f_{iy}\cos\theta_s\},\\
\mathbf{CM_\theta} &= \{CM_x\cos\theta_s - CM_y\sin\theta_s, CM_x\sin\theta_s + CM_y\cos\theta_s\},\\
\beta &= \arctan\frac{(CM_x\sin\theta_s+CM_y\cos\theta_s)-(f_{ix}\sin\theta_s+f_{iy}\cos\theta_s)}{(CM_x\cos\theta_s-CM_y\sin\theta_s)-(f_{ix}\cos\theta_s-f_{iy}\sin\theta_s)}\\
\tan\beta &= \frac{(CM_x\sin\theta_s+CM_y\cos\theta_s)-(f_{ix}\sin\theta_s+f_{iy}\cos\theta_s)}{(CM_x\cos\theta_s-CM_y\sin\theta_s)-(f_{ix}\cos\theta_s-f_{iy}\sin\theta_s)}\\
&= \frac{\sin\theta_s(CM_x-f_{ix})+\cos\theta_s(CM_y-f_{iy})}{\cos\theta_s(CM_x-f_{ix})-\sin\theta_s(CM_y-f_{iy})}
\end{aligned}$$

$$\begin{aligned}
&\tan\beta\cos\theta_s(CM_x - f_{ix}) - \tan\beta\sin\theta_s(CM_y - f_{iy})\\
&\quad = \sin\theta_s(CM_x - f_{ix}) + \cos\theta_s(CM_y - f_{iy})
\end{aligned}$$

$$\begin{aligned}
&\tan\beta\cos\theta_s(CM_x - f_{ix}) - \cos\theta_s(CM_y - f_{iy})\\
&\quad = \tan\beta\sin\theta_s(CM_y - f_{iy}) + \sin\theta_s(CM_x - f_{ix})
\end{aligned}$$

$$\begin{aligned}
&\cos\theta_s(\tan\beta CM_x - \tan\beta f_{ix} - CM_y + f_{iy})\\
&\quad = \sin\theta_s(\tan\beta CM_y - \tan\beta f_{iy} + CM_x - f_{ix})
\end{aligned}$$

$$\begin{aligned}
\frac{\sin\theta_s}{\cos\theta_s} &= \frac{(\tan\beta CM_x-\tan\beta f_{ix}-CM_y+f_{iy})}{(\tan\beta CM_y-\tan\beta f_{iy}+CM_x-f_{ix})}\\
&= tan\theta_s.
\end{aligned}$$

We can compute θ_s for each vertex as follows: first determine which of the three vectors deciding rotation sense is between the other two. The transition from clockwise (cw) to counterclockwise (ccw) rotation occurs when this mid vector $\vec{\mathbf{v}}_{\mathbf{mid}}$ is aligned with the vector $\vec{\mathbf{v}}_{\mathbf{CM}} = \mathbf{f_i R}_\theta - \mathbf{CMR}_\theta$ from the vertex $\mathbf{f_i}$ (of the rotated part) to the centre of mass **CM**. Since $\vec{\mathbf{v}}_{\mathbf{mid}}$ is fixed by the fence angle, push direction and coefficient of friction, we can find the θ that results in this alignment.

We can use uncertainty bounds for Centre of Mass, part vertices and friction, to bound the range of θ's in which θ_s may fall. To resolve the problem that part orientations in this uncertainty region will map to two possible outcomes, we simply add each of the discretized thetas in this range to the code sets for both outcomes. As the planner searches for a plan sequence which maps to a single output part orientation, these extra mappings will be resolved like any other. This method for counteracting uncertainty was

also tested as an experimental option with limited success. Adding possible outcomes can make finding a plan with a unique outcome difficult or impossible. In other words by subtracting away all the uncertain transitions we may not be left with a viable plan.

Chapter 5

Vision-based Part Orienting

5.1 Introduction

Vision for robots must provide 1) efficient use of time and other resources 2) timely, relevant information on which the agent can act, and 3) sufficient accuracy in localization or matching observations to a map or model to perform a task reliably. To achieve these characteristics, systems must frequently exploit prior calibration, restricted processing windows and efficient specialized image processing hardware. In this chapter we describe in detail some of the steps required to make a calibrated stereo vision system work with the speed and reliability required for a robot arm manipulating simple parts. Most of the material in this chapter has been previously published in [46] and [47].

The system described here is representative of visual part orienting. It uses a stereo pair of cameras, calibrated to a robot arm's workspace to localize the parts to be positioned. A high speed pipeline processor computes the zero crossings of $\nabla^2 G$ for regions of both images. The calibration mapping allows selection of these subwindows and a fixed range of disparities based on the robot's workspace region of interest. Calibration also allows rectification of the right image into the left according to the epipolar transformation using an image warping device, for use in the correlation match algorithm. The resulting disparities are passed to a transputer network where edge pixels of similar orientation and depth are grouped and fitted to a model. Simple arm sequences parameterized by the computed location are executed to pick and place the part in a desired final location. These essential ideas of calibration, windowing, rectification by warp, correlation stereo and simple localization by line fitting form the basis of a viable and fast stereo vision system for manipulation.

The idea of transforming a stereo pair of images so that the corresponding epipolar

lines are aligned with the x axis has been proposed as a means of reducing the search required to determine correspondences [4, 28]. Essentially searching along the epipolar lines reduces the search to one dimension. Rather than aligning the epipolar lines with the x axis we warp only the right image, so that for each disparity along the epipolar line, the right image pixels are moved into correspondence with those of the left image. To achieve this transformation at sufficiently high rates we have devised a biquadratic approximation, which can be used by an image warping engine to perform the rectification at frame rates.

The notion of 'restricting attention' or selecting only a relevant subwindow of a standard 512×480 image is another key aspect of getting the information the robot needs for the job. Our agent is a fixed manipulator and therefore we can similarly fix our region of interest to the environs of the work surface. Knowing the range of distances likely to be of interest we can also fix the range of disparities we check, further reducing computation time.

The vision system localizes parts based on the disparities extracted by the high speed pipeline. There are two types of parts addressed here: right angle parts, including the shelf angle bracket used as the example in this chapter and the triangle and "L" parts used in the detailed experiments of chapter 6; and the curved "G" part. The right angle parts are localized by grouping edgels via a local Hough transform, then applying a 3-D line fit to determine the components or "arms" of the part to be manipulated. The two arm components of the part are associated by applying two constraints: that they must be approximately orthogonal in 3-space and separated at most by a distance proportional to the size of the part. For the "G" the localization method is based on its oval shape with a gap on one side. These are crude but sufficient methods for obtaining part location to within the tolerances defined by the task.

5.2 Stereo Vision

5.2.1 Calibration

In order to relate the coordinate frame of a robot to that of its cameras, some level of calibration is useful. Since the cameras used are fixed to the frame suspending the robot we chose to perform a rather precise calibration. To that end a 2.5cm diameter white dot was affixed to the robot's gripper, and used as the target for generating pairs of world and image locations for both of the stereo pair of cameras. To generate calibration data the robot arm is sent to a series of world locations which form a 3D block of points. For each arm position a blob finder determines the image location for each camera and the DIGICOLOR frame grabber. The world location is determined by the commanded location as executed by the inverse kinematics of the robot arm. Having acquired a set of over 100 (world,image) point pairs we run a version of Tsai's camera calibration algorithm [70] (implemented for the LCI by R. Barman).

The blob finder used to determine the image locations of the gripper dot was intended to run without user intervention. For each new plane of arm positions it flood fills the whole image looking for intensity transitions of sufficient magnitude, and hence "white" regions whose sizes fall within the blob thresholds. As more positions are accumulated the program restricts its search region according to where other blobs in the same row and column have been found. When a likely blob has been found, its immediate region is histogrammed and the modes determined to find the optimal threshold for discriminating blob from surround. The centre of mass of the blob or dot image location is computed accordingly.

For each camera the available implementation of Tsai's algorithm returns the centre of the frame buffer relative to the lens (C_x, C_y), the interpixel distance in the image plane (dx, dy), the horizontal scale factor S_x, the first order radial distortion k_1, and the camera's true focal length f. These internal camera parameters allow us to transform pixels to image plane coordinates and vice versa, and to project points in the camera coordinate frame into the image. The external parameters calculated include roll, pitch

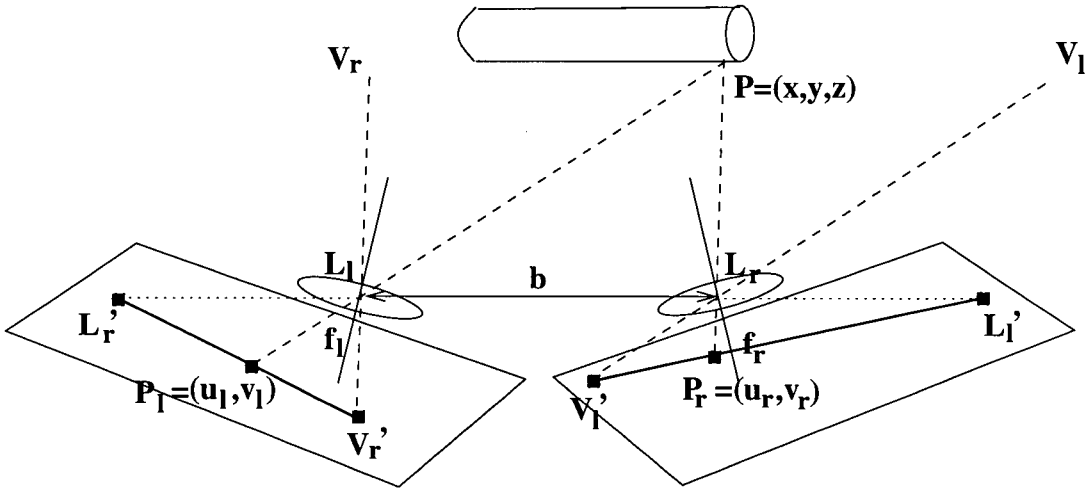


Figure 5.1: Geometry of a stereo pair with arbitrary relative camera positions.

and yaw and the translation vector for the camera in the robot's coordinate frame. Thus we have transformations from the right and left cameras to the robot frame, and can compute the transformation from one camera to the other.

The root mean squared error of the calibration is approximately 0.5 pixel, depending on the camera positions and the region of world points used. The size of the error is dominated by the poor kinematic calibration of the robot itself. The robot has high precision in repeatability, but tests indicate that depending on the robot configuration, the position resulting from a commanded world (x, y, z) position may vary by 1cm. Since we are interested in generating part locations for manipulation by the robot, the fact that robot inaccuracies are absorbed into the calibrated camera model is not a problem. We want to calibrate *relative* to the robot so that even though in an absolute sense the position is slightly in error, the position is correct in terms of commanded robot positions.

5.2.2 Standard Transformations

Much has been written about depth from stereo images, but we will touch only briefly on the basics of the method implemented for sensor based manipulation. For a more complete exposition on the subject see Horn [29].

Having calibrated both cameras to the manipulator workspace we can proceed by

assuming that the homogeneous transformation matrices from the right camera frame to the left R_{rl}, and left to right R_{lr}, are known. Corresponding points in the right and left images will occur along epipolar lines. An epipolar plane contains the lens centres L_l and L_r of the left and right cameras. An epipolar line occurs at the intersection of an epipolar plane with one of the image planes.

For a particular stereo camera configuration all the epipolar lines in an image will originate at the projection of the other camera's lens centre. A world point P will be imaged on epipolar lines defined by a plane containing P, L_r and L_l. In the left image a point projected at P_l could be any point on a ray from L_l through the actual world point P to the vanishing point. When we project this ray to the right coordinate frame we have a line from the projection of the left camera origin $L_l\prime$ to the projection of the vanishing point V_l. Thus in the right image the corresponding P_r will appear on the epipolar line bounded by projections of L_l ($L_l\prime$) and the vanishing point V_l of the ray from L_l to P, $V_l\prime$. This arrangement is illustrated in figure 5.1. These epipolar lines constrain where we search for correspondences in one image given an image feature in the other image.

Once a corresponding pair of image points (P_l, P_r) is located on the appropriate epipolar line, the depth of the world point they depict can be computed. For a camera c we know the ratios

$$\frac{u_c}{f_c} = \frac{x_c}{z_c} \; and \; \frac{v_c}{f_c} = \frac{y_c}{z_c}$$

from standard projective geometry. Given the transformation R_{rl} from right to left and $(\frac{u_r}{f_r}, \frac{v_r}{f_r}, 1)$ we also know the relationships:

$$\begin{aligned}
\left(r_{0,0}\tfrac{u_r}{f_r} + r_{0,1}\tfrac{v_r}{f_r} + r_{0,2}\right) z_r + r_{0,3} &= \tfrac{u_l}{f_l} z_l, \\
\left(r_{1,0}\tfrac{u_r}{f_r} + r_{1,1}\tfrac{v_r}{f_r} + r_{1,2}\right) z_r + r_{1,3} &= \tfrac{v_l}{f_l} z_l, \\
\left(r_{2,0}\tfrac{u_r}{f_r} + r_{2,1}\tfrac{v_r}{f_r} + r_{2,2}\right) z_r + r_{2,3} &= z_l,
\end{aligned}$$

where the $r_{i,j}$ are the elements for R_{rl}. We know f_l and f_r from calibration, so we use

the first and third equation to solve for z_r, substituting:

$$\begin{aligned} C_1 &= \left(r_{0,0}\tfrac{u_r}{f_r} + r_{0,1}\tfrac{v_r}{f_r} + r_{0,2}\right); \\ C_2 &= r_{0,3}; \\ C_3 &= \tfrac{u_l}{f_l}; \\ C_4 &= \left(r_{2,0}\tfrac{u_r}{f_r} + r_{2,1}\tfrac{v_r}{f_r} + r_{2,2}\right); \\ C_5 &= r_{2,3}; \end{aligned}$$

The result is the following solution for z_r:

$$\frac{+\begin{pmatrix} C_1 z_r + C_2 = C_3 z_l \\ -C_3(C_4 z_r + C_5 = z_l) \end{pmatrix}}{(C_1 - C_3 C_4) z_r + (C_2 - C_3 C_5) = 0}$$

$$z_r = \tfrac{C_3 C_5 - C_2}{(C_1 - C_3 C_4)}$$

5.2.3 Edge-based Correlation

Bülthoff, Little and Poggio propose a three stage parallel method for computing optical flow [12]. First a matching phase compares (ANDs) two images at pixel offsets which represent various point velocities. The matched image features may be edges or raw or filtered intensities. For each offset match an excitatory neighbourhood is summed, on the assumption that if the motion field is locally uniform for the velocity indicated by the offset, a large total excitation will result. A winner take all strategy selects the offset for each feature which generates the maximum excitation.

A similar approach can be taken to compute correspondences for stereo matching. The images are now a pair in space not time, and the offsets, or disparities of interest are those along the epipolar lines described above. In the interest of reducing data and therefore processing time, the implemented stereo algorithm uses the "edges" or zero crossings of $\nabla^2 G$, as features giving a rather sparse set of correspondences.

5.2.4 DATACUBE Pipeline Implementation

The implemented vision system uses a pipeline consisting of a Digicolor colour frame grabber and MaxVideo 200 (MV200) image processing system. The MV200 includes a

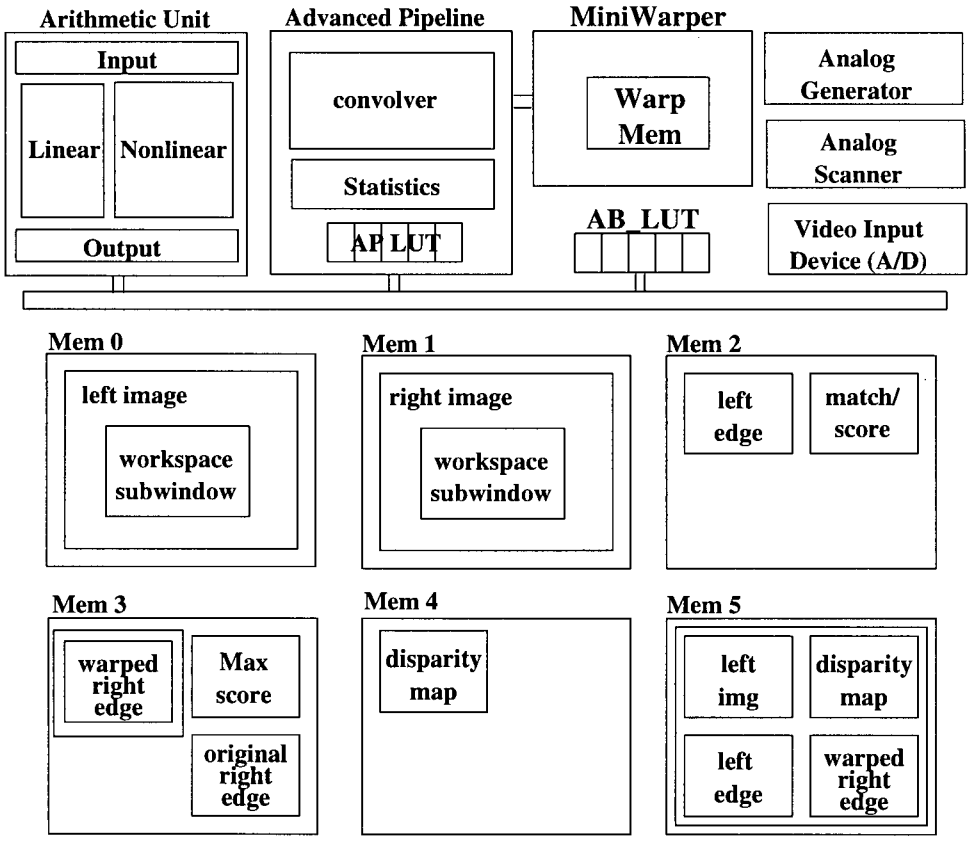


Figure 5.2: Relevant components of the MV200. The mapping of intermediate images to memory is indicated.

crossbar and 6 frame memories, as well as analog and digital grabbing and display functions, and convolution and arithmetic operations. In addition a MiniWarper is installed on the MV200, allowing polynomial and arbitrary warps of images. Most operations in the pipeline can be performed at 30Hz for a 512x480 pixel image. Computations and algorithms are not particularly flexible because they are specifically mapped to the hardware elements. Each frame memory can transmit and receive one image in each direction per pass interval, so the mapping of intermediate images to memories is important. Naturally the lookup tables (LUTs), convolver and arithmetic unit provided can only be allocated to one operation on images per time cycle as well. The components of the MV200 and the memory allocation used by the implemented stereo algorithm is illustrated in figure 5.2. In DATACUBE parlance allocated image regions in memory are called surfaces, operations moving images from a transmitting source image to a receiving destination image are referred to as pipes.

Two pipes are set up to run continuously at 30Hz. The grabbing pipeline continuously acquires image pairs from the cameras and stores them in the indicated surfaces on MEM1 and MEM2. The display pipe continuously outputs the surface on MEM5 composed of four display windows containing the left image, the disparity image, the left edge image and warped right edge image. These connections are illustrated in figure 5.3.

The computation of stereo disparities via the MV200 is described in a schematic form below. The operations under each pass take place simultaneously. After the image is acquired the algorithm is performed on smaller image subwindows extracted based on the size and location of the workspace region of interest. Windows smaller than the NTSC standard 512x480 pixels can be processed at correspondingly higher rates. Usually eight disparities are tested for each image pair although our experiments tested larger numbers of disparities. The disparity value represents a distance along the epipolar line for a particular pixel in the left image, thus the right image is warped based on a different transformation for each disparity.

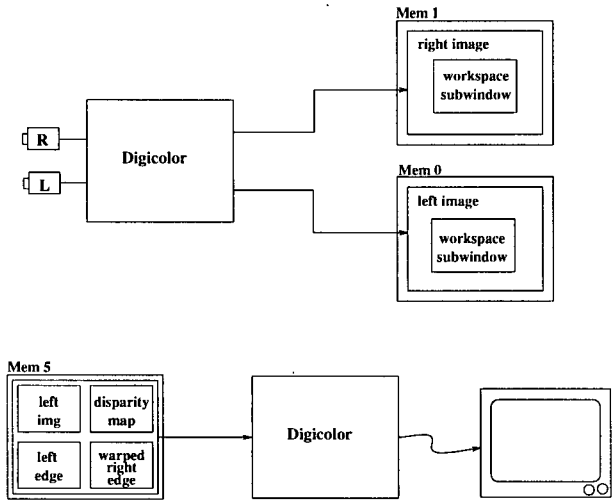


Figure 5.3: Stereo image acquisition and display of the subdivided result window from MEM05 run continuously.

CORRELATION STEREO ALGORITHM

PASS 1:

$$\boxed{\text{disparity-map}} \wedge \boxed{\text{left-edge-image}} \Rightarrow \boxed{\text{disparity-map}}$$

$$\boxed{\text{right-img-subwind}} \otimes \boxed{\nabla^2 G} \Rightarrow \boxed{\text{orig-right-edge}}$$

PASS 2:

$$\boxed{\text{disparity-map}} \Rightarrow \boxed{\text{display [dispar-map]}}$$

$$\boxed{\text{orig-right-edge}} \otimes \begin{bmatrix} 11 \\ 11 \end{bmatrix} \Rightarrow \left\{ \begin{array}{ll} \boxed{0 < pix < 4} & \Rightarrow edge \\ else & \Rightarrow 0 \end{array} \right\} \Rightarrow \boxed{\text{orig-right-edge}}$$

PASS 3:

$$\boxed{\text{left-edge-img}} \Rightarrow \boxed{\text{display [left-edge]}}$$

$$\boxed{\text{left-img-subwind}} \otimes \boxed{\nabla^2 G} \Rightarrow \boxed{\text{left-edge-img}}$$

$$\boxed{\text{const[0]}} \Rightarrow \boxed{\text{disparity-map}}$$

$$\boxed{\text{orig-right-edge}} \Rightarrow \boxed{\text{MiniWarper [MEM]}}$$

PASS 4:

$$\boxed{\text{left-edge-img}} \otimes \begin{bmatrix} 11 \\ 11 \end{bmatrix} \Rightarrow \left\{ \begin{array}{ll} \boxed{0 < pix < 4} & \Rightarrow edge \\ else & \Rightarrow 0 \end{array} \right\} \Rightarrow \boxed{\text{left-edge-img}}$$

$$\boxed{\text{left-img-subwind}} \Rightarrow \boxed{\text{display [left-image]}}$$

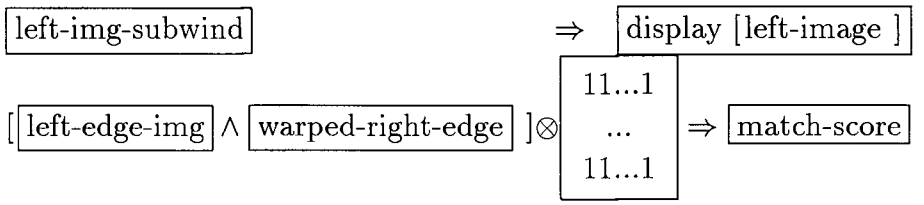
$$\boxed{\text{const[0]}} \Rightarrow \boxed{\text{max-score}}$$

FOR (cur-dispar = min-dispar, max-dispar)

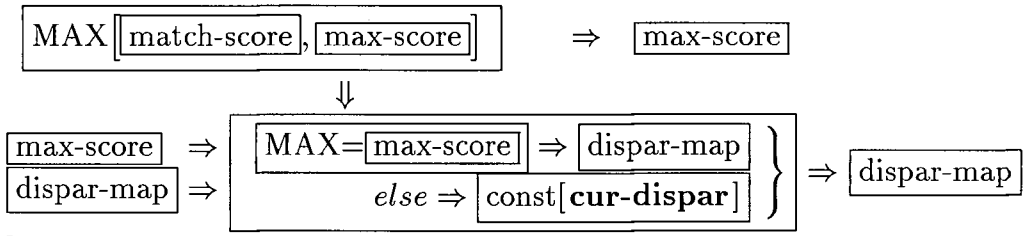
PASS 5:

PASS 6:

PASS 7:



PASS 8:



end FOR

PASS1 ANDs the disparity map and the left edge image, leaving the disparity map with values only at edge locations. At the same time the right image subwindow is passed through the convolver where it is filtered with $\nabla^2 G$, a constant value (15) is added to the results to suppress values wandering near zero. A zero or one determined by the sign of the convolved values is stored in the right edge surface. **PASS2** passes the current disparity image to the display window and convolves this sign surface with a 2×2 array of 1's. If the outcome is between 1 and 3 a sign change, or zero crossing is deemed to have occurred. The lookup table in MEM2 is set to return the edge constant (0x71) in this range, and zero otherwise.

PASS3 sends the previously computed left edge image to the appropriate display window, computes the sign of $\nabla^2 G$ for the left image subwindow, initializes the disparity surface to a constant value, and passes the right edge image to the memory surface inside the MiniWarper. **PASS4** computes the zero crossing edges for the left image via

convolution and lookup in the same manner as the right image. The left image subwindow is passed to the display surface and the surface containing the maximum score values for each image location is initialized to a constant.

PASS5-PASS8 are performed for each of a range of disparity values. **PASS5** is actually a sequence of short passes where subwindows in the right edge image stored in the MiniWarper are warped according to disparity dependent coefficients (see section 5.2.4 below).

PASS6 sends the transformed right edge image to the warped right edge surface. At the same time the previous warped right edge is sent to the display surface. **PASS7** displays the left image and performs the match and score computation. Matching is done via the lookup table AB-LUT for which the 3 nibbles of its 12 bit address can have separate sources. The low nibble of the left edge image feeds the LUT low nibble, a constant zero feeds the second nibble and low nibble of the warped right edge feeds the LUT high nibble. The LUT is set to zero except address 257 (0x101) which indicates a match. The output of the LUT is piped to the convolver where it is convolved with an 8×8 matrix of 1's. The resulting score or excitation value is transmitted to the match score surface.

PASS8 uses the arithmetic unit (AU) to perform two tasks simultaneously. The Nonlinear part of the AU uses ALUs to compute and output the pixelwise maximum of the current match score and the previous maximum score. The new maximum is then compared to the old maximum for each pixel producing a binary equal/not equal result which is transmitted to the Linear part of the AU. The Linear part outputs either the value of the previous disparity map or the current constant disparity value depending on the comparator result. The updated maximum scores and disparity map are stored on the appropriate surfaces.

The program cycles through these passes at a rate of about 4Hz (depending on the number of disparities searched) until terminated.

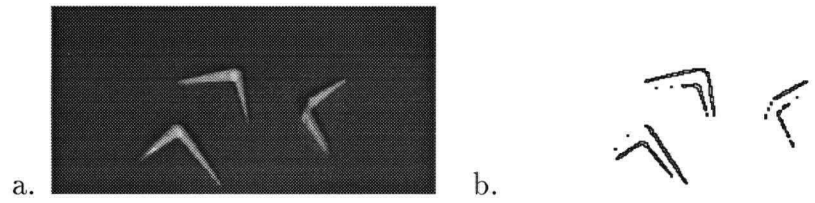


Figure 5.4: a) the left image, b) the $\nabla^2 G$ zero crossing (edge) image.

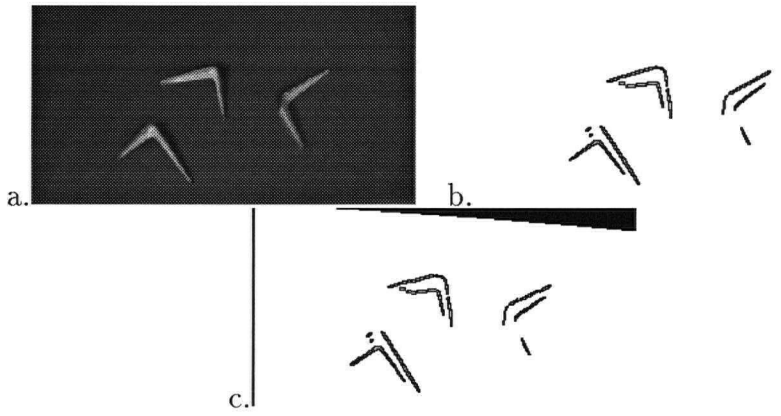


Figure 5.5: a) Right image, b) the $\nabla^2 G$ zero crossing (edge) image, c) the right edge image warped to match the left image

Windows and Disparities

In order to obtain useful results from this algorithm we would like to choose the fixed range of disparities used, to reflect likely matches in the manipulator workspace. To determine these disparities we specify a region at the work surface by its corner locations. Currently we use a 25x60x10 cm region to generate the workspace windows and appropriate disparities. The location and dimensions are specified by creating a file of coordinates and are thus easily modifiable. We iterate through points in this region on a grid of about $\frac{1}{3}$cm granularity, projecting them according to the left and right camera models. The disparities or differences between X coordinates in the left and right image, are computed and histogrammed. From the histogram we observe an approximately normal distribution of observed disparities. We therefore choose a range of disparities ± 1 standard deviation about the mean observed disparity. The searched disparities are then at discrete points in this range determined by the number of match iterations performed

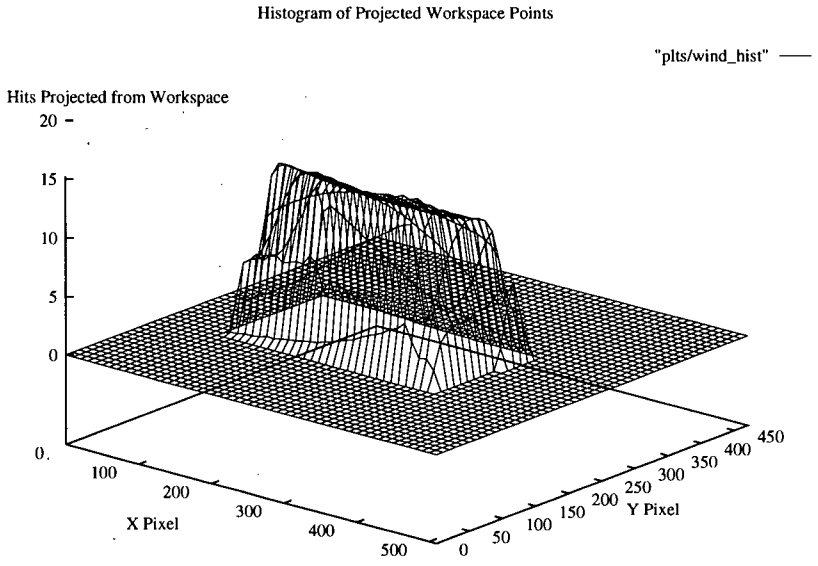


Figure 5.6: Histogram of points in the robot workspace projected into the image frame (8+).

At the same time the disparities are tested, pixels in maps of the right and left camera images are incremented each time a workspace point is projected to them. Afterward a second procedure selects the workspace image window by iteratively removing the border row or column with the lowest number of hits from each map, until five percent of the hits have been removed. The remaining windows contain 95% of the projected points from the modelled workspace. The position and size of these windows determines the subwindows extracted from each full original camera image (see fig. 5.2).

Warping Regions

As described above, we want to match a point in the left image (u_l, v_l) to its corresponding point in the right image (u_r, v_r) . We know that (u_r, v_r) will fall on the epipolar line in the right image defined by the projection of L_l (L_l') and the vanishing point V_l (V_l'). L_l' can be precomputed given our calibrated camera models by projecting the origin of the left camera into the image plane of the right camera. For each (u_l, v_l) we can compute

$V_l' = (x_V, y_V)$ by

$$x_V = f_r \frac{(r_{0,0}u_l + r_{0,1}v_l + r_{0,2}f_l)}{(r_{2,0}u_l + r_{2,1}v_l + r_{2,2}f_l)},$$
$$y_V = f_r \frac{(r_{1,0}u_l + r_{1,1}v_l + r_{1,2}f_l)}{(r_{2,0}u_l + r_{2,1}v_l + r_{2,2}f_l)},$$

where $R_{lr} = \{r_{i,j}\}$ is the rotation matrix from the left to the right camera frame. We can also compute the pixel coordinates of (u_V, v_V) of V_l' using the internal camera parameters computed during calibration. We can now compute the equation of the epipolar line in the image,

$$m_e = \frac{(v_V - v_L)}{(u_V - u_L)},$$
$$b_e = v_L - u_L m_e,$$

The DATACUBE MiniWarper contains a memory store, an address generator and interpolating multiply and accumulate (IMAC) device. A transmit gateway produces a stream of target pixel locations, the address generator then computes the addresses of the source pixels to be moved to these locations. The IMAC performs nearest neighbour interpolation on neighbourhoods transferred by the transmit gateway. The warping operation is specified by a polynomial equation, where the values of u_t and v_t are the target address and the results $u_s = f(u_t, v_t)$ and $v_s = g(u_t, v_t)$ are the source address from which the final value at (u_t, v_t) is to be obtained. The MiniWarper allows biquadratic polynomials of the form

$$\begin{aligned} u_s &= a_{2,0}u_t^2 + a_{0,2}v_t^2 + a_{1,1}u_t v_t + a_{1,0}u_t + a_{0,1}v_t + a_{0,0} \\ v_s &= b_{2,0}u_t^2 + b_{0,2}v_t^2 + b_{1,1}u_t v_t + b_{1,0}u_t + b_{0,1}v_t + b_{0,0}. \end{aligned} \tag{5.1}$$

Different coefficients can be specified for subwindows of the warped image.

As we search the range of disparities $d = (u_r - u_l)$ for points corresponding to pixels in the left image, we move along the epipolar line in the right image. We would like to transform the right image for each disparity so that the left and warped right edge images can be simply ANDed as indicated in **PASS 7** above. This requires that for location (u_l, v_l) we move a corresponding pixel (u_r, v_r) from along the epipolar line in the right image, to the address (u_l, v_l) . Knowing m_e and b_e, for some disparity d we determine the address of the source pixel in the right image by:

$$u_r = u_l + d,$$
$$v_r = (u_l + d)m_e + b_e.$$

We cannot exactly reproduce the epipolar line transformation with the set of biquadratic coefficients, we therefore approximate m_e and b_e. Because the warper's source pixel address computation is restricted to the basis $\{1, u, u^2, uv, v, v^2\}$ and the m_e term must be multiplied by u_l, m_e is approximated using the linear basis $\{1, u, v\}$. The approximation proceeds as follows: the current image window is sampled and m_e and b_e are computed at each sample point. Singular value decomposition and back substitution are then performed to approximate the m_e surface using the reduced basis, and the b_e surface using the biquadratic basis. The error is summed over all pixels in the window and if it is too high, the window is subdivided and the process is repeated on the new subwindows. Iteration terminates when the error is sufficiently small (99% of the image points have less than one pixel error in the estimated v, $|v - \hat{v}| < 1$, or the maximum limit on approximation regions is reached). A typical epipolar transform ($v_r = f(u_l, v_l)$), its approximation and the difference surface between them are illustrated in figure 5.7.

For each subregion the approximated coefficients c_i for m_e and g_j for b_e are used to compute the source pixel address (u_r, v_r) for disparity d:

$$
\begin{aligned}
m_e &= c_1 + c_2 u_l + c_3 v_l, \\
b_e &= g_1 + g_2 u_l + g_3 u_l^2 + g_4 u_l v_l + g_5 v_l + g_6 v_l^2, \\
v_r &= m_e(u_l + d) + b_e \\
&= (c_1 + c_2 u_l + c_3 v_l)(u_l + d) + g_1 + g_2 u_l + g_3 u_l^2 + g_4 u_l v_l + g_5 v_l + g_6 v_l^2 \\
&= c_1 d + c_1 u_l + c_2 d u_l + c_2 u_l^2 + c_3 u_l v_l + c_3 d v_l + g_1 + g_2 u_l + g_3 u_l^2 + g_4 u_l v_l + g_5 v_l + g_6 v_l^2 \\
&= (g_1 + c_1 d) + (c_1 + c_2 d + g_2) u_l + (c_2 + g_3) u_l^2 + (c_3 + g_4) u_l v_l + (c_3 d + g_5) v_l + g_6 v_l^2
\end{aligned}
$$

For each disparity and region then we can compute a set of warper coefficients to be used in equation 5.1:

$$
\begin{aligned}
a_{0,0} &= (g_1 + c_1 d), & b_{0,0} &= d, \\
a_{1,0} &= (c_1 + c_2 d + g_2), & b_{1,0} &= 1.0, \\
a_{2,0} &= (c_2 + g_3), & b_{2,0} &= 0.0, \\
a_{1,1} &= (c_3 + g_4), & b_{1,1} &= 0.0, \\
a_{0,1} &= (c_3 d + g_5), & b_{0,1} &= 0.0, \\
a_{0,2} &= g_6, & b_{0,2} &= 0.0.
\end{aligned}
$$

PASS 5 represents multiple warp image subwindows being transformed according to their computed coefficients via individual passes.

One difficulty with the warper is that if the source address is outside the image

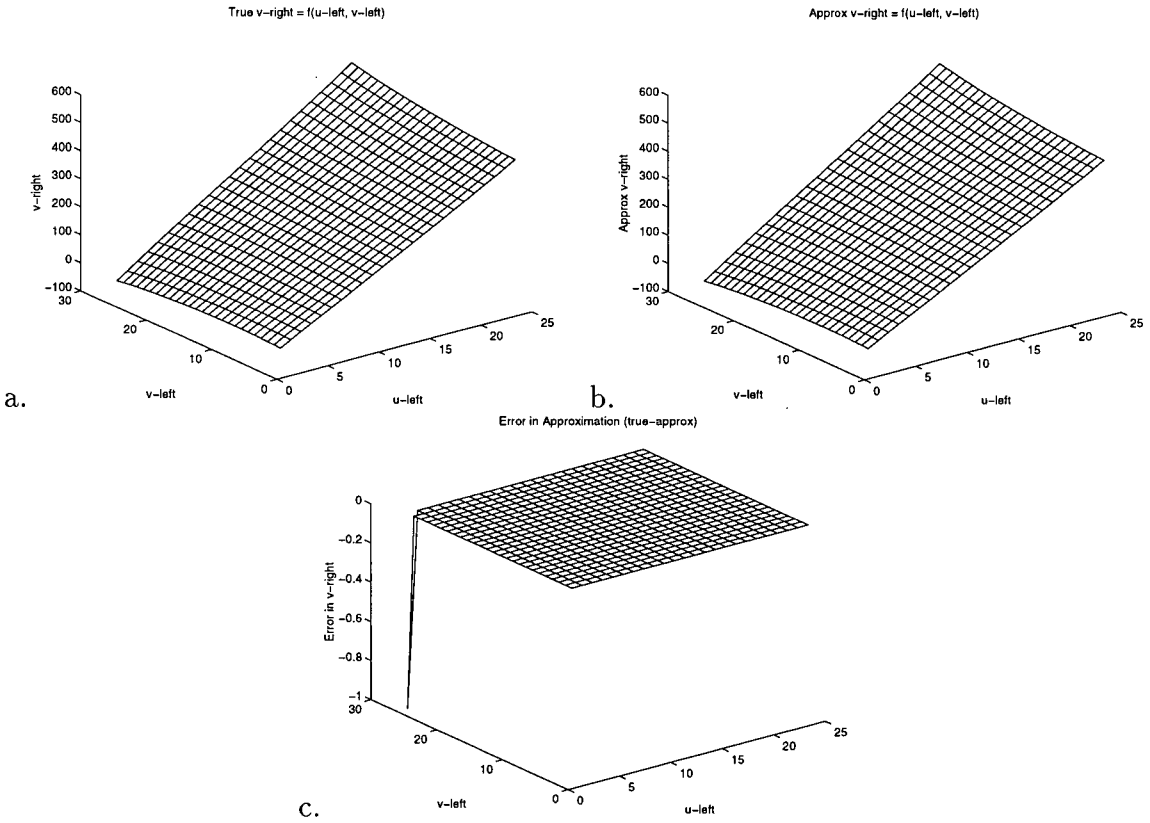


Figure 5.7: a) True $v_r = f(u_l, v_l)$, b) Approximated $v_r = f_a(u_l, v_l)$ c) the difference between the two surfaces.

window, no information is available to map to the target pixel. This is particularly evident with the epipolar line transformation because the lines tend to converge at L'_l. However, since we are not using the full camera image, the additional image values are available. We attempt to ensure dense data in the image window of interest by extending the right edge image and warp image until 99% of the source pixels for the target pixels in the workspace window are included, or the warp window has been increased by 40% over the original workspace window.

5.3 Part Localization

The set of parts to be oriented by our system includes right angle parts (a triangular part, an "L" shaped part and the angle bracket) and a "G" shaped part. For the former we

group pixels into edges in the image, then fit a 3-D line to the depth values corresponding to the grouped pixels. A perpendicular pair of such lines define the part model. For the "G" a simpler technique is employed, taking advantage of its oval shape with a 'gap' on one side.

5.3.1 Right Angle Part Localization

Disparity images from the MV200 are written to a British Aerospace transputer frame-store. The system which computes depth values and object locations is distributed on a small group of transputers, which are connected to the transputers which control the robot arm.

The object models and methods for localization are quite simple and deliberately so. The goal is to tease out what task information is being used, so the simpler the models and assumptions the better.

Grouping

We use image grouping to guide grouping of 3-D points. The localization process begins by examining the image neighbourhood of each disparity marked edge pixel using a local Hough transform [19]. The standard Hough transform applied to edge grouping computes a value $\rho = u_i \cos\theta + v_i \sin\theta$ for every edge pixel, for some discretization of $\theta = [0, \pi]$ and ρ. Peaks in the (ρ, θ) histogram indicate many edgels voting for the lines defined by these parameters.

The variant we have applied uses a precomputed neighbourhood template of θ values with respect to the central edge pixel illustrated in figure 5.8. This allows each other edge pixel in the neighbourhood to vote for a θ defined by its pairing with the centre pixel. For each edgel this template is applied, a histogram constructed and the peak θ is chosen. The appropriate ρ is computed and the global (ρ, θ) histogram is incremented. Each edgel is associated with only one point in the parameter space. We achieved better results with this method versus a true Hough transform, because it avoided repeated passes over the edge pixel set and computing ρ values for unlikely θ's at each point.

				1.77	1.57	1.37				
			2.03	1.82	1.57	1.33	1.11			
		2.36	2.16	1.89	1.57	1.25	0.98	0.79		
	2.68	2.55	2.36	2.03	1.57	1.11	0.79	0.59	0.46	
2.94	2.90	2.82	2.68	2.36	1.57	0.79	0.46	0.32	0.24	0.20
0.00	0.00	0.00	0.00	0.00	999.00	0.00	0.00	0.00	0.00	0.00
0.20	0.24	0.32	0.46	0.79	1.57	2.36	2.68	2.82	2.90	2.94
	0.46	0.59	0.79	1.11	1.57	2.03	2.36	2.55	2.68	
		0.79	0.98	1.25	1.57	1.89	2.16	2.36		
			1.11	1.33	1.57	1.82	2.03			
				1.37	1.57	1.77				

Figure 5.8: A map of the line θ values suggested by edgels in the corresponding positions relative to the current (central) edge pixel.

Local grouping appears to generate better discrimination because points which may have appropriate (ρ, θ) values will not be assigned to a line that locally "looks" inappropriate.

When all edgels have been examined, peaks in a global θ histogram determine which ρ histogram to search. As edgels conforming to these peak values are selected, they are further grouped based on their image distance from the current mass of points for the line. These distant points are split off as separate lines with the same parameters. We can also include edgels which fall in the parameter space neighbourhood of the peak (ρ, θ). The discretization of ρ and θ determine the power of the system to discriminate among lines which may lie close together in the image.

Matching

These groups of edge pixels and their 3D locations are shipped off to another transputer where a least squares fit algorithm computes the parameters for a straight line through them in 3-space. These world points are then shipped off to another transputer where a least squares fit algorithm computes the parameters for a straight line. Some pixels are eliminated because their world locations would place them below the robot's work surface. The 3D lines resulting from a typical fitting process (ordered by number of points) have

been reprojected by the left camera model and are shown in figure 5.9 superimposed on the disparity edges.

Figure 5.9: Projections of 3-D lines fitted to edge data.

The confidence ordering assigned to these lines is based on the number of pixel/world values used to approximate them. To separate the shelf angle brackets shown in the example images, or any right angle part, we apply two constraints to all pairs of lines: first is their 3-D distance too large to be part of the same bracket; and second are they orthogonal in 3-space. The outcome of this entire grouping, fitting and pairing process is illustrated in figure 5.10.

Whether a left or right edge or some mixture of points is represented by a line, a bracket arm can still be grasped because the robot's gripper opens to a width of 5cm and the angle bracket arms are less than 1cm wide. Line 1 in figure 5.9 demonstrates such a mixture of points. By choosing the average x and y values as a grasp point, the robot can still successfully grasp the angle bracket. The entire process requires 5-7 seconds on the transputers, increasing with the number of line segments that must be fitted and tested. On an ULTRASPARC architecture similar computations require much less than 1 sec., but transmitting commands to the robot becomes difficult. Naturally this model is very simple and can be fooled by more complicated parts or scenarios, the point is however that for this task, on these parts, this simple part model with accurate depth data is sufficient.

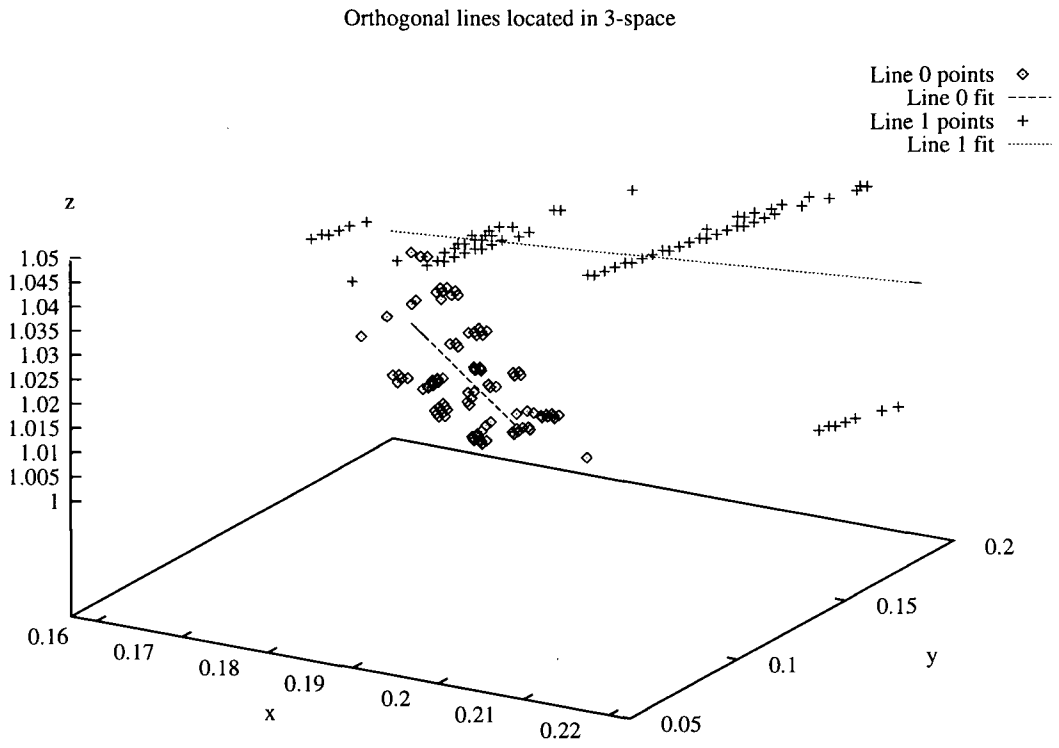


Figure 5.10: The graph shows a pair of lines, satisfying the orthogonality constraint, computed by fitting the illustrated 3-D edge points.

5.3.2 "G" Localization

Again disparity images from the MV200 are written to a British Aerospace transputer framestore, but in the case of localization for the "G" part, much less effort is devoted to grouping and fitting.

A depth location is computed for each disparity marked edge pixel and their centroid is computed. Points too distant to be part of the same "G" are discarded. A histogram is then computed of the orientation from $0 - 2\pi$ of each depth point with respect to the centroid. A fixed "gap window" of width Q bins is then slid across the histogram in search of the range of angles with fewest hits in the histogram. This range is then assumed to be the gap in the "G" (illustrated in figure 5.11), the centroid is assumed to be the centre of mass for the part.

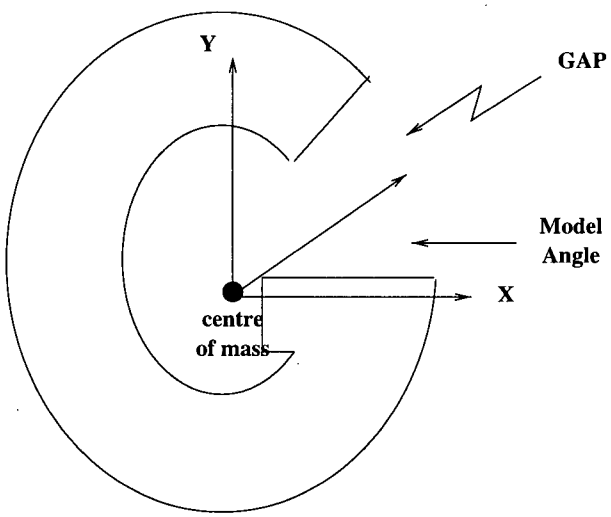


Figure 5.11: The "G" part is localized by histogramming the orientation of depth points relative to their centroid. A valley or minimum in this histogram should indicate the location of the gap in the "G" shape. The orientation at which this minimum occurs is assumed to correspond to the model angle in the figure and the centroid corresponds to the centre of mass. These two values determine the part's world position.

Knowing the fixed relationship between the gap and the centre of mass we now know the world location of the "G" part. This method does not incorporate much knowledge of the part, and is rather 2-D in nature. As long as the gap is a visible feature however, it performs well as a localization scheme.

5.4 Summary

In this chapter we have discussed some detailed issues in implementing a calibrated stereo vision system for manipulation. The system described makes extensive use of a precise calibration among the robot arm and stereo pair of cameras. Knowledge of this relationship allows constraints to be imposed on the region of interest in the images and the disparity values of interest for the robot workspace; it also provides the epipolar transformation. The latter is used to generate a piecewise biquadratic approximation for the epipolar transform, which can be used by a special purpose image warper to rectify the right image for correlation matching with the left, to produce correspondences at frame rates. As part of a larger image processing pipeline, edge pixel correspondences can be generated from image pairs at $4Hz$. Generally calibrated systems involve a tradeoff where match/calibration computations are pushed to the offline side of the information extraction process.

The localization processes we have described for our right angle ("L" and triangle) and "G" shaped parts are relatively simple. Our idea was to build simple localization systems, sufficient to the orienting task rather than complex general purpose vision systems. One side effect is that if your model is not very precise or detailed, then the system can tolerate an equivalent amount of imprecision in the actual part. The result is a system which is insensitive to small variations in part features.

Chapter 6

Experimental Results

In this chapter we detail our experimental results. An analysis of these results is given in the next chapter. The goals of the following experimental exploration of our part orienting systems, set out in chapter 3, are as follows: 1) evaluate system performance with regard to the feature parameterizations represented by the implemented part orienting systems; 2) perform fractional factorial experiments to determine the relative importance of parameters and the degree of attention they deserve; and 3) apply sensitivity analysis to determine the tolerable levels of accuracy for the systems. In chapter 7 we apply a cost/performance metric to evaluate these empirical findings under a uniform cost measure.

6.1 System Task Parameterizations

The premise of this investigation is that we have one task and two implementations representing different parameterizations of the same underlying physical 'computation'. The features or 'measurements' used define the parameterization. Anyone who builds systems knows that they contain many parameters and thresholds which must be chosen by the designer: sometimes based on theory, sometimes via intuition and limited testing. All of these design choices affect the performance of the system. Below we enumerate the actual parameterizations of the part orienting task represented by the push planner and vision based systems in the preceding chapters.

Push Orienting

Eleven parameters or factors were selected as representative for the push planning systems. These are described in brief in table 6.1. The parameters F, C and V indicate the accuracy of the measured values of the friction coefficient, centre of mass and vertices

Push Orienting Parameters		
Label	Parameter	Description
F	μ	accuracy of measured friction between fence and part
C	C of M	accuracy of the relative position of the part's centre of mass
V	$\{v_1, ..., v_n\}$	accuracy of the set of relative positions of the vertices of the part's convex hull
D	δ_α	the discretization factor of the set of possible fence angles
L	α limits	limit on the range of fence angles searched
S	steepness	limits on fence steepness (qualitative)
P	push distance	limits on push distance
N	plan length	preference for plans with fewer steps
A	friction uncertainty	uncertainty compensation for friction measurement, avoid transitions which measurements errors make uncertain
B	C of M uncertainty	uncertainty compensation for Centre of Mass measurement, avoid transitions which measurements errors make uncertain
K	vertex uncertainty	uncertainty compensation for vertex measurements, avoid transitions which measurements errors make uncertain

Table 6.1: Parameters for the push planners.

respectively. D is the discretization factor for the fence angles and hence the branching factor for the tree search performed by the push planner, i.e. π/δ_α fence angles are considered. In our coordinate system fence angles near 0 or π are essentially end on to the part and are too steep to allow viable pushes. We therefore set a limit L on the set of fence angles and search only from $\frac{0+L}{\delta_\alpha}$ to $\frac{\pi-L}{\delta_\alpha}$.

Our experience with executing push plans has shown that steeper fence angles are less reliable because, with limited fence length and push distance, parts tend to "fall off" the end of the fence or wind up under the fence because they were left out of its range by the preceding fence angle. As a result we have added a constraint S to the push planner which prefers plans with shallower fence angles. This is achieved by evaluating the average and steepest fence angles and preferring plans with either a shallower (closer to $\frac{\pi}{2}$) steepest fence angle or an equally steep, steepest fence but a shallower average

steepness. We chose to evaluate the presence and absence of this constraint to determine its usefulness and its effects on other aspects of the planner.

Inaccuracies in fence angle were tested as a possible source of error, but in general the fence angle and part outcome angle discretizations were so coarse that the tiny uncertainty in fence angle between the wrist encoder ticks had no effect.

Peshkin [50] describes methods for bounding the required push distance to align a part with a fence, based on the slowest possible centre of rotation. We have used his formulation to compute a bound on maximum push distance for each push in a plan, depending on which transitions are required at each push. Minimizing push distance is another constraint P, we added because of the real physical limits of an implemented push planner. Plans with the shortest total push distance or shortest maximum push are preferred.

In terms of costs for an active push orienter, each additional push is expensive because it requires a motion sequence. We have therefore added a constraint N which prefers plans with fewer pushes.

Finally factors A, B, and K indicate whether the uncertainty compensation described in section 4.4 is turned on for friction, centre of mass and vertex uncertainty respectively.

Visual Orienting

Nine parameters were selected as representative for the vision orienting system. These are described in brief in table 6.2.

Two camera positions were tested under the assumption that if cameras were placed nearer to the work surface, parts would have greater extent in the image and therefore be localized better. Originally, the vision system was developed with the stereo pair of cameras fixed about a metre above the work surface on the robot's frame. To test the effect of camera position M, a second location on a pair of tripods about 50cm closer to the work surface and 50cm farther apart was tested.

D and S are parameters internal to the DATACUBE stereo system. D is the number of disparities searched in the winner-take-all phase of the correlation stereo algorithm. S

Visual Orienting Parameters		
Label	Parameter	Description
M	camera position	the position and orientation of the cameras
D	$\{d_1, ..., d_m\}$	fixed range of discrete disparities
S	$\sigma_{\nabla^2 G}$	σ for the edge detector
C	calibration accuracy	number of points from which calibration is computed
A	warp accuracy	accuracy of the approximation for the epipolar warp
L	lighting	degree of special lighting required
Z	Z constraint	elimination of world points assigned depth inconsistent with work table height
P, Q	part model parameters	δ_ρ, δ_θ discretization factors for local Hough transform (triangle and "L") angle discretization and gap width for "G" localization
I	intensity image	

Table 6.2: Parameters for the Visual Orienter.

is the level of σ for $\nabla^2 G$.

Our calibration system typically acquires 100 to 180 image-world point pairs to compute the internal and external camera parameters, which are in turn used to compute the warp approximation for matching. To test whether this level of calibration accuracy C is required we subsampled the calibration data acquired for each imaging condition to approximately $\frac{1}{4}$ of the original set of calibration points, and computed a less accurate camera model.

For the warp approximation we have allowed up to 12 subregions to be approximated separately, depending on the accuracy of the warp specified. To determine how necessary this level of approximation accuracy A is, we limited the number of approximation regions to 2 in the low case.

We originally set up a pair of $150W$ photographic lights for our vision system to provide good, even contrast. For our experiments we decided to determine how crucial this lighting L was by testing the system with this special lighting and with normal room

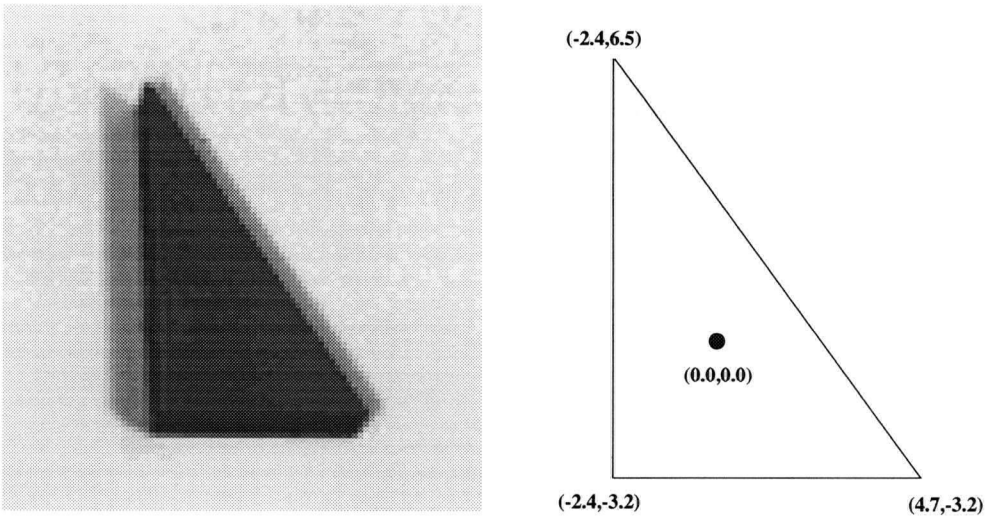


Figure 6.1: On the left is an image of the Triangular part used in our experiments, on the right the model used by the push planner.

lighting.

The depth values generated by the DATACUBE system can be quite inaccurate. One way of eliminating bad points is to use the known Z location of the work surface to eliminate spurious points. To determine how important this constraint was to system performance, we set it to very close to the work surface and a relatively wide band around it ($\pm 20cm$).

The model parameters denoted P and Q are different for the triangle and "L" versus the "G" since the localization methods are different. For the triangle and "L", P is the discretization of the ρ dimension for the local Hough transform and Q is the size of the orientation template used to vote on the orientation for the current edge pixel. For the "G", P is the discretization of 2π used to histogram the orientation of depth points with respect to the centroid. Q is the window or number of bins in this histogram over which a minimum number of hits determines the angle of the "G"'s gap.

6.1.1 Test Parts

The experiments described in this chapter were performed on the parts pictured in figures 6.1, 6.2 and 6.3. In terms of the push orienting system in particular, these parts span

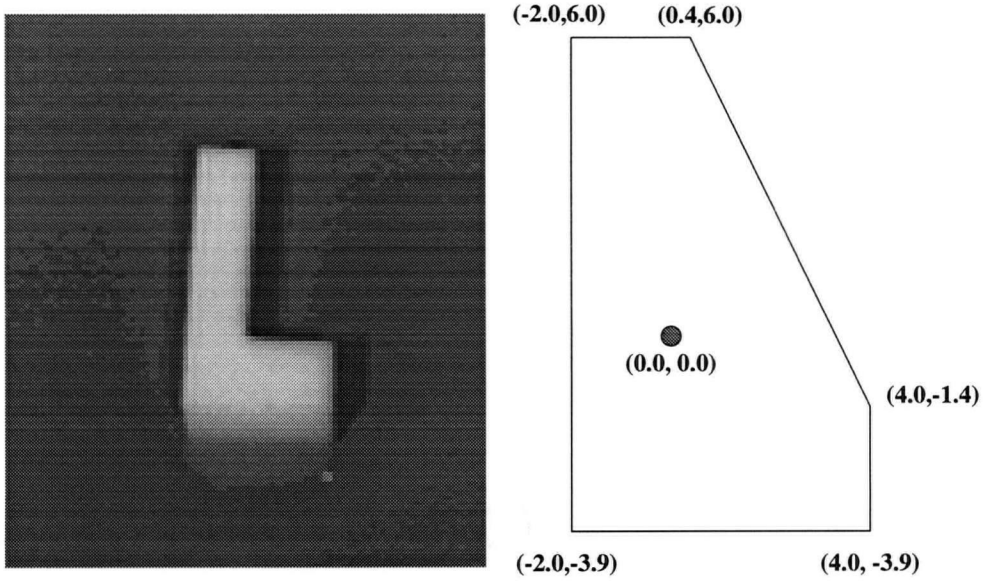


Figure 6.2: On the left is an image of the L-shaped part used in our experiments, on the right the model used by the push planner.

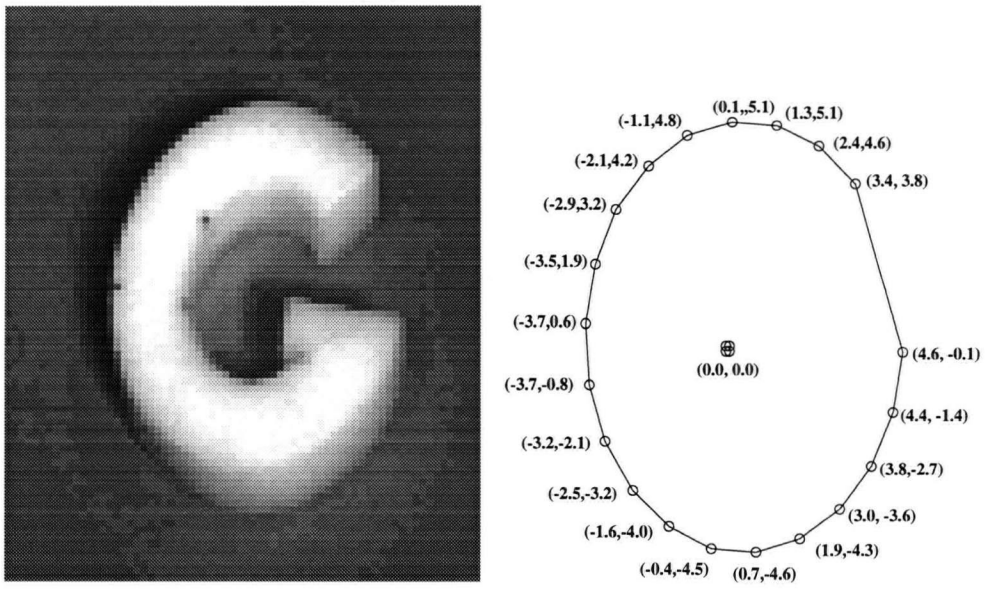


Figure 6.3: On the left is an image of the G-shaped part used in our experiments, on the right the 20 vertex model used by the push planner.

a range of complexity. The triangle (T) has three stable sides. The "L" has two short sides which tend not to act as stable resting positions and the "G" has been modelled by a polygon with 20 vertices rather than a true curve.

6.2 Methods

Based on the parameterizations described above for the push planner and vision system, we have set up a $\frac{1}{8}$ and $\frac{1}{4}$ fractional two level factorial experiment for the systems respectively. Such experiments provide useful information on the significance of various factors and their interactions.

The proportion of success values generated for the outcome of each treatment are generated at least in part by simulators. We have working physical systems, and both push plans and vision based manipulation have been demonstrated to work in the real world, but for the volume of trials required for statistical analysis we were obliged to resort to simulation. In particular to test the push planner's performance under variation in measured part parameters would have required many parts with known variation and we did not have access to such a test set. We do not have CAD models or other precise specifications for the parts used, so we cannot precisely state the bound on the error in our estimates of friction or centre of mass. This makes it difficult to precisely compare real world push planner outcomes to our simulations. Anecdotally, we actually used the simulator to determine why one of the triangle push plans we computed had real world success of only 30% for the part's nominal measured parameters. Perturbing these nominal values in simulation we saw the same drop in performance for even small perturbations, and tracked the problem to a push which depended on an alignment within the uncertainty range near the *cw*/*ccw* transition. Similarly we observed relatively poor performance for many of the tested "G" orienting plans.

6.2.1 The $\frac{1}{8}$ and $\frac{1}{4}$ Fractional Factorial Designs

Clearly we cannot run 2^p trials for as many parameters p as we have selected for the push planner and vision system. We will therefore use a $\frac{1}{8}$ fractional factorial experiment

with $\frac{2^{11}}{2^3} = 2^8$ treatments tested for the push planner, and a $\frac{1}{4}$ experiment with $\frac{2^9}{2^2} = 2^7$ treatments tested for the vision system. The process of generating fractional designs was described in section 3.1.1.

The identity relation used to generate the fraction of experiments for the push planner is

$$I = CVDLSP = FVDLNA = FCSPNA = FCDLBK$$
$$= FVSPBK = CVNABK = DLSPNABK.$$

The generated fractional factorial experiment requires 256 trials, but provides information on the main effects and all two-factor interactions confounded with only three or higher order interactions.

The identity relation used to generate the fraction of experiments for the vision system is

$$I = MDSCAL = MDSZPQ = CALZPQ. \tag{6.1}$$

This experiment requires 128 trials and also provides information on the main effects and all two factor interactions confounded with only three or higher order interactions.

We generated these identities via brute force search for the identity which confounded the highest order interactions with main effects and two factor interactions. In other words, since we believe interactions of many factors will be less significant than those with relatively few, we want our low order interactions to be confounded with the largest interactions possible. This search process itself is time consuming for large numbers of parameters, although some texts on the Taguchi method, for example, provide precomputed designs which could be useful [55].

The specific treatments applied for our examples and the outcomes observed are tabulated in appendices A and B for the push planner and vision system respectively.

In the text references to detailed calculations for two factor interactions refer to the idea that for effect AB we subtract the effect of B while A is low from the effect of B while A is high. A negative effect for AB can represent a large positive effect for B with A low, or a small negative effect etc. To determine the meaning of a significant interaction

we must frequently refer to the specific combination of treatments used to calculate the effect.

6.2.2 Simulators

The simulator for the push planner takes the part model and the selected plan (with associated maximum push distances) and outputs a success or fail based on the same *cw* or *ccw* rotation decisions used by the push planner. In addition, however, the simulator approximates the part's speed of alignment with the fence by sampling a normally distributed alignment distance with standard deviation $\frac{1}{50}$ of the maximum push distance and adding it to the minimum alignment distance. This means plans can fail because the part never becomes fully aligned with the fence. The part's slide along the fence after alignment is simulated and may cause it to "fall off" the limited extent of our fence. If some inaccuracies are assumed in the measured features, a normally distributed perturbation is added to the appropriate vectors potentially causing "bad" rotations. Outcome proportions of success are based on simulating 1000 plan executions with uniformly distributed randomly generated initial part orientations.

The experiments involving the vision system are based on several suites of test images obtained by using the CRS robot arm to place each of the parts in a series of positions and orientations spanning the workspace. Separate image sets were collected for each of the four camera position and lighting combinations (see table 6.10). The success rates obtained are based on using the known part position for each stereo image pair, compared to the position extracted by the vision system plus the constraints imposed by the arm and gripper. The image sets are based on a grid of 3 X positions by 5 Y positions plus 8 orientations at each (X, Y) location; in total 120 image pairs per set.

For the vision system, only the graspability of the part, given estimated position from the vision system and known world position, is simulated. Essentially we use a model of the gripper approaching the estimated position, and the part in the known position to determine whether a collision or successful manipulation will occur. The simulator returns success or failure.

Parameter Levels			
Parameter	Label	High Value	Low Value
Measured Friction	F	T $\mathcal{N}(0.33, 0.04)$ L $\mathcal{N}(0.30, 0.03)$ G $\mathcal{N}(0.23, 0.015)$	0.33 0.30 0.23
Measured Centre of Mass	C	$\mathcal{N}(CofM, 0.25cm)$	Measured Value CofM
Measured Vertex Position	V	$\mathcal{N}(V_i, 0.1cm)$	Measured Position V_i
Discretization Factor	D	60	30
Fence Angle Limits	L	$[5, (180-5)]$ deg	$[30, (180-30)]$ deg
Fence Steepness Limit	S	TRUE	FALSE
Push Distance Limit	P	TRUE	FALSE
Plan Length Limit	N	TRUE	FALSE
Friction Uncert. Compensation	A	TRUE	FALSE
C of M Uncert. Compensation	B	TRUE	FALSE
Vertex Uncert. Compensation	K	TRUE	FALSE

Table 6.3: Parameter values for push planner. Qualitative factors are indicated by TRUE and FALSE indicating whether the property is active. $\mathcal{N}(\mu, \sigma)$ represents a value drawn from a normal distribution with the indicated parameters.

6.3 Push Planner — Parameters and Outcomes

Table 6.3 indicates the actual values assigned to the parameters of the push planner for the treatments specified for our fractional factorial experiment. Many of these parameters are essentially treated as qualitative TRUE/FALSE options. See appendix A for tabulated treatments applied and probability of success outcomes observed.

Tables 6.4 through 6.9 contain the analysis of variance computations for the proportion of success and time required for plan computation (in minutes) respectively, for the three parts under consideration. All main effect values are indicated but two factor interactions with F value less than $F_\alpha = 2.71$ for $\alpha = 0.1$, have been pooled. We are primarily concerned with rate of success, but the timing ANOVA tables presented give an empirical version of how various parameter settings contribute to planner complexity.

ANOVA - Success Rate for Triangle					
Source		dof	Sum of Squares	Mean Squares	F
	F	1	0.017906	0.017906	0.109570
	C	1	0.530975	0.530975	3.249130
	V	1	0.007758	0.007758	0.047471
	D	1	0.202568	0.202568	1.239553
	L	1	1.616058	1.616058	9.888948
	S	1	0.478952	0.478952	2.930793
	P	1	0.007521	0.007521	0.046020
	N	1	0.211720	0.211720	1.295551
	A	1	0.033667	0.033667	0.206011
	B	1	0.016727	0.016727	0.102358
	K	1	0.018701	0.018701	0.114437
	DL	1	0.777070	0.777070	4.755029
	LS	1	0.702501	0.702501	4.298727
	LP	1	0.483748	0.483748	2.960139
	SK	1	1.223321	1.223321	7.485722
Error	ε	176	28.762029	0.163421	1.000000
Total	T	255	35.091222	214.729460	1.000000

Table 6.4: Analysis of Variance summary for for main factor effects and significant 2 factor interactions for proportion of success for triangle part for the push planner.

ANOVA - Planning Time for Triangle					
Source		dof	Sum of Squares	Mean Squares	F
	F	1	58.809727	58.809727	0.033405
	C	1	259.411289	259.411289	0.147348
	V	1	7.380278	7.380278	0.004192
	D	1	507985.834727	507985.834727	288.541304
	L	1	221988.211914	221988.211914	126.091642
	S	1	2058.890625	2058.890625	1.169472
	P	1	76381.989067	76381.989067	43.385774
	N	1	12387.690000	12387.690000	7.036338
	A	1	43.312852	43.312852	0.024602
	B	1	1184.650352	1184.650352	0.672894
	K	1	4798.736984	4798.736984	2.725733
	DL	1	185222.640625	185222.640625	105.208411
	DP	1	66471.582101	66471.582101	37.756559
	LP	1	36323.595156	36323.595156	20.632185
	DN	1	11541.473477	11541.473477	6.555678
	LN	1	8820.731602	8820.731602	5.010268
	PN	1	6407.668963	6407.668963	3.639623
	NB	1	6012.386984	6012.386984	3.415099
	DK	1	6370.700278	6370.700278	3.618625
	LK	1	8839.917101	8839.917101	5.021166
Error	ε	171	301050.755885	1760.530736	1.000000
Total	T	255	1464216.369983	831.690319	1.000000

Table 6.5: Analysis of Variance summary for for main factor effects and significant 2 factor interactions for planning time for triangle part.

6.3.1 Triangle Push Planner Results

Using our F test for significance we conclude that only those effects and interactions with F value greater than $F_{0.1} = 2.71$ make a significant contribution to the variance observed. For the triangle, table 6.4 tells us that the significant effects are C, L, S, DL, LS, LP and SK. The most significant effect is that for angle limit L, and referring to the effect table in appendix A.1.1 we see that the effect is negative. In other words, the rate of success when L is low is greater than that when it is high. S also exhibits significant positive effect. Clearly our efforts to limit steep fence angles are successful for the triangle part. Given that uncertainty in centre of mass C significantly degrades results, we see that plans are sensitive to inaccuracy in this measured feature. Uncertainties in friction and vertex location show less significant effects, probably because variation in centre of mass is very much like perturbing all vertices simultaneously, thus resulting in greater impact on outcome.

For the significant interactions, the most significant effect is SK which is negative. In this case vertex compensation K has a large negative effect when steepness minimization S is used and a strong positive effect when S is not used. Since K has insignificant effect and is a qualitative factor we will simply set it low for future trials. We see significant interactions for L with D, P and S. For LP and LS the effects are positive, but we clearly want to set L low because of its individual negative effect and S high because of its positive effect. In fact, when L is low the effect of S is small so that setting S high and L low has little detrimental effect. In the case of DL we see a negative effect supported by the negative effect of L and the insignificant, but also negative effect of D.

Sensitivity for Triangle Push Planner

The nonqualitative but significant effect demonstrated by our experiment on the push planner for the triangle is that of the fence angle angle limitation L. We also see a significant interaction between L and D however. Rather than attempting to optimize L alone, we performed a sensitivity analysis in D and L about their low settings. Because we are interested in optimizing our planner for the real world, simulations were run

with perturbations in part features (i.e. F, C, V high). The results of this search in $D \in (25, 35)$ and $L \in (25, 35)$ are illustrated in figure 6.4. From the figure we observe

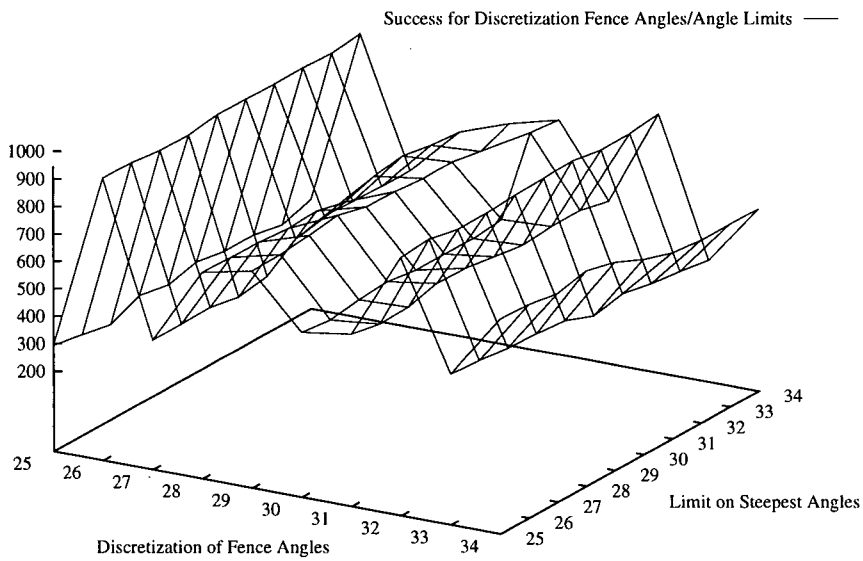


Figure 6.4: Plot of orienting success versus fence angle discretization and angle limits for triangle part with push planner.

that success rate oscillates with fence angle discretizations D. For the region under consideration the best results are at $D = 26$.

A search along the $D = 26$ contour should yield the optimal L value for our triangle push planner. The result of such a search is shown in figure 6.5. The value $L = 35$ was chosen as the best value for the planner since it was among the high success values and not too near the dropoff at $L = 39$. The planner parameterization chosen as "best" was that with steepness S, and pushdistance P high, and $D = 26$ and $L = 35$.

Using this best planner a plan was generated for the triangle (goal $\theta_g = \frac{\pi}{4}$). This plan was then tested under various noise conditions for the vertex, centre of mass and friction measurements. Figure 6.6 summarizes the result of adding normally distributed perturbations to the indicated aspects of the part model. The standard deviation of these perturbations increased as the indicated proportion of the maximum radius from

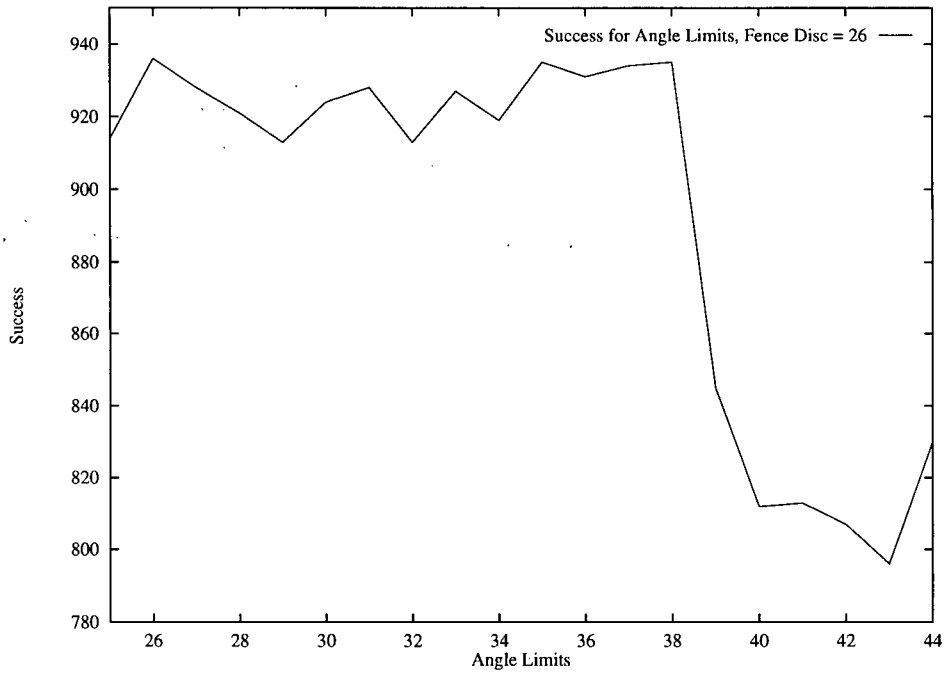


Figure 6.5: Plot of triangle orienting success with push planner (out of 1000) for various angle limits given discretization of 26.

the centre of mass to the vertices for the geometric features. For friction, the standard deviation was the indicated proportion of the nominal value.

6.3.2 "L" Push Planner Results

Again we examine our tabulated ANOVA (table 6.6) to determine the significant effects for the "L" part. The effect L is again the largest effect and is negative. For the "L" the D effect has increased to a significant negative effect and the interaction of the two DL is negative. This suggests another search in D and L to optimize our choice of these parameters.

Other significant effects include the negative effect of noise in the centre of mass, and the negative effect of N minimizing the number of pushes in a plan. Although LS is a positive effect we again will choose to set L low because of its individual effect and interaction with D. The negative effect of S while L is low is again small. The effect of P with L low was negative so we set P low. Examining the negative effect for SK

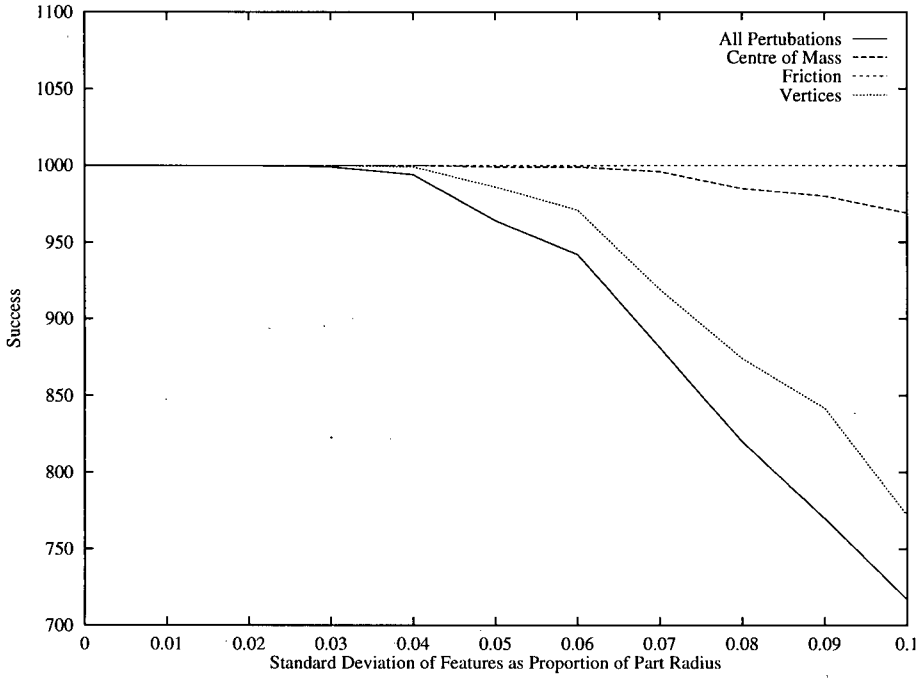


Figure 6.6: The plots in the above graph represent the successes in orienting the triangle using the push planner, out of 1000 trials, given a normally distributed perturbation of model features with standard deviation indicated as fraction of the maximum distance from the centre of mass to a vertex. Friction μ perturbations had standard deviation equal to the proportion of the nominal friction value. Perturbation of vertices, Centre of mass and μ were tested separately.

ANOVA - Success Rate for "L"					
Source		dof	Sum of Squares	Mean Squares	F
	F	1	0.166842	0.166842	1.081314
	C	1	0.445341	0.445341	2.886274
	V	1	0.075321	0.075321	0.488159
	D	1	1.010810	1.010810	6.551108
	L	1	2.454696	2.454696	15.908996
	S	1	0.270991	0.270991	1.756303
	P	1	0.029939	0.029939	0.194039
	N	1	0.442584	0.442584	2.868407
	A	1	0.058166	0.058166	0.376976
	B	1	0.038863	0.038863	0.251871
	K	1	0.001625	0.001625	0.010529
	DL	1	0.492794	0.492794	3.193819
	LS	1	1.231358	1.231358	7.980487
	LP	1	0.499915	0.499915	3.239974
	SP	1	0.609442	0.609442	3.949822
	SK	1	1.210644	1.210644	7.846240
Error	ε	175	27.001814	0.154296	1.000000
Total	T	255	36.041145	233.584317	1.000000

Table 6.6: Analysis of Variance summary for for main factor effects and significant 2 factor interactions for proportion of success for "L" part for the push planner.

ANOVA - Planning Time for "L"					
Source		dof	Sum of Squares	Mean Squares	F
	F	1	656.854184	656.854184	0.023022
	C	1	3630.313546	3630.313546	0.127236
	V	1	53883.950017	53883.950017	1.888534
	D	1	3548043.140625	3548043.140625	124.352439
	L	1	2114116.000000	2114116.000000	74.095909
	S	1	6661.960434	6661.960434	0.233490
	P	1	78328.849484	78328.849484	2.745283
	N	1	89642.855017	89642.855017	3.141819
	A	1	189174.253403	189174.253403	6.630212
	B	1	516.804444	516.804444	0.018113
	K	1	37293.242296	37293.242296	1.307060
	FC	1	111347.347656	111347.347656	3.902521
	DL	1	1908985.506984	1908985.506984	66.906459
	DN	1	88619.088025	88619.088025	3.105937
	LN	1	100115.551775	100115.551775	3.508867
	DA	1	169604.978477	169604.978477	5.944345
	LA	1	108075.192713	108075.192713	3.787838
	PK	1	119858.767539	119858.767539	4.200831
	BK	1	137319.654444	137319.654444	4.812803
Error	ε	172	4907530.760911	28532.155587	1.000000
Total	T	255	13773405.071975	482.732720	1.000000

Table 6.7: Analysis of Variance summary for for main factor effects and significant 2 factor interactions for planning time for "L" part.

suggests a strong negative effect of K when S is high, so we set K low.

Sensitivity for "L" Push Planner

Again the quantitative factors of interest are D and L at their low settings and we perform a search over $D \in (25, 35)$ and $L \in (25, 35)$ with the noise conditions F, C and V high, the results of which are illustrated in figure 6.7. Examining the graph we again see the

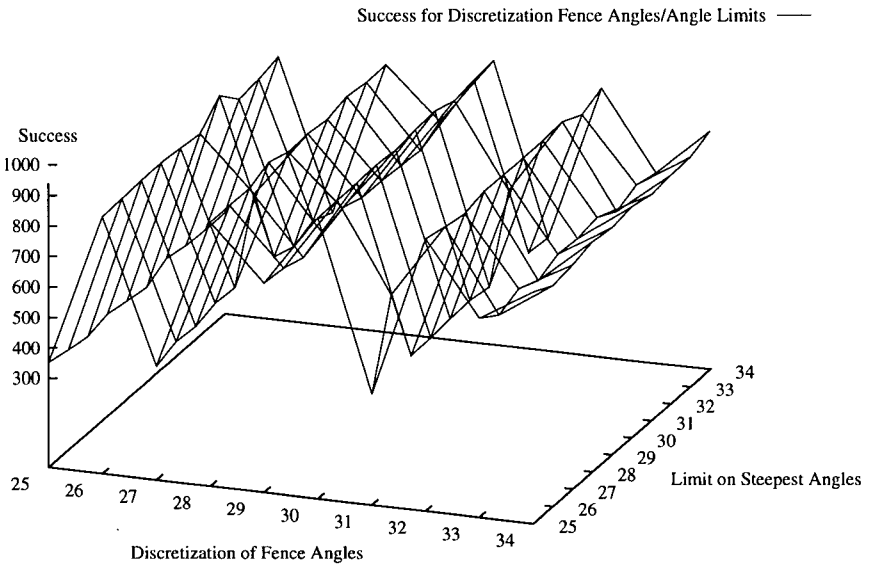


Figure 6.7: Plot of orienting success versus fence angle discretization and angle limits.

oscillation effect with respect to D. For the "L" however, the best value is $D = 30$, and in the same plot we can locate the peak at $L = 31$. We can then select the best planner options as steepness minimization S high and all other parameters low, with $D = 30$ and $L = 31$.

Using this best planner a plan was generated for the "L" (goal $\theta_g = \frac{\pi}{4}$). This plan was then tested under various noise conditions for the vertex, centre of mass and friction measurements as described for the triangle. Figure 6.8 summarizes the result of adding normally distributed perturbations to the indicated aspects of the part model.

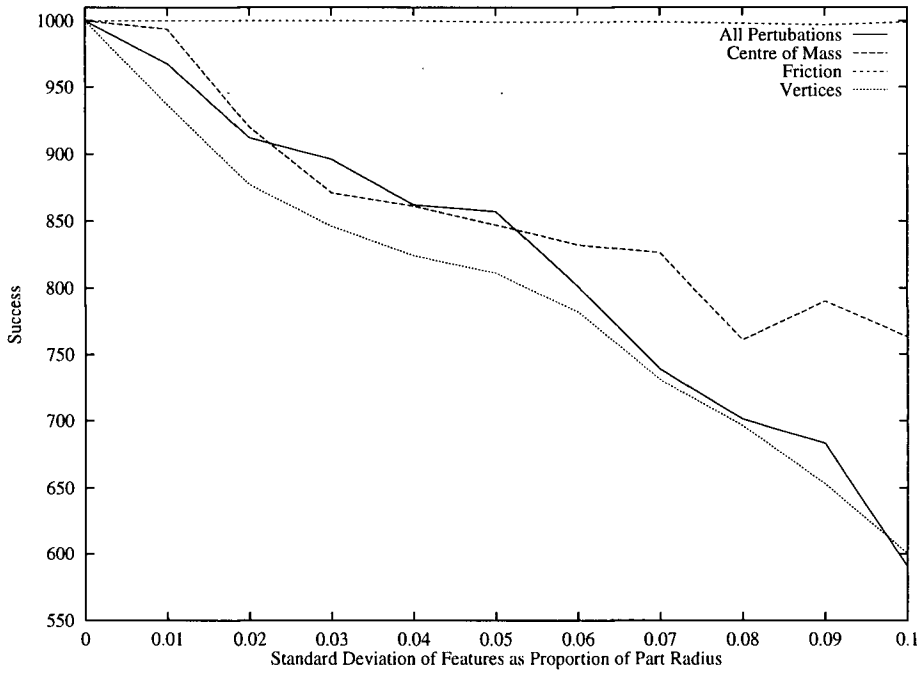


Figure 6.8: The plots in the above graph represent the successes in orienting the "L" using the push planner, out of 1000 trials, given a normally distributed perturbation of model features with standard deviation indicated as fraction of the maximum distance from the centre of mass to a vertex. Friction μ perturbations had standard deviation equal to the proportion of the nominal friction value. Perturbation of vertices, Centre of mass and μ were tested separately.

ANOVA - Success Rate for "G"					
Source		dof	Sum of Squares	Mean Squares	F
	F	1	0.285746	0.285746	4.243685
	C	1	0.936064	0.936064	13.901728
	V	1	4.854962	4.854962	72.102274
	D	1	0.336838	0.336838	5.002460
	L	1	0.153316	0.153316	2.276939
	S	1	1.959691	1.959691	29.103864
	P	1	0.010225	0.010225	0.151860
	N	1	0.000151	0.000151	0.002237
	A	1	0.003017	0.003017	0.044806
	B	1	0.012117	0.012117	0.179959
	K	1	0.034409	0.034409	0.511024
	FV	1	0.215700	0.215700	3.203419
	CV	1	0.985445	0.985445	14.635093
	VL	1	0.183693	0.183693	2.728065
	VS	1	1.028707	1.028707	15.277592
	LS	1	0.402762	0.402762	5.981522
	FP	1	0.284569	0.284569	4.226205
	DK	1	0.232587	0.232587	3.454211
	SK	1	0.377889	0.377889	5.612131
	NK	1	0.196092	0.196092	2.912214
Error	ε	171	11.514180	0.067334	1.000000
Total	T	255	24.008162	356.551289	1.000000

Table 6.8: Analysis of Variance summary for for main factor effects and significant 2 factor interactions for proportion of success for "G" part for the push planner.

6.3.3 "G" Push Planner Results

The most noticeable difference between the "G" and the other parts is the extreme sensitivity to noise in part features (table 6.8). All of the noise factors F, C and V have significant negative effects and they interact with each other and with other factors in significant ways. The oddest effect is perhaps that noise in the friction measurement is aggravated by minimizing push distance (interaction FP). Perhaps shorter push distances imply steeper fence angles and, on average, these are more likely to depend critically on friction vectors.

ANOVA - Planning Time for "G"					
Source		dof	Sum of Squares	Mean Squares	F
	F	1	7.568230	7.568230	0.002215
	C	1	1066.906954	1066.906954	0.312280
	V	1	2086.110470	2086.110470	0.610598
	D	1	700326.639715	700326.639715	204.983429
	L	1	290780.451650	290780.451650	85.110534
	S	1	27466.501954	27466.501954	8.039360
	P	1	58172.917588	58172.917588	17.027032
	N	1	423.630730	423.630730	0.123995
	A	1	59838.842475	59838.842475	17.514643
	B	1	2799.181512	2799.181512	0.819312
	K	1	22911.552432	22911.552432	6.706140
	FC	1	16672.458604	16672.458604	4.879977
	DL	1	251321.533517	251321.533517	73.561031
	DS	1	26158.008057	26158.008057	7.656368
	LS	1	20533.397319	20533.397319	6.010062
	DP	1	51746.297354	51746.297354	15.145980
	LP	1	17162.364610	17162.364610	5.023371
	SP	1	11683.132840	11683.132840	3.419617
	DA	1	54510.089220	54510.089220	15.954934
	LA	1	27550.812744	27550.812744	8.064037
	DK	1	20508.328387	20508.328387	6.002724
	LK	1	15717.062293	15717.062293	4.600335
	AK	1	16126.089715	16126.089715	4.720056
Error	ε	168	573972.617630	3416.503676	1.000000
Total	T	255	2269542.495997	664.288030	1.000000

Table 6.9: Analysis of Variance summary for for main factor effects and significant 2 factor interactions for planning time for "G" part for the push planner.

The strongest non-noise effect is steepness minimization S which is positive. D again is a significant negative effect and L, while not above the significance test is also negative.

The positive interaction VL merely tells us that L has a negative effect for the low noise condition. The negative effect of VS indicates S's positive effect at low noise. For LS, detailed examination shows that the effect of S is positive for both high and low levels of L, so we chose to set L low because of its moderately high individual effect. Although K might have a small positive effect at low N and D settings, it imposes a negative effect at high S, and since S must be high K was set low.

Sensitivity for "G" Push Planner

In part because the other parts demonstrated significant interaction in D and L, we performed our optimization for D over $D \in (25, 35)$ and $L \in (25, 35)$, resulting in the surface illustrated in figure 6.9. Clearly push orienting with a success rate of about 10%

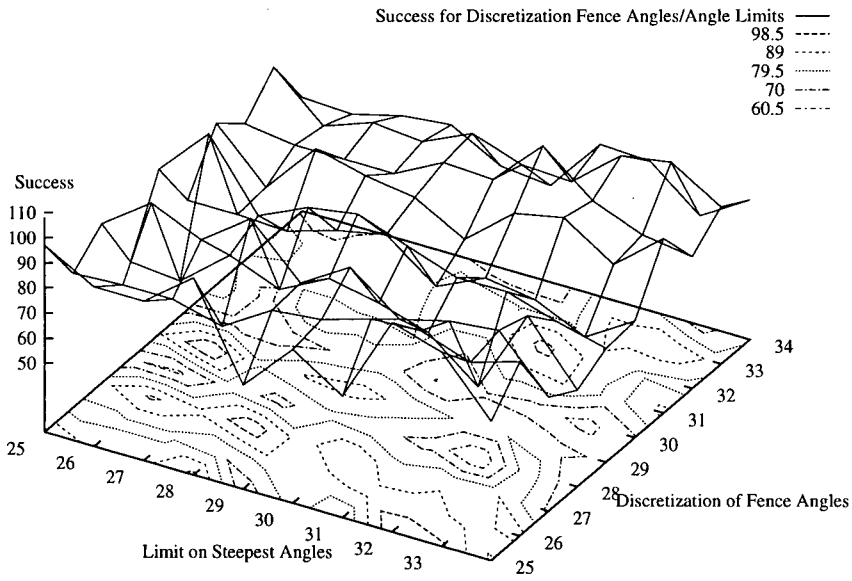


Figure 6.9: Plot of orienting success versus fence angle discretization and angle limits for "G" and push planner.

is unsatisfactory, but upon examination of the outcomes in appendix A.3.1, we can see

that any treatment under which noise was applied achieved very poor success. The "G" with 20 vertices has too many points where it can fail in any given push since several alignments occur for each push action.

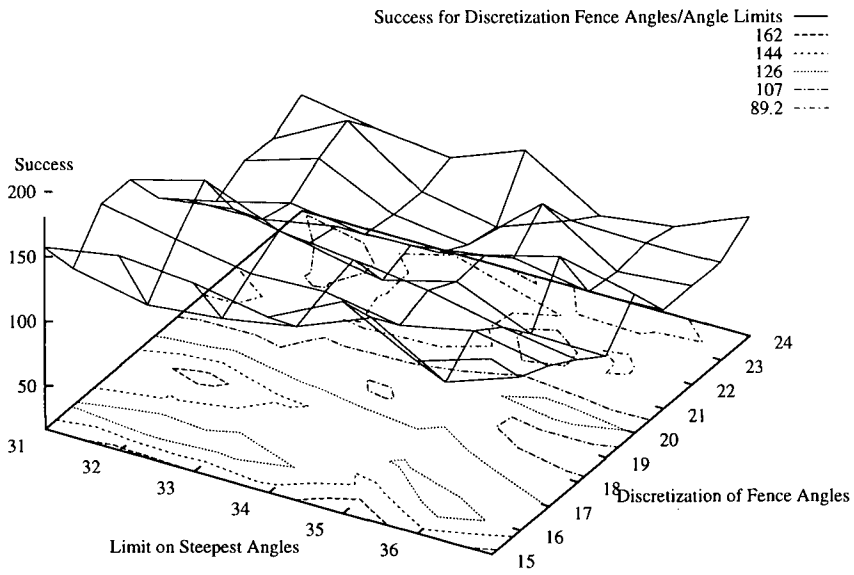


Figure 6.10: Extended plot of orienting success versus fence angle discretization and angle limits for "G" and push planner.

We attempted to proceed with the search in D and L with the results indicated in figure 6.10, but still failed to achieve better than a 20% success rate.

Finally in order to test plan robustness we chose one of the original treatment plans with S and P high (but without noise factors) which achieved perfect success. The results are shown in figure 6.11. The pronounced sensitivity of the "G" push orienter to noise is demonstrated by the fact that the rate of success falls below 10% with perturbation of the vertices with standard deviation of about 2mm.

6.4 Vision and Manipulation — Parameters and Outcomes

Table 6.10 indicates the actual values assigned to the parameters of the vision system for the treatments specified for our fractional factorial experiment. Camera position M,

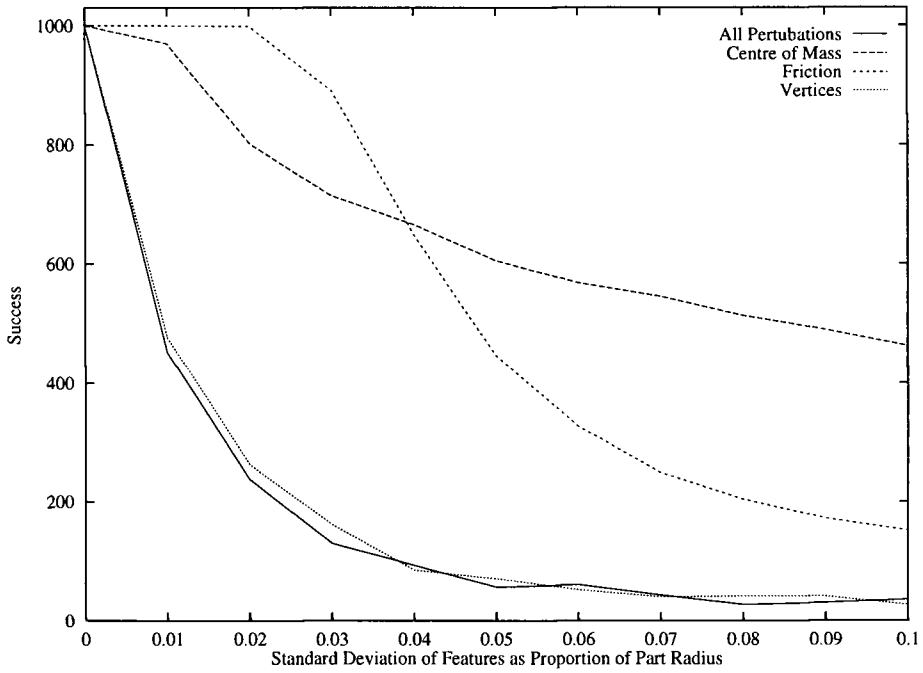


Figure 6.11: The plots in the above graph represent the probability of successfully orienting the "G" using the push planner, given a normally distributed perturbation of model features with standard deviation indicated as fraction of the maximum dimension of the part. Friction μ perturbations had standard deviation equal to the proportion of the nominal friction value. Perturbation of vertices, Centre of mass and μ were tested separately.

Parameter Levels				
Parameter		Label	High Value	Low Value
Camera Position		M	tripod	fixed on robot frame
Disparity Range		D	18	8
Sigma		S	1.2 (8×8)	0.645 (4×4)
Calib. Accuracy		C	100-180 points	30-50 points
Warp Accuracy		A	12 regions	2 regions
Lighting		L	Special lighting ($2 \times 150W$)	Standard room lighting
Z Constraint		Z	$\pm 5cm$	$\pm 20cm$
Discretization	T/L - Rho	P	20	8
	G - 2π		360	60
Window Size	T/L - Orient.	Q	10	4
	G - Gap		20	8

Table 6.10: Parameter settings for vision factorial experiment. Camera positions are either fixed to the robot frame at the same level as the robot base, or closer to the work surface, but farther apart, on a pair of tripods. T, L and G are used to indicate the triangle, "L" and "G" parts for factors P and Q.

calibration accuracy C and lighting L have been treated as qualitative factors in this experiment. See appendix A.3.2 for tabulated treatments applied and probability of success outcomes observed.

Tables 6.11 through 6.16 contain the analysis of variance computations for the proportion of success and time required for visual part localization (in seconds) respectively, for the three parts under consideration. All main effect values are indicated but two factor interactions with F value less than $F_\alpha = 2.79$ for $\alpha = 0.1$, have been pooled. We are primarily concerned with rate of success, but again we include timing ANOVA tables as an empirical version of how various parameter settings contribute to vision system complexity.

6.4.1 Triangle Vision-based Orienter Results

The triangle localization using vision performed rather poorly, with a success rate of about $15 - 18\%$. This may have been because the system was more closely tuned to the

ANOVA - Success Rate for Triangle					
Source		dof	Sum of Squares	Mean Squares	F
	M	1	0.000005	0.000005	0.005005
	D	1	0.000912	0.000912	0.934771
	S	1	0.000001	0.000001	0.000556
	C	1	0.000044	0.000044	0.045043
	A	1	0.000825	0.000825	0.845798
	L	1	0.005756	0.005756	5.899456
	Z	1	0.001641	0.001641	1.682143
	P	1	0.007683	0.007683	7.874653
	Q	1	0.002153	0.002153	2.207083
	CA	1	0.003560	0.003560	3.648443
	ML	1	0.010786	0.010786	11.055433
Error	ε	73	0.071222	0.000976	1.000000
Total	T	127	0.104587	107.198384	1.000000

Table 6.11: Analysis of Variance summary for for main factor effects and significant 2 factor interactions for proportion of success for vision orienter for triangle part.

"L", or because of the low contrast between the grey triangle and the blue work surface.

Nonetheless the effects which test significant for the triangle are a negative effect for lighting L, and a positive effect for P the discretization of the ρ range for the Hough transform (table 6.11). The interaction CA is relatively small and the detailed calculation suggests that if A is low C should be low and if C is high A should be high. Since they are individually insignificant and they are both expensive calculations we will set them low.

ML is a significant interaction of camera position and lighting. Our experience with the tripod camera position (M high) was quite poor. The viewpoints we chose, although closer to the work surface were more oblique and caused the parts to obscure themselves. The detailed calculation for the effect suggests that for M low, L should be low and this was the setting we chose.

ANOVA - Extraction Time for Triangle					
Source		dof	Sum of Squares	Mean Squares	F
	M	1	2605.899410	2605.899410	2051.788812
	D	1	0.999451	0.999451	0.786931
	S	1	1.968038	1.968038	1.549561
	C	1	0.801767	0.801767	0.631282
	A	1	10.533195	10.533195	8.293448
	L	1	52.116861	52.116861	41.034889
	Z	1	10.278770	10.278770	8.093123
	P	1	311.473844	311.473844	245.242984
	Q	1	308.207424	308.207424	242.671126
	MA	1	14.860634	14.860634	11.700713
	SL	1	5.317925	5.317925	4.187137
	MZ	1	8.240361	8.240361	6.488155
	AZ	1	18.276036	18.276036	14.389875
	MQ	1	6.761198	6.761198	5.323517
	LQ	1	4.201225	4.201225	3.307889
	ZQ	1	4.818316	4.818316	3.793764
	PQ	1	72.531211	72.531211	57.108393
Error	ε	67	85.094167	1.270062	1.000000
Total	T	127	3522.379833	2773.391600	1.000000

Table 6.12: Analysis of Variance summary for for main factor effects and significant 2 factor interactions for extraction time for triangle part.

Sensitivity for Triangle Vision System

For optimizing the vision system for the triangle we chose the parameter settings with all factors low except P the ρ discretization parameter. For P we did a search around its high setting to determine the best value. The results of this search are presented

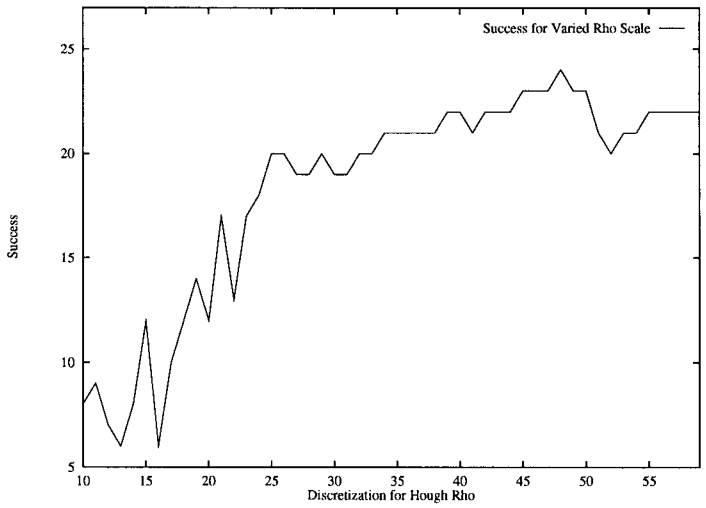


Figure 6.12: Plot of vision system success for triangle part versus values for ρ discretization for local Hough Transform.

in figure 6.12 where we observe the best P value is 48. The corresponding growth in extraction time with D is plotted in figure 6.13.

6.4.2 "L" Vision-based Orienter Results

The significant main effects for the vision system for "L" are M, D, S and P. M is the large negative effect resulting from the poor tripod position (table 6.13). D the disparity range, S the σ value, and P the size of the orientation template are all positive effects.

The pronounced effects for M, S, and P allow us to set them independently of MS and MP. Examining LP and SL, we see that SL is positive and thus the very significant effect S votes for L positive as does the basic (though insignificant) L effect. The less significant interaction LP will not be considered in choosing L high. The overall positive effect of Q, given other parameter settings, is small, and high Q increases computation time so Q will be set low.

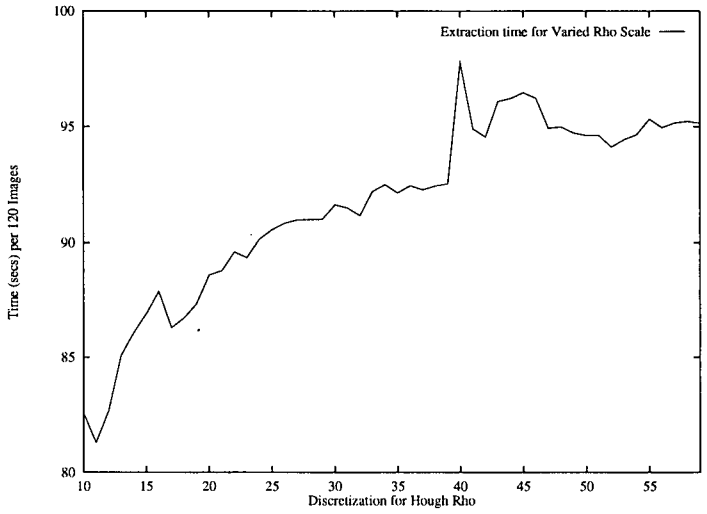


Figure 6.13: Plot of time for triangle part visual localization (total for 120 trials) versus values for ρ discretization for local Hough Transform.

ANOVA - Success Rate for "L"					
Source		dof	Sum of Squares	Mean Squares	F
	M	1	0.275808	0.275808	117.403444
	D	1	0.010786	0.010786	4.591346
	S	1	0.526381	0.526381	224.065112
	C	1	0.000396	0.000396	0.168356
	A	1	0.002153	0.002153	0.916606
	L	1	0.003738	0.003738	1.590955
	Z	1	0.005317	0.005317	2.263457
	P	1	0.195964	0.195964	83.416262
	Q	1	0.000287	0.000287	0.122168
	MS	1	0.011724	0.011724	4.990413
	SL	1	0.045942	0.045942	19.556348
	MP	1	0.018568	0.018568	7.903969
	LP	1	0.008477	0.008477	3.608459
	MQ	1	0.020630	0.020630	8.781546
	LQ	1	0.014063	0.014063	5.986232
	PQ	1	0.015495	0.015495	6.595917
Error	ε	68	0.159748	0.002349	1.000000
Total	T	127	1.315477	559.960591	1.000000

Table 6.13: Analysis of Variance summary for for main factor effects and significant 2 factor interactions for proportion of success for vision orienter for "L" part.

ANOVA - Extraction Time for "L"					
Source		dof	Sum of Squares	Mean Squares	F
	M	1	1297.984661	1297.984661	916.649949
	D	1	3.314353	3.314353	2.340630
	S	1	65.833441	65.833441	46.492244
	C	1	0.003656	0.003656	0.002582
	A	1	0.130678	0.130678	0.092286
	L	1	8.391034	8.391034	5.925833
	Z	1	18.921739	18.921739	13.362724
	P	1	157.838372	157.838372	111.467062
	Q	1	160.624250	160.624250	113.434477
	MA	1	6.859214	6.859214	4.844046
	SL	1	76.713626	76.713626	54.175942
	MZ	1	8.799960	8.799960	6.214621
	MP	1	32.892755	32.892755	23.229198
	ZP	1	5.575727	5.575727	3.937635
Error	ε	70	99.120636	1.416009	1.000000
Total	T	127	1943.004101	1372.169229	1.000000

Table 6.14: Analysis of Variance summary for for main factor effects and significant 2 factor interactions for extraction time for "L" part.

Sensitivity for "L" Vision System

The parameter settings we have settled upon then are D, S, L and P high, and the other factors low. We can optimize on D the number of disparities, and P the ρ discretization factor, independently since our analysis indicates no significant interaction between them.

Figure 6.14 illustrates our search over $D \in (15, 35)$ for the best number of disparities. For the current set of coefficients the DATACUBE system could not search more than 35 disparity levels.

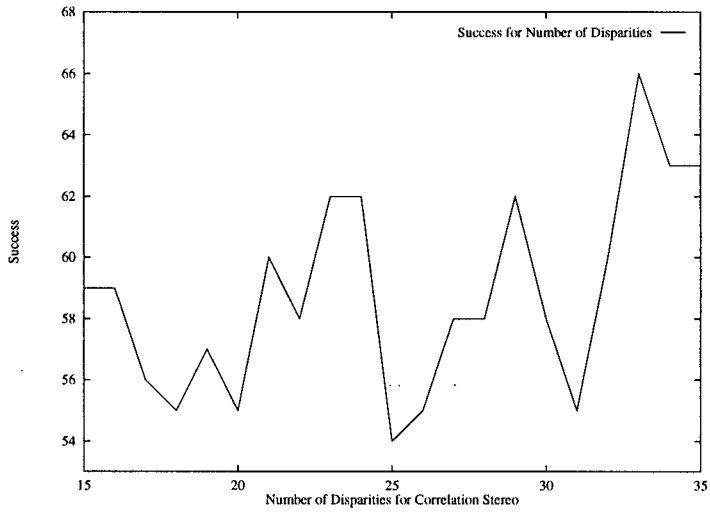


Figure 6.14: Plot of success for "L" part versus number of disparities for correlation stereo.

The best observed number of successes out of 120 trials was 66 for 33 disparities. It is unclear why there were such erratic swings in the number of successes (or extraction time in figure 6.15) as the number of disparities increased, unless more disparities caused greater fragmentation of 3-D line segments.

Having chosen the best observed disparity level $D = 33$, we will now search for the best possible ρ discretization value. The search over $P \in (10, 60)$ is illustrated in figure 6.16 The best success rate (72 of 120) was found in the range $P \in (45, 47)$ so we set $P = 45$. Figure 6.17 illustrates the increase in extraction costs for the range of P and $D = 33$.

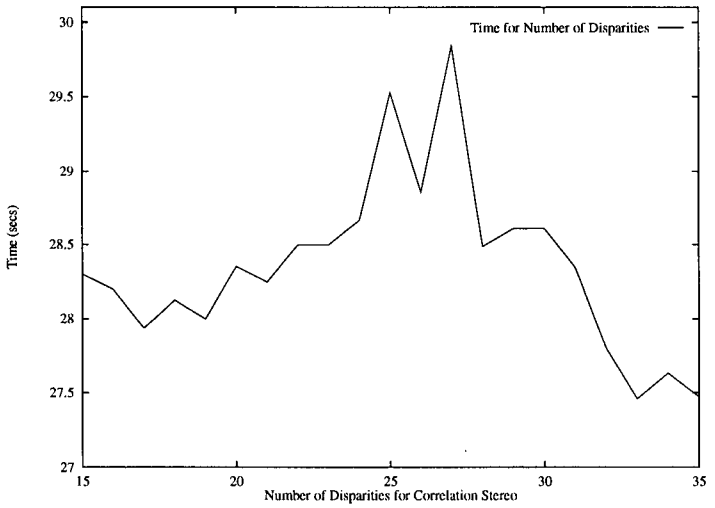


Figure 6.15: Plot of time for "L" part localization (for 120 trials) versus number of disparities for correlation stereo.

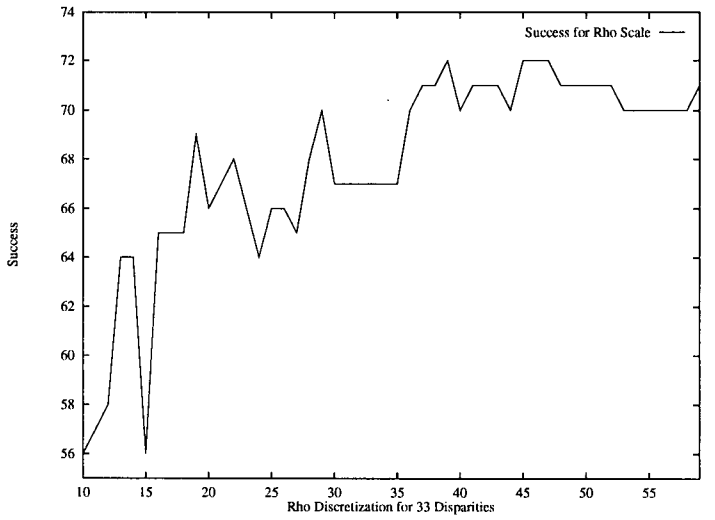


Figure 6.16: Plot of success for "L" part versus rho discretization for local Hough Transform for 33 disparities.

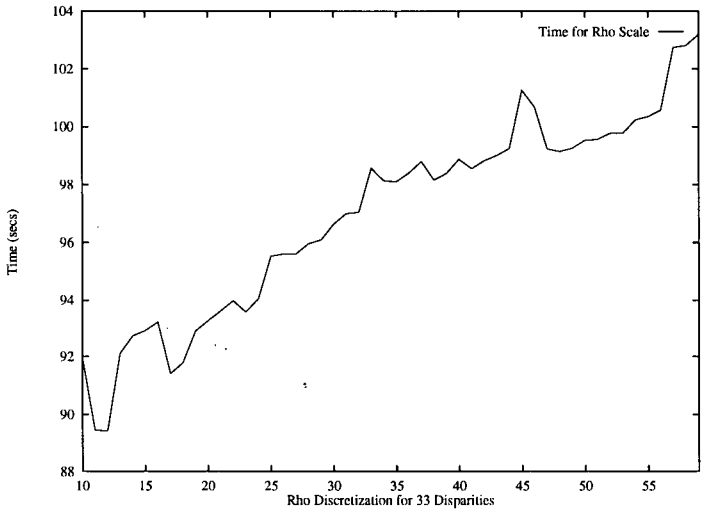


Figure 6.17: Plot of time for "L" part visual localization (for 120 trials) versus rho discretization for local Hough Transform for 33 disparities.

6.4.3 "G" Vision-based Orienter Results

The "G" localization strategy is essentially a 2-D process of searching for a gap in an oval shape, it therefore performs very badly under the tripod camera positions (table 6.15). This is illustrated by the extreme negative effect of the camera factor M. For the "G" part the system prefers the smaller σ S and structured lighting L. P, in this case the discretization of 2π used to classify depth points with respect to the centroid, is a large positive effect. For the "G", Q is the number of histogram bins in the window over which a minimum number of hits is sought, and it has a large negative effect. PQ has a large positive effect, but we chose to look at this interaction by searching around high P and low Q as a result of their individual effects. Since all interactions including M tend to have near zero successes for M high, effects ML, MP and MQ simply reflect the main effects of L, P, and Q.

Sensitivity for "G" Vision System

The selected parameter settings are L high, P high and S low. In order to save time we used a set of previously generated disparity images which also had D high. This is not inconsistent since D did have a small positive effect, although since it is insignificant

ANOVA - Success Rate for "G"					
Source		dof	Sum of Squares	Mean Squares	F
	M	1	6.261309	6.261309	432.408151
	D	1	0.010546	0.010546	0.728344
	S	1	0.055840	0.055840	3.856329
	C	1	0.000983	0.000983	0.067915
	A	1	0.005060	0.005060	0.349452
	L	1	0.079761	0.079761	5.508290
	Z	1	0.001401	0.001401	0.096759
	P	1	0.414544	0.414544	28.628584
	Q	1	0.289213	0.289213	19.973149
	ML	1	0.083613	0.083613	5.774333
	MP	1	0.403014	0.403014	27.832254
	MQ	1	0.279595	0.279595	19.308928
	PQ	1	0.723223	0.723223	49.945996
Error	ε	71	1.028086	0.014480	1.000000
Total	T	127	9.636188	665.478486	1.000000

Table 6.15: Analysis of Variance summary for for main factor effects and significant 2 factor interactions for proportion of success for vision orienter for "G" part.

ANOVA - Extraction Time for "G"					
Source		dof	Sum of Squares	Mean Squares	F
	M	1	4573.633186	4573.633186	159.179321
	D	1	3.597744	3.597744	0.125215
	S	1	1.995649	1.995649	0.069456
	C	1	13.757258	13.757258	0.478803
	A	1	3.760284	3.760284	0.130872
	L	1	386.164376	386.164376	13.439946
	Z	1	229.452234	229.452234	7.985785
	P	1	39.623771	39.623771	1.379054
	Q	1	241.778629	241.778629	8.414789
	DS	1	112.354725	112.354725	3.910359
	ML	1	208.614438	208.614438	7.260553
	ZP	1	268.407018	268.407018	9.341555
	ZQ	1	446.944817	446.944817	15.555330
	PQ	1	1791.573802	1791.573802	62.353383
Error	ε	70	2011.280873	28.732584	1.000000
Total	T	127	10332.938803	359.624419	1.000000

Table 6.16: Analysis of Variance summary for for main factor effects and significant 2 factor interactions for extraction time for "G" part.

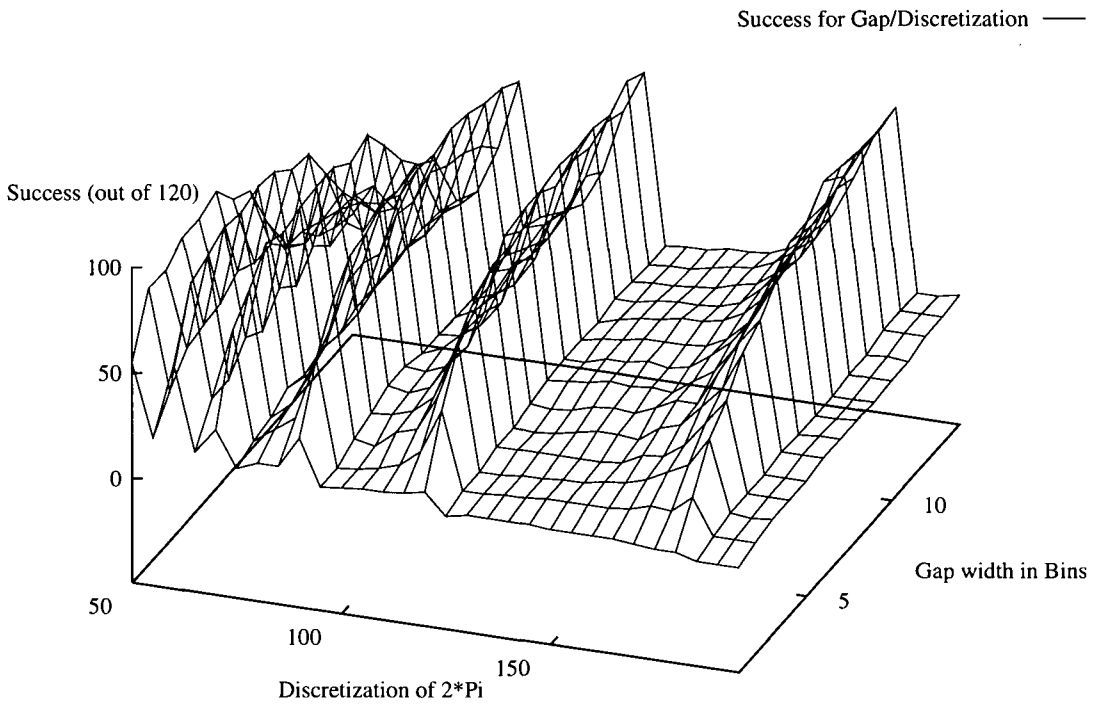


Figure 6.18: Plotting success rates for ranges of the gap window versus discretization values for 2π we can observe a pronounced cyclical trend, for the "G" vision system.

it does not matter how we set it. Clearly from table 6.15 P and Q have strong effects and significant interactions. We did a search about P's high value and Q's low value to produce the surface illustrated in figure 6.18. The plot demonstrates that there is an obvious cyclical relationship between gap size and discretization. We performed a further search in the peak region of each contour to produce the detailed plots in figure 6.19. No doubt this recurring relationship is related to the fact that the gap in the actual part has a width of about 50 degrees. The system performs better in ranges of gap and discretization related to this 50 : 360 ratio. We found two scenarios where 100 successes out of 120 trials were achieved: at $P = 120$ and $Q = 13$ and at $P = 72$ and $Q = 6$. For our choice of a best parameter set, we set the parameters to the smaller values because

of the cost associated with the search for the minimum gap window.

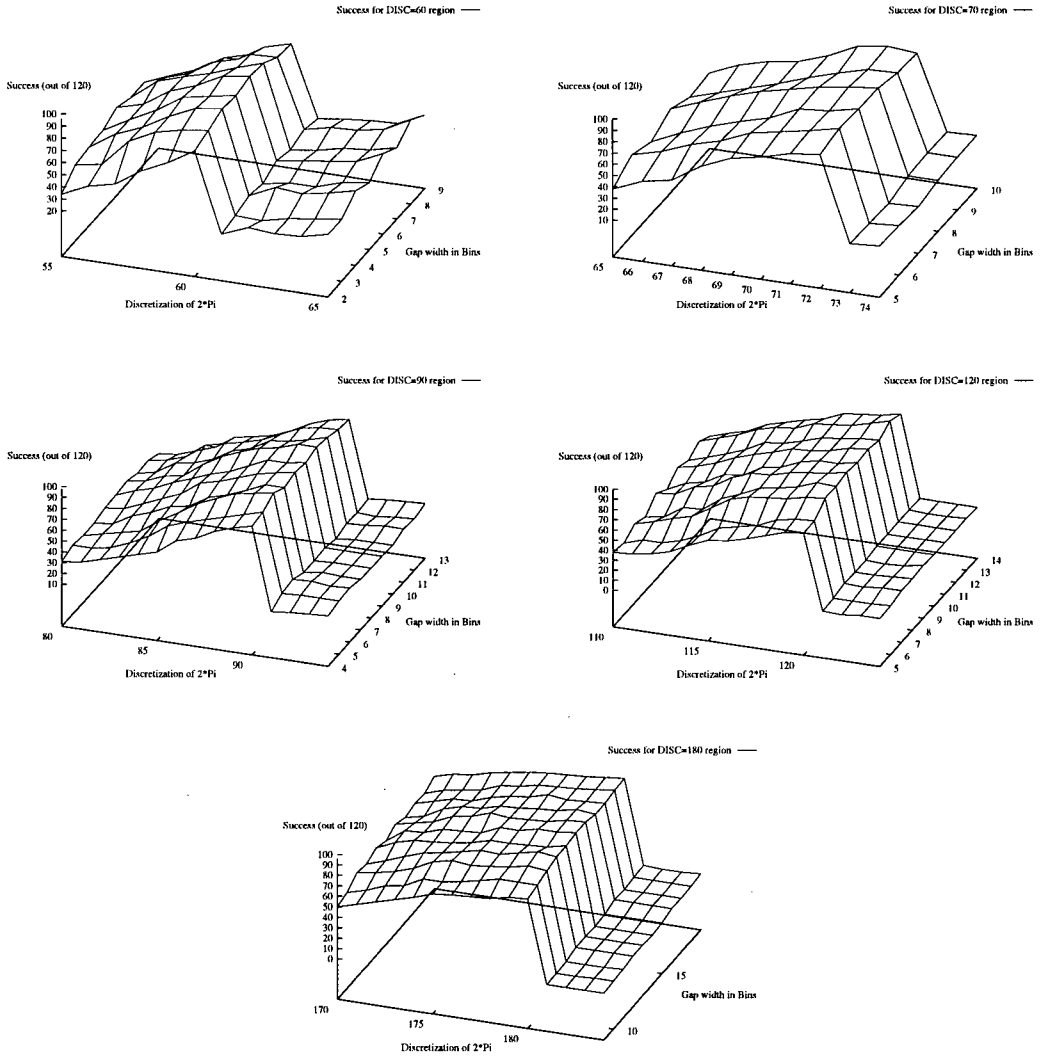


Figure 6.19: The plots above are detailed regions of the full plot of figure 6.18, in the regions disc=60, 70, 90, 120, and 180, where peaks occurred in the original plot.

Chapter 7

Analysis of Results

In this chapter we review the results, collected in chapter 6, for the factorial experiments and sensitivity analysis used to determine parameter values for our two part orienting systems. We discuss the surprising variety of optimal values occurring for different parts; part information clearly manifests itself in these part specific parameterizations. Finally we review how ε-complexity can be applied to these systems as a cost/performance metric.

7.1 Optimization of Information for Tasks

In chapter 6 we demonstrated the usefulness of treating robotic systems as complex parameterizations of tasks. We used factorial experiments to identify significant parameters and interactions among parameters, and then used sensitivity analysis to set the optimal values for those significant parameters.

For the three parts studied (the triangle, "G" and "L") each of the push orienting and vision based orienting systems demonstrated different behaviours and different optimal parameters. For the push planner we saw that plans for the triangle were primarily sensitive to variations in centre of mass, and that minimizing plan steepness and eliminating a large range of steep angles from the plan search were beneficial to plan success (factors C, S and L). We examined the interaction of fence angle discretization D and L and determined optimal values for these parameters to be $D = 26$ and $L = 35$. The result is a plan relatively insensitive to friction and centre of mass variation, but sensitive to vertex perturbations with standard deviation greater than 4% of part dimension. Most of the information relevant to this task appears to be contained in parameters S, L and V.

For the "L" part under the push orienter the most significant parameters are cen-

tre of mass variation C, fence angle discretization D, the limit on steep angles L and minimization of plan length N. Again we optimized D and L to set their values at $D = 30$ and $L = 31$. The "best" plan for orienting the "L" was insensitive to friction variation, but success dropped off steadily and immediately with perturbations (with standard deviations as small as 1% of part dimension) in centre of mass and vertices.

The "G" fared badly under push orienting with extreme sensitivity to the noise parameters F, C, and V . Other significant parameters included D and S again, fence angle discretization and steepness minimization. For consistency a search for optimal D and L parameters was made, but since optimization is performed in the presence of noise; success rates were below 20% and therefore little progress could be made. A plot of the sensitivity of a "good" plan to noise showed a success rate falling to 10% for perturbations with standard deviation as small as 4% of part dimension in the vertices. The relevant information for this part and orienting method is captured by F, C, V, D and S.

The vision system also showed varied performance with the different parts. The triangle had rather poor success with the vision system, perhaps because it required the greatest positional accuracy for a successful grasp. The most significant parameters according to our analysis were the lighting conditions L, and the model parameter P controlling discretization of ρ for the local Hough transform. We performed a search on P to determine the optimal value $P = 48$.

The vision system was moderately successful with the "L". The most significant parameters were camera placement M, number of disparities D, the σ value for $\nabla^2 G$, S, and P the ρ discretization value. In the optimization process we did a search over the number of disparities searched to determine $D = 33$. Having fixed D we determined the optimal value $P = 45$. It is interesting that variation of 10% in measured part parameters can drive the push planner success rate for the "L" to the same 60% level of our optimized vision system. The vision system would be unlikely to notice such small perturbations since it uses a gross model.

For the "G" the relevant parameters are M, S, L, P and Q. Model parameters P and Q now indicate the discretization of 2π orientations and the gap width in histogram bins

for the "G" respectively. A search in P and Q demonstrated the relationship between the width of the real gap as a fraction of the circle, giving cyclical peaks in $P-Q-space$. The best values selected were $P = 72$ and $Q = 6$. This optimized parameterization achieved a success rate of 83%. In comparison, a perturbation with standard deviation of 0.5% of part dimension in vertices results in a lower success rate for the push planner. Again the vision system is unlikely to be sensitive to such small perturbations in true part shape.

Clearly each of the part orienting systems' interactions with the parts is different; reflecting different aspects of the information involved. This is true even for the basic manipulation operation of orienting a part. Although, for example, the triangle and the "L" would appear very similar under both the pushing and vision models used, analysis demonstrates that the performance of each system for each part and the parameters which exhibit significance are quite different.

The push planner has perfect performance at orienting all three parts under noise-free conditions, however even small errors or variations in the centre of mass or vertices quickly degrade success rates. As parts become more complex this brittleness with respect to world variance from the predefined model becomes more pronounced. The vision system has much higher error rates, but is never tested with perfect data. At each step (calibration, epipolar approximation, edge detection, grouping, line fitting) more noise is introduced. Nevertheless the vision system appears to be able to produce comparable performance to the push planner for the "G".

Whatever the claims for the robustness of push orienting, the results of the experiments described here reflect the problem of models that cannot easily account for the true degree of variance in the world. That is, model-based systems must account for variations in the world, and the greater the complexity of the models, as demonstrated by the "G" under pushing, the more difficult it is to account for all possible variations. For our vision systems the models were not very precise, and so we would expect them to tolerate an equivalent degree of uncertainty in the real parts.

The most significant lesson learned from this analysis is that extensive analysis of actual robotic implementations is a necessary step to understanding such systems. This

type of analysis reveals aspects of the implementation which cannot be predicted *a priori.* It also clearly defines what information is necessary for a particular implementation, and what the strengths and weaknesses of these implementations are.

7.2 Cost/performance Analysis

For our metric on system cost versus performance we draw on the notation and model of computation used in information-based complexity. Information based complexity attempts to integrate the costs of using noisy, partial, priced information in approximation tasks, into an overall complexity measure computed with respect to the error ε achieved in the approximation. Clearly we can view many tasks in sensing and robotics as attempts to approximate some ideal task function to within acceptable error limits, given partial noisy information about the world. The lines between sensing and action become blurred when we think of measurements as the result of instruments interacting with the physical world, and the areas of active vision and tactile sensing. In the following discussion we apply the an analysis based on the ε-complexity formalism to our part orienting systems as a useful descriptive tool allowing a comparison of their performance. We observe however that having attempted to stretch other formalisms such as "part entropy", "information invariants" and decision theoretic models to describe and compare our part-orienting systems, information based complexity seems the most promising option for making formal statements about the relative complexity of robotic tasks.

7.2.1 The Formalism

To describe a problem in the ε-complexity formalism we require a precise problem formulation; a specification of allowable information and a model of computation [68, 72].

Generally problems are defined with respect to a solution operator S, for some set F and normed linear space G;

$$S : F \rightarrow G.$$

The elements of F are referred to as problem elements. For our part orienting problems $F = \{p_i\}$, where p_i specifies the properties of a particular instantiation of a part including

such things as precise shape, centre of mass, and initial world location. The problem is to transform any part from the range of possible parts, in the range of possible initial positions, to a desired final position $\theta_g \in G$.

The information available to our two systems is partial and noisy. In general, information is described by

$$y = N(p),$$
$$N(p) = [L_1(p), L_2(p), ..., L_n(p)]$$

where L_i are information operations. Generally $N(p)$ is an equivalence class where some range of p values are mapped to the same y due to the limits of measurement or arithmetic precision.

An approximation U is a pair (ϕ, N) where ϕ is the computation used to approximate S.

$$\phi : N(F) \rightarrow G$$

For some p then, the absolute error for U is

$$e(U) = ||S(p) - \phi(N(p))||$$

If the absolute error criterion is being applied then to be an ε-approximation, $||S(p) - \phi(N(p))|| \leq \varepsilon, \forall p \in F$. The cost of an approximation at a point p is the sum of its information cost and its combinatoric cost.

$$cost(U, p) = cost(N, p) + cost(\phi, N(p))$$

The model of computation generally used in the ε-complexity formalism is to attribute a cost $c_1 > 0$ to each L_i and a cost of 1 to each arithmetic or comparison operation. For our purposes we will also include a constant cost c_2 for each robot arm motion required by the orienting system. This is a useful way for us to attribute the costs of each method, but is probably oversimplified for analysing the relative performance of robotic programs in detail.

Finally there are three possible cases in which we can examine complexity: worst case; average case and probabilistic. We will only concern ourselves with the worst and

average cases. For the worst case setting error and cost are defined by

$$e(U) = \sup_{p \in F} ||S(p) - \phi(N(p))||,$$
$$cost(U) = \sup_{p \in F} cost(U, p).$$

In other words, the overall error is the largest error for all parts in F, and the cost is the greatest cost for all parts in F.

For the average case setting a probability distribution μ is assumed to be known *a priori* on the set F. Error and cost are defined as follows:

$$e(U) = \sqrt{\int_F ||S(p) - \phi(N(p))||^2 \mu(df)}$$
$$cost(U) = \int_F cost(U, p) \mu(df)$$

7.2.2 Vision Orienting

The vision system used by our orienter U_1 can be described in the ε-complexity framework as using a single function evaluation $N(p) = \{L_0(p)\}$ which is the image of disparity marked edge pixels delivered by the DATACUBE system. These are essentially noisy data, which are operated on to generate an approximation of part location. The information cost per approximation then is c_1, as defined above.

The computational cost of the preprocessing for our vision system is significant. First a set of calibration data points are collected, using the robot arm as part of the measurement system. This requires about 200 robot motions to collect up to 180 calibration points. Extracting the marker on the arm from the image pair requires a search of up to $512 \cdot 480$ pixels in each of a stereo pair of images. For calibration points C and image size M, collecting the calibration data costs

$$\text{calibration cost} = C \cdot (c_2 + 2 \cdot M), \tag{7.1}$$

where c_2 is the cost of moving the arm (as defined above).

The next step in starting up the vision system is to approximate the epipolar warp. Histograms of image locations and disparities are generated by sampling a workspace region and projecting points into the images. Singular value decomposition is used to approximate the epipolar warp for successively smaller binary subdivisions of the image, until the error is sufficiently low, or the upper bound b on regions is reached. If the upper

bound is reached all of the approximations in the binary tree have been computed giving us $a = 2^{log_2(b)+1} - 1$ approximations of complexity s, computed. Finally the warp window is expanded by computing the approximated source pixel for every pixel in the left image window. For workspace sampling $S_x \cdot S_y \cdot S_z$, and image window M then, the complexity of the precomputation is

$$\text{warp approx. cost} = S_x \cdot S_y \cdot S_z + a \cdot s + M \tag{7.2}$$

Finally, considering the more complicated the extraction of the right angle (vs "G") part position requires processing $d << M$ disparity marked edge pixels, grouping them into 2-D lines and fitting 3-D lines to them. This requires an initial pass through the image window, an application of the $Q \cdot Q$ orientation template for each edgel and a second pass through the edge list for grouping. Groups of pixels g are then fit with a line in 3-D with cost f, and finally each line is compared with the others to find a perpendicular pair. The combinatoric cost for this process is

$$\text{cost}(\phi, N(p)) = M + d \cdot Q \cdot Q + d + g \cdot f + g \cdot g \tag{7.3}$$

The total cost for the part orienting system for orienting P parts in its lifetime is

$$\begin{aligned} \text{cost}(U_1) = C \cdot (c_2 + 2 \cdot M) + S_x \cdot S_y \cdot S_z + a \cdot s + M \\ + P \cdot (M + d \cdot Q \cdot Q + d + g \cdot f + g \cdot g + c_1 + 2 \cdot c_2), \end{aligned} \tag{7.4}$$

where, for each part oriented, we have information cost c_1 for receiving a disparity image, and we have added cost $2 \cdot c_2$ to represent the two motions required to pick and place the part once it is localized.

Error in the vision system comes from noise in the process $N(p)$ rather than from a lack of discrimination, and is therefore difficult to evaluate exactly. Figures 7.1 and 7.2 illustrate observed position (L_2-norm) and orientation localization error, for the vision system's optimal "G" orienting parameterization. Experimentally then, we conclude that the average case position error is $e_{avg-pos} = 0.84cm$, the average case orientation error is $e_{avg-or} = 0.048rad$, and the worst case errors are $e_{wor-pos} = 5.26cm$ and $e_{wor-or} = 0.98rad$ respectively.

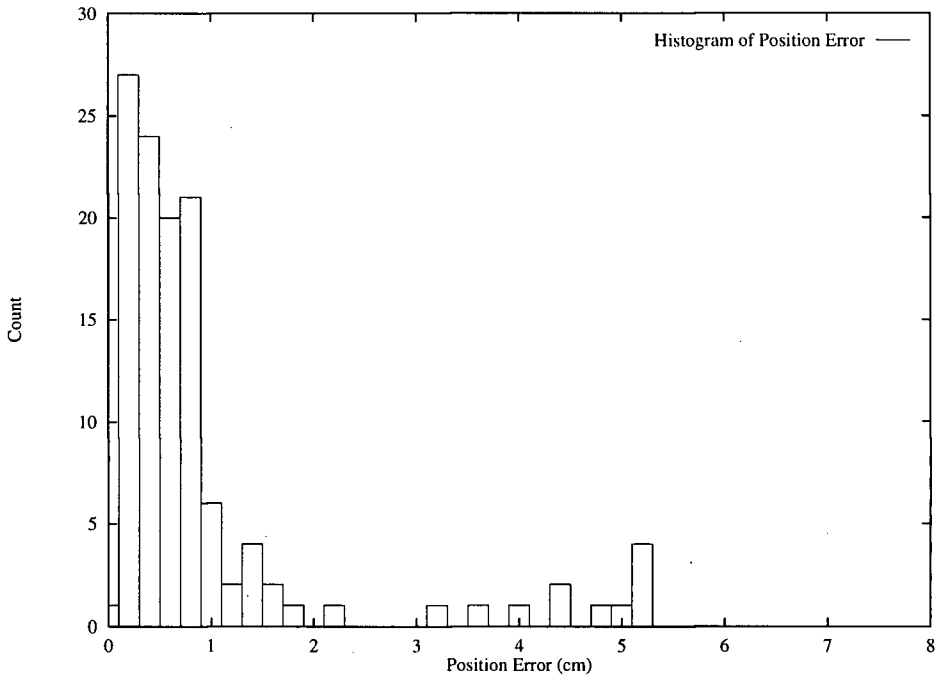


Figure 7.1: Histogram of position error (Euclidean distance) for vision system with "G" part for 120 images/part positions. The average X error was 0.55cm the average Y error 0.64cm, the maximum error was 5.26cm the minimum 0.088 cm.

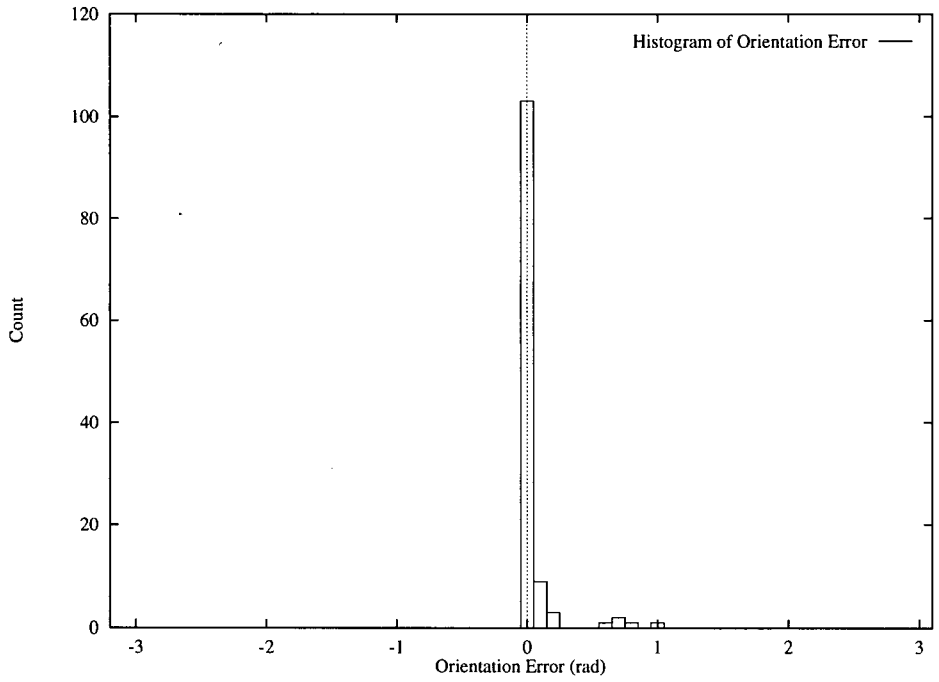


Figure 7.2: Histogram of orientation error for vision system with "G" part for 120 images/part positions. The average orientation error was 0.048 rad., the maximum error was 0.98 rad. the minimum 0.00018 rad.

7.2.3 Push Orienting

Our application of the notation of information-based complexity to the push orienting system is a stretch of the normal usage of the formalism. Clearly, plans are computed under the assumption that knowledge of the parts to be oriented is complete and exact, very much a combinatoric approach. We argue however that the resulting sequence of pushes can be appropriately analysed in this way since inaccurate or noisy part information in turn affects its performance. Every push has a cost in terms of time, space and energy. We can view it as an information function $L(p_i)$ on each part passing through the system. The "value" returned is a *cw* or *ccw* rotation and alignment, yielding a more restricted set of possible part orientation states, much in the same way a binary search in a continuous domain progressively restricts a value to an acceptably small interval [68]. Noise or deviation from expected canonical part parameters causes the part to give erroneous "answers" to these questions posed by the pusher's fence angle. We will apply our analysis to this mechanical computation.

Let us examine $U_2 = (\phi_2, N_2)$ the push planner implementation of part orienting. Let each element of F be a vector representing a part, including the $v_i = (x, y)$ position of each part vertex, the $C = (x, y)$ position of the centre of mass, the coefficient of friction between the part and the fence f, and the initial part orientation θ_i. $p \in F$ then is an instantiation of an incoming part to the push planner. G is the space of outcome orientations for the part. S is an ideal plan which maps a part with properties p to the goal orientation θ_g.

The mapping ϕ is a push plan precomputed for some $p_0 \in F$. We assume that all the possible parts in F have vertices v_i, centre of mass C and coefficient of friction f, normally distributed with mean zero ($\mathcal{N}(0, \sigma)$), about the corresponding features of p_0. In other words consider all incoming parts to be in the same equivalence class $N(p_0)$ with p_0.

Now as we have argued above we will assume that each push is an information operation L_i. The cardinality $n(p)$ of $N(p)$ is then the same as the length of the plan ϕ. The push orienter is a fixed approximator to the orienting problem for a part p_0 and desired

outcome angle θ_g, i.e. we assign a fixed cost to each physical push operation for a fixed length plan and thus the cost of $N(p)$ is fixed.

Let $\theta_i = g(\theta_{i-1}, \alpha_i, p)$, be the decision computation which maps θ_{i-1} to θ_i given that fence angle α_i is applied to part p in state θ_{i-1}. A push sequence is then a cascade of these transitions of the form

$$\theta_f = g(g(...g(\theta_0, \alpha_0, p), ...), \alpha_f, p).$$

As we have seen in the experiments of chapter 6, parts which vary from those on which plans are based cause errors in outcome orientations because they rotate in unexpected ways at critical points (see figure 4.3). For any particular plan ϕ the overall worst case error will depend on the worst outcome θ_f of the possible bad rotations for the set of possible parts F. As above

$$\begin{aligned} e_{wor}(U) &= \sup_{p \in F} ||S(p) - \phi(N(p))||, \\ cost(U) &= \sup_{p \in F} cost(U, p). \end{aligned}$$

Similarly the average case error is defined

$$\begin{aligned} e_{avg}(U) &= \sqrt{\int_F ||S(p) - \phi(N(p))||^2 \mu(df)} \\ cost(U) &= \int_F cost(U, p) \mu(df), \end{aligned}$$

where μ is the same distribution $\mathcal{N}(0, \sigma)$ indicated for part variation. Since our push planner always executes the same plan independent of the particular p_i, the cost for both the worst and average cases is fixed at $c_2 \cdot k$ for plan of length k.

There are essentially two parts to the push planner: the precomputed planning stage which is lengthy and has dominant combinatoric complexity, and the actual push execution stage where straight line push motions are generated via spline approximation.

The initial stage of the planner sets up the configuration maps and code sets which form the basic data structure upon which the planner operates. Essentially this requires testing the outcome of each of the available discretized fence angles for each of a fixed discretization of the input part angles q. The number of fence angles is $B = \frac{\pi - 2L}{D}$, where L and D are the steepness limits and fence discretization factors tested in chapter 6. This preprocessing requires on the order of $q \cdot B$ operations. The planner performs recursive

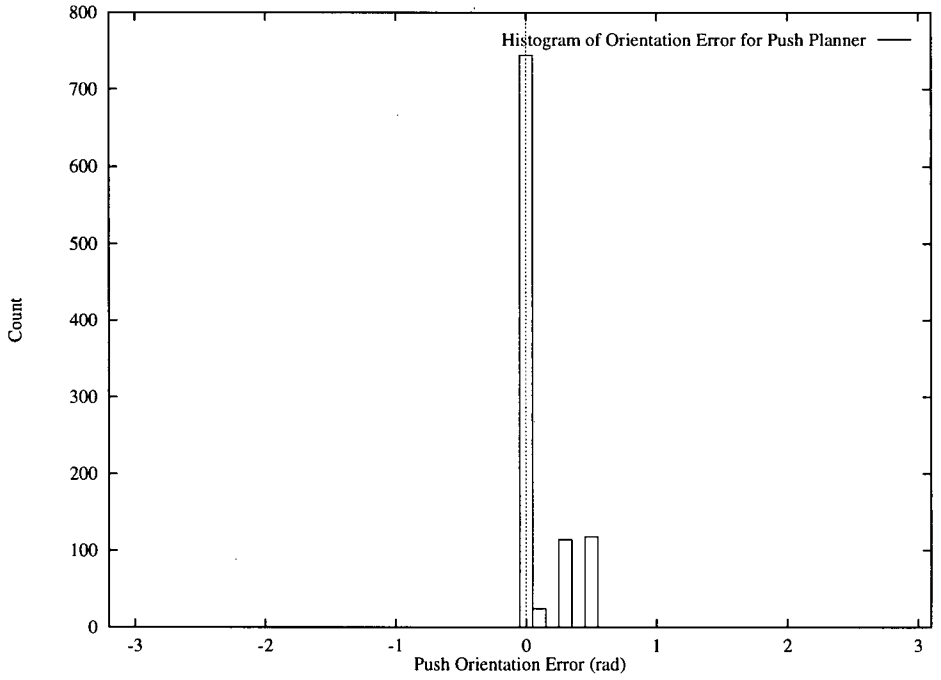


Figure 7.3: Histogram of orientation error for push plan with "G" part for 1000 trials on part positions, and 0.5% standard deviation perturbation in all part features. The average orientation error was 0.10 rad., the maximum error was 0.52 rad. the minimum 0.00 rad.

tree search to fixed depth d, with branching B, the number of fence angles. The planner then requires on the order of B^d operations.

At each step of the action sequence, a sequence of spline approximations are required to generate the straight line motion required for push planning. We will lump this cost along with the other real costs of generating motion in the physical world into the constant c_2. Thus, for the push planner we define the total cost over the lifetime of the system as

$$cost(U_2) = q \cdot B + B^k + P \cdot k \cdot c_2, \tag{7.5}$$

where P is the total number of parts to be oriented by the system. Empirical bounds on the term $O(qB + B^k)$ can be observed in the time outcome tables of appendix A.

In determining ε we look back to figures 6.6, 6.8 and 6.11. The perturbations indicated in these graphs essentially tell us the size of the family of parts in F which are in the equivalence class for p_0 (or $\varepsilon \propto \% - \text{part-dimension}$). For the worst case, e_{wor} will tend

to go to the resting position for p_i on the final fence which is "furthest" from θ_g, almost immediately and is bounded by this value. Figure 7.3 shows a histogram of outcome orientation errors for the "G" with feature perturbations of standard deviation 0.5% of part radius ($\approx 0.03cm$). We can see that distinct peaks corresponding to resting positions form. The observed worst case error is $e_{wor} = 0.52rad$. The average case is related to the falling success rates illustrated in the indicated tables. The observed average error was $e_{avg} = 0.1rad$.

7.2.4 Summary

Equations 7.4 and 7.5 give us a concrete cost comparison for the particular implemented push planning and vision systems. Both systems have a clear cost associated with pre-computation or calibration. For the vision system this computation is dominated by the calibration sequence. With the push planner, the tree traversal associated with finding plans dominates.

Both systems also have a cost associated with each individual part orientation. For the push planner the major component is the cost c_2, of each push motion. For the vision system sensing and combinatoric costs could approach the magnitude of the pick and place actions c_2. When we include the constants c_1 and c_2 in our expressions for the cost of our part orienting systems we should point out that c_1 is at least an order of magnitude more time consuming than a mathematical computation counted by the combinatoric complexity expression; and in turn c_2 is correspondingly more expensive than c_1. With this in mind, when we examine equations 7.4 and 7.5, we can initially see that if the two systems had the same ε error, the vision system would be much more efficient. However, only for the "G" part have we demonstrated that the vision system can approach the average error of the push orienter.

Chapter 8

Conclusion

8.1 Information for Tasks

Information used in robotic tasks can take many forms and be encoded not only in the implementation of algorithms but in engineered environments and sensing strategies. To determine what measurements or information are sufficient for a task implementation we must determine "what to model or sense", "how precisely" and "how accurately". In this thesis we have put forward a view of complex systems for robotic tasks as parameterizations of task functions. These parameters encode task information, usually in the form of features or combined measurements. In general, information is extracted by calibrated measurement of the physical world.

For a particular robotic task goal, there should be some fundamental underlying information requirements. Comparing different implementations of the same goal can give us insight into these requirements. The main contribution of this thesis is a detailed method for comparing and evaluating disparate robotic systems, which clearly demonstrates how even small variations in the goal of a system can radically vary the focus of the information required by the system.

8.1.1 A Methodology for Comparing and Evaluating Task Implementations

Information in robotic tasks is only valuable with respect to task performance. By examining system performance we can compare disparate systems. We have drawn from a number of disciplines including design of experiments, analysis of variance, sensitivity analysis and information based complexity to devise a methodology for evaluating and comparing complex robotic systems in this way. The method we have described and demonstrated in this thesis has essentially four parts:

1. Determine the task parameterization.

2. Use statistical factorial experiments to identify system parameters which are critical to task performance, allowing resources to be allocated appropriately.

3. Perform sensitivity analysis to determine optimal choice of parameters values.

4. Use cost-performance metrics for comparison of sensori-computational systems.

These steps were applied to two very different part orienting systems, one based on push plans, the other a vision based pick and place system. Factorial experiments allowed us to identify critical parameters and parameter interactions, and sensitivity analysis helped us set appropriate parameter values with respect to orienting success.

We have shown that this methodology is useful in extracting the critical aspects of task information for a particular implementation, and in providing a well defined structure in which to analyse and compare differing implementations.

8.2 Comparison of Task Implementations

Our observation in performing our analysis of the performance for the two orienting systems over a set of three parts, is that even such a simple task generated diverse responses in terms of which parameters and interactions were significant. Even *within* the same orienting systems responses were diverse. Clearly the individual part information and its inherent complexity play a major role in these systems.

Experimental systems in vision and robotics have rarely been rigorously tested and documented. An important conclusion of this thesis is that it is vacuous to propose a robot solution without a clear analysis of its system parameters and how they interact with each other, and with task success. This is especially true as the task becomes more complicated. In the analysis of this thesis we have clearly demonstrated that it is these parameterizations that define the information content of the task. These results point the way toward a theory of practical and economic application of robotics in manufacturing and other industries. In other words we are developing a quantitative science of robotics.

8.3 Future Work

The analysis presented in this thesis can and should be applied to more complex systems. We hypothesize that such an analysis of complex robotic implementations can reveal many unexpected sources or encodings of task information, which can yield more effective and well understood systems. One way of approaching large system implementations is to use the tools presented in this thesis to guide system design and optimization.

We would also like to examine other aspects of optimally focusing information resources for particular robots and tasks in areas such as attending, navigating, windowing and tracking.

Bibliography

[1] James H. Abbs and Carolee J. Winstein. Functional Contributions of Rapid Automatic Sensory Based Adjustments to Motor Output. In M. Jeannerod, editor, *Attention and Performance XIII: Motor Representation and Control*, pages 627–652. Lawrence Erlbaum and Associates, Hillsdale, N.J., 1990.

[2] Eric W. Aboaf, Steven M. Drucker, and Christopher G. Atkeson. Task-Level Robot Learning: Juggling a Tennis Ball More Accurately. In *Proceedings 1989 IEEE International Conference on Robotics and Automation*, volume 3, pages 1290–1295, Scottsdale, Arizona, May 1989.

[3] Gerry B. Andeen. Toward a science of assembly. In *Proceedings of the International Workshop on Some Some Critical Issues in Robotics*, Singapore, 1995. US Army Research Office (AMC) - Far East.

[4] Nicholas Ayache and Charles Hansen. Rectification of images for binocular and trinocular stereovision. In *1988 Int. Conf. on Pattern Recognition*, volume 1, pages 11–16, Rome, Italy, October 1988.

[5] Ruzena Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):996–1005, August 1988.

[6] Manuel Blum and Dexter Kozen. On the Power of the Compass. In *19th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 132–142, Ann Arbor, Michigan, October 16-18 1978.

[7] George E. P. Box, William G. Hunter, and J. Stuart Hunter. *Statistics for Experimenters: An Introduction to Design, Data Analysis and Model Building.* John Wiley and Sons, New York, N.Y., 1978.

[8] Ronen I. Brafman, Jean-Claude Latombe, and Yoav Shoham. Towards knowledge-level analysis of motion planning. In *Proceedings of the 11th National Conference on AI (AAAI93)*, pages 670–675, Washington, DC, July 1993.

[9] Rodney A. Brooks. Intelligence without reason. Technical Report AI Memo 1293, MIT AI Lab, April 1991.

[10] Randy C. Brost. Automatic grasp planing in the presence of uncertainty. *International Journal of Robotics Research*, 7(1):3–17, February 1988.

[11] Randy C. Brost. *Analysis and Planning of Planar Manipulation Tasks.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA., January 1991. Appears as CMU Computer Science Tech Report CMU-CS-91-149.

[12] H. Bulthoff, J. Little, and T. Poggio. A Parallel Algorithm for Real-time Computation of Optical Flow. *Nature*, 337(6207):549–553, February 1989.

[13] Michael Edward Caine. *The Design of Shape From Motion Constraints.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Mass, September 1993. Appears as MIT AI Lab Tech Report 1425.

[14] John F. Canny and Kenneth Y. Goldberg. A "RISC" Paradigm for Industrial Robotics. In *1994 IEEE International Conference on Robotics and Automation (ICRA94)*, volume 3, pages 1951–1957, San Diego, CA, May 1994.

[15] T. L. Dean and M. P. Wellman. *Planning and Control.* Morgan-Kaufmann Publishers, San Mateo, Cal., 1991.

[16] Aloke Dey. *Orthogonal Fractional Factorial Designs.* Halsted Press, John Wiley and Sons, New York, NY, 1985.

[17] Bruce Donald and Jim Jennings. Sensor Interpretation and Task Directed Planning Using Perceptual Equivalence Classes. In *IEEE International Conference on Robotics and Automation*, Anaheim, Ca., 1991.

[18] Bruce Randall Donald. On information invariants in robotics. Technical Report TR 93-1341, Cornell University, Ithaca, New York, April 1993.

[19] R.O. Duda and P.E. Hart. Use of the Hough Transformation To Detect Lines and Curves in Pictures. *Communications of the ACM*, 15(1):11–15, January 1972.

[20] Brian Scott Eberman. *Contact Sensing: A Sequential Decision Approach to Sensing Manipulation Contact Features.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Mass., February 1995.

[21] Michael Erdmann. Action subservient sensing and design. In *1993 IEEE International Conference on Robotics and Automation*, volume 2, pages 592–598, Atlanta, Georgia, May 2-6, 1993.

[22] Michael Erdmann. Understanding action and sensing by designing action-based sensors. In *Proceedings of the IEEE ICRA Workshop on Design*, San Diego, Ca., May 1994.

[23] John T. Feddema, C.S. George Lee, and Owen Robert Mitchell. Weighted Selection of Image Features for Resolved Rate Visual Feedback Control. *IEEE Transactions on Robotics and Automation*, 7(1):31–47, February 1991.

[24] R. James Firby. Task directed sensing. In *Sensor Fusion II: Human and Machine Strategies*, volume SPIE 1198, pages 480–489, 1989.

[25] Gene F. Franklin and J. David Powell. *Digital Control of Dynamic Systems.* Addison-Wesley Publishing Co., Reading, Mass., 1980.

[26] K.S. Fu, R.C. Gonzalez, and C.S.G. Lee. *Robotics: Control, Sensing, Vision and Intelligence.* McGraw-Hill Book Company, New York, N.Y., 1987.

[27] Kenneth Y. Goldberg and Matthew T. Mason. Bayesian grasping. In *Proceedings of the International Conference on Robotics and Automation*, pages 1264–1269. IEEE, 1990.

[28] Richard Hartley and Rajiv Gupta. Computing matched-epipolar projections. In *Proceedings 1993 Conf on Computer Vision and Pattern Recognition (CVPR93)*, pages 549–555, New York, NY, 1993.

[29] Berthold Klaus Paul Horn. *Robot Vision.* The MIT Press, Cambridge, Mass., 1986.

[30] Marc Jeannerod and B. Biguer. Visuomotor Mechanisms in Reaching Within Extrapersonal Space. In D.J. Ingle, M.A. Goodale, and R.J.W. Mansfield, editors, *Analysis of Visual Behavior.* MIT Press, Cambridge, Mass., 1982.

[31] William J. Karnavas, Paul J. Sanchez, and A. Terry Bahill. Sensitivity analyses of continuous and discrete systems in the time and frequency domains. *IEEE Transactions on Systems, Man and Cybernetics*, 23(2):488–501, March/April 1993.

[32] P. Kazanzides, J. Zuhars, B. Mittlestadt, P. Cain, F. Smith, L. Rose, and B. Musits. Architecture of a surgical robot. In *Proceedings of the 1992 IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 1624–9, Chicago, Il., October 1992.

[33] Oscar Kempthorne. *The Design and Analysis of Experiments.* John Wiley and Sons, New York, N.Y., 1952.

[34] Jak Kirman, Kenneth Basye, and Thomas Dean. Sensor abstraction for control of navigation. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation (ICRA'91)*, pages 2812–2817, Sacramento, Ca., April 1991.

[35] Anthony Lazanas and Jean-Claude Latombe. Landmark-based robot navigation. In *Proceedings of the 10th National Conference on AI (AAAI'92)*, pages 816–822, San Jose, Cal., July 1992.

[36] Susan J. Lederman and Roberta L. Klatzky. Hand Movements: A Window into Haptic Object Recognition. *Cognitive Psychology*, 19:342–368, 1987.

[37] D.G. Lowe. Three-Dimensional Object Recognition from Single Two Dimensional Images. *Artificial Intelligence*, 31:355–395, 1987.

[38] D. M. MacKay. The significance of 'feature sensitivity'. In *Models of the Visual Cortex.* John Wiley and Sons Ltd, 1985.

[39] A.K. Mackworth. Constraints, Descriptions and Domain Mappings in Computational Vision. In O.J. Braddock and A. C. Sleigh, editors, *Physical and Biological Processing of Images*, pages 33–40. Springer Verlag, Berlin, 1983.

[40] A.K. Mackworth. Adequacy Criteria for Visual Knowledge Representation. Technical Report 87-4, Dept. of Computer Science, U.B.C., Vancouver, B.C. Canada, January 1987.

[41] Matthew T. Mason. Mechanics and Planning of Manipulator Pushing Operations. *The International Journal of Robotics Research*, 5(3):52–71, 1986.

[42] Matthew T. Mason. Kicking the sensing habit. *AI Magazine*, pages 59–59, Spring 1993.

[43] Maja Mataric. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, June 1992.

[44] William Mendenhall, Richard L. Scheaffer, and Dennis D. Wackerly. *Mathematical Statistics with Applications.* Duxbury Press, Boston, Mass., 1981.

[45] P. S. Moharir. *Pattern-recognition Transforms.* Electronic Circuits and Systems Series. Research Studies Press Ltd. (John Wiley and Sons Inc.), Somerset, England, 1992.

[46] Jane Mulligan. Fast calibrated stereo vision for manipulation. In *Proceedings 1996 IEEE International Conference on Robotics and Automation (ICRA '96)*, volume 3, pages 2326–2331, Minneapolis, Minnesota, April 1996.

[47] Jane Mulligan. Fast calibrated stereo vision for manipulation. *Journal of Real-time Imaging*, 1996. To Appear.

[48] B.K. Natarajan. On Planning Assemblies. In *Proceedings of the 4th Annual Symposium on Computational Geometry*, pages 299–308, Urbana Champaign, Ill., 1988.

[49] Edward W. Packel and J. F. Traub. Information-based complexity. *Nature*, 328:29–33, July 1987.

[50] Michael A. Peshkin and Arthur C. Sanderson. The motion of a pushed, sliding workpiece. *IEEE Journal of Robotics and Automation*, 4(6):569–598, 1988.

[51] Michael A. Peshkin and Arthur C. Sanderson. Planning robotic manipulation strategies for workpieces that slide. *IEEE Journal of Robotics and Automation*, 4(5):524–531, 1988.

[52] Semyon Rabinovich. *Measurement Errors: Theory and Practice.* American Institute of Physics, New York, NY, 1993.

[53] W. Richards and A. Jepson. What makes a good feature? Technical Report AI Memo No. 1356/CBIP Paper No. 72, Massachusetts Institute of Technology Artificial Intelligence Laboratory, Cambridge, Mass., April 1992.

[54] Steven Roman. *Coding and Information Theory*, volume 134 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, N.Y., 1992.

[55] Ranjit K. Roy. *A Primer on the Taguchi Method*. Competitive Manufacturing Series. Van Nostrand Reinhold, New York, NY, 1990.

[56] Jagdish S. Rustagi. *Introduction to Statistical Methods*, volume I. Rowman and Allanheld, Totowa, New Jersey, 1984.

[57] Arthur C. Sanderson. Parts entropy for robotic assembly system design. In *International Conference on Robotics*, pages 600–608, Atlanta, Georgia, May 13-15 1984.

[58] Lee W. Schruben and V. James Cogliano. An experimental procedure for simulation response surface model identification. *Communications of the ACM*, 30(8):716–730, August 1987.

[59] Larry S. Shapiro and J. Michael Brady. A modal approach to feature-based correspondence. In *Proceedings of the British Machine Vision Conference*, pages 78–85, Glasgow, UK, 1991.

[60] Steven B. Skaar, William H. Brockman, and R. Hanson. Camera Space Manipulation. *The International Journal of Robotics Research*, 6(4):20–32, 1987.

[61] Jack Snoeyink and Jorge Stolfi. Objects that cannot be taken apart with two hands. In *Proceedings of the 9th ACM Symposium on Computational Geometry*, 1993.

[62] Fridtjof Stein and Gerard Medioni. Structural indexing: Efficient 2-d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 14(12):1198–1204, December 1992.

[63] Anne Penfold Street and Deborah J. Street. *Combinatorics of Experimental Design*. Clarendon Press, Oxford, UK, 1987.

[64] P. H. Sydenham, N. H. Hancock, and R. Thorn. *Introduction to Measurement Science and Engineering*. Wiley Series in Measurement Science and Technology. John Wiley and Sons, Chichester, UK, 1989.

[65] Konstantinos A. Tarabanis, Peter K. Allen, and Roger Y. Tsai. A survey of sensor planning in computer vision. *IEEE Transactions on Robotics and Automation*, 11(1):86–104, February 1995.

[66] Albert Tarantola. *Inverse Problem Theory: Methods for Data Fitting and Model Parameter Estimation*. Elsevier, Amsterdam, 1987.

[67] Russell H. Taylor. Planning and execution of straight line manipulator trajectories. In *Robot Motion: Planning and Control*, pages 265–286. MIT Press, Cambridge, Mass., 1982.

[68] J. F. Traub, G. W. Wasilkowski, and H. Wozniakowski. *Information-Based Complexity.* Computer Science and Scientific Computing. Academic Press Inc., San Diego, CA, 1988.

[69] Anne Treisman. Preattentive Processing in Vision. *Computer Vision, Graphics and Image Processing*, 31:156–177, 1985.

[70] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323–344, August 1987.

[71] Matthew A. Turk and Alex P. Pentland. Face Recognition using Eigenfaces. In *Proceedings of the 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR91*, pages 586–591, Lahaina, Maui, Hawaii, June 1991.

[72] Arthur G. Werschulz. *The Computational Complexity of Differential and Integral Equations.* Oxford Mathemtical Monographs. Oxford University Press, Oxford, UK, 1991.

[73] Peter Whaite and Frank Ferrie. Autonomous exploration: Driven by uncertainty. Technical Report TR-CIM-93-17, McGill Research Centre for Intelligent Machines, March 1993.

[74] Jeff Wiegley, Ken Goldberg, Mike Peshkin, and Mike Brokowski. A complete algorithm for designing passive fences to orient parts. In *Proceedings 1996 IEEE International Conference on Robotics and Automation (ICRA'96)*, volume 2, pages 1133–1139, Minneapolis, Minnesota, April 1996.

Appendix A

Treatments and Outcomes - Push Planner

A.1 Triangle

The following tables summarize effects and treatment outcomes for the triangle part under the push planner as described in section 6.3.1.

A.1.1 Success Rate

Effects

Factor Effects			
Factor(s)	Effect	Factor(s)	Effect
1 (0)	0.479	LN (90)	0.008
F (1)	-0.017	SN (A0)	-0.035
C (2)	-0.091	PN (C0)	0.021
FC (3)	0.014	A (100)	-0.023
V (4)	0.011	FA (101)	-0.066
FV (5)	0.028	CA (102)	-0.062
CV (6)	0.058	VA (104)	0.010
D (8)	-0.056	DA (108)	-0.010
FD (9)	-0.035	LA (110)	-0.006
CD (A)	-0.008	SA (120)	0.076
VD (C)	0.030	PA (140)	-0.070
L (10)	-0.159	NA (180)	-0.044
FL (11)	-0.004	B (200)	-0.016
CL (12)	-0.026	FB (201)	-0.004
VL (14)	0.032	CB (202)	-0.007
DL (18)	-0.110	VB (204)	0.052
S (20)	0.087	DB (208)	-0.028
FS (21)	-0.031	LB (210)	0.058
CS (22)	-0.020	SB (220)	0.040
VS (24)	0.023	PB (240)	0.013
DS (28)	0.027	NB (280)	-0.048
LS (30)	0.105	AB (300)	0.009

Factor Effects cont'd			
Factor(s)	Effect	Factor(s)	Effect
P (40)	0.011	K (400)	0.017
FP (41)	-0.048	FK (401)	-0.038
CP (42)	-0.013	CK (402)	0.043
VP (44)	0.036	VK (404)	0.080
DP (48)	-0.054	DK (408)	0.044
LP (50)	0.087	LK (410)	-0.042
SP (60)	-0.080	SK (420)	-0.138
N (80)	-0.058	PK (440)	0.001
FN (81)	0.036	NK (480)	-0.056
CN (82)	0.021	AK (500)	0.032
VN (84)	0.014	BK (600)	-0.018
DN (88)	-0.004		

Outcomes

Treatment Outcomes							
Levels	Obs	Levels	Obs	Levels	Obs	Levels	Obs
1	0.905	fcv	0.716	dl	0.026	fcvdl	0.211
fds	0.976	cvds	0.480	fls	0.000	cvls	0.281
fdp	0.591	cvdp	0.535	flp	0.567	cvlp	0.681
sp	1.000	fcvsp	0.729	dlsp	1.000	fcvdlsp	0.000
cdn	0.677	fvdn	0.499	cln	0.073	fvln	0.324
fcsn	0.826	vsn	0.000	fcdlsn	0.727	vdlsn	0.512
fcpn	0.587	vpn	0.723	fcdlpn	0.613	vdlpn	0.313
cdspn	0.582	fvdspn	0.693	clspn	0.010	fvlspn	0.769
cda	0.689	fvda	0.667	cla	0.002	fvla	0.290
fcsa	0.860	vsa	0.000	fcdlsa	0.353	vdlsa	0.774
fcpa	0.444	vpa	0.000	fcdlpa	0.000	vdlpa	0.149
cdspa	0.000	fvdspa	0.309	clspa	0.008	fvlspa	0.545
na	0.906	fcvna	0.630	dlna	0.000	fcvdlna	0.115
fdsna	0.889	cvdsna	0.543	flsna	0.952	cvlsna	0.841
fdpna	0.000	cvdpna	0.000	flpna	0.318	cvlpna	0.000
spna	1.000	fcvspna	0.467	dlspna	1.000	fcvdlspna	0.673
fcdb	0.736	vdb	0.752	fclb	0.071	vlb	0.094
csb	0.461	fvsb	0.000	cdlsb	0.524	fvdlsb	0.657
cpb	0.005	fvpb	0.563	cdlpb	0.000	fvdlpb	0.000
fcdspb	0.662	vdspb	0.851	fclspb	0.758	vlspb	0.964
fnb	0.788	cvnb	0.000	fdlnb	0.524	cvdlnb	0.002
dsnb	1.000	fcvdsnb	0.000	lsnb	1.000	fcvlsnb	0.630
dpnb	0.000	fcvdpnb	0.822	lpnb	0.848	fcvlpnb	0.931
fspnb	0.761	cvspnb	0.840	fdlspnb	0.000	cvdlspnb	0.003
fab	0.808	cvab	0.721	fdlab	0.000	cvdlab	0.030
dsab	1.000	fcvdsab	0.041	lsab	0.999	fcvlsab	0.749
dpab	0.000	fcvdpab	0.498	lpab	0.976	fcvlpab	0.482
fspab	0.742	cvspab	0.713	fdlspab	0.887	cvdlspab	0.614
fcdnab	0.000	vdnab	0.000	fclnab	0.126	vlnab	0.000
csnab	0.000	fvsnab	0.957	cdlsnab	0.639	fvdlsnab	0.723
cpnab	0.881	fvpnab	0.000	cdlpnab	0.000	fvdlpnab	0.000
fcdspnab	0.498	vdspnab	0.832	fclspnab	0.024	vlspnab	0.964
fcdk	0.686	vdk	0.810	fclk	0.006	vlk	0.034
csk	0.000	fvsk	0.719	cdlsk	0.650	fvdlsk	0.742
cpk	0.748	fvpk	0.559	cdlpk	0.801	fvdlpk	0.297
fcdspk	0.631	vdspk	0.845	fclspk	0.565	vlspk	0.000

Treatment Outcomes (cont'd)							
Levels	Obs	Levels	Obs	Levels	Obs	Levels	Obs
fnk	0.848	cvnk	0.903	fdlnk	0.305	cvdlnk	0.022
dsnk	0.000	fcvdsnk	0.821	lsnk	0.045	fcvlsnk	0.862
dpnk	1.000	fcvdpnk	0.508	lpnk	1.000	fcvlpnk	0.386
fspnk	0.000	cvspnk	0.755	fdlspnk	0.000	cvdlspnk	0.001
fak	0.800	cvak	0.842	fdlak	0.000	cvdlak	0.003
dsak	0.995	fcvdsak	0.657	lsak	1.000	fcvlsak	0.645
dpak	0.952	fcvdpak	0.448	lpak	1.000	fcvlpak	0.655
fspak	0.971	cvspak	0.000	fdlspak	0.000	cvdlspak	0.622
fcdnak	0.593	vdnak	0.896	fclnak	0.059	vlnak	0.000
csnak	0.831	fvsnak	0.000	cdlsnak	0.547	fvdlsnak	0.168
cpnak	0.774	fvpnak	0.692	cdlpnak	0.040	fvdlpnak	0.743
fcdspnak	0.000	vdspnak	0.879	fclspnak	0.046	vlspnak	0.869
bk	0.916	fcvbk	0.000	dlbk	0.000	fcvdlbk	0.214
fdsbk	0.711	cvdsbk	0.608	flsbk	0.867	cvlsbk	0.874
fdpbk	0.000	cvdpbk	0.914	flpbk	0.884	cvlpbk	0.595
spbk	0.000	fcvspbk	0.886	dlspbk	0.000	fcvdlspbk	0.462
cdnbk	0.906	fvdnbk	0.801	clnbk	0.129	fvlnbk	0.318
fcsnbk	0.000	vsnbk	0.512	fcdlsnbk	0.002	vdlsnbk	0.000
fcpnbk	0.759	vpnbk	0.000	fcdlpnbk	0.751	vdlpnbk	0.760
cdspnbk	0.000	fvdspnbk	0.675	clspnbk	0.836	fvlspnbk	0.960
cdabk	0.735	fvdabk	0.795	clabk	0.000	fvlabk	0.927
fcsabk	0.850	vsabk	0.962	fcdlsabk	0.183	vdlsabk	0.697
fcpabk	0.000	vpabk	0.877	fcdlpabk	0.000	vdlpabk	0.901
cdspabk	0.742	fvdspabk	0.711	clspabk	0.833	fvlspabk	0.000
nabk	0.000	fcvnabk	0.768	dlnabk	0.021	fcvdlnabk	0.236
fdsnabk	0.883	cvdsnabk	0.789	flsnabk	0.000	cvlsnabk	0.629
fdpnabk	0.799	cvdpnabk	0.843	flpnabk	0.587	cvlpnabk	0.000
spnabk	0.000	fcvspnabk	0.000	dlspnabk	0.000	fcvdlspnabk	0.000

A.1.2 Planning Time

Effects

Factor Effects			
Factor(s)	Effect	Factor(s)	Effect
1 (0)	48.208	LN (90)	-11.740
F (1)	-0.959	SN (A0)	1.913
C (2)	-2.013	PN (C0)	10.006
FC (3)	-5.590	A (100)	-0.823
V (4)	0.340	FA (101)	-3.316
FV (5)	7.628	CA (102)	1.686
CV (6)	-1.673	VA (104)	-0.698
D (8)	89.091	DA (108)	-0.407
FD (9)	-0.866	LA (110)	-4.019
CD (A)	-1.893	SA (120)	1.917
VD (C)	0.642	PA (140)	4.513
L (10)	58.895	NA (180)	1.647
FL (11)	1.536	B (200)	-4.302
CL (12)	-2.036	FB (201)	-0.376
VL (14)	1.588	CB (202)	-3.169
DL (18)	53.797	VB (204)	-3.646
S (20)	5.672	DB (208)	-4.486
FS (21)	4.017	LB (210)	-5.251
CS (22)	1.885	SB (220)	4.648
VS (24)	-2.382	PB (240)	5.037
DS (28)	5.177	NB (280)	9.692
LS (30)	4.110	AB (300)	0.318
P (40)	-34.547	K (400)	8.659
FP (41)	-1.448	FK (401)	-0.557
CP (42)	1.021	CK (402)	0.241
VP (44)	-0.848	VK (404)	-3.910
DP (48)	-32.228	DK (408)	9.977
LP (50)	-23.823	LK (410)	11.753
SP (60)	2.963	SK (420)	4.020
N (80)	-13.912	PK (440)	-1.873
FN (81)	0.642	NK (480)	-2.349
CN (82)	4.482	AK (500)	-3.681
VN (84)	-0.526	BK (600)	4.622
DN (88)	-13.429		

Outcomes

Treatment Outcomes							
Levels	Obs	Levels	Obs	Levels	Obs	Levels	Obs
1	1.950	fcv	1.900	dl	383.033	fcvdl	377.600
fds	21.983	cvds	22.217	fls	11.417	cvls	11.233
fdp	3.367	cvdp	7.517	flp	1.683	cvlp	3.183
sp	1.100	fcvsp	0.667	dlsp	30.867	fcvdlsp	142.417
cdn	24.583	fvdn	63.283	cln	10.867	fvln	10.883
fcsn	1.767	vsn	1.817	fcdlsn	134.783	vdlsn	128.500
fcpn	0.633	vpn	0.683	fcdlpn	17.917	vdlpn	42.433
cdspn	45.483	fvdspn	38.067	clspn	11.417	fvlspn	3.267
cda	75.000	fvda	72.783	cla	12.433	fvla	12.433
fcsa	2.067	vsa	2.050	fcdlsa	169.983	vdlsa	169.117
fcpa	0.717	vpa	2.333	fcdlpa	41.700	vdlpa	158.350
cdspa	76.117	fvdspa	41.850	clspa	10.033	fvlspa	8.117
na	2.033	fcvna	2.017	dlna	123.433	fcvdlna	147.517
fdsna	74.700	cvdsna	82.917	flsna	11.933	cvlsna	4.383
fdpna	23.167	cvdpna	81.067	flpna	10.000	cvlpna	2.767
spna	1.917	fcvspna	1.850	dlspna	48.350	fcvdlspna	62.167
fcdb	20.950	vdb	65.983	fclb	12.133	vlb	11.583
csb	1.967	fvsb	1.883	cdlsb	140.717	fvdlsb	148.850
cpb	0.217	fvpb	1.000	cdlpb	18.067	fvdlpb	31.117
fcdspb	10.200	vdspb	13.117	fclspb	5.183	vlspb	9.867
fnb	1.850	cvnb	1.800	fdlnb	141.683	cvdlnb	149.067
dsnb	79.900	fcvdsnb	67.000	lsnb	10.717	fcvlsnb	10.733
dpnb	14.733	fcvdpnb	10.617	lpnb	1.417	fcvlpnb	1.817
fspnb	0.433	cvspnb	1.150	fdlspnb	130.983	cvdlspnb	41.500
fab	1.333	cvab	1.283	fdlab	154.000	cvdlab	149.500
dsab	46.450	fcvdsab	46.400	lsab	7.733	fcvlsab	7.767
dpab	46.617	fcvdpab	9.433	lpab	1.900	fcvlpab	2.067
fspab	0.817	cvspab	1.250	fdlspab	164.567	cvdlspab	167.100
fcdnab	25.033	vdnab	24.883	fclnab	4.283	vlnab	4.267
csnab	0.700	fvsnab	0.683	cdlsnab	169.133	fvdlsnab	170.883
cpnab	0.217	fvpnab	0.150	cdlpnab	51.050	fvdlpnab	76.050
fcdspnab	17.350	vdspnab	4.683	fclspnab	3.483	vlspnab	3.333
fcdk	21.817	vdk	21.717	fclk	3.700	vlk	3.700
csk	0.600	fvsk	1.800	cdlsk	374.333	fvdlsk	425.467
cpk	1.300	fvpk	0.250	cdlpk	110.617	fvdlpk	159.517
fcdspk	60.233	vdspk	31.950	fclspk	1.950	vlspk	3.333

Treatment Outcomes (cont'd)							
Levels	Obs	Levels	Obs	Levels	Obs	Levels	Obs
fnk	0.600	cvnk	0.583	fdlnk	128.300	cvdlnk	129.150
dsnk	21.400	fcvdsnk	20.883	lsnk	3.617	fcvlsnk	3.567
dpnk	3.917	fcvdpnk	2.400	lpnk	0.800	fcvlpnk	1.067
fspnk	0.600	cvspnk	0.250	fdlspnk	50.500	cvdlspnk	111.050
fak	0.650	cvak	0.683	fdlak	271.767	cvdlak	271.683
dsak	35.883	fcvdsak	35.600	lsak	6.317	fcvlsak	6.433
dpak	25.800	fcvdpak	4.800	lpak	1.183	fcvlpak	1.050
fspak	0.950	cvspak	0.500	fdlspak	144.467	cvdlspak	71.083
fcdnak	36.500	vdnak	36.067	fclnak	6.283	vlnak	6.467
csnak	1.017	fvsnak	1.000	cdlsnak	218.617	fvdlsnak	220.850
cpnak	0.167	fvpnak	0.200	cdlpnak	158.433	fvdlpnak	59.800
fcdspnak	13.650	vdspnak	6.967	fclspnak	3.050	vlspnak	3.283
bk	0.633	fcvbk	0.667	dlbk	247.600	fcvdlbk	202.100
fdsbk	64.717	cvdsbk	39.933	flsbk	7.033	cvlsbk	7.000
fdpbk	41.217	cvdpbk	31.150	flpbk	5.717	cvlpbk	2.133
spbk	1.817	fcvspbk	1.750	dlspbk	146.267	fcvdlspbk	72.567
cdnbk	61.717	fvdnbk	61.650	clnbk	10.750	fvlnbk	10.767
fcsnbk	1.800	vsnbk	1.783	fcdlsnbk	201.800	vdlsnbk	205.733
fcpnbk	0.683	vpnbk	0.567	fcdlpnbk	162.717	vdlpnbk	126.450
cdspnbk	35.417	fvdspnbk	32.783	clspnbk	10.267	fvlspnbk	8.383
cdabk	88.367	fvdabk	82.100	clabk	13.117	fvlabk	12.283
fcsabk	2.100	vsabk	2.100	fcdlsabk	268.967	vdlsabk	269.600
fcpabk	1.250	vpabk	0.483	fcdlpabk	26.600	vdlpabk	71.400
cdspabk	46.317	fvdspabk	18.567	clspabk	3.250	fvlspabk	2.833
nabk	0.700	fcvnabk	0.700	dlnabk	139.717	fcvdlnabk	140.217
fdsnabk	27.767	cvdsnabk	29.133	flsnabk	4.083	cvlsnabk	4.167
fdpnabk	15.050	cvdpnabk	17.200	flpnabk	1.067	cvlpnabk	4.533
spnabk	0.667	fcvspnabk	0.183	dlspnabk	128.467	fcvdlspnabk	144.100

A.2 "L" Part

The following tables summarize effects and treatment outcomes for the "L" part under the push planner as described in section 6.3.2.

A.2.1 Success Rate

Effects

Factor Effects			
Factor(s)	Effect	Factor(s)	Effect
1 (0)	0.441	LN (90)	0.016
F (1)	-0.051	SN (A0)	0.036
C (2)	-0.083	PN (C0)	-0.001
FC (3)	0.013	A (100)	-0.030
V (4)	-0.034	FA (101)	-0.082
FV (5)	0.051	CA (102)	-0.023
CV (6)	0.070	VA (104)	0.011
D (8)	-0.126	DA (108)	0.045
FD (9)	0.019	LA (110)	-0.003
CD (A)	0.015	SA (120)	0.029
VD (C)	0.020	PA (140)	-0.047
L (10)	-0.196	NA (180)	0.021
FL (11)	0.015	B (200)	-0.025
CL (12)	-0.017	FB (201)	-0.038
VL (14)	-0.002	CB (202)	-0.062
DL (18)	-0.088	VB (204)	0.044
S (20)	0.065	DB (208)	-0.045
FS (21)	-0.007	LB (210)	0.008
CS (22)	-0.029	SB (220)	0.059
VS (24)	0.014	PB (240)	-0.040
DS (28)	0.022	NB (280)	-0.031
LS (30)	0.139	AB (300)	0.010
P (40)	-0.022	K (400)	-0.005
FP (41)	-0.036	FK (401)	0.003

Factor Effects - cont'd			
Factor(s)	Effect	Factor(s)	Effect
CP (42)	0.028	CK (402)	0.056
VP (44)	-0.023	VK (404)	0.029
DP (48)	-0.049	DK (408)	-0.006
LP (50)	0.088	LK (410)	-0.038
SP (60)	-0.098	SK (420)	-0.138
N (80)	-0.083	PK (440)	0.018
FN (81)	-0.020	NK (480)	-0.054
CN (82)	0.007	AK (500)	0.020
VN (84)	0.052	BK (600)	0.001
DN (88)	0.029		

Outcomes

Treatment Outcomes							
Levels	Obs	Levels	Obs	Levels	Obs	Levels	Obs
1	1.000	fcv	0.550	dl	0.010	fcvdl	0.130
fds	0.741	cvds	0.569	fls	0.871	cvls	0.755
fdp	0.802	cvdp	0.856	flp	0.727	cvlp	0.770
sp	0.877	fcvsp	0.770	dlsp	1.000	fcvdlsp	0.000
cdn	0.000	fvdn	0.404	cln	0.005	fvln	0.561
fcsn	0.828	vsn	0.000	fcdlsn	0.620	vdlsn	0.000
fcpn	0.633	vpn	0.000	fcdlpn	0.808	vdlpn	0.000
cdspn	0.910	fvdspn	0.475	clspn	0.000	fvlspn	0.843
cda	0.768	fvda	0.435	cla	0.016	fvla	0.001
fcsa	0.786	vsa	0.057	fcdlsa	0.253	vdlsa	0.601
fcpa	0.516	vpa	0.834	fcdlpa	0.000	vdlpa	0.000
cdspa	0.000	fvdspa	0.259	clspa	0.000	fvlspa	0.001
na	1.000	fcvna	0.669	dlna	0.009	fcvdlna	0.102
fdsna	0.981	cvdsna	0.752	flsna	0.966	cvlsna	0.838
fdpna	0.000	cvdpna	0.000	flpna	0.011	cvlpna	0.000
spna	0.581	fcvspna	0.524	dlspna	1.000	fcvdlspna	0.716
fcdb	0.906	vdb	0.630	fclb	0.132	vlb	0.017
csb	0.889	fvsb	0.000	cdlsb	0.000	fvdlsb	0.722
cpb	0.010	fvpb	0.763	cdlpb	0.000	fvdlpb	0.000
fcdspb	0.001	vdspb	0.706	fclspb	0.648	vlspb	0.973
fnb	0.373	cvnb	0.095	fdlnb	0.007	cvdlnb	0.007
dsnb	1.000	fcvdsnb	0.484	lsnb	1.000	fcvlsnb	0.480
dpnb	0.487	fcvdpnb	0.564	lpnb	1.000	fcvlpnb	0.003
fspnb	0.787	cvspnb	0.752	fdlspnb	0.013	cvdlspnb	0.002
fab	0.529	cvab	0.798	fdlab	0.000	cvdlab	0.008
dsab	1.000	fcvdsab	0.071	lsab	1.000	fcvlsab	0.847
dpab	0.000	fcvdpab	0.497	lpab	0.761	fcvlpab	0.128
fspab	0.516	cvspab	0.334	fdlspab	0.976	cvdlspab	0.471
fcdnab	0.000	vdnab	0.900	fclnab	0.001	vlnab	0.636
csnab	0.000	fvsnab	0.827	cdlsnab	0.683	fvdlsnab	0.305
cpnab	0.941	fvpnab	0.072	cdlpnab	0.000	fvdlpnab	0.000
fcdspnab	0.000	vdspnab	0.897	fclspnab	0.000	vlspnab	0.893
fcdk	0.891	vdk	0.867	fclk	0.002	vlk	0.003
csk	0.000	fvsk	0.836	cdlsk	0.593	fvdlsk	0.060
cpk	0.948	fvpk	0.772	cdlpk	0.813	fvdlpk	0.891
fcdspk	0.323	vdspk	0.001	fclspk	0.358	vlspk	0.000

Treatment Outcomes (cont'd)							
Levels	Obs	Levels	Obs	Levels	Obs	Levels	Obs
fnk	0.840	cvnk	0.851	fdlnk	0.207	cvdlnk	0.009
dsnk	0.000	fcvdsnk	0.652	lsnk	0.000	fcvlsnk	0.767
dpnk	0.569	fcvdpnk	0.545	lpnk	1.000	fcvlpnk	0.000
fspnk	0.000	cvspnk	0.954	fdlspnk	0.000	cvdlspnk	0.000
fak	0.576	cvak	0.777	fdlak	0.078	cvdlak	0.002
dsak	1.000	fcvdsak	0.537	lsak	1.000	fcvlsak	0.000
dpak	0.952	fcvdpak	0.865	lpak	0.941	fcvlpak	0.720
fspak	0.513	cvspak	0.000	fdlspak	0.000	cvdlspak	0.394
fcdnak	0.000	vdnak	0.796	fclnak	0.056	vlnak	0.003
csnak	0.853	fvsnak	0.093	cdlsnak	0.000	fvdlsnak	0.788
cpnak	0.799	fvpnak	0.584	cdlpnak	0.824	fvdlpnak	0.000
fcdspnak	0.000	vdspnak	0.551	fclspnak	0.633	vlspnak	0.708
bk	1.000	fcvbk	0.633	dlbk	0.000	fcvdlbk	0.099
fdsbk	0.564	cvdsbk	0.543	flsbk	0.845	cvlsbk	0.774
fdpbk	0.250	cvdpbk	0.000	flpbk	0.836	cvlpbk	0.724
spbk	1.000	fcvspbk	0.683	dlspbk	0.000	fcvdlspbk	0.378
cdnbk	0.985	fvdnbk	0.512	clnbk	0.004	fvlnbk	0.514
fcsnbk	0.000	vsnbk	0.933	fcdlsnbk	0.001	vdlsnbk	0.577
fcpnbk	0.806	vpnbk	0.106	fcdlpnbk	0.097	vdlpnbk	0.000
cdspnbk	0.000	fvdspnbk	0.412	clspnbk	0.618	fvlspnbk	0.728
cdabk	0.687	fvdabk	0.691	clabk	0.018	fvlabk	0.966
fcsabk	0.763	vsabk	0.898	fcdlsabk	0.709	vdlsabk	0.038
fcpabk	0.709	vpabk	0.628	fcdlpabk	0.000	vdlpabk	0.000
cdspabk	0.598	fvdspabk	0.684	clspabk	0.941	fvlspabk	0.072
nabk	0.493	fcvnabk	0.326	dlnabk	0.000	fcvdlnabk	0.106
fdsnabk	0.710	cvdsnabk	0.642	flsnabk	0.000	cvlsnabk	0.001
fdpnabk	0.825	cvdpnabk	0.513	flpnabk	0.000	cvlpnabk	0.652
spnabk	0.000	fcvspnabk	0.000	dlspnabk	0.000	fcvdlspnabk	0.000

A.2.2 Planning Time

Effects

Factor Effects			
Factor(s)	Effect	Factor(s)	Effect
1 (0)	124.136	LN (90)	39.551
F (1)	-3.204	SN (A0)	-21.571
C (2)	7.532	PN (C0)	2.929
FC (3)	-41.711	A (100)	54.368
V (4)	-29.016	FA (101)	-6.771
FV (5)	27.801	CA (102)	5.041
CV (6)	-26.924	VA (104)	-11.836
D (8)	235.453	DA (108)	51.479
FD (9)	-3.124	LA (110)	41.093
CD (A)	7.558	SA (120)	7.668
VD (C)	-29.072	PA (140)	14.308
L (10)	181.750	NA (180)	21.804
FL (11)	-3.861	B (200)	-2.842
CL (12)	5.411	FB (201)	-14.306
VL (14)	-30.367	CB (202)	-0.704
DL (18)	172.708	VB (204)	-16.363
S (20)	10.203	DB (208)	-3.357
FS (21)	19.705	LB (210)	-8.891
CS (22)	-20.947	SB (220)	-18.927
VS (24)	17.753	PB (240)	-1.734
DS (28)	10.461	NB (280)	16.040
LS (30)	9.034	AB (300)	24.257
P (40)	-34.984	K (400)	24.139
FP (41)	1.439	FK (401)	-1.484
CP (42)	18.040	CK (402)	-16.319
VP (44)	-21.272	VK (404)	16.176
DP (48)	-32.835	DK (408)	25.390
LP (50)	-26.217	LK (410)	25.659
SP (60)	-18.693	SK (420)	19.593
N (80)	37.426	PK (440)	-43.276
FN (81)	-12.623	NK (480)	-9.547
CN (82)	30.764	AK (500)	-19.501
VN (84)	-6.533	BK (600)	-46.321
DN (88)	37.211		

Outcomes

Treatment Outcomes							
Levels	Obs	Levels	Obs	Levels	Obs	Levels	Obs
1	1.117	fcv	1.117	dl	241.283	fcvdl	241.033
fds	39.517	cvds	39.467	fls	6.700	cvls	6.633
fdp	16.533	cvdp	27.717	flp	2.333	cvlp	2.033
sp	1.017	fcvsp	0.667	dlsp	140.017	fcvdlsp	197.350
cdn	39.767	fvdn	39.400	cln	6.683	fvln	6.750
fcsn	1.100	vsn	1.117	fcdlsn	242.033	vdlsn	237.083
fcpn	1.583	vpn	2.383	fcdlpn	889.117	vdlpn	206.817
cdspn	38.233	fvdspn	40.100	clspn	6.450	fvlspn	4.883
cda	66.017	fvda	54.117	cla	9.983	fvla	14.883
fcsa	2.300	vsa	2.383	fcdlsa	346.900	vdlsa	344.317
fcpa	2.050	vpa	2.850	fcdlpa	192.633	vdlpa	270.800
cdspa	113.083	fvdspa	101.833	clspa	20.000	fvlspa	20.067
na	3.117	fcvna	3.117	dlna	342.500	fcvdlna	339.467
fdsna	68.083	cvdsna	63.833	flsna	14.017	cvlsna	9.733
fdpna	63.533	cvdpna	81.633	flpna	21.500	cvlpna	27.050
spna	2.533	fcvspna	2.617	dlspna	245.300	fcvdlspna	200.633
fcdb	85.050	vdb	85.983	fclb	14.500	vlb	14.750
csb	2.383	fvsb	2.367	cdlsb	236.617	fvdlsb	236.883
cpb	1.700	fvpb	2.383	cdlpb	74.517	fvdlpb	53.233
fcdspb	49.600	vdspb	22.150	fclspb	5.300	vlspb	5.000
fnb	1.100	cvnb	1.100	fdlnb	261.267	cvdlnb	234.083
dsnb	86.183	fcvdsnb	85.267	lsnb	14.500	fcvlsnb	14.450
dpnb	41.117	fcvdpnb	54.900	lpnb	11.200	fcvlpnb	10.583
fspnb	1.483	cvspnb	1.983	fdlspnb	201.450	cvdlspnb	167.350
fab	1.467	cvab	1.467	fdlab	358.300	cvdlab	353.400
dsab	56.517	fcvdsab	124.617	lsab	18.750	fcvlsab	9.733
dpab	55.583	fcvdpab	38.117	lpab	5.783	fcvlpab	6.367
fspab	1.433	cvspab	1.333	fdlspab	960.817	cvdlspab	254.283
fcdnab	112.650	vdnab	112.583	fclnab	20.650	vlnab	20.850
csnab	3.133	fvsnab	3.117	cdlsnab	728.783	fvdlsnab	768.850
cpnab	2.933	fvpnab	1.967	cdlpnab	1888.733	fvdlpnab	323.500
fcdspnab	107.850	vdspnab	103.883	fclspnab	20.000	vlspnab	15.700
fcdk	85.333	vdk	85.867	fclk	14.517	vlk	14.500
csk	2.383	fvsk	2.383	cdlsk	777.817	fvdlsk	791.917
cpk	0.717	fvpk	0.450	cdlpk	206.300	fvdlpk	297.850
fcdspk	39.000	vdspk	36.400	fclspk	6.183	vlspk	5.850

Treatment Outcomes (cont'd)							
Levels	Obs	Levels	Obs	Levels	Obs	Levels	Obs
fnk	1.300	cvnk	1.317	fdlnk	511.150	cvdlnk	503.600
dsnk	39.483	fcvdsnk	39.500	lsnk	6.733	fcvlsnk	6.633
dpnk	17.400	fcvdpnk	24.267	lpnk	2.283	fcvlpnk	5.300
fspnk	1.333	cvspnk	1.217	fdlspnk	359.683	cvdlspnk	455.383
fak	3.133	cvak	3.100	fdlak	404.117	cvdlak	341.600
dsak	112.717	fcvdsak	111.767	lsak	20.700	fcvlsak	20.583
dpak	83.250	fcvdpak	104.533	lpak	7.700	fcvlpak	10.517
fspak	3.033	cvspak	2.667	fdlspak	730.000	cvdlspak	495.667
fcdnak	53.100	vdnak	56.367	fclnak	9.600	vlnak	9.717
csnak	1.767	fvsnak	1.750	cdlsnak	1095.317	fvdlsnak	1134.300
cpnak	1.117	fvpnak	1.700	cdlpnak	619.600	fvdlpnak	401.283
fcdspnak	31.667	vdspnak	61.167	fclspnak	5.067	vlspnak	7.717
bk	1.333	fcvbk	1.533	dlbk	788.500	fcvdlbk	244.917
fdsbk	41.417	cvdsbk	49.867	flsbk	7.117	cvlsbk	8.767
fdpbk	33.117	cvdpbk	33.850	flpbk	3.267	cvlpbk	6.967
spbk	1.983	fcvspbk	1.200	dlspbk	157.017	fcvdlspbk	99.800
cdnbk	85.483	fvdnbk	85.967	clnbk	14.450	fvlnbk	14.650
fcsnbk	2.400	vsnbk	2.383	fcdlsnbk	510.900	vdlsnbk	510.883
fcpnbk	1.600	vpnbk	1.233	fcdlpnbk	181.183	vdlpnbk	180.517
cdspnbk	82.800	fvdspnbk	83.833	clspnbk	9.083	fvlspnbk	9.350
cdabk	112.400	fvdabk	111.867	clabk	20.667	fvlabk	20.750
fcsabk	3.117	vsabk	3.117	fcdlsabk	363.783	vdlsabk	368.183
fcpabk	1.867	vpabk	1.217	fcdlpabk	237.067	vdlpabk	148.450
cdspabk	61.467	fvdspabk	56.717	clspabk	5.733	fvlspabk	5.450
nabk	1.567	fcvnabk	1.550	dlnabk	719.500	fcvdlnabk	726.533
fdsnabk	57.050	cvdsnabk	56.383	flsnabk	10.200	cvlsnabk	10.183
fdpnabk	49.817	cvdpnabk	37.800	flpnabk	6.117	cvlpnabk	5.400
spnabk	1.550	fcvspnabk	1.417	dlspnabk	301.017	fcvdlspnabk	378.667

A.3 "G" Part

The following tables summarize effects and treatment outcomes for the "G" part under the push planner as described in section 6.3.3.

A.3.1 Success Rate

Effects

Factor Effects			
Factor(s)	Effect	Factor(s)	Effect
1 (0)	0.237	LN (90)	0.018
F (1)	-0.067	SN (A0)	-0.032
C (2)	-0.121	PN (C0)	-0.005
FC (3)	0.041	A (100)	0.007
V (4)	-0.275	FA (101)	-0.035
FV (5)	0.058	CA (102)	0.007
CV (6)	0.124	VA (104)	-0.050
D (8)	-0.073	DA (108)	0.009
FD (9)	-0.022	LA (110)	0.029
CD (A)	-0.009	SA (120)	0.019
VD (C)	0.035	PA (140)	-0.022
L (10)	-0.049	NA (180)	-0.050
FL (11)	-0.042	B (200)	-0.014
CL (12)	-0.010	FB (201)	0.004
VL (14)	0.054	CB (202)	-0.018
DL (18)	-0.029	VB (204)	0.050
S (20)	0.175	DB (208)	0.032
FS (21)	-0.019	LB (210)	-0.011
CS (22)	-0.047	SB (220)	-0.027
VS (24)	-0.127	PB (240)	-0.024
DS (28)	-0.000	NB (280)	-0.022
LS (30)	0.079	AB (300)	0.008
P (40)	-0.013	K (400)	-0.023
FP (41)	-0.067	FK (401)	-0.043

Factor Effects - cont'd			
Factor(s)	Effect	Factor(s)	Effect
CP (42)	0.006	CK (402)	0.045
VP (44)	-0.002	VK (404)	0.033
DP (48)	-0.011	DK (408)	-0.060
LP (50)	-0.007	LK (410)	-0.010
SP (60)	-0.033	SK (420)	-0.077
N (80)	-0.002	PK (440)	-0.027
FN (81)	-0.025	NK (480)	-0.055
CN (82)	-0.011	AK (500)	0.038
VN (84)	0.004	BK (600)	0.016
DN (88)	0.035		

Outcomes

Treatment Outcomes							
Levels	Obs	Levels	Obs	Levels	Obs	Levels	Obs
1	0.000	fcv	0.167	dl	0.063	fcvdl	0.024
fds	0.974	cvds	0.062	fls	0.950	cvls	0.173
fdp	0.000	cvdp	0.047	flp	0.000	cvlp	0.000
sp	1.000	fcvsp	0.278	dlsp	1.000	fcvdlsp	0.049
cdn	0.103	fvdn	0.131	cln	0.000	fvln	0.091
fcsn	0.635	vsn	0.000	fcdlsn	0.418	vdlsn	0.260
fcpn	0.644	vpn	0.201	fcdlpn	0.193	vdlpn	0.109
cdspn	0.455	fvdspn	0.075	clspn	0.783	fvlspn	0.203
cda	0.027	fvda	0.030	cla	0.069	fvla	0.036
fcsa	0.730	vsa	0.074	fcdlsa	0.272	vdlsa	0.033
fcpa	0.200	vpa	0.000	fcdlpa	0.000	vdlpa	0.028
cdspa	0.559	fvdspa	0.025	clspa	0.404	fvlspa	0.045
na	0.001	fcvna	0.007	dlna	0.491	fcvdlna	0.015
fdsna	0.958	cvdsna	0.055	flsna	0.988	cvlsna	0.345
fdpna	0.000	cvdpna	0.000	flpna	0.260	cvlpna	0.026
spna	1.000	fcvspna	0.085	dlspna	1.000	fcvdlspna	0.000
fcdb	0.246	vdb	0.181	fclb	0.048	vlb	0.045
csb	0.000	fvsb	0.000	cdlsb	0.001	fvdlsb	0.147
cpb	0.000	fvpb	0.259	cdlpb	0.000	fvdlpb	0.085
fcdspb	0.359	vdspb	0.109	fclspb	0.182	vlspb	0.125
fnb	0.702	cvnb	0.000	fdlnb	0.001	cvdlnb	0.020
dsnb	1.000	fcvdsnb	0.228	lsnb	1.000	fcvlsnb	0.266
dpnb	0.932	fcvdpnb	0.263	lpnb	0.000	fcvlpnb	0.004
fspnb	0.989	cvspnb	0.122	fdlspnb	0.445	cvdlspnb	0.043
fab	0.203	cvab	0.242	fdlab	0.003	cvdlab	0.000
dsab	0.971	fcvdsab	0.034	lsab	0.667	fcvlsab	0.096
dpab	0.692	fcvdpab	0.000	lpab	0.447	fcvlpab	0.000
fspab	0.467	cvspab	0.076	fdlspab	0.943	cvdlspab	0.219
fcdnab	0.000	vdnab	0.020	fclnab	0.173	vlnab	0.003
csnab	0.000	fvsnab	0.092	cdlsnab	0.703	fvdlsnab	0.164
cpnab	0.233	fvpnab	0.016	cdlpnab	0.000	fvdlpnab	0.048
fcdspnab	0.179	vdspnab	0.000	fclspnab	0.004	vlspnab	0.426
fcdk	0.000	vdk	0.079	fclk	0.101	vlk	0.012
csk	0.000	fvsk	0.088	cdlsk	0.432	fvdlsk	0.046
cpk	0.686	fvpk	0.113	cdlpk	0.010	fvdlpk	0.030
fcdspk	0.246	vdspk	0.196	fclspk	0.352	vlspk	0.122

Treatment Outcomes (cont'd)							
Levels	Obs	Levels	Obs	Levels	Obs	Levels	Obs
fnk	0.501	cvnk	0.111	fdlnk	0.083	cvdlnk	0.023
dsnk	0.000	fcvdsnk	0.044	lsnk	1.000	fcvlsnk	0.199
dpnk	1.000	fcvdpnk	0.184	lpnk	0.450	fcvlpnk	0.012
fspnk	0.174	cvspnk	0.199	fdlspnk	0.000	cvdlspnk	0.039
fak	0.460	cvak	0.152	fdlak	0.025	cvdlak	0.000
dsak	1.000	fcvdsak	0.064	lsak	1.000	fcvlsak	0.083
dpak	0.000	fcvdpak	0.038	lpak	0.986	fcvlpak	0.000
fspak	0.988	cvspak	0.108	fdlspak	0.073	cvdlspak	0.131
fcdnak	0.472	vdnak	0.000	fclnak	0.000	vlnak	0.003
csnak	0.753	fvsnak	0.007	cdlsnak	0.259	fvdlsnak	0.051
cpnak	0.499	fvpnak	0.000	cdlpnak	0.000	fvdlpnak	0.144
fcdspnak	0.000	vdspnak	0.118	fclspnak	0.269	vlspnak	0.103
bk	0.594	fcvbk	0.394	dlbk	0.000	fcvdlbk	0.022
fdsbk	0.471	cvdsbk	0.219	flsbk	0.865	cvlsbk	0.425
fdpbk	0.000	cvdpbk	0.000	flpbk	0.000	cvlpbk	0.000
spbk	1.000	fcvspbk	0.246	dlspbk	0.000	fcvdlspbk	0.043
cdnbk	0.365	fvdnbk	0.178	clnbk	0.075	fvlnbk	0.030
fcsnbk	0.094	vsnbk	0.004	fcdlsnbk	0.002	vdlsnbk	0.250
fcpnbk	0.000	vpnbk	0.025	fcdlpnbk	0.002	vdlpnbk	0.295
cdspnbk	0.000	fvdspnbk	0.048	clspnbk	0.509	fvlspnbk	0.275
cdabk	0.332	fvdabk	0.030	clabk	0.159	fvlabk	0.521
fcsabk	0.523	vsabk	0.153	fcdlsabk	0.517	vdlsabk	0.089
fcpabk	0.594	vpabk	0.095	fcdlpabk	0.000	vdlpabk	0.000
cdspabk	0.421	fvdspabk	0.043	clspabk	0.825	fvlspabk	0.020
nabk	1.000	fcvnabk	0.039	dlnabk	0.000	fcvdlnabk	0.000
fdsnabk	0.806	cvdsnabk	0.071	flsnabk	0.000	cvlsnabk	0.386
fdpnabk	0.000	cvdpnabk	0.161	flpnabk	0.177	cvlpnabk	0.000
spnabk	0.000	fcvspnabk	0.096	dlspnabk	0.999	fcvdlspnabk	0.000

A.3.2 Planning Time

Effects

Factor Effects			
Factor(s)	Effect	Factor(s)	Effect
1 (0)	56.532	LN (90)	-3.222
F (1)	0.344	SN (A0)	6.880
C (2)	4.083	PN (C0)	2.824
FC (3)	16.140	A (100)	30.577
V (4)	5.709	FA (101)	1.109
FV (5)	-3.364	CA (102)	4.509
CV (6)	7.787	VA (104)	7.603
D (8)	104.607	DA (108)	29.184
FD (9)	0.814	LA (110)	20.748
CD (A)	4.361	SA (120)	4.779
VD (C)	5.678	PA (140)	-6.182
L (10)	67.405	NA (180)	3.232
FL (11)	0.758	B (200)	6.613
CL (12)	6.021	FB (201)	-4.840
VL (14)	5.305	CB (202)	-2.554
DL (18)	62.665	VB (204)	-3.271
S (20)	20.716	DB (208)	5.540
FS (21)	2.609	LB (210)	4.045
CS (22)	1.355	SB (220)	-1.998
VS (24)	7.966	PB (240)	-7.931
DS (28)	20.217	NB (280)	-0.734
LS (30)	17.912	AB (300)	-6.356
P (40)	-30.149	K (400)	-18.921
FP (41)	-1.372	FK (401)	-1.607
CP (42)	4.232	CK (402)	-3.906
VP (44)	2.702	VK (404)	-8.330
DP (48)	-28.435	DK (408)	-17.901
LP (50)	-16.376	LK (410)	-15.671
SP (60)	13.511	SK (420)	-10.915
N (80)	2.573	PK (440)	-0.505
FN (81)	8.088	NK (480)	2.584
CN (82)	6.588	AK (500)	-15.874
VN (84)	2.112	BK (600)	2.780
DN (88)	1.932		

Outcomes

Treatment Outcomes							
Levels	Obs	Levels	Obs	Levels	Obs	Levels	Obs
1	1.500	fcv	3.033	dl	165.883	fcvdl	160.450
fds	31.767	cvds	17.317	fls	3.933	cvls	2.383
fdp	31.000	cvdp	5.700	flp	1.267	cvlp	0.900
sp	0.517	fcvsp	0.950	dlsp	130.500	fcvdlsp	77.917
cdn	24.200	fvdn	34.533	cln	5.167	fvln	4.550
fcsn	1.033	vsn	1.100	fcdlsn	155.500	vdlsn	151.667
fcpn	0.433	vpn	0.600	fcdlpn	94.683	vdlpn	19.917
cdspn	7.817	fvdspn	17.483	clspn	4.717	fvlspn	3.100
cda	37.817	fvda	40.250	cla	4.150	fvla	4.633
fcsa	1.250	vsa	1.333	fcdlsa	441.350	vdlsa	440.450
fcpa	1.750	vpa	3.867	fcdlpa	61.350	vdlpa	177.633
cdspa	58.133	fvdspa	56.567	clspa	11.567	fvlspa	11.883
na	3.800	fcvna	3.867	dlna	246.800	fcvdlna	246.800
fdsna	47.150	cvdsna	46.583	flsna	4.933	cvlsna	7.050
fdpna	14.583	cvdpna	50.233	flpna	3.117	cvlpna	5.467
spna	0.900	fcvspna	0.883	dlspna	204.250	fcvdlspna	698.933
fcdb	16.433	vdb	30.100	fclb	5.717	vlb	5.250
csb	1.083	fvsb	1.117	cdlsb	96.233	fvdlsb	295.083
cpb	1.117	fvpb	2.333	cdlpb	164.750	fvdlpb	45.483
fcdspb	39.867	vdspb	51.483	fclspb	5.817	vlspb	6.517
fnb	2.967	cvnb	3.333	fdlnb	99.633	cvdlnb	87.833
dsnb	92.167	fcvdsnb	90.983	lsnb	12.933	fcvlsnb	13.917
dpnb	15.700	fcvdpnb	13.667	lpnb	7.233	fcvlpnb	6.400
fspnb	2.967	cvspnb	1.567	fdlspnb	113.250	cvdlspnb	242.633
fab	2.700	cvab	3.750	fdlab	315.400	cvdlab	467.850
dsab	83.350	fcvdsab	92.633	lsab	18.867	fcvlsab	11.817
dpab	41.300	fcvdpab	36.950	lpab	13.650	fcvlpab	3.167
fspab	1.967	cvspab	2.350	fdlspab	178.583	cvdlspab	192.167
fcdnab	123.133	vdnab	79.233	fclnab	11.117	vlnab	13.433
csnab	3.800	fvsnab	3.883	cdlsnab	373.300	fvdlsnab	370.850
cpnab	2.450	fvpnab	1.767	cdlpnab	68.033	fvdlpnab	59.483
fcdspnab	47.317	vdspnab	123.667	fclspnab	8.600	vlspnab	14.050
fcdk	90.150	vdk	86.533	fclk	8.800	vlk	9.150
csk	2.767	fvsk	1.917	cdlsk	90.700	fvdlsk	81.817
cpk	0.967	fvpk	0.900	cdlpk	25.267	fvdlpk	25.867
fcdspk	22.500	vdspk	27.550	fclspk	6.500	vlspk	5.350

Treatment Outcomes (cont'd)							
Levels	Obs	Levels	Obs	Levels	Obs	Levels	Obs
fnk	1.350	cvnk	2.083	fdlnk	146.533	cvdlnk	83.483
dsnk	63.183	fcvdsnk	60.583	lsnk	9.467	fcvlsnk	5.550
dpnk	8.583	fcvdpnk	7.817	lpnk	2.733	fcvlpnk	1.350
fspnk	0.483	cvspnk	1.000	fdlspnk	56.967	cvdlspnk	125.133
fak	1.367	cvak	1.367	fdlak	142.067	cvdlak	169.383
dsak	39.933	fcvdsak	40.100	lsak	6.133	fcvlsak	4.000
dpak	5.733	fcvdpak	8.950	lpak	1.250	fcvlpak	2.933
fspak	1.167	cvspak	1.200	fdlspak	182.050	cvdlspak	217.250
fcdnak	127.050	vdnak	125.550	fclnak	19.250	vlnak	11.500
csnak	3.783	fvsnak	3.750	cdlsnak	171.950	fvdlsnak	163.917
cpnak	1.633	fvpnak	3.167	cdlpnak	40.133	fvdlpnak	56.750
fcdspnak	33.400	vdspnak	54.450	fclspnak	2.383	vlspnak	5.417
bk	1.017	fcvbk	1.350	dlbk	196.250	fcvdlbk	165.233
fdsbk	33.717	cvdsbk	59.500	flsbk	5.750	cvlsbk	9.617
fdpbk	12.650	cvdpbk	9.083	flpbk	1.867	cvlpbk	1.717
spbk	1.783	fcvspbk	2.200	dlspbk	176.083	fcvdlspbk	92.717
cdnbk	30.600	fvdnbk	33.450	clnbk	5.233	fvlnbk	8.267
fcsnbk	1.100	vsnbk	1.100	fcdlsnbk	262.617	vdlsnbk	228.833
fcpnbk	2.167	vpnbk	0.950	fcdlpnbk	76.250	vdlpnbk	20.167
cdspnbk	16.617	fvdspnbk	15.033	clspnbk	1.733	fvlspnbk	4.383
cdabk	46.750	fvdabk	27.483	clabk	6.150	fvlabk	6.967
fcsabk	1.417	vsabk	1.350	fcdlsabk	222.850	vdlsabk	224.400
fcpabk	1.367	vpabk	0.933	fcdlpabk	86.233	vdlpabk	90.083
cdspabk	24.633	fvdspabk	16.050	clspabk	4.183	fvlspabk	4.567
nabk	1.150	fcvnabk	0.900	dlnabk	263.483	fcvdlnabk	277.033
fdsnabk	42.700	cvdsnabk	39.583	flsnabk	6.267	cvlsnabk	6.217
fdpnabk	81.783	cvdpnabk	12.417	flpnabk	2.700	cvlpnabk	7.633
spnabk	2.700	fcvspnabk	2.600	dlspnabk	160.433	fcvdlspnabk	161.133

Appendix B

Treatments and Outcomes - Vision Orienter

B.1 Triangle

The following tables summarize effects and treatment outcomes for the triangle part under the vision orienter as described in section 6.4.1.

B.1.1 Success Rate

Effects

Factor Effects			
Factor(s)	Effect	Factor(s)	Effect
1 (0)	0.084	MZ (41)	-0.002
M (1)	0.000	DZ (42)	-0.004
D (2)	-0.005	SZ (44)	-0.005
MD (3)	-0.002	CZ (48)	-0.003
S (4)	-0.000	AZ (50)	-0.006
MS (5)	-0.007	LZ (60)	0.009
DS (6)	0.004	P (80)	0.015
C (8)	0.001	MP (81)	-0.000
MC (9)	0.001	DP (82)	-0.005
DC (A)	-0.000	SP (84)	0.005
SC (C)	-0.001	CP (88)	-0.004
A (10)	0.005	AP (90)	0.004
MA (11)	-0.002	LP (A0)	-0.000
DA (12)	-0.003	ZP (C0)	-0.004
SA (14)	0.002	Q (100)	-0.008
CA (18)	0.011	MQ (101)	0.005
L (20)	-0.013	DQ (102)	-0.004
ML (21)	0.018	SQ (104)	0.009
DL (22)	0.000	CQ (108)	0.002
SL (24)	0.007	AQ (110)	-0.004
CL (28)	0.004	LQ (120)	0.006
AL (30)	-0.002	ZQ (140)	-0.003
Z (40)	-0.007	PQ (180)	-0.008

Outcomes

Treatment Outcomes							
Levels	Obs	Levels	Obs	Levels	Obs	Levels	Obs
1	0.100	md	0.108	ms	0.058	ds	0.100
ca	0.133	mdca	0.100	msca	0.092	dsca	0.092
cl	0.050	mdcl	0.083	mscl	0.058	dscl	0.050
al	0.033	mdal	0.067	msal	0.067	dsal	0.050
mcz	0.083	dcz	0.067	scz	0.042	mdscz	0.042
maz	0.092	daz	0.092	saz	0.075	mdsaz	0.042
mlz	0.100	dlz	0.067	slz	0.067	mdslz	0.083
mcalz	0.083	dcalz	0.083	scalz	0.075	mdscalz	0.117
mcp	0.083	dcp	0.142	scp	0.092	mdscp	0.092
map	0.125	dap	0.142	sap	0.133	mdsap	0.125
mlp	0.067	dlp	0.075	slp	0.083	mdslp	0.117
mcalp	0.075	dcalp	0.100	scalp	0.092	mdscalp	0.100
zp	0.150	mdzp	0.083	mszp	0.100	dszp	0.100
cazp	0.167	mdcazp	0.075	mscazp	0.108	dscazp	0.100
clzp	0.092	mdclzp	0.067	msclzp	0.108	dsclzp	0.067
alzp	0.100	mdalzp	0.083	msalzp	0.067	dsalzp	0.083
mcq	0.092	dcq	0.083	scq	0.108	mdscq	0.075
maq	0.075	daq	0.075	saq	0.117	mdsaq	0.050
mlq	0.075	dlq	0.050	slq	0.058	mdslq	0.067
mcalq	0.133	dcalq	0.033	scalq	0.100	mdscalq	0.067
zq	0.083	mdzq	0.092	mszq	0.067	dszq	0.125
cazq	0.083	mdcazq	0.092	mscazq	0.042	dscazq	0.075
clzq	0.042	mdclzq	0.067	msclzq	0.083	dsclzq	0.100
alzq	0.042	mdalzq	0.067	msalzq	0.033	dsalzq	0.083
pq	0.100	mdpq	0.075	mspq	0.083	dspq	0.092
capq	0.050	mdcapq	0.100	mscapq	0.092	dscapq	0.183
clpq	0.075	mdclpq	0.100	msclpq	0.083	dsclpq	0.058
alpq	0.058	mdalpq	0.075	msalpq	0.183	dsalpq	0.058
mczpq	0.092	dczpq	0.042	sczpq	0.092	mdsczpq	0.050
mazpq	0.083	dazpq	0.058	sazpq	0.100	mdsazpq	0.050
mlzpq	0.108	dlzpq	0.058	slzpq	0.067	mdslzpq	0.100
mcalzpq	0.117	dcalzpq	0.042	scalzpq	0.083	mdscalzpq	0.075

B.1.2 Extraction Time

Effects

Factor Effects			
Factor(s)	Effect	Factor(s)	Effect
1 (0)	25.809	MZ (41)	0.507
M (1)	9.024	DZ (42)	0.040
D (2)	0.177	SZ (44)	0.006
MD (3)	0.111	CZ (48)	0.008
S (4)	-0.248	AZ (50)	0.756
MS (5)	-0.336	LZ (60)	0.033
DS (6)	0.190	P (80)	3.120
C (8)	-0.158	MP (81)	0.098
MC (9)	-0.048	DP (82)	0.017
DC (A)	-0.067	SP (84)	0.119
SC (C)	-0.153	CP (88)	0.000
A (10)	-0.574	AP (90)	-0.209
MA (11)	-0.681	LP (A0)	0.167
DA (12)	0.031	ZP (C0)	-0.192
SA (14)	-0.221	Q (100)	3.103
CA (18)	-0.034	MQ (101)	0.460
L (20)	1.276	DQ (102)	0.113
ML (21)	-0.095	SQ (104)	0.018
DL (22)	-0.054	CQ (108)	-0.066
SL (24)	0.408	AQ (110)	0.052
CL (28)	-0.241	LQ (120)	0.362
AL (30)	0.346	ZQ (140)	0.388
Z (40)	0.567	PQ (180)	1.506

Outcomes

Treatment Outcomes							
Levels	Obs	Levels	Obs	Levels	Obs	Levels	Obs
1	17.737	md	29.060	ms	27.009	ds	18.506
ca	19.543	mdca	25.702	msca	24.162	dsca	19.537
cl	20.236	mdcl	28.356	mscl	28.401	dscl	19.022
al	20.453	mdal	26.569	msal	27.237	dsal	19.573
mcz	29.596	dcz	18.650	scz	18.910	mdscz	27.265
maz	28.811	daz	19.481	saz	18.078	mdsaz	26.669
mlz	27.526	dlz	19.696	slz	19.809	mdslz	28.752
mcalz	28.653	dcalz	19.606	scalz	19.419	mdscalz	28.374
mcp	30.732	dcp	20.247	scp	20.101	mdscp	30.583
map	26.994	dap	20.033	sap	20.936	mdsap	26.502
mlp	30.410	dlp	22.039	slp	22.122	mdslp	31.911
mcalp	28.813	dcalp	21.689	scalp	20.416	mdscalp	28.739
zp	19.393	mdzp	30.390	mszp	29.678	dszp	19.703
cazp	19.812	mdcazp	29.400	mscazp	28.077	dscazp	20.313
clzp	20.855	mdclzp	29.019	msclzp	29.361	dsclzp	21.161
alzp	21.198	mdalzp	29.782	msalzp	30.728	dsalzp	20.925
mcq	30.140	dcq	19.371	scq	20.040	mdscq	29.469
maq	27.545	daq	18.683	saq	20.088	mdsaq	26.873
mlq	30.991	dlq	20.843	slq	20.295	mdslq	30.764
mcalq	28.639	dcalq	20.073	scalq	20.077	mdscalq	28.834
zq	18.828	mdzq	30.950	mszq	28.929	dszq	19.935
cazq	20.176	mdcazq	31.136	mscazq	27.539	dscazq	19.797
clzq	20.454	mdclzq	29.848	msclzq	30.640	dsclzq	21.767
alzq	21.801	mdalzq	32.691	msalzq	31.658	dsalzq	22.662
pq	24.261	mdpq	35.449	mspq	33.626	dspq	24.569
capq	23.240	mdcapq	33.240	mscapq	28.753	dscapq	24.205
clpq	26.355	mdclpq	34.561	msclpq	36.023	dsclpq	24.857
alpq	25.565	mdalpq	33.309	msalpq	33.688	dsalpq	25.839
mczpq	35.602	dczpq	23.102	sczpq	24.494	mdsczpq	33.865
mazpq	34.908	dazpq	23.745	sazpq	24.045	mdsazpq	32.722
mlzpq	34.278	dlzpq	26.521	slzpq	25.896	mdslzpq	37.169
mcalzpq	36.521	dcalzpq	26.400	scalzpq	25.817	mdscalzpq	36.922

B.2 "L" Part

The following tables summarize effects and treatment outcomes for the "L" part under the vision orienter as described in section 6.4.2.

B.2.1 Success Rate

Effects

Factor Effects			
Factor(s)	Effect	Factor(s)	Effect
1 (0)	0.282	MZ (41)	0.008
M (1)	-0.093	DZ (42)	-0.002
D (2)	0.018	SZ (44)	0.001
MD (3)	-0.009	CZ (48)	-0.002
S (4)	0.128	AZ (50)	-0.010
MS (5)	0.019	LZ (60)	0.006
DS (6)	0.006	P (80)	0.078
C (8)	0.004	MP (81)	-0.024
MC (9)	-0.003	DP (82)	-0.012
DC (A)	-0.007	SP (84)	0.011
SC (C)	-0.004	CP (88)	-0.004
A (10)	0.008	AP (90)	-0.003
MA (11)	-0.008	LP (A0)	-0.016
DA (12)	-0.005	ZP (C0)	-0.006
SA (14)	-0.003	Q (100)	0.003
CA (18)	0.009	MQ (101)	-0.025
L (20)	0.011	DQ (102)	-0.003
ML (21)	-0.012	SQ (104)	0.011
DL (22)	0.004	CQ (108)	0.000
SL (24)	0.038	AQ (110)	0.004
CL (28)	0.001	LQ (120)	0.021
AL (30)	-0.004	ZQ (140)	0.005
Z (40)	-0.013	PQ (180)	-0.022

Outcomes

Treatment Outcomes							
Levels	Obs	Levels	Obs	Levels	Obs	Levels	Obs
1	0.175	md	0.192	ms	0.233	ds	0.317
ca	0.192	mdca	0.208	msca	0.283	dsca	0.342
cl	0.200	mdcl	0.133	mscl	0.217	dscl	0.358
al	0.158	mdal	0.125	msal	0.267	dsal	0.417
mcz	0.150	dcz	0.200	scz	0.258	mdscz	0.275
maz	0.183	daz	0.158	saz	0.225	mdsaz	0.267
mlz	0.150	dlz	0.192	slz	0.292	mdslz	0.292
mcalz	0.133	dcalz	0.208	scalz	0.292	mdscalz	0.275
mcp	0.275	dcp	0.383	scp	0.375	mdscp	0.342
map	0.225	dap	0.408	sap	0.375	mdsap	0.400
mlp	0.142	dlp	0.333	slp	0.442	mdslp	0.408
mcalp	0.192	dcalp	0.292	scalp	0.442	mdscalp	0.400
zp	0.375	mdzp	0.200	mszp	0.350	dszp	0.408
cazp	0.433	mdcazp	0.200	mscazp	0.333	dscazp	0.392
clzp	0.250	mdclzp	0.175	msclzp	0.400	dsclzp	0.417
alzp	0.300	mdalzp	0.158	msalzp	0.308	dsalzp	0.442
mcq	0.125	dcq	0.225	scq	0.292	mdscq	0.225
maq	0.192	daq	0.242	saq	0.350	mdsaq	0.183
mlq	0.100	dlq	0.317	slq	0.308	mdslq	0.300
mcalq	0.133	dcalq	0.292	scalq	0.475	mdscalq	0.308
zq	0.175	mdzq	0.142	mszq	0.250	dszq	0.342
cazq	0.175	mdcazq	0.150	mscazq	0.242	dscazq	0.367
clzq	0.250	mdclzq	0.192	msclzq	0.283	dsclzq	0.425
alzq	0.242	mdalzq	0.175	msalzq	0.283	dsalzq	0.408
pq	0.300	mdpq	0.167	mspq	0.358	dspq	0.433
capq	0.417	mdcapq	0.183	mscapq	0.317	dscapq	0.417
clpq	0.300	mdclpq	0.117	msclpq	0.383	dsclpq	0.467
alpq	0.308	mdalpq	0.133	msalpq	0.350	dsalpq	0.483
mczpq	0.133	dczpq	0.333	sczpq	0.342	mdsczpq	0.300
mazpq	0.133	dazpq	0.350	sazpq	0.367	mdsazpq	0.333
mlzpq	0.133	dlzpq	0.258	slzpq	0.500	mdslzpq	0.392
mcalzpq	0.125	dcalzpq	0.317	scalzpq	0.483	mdscalzpq	0.333

B.2.2 Extraction Time

Effects

Factor Effects			
Factor(s)	Effect	Factor(s)	Effect
1 (0)	22.786	MZ (41)	0.524
M (1)	6.369	DZ (42)	-0.074
D (2)	0.322	SZ (44)	0.058
MD (3)	0.165	CZ (48)	0.028
S (4)	1.434	AZ (50)	0.089
MS (5)	0.102	LZ (60)	0.014
DS (6)	-0.100	P (80)	2.221
C (8)	0.011	MP (81)	1.014
MC (9)	0.055	DP (82)	-0.012
DC (A)	-0.019	SP (84)	0.177
SC (C)	-0.270	CP (88)	0.174
A (10)	0.064	AP (90)	-0.157
MA (11)	-0.463	LP (A0)	-0.173
DA (12)	0.051	ZP (C0)	-0.417
SA (14)	-0.100	Q (100)	2.240
CA (18)	0.150	MQ (101)	-0.272
L (20)	-0.512	DQ (102)	-0.055
ML (21)	0.319	SQ (104)	-0.063
DL (22)	-0.085	CQ (108)	-0.309
SL (24)	-1.548	AQ (110)	0.161
CL (28)	0.134	LQ (120)	-0.119
AL (30)	-0.286	ZQ (140)	0.109
Z (40)	0.769	PQ (180)	-0.345

Outcomes

Treatment Outcomes							
Levels	Obs	Levels	Obs	Levels	Obs	Levels	Obs
1	14.717	md	22.300	ms	23.715	ds	19.010
ca	16.960	mdca	22.427	msca	23.417	dsca	19.311
cl	17.630	mdcl	22.804	mscl	22.484	dscl	16.261
al	17.360	mdal	20.313	msal	21.917	dsal	16.918
mcz	21.938	dcz	17.891	scz	18.634	mdscz	23.990
maz	21.143	daz	17.708	saz	19.853	mdsaz	24.022
mlz	23.131	dlz	17.949	slz	17.159	mdslz	24.898
mcalz	24.158	dcalz	17.914	scalz	18.324	mdscalz	25.994
mcp	25.077	dcp	18.190	scp	21.692	mdscp	29.978
map	24.194	dap	17.441	sap	22.144	mdsap	28.463
mlp	26.227	dlp	18.313	slp	18.337	mdslp	28.301
mcalp	25.342	dcalp	18.464	scalp	18.609	mdscalp	24.427
zp	17.969	mdzp	24.859	mszp	28.040	dszp	20.687
cazp	17.546	mdcazp	27.097	mscazp	28.286	dscazp	20.895
clzp	17.877	mdclzp	28.424	msclzp	28.077	dsclzp	18.299
alzp	18.592	mdalzp	27.382	msalzp	26.724	dsalzp	18.397
mcq	22.141	dcq	17.201	scq	20.477	mdscq	25.240
maq	23.478	daq	20.373	saq	21.383	mdsaq	27.528
mlq	25.593	dlq	20.661	slq	18.991	mdslq	24.989
mcalq	24.423	dcalq	20.907	scalq	19.520	mdscalq	24.179
zq	19.149	mdzq	26.015	mszq	31.471	dszq	22.213
cazq	19.713	mdcazq	25.691	mscazq	27.258	dscazq	22.671
clzq	20.121	mdclzq	25.576	msclzq	26.399	dsclzq	19.338
alzq	20.525	mdalzq	25.577	msalzq	25.082	dsalzq	21.091
pq	19.685	mdpq	25.682	mspq	28.962	dspq	23.624
capq	20.778	mdcapq	27.185	mscapq	28.259	dscapq	24.223
clpq	21.314	mdclpq	28.955	msclpq	27.615	dsclpq	21.077
alpq	21.010	mdalpq	27.684	msalpq	27.054	dsalpq	20.752
mczpq	27.438	dczpq	19.853	sczpq	23.558	mdsczpq	30.348
mazpq	27.264	dazpq	20.515	sazpq	24.487	mdsazpq	31.223
mlzpq	28.067	dlzpq	20.833	slzpq	20.107	mdslzpq	28.698
mcalzpq	28.881	dcalzpq	20.771	scalzpq	20.519	mdscalzpq	28.590

B.3 "G" Part

The following tables summarize effects and treatment outcomes for the "G" part under the vision orienter as described in section 6.4.3.

B.3.1 Success Rate

Effects

Factor Effects			
Factor(s)	Effect	Factor(s)	Effect
1 (0)	0.223	MZ (41)	-0.008
M (1)	-0.442	DZ (42)	0.008
D (2)	0.018	SZ (44)	-0.011
MD (3)	-0.018	CZ (48)	-0.009
S (4)	-0.042	AZ (50)	0.002
MS (5)	0.039	LZ (60)	-0.001
DS (6)	-0.008	P (80)	0.114
C (8)	-0.006	MP (81)	-0.112
MC (9)	0.005	DP (82)	0.005
DC (A)	-0.001	SP (84)	-0.029
SC (C)	-0.008	CP (88)	-0.005
A (10)	0.013	AP (90)	0.003
MA (11)	-0.012	LP (A0)	0.007
DA (12)	0.007	ZP (C0)	0.001
SA (14)	0.002	Q (100)	-0.095
CA (18)	-0.003	MQ (101)	0.093
L (20)	0.050	DQ (102)	0.004
ML (21)	-0.051	SQ (104)	0.020
DL (22)	-0.005	CQ (108)	-0.002
SL (24)	-0.010	AQ (110)	0.001
CL (28)	-0.008	LQ (120)	-0.014
AL (30)	-0.013	ZQ (140)	-0.001
Z (40)	0.007	PQ (180)	0.150

Outcomes

Treatment Outcomes							
Levels	Obs	Levels	Obs	Levels	Obs	Levels	Obs
1	0.450	md	0.008	ms	0.000	ds	0.400
ca	0.500	mdca	0.008	msca	0.000	dsca	0.525
cl	0.700	mdcl	0.000	mscl	0.000	dscl	0.592
al	0.667	mdal	0.000	msal	0.000	dsal	0.642
mcz	0.000	dcz	0.533	scz	0.408	mdscz	0.000
maz	0.011	daz	0.608	saz	0.458	mdsaz	0.000
mlz	0.000	dlz	0.717	slz	0.675	mdslz	0.000
mcalz	0.000	dcalz	0.750	scalz	0.583	mdscalz	0.000
mcp	0.008	dcp	0.508	scp	0.375	mdscp	0.000
map	0.000	dap	0.550	sap	0.408	mdsap	0.008
mlp	0.008	dlp	0.667	slp	0.433	mdslp	0.008
mcalp	0.008	dcalp	0.625	scalp	0.425	mdscalp	0.000
zp	0.500	mdzp	0.000	mszp	0.000	dszp	0.392
cazp	0.525	mdcazp	0.000	mscazp	0.000	dscazp	0.442
clzp	0.650	mdclzp	0.009	msclzp	0.000	dsclzp	0.417
alzp	0.625	mdalzp	0.000	msalzp	0.000	dsalzp	0.475
mcq	0.000	dcq	0.108	scq	0.117	mdscq	0.000
maq	0.000	daq	0.050	saq	0.108	mdsaq	0.000
mlq	0.000	dlq	0.092	slq	0.092	mdslq	0.000
mcalq	0.000	dcalq	0.075	scalq	0.083	mdscalq	0.000
zq	0.042	mdzq	0.000	mszq	0.000	dszq	0.092
cazq	0.067	mdcazq	0.000	mscazq	0.000	dscazq	0.125
clzq	0.058	mdclzq	0.000	msclzq	0.000	dsclzq	0.075
alzq	0.075	mdalzq	0.000	msalzq	0.000	dsalzq	0.125
pq	0.458	mdpq	0.008	mspq	0.000	dspq	0.500
capq	0.575	mdcapq	0.008	mscapq	0.000	dscapq	0.525
clpq	0.725	mdclpq	0.000	msclpq	0.000	dsclpq	0.592
alpq	0.717	mdalpq	0.000	msalpq	0.000	dsalpq	0.692
mczpq	0.000	dczpq	0.592	sczpq	0.417	mdsczpq	0.000
mazpq	0.011	dazpq	0.708	sazpq	0.525	mdsazpq	0.000
mlzpq	0.000	dlzpq	0.792	slzpq	0.642	mdslzpq	0.000
mcalzpq	0.000	dcalzpq	0.800	scalzpq	0.550	mdscalzpq	0.000

B.3.2 Extraction Time

Effects

Factor Effects			
Factor(s)	Effect	Factor(s)	Effect
1 (0)	22.052	MZ (41)	-0.279
M (1)	11.955	DZ (42)	-0.385
D (2)	-0.335	SZ (44)	-0.888
MD (3)	0.448	CZ (48)	0.581
S (4)	-0.250	AZ (50)	-0.397
MS (5)	0.186	LZ (60)	-1.115
DS (6)	1.874	P (80)	1.113
C (8)	-0.656	MP (81)	0.164
MC (9)	-0.153	DP (82)	0.370
DC (A)	1.068	SP (84)	0.638
SC (C)	1.367	CP (88)	0.295
A (10)	0.343	AP (90)	0.385
MA (11)	1.035	LP (A0)	1.301
DA (12)	-0.617	ZP (C0)	2.896
SA (14)	-0.980	Q (100)	-2.749
CA (18)	0.186	MQ (101)	-1.037
L (20)	-3.474	DQ (102)	0.024
ML (21)	-2.553	SQ (104)	0.294
DL (22)	0.185	CQ (108)	0.480
SL (24)	0.150	AQ (110)	-0.424
CL (28)	1.309	LQ (120)	0.720
AL (30)	-0.597	ZQ (140)	-3.737
Z (40)	-2.678	PQ (180)	-7.482

Outcomes

Treatment Outcomes							
Levels	Obs	Levels	Obs	Levels	Obs	Levels	Obs
1	22.360	md	24.309	ms	23.889	ds	12.873
ca	12.572	mdca	26.551	msca	24.488	dsca	13.150
cl	12.565	mdcl	19.348	mscl	20.408	dscl	12.707
al	12.490	mdal	19.602	msal	20.352	dsal	12.779
mcz	23.447	dcz	15.403	scz	12.932	mdscz	37.156
maz	52.660	daz	13.933	saz	12.557	mdsaz	22.852
mlz	19.470	dlz	12.519	slz	12.427	mdslz	19.716
mcalz	19.033	dcalz	12.377	scalz	12.697	mdscalz	22.485
mcp	24.329	dcp	14.760	scp	19.435	mdscp	36.245
map	41.315	dap	21.411	sap	22.057	mdsap	41.314
mlp	33.710	dlp	21.468	slp	21.485	mdslp	34.463
mcalp	33.644	dcalp	21.444	scalp	21.179	mdscalp	33.978
zp	21.164	mdzp	39.383	mszp	40.089	dszp	22.136
cazp	21.000	mdcazp	39.397	mscazp	39.589	dscazp	21.975
clzp	21.204	mdclzp	31.935	msclzp	32.928	dsclzp	21.094
alzp	21.004	mdalzp	32.527	msalzp	26.898	dsalzp	12.596
mcq	42.196	dcq	21.540	scq	22.646	mdscq	43.037
maq	43.156	daq	22.102	saq	22.248	mdsaq	42.889
mlq	34.431	dlq	21.639	slq	22.097	mdslq	36.014
mcalq	34.238	dcalq	21.217	scalq	21.906	mdscalq	35.232
zq	21.091	mdzq	24.605	mszq	23.080	dszq	12.920
cazq	12.330	mdcazq	23.325	mscazq	23.305	dscazq	12.651
clzq	12.224	mdclzq	18.739	msclzq	18.748	dsclzq	11.982
alzq	11.970	mdalzq	18.616	msalzq	19.126	dsalzq	12.290
pq	12.178	mdpq	23.234	mspq	23.871	dspq	12.808
capq	12.456	mdcapq	23.761	mscapq	24.011	dscapq	13.047
clpq	12.414	mdclpq	19.504	msclpq	20.191	dsclpq	14.514
alpq	12.598	mdalpq	19.750	msalpq	20.789	dsalpq	12.601
mczpq	23.445	dczpq	12.388	sczpq	13.416	mdsczpq	23.827
mazpq	23.260	dazpq	12.572	sazpq	12.990	mdsazpq	25.356
mlzpq	19.105	dlzpq	12.378	slzpq	12.678	mdslzpq	20.038
mcalzpq	25.936	dcalzpq	12.568	scalzpq	12.529	mdscalzpq	19.548