Intentionally left blank

TABLE OF CONTENTS

ABSTRACT

This thesis presents a learning framework for resource constrained parameter tuning of a general stereo vision algorithm. Stereo vision is an inexpensive method for calculating depth information in a scene by triangulating features from two different viewpoints. This depth information is an important part of 3-D object recognition used in image understanding applications. One of the significant difficulties in stereo vision, as well as in other vision algorithms, lies in adjusting the variety of parameters to maximize performance. Typically, this task is the responsibility of the designer, and it can entail extensive work in tuning the algorithm over the given image domain. This effort is performed with a minimum of formal guidelines, relying primarily on the skill and patience of the designer. The solution proposed here is a system called TEACHER 4.2 (TEchniques for the Automatic Creation of HEuRistics). It consists of a statistical method for systematically and intelligently improving the parameters of the vision algorithm. This method operates under a user-specified deadline, automatically making the trade-off between the number of new parameter sets to consider and the degree of testing to perform on each. Testing time is divided into stages in which subtests are performed on the parameter sets. The first stage generates new parameter sets as necessary and performs initial tests. Successive stages select a subset of the top parameter sets from the previous stage and continue testing. The final result is the parameter set deemed best by the final stage. Various scheduling strategies are explored within the stage framework, including one based on minimizing the statistical risk of the performance of the resulting parameter set being incorrect. The results obtained by this approach show that a significant performance improvement can be obtained by automatic and systematic exploration of the algorithm parameter space.

ACKNOWLEDGEMENTS

I would like to thank my wife, Jill, for all the support she has given me during the past two years. She has made the difficult times bearable, and has always helped me keep life in perspective. In addition, she has helped many times, and in many ways, with the completion of this thesis; without her, this thesis could not have been completed in a timely fashion. I would also like to thank my advisor, Professor Benjamin Wah, for his insightful comments and guidance. His dedication has played an integral role in making this thesis possible.

I would also like to offer my thanks to Arthur Ieumwananonthachai and Akiko N. Aizawa. It was our work together that formed the foundation for this thesis. Dr. Banavar Sridhar of NASA Ames Research Center was also helpful in supplying initial test images. Additional thanks go out to my officemates Kumar Ganapathy and Vijay Karamcheti for their refreshing conversation and entertaining anecdotes on the proper approach to life. Further thanks go to the entire group for their helpful comments and criticism.

I would also like to gratefully acknowledge the financial support given by the National Aeronautics and Space Administration under Contract NCC 2-481, the National Science Foundation under Grant MIP 88-10584, and Sumitomo Electric Industries, Ltd., Osaka, Japan.

Finally, I would like to thank my parents, Roger and Jean, for successfully helping me to reach this stage of my life.

Table of Contents

qqq

CHAPTER 1

INTRODUCTION

Image understanding is an important area of research in a variety of applications. Some sample domains include mobile robot navigation, surveillance, industrial manufacturing, and defense. A key aspect to image understanding lies in recognizing the objects that are perceived by the machine. Imaging cameras, however, capture two-dimensional projections of the real world. Recognizing an object from a single projection is far more difficult than recognizing it from a 3-D representation of the scene. Stereo vision bridges this gap by gaining 3-D knowledge of the scene from the available 2-D projections. It has advantages over other methods such as laser range-finding due to its low cost and its tolerance of a variety of surface reflectance properties. Nevertheless, constructing the necessary algorithms for full machine vision remains a difficult task. Due to the magnitude of the general vision problem, only slow and incremental advances have been made toward approximating human visual performance. To aid in the development of more successful vision algorithms, it is important to exploit the combination of human and machine efforts at the algorithm construction level. To this end, it is necessary to recognize the strengths of both humans and computers, and to partition the work for algorithm design.

A logical partitioning of the algorithm construction effort results in relying on human expertise for the design of the general algorithm framework, while computer exploration within this framework searches for the appropriate complete algorithm. This approach works well with standard vision algorithms such as stereo vision due to the large number of adjustable parameters. Given the parametric form of the algorithm, it becomes the responsibility of the machine to find suitable values. The parameters serve many purposes. Some examples include detection thresholds, set sizes, and the configuration of image preprocessing stages. Frequently, the particular parameter values that are used in the final algorithm are set heuristically. There is often little or no theory behind the values that are used, and a significant amount of hand-tuning is used to find acceptable values. Since parameter tuning is a tedious process, machine automation allows more extensive testing than humans may find personally tolerable. As recognized in a panel discussion chaired by Sklansky in 1988 [27], one of the bottlenecks to effective vision applications is the amount of effort necessary for tuning the algorithm. Not only is tuning necessary for the initial implementation in the laboratory, but also for the installation of the system in the field. Additional factors that necessitate modification include varying image domains, different lighting conditions, drifting camera parameters, and the implementation of the algorithm on different computer systems. [1] For example, different sets of parameters may be needed for a system designed for both day and night vision applications. Additionally, parameters may need adjusting after the vision system is ported to another computer, for example, one that might lack special image-processing hardware. These adjustments can be costly because the novice user is typically unqualified to make them. Worse yet, even an expert can be inadequate because many systems require the extensive knowledge of the original code to make appropriate adjustments. Another benefit is the ability to train for different domains and automatically switch context as necessary. By putting the responsibility for tuning in the hands of the machine, the need for human intervention can be greatly reduced. This results in a cost savings consisting of both time and effort.

It is clear that a systematic method for automatically improving algorithm parameters would be of significant value. The goal set forth in this thesis is to present an intelligent parameter-tuning framework and to analyze a statistical method for systematically exploring the parameter space. The target problem is stereo vision and the objective is to find the parameters that maximize the average performance over a database of test images. There are three main reasons why the approach described here is successful: (1) the relationship between the parameters and the algorithm performance is unknown without running tests, (2) small test images that reflect realistic application domains are available, and (3) there is little knowledge available for creating new parameter sets (the domain is knowledge-lean). As it is impossible to evaluate all of the possible parameter sets, the method described here proposes a limited set of parameter values and evaluates them by a given deadline. Because it is designed to work under a deadline it is well-adapted to giving incremental improvement on an algorithm. This allows the results from previous applications to be used as the starting point for further parameter tuning. The central aspect of the approach developed here is a *statistical* method for trading off between the number of new parameter sets to generate and the number of tests to perform on the existing ones. This is done by performing some initial empirical tests to gain a model of the problem and then scheduling the available time. The total time is divided into stages (typically one or two) in which a particular testing strategy will be followed. This strategy determines which parameter set to test next and can use the results of previous tests to make this decision. The creation of new parameter values is also handled by the system in the form of a rule base. This allows the system to employ any

[1] Implementation on a different computer can change the performance bottleneck, thus shifting the time critical area of the algorithm.

available knowledge of how past test results may reveal how specific parameter values should be modified.

An important aspect of the statistical method is the generality of the approach in that no assumptions are made that make the method applicable only to vision. The method has also been successfully applied to other domains as well [18]. The reason this method is so general is that it requires simply a problem in which subtests can be performed to give an estimate of the algorithm's performance. Clearly, this approach is useful primarily when empirical tests are the most effective method for determining algorithm fitness. Algorithms of this sort are typically heuristic in nature, and, therefore, this method is effectively limited to this class of problems. As many problems in the vision domain and elsewhere rely heavily on heuristics, it is important to have a tool for automatically improving this component.

A brief overview of the entire thesis is now presented so that the reader can quickly find any particular area of interest. The next chapter, Chapter 2, gives a background to the stereo vision problem. It includes a summary of the complexities encountered and some of the problems to be overcome. Following this, Chapter 3 presents the TEACHER 4.2 framework and how it is applied to the stereo vision problem. Chapter 4 then discusses the statistical guidance strategies that are used by TEACHER to schedule the available time resource. This chapter gives great detail on the statistical approach that is the heart of the system. The specific implementation details of the entire system are presented in Chapter 5. This includes both details of the specific stereo vision system and some of the more precise details of how the TEACHER system performs its task. Experimental results are presented in Chapter 6 as verification of the methodology, and possible future work is presented in Chapter 7. Chapter 8 finishes with conclusions about the TEACHER system and a discussion of what was discovered through this research.

CHAPTER 2

STEREO VISION BACKGROUND

The following sections are an introduction to the stereo vision problem. In particular, the general goal and methods of stereo vision are discussed. Later sections briefly detail the specific difficulties of implementing these methods and some proposed solutions.

2.1. General Stereo Vision

The purpose of stereo vision is to determine the depth of the *visible* portions of a scene. This is known as the 2½–D sketch [21]. The 2½–D sketch is the depth of all visible surfaces. In contrast, the true 3-D sketch would include depth information of occluded surfaces as well. By using a model database, however, an approximation of the 3-D sketch can be obtained from the 2½–D sketch. The final purpose, of course, is to use this information for 3-D object recognition.

Stereo vision is not the only approach to determining depth in a scene. Some other methods include laser range-finding, depth from focus, and depth (or shape) from shading [17]. There are a variety of reasons for using stereo vision instead of other methods as a key stage in 3-D object recognition. Perhaps the most intuitively attractive aspect is that humans, as well as many other biological systems, use binocular vision as a way of judging depth. An additional benefit is cost. Stereo vision can be performed with two simple CCD video cameras and the requisite computational hardware and software. This contrasts with the comparatively expensive equipment necessary for laser range-finding. Another drawback of laser range-finding is that it can have great difficulties if the reflective properties of the targeted surface are poor. Methods that use focus or shading share the cost benefit of stereo vision, yet are not as robust. Focus information requires the scene to be digested piecewise as the depth of field of the camera must be small to obtain accurate distance measurements. Using shading information also is problematic in that it depends on foreknowledge of the imaged surfaces in the form of the reflectance map. Drawbacks such as the ones mentioned above can significantly impede practical real-time vision systems that may be used for applications such as mobile robot navigation. Although not perfect, stereo vision stands out as a viable contender for real-world vision applications.

The standard two camera (binocular) stereo vision setup is shown in Figure 2.1. Here the two cameras are pictured facing into the page. The object being imaged is the pentagon.

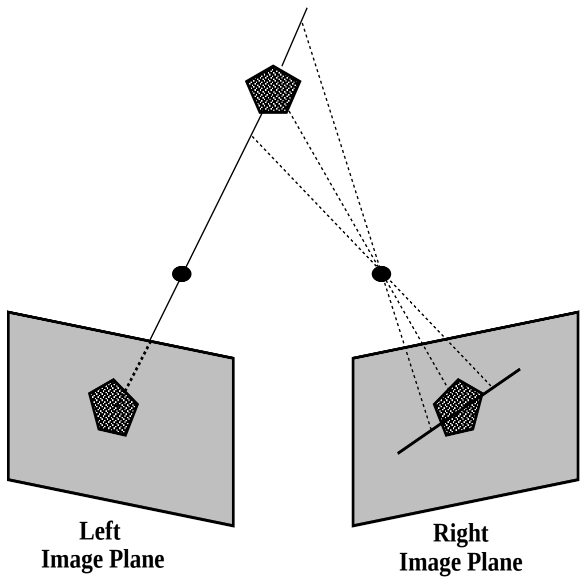


Figure 2.1. Stereo camera setup.

One image alone (left) narrows the position in 3-space to a line passing through the focus of the camera and the position of the object in the image. The matching position of the object in the second image (right) conveys where along that line the object lies.

2.2. Depth Determination Procedure

The general procedure for discrete binocular stereo vision can be summarized in five steps:

(1) Obtain two images from different viewpoints.

(2) Extract the tokens, or scene features, from each image to use for matching. (This information is known as the primal sketch.) [21]

(3) Determine the correspondence between tokens found in each image. (This results in the disparity map.)

(4) Translate disparity values into depth values. (This results in the depth map.)

(5) Interpolate over the depth map as necessary for the desired resolution of depth information.

The first step can be accomplished using two cameras separated by a small distance, e.g., 25 centimeters. A typical image might be 512×512 pixels with 8 bits for recording pixel intensity, i.e., 256 gray-levels.

In the second step, the matchable features can be either the raw intensity levels or the tokens that are extracted from the intensity information. If the intensity levels themselves are used, then the matching problem becomes that of finding the necessary degree of stretching and condensing necessary to map one image onto the other. This operation is computationally intensive, and when high resolution is desired can quickly become impractical. Preprocessing is typically used to extract higher-level tokens. For example, the extracted tokens could include edge points (zero-crossings) [10, 28], edge segments [1], and/or corner points [32]. The advantage of token matching to intensity matching is not limited to computational benefits alone. Tokens, such as edges, tend to emphasize areas of interest for object identification, and these are points at which depth information can be most useful.

The third step in stereo matching is known as the correspondence problem and it has traditionally been the most difficult. If the matching criteria are too loose, then depth information is lost due to ambiguous potential matches. However, if the matching criteria are too rigid, then depth information is lost because no match can be found. There is no known analytic function to optimize for the selection of the matching criteria. This is best approached, therefore, by heuristic methods generated either by human or machine.

The fourth step in the stereo matching procedure is to transform disparities into depths. This is straightforward when given the geometry of the camera setup. Reference [17] gives a concise outline of this procedure.

The fifth and final step of interpolating depth values can be solved with a variety of interpolation methods as well as with knowledge-driven interpolation schemes. One simple approach is to label each point in the image as having the depth of the nearest known depth measurement point. Another method is to find the three known points that enclose the desired point in a triangle. A planar patch is then fitted to these three points, and the point of interest is interpolated as lying on this plane. The latter method is perhaps more desirable as it ensures that the depth measurements are continuous. For the purposes of demonstration, however, the nearest-neighbor method was used in this thesis (see Section 6.1).

Using the general algorithm as outlined above, the methods suited for machine tuning lie in steps (2) and (3). Accordingly, this area is the focus of this thesis.

2.3. Complexities of Stereo Vision

Under the above general view of the stereo matching procedure, the computational goals fall into three areas:

(1) speed of matching,

(2) density of matches, and

(3) accuracy of matches.

With regard to the first area, there are a number of factors that contribute to slowing the algorithm’s execution. One problem can be the number of tokens considered for matching. If the feature extraction algorithm is not adjusted properly for the given image, it can extract tokens over 90% of the image area. Although this might allow dense matching, it can significantly hamper the performance if the distribution of tokens causes unfruitful search overhead. Another speed-related problem lies in preprocessing the image for token extraction. The computational burden here often rests in blurring the image. The blurring is used to filter out high-frequency detail and noise to reduce the number of extracted tokens for use in an iterative refinement process. Improper parameter settings at this stage can result in unnecessary filtering overhead.

The speed problems are directly related to the difficulty of generating a large number of matches. To decrease matching time, it is useful to decrease the total number of tokens to match. This trade-off results in a loss of depth information. If tuned properly, however, the algorithm can balance these two components to actually enhance performance. It is important to note that in most applications the objective is *ill-defined*. That is to say that the user may not know the precise performance trade-off that is necessary, but rather that certain requirements must be met. For example, the algorithm might have to process an image in less than a minute and with no more than a 5% error. When the requirements are met, however, the user may wish to express how surpassing the requirements reflects on algorithm fitness. This then gives the system insight necessary to find even better parameters.

A more difficult problem relating to the density of matches is the trade-off between resolution and uniqueness. If there is more than one potential match for a token, then it is not unique and cannot be well-matched. As the density of the tokens increases, there are more potential matches for any given token. Instead of this greater density increasing performance, it acts to hinder the overall stereo matching ability. Therefore, if the tokens cannot be more precisely characterized, or the search region cannot be reduced, then the problem becomes only worse.

There are four major approaches used to solve the difficulties mentioned above. The first is the use of three physical constraints of the stereo setup. These constraints are known as the uniqueness constraint, the ordering constraint, and the epipolar constraint. They are based on the physical characteristics of the stereo setup and are detailed in the reference [21].

The second approach used to improve matching performance is to obtain highly detailed information about each token. Therefore, even if the density of tokens is high, the density of *similar* tokens will be lower. Attributes of a token that might be extracted for matching should include type, position, orientation, and size. Nevertheless, because each token must have exactly one match derived from two physically different images, there will always be a trade-off between token descriptions that are too specific and those that are too vague.

Another method for improving matching performance is by employing figural constraints. The hypothesis is that if a series of tokens are in a particular pattern in the image, then they will have a similar pattern in the other [22]. This heuristic constraint is useful especially if the tokens are edge points. In this case, entire contours of tokens can be matched in a process analogous to closing a zipper.

The fourth aid to the matching problem is both powerful and widely used. It addresses the resolution versus uniqueness trade-off by incrementally improving the depth estimate throughout the scene. The procedure starts by extracting few, but prominent, tokens and then uses a large search region. This provides a good initial depth estimate. Further stages extract more tokens but can use previous depth information to reduce the size of the search region. Each stage of this algorithm is called a channel, and each channel has a granularity defined by the "coarseness" of the tokens extracted at that channel. One of the attractive aspects of this approach is that it is thought to be similar to the process used in biological visual systems [10].

2.4. Previous Work

Related work that attempts to address some of the problems mentioned above falls under the category of adaptive stereo vision. The adaptive vision area is broad, and much work has been done to overcome some of the brittleness of vision algorithms. In the area of adaptive stereo vision, two prominent approaches include the use of *metaparameters* and the use of *iterative refinement*. An example of the former can be seen in Weng, Ahuja, and Huang’s work on two-view matching [32]. Here, the detection of edges for use in the matching stage is performed in two steps. The first step calculates the intensity gradient over the entire image and then uses a histogram of the resulting values to set edge-detection thresholds. The fraction of the edges that are earmarked for detection is fixed beforehand. There are two drawbacks to this approach. The first is that the tunable parameters are merely heuristic parameters abstracted by one level. This means that potential values, e.g., 10% and 90%, given above must still be set through experimentation. These metaparameters can make the algorithm significantly more robust than simply setting the original parameters to fixed values; however, additional improvement could be made by using an automated system to set the metaparameters themselves.

The second drawback to using metaparameters is that it is necessary to gather statistics during run-time. In time critical applications this overhead may outweigh the benefit. Additionally, the statistical distribution may remain relatively constant. In this case, it is better to have the original first-level parameters set through off-line experimentation.

Another example of the use of metaparameters is seen in Tanaka and Kak's work in rule-based stereo matching [30]. Here, the control parameters are embedded in the system in the form of rules. The matching strategy dynamically shifts among four approaches as necessary. The four approaches consist of: (1) dominant feature matching, (2) geometric constraint matching, (3) zero-crossing contour matching, and (4) the Marr Poggio Grimson matching algorithm. Although the combination of these methods under central control improves robustness, there is now the overhead of coordinating them. The two basic problems of metaparameters (with respect to the rules) are again present here also, the first being run-time decision overhead, and the second being the need to tune the metaparameters.

Another area of adaptive stereo vision is in the iterative refinement of the parameters. An example in this area is the work of Takahashi and Tomita [29]. The focus here is in calculating stereo camera parameters from the two images alone. These parameters are those related to camera position and orientation, which are sometimes subject to drift over time. Although the algorithm must be invoked only periodically, it still requires run-time computation. Additionally, it depends on the existence of an analytically correct solution. In the case of finding the camera parameters, this is appropriate, but many times the search methods for the parameters will be ill-defined. In this case, intelligent or expert experimentation is the best available tool.

In summary, there are various approaches to improve the robustness and performance of the algorithm. Table 2.1 summarizes the previous work presented here with the methods, goals, and drawbacks. Nevertheless, as the heuristic component of the algorithm cannot be completely removed, it becomes necessary to find a method to cope with it. As has been shown, metaparameters alleviate some of the problem as do dynamic methods such as statistics gathering and iterative refinement. However, in many applications, preprocessing time is available for the system to adjust itself. This would remove the burden of dynamic correctional computation and could address the tuning of the heuristic components. The method proposed to counter these problems is the focus of this thesis and is fully discussed in Chapter 3.

Table 2.1. Previous Approaches and Drawbacks

Purpose	Approach	Weakness	Reference
Robust stereo vision	Robust algorithm with metaparameters	Metaparameters must be hand tuned	[31]
Robust stereo vision	Rule-base for operation under various algorithms	Overhead of rule management	[29]
Self-adjusting stereo setup	Iterative refinment of stereo configuration parameters	Run-time overhead, addresses only deterministic parameters	[28]

2.5. Issues on Heuristic Components

The combination of the above approaches is powerful for improving both the speed of the matching procedure and the density of the resulting depth information in stereo vision. Nevertheless, the implementation of these methods raises a variety of questions. What values should the heuristic components of the algorithms take? How does one accommodate tuning parameters that have a continuous range or could take on an infinite number of values? How does one perform tuning of the parameters when little or no knowledge is available on how to tune them?

To start with the first question, one must first examine the algorithm heuristic components. For example, how much detail should be extracted for each token? How many tokens should be extracted for a single pass of the matching algorithm? How many channels are necessary? What parameters should be used to extract the proper number of tokens?

The answers to these questions are particularly problematic. Without explicit assumptions about the characteristics of the scene and the tokens, there is no optimal or analytic solution for obtaining fast and dense depth information. The methods described above are heuristic, and their specific implementation in one domain may be far different in another. For example, the image domain for a factory setup under fixed lighting is very different from that found on a mobile vehicle under natural lighting conditions. The former can benefit from a myriad of simplifying assumptions, whereas the latter requires much more robust algorithm performance. In addition, some domains might require varying resolutions of imaging which can again lead to different trade-offs. Furthermore, two setups might have greatly varying performance requirements. A manufacturing robot might have a primary concern with speed, whereas an inspection robot might be more concerned with accuracy. As these performance demands change, so do the parameters controlling the algorithm. The problem now lies in providing the appropriate parameters to the matching procedure for maximizing performance in the particular image domain.

Table 2.2 lists some typical parameters that must be tuned in a general token-based stereo matching algorithm. (The specific algorithm that is implemented in this thesis is described in Section 5.1. Sample values for the parameters listed here are given in Table 5.1 on p. 44.)

There are two remaining problems, presented earlier, that hamper finding the appropriate parameters. First, as is evident from the table, the parameters are continuous, and can sometimes be unbounded. This means that the parameter space is infinite, and, unless certain restricting assumptions are made, it becomes impossible to test all possible parameter combinations. The second problem is that there is little information available to guide the search for the appropriate parameters. This is to say that the domain of parameter-tuning is knowledge-lean. Therefore, there is little recourse but to generate parameter sets and test them in a controlled manner.

The purpose of this thesis is to propose a more robust and more sophisticated approach. This is accomplished by starting with some initial parameter values. Each unique set of values is considered a separate entity that could possibly lead to an improvement. The system decides how many new entities to create, how to create them, and how to schedule available testing time. It is important to note that, as the available time is not unlimited, it is necessary to schedule tests appropriately. The time constraint is guaranteed to exist in any problem in which there is a potentially infinite set of items that must be explored. Problems of this class are ones in which empirical tests are the best or the only way to evaluate performance. The system used here recognizes that and, under a statistical model, makes the

Table 2.2. Stereo Algorithm Parameter List

Parameter	Function and Purpose
Number of Channels	Aids matching density by refining the search region and obtaining increasingly dense depth estimates.
Blurring Kernel Size	Width (σ) of the Gaussian filter used to select the strength of detected edges. This, along with the edge detection thresholds, controls the density of edges detected in a particular channel.
High Threshold	Upper limit of gradient strength for hysteresis of edge thresholding.
Low Threshold	Lower limit on gradient strength allowed for edge classification.
Initial Search Window	Size of region to initially consider for a match.
Similarity Thresholds	*Used to determine if tokens are a match (similar).*
Gradient	Difference in gradient magnitude
Orientation	Difference in edgel orientation
Vertical Position	Difference in vertical position

trade-off in a mathematical fashion. This allows a systematic exploration of the immense search space, as well as the incorporation of methods such as gradient ascent, to combat the knowledge-lean aspect of the problem. The specific details of this approach are presented in Chapters 3 and 4.

2.6. Algorithm Quality

To automate the process of searching for the appropriate algorithm parameters, it is necessary for the user to specify first the criteria for rating algorithm quality in the form of an objective function. In most applications, the designer is faced with overall performance requirements that must be met. It is from this that the quality of an algorithm is measured.

For example, requirements could consist of a time limit or a minimum accuracy. However, in this respect the objective is ill-defined. In other words, the precise performance trade-off of accuracy or speed is not known precisely and can be left to the system to decide. With respect to the stereo-matching problem, there are three factors used to judge performance.

(1) *Speed*: The rate at which a unit area of the depth map is calculated.

(2) *Density*: The fraction of the image for which depth information is determined.

(3) *Accuracy*: The average error in depth measurements.

The relative importance of each factor, of course, depends on the application. The bias of application-specific modifications to the algorithm can be set by the user by formulating a particular equation of these three measurements. (This is called the *objective function* and is fully discussed in Chapter 3 and Section 5.2.) Without specifying minimum allowable values in the performance criterion, an example performance equation might be

$$Performance = speed \times density \times accuracy \ . \tag{2.1}$$

This example objective function is specified as a mathematical function, but because the actual objective is ill-defined, the function represents a family of functions. It is a family because the components can themselves be considered as functions. For example, the accuracy component can be expressed parametrically, thus it would take a functional form itself. Similarly, other more specific, or even more general, measures may be substituted as components in the objective function. An example would perhaps incorporate the end goal of the vision system such as 3-D model matching into the function. This would allow the stereo vision system to be tuned with respect to its specific impact on the vision task at hand. In general, it is the task of the system to hypothesize a set of alternative objectives by performing tests. These objectives will be implicitly arrived upon by natural trade-offs made by the system. It is the user’s responsibility to review the machine-generated results and select the most appropriate *explicit* objective trade-off.

In a more realistic objective than the one shown in Eq. (2.1) above, the performance equation would set bounds on allowable time, acceptable density, and acceptable accuracy of depth information. A specific formulation used for parameter learning is given in Section 5.2. This function consists of a time penalty, an accuracy weight, and a density weight to measure performance.

Even with the measure of the algorithm quality clearly defined, it is still not straightforward to automate the process of finding better parameter values. As the parameters are heuristic in nature, and there is little knowledge available to use in the tuning process, it is necessary to perform tests. Each evaluation of the performance requires a test, and this can be time-consuming. To fully evaluate the performance of a parameter set, it is necessary to test it over the entire training domain database. Clearly, a gradient ascent type method is impractical here. For such a method to work, it would be necessary to perform full evaluation frequently. A more practical approach is the topic of the following chapter.

CHAPTER 3

TEACHER 4.2: A SYSTEM FOR LEARNING STEREO VISION PARAMETERS

TEACHER is a system that implements TEchniques for the Automatic Creation of HEuRistics. It has evolved over the course of various research projects in the last six years and has been applied to improving many algorithms that rely strongly on a heuristic component.

The original TEACHER 1.0 system [35] was developed to learn dominance relations in combinatorial search problems. In this case, TEACHER used a variety of learning methods and learned by examples that could be classified as either positive (support of the hypothesis) or negative (rejection). The focus was on the representation and organization of knowledge used in finding new dominance relations for combinatorial search problems. As a consequence, this system did not make explicit use of the time constraint to schedule resources in a statistical fashion. Time constraints were considered to the extent that there was a known time limit in which it was expected to find a dominance heuristic. If this deadline was not met, then the user was queried as to whether the system should continue or stop.

TEACHER version 2.0 [20] learned heuristics for numeric optimization problems. The specific aspects of search heuristics explored here were the bound, guidance, and transformation functions. Nevertheless, like TEACHER 1.0, TEACHER 2.0 did not adjust its strategies under consideration of the given time limit. The main advancement of version 2.0 was its combination of learning strategies into an organized approach consisting of three phases. The phases were (1) generation, (2) management, and (3) evaluation. For greater efficiency, the appropriate learning task was identified in each phase, and, based on the target problem, the appropriate learning strategies were employed.

Version 3.0 of TEACHER [31] was used to learn artificial neural-network configurations for a set of training patterns. This system marked a step away from previous versions in that it was impossible to divide the test phase of the generate-and-test approach into subtests that could be generalized. In this case, each subtest corresponded to a training epoch of the network configuration in which a very limited amount of training was performed on the selected configuration. Since complete training was typically very long and had an unpredictable duration, it became necessary to assess and predict the progress if further training of a particular configuration were to be continued. As a result, when feedback became available, i.e., when networks had been trained completely, it was necessary to assign credit to the various actions and rules carried out in the past that led to this feedback.

This is the traditional credit assignment problem; it was addressed in TEACHER 3.0 by partitioning the learning experiments into the classes of *Type I* and *Type II*. Type I tests were used to allot training epochs to the network configurations, using a given set of network-assessment heuristics. Type II tests were used to apportion the credit of feedback and to fine-tune the network-assessment heuristics. With this approach it was still possible to perform small test cycles to allow timely pruning of unworthy candidate networks, and to complete the learning process within a given time limit. Note that a greater emphasis was placed here on the credit assignment problem than on the scheduling of computational resources.

Recently, TEACHER 4.0 [18] was developed to study the scheduling of computational resources in a generate-and-test paradigm. A statistical methodology was used to address the resource constraint of limited available time directly. This problem can be expressed as a search through the space of possible parameter sets. Each potential parameter set is called a candidate parameter set, or more simply, a *candidate*. This search can be characterized by an objective to be optimized and a set of constraints to be satisfied. Since the target problems studied were characterized by benchmarks that can complete in a relatively small amount of time, the credit assignment problem studied in TEACHER 3.0 does not exist here, and emphasis was placed on studying various scheduling policies.

Based on TEACHER 4.0, TEACHER 4.1 [18] has been used to improve process mapping heuristics under the AXE framework [33, 34]. The target problem is characterized by a well-defined objective for optimization.

The version of TEACHER utilized in this thesis (version 4.2) is used to improve the heuristic components found in a general stereo vision algorithm. The goal is to find a set of stereo parameters that maximize the overall performance of the matching algorithm. This system shares many similarities with TEACHER 4.1, used for the process-mapping problem, because reasonable subtests can be performed, and a statistical measure can again be used as a predictor of performance. It is different from the process-mapping problem in the sense that there is no well-defined objective for optimization. Consequentially, it is necessary to evaluate different objectives of the target problem and let the user pick the best alternative.

Informally, the objective of the methodology used for such a search is to maximize the quality of the parameter set, expressed by the *objective function*. In other words, it is necessary to perform the best trade-off between the time taken to match the tokens in the scene and the quantity and accuracy of the matched tokens. This objective is specified by the user. Other possible objectives include maximizing the worst-case or average-case performance. Unfortunately, there is no good model to show how a particular set of parameters might impact the value of the objective function. Therefore, its effects can be measured only through actual application of the particular parameters to a stereo image pair. Similarly, there are no well-defined constraints that define the search space for the parameters. This is primarily because some parameters have an unbounded range, and the continuous range results in an infinite number of possible parameter values. Consequently, this search is considered to have an ill-defined objective as well as ill-defined constraints.

Due to the size of the parameter space and the amount of time required for the full evaluation of a parameter set, it is impractical to enumerate the objective values of all parameter sets. One way to solve this kind of problem is to have knowledgeable experts handtune the parameters themselves. However, the set of possible parameter values is extremely large, and the complex interdependence of the parameters might be unknown even to the experts. Another possible solution is to perform extensive experimentation by high-speed computers. Use of this technique alone, however, is impractical because the search space is too large.

These two approaches can be combined by using expert knowledge to guide the search for better candidates and by utilizing high-speed computers to perform computation-intensive experiments. The generate-and-test framework is a form of this approach. It uses expert knowledge (in the form of rules) to generate new parameters and high-speed computers to evaluate the quality of the proposed parameters against test images from the target domain.

3.1. Candidate-Generation Methods

To successfully implement the solution discussed above, an efficient and automated method for generating and evaluating new candidates must be found.

The problem of discovering new heuristics was first addressed by Pearl [24], who explored the paradigm that heuristics are discovered through the construction of simplified models of the problem domain. Pearl’s scheme involves systematically refining, removing, and relaxing constraints from the original problem until a semi-decomposable model is identified. The solution to the resulting model is then a heuristic to the original problem.

Other paradigms for machine learning can be applied as well. Some machine-learning strategies studied in the literature include rote learning, learning by instruction, learning by deduction, learning by analogy, and learning by induction [23].

Most of these learning methods, however, have drawbacks that make them impractical or ineffective in this case. Rote learning and learning by instruction, for example, are inappropriate because a teacher is necessary to provide the missing domain knowledge. In the case studied here, that knowledge is unavailable even to the experts. Learning by deduction is inappropriate because the problem is heuristic in nature and existing domain knowledge is stored implicitly. If the expression of heuristic knowledge were formed explicitly, it would pose additional problems for reformulation and restructuring. Under these conditions, deduction offers little promise. Learning by analogy is also inappropriate due to the difficulty in finding similar domains in which a solution already exists. Again, the implicit storage of domain knowledge hinders this kind of approach.

The most suitable learning mechanism for this problem is learning by induction. Inductive learning is the process of generalizing from a given set of examples. This includes learning by experimentation and learning by example. Both methods are appropriate because they map well to the problem of evaluating points of the objective function and of learning an approximate form of the empirically evaluated objective function.

Since the domain of stereo vision parameter tuning is knowledge-lean, i.e., the knowledge that leads to the generation of better candidates is either very difficult to acquire or too expensive to be useful, it is more important to have an efficient procedure for testing the generated candidates than to develop methods for capturing new knowledge for candidate generation. The focus here is on studying the generate-and-test mechanism operating under a *known fixed deadline*. The major problem addressed is on the scheduling of resources for exploring the candidate space systematically so that the best parameters are obtained upon reaching the deadline. The primary trade-off is between the number of candidates to be evaluated and the amount of evaluation performed on each.

3.2. Generate-and-Test Framework

The generate-and-test framework used here is geared toward searching an ill-defined space under a given time constraint (see Figure 3.1). Before examining its structure, it is necessary to define some terms. As mentioned earlier, a candidate refers to a particular parameter set of interest. The set of all candidates currently under consideration for testing is called the *candidate-pool*; the set of images used for testing the candidates is called the *test-database*.

Since the performance is optimized over the entire database, it is important for the test-database to be relatively homogeneous, or consisting of images from the same general class. If the images come from different classes, then the discovered parameter set might perform in a consistently mediocre manner. To overcome this problem, one could used a preprocessing stage to do some rudimentary image classification. The discovered parameter sets would then be for the separate classes of images contained in the database. During run-time, an extra module could be used to classify the image and select the predetermined algorithm parameters. In cases in which the database is small, or homogeneous image classes are not readily available, the images could be partitioned to create small test cases.

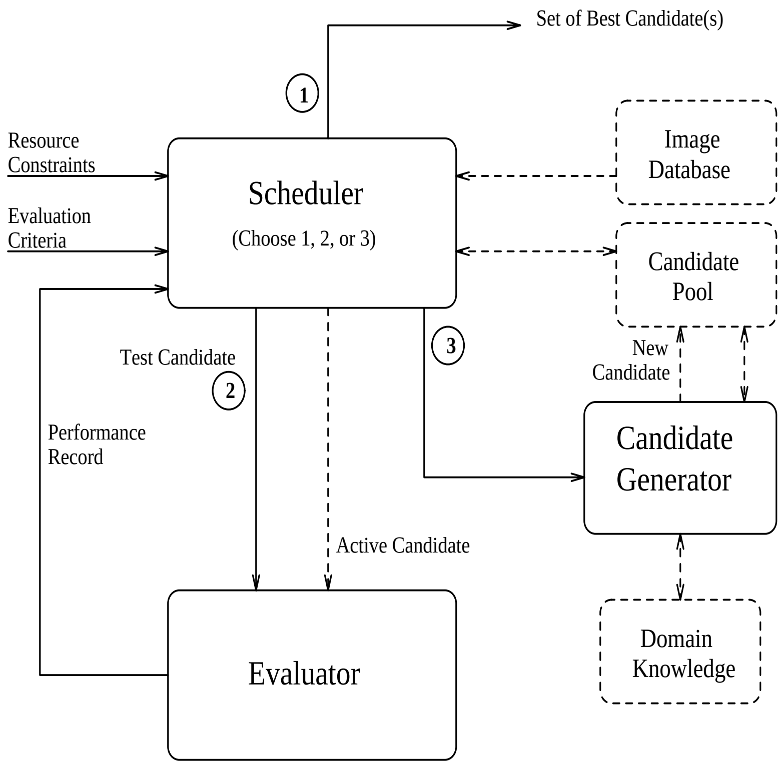


Figure 3.1. TEACHER's generate-and-test framework.

The three main parts that comprise the core of the framework are the *candidate generator*, the *candidate evaluator*, and the *resource scheduler*. The candidate generator creates new candidates for consideration; the candidate evaluator tests the candidates on the test images and records their performance; and the resource scheduler determines which candidate will be tested next. Due to the generality of the framework, it can be applied to other domains such as learning dominance functions in dynamic programming problems [35], learning selection functions in combinatorial searches [20], and learning neural-network configurations for a given set of training patterns [19, 31].

If there is much domain knowledge available, then the candidate generator can be very sophisticated. In this case, the candidate evaluator plays a subordinate role. However, in knowledge-lean domains, the candidate generator is relatively primitive. Therefore, the burden of selecting good heuristics shifts to the resource scheduler and the candidate evaluator. Such is the case in the stereo vision domain. Consequently, the focus is placed on the resource scheduler and the candidate evaluator. It is worth noting, however, that the strategies developed are general enough to be applicable to knowledge-rich domains as well.

To avoid spending a large amount of time on poor candidates, the evaluation process is divided into small subtests called *quanta*. This allows the system to perform additional tests on candidates *only* if they demonstrate some merit during prior quanta. During one quantum of time, tests are performed on the selected candidate using test images from a database supplied by the user. There are two possibilities for the test images used here. They can be either relatively small images (64×64) that give only a mild indication of fitness of the candidate, or they can be fairly large images (512×512) that give a good indication of candidate fitness. The advantage of the latter approach is that the candidate is tested under more realistic conditions; however, its drawback is that the duration of one quantum must be large enough to accommodate the test. This can then have an adverse effect on the system's performance. In the stereo vision domain, parameter set performance scales relatively well from small to large test images. Therefore, candidates are tested using small images, with the goal being to find the candidate with the greatest average performance. Of course, the optimal objective is to find a candidate that performs the best in all cases. This, however, is impractical or impossible to verify unless all candidates are evaluated over the exact same (and possibly infinite) set of test images representing the application domain. This would imply that all candidates (including the worst) must be evaluated to an equal degree.

In other application domains, the behavior of candidates may not scale well with problem size. For example, in designing an artificial neural network [31], it may not be possible to assess the quality of a network configuration at the end of a quantum. This happens because the network is too large to be fully tested. It is also not possible to train a small network and then generalize it to a larger one. Consequently, the decisions made at the end of a quantum are heuristic rather than statistical and may be prone to additional error.

At the end of each quantum the scheduler selects one of the following actions to perform:

(1) Select the next candidate to test from the candidate pool.

(2) Generate a new candidate to be placed in the candidate pool (and possibly remove an existing one from the pool).

(3) If the deadline has been reached, select a set of the best candidates and terminate testing.

The decision between choices (1) and (2) is made based on the current performance of the candidates in the pool and the amount of evaluation that has been performed on each. One simple method for determining when to generate new candidates for the pool is to generate new ones whenever existing ones have been evaluated to have a performance value known to within a statistical confidence level. (Another approach is discussed in Section 5.4.)

If the decision is made to pursue choices (1) or (3), then the candidate(s) are selected based on an *evaluation criterion*. This consists of two parts: the *goodness function* and the *guidance strategy*.

The goodness function is an estimator of the value of the objective function. It is used to select from the pool the candidate that most likely performs the best. It is needed because candidates may not be fully evaluated to within a statistical confidence when testing is terminated.

The guidance strategy is used if testing is continued to select the candidate to be evaluated during the next quantum. The goal of the guidance strategy is to choose a candidate that maximizes the probability that the candidates with the highest objective values also have the highest goodness values. It is not always best to select the most promising candidate to test because a candidate may show less promise when limited tests have been performed but might appear better with more tests. Moreover, with limited resources, it might be necessary to explore more candidates early in the generate-and-test process and focus on a limited set of promising ones as time runs out. This trade-off is addressed formally in Chapter 4, in which a statistical model for sequential selection is presented.

If the decision is made to pursue choice (2), then a new candidate must be generated. The candidate generator should be intelligent in guiding the generation of new, and ideally better, candidates. To this end, it should use the past performance of existing candidates as well as any available domain knowledge. This part of the framework can be improved incrementally by using high-level learning strategies such as learning by deduction, learning by explanation, and relaxation methods [24]. Generate-and-test is not applicable to improving the candidate generator itself because the overhead of testing new rules can be prohibitive. For the scope of this thesis, the candidate generator is driven by a fixed set of rules.

An outline of the actual code comprising the TEACHER framework can be found in Appendix B.

CHAPTER 4

STATISTICAL GUIDANCE STRATEGIES

The problem of finding the best candidate by performing a series of tests has traditionally been of great interest. Many problems can be formulated in which the quality of a candidate is a function of a collection of measurements on that candidate. Since each measurement typically has a certain aquisition cost there is a desire to find the best candidate (according to some metric of "best") with the minimum cost. A simple example might be to find the safest automobile. One might start with a given set of automobiles and then perform crash-tests to determine an average safety factor. Each test here requires demolishing a car — a very expensive procedure. A systematic method for finding the best car *while minimizing the cost* would be highly beneficial. Since the goal is to select the best candidate, the study of this problem has been called *selection theory*. For the problem of finding the best stereo vision parameters, complete and independent test cases can be evaluated within one quantum. Therefore, statistical selection methods are applicable and appropriate. The efficiency of the selection strategy is very important for the success of the generate-and-test framework, because the stereo vision parameter-tuning domain is knowledge-lean.

The first part of this chapter outlines the problem, the previous work in the area, and the difficulties found in applying the previous work to the problem at hand. Next, a multistage testing strategy that addresses these difficulties is described. Following this are a presentation and evaluation of various selection strategies within the multistage testing framework. Finally, the approach for realistically applying these strategies to the task of parameter tuning is described.

4.1. Problem Overview

The generate-and-test framework is faced with choosing the best candidate from a pool of candidates. Each candidate in this pool has an associated set of performance values obtained by testing it on images from the database. A statistical formulation of this problem can be expressed as: given a set of populations consisting of normally distributed numbers (with unknown means and variances), select the one with the highest population mean by testing a certain number of samples from these populations.[2] A common abstraction used to

[2] Population mean and variance are properties of a population. They can be estimated by the sample mean and variance if limited samples are drawn from the population or if the population is infinite in size.

approach this problem is to view the populations as bags containing numbered tiles. Each sample is therefore called a pick. In the case described in this thesis, however, the populations are candidate parameter sets, and the elements corresponding to the numbered tiles are the performance values associated with applying the candidate to the given test images.[3] Making one pick from a population is analogous to testing the candidate on one image. The goal here is to choose the candidate with the highest mean[4] within a fixed, and known, number of tests (or picks).

In the past, the problem of finding the population with the best mean has been known as the sequential selection problem. The goal is to design the *guidance strategy* that tells which candidate to test next, so that the likelihood of selecting the best candidate is maximized. The ideal goal is to find an optimal guidance strategy for scheduling tests. This strategy should always find populations that have higher means than those selected by other strategies. Unfortunately, the populations that are selected for testing depend on the values of the samples made so far. Therefore, the best information available to judge a guidance strategy is the distribution of the population means found by that strategy. Different guidance strategies can be compared based on this information only.

Existing guidance strategies can be classified into the classes of *static* and *dynamic* strategies. Static guidance strategies have a selection sequence that is determined ahead of time. This sequence is independent of the values of the picks during the selection process. One simple strategy of this type is the *round-robin* strategy. It takes samples from each population in turn. Its drawback is that the number of picks made from each population is the same. In this case, it seems unnecessary to demand that the worst candidate be tested just as much as the best, ignoring how initial tests may quickly demonstrate the disparity in performance between them. This problem is present in all static guidance strategies.

In contrast to the static approach, dynamic guidance strategies select the candidate for testing based on known sample values. An example of this is the *greedy* method. With this strategy, samples are taken from the population that currently has the maximum sample mean. The problem here, however, is that only candidates that look good at the current stage are considered. Therefore this strategy might discard the best candidate at an early stage. This behavior, of course, results in poor overall performance for the system.

[3] A test image is randomly drawn from the image database.

[4] The highest mean is used as the objective here. As discussed in Chapter 2, other objectives may be specified.

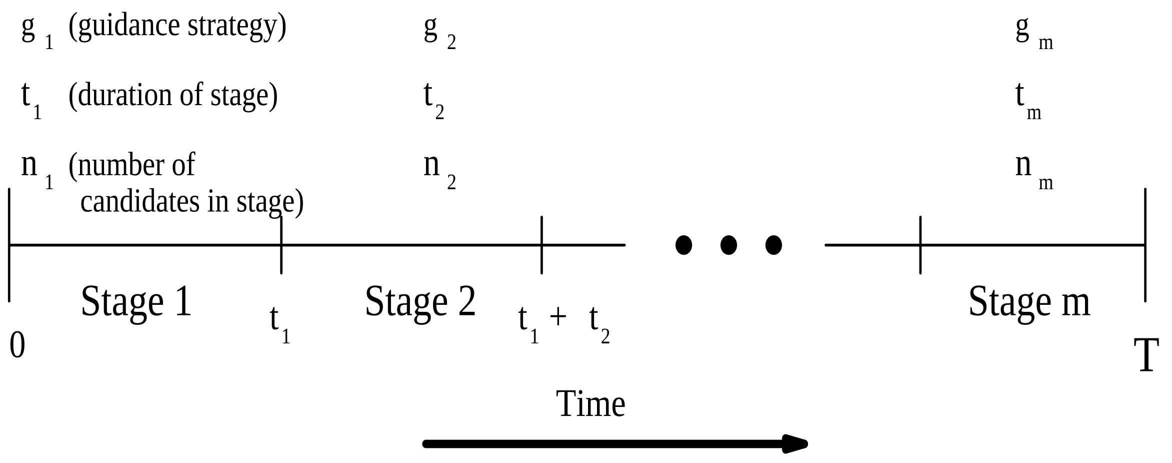
Work related to the selection problem was pioneered by Bechhofer in 1954 [2, 3]. The result of his method allowed the tester to determine the minimum number of picks necessary to determine which population was the best to within a certain degree of confidence. Many extensions have been proposed by researchers to accommodate various trade-offs and other goals of selection [8, 12, 14, 25]. Some papers deal with the case of an unequal number of samples [4, 11, 15, 26]. Nevertheless, optimal solutions to this problem do not exist.

Although much previous work has been performed in this area, the existing methods have some drawbacks that make them unacceptable for this work. The major problem with applying the traditional methods to the time-constrained selection processes is that the emphasis has been on the *stopping criteria* for testing a *finite* set of populations, rather than on finding the best population within a given time constraint. The traditional sequential selection problem dictates that in each time-step all of the populations are sampled once. Following this, a measure is calculated to determine if the desired degree of confidence has been attained, or if testing should continue. None of the previous methods has considered the case in which there are possibly more populations than total allowed testing time, or when one knows a priori how many picks the system will be able to make. Both of these conditions are highly relevant in the case studied here because the deadline is known beforehand, and there are often far more parameter sets than could possibly be tested. In the next section, a general guidance framework is presented to deal with these problems.

4.2. Multistage Testing

As a result of the previous section, guidance strategies must be developed that take into account limited resources such as testing time, and that can deal with situations in which there are so many candidates that it would be impossible to test them all even once. Because of these conditions, the guidance strategies must take into account the trade-off between the number of populations that can be tested and the accuracy of the sample-mean values of these populations. (Sample-mean is important because it is used to select the best population in the end.) If more populations are tested, then the accuracy of the sample-mean values would decrease, and the final selection of the best population would likely be in error. If more samples are tested from a population, then fewer populations can be tested, and therefore there is a greater chance of missing a good population entirely.

To overcome these problems, it is necessary to formulate a general guidance strategy, $G(T)$. This is accomplished as a series of stages in which each stage is represented by $G_i(g_i, t_i, n_i)$, where i ranges from 1 to m, the number of stages. Each stage is then characterized by the triplet g_i, t_i, n_i. The particular guidance strategy to be used for stage i is g_i; the duration of the stage is t_i; and the number of candidates to be considered for testing during that stage is n_i. This multistage testing procedure (see Figure 4.1) can accommodate both static and dynamic guidance strategies. Under this methodology, the early stages correspond to coarse initial testing used to weed out unworthy candidates, and the latter stages correspond to more careful evaluation of the better candidates. Of course, only the candidates that have the top n_{i+1} sample-mean values at the end of stage i are carried over into stage $i+1$ for further testing. Finally, only the top candidate is selected at the end of the last stage. This entire procedure allows many candidates to be tested in the early stages, while accurate sample-mean values for the better candidates can be obtained by more thorough testing in later stages.



g_1 (guidance strategy)
t_1 (duration of stage)
n_1 (number of candidates in stage)
g_2
t_2
n_2
g_m
t_m
n_m
Stage 1
Stage 2
Stage m
0
t_1
$t_1 + t_2$
T
Time

Figure 4.1. The multistage testing procedure.

The goal now is to find the strategy parameters comprising the best guidance strategy. This is not unlike the original problem of finding the best parameters for the stereo vision algorithm. Here, however, there are different techniques available to solve the problem. The performance of a particular strategy is measured by the average quality of the candidate that it finds by the deadline. The performance of multistage testing depends on the number of stages and the strategy parameter values for each stage. The values of these parameters must be selected appropriately based on the characteristics of the given problem. Factors that affect these values include: (1) the size of each population, (2) the distribution of each population, (3) the total amount of testing time, (4) the number of possible populations, and (5) the distribution of the population means of the populations as a whole.

There are two well-known methods for evaluating the performance of guidance strategies. Monte Carlo simulation is a method that can be applied to any guidance strategy. It produces the empirical distribution of the population mean of the candidate selected by that guidance strategy. Mathematical analysis is another alternative. Although closed-form mathematical equations are easier to apply, they are usually very difficult (and sometimes impossible) to analyze. Therefore, a compromise is necessary to derive limited analytic results that provide insight into the relationship between the values of guidance parameters and the strategy performance. Simulation is then applied in cases in which mathematical analysis is intractable.

4.3. Performance Evaluation of Multistage Selection

In this section, the performance of several multistage-selection strategies is evaluated. The focus, however, is on only one- and two-stage strategies. This is for two reasons: First, the amount of improvement of three stages vs. two stages is significantly less than that of two stages vs. one stage. Secondly, the number of parameters that must be selected is proportional to the number of stages. Currently, there are no analytic methods for determining the best values for these parameters. Further study will be necessary to surmount this problem. (See Chapter 7.)

The goal here then is to find systematic methods for determining the values of multistage guidance parameters that are most suited to the given problem specification. First, the assumptions used in the analysis are presented. Then, the general steps used in analyzing the multistage guidance strategy along with an upper bound on its performance are presented. These steps are then applied to specific cases. Finally, stage value selection of static strategies is analyzed and simulations are used to evaluate some dynamic strategies.

4.3.1. Assumptions

The first necessary assumption is that each population (which corresponds to the performance values of a candidate) is normally distributed, and that the variances of all populations are equal. Further, each sample value within a population is assumed to be independent and identically distributed (i.i.d.). Empirical verification of this assumption is presented in Table 6.5 on p. 57.

The second assumption is that the distribution of the population means is known. This distribution gives the probability that a randomly selected candidate will have a particular population-mean value. The values of population means are also assumed to be i.i.d.. In many cases, this distribution can be approximated by a normal distribution. Furthermore, it is assumed that each candidate has an infinite number of samples. This assumption is reasonable for a typically sized training set and a typical deadline. In cases in which populations are created, rather that selected, the i.i.d. assumption is on weaker ground. For example, if new populations are created based on a modification of the existing populations with high means, then there will be a dependence. As is shown later, however, empirical results demonstrate that this does not significantly weaken the approach.

For statistical analysis, it is necessary to know the value of the population variance. In practice, the variance of populations may have to be estimated by presampling the populations. Another assumption made here is that the cost of sampling is unity. In reality, the duration of an individual test can vary so that the cost in terms of time is not held constant. This could cause a problem with scheduling if the cost of an experiment was ignored entirely. However, testing time is taken into account through the objective function, (cf. Section 5.2); therefore, cost is considered, albeit indirectly.

4.3.2. Analysis of the general multistage selection method

Multistage guidance strategies must be analyzed stage by stage. This section presents the general steps that can be used to analyze each stage of a multistage guidance strategy. These techniques can be applied to any given guidance strategy. In addition, an upper bound on the performance of guidance strategies is presented. This can be used to judge the quality of a particular guidance strategy in light of the theoretical limits on performance.

In this section, *T* represents the duration of the current stage, *n* the number of candidates to evaluate in the stage, and *k* the number of candidates to select for further evaluation in the following stages, if any.

There are four general steps in the analysis of each stage. Each uses information available from the analysis of the previous stage.

(1) Determine the joint probability distribution (p.d.f. $f_{\bar{x},\mu,s}$) among the population mean, the sample mean, and the number of samples per population for a random candidate at the end of the current stage.

For the first stage, $f_{\bar{x},\mu,s}$ can be found based on the guidance strategy and the given distribution of population means, f_{μ}. For other stages, $f_{\bar{x},\mu,s}$ can be found based on the guidance strategy used by a particular stage and $f_{\bar{x},\mu,s(k:n)}(x,\mu,s)$ [5] from step (4) of the previous

[5] The subscript notation *(k:n)* means the top *k* out of *n* candidates.

stage. This is the only step in which the chosen guidance strategy may affect the performance.

(2) Determine the probability distribution of the sample means of the candidates evaluated at the end of the current stage (c.d.f. $F_{\overline{x}}$ and p.d.f. $f_{\overline{x}}$). This is the marginal distribution of $\overline{x}$ and is found based on $f_{\overline{x},\mu,s}(x,\mu,s)$ from step (1).

$$f_{\overline{x}}(x) = \sum_{s} \int_{y=-\infty}^{+\infty} f_{\overline{x},\mu,s}(x,y,s)\, dy \tag{4.1}$$

The distribution of the sample means is necessary because the selection of candidates is based on the current knowledge of its performance: namely, the sample mean.

(3) Determine the distribution (p.d.f. $f_{\overline{x}(k:n)}$) of the candidates that have one of the top k among n sample means at the end of the current stage.

At the end of each stage, the k candidates with the highest sample means are selected to be evaluated further in the following stage, if any. The p.d.f. of a random candidate from among the k selected candidates, $f_{\overline{x}(k:n)}$, is simply the average of the top k-order statistics of the sample mean, $f_{\overline{x}}(x)$:

$$f_{\overline{x}(k:n)}(x) = \frac{f_{\overline{x}}(x)}{k} \sum_{i=n-k+1}^{n} i \binom{n}{i} \left[F_{\overline{x}}(x)\right]^{i-1} \left[1-F_{\overline{x}}(x)\right]^{n-i} . \tag{4.2}$$

The justification for this step can be more easily seen through a short example. In the case where one is interested in the distribution of the top k populations, the desired function is the probability that the r.v. X takes on the specific value x. In statistical terms this is

$$f_{X(k:n)}(x) = P\{\, X=x \mid X \in (\,(n-k+1){:}n \cup (n-k+2){:}n \cup \cdots \cup n{:}n\,)\,\}$$

where the expression $i:n$ means population i out of n. Using Bayes' rule, this reduces to

$$\frac{P\{\,(X=x) \cap (X \in (\,(n-k+1){:}n \cup (n-k+2){:}n \cup \cdots \cup n{:}n))\,\}}{X \in (\,(n-k+1){:}n \cup (n-k+2){:}n \cup \cdots \cup n{:}n\,)} .$$

The denominator of this, however, is 1 because X is defined to come from the combination of the top k populations. Therefore the expression reduces to

$$P\{\,[\,(X=x) \cap (X \in (n-k+1){:}n)\,] \cup [\,(X=x) \cap (X \in (n-k+2){:}n)\,] \cup \cdots$$
$$\cup\, [\,(X=x) \cap (X \in n{:}n)\,]\} .$$

As the populations are necessarily disjoint, i.e., $P\{(X \in i:n) \cap (X \in j:n) \text{ for } i \neq j\}=0$, the probability of the unions is the sum of the probabilities

$$f_{X(k:n)}(x) = P\{\,(X=x) \cap (X \in (n-k+1):n)\,\} + P\{\,(X=x) \cap (X \in (n-k+2):n)\,\} + \cdots$$
$$+ P\{\,(X=x) \cap (X \in n:n)\} \;.$$

By again using Bayes' rule this becomes

$$P\{\,X=x \,|\, X \in (n-k+1):n\,\}\, P\{X \in (n-k+1):n\}$$
$$+\, P\{\,X=x \,|\, X \in (n-k+2):n\,\}\, P\{X \in (n-k+2):n\} + \cdots$$
$$+\, P\{\,X=x \,|\, X \in n:n\,\}\, P\{X \in n:n\} \;.$$

By using the assumption that all populations are the same size (namely infinite) it can be seen that $P\{X \in i:n\}=1/k$ for all i. This leads to the final reduction,

$$f_{X(k:n)}(x) = \frac{1}{k} \sum_{i=n-k+1}^{n} P\{\,X=x \,|\, X \in i:n\,\}$$

which is the average of the top k-order statistics. With this explanation, it is now possible to continue with the fourth step in the process.

(4) Determine the joint probability distribution (p.d.f. $f_{\overline{x},\mu,s(k:n)}$) between the population means and the sample means for the k candidates with the top sample means among n original candidates. This is the only result that must be carried over to the next stage. Using Bayes' theorem, one can derive the equation

$$\begin{aligned} f_{\overline{x},\mu,s(k:n)}(x,\mu,s) &= P[\,x\,,\mu\,,s \,|\, x \text{ is in the top } k\,] \\ &= \frac{P[\,x \text{ is in top } k \,|\, x\,,\mu\,,s\,]\; f_{\overline{x},\mu,s}(x\,,\mu\,,s)}{P[\,x \text{ is in top } k\,]} \\ &= \frac{f_{\overline{x}(k:n)}(x)\; f_{\overline{x},\mu,s}(x\,,\mu\,,s)}{f_{\overline{x}}(x)} \;. \end{aligned} \tag{4.3}$$

For the last stage, k=1, Eq. (4.2) reduces to

$$f_{\overline{x}(1:n)}(x) \;=\; n\; f_{\overline{x}}(x)\, [F_{\overline{x}}(x)]^{n-1} \;. \tag{4.4}$$

In the last stage, it is also necessary to find the distribution of the population mean of the candidate with the highest sample mean, $f_{\mu(selected)}$. This can be found by using $f_{\overline{x},\mu,s(k:n)}(x,\mu,s)$ from the last stage. This is simply the marginal distribution and is expressed as

$$f_{\mu(selected)}(\mu) = \sum_{s} \int_{x=-\infty}^{+\infty} f_{\bar{x},\mu,s(1:n)}(x,\mu,s)\, dx \;. \tag{4.5}$$

The expected value of the population mean of the candidate selected by this method is

$$E[\mu_{(selected)}] = \int_{y=-\infty}^{+\infty} y\; f_{\mu(selected)}(y)\, dy \;. \tag{4.6}$$

As a benchmark for comparison, an estimator of the upper bound on the performance of guidance strategies can be found statistically. Given a deadline of T time units, at most T candidates can be evaluated (assuming that each subtest takes 1 time unit). The best that can happen is that the population mean of each candidate is found after only one subtest. (This also happens if the population variance is negligibly small.) In this case one can always select the candidate with the highest average performance from the T candidates. Since T different candidates are selected randomly from the candidate space, the distribution of the candidate with the highest population mean is the T^{th}-order statistic of the population-mean distribution. If $F_{\mu}(x)$ and $f_{\mu}(x)$ are, respectively, the cumulative distribution function (c.d.f.) and probability density function (p.d.f.) of the population mean of a random candidate, then the p.d.f. of the best of T random candidates is [16]

$$f_{best}(x) = T\,[F_{\mu}(x)]^{T-1}\, f_{\mu}(x) \;. \tag{4.7}$$

This upper "bound" is somewhat loose because it assumes one would be able to determine the candidate with the best population mean every time with only one sample from each population. It is also important to note that comparisons against this measure are typically made with respect to the expected value of the distribution. This results is comparing with the average of the best case. What would be more useful as a bound is the best of the average case. For a statistical study, however, Eq. (4.7) is a good approximation of what is sought and is certainly useful for judging strategy performance.

4.3.3. Static guidance strategies

In static guidance strategies, the selection sequence is fixed beforehand and is independent of the values of the picks made during the selection process. Since the sequence is fixed, it is likely that testing time could be wasted on some populations that have already shown poor performance from the first few initial tests. This could be partly overcome by using a multistage selection process which would have the effect of pruning off inferior candidates early in the process.

Static strategies do have one point in their favor: they are much easier to analyze than dynamic strategies. In addition, they can be used to obtain some initial statistics for the first stage of a multistage process employing dynamic guidance strategies. In this section, the performances of the *single-stage round-robin* and the *two-stage round-robin* strategies are analyzed. Also, an empirical method for determining the parameter values for these strategies is provided.

The *single-stage round-robin* strategy, $G(RR, T, n)$, takes samples (subtests) from each population (or candidate) in turn. Given the number of candidates to be evaluated, n, and the deadline, T, the number of subtests, s, for each candidate is equal to T/n. Assuming that each candidate is normally distributed with the same variance, σ^2, and the distribution of all population means is f_μ, one can determine the joint distribution of the population mean and sample mean when the deadline is reached.

$$f_{\bar{x},\mu,s}(x,\mu,s) = \begin{cases} f_{\bar{x}\,|\,\mu,s}\left(x\,|\,\mu,\dfrac{T}{n}\right) f_\mu(\mu) = \dfrac{f_\mu(\mu)}{\sigma\sqrt{2\pi n/T}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2 n/T}} & \text{for } s = \dfrac{T}{n}\ , \\ \\ 0 & \text{otherwise.} \end{cases} \tag{4.8}$$

From this equation, the general steps described in the previous section are followed to arrive at the distribution of the population mean of the candidate selected when time runs out.

$$f_{\mu(selected)}(\mu) = n\, f_\mu(\mu) \int_{x=-\infty}^{+\infty} f_{\bar{x}\,|\,\mu,s}\left(x\,|\,\mu,\frac{T}{n}\right) \left[\int_{\mu=-\infty}^{+\infty} F_{\bar{x}\,|\,\mu,s}\left(x\,|\,\mu,\frac{T}{n}\right) f_\mu(\mu)\, d\mu\right]^{n-1} dx \tag{4.9}$$

$$E[\mu_{(selected)}] = \int_{x=-\infty}^{+\infty} f_{\bar{x}(1:n)}(x) \int_{\mu=-\infty}^{+\infty} \frac{\mu\, f_{\bar{x},\mu,s}\left(x,\mu,\frac{T}{n}\right)}{f_{\bar{x}}(x)}\, d\mu\, dx \tag{4.10}$$

$$= \int_{x=-\infty}^{+\infty} f_{\bar{x}(1:n)}(x)\, E[\,\mu\mid \bar{x}=x\,]\, dx\ .$$

A normal distribution of population means is a case of special interest because it is the same one found in the stereo vision parameter tuning problem. The more specific results for $N(\mu_0, \sigma_0^2)$ are derived from Eqs. (4.4), (4.8), and (4.10). This leads to

$$f_{\bar{x}(1:n)}(x) = n \frac{1}{\sqrt{2\pi(\sigma^2 n/T + \sigma_0^2)}} e^{-\left(\frac{x-\mu_0{}^2}{2(\sigma^2 n/T + \sigma_0^2)}\right)} \Phi\left(\frac{x-\mu_0}{\sqrt{\sigma^2 n/T + \sigma_0^2}}\right)^{n-1} \tag{4.11}$$

$$E[\mu_{(selected)}] = \int_{x=-\infty}^{+\infty} \frac{x\sigma_0^2 + \mu_0\sigma^2 n/T}{\sigma^2 n/T + \sigma_0^2} f_{\bar{x}(1:n)}(x)\, dx$$

$$= \mu_0 + \int_{x=-\infty}^{+\infty} \frac{\sigma_0^2\, x\, n\, e^{-(x^2/2)}[\Phi(x)]^{n-1}}{\sqrt{2\pi(\sigma^2 n/T + \sigma_0^2)}}\, dx\ ,$$

where $\Phi(x)$ is the c.d.f. for an $N(0,1)$ distribution.

From the equations above, it is clear that only the ratio between σ^2 and T affects the performance. As T increases (or σ^2 decreases), the average performance of the candidate selected by this strategy improves. This happens because the accuracy of the sample mean (as a predictor of the population mean) increases as either more time is given, or if the variances are reduced.

In this simple strategy, the only parameter that can be adjusted is n, the number of candidates to be evaluated. One method for determining the optimal n, $1 \le n \le T$, which leads to optimum $E[\mu_{(selected)}]$ for a given problem and deadline, is to find the n that makes $\frac{dE[\mu_{(selected)}]}{dn} = 0$. In most cases, however, this cannot be solved easily because it is difficult to reach a closed-form solution for the integration.

If there is only one n that makes the derivative equal to 0, then there is a simple method for finding the optimal value of n for the given problem. The condition implies that if $E[\mu_{(selected)}]$ at n_1 is larger than that at n_2, and $n_1 < n_2$, then $n_{opt} < n_2$. However, if $n_1 > n_2$, then $n_{opt} > n_2$. Figure 4.2 shows that this condition is usually met. Under this condition, the optimal n can be found by a binary search.

Often, more complex multistage strategies use the round-robin strategy in the first stage. Therefore, it is useful to derive the joint distribution of the population means and the sample means of the top k candidates at the end of the first stage.

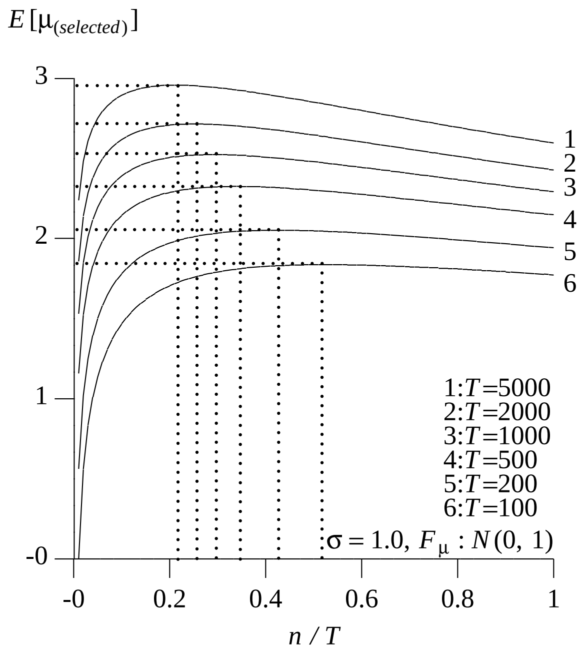


Figure 4.2. Effect of *n* on the performance of the single-stage round-robin selection strategy.

$$f_{\bar{x},\mu,s(k:n)}(x,\mu,s) = \begin{cases} \dfrac{1}{k} \displaystyle\sum_{i=n-k+1}^{n} i \binom{n}{i} \left[F_{\bar{x}}(x)\right]^{i-1} \left[1-F_{\bar{x}}(x)\right]^{n-i} f_{\bar{x}\,|\,\mu,s}\left(x\,|\,\mu,\frac{T}{n}\right) f_\mu(\mu) & s = \dfrac{T}{n}\ , \\ 0 & \text{otherwise.} \end{cases} \tag{4.12}$$

The *two-stage round-robin* strategy, $\{G_1(RR,\ rT,\ n_1)\,,\ G_2(RR,\ (1-r)\,T,\ n_2)\}$, is built on top of the single-stage round-robin strategy. In this strategy, a fraction, *r*, of the total time, *T*, is used in the first stage to evaluate n_1 candidates. At the end of the first stage, n_2 candidates ($n_2 < n_1$) with the largest sample means are taken into the second stage for further evaluation. The second stage takes all of the remaining time and also uses a round-robin strategy.

For each stage, the number of subtests performed on each candidate is the same, with $s_1 = \frac{rT}{n_1}$, and $s_2 = \frac{(1-r)T}{n_2}$. From this, one can use Bayes' theorem to find $f^{(2)}_{\bar{x},\mu,s}$, the joint p.d.f. of the population mean and sample mean of the selected candidate at the end of the second stage.

$$f^{(2)}_{\bar{x},\mu,s}(x,\mu,s) = \begin{cases} \displaystyle\int_{u=-\infty}^{+\infty} f_{\bar{x}|\mu,s}\left(\frac{(s_1+s_2)x - s_1 u}{s_2} \,|\, \mu, s_2\right) f_{\bar{x},\mu,s(n_2:n_1)}\left(u, \mu, \frac{rT}{n_1}\right) du & \text{for } s = s_1 + s_2 \ , \\ 0 & \text{otherwise.} \end{cases} \tag{4.13}$$

Eq. (4.13) can be solved by using Eq. (4.12).

When $s_2 \gg s_1$, the above equation can be simplified by considering the s_1 samples selected in stage 1 to be completely random when analyzing the performance of the second stage. In this case, the conditional distribution of the sample means obtained in the first stage is ignored. With this simplifying assumption, one can find $f_{\mu(selected)}$ in terms of $g_{\mu(n_2:n_1)}$, the p.d.f. of the population mean of the top n_2 candidates from the first stage. This result is based on Eq. (4.9).

$$g_{\mu(n_2:n_1)}(\mu) = \int_{x=-\infty}^{+\infty} f_{\bar{x},\mu,s(n_2:n_1)}\left(x, \mu, \frac{rT}{n_1}\right) dx \tag{4.14}$$

$$f_{\mu(selected)}(\mu) = n \, g_{\mu(n_2:n_1)}(\mu) \int_{x=-\infty}^{+\infty} f_{\bar{x}|\mu,s}(x \,|\, \mu, s_1+s_2) \left[\int_{y=-\infty}^{+\infty} F_{\bar{x}|\mu,s}(x \,|\, y, s_1+s_2) \, g_{\mu(n_2:n_1)}(y) \, dy \right]^{n-1} dx \, .$$

The distribution of $\mu_{(selected)}$ for single-stage and two-stage round-robin strategies for a problem with normally distributed population means is shown in Figure 4.3. The values are computed using Eqs. (4.11) and (4.14) and are compared to the results of Monte Carlo simulations, the upper bound from Section 4.3.2, and the original f_μ. From this graph, it is clear that a two-stage round-robin strategy performs much better than even the best single-stage round-robin strategy.

A two-stage round-robin strategy has more parameters that have to be selected: namely, r, n_1, and n_2. One method for finding appropriate values for these parameters is to

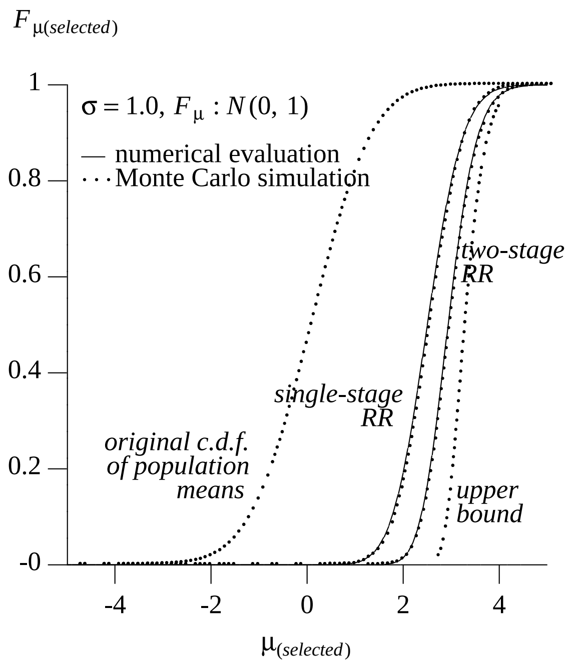


Figure 4.3. Performance of single-stage and two-stage round-robin strategies for T=100.

make the same assumption made for the single-stage round-robin strategy, the assumption being that there is only one local maximum which is also the global maximum. With this assumption, the values of r, n_1, and n_2 can again be determined by using a binary search. This is far more efficient than evaluating all possible combinations of r, n_1, and n_2. Figure 4.4 shows the results of using this method to select parameters for various values of r. From this figure, $r = 0.75$ appears to give the best performance.

4.3.4. Dynamic guidance strategies

Dynamic guidance strategies select candidates to test based on dynamic performance information. The candidate to test is selected based on the sample mean, the sample variance, or the distribution of performance values gathered thus far for all candidates under consideration. All dynamic strategies require the application of a static strategy in the first stage in order to gather the initial statistics necessary for the decision making process. In this section, two dynamic guidance strategies and their performance evaluations are presented.

The simplest dynamic guidance strategy is the *two-stage round-robin/greedy* strategy, $\{G_1(\text{RR}, r\,T,\ n_1),\ G_2(\text{Greedy}, (1-r)\,T,\ n_2)\}$. During the second stage, in which the greedy strategy is used, samples are taken from the population that currently has the

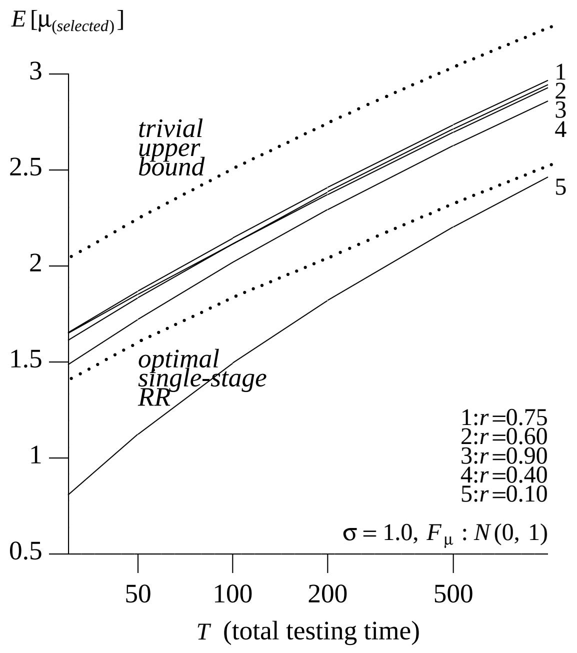


Figure 4.4. Effect of r with $f_\mu = N(0,1)$ and $\sigma = 1$ for a two-stage round-robin method.

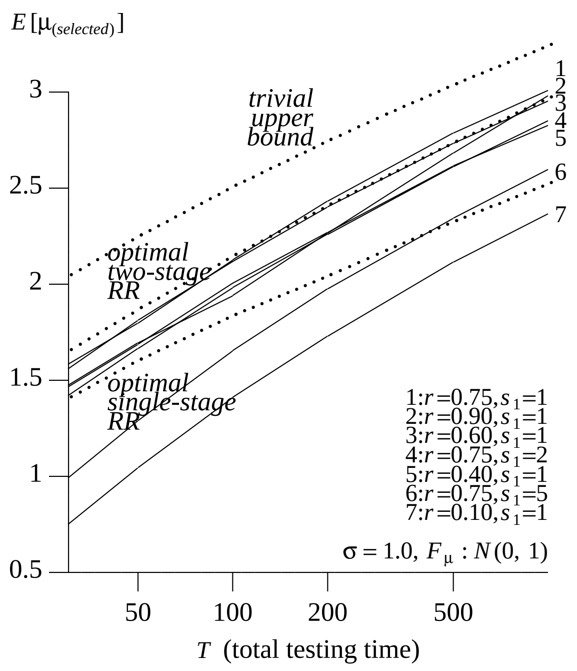


Figure 4.5. Effect of r with $f_\mu = N(0,1)$ and $\sigma = 1$ for a two-stage round-robin/greedy method with various s_1.

maximum sample mean. (See Figure 4.5.) The problem with this method is that it considers only candidates that look good in the current stage and therefore could possibly discard the best candidate early in the process.

There are other dynamic strategies (such as the one proposed in [9]) that focus on the candidates with high sample means, but also test the ones with smaller means if they are not tested enough. The major problem with these is that they do not take time constraints into consideration. For that reason, none of these methods will be discussed in this thesis.

To overcome the problem with the greedy method while avoiding extensive tests on the inferior candidates, the *minimum-risk* guidance strategy is proposed. To describe this strategy, it is first necessary to define some useful terms. The loss function is defined as the squared error of the sample mean in predicting the population mean. Since the goal is to have the estimator be accurate, it follows that it is best to have the loss be as small as possible. The loss, L_i, for population i, is expressed as

$$L_i(\mu_i, \bar{x}_i) = (\mu_i - \bar{x}_i)^2 , \tag{4.15}$$

where μ_i is the population mean, and $\bar{x}_i$ is the sample mean.

Next, the risk function is defined as the *expected* loss for a population. However, since it is necessary to be concerned only with the risk of the population selected at the end, the overall risk should be expressed as the sum of the risk for each population weighted by the probability of that population being the best. For m populations, the risk after test k is expressed as

$$\begin{aligned} R^k &= E[L^k(\mu_{(selected)}, \bar{x}_{(selected)})] \\ &= \sum_{i=1}^{m} P_i^k E[L_i(\mu_i, \bar{x}_i)] . \end{aligned} \tag{4.16}$$

Here P_i^k represents the likelihood that population i is the best and will be selected at stage k.

The minimum-risk strategy then selects the next candidate for testing so that the expected risk will be smallest after that test. Its major advantage over strategies proposed in the past is that it takes into account the time constraint when making its selection. This consideration of time is, however, still approximate as each test is considered to be performed in unit time.

To minimize the risk at an arbitrary future stopping point, one could use dynamic programming [7] to find the best candidate to test. The major difficulty of this approach is that the number of choices in each step is the same as the number of candidates. Therefore, the number of possible choices grows exponentially with the number of steps. In practice, one can only study a policy that selects the candidate to test as the one with the minimum risk one step into the future.

The components of the risk function from Eq. (4.16) are presented below. Under the assumption that σ_i, the population standard deviation, is unknown, the variable

$$\frac{\mu_i - \bar{x}_i}{s_i / \sqrt{n_i^k - 1}}$$

has a Student's t distribution with $(n_i^k - 1)$ degrees of freedom and a variance of $(1 + 2/(n_i^k - 3))$, assuming $n_i^k > 3$. This leads to the following equation:

$$E[L_i(\mu_i, \bar{x}_i)] = E[(\mu_i - \bar{x}_i)^2] \tag{4.17}$$
$$= \left(1 + \frac{2}{n_i^k - 3}\right) \frac{E[s_i^2]}{n_i^k}.$$

However, $E[s_i^2]$ is equal to σ_i^2, and, since this is not known, s_i^2 must be used instead. This increases the risk by a factor of $2/(n_i^k - 3)$.

The statistical definition of P_i^k is also defined using a t distribution as follows [13].

$$P_i^k = \int_{t=-\infty}^{+\infty} \prod_{j \neq i} F_t\left(\nu_j,\ t\,\frac{s_i/\sqrt{n_i}}{s_j/\sqrt{n_j}} + \frac{\bar{x}_i - \bar{x}_j}{s_j/\sqrt{n_j}}\right) f_t(\nu_i, t)\, dt \tag{4.18}$$

Here $F_t(\nu, x)$ and $f_t(\nu, x)$ are the c.d.f. and p.d.f., respectively, of the t distribution with ν degrees of freedom.

With a one-step look-ahead policy, only one of n_j^k can be increased by 1, simulating a sample of population i. Under this constraint i must be selected to minimize Eq. (4.16) in stage $(k+1)$.

$$i = \min_j R^{k+1} \mid \quad n_j^{k+1} = n_j^k + 1 . \tag{4.19}$$

In the special case in which one knows the "best" population, the choice of this strategy is simple: only the "best" population is tested further. This follows because $P_i = 1$ if population i is best, and $P_j = 0$ for all $j \neq i$.

The actual value of P_i^k is calculated, based on the information currently available in stage k. The unbiased estimators $\bar{x}_i$ and s_i are used for μ_i and σ_i, respectively. This implies that one needs at least four samples from each population to apply the minimum-risk guidance strategy.

For studying the performance of dynamic strategies, it suffices to use Monte Carlo simulations. Note that it is much harder to determine the appropriate values for the parameters of dynamic guidance strategies. One heuristic method for selecting parameters for two-stage dynamic strategies is to use the same r, n_1, and n_2 as for a two-stage round-robin strategy.

Figure 4.6 shows the average performance, $E[\mu_{(selected)}]$. The results are plotted over a range of deadlines. The dynamic strategies here are shown using the same parameter values as the two-stage round-robin strategy. These results show that dynamic strategies can outperform even the best static strategies.

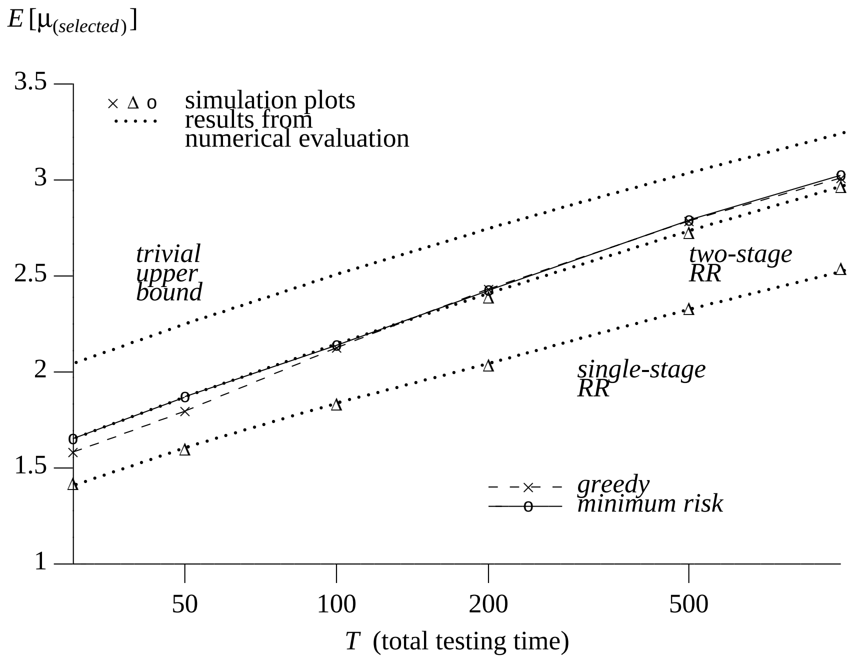


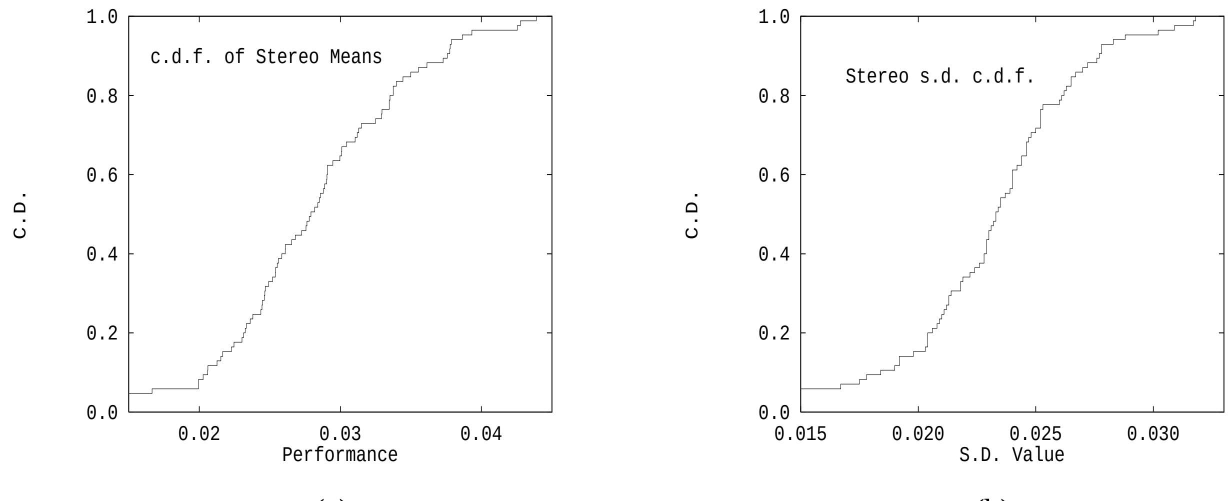
Figure 4.6. Performance comparison for single-stage round-robin, two-stage round-robin, two-stage round-robin/greedy, and two-stage round-robin/minimum-risk guidance strategies for $f_\mu = N(0,1)$ and $\sigma = 1$.

4.4. Application of Strategies

The analysis presented in the previous section demonstrates the method for determining the parameters for multistage selection. For these results to be useful, however, they must be applied to actual real-world problems. In practice, the distributions of the population means and variances are usually unknown. Therefore, it is necessary to estimate them by statistical tests before the analytical results can be applied. One approach is to allot the initial 10 percent of the available time to perform tests to acquire an estimate of these parameters. Four samples per candidate are tested using as many candidates as can be tested during this initial period.

After the statistical parameters are estimated, the parameters of the selection process are selected based on the analysis presented in the previous section. In the case of dynamic strategies, the same parameters (r, n_1, and n_2) as the two-stage round-robin strategy are used.

To evaluate this application strategy, it is necessary to model the distribution of performance values of candidate parameter sets for the stereo vision problem. This allows the use of Monte Carlo simulations to verify performance. The empirical c.d.f. for the population means and the population variances of 85 candidate parameter sets is shown in Figure 4.7. A normal distribution is fit to this using these means and variances in order to generate the random populations for testing. Under this method, the results based on this application strategy are shown in Figure 4.8.



(a) (b)

Figure 4.7. Distribution of population means (a) and population standard deviations (b) of 85 candidate parameter sets.

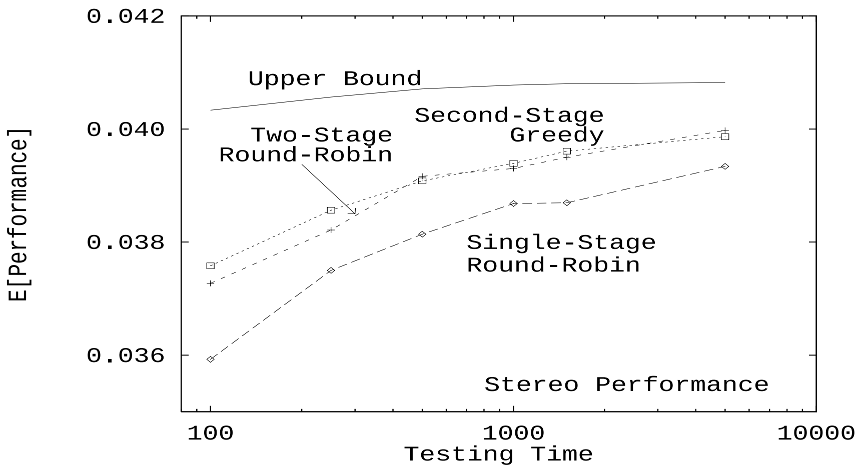


Figure 4.8. Performance of various guidance strategies assuming a normal distribution of population means in accordance with Figure 4.7 (a).

CHAPTER 5

IMPLEMENTATION DETAILS

This chapter presents the details of the specific implementation of both the stereo algorithm and the generate and test system. With this information it is clear how the real system functions and handles the variety of problems that arise.

5.1. Stereo Algorithm

The stereo algorithm used here is a general binocular token-matching algorithm. It operates on two images (hereafter referred to as the left and right images) consisting of a rectangular array of square pixels. The raw image input data are in the form of grayscale maps (8 bits per pixel). Average training images in the database are of the size 128×128 pixels, but larger images are easily accommodated. The matching of corresponding features in the images occurs on a medium level. The tokens that are matched consist of edge pixels or *edgels*, but this can easily be extended to include segments, corners, or other features. The edgel extraction is done using an enhanced version of the Canny edge detector [5]. This algorithm employs hysteresis for greater edge continuity and retains midprocessing information to be used for matching. It was selected over other methods such as the Laplacian of a Gaussian (LoG) because of the hysteresis, as well as the greater flexibility it allows in terms of adjusting the edge-detection parameters. The outline of the stereo matching code as well as a sample run are presented in Appendix C. The parameters necessary for the edge-detection stage are:

(1) the width of the Gaussian blurring kernel,

(2) the high gradient threshold for edge detection, and

(3) the low gradient threshold for edge detection.

The information retained for each edgel includes its position, its X- and Y-directional gradients, and the chain of edgels to which it is connected. (The latter is used for matching extended contours.) In addition, the position of each edgel is found with subpixel resolution.

The token matching stage is based on iterative depth refinement as discussed in Section 2.4. Figure 5.1 shows the flowchart of the stereo matching algorithm using this technique. For this, the necessary parameters include the edge-detection parameters for each channel as well as the number of channels. It is important to note that, if multiple processors are available, then a degree of the token extraction for each channel can proceed in parallel.

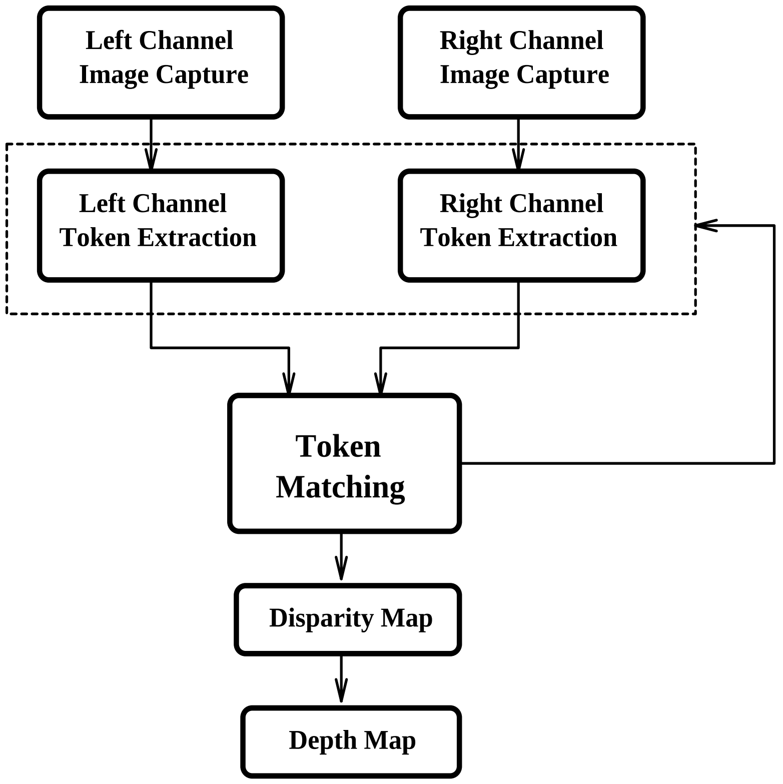

Left Channel Image Capture
Right Channel Image Capture
Left Channel Token Extraction
Right Channel Token Extraction
Token Matching
Disparity Map
Depth Map

Figure 5.1. Stereo algorithm flowchart.

Furthermore, the availability of such resources can have a significant impact on the parameters selected by the learning algorithm. This, of course, is dependent on the exact objective function set forth to measure candidate fitness. (See Section 5.2.)

There are now a variety of parameters available for tuning. For this implementation only a few were selected. The choice for the initial subset was made regarding the impact and function that the parameter had on the performance. Table 5.1 shows the value ranges for the parameters. The purpose and function of these parameters were shown in Table 2.2 previously. Note that the unit of "pixel" as used in the table is a distance measure equal to the width of one pixel. (Pixels are assumed to be square.)

Of the parameters listed, the number of channels, the blurring factor for each channel, and the edge detection gradient thresholds have been selected for actual parameter tuning implementation. These have the greatest effect on performance and are useful for verifying performance intuitively. For a typical candidate, the number of channels varied from one to five, and the σ of the blurring kernel varied from 0.1 to 5.0. The other parameters were assigned values selected by hand, and held constant throughout the experimentation.

Table 5.1. Stereo Algorithm Parameter Ranges

Parameter	**Range**	**Units**
Number of Channels	≥ 1	
Blurring Kernel Size	*0.1 – 5.0*	*pixels*
High Threshold	*1 – 255*	$\frac{\partial\, intensity}{\partial\, pixel}$
Low Threshold	*0 – 255*	$\frac{\partial\, intensity}{\partial\, pixel}$
Initial Search Window	*1 – 200*	*pixels*
Similarity Thresholds		
Gradient	*1 – 50*	$\Delta \frac{\partial\, intensity}{\partial\, pixel}$
Orientation	$\lvert \bullet \rvert \leq 30$	*degrees*
Vertical Position	$\pm\ 1 - 3$	*pixels*

The final form for a candidate parameter can be viewed as a set of triplets. Each triplet gives the parameters for a particular channel. For example, the expression $\{(\sigma_1, lt_1, ht_1), (\sigma_2, lt_2, ht_2)\}$ would represent the parameters comprising a two-channel candidate.

5.2. Objective Function

The particular objective function that the user defines will indirectly determine the type of candidates that the system finds to perform well. The objective function used here, therefore, is more sophisticated than the naive form shown in Eq. (2.1). The desired form of the objective function would be a function of only the stereo vision algorithm parameters and would return a measure of quality. However, because this relationship is unknown, the objective function is expressed as a function of intermediate values gathered by performing tests on the candidate. The view taken in formulating it was that a fixed amount of time is allowed for completion of the algorithm. Exceeding (and even approaching) this limit penalizes the fitness of a candidate. This is consistent with many real-world situations in which the vision system plays a role in a chain of cognitive acts. In this situation the vision system is given a fixed amount of time to perform its function. Performing faster than this deadline would not be considered particularly useful. Other than the speed of the stereo algorithm, the only aspects considered were to maximize both the density and accuracy. Density is measured in terms of the fraction of the image for which corresponding edgels were found. Accuracy is measured by probing the resulting disparity map and comparing the values with hand-calculated disparities. For the test images in this database, ten points were measured per image. The accuracy used by the objective function is the normalized average disparity error with 0 error mapping to 1, and an error of $e_{\max}$ pixels or more mapping to 0. (The value of $e_{\max}$ used for this system was varied to simulate the effect of different application requirements.) The objective function then is a product of density (in matches per square pixel), normalized accuracy, and the time penalty function. The penalty function, $p(t)$, is a function of time for the test, t, and is shown in Eq. (5.1).

$$p(t) = \frac{1 + e^{-q t_0}}{1 + e^{q (t - t_0)}} \tag{5.1}$$

The parameter q allows tailoring the penalty function with respect to the time limit t_0. The graph of the penalty function for $q = 0.05$, $t_0 = 120$ seconds is shown in Figure 5.2.

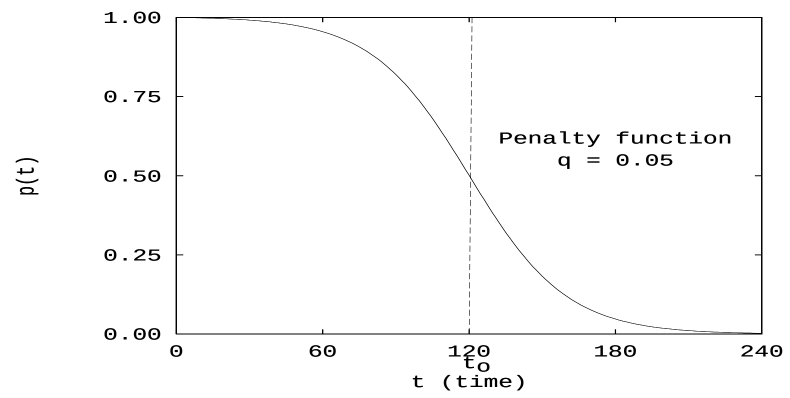


Figure 5.2. Time penalty function.

The final form of the objective function (Eq. (5.3)) is the average quality of the candidate as measured over each test image in the database. The quality measure over one image, i, with parameter set, P, is $q(i, P)$. This is a function of three intermediate test results which are each functions of the image and the algorithm parameters. The three components are density of disparity measurements, $d(i, P)$, average accuracy of disparity measurements, $a(i, P)$, and time to process the image $t(i, P)$.

$$q(i, P) = d(i, P) \times a(i, P) \times p(t(i, P)) . \tag{5.2}$$

The objective function, $O(P)$, now takes the form shown in Eq. (5.3) where P is the set of candidate parameters and B is the set of the N images in the database.

$$\begin{aligned} O(P, B) &= q_{avg} \\ &= \frac{1}{N} \sum_{i \in B} q(i, P) . \end{aligned} \tag{5.3}$$

The objective function stated here is really but one in a family of possible objectives. By varying components such as t_0, q, and $e_{\max}$, the user can make the system search for particular parameter sets for different applications (cf. Tables 6.2 and 6.3). This kind of information is of interest because it shows how the system focuses towards different classes of candidates under different system requirements.

5.3. Candidate Evaluation

Testing of the candidates was performed on a Sparc IPC workstation using the general binocular stereo algorithm implemented in C. Typical execution times were roughly 20 seconds per channel on the 128×128 grayscale images. Of course, this time could vary widely depending on the number of tokens found in each channel.

5.4. Image Database

The training image database was obtained as a series of black and white photographs with a standard camera. These images were taken in outdoor daytime conditions and consisted primarily of building scenes. This was done to ensure that the database covered a specific class of images (daytime and edge-oriented), yet had the complex properties inherent in real-world scenes (as opposed to those of synthetic images).

The images were digitized with a Sharp 24-bit color scanner at roughly 85 dpi and then quantized to 256 gray levels. The size of the images after digitization was 256×256 pixels.

For training purposes, however, the images were broken into 4 pieces each, thereby making each test image 128×128 pixels.

For the purposes of stereo matching, the stereo images were hand-rectified and mounted to ensure a straight epipolar line. Due to the lack of a tripod and the effect of foreshortening, however, some images could not be fully rectified throughout the scene. The stereo matcher does try to accommodate for some error along the epipolar, but nevertheless, this effect served as an added impediment to effective matching. Accuracy points were measured by finding correspondences by hand on magnified copies of the images. Accuracy measurements made this way are precise to within one pixel. Regardless of some of the above problems, the database serves as a fair representation of real-world images and challenges the system with many of the difficulties that one would expect to encounter. The entire database is presented in Appendix D.

5.5. Candidate Generation

Initially, some sample parameter sets (or candidates) are specified by the user. As testing progresses, however, the system needs to be able to create its own candidates. The generation of new candidates is handled by using rules to represent expert knowledge in the area of candidate generation. Due to the limited amount of time available for testing, the candidate generator should refer to the performance of previous candidates when generating new ones. Currently, two methods have been investigated: random and greedy. In the random approach, a new candidate is generated by a random perturbation from one of the existing candidates. The candidate used as a basis for this perturbation must lie in the top third of the existing candidates. In the greedy approach, the generator tries to follow the "direction" of the greatest improvement in performance, based on the performance of candidates already generated. The direction is expressed as a vector of the delta values between the parameters of the two candidates.

The current rule-based implementation is flexible enough to allow adding further domain knowledge as it becomes available. New operators and rules can easily be added to give incremental improvement. The rule base consists of a set of assertions called the *working memory* and a set of rules for operating on it. A rule consists of two parts: a list of conditions and a list of actions. When all conditions within a rule are met, the rule is said to fire, and all the associated actions are carried out. In general, a rule-based system allows asynchronous execution of rules by relying on a conflict resolution strategy to select one rule at a time to fire. In this case, however, the rule base executes the rules sequentially. A condition can be either a pattern or a function. A pattern is considered met by finding a unification between the pattern and an assertion within the working memory. A function is considered met when it returns a nonempty value. Variable assignments and wildcards can be used in both patterns and functions. There are three types of basic actions: (1) delete an existing assertion from working memory, (2) add a new assertion to working memory, and (3) execute a function. Actions can use values of variables assigned in the conditional parts of the rules. This approach allows a very flexible system for analyzing existing candidates and generating new ones.

The creation of a candidate follows a regular procedure. During the creation process, only well-tested candidates are considered worthy of reference. (Relatively untested candidates are avoided because their goodness may not yet be particularly reliable or stable.) A candidate is considered well-tested if the confidence that the sample mean lies within a certain range of the population mean is above a given level. The first step in candidate creation is to select a reference candidate from which to create the new one. This candidate is randomly selected from the top third of the well-tested candidates in the pool. Next, the candidate's neighbors are checked to see if they have better performance. If so, then the reference candidate is updated to be the best neighbor. This candidate will now serve as the basis for the new candidate. Following this, the neighbors of the reference candidate are again polled for their performance values. The "direction" of the greatest change in performance is then recorded. This direction is represented as a numerical vector that specifies the change in the parameters between the two candidates. The vector is called a *transform*. If no existing candidates lie in the path of this transform, then the transform is applied to the reference candidate to generate the new one. However, if this transform leads into a region of the parameter space already occupied by a candidate, then a random transformation is created to move into unexplored space. The random transform is created by first recording the transforms from the reference candidate to the neighbors. One of the channels of the transform is then selected for modification. An individual replacement channel transformation is then randomly created. If it is found to be similar to any of the existing transforms for that channel, then the channel transform is recreated, but with a greater magnitude. This process continues until the new transform leads to an unexplored region of the parameter space.

This method of candidate generation combines two important concepts. The first is that existing good transformations are followed. This gives an effect similar to hill-climbing on the parameter space. The second benefit of this method is that randomness is employed as a hedge against stagnation in a local-maximum on the objective function surface. This randomness is exploited in two ways. First, it is used as the reference candidate and is set by randomly selecting from the top third of all candidates, rather than just using the top, or *incumbent*, candidate. Second, as the local regions of the parameter space become more well-known, the transforms become more radical, thus ensuring a breakaway from the current region.

Table 5.2 lists the candidate generation rules more explicitly. In the table, C_{ref} is the chosen reference candidate, $C_{\max}$ is the best neighbor of the reference candidate, and C_{inc} is the incumbent (or best) candidate. The precise rules used by the system are presented in Appendix A.

Table 5.2. Rules Used for Candidate Generation

Rule No.	Description
1	Randomly select C_{ref} from the top third of the well-tested candidates.
2	If there is a candidate in the same neighborhood as C_{ref} that is better than C_{ref} and has a greater quality disparity from C_{ref} than all others in the neighborhood, then use that candidate instead of C_{ref}.
3	If no candidate in the neighborhood of C_{ref} has a performance in the top $2/3$ of all well-tested candidates, then use C_{inc} instead of C_{ref}.
4	Find a candidate, $C_{\max}$, with the greatest disparity from C_{ref} among all candidates in the neighborhood. Record $T_{\max}$ as the transformation that creates this disparity and Δ as the change in quality.
5	Use the transform that caused the highest increase in performance in the neighborhood of C_{ref} as the basis transform for generating a new candidate.
6	If there are no existing candidates in the direction of the basis transformation, then use the basis transformation to generate the new candidate.
7	Otherwise, generate progressively greater transforms until one is found that leads to an unexplored region of the parameter space. Then apply this transform to C_{ref} to generate the new candidate.

CHAPTER 6

EXPERIMENTAL RESULTS

This chapter presents a verification of the effectiveness of the methods described in the previous chapters. The results were generated from actual runs of different durations and with different objective function formulations. (Both actual runs and simulated runs are presented.) Before performing the experiments in the actual runs, the candidate pool was seeded with six candidates that were generated by hand (see Table 6.1). All experiments started with these candidates to show the different types of candidates that can be discovered as the available time and the objective function are modified. From these initial candidates, the rules shown in Table 5.1 were applied to generate new candidates as necessary. At the end of each run, all of the candidates were then fully evaluated over all of the images in the database. (Here the database consisted of 30 128×128 images.) This full evaluation allows judging the performance of the system by giving insight into the performance/time trade-off. This performance loss is the difference between the best found under the guided system and the best found by exhaustive tests. The time gain is the amount of time saved by using a guidance strategy rather than blind exhaustive tests. The full evaluation corresponds to finding the values of the objective function for the given candidates.

During actual tests, the testing strategy that was used was a two-stage round-robin/minimum-risk strategy. The division of time between the two stages was equal, with f=0.5. This fraction was determined heuristically from experiments using a simulator on a variety of problems. This value appears to be fairly robust. Presampling was performed for 10% of the available time to determine the parameters μ_0, σ_0^2, and $\overline{\sigma^2}$. These parameters were then used to determine the two-stage strategy parameters of n_1 and n_2 (cf. Section 5.2). During the available presampling time, tests were taken from as many candidates as possible while still ensuring that 4 tests were performed on each selected candidate.

Results from a combination of actual and simulated runs are combined to present a full picture, because it is very time-consuming to perform extensive tests on candidates whose individual test times can range from 1 to 2 minutes. For 200 candidates tested on 30 images, the total testing time can last up to 7 days. For this reason, it can be beneficial to perform additional simulations using the results recorded from the actual tests, if the simulations are close to observed actual results. As simulation can take less than an hour to complete, it clearly results in a great time savings. The problem with simulation, however, is that candidate generation is not modeled. Therefore, the candidates that were generated by the one

Table 6.1. Hand-tuned Seed Candidates

Candidate number	**Channel width**	**Low threshold**	**High threshold**
1	1.3	2.0	5.0
2	0.6	2.0	5.0
3	0.9	2.0	5.0
	0.6	2.0	5.0
4	1.8	2.0	5.0
	1.5	2.0	5.0
5	2.4	2.0	5.0
	1.7	2.0	5.0
	1.0	2.0	5.0
6	3.0	2.0	5.0
	1.0	2.0	5.0
	0.4	2.0	5.0
7	1.8	2.0	5.0
	1.3	2.0	5.0
	0.8	2.0	5.0
	0.6	2.0	5.0

---

recorded run will be the only ones used during the simulation. Nevertheless, empirical evidence has shown that simulation performance is very close to that of actual tests. Some justification for this conclusion can be found by comparing the results shown in Table 6.2 for real tests (those with candidate generation) and the results shown in Table 6.3 (the simulations).

For the actual (nonsimulated) tests of the system, three objective functions were tested for three different durations. The parameters of the objective functions that were used as well as the test durations are presented in Table 6.2. Each entry of this table lists the identity of the best parameter set that was found, as well as its performance, i.e., the value of the objective function after 30 tests. The parameter set for each channel is encoded in the form channel width - low-threshold - high-threshold with the separate channels separated by a "/". The number in parentheses at the end of the candidate name indicates that this was the nth candidate generated to be tested.

Table 6.2. Performance from Actual Tests

Objective Parameters	**Total Duration (in # of tests)**	
	200	**400**
$t_0 = 60$ $e_{\max} = 5$	2.4-2.0-5.0/1.7-2.0-5.0/1.3-0.5-2.2(22) 0.0369	2.4-2.0-6.6/1.5-1.7-5.0/1.0-2.0-5.0(18) 0.0416
$t_0 = 60$ $e_{\max} = 2.5$	1.3-0.5-4.3/1.5-2.0-4.8(29) 0.0143	2.4-2.5-5.0/1.7-2.0-5.0/1.0-1.6-5.0(140) 0.0251
$t_0 = 45$ $e_{\max} = 5$	1.6-1.4-2.5/0.7-4.0-8.3(30) 0.0284	3.0-2.0-5.0/1.4-2.1-2.1/1.1-1.7-2.2(109) 0.0430
$t_0 = 30$ $e_{\max} = 2.5$	2.4-2.0-5.0/1.4-2.0-6.2/1.0-2.0-5.0(48) 0.0199	1.8-0.6-5.0/1.7-4.0-4.9/1.3-2.0-5.0/0.9-0.7-5.6(134) 0.0200

Simulated results for the case of a fixed duration and different objectives are shown in Table 6.3. Here two of the parameters comprising the objective function are shown on different axes of the table. Each entry in the table shows the identity of the best candidate as well as its performance value. The testing duration for these tests was 400 quanta.

Table 6.3. (a) Performance from Simulations with Varying Objectives

Time Limit (t_0)	**Error Cut-off ($e_{\max}$)**			
	2.5	**5**	**7.5**	**10**
5	2.4-2.0-5.0/ 1.7-2.0-5.0/ 1.0-2.0-5.0 0.0163	1.5-2.3-2.7(24) 0.0225	3.0-2.0-5.0/ 1.4-2.1-2.1/ 1.1-1.7-2.2(109) 0.0319	1.5-2.3-2.7(24) 0.0342
10	2.3-2.0-5.0/ 2.3-1.0-4.3/ 1.0-2.0-3.7(156) 0.0146	3.1-1.2-5.8/ 1.1-4.6-4.6/ 0.8-1.1-3.7(122) 0.0225	2.8-2.0-5.0/ 1.9-3.5-4.4/ 1.2-2.5-6.1(48) 0.0312	1.5-2.3-2.7(24) 0.0350
20	2.2-0.7-3.6/ 1.5-2.4-3.1/ 1.0-2.2-5.0(34) 0.0167	2.7-2.7-7.2/ 1.7-2.0-5.0/ 1.0-2.0-3.7(85) 0.0264	3.0-2.0-5.0/ 1.4-2.1-2.1/ 1.1-1.7-2.2(109) 0.0384	3.0-2.0-5.0/ 1.4-2.1-2.1/ 1.1-1.7-2.2(109) 0.0408
30	2.2-0.7-3.6/ 1.5-2.4-3.1/ 1.0-1.5-4.1(58) 0.0173	2.8-2.0-5.0/ 1.9-3.5-4.4/ 1.2-2.5-6.1(48) 0.0319	3.0-2.0-5.0/ 1.4-2.1-2.1/ 1.1-1.7-2.2(109) 0.0431	2.4-1.0-6.4/ 1.7-2.0-5.0/ 1.0-2.0-3.7(104) 0.0390
40	2.4-2.0-3.8/ 1.7-2.7-4.0/ 0.6-1.5-4.5(94) 0.0173	2.4-3.2-7.9/ 1.7-2.0-5.0/ 1.0-2.9-3.7(121) 0.0344	2.4-3.2-7.9/ 1.7-2.0-5.0/ 1.0-2.0-3.7(98) 0.0399	2.7-2.7-7.2/ 1.7-2.0-5.0/ 1.0-2.0-3.7(85) 0.0424
50	3.0-2.0-5.0/ 1.6-1.4-4.6/ 0.8-1.1-3.7(58) 0.0246	2.8-2.0-5.0/ 1.3-3.5-4.6/ 0.6-2.5-6.1(39) 0.0342	2.4-1.0-6.4/ 1.7-2.0-5.0/ 1.0-2.0-3.7(104) 0.0433	2.4-2.0-2.8/ 1.7-2.7-4.0/ 0.6-2.5-6.1(96) 0.0434
60	2.4-2.0-5.0/ 2.1-2.0-5.4/ 1.0-0.5-5.2(69) 0.0266	2.4-1.0-6.4/ 1.7-2.0-5.0/ 1.0-2.0-3.7(104) 0.0392	2.4-1.3-6.3/ 1.7-2.0-5.0/ 1.0-2.0-3.7(45) 0.0460	2.8-2.0-5.0/ 1.3-3.5-4.6/ 0.6-2.5-6.1(39) 0.0456

Table 6.3. (b) Performance from Simulations with Varying Objectives

Time Limit (t_0)	**Error Cut-off ($e_{\max}$)**			
	2.5	**5**	**7.5**	**10**
70	2.4-1.0-6.4/ 1.7-2.0-5.0/ 1.0-2.0-3.7(104) 0.0279	2.7-2.6-5.0/ 1.4-2.1-6.2/ 1.0-2.0-2.4(75) 0.0400	2.8-2.0-5.0/ 1.9-3.5-4.4/ 1.2-2.5-6.1(48) 0.0448	3.9-2.5-6.6/ 1.4-1.0-4.8/ 1.1-1.7-2.2(143) 0.0495
80	3.0-2.0-5.0/ 1.4-2.1-2.1/ 1.1-1.7-2.2(109) 0.0326	2.9-0.5-4.4/ 2.2-2.0-5.0/ 1.0-2.0-3.7(151) 0.0405	2.7-2.6-5.0/ 1.4-2.1-6.2/ 1.0-2.0-2.4(75) 0.0484	2.4-3.2-7.9/ 1.7-2.0-5.0/ 1.0-2.0-3.7(98) 0.0538
90	3.0-2.0-5.0/ 1.4-2.1-2.1/ 1.1-1.7-2.2(109) 0.0330	3.0-2.0-5.0/ 1.0-2.0-2.5/ 1.3-0.7-0.7(30) 0.0411	1.8-2.0-5.0/ 1.3-2.0-5.0/ 0.8-2.0-5.0/ 0.9-1.4-1.4(25) 0.0478	3.0-2.0-5.0/ 1.4-2.1-2.1/ 1.1-1.7-2.2(109) 0.0607
120	3.0-2.0-5.0/ 1.4-2.1-2.1/ 1.1-1.7-2.2(109) 0.0336	2.7-2.6-5.0/ 1.4-2.1-6.2/ 1.0-2.0-5.0(44) 0.0429	3.0-2.0-5.0/ 1.4-2.1-2.1/ 1.1-1.7-2.2(109) 0.0581	2.7-2.6-5.0/ 1.4-2.1-6.2/ 1.0-2.0-5.0(44) 0.0536
240	2.4-2.0-5.0/ 0.8-4.2-4.6/ 0.6-2.5-6.1(124) 0.0239	2.4-2.0-5.0/ 2.1-2.0-5.4/ 1.0-0.5-5.2(69) 0.0420	2.8-2.0-5.0/ 1.5-3.0-5.5/ 0.6-2.5-6.1(55) 0.0478	2.6-2.0-6.6/ 2.2-3.4-5.1/ 0.9-2.5-6.2(132) 0.0535
300	2.3-2.0-5.0/ 1.9-2.0-3.4/ 1.0-0.5-5.2(59) 0.0277	3.0-2.0-5.0/ 1.4-2.1-2.1/ 1.1-1.7-2.2(109) 0.0512	2.4-2.0-5.0/ 1.7-2.0-5.0/ 0.9-2.3-3.8(64) 0.0475	3.0-2.0-5.0/ 1.4-2.1-2.1/ 1.1-1.7-2.2(109) 0.0215
480	3.1-1.2-5.8/ 1.1-4.6-4.6/ 0.8-1.1-3.7(122) 0.0300	3.1-1.2-5.8/ 1.1-4.6-4.6/ 0.8-1.1-3.7(122) 0.0445	3.0-2.0-5.0/ 1.6-1.4-4.6/ 0.8-1.1-3.7(58) 0.0494	2.8-2.0-5.0/ 1.1-5.5-5.8/ 0.6-2.5-6.1(90) 0.0529

As can be seen from Table 6.3 (a) and (b), the objective function plays a major role in determining the sought after solution. The difference in the results for varying t_0 and $e_{\max}$ demonstrates the two purposes of the objective function: the ability to express the goals of the designer and to guide the search in an automated fashion. It is also beneficial to see the effect of extending the available testing time. As would be expected, spending more time should produce better solutions. Simulated results for the case of a fixed objective and different testing durations and guidance strategies are shown in Table 6.4. The objective function used here was with $t_0 = 30$, and $e_{\max} = 2.5$.

As stated in Section 5.2.1, it was assumed that the populations were of normally distributed numbers. Although this cannot be guaranteed for all imaginable candidates, the performance values of typical candidates fit the normal assumption well. One hundred different candidate parameter sets were evaluated over 30 test images. Each set of 30 points was then tested for normality using both the Kolmogorov-Smirnov test and the Geary test [6]. The results summarized in Table 6.5 demonstrate the validity of the normality assumption.

It was also assumed in Section 5.2.1 that the population means themselves were i.i.d., i.e., the mean from each population came from the same distribution and was independent of the others. This is not true, however, because newly generated candidates are based on previous ones. Therefore, a dependence is almost guaranteed. Nevertheless, the empirical evidence demonstrates that this fact has little bearing on the actual performance. As can be seen from Figure 4.7 (a), the overall distribution is still primarily normal, and it is this fact that is most important.

6.1. Pictorial Results

The best candidates found by the various strategies listed above are difficult to appreciate when the results are listed in a strictly numerical format. This section presents some graphical results in the form of input images and matched results for the best performing seed candidate and the candidate found after 200 tests. These were generated with $t_0 = 30$, and $e_{\max} = 5$. Each set of images consists of the stereo pair, the left and right edge-maps found on the original channels, and those found on the discovered channels. The 3-D relief of the matched edges and the disparity map are also presented for both parameter sets. The disparity map is generated by interpolating the entire image area by nearest neighbor disparity values. It is meant to give only an indication of depth throughout the scene, since this type of interpolation does *not* preserve object outlines and appearances. The left and right images are displayed with the right image on the left so that it is possible for the viewer to

Table 6.4. Performance from Simulations with Varying Durations and Strategies

Duration	Strategy			
	Single-stage Round-robin	**Two-stage Round-robin**	**Two-stage RR/Greedy**	**Two-stage RR/Min-risk**
100	2.4-2.0-5.0/ 1.6-3.0-5.1/ 0.6-2.5-6.1(81) 0.0167	2.3-2.0-5.0/ 1.7-2.0-5.0/ 1.4-2.3-5.6(155) 0.0170	1.3-3.1-5.3/ 1.5-2.0-4.8(41) 0.0125	2.8-1.0-5.0/ 1.5-3.0-5.5/ 0.6-2.5-6.1(87) 0.0151
200	2.3-2.0-5.0/ 1.7-2.0-5.0/ 1.0-2.0-3.7(61) 0.0176	2.6-2.0-6.6/ 1.7-4.5-4.5/ 0.6-2.5-6.1(93) 0.0193	2.4-2.8-7.0/ 1.7-2.0-5.0/ 1.0-2.0-3.7(69) 0.0188	3.0-2.0-5.0/ 1.6-1.4-4.6/ 0.8-1.1-3.7(58) 0.0203
300	2.3-2.0-5.0/ 1.9-2.0-3.4/ 1.0-0.5-5.2(59) 0.0198	2.7-2.6-5.0/ 1.4-2.1-6.2/ 1.0-2.0-5.0(44) 0.0216	2.6-2.0-6.6/ 2.2-3.4-5.1/ 0.9-2.5-6.2(132) 0.0192	3.0-2.0-5.0/ 1.6-1.4-4.6/ 0.8-1.1-3.7(58) 0.0203
400	2.4-2.0-3.8/ 1.7-2.7-4.0/ 0.7-2.5-4.4(137) 0.0205	2.8-2.0-5.0/ 1.3-3.5-4.6/ 0.6-2.5-6.1(39) 0.0251	2.4-2.0-5.0/ 2.3-3.4-5.1/ 0.6-1.0-6.1(107) 0.0158	3.0-2.0-5.0/ 1.4-2.1-2.1/ 1.1-1.7-2.2(109) 0.0251
500	3.0-3.3-5.0/ 1.6-1.5-1.5/ 0.8-0.8-3.7(144) 0.0200	3.0-2.0-5.0/ 1.4-2.1-2.1/ 1.1-1.7-2.2(109) 0.0251	3.0-2.0-5.0/ 1.4-2.1-2.1/ 1.1-1.7-2.2(109) 0.0251	2.6-2.0-6.6/ 1.7-4.5-4.5/ 0.6-2.5-6.1(93) 0.0193

---

merge them by crossing the eyes if desired. One thing that is not appreciable from the pictures, however, is the speed-up in execution. In all cases, the machine-tuned parameters performed faster and resulted in a 15% to 30% improvement as measured by the objective function. In the cases shown here, the best original parameter set was candidate number 5 from Table 6.1. The best parameter set found after a simulation with 600 time units with the above stated objective was used for the results below. This candidate was 2.2-2.0-7.0/1.6-3.8-3.8/1.3-3.0-3.6(41). Performance measures of each image are presented separately.

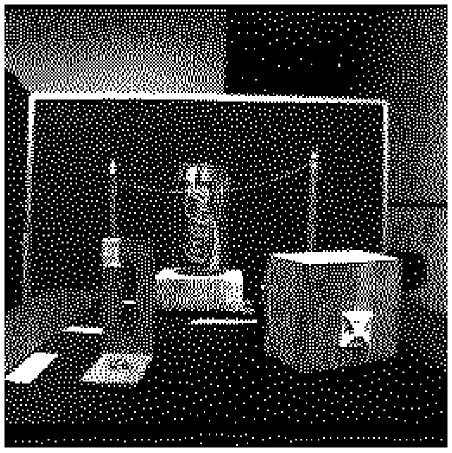
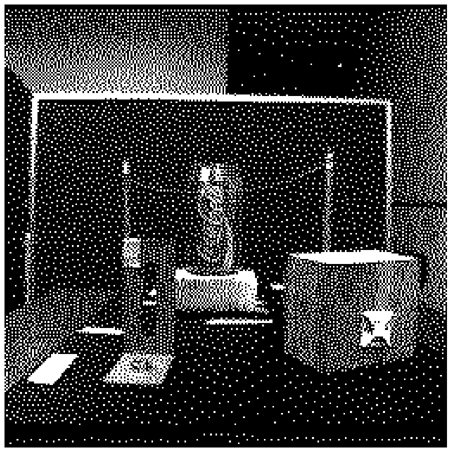
Table 6.5. Test of Normality with $t_0 = 30$ seconds, and $e_{\max} = 2.5$. A total of 327 candidates, each with 30 points, were tested. At $\alpha = 0.05$, the Kolmogorov-Smirnov test with 30 samples requires the value to be less than 0.24, while the Geary test at $\alpha = 0.05$ requires the absolute value to be less than 2.042 [6, 16].

Test	**Min**	**Avg**	**Max**	**# Fails / Total**
Kolmogorov-Smirnov ($\alpha = 0.2$, KS < 0.23)	0.073	0.199	0.418	68/327
Geary ($\alpha = 0.05$, $\lvert G \rvert < 2.042$)	0.0086	1.489	3.588	77/327

This first example, shown in Figure 6.1, is a coke can scene obtained from Dr. Sridhar at NASA Ames Research Center. It is shown here just to demonstrate sample performance on a relatively simple scene. The original parameter setting processed the scene in 62.23 seconds, while the discovered parameters took only 51.64 seconds. On top of this improvement, it is noticeable from Figures 6.1 (o) and 6.1 (p) that the matching was slightly improved.

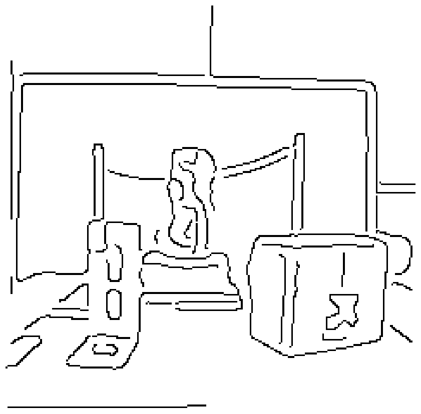
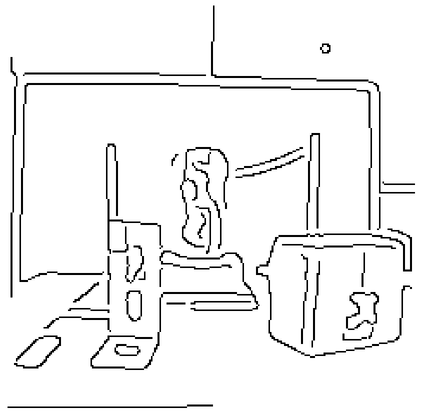
Figure 6.2 shows a sample building scene. This image was processed in 115.26 seconds with the original parameters, and 78.83 seconds with the discovered ones. The accuracy, however, dropped from 2.83 pixels to 3.12, but as the objective function emphasized speed, the new parameter set had a performance of 0.021 as opposed to the original 0.017.

Figure 6.3 shows a sample street scene. This image was processed in 130.71 seconds with the original parameters, and 100.10 seconds with the discovered ones. In this case, the accuracy dropped from 1.4 pixels to 2.4. Nevertheless, the great increase in speed allowed the objective function to increase from 0.024 to 0.034.

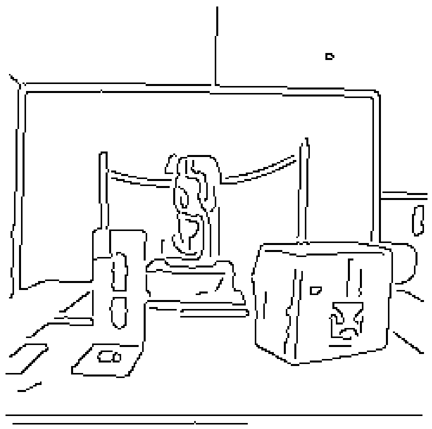
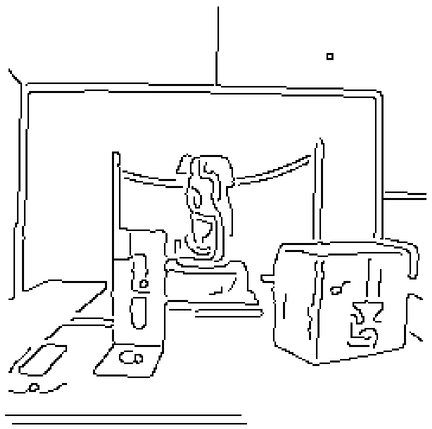
(a) (b)

Figure 6.1. (a) Right image, (b) Left image.

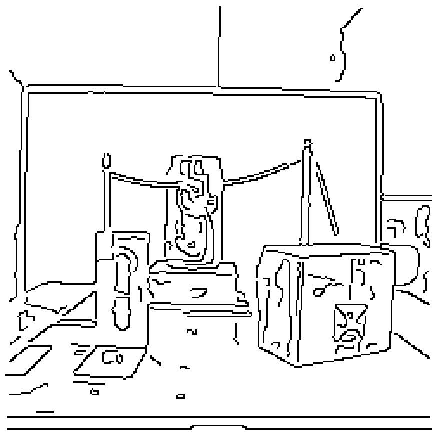
(c) (d)

Figure 6.1. (c) Original right channel 1, (d) Original left channel 1.

(e) (f)

Figure 6.1. (e) Original right channel 2, (f) Original left channel 2.


(g)

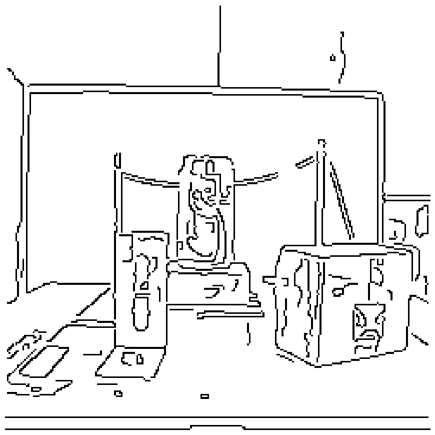
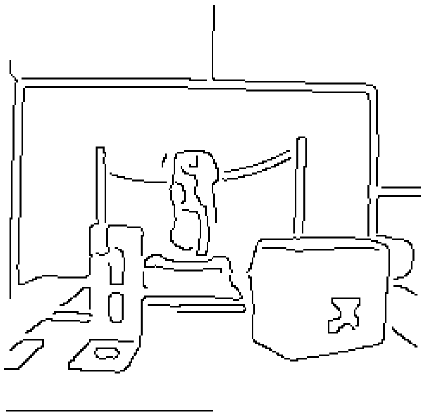

(h)

Figure 6.1. (g) Original right channel 3, (h) Original left channel 3.


(i)

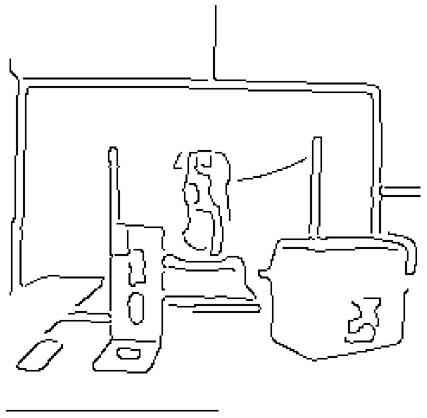
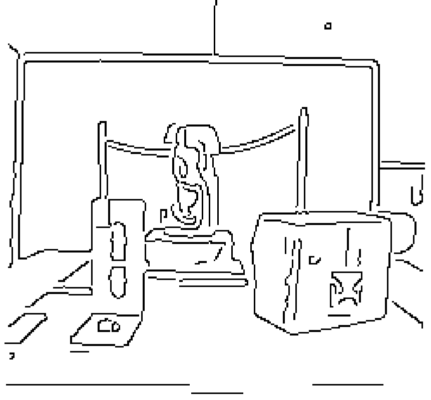

(j)

Figure 6.1. (i) New right channel 1, (j) New left channel 1.


(k)

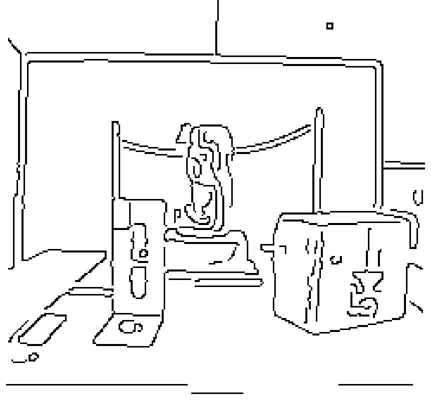
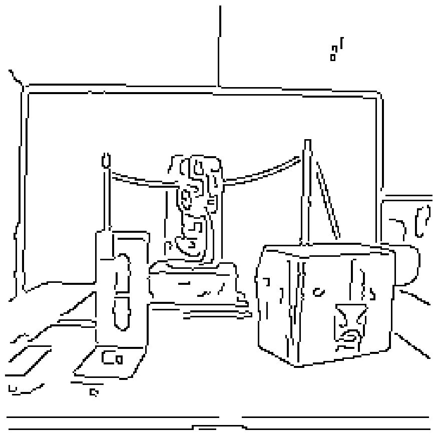
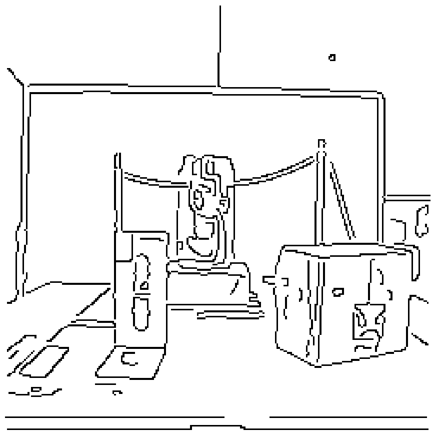
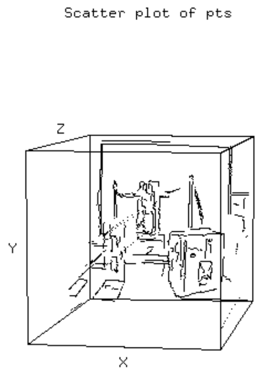

(l)

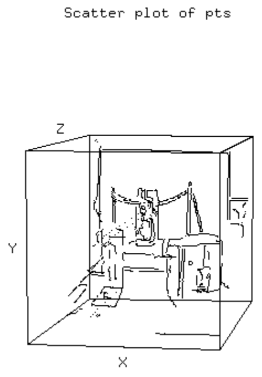
Figure 6.1. (k) New right channel 2, (l) New left channel 2.

(m) (n)

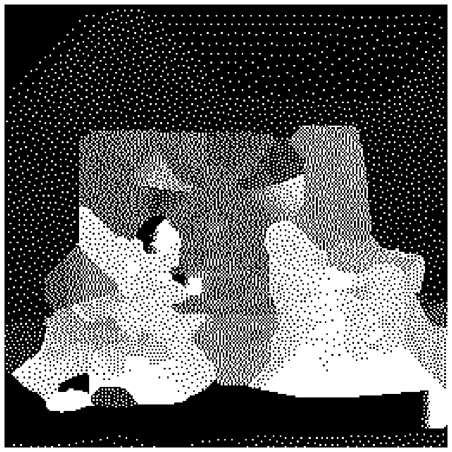
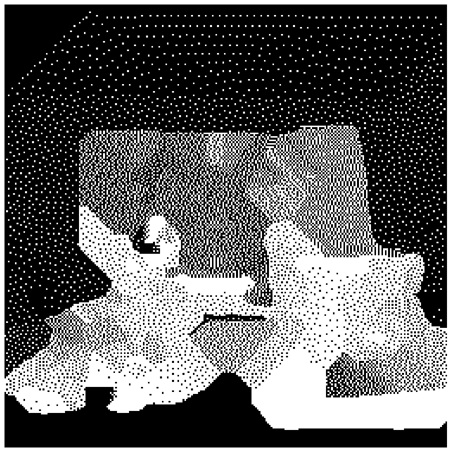
Figure 6.1. (m) New right channel 3, (n) New left channel 3.





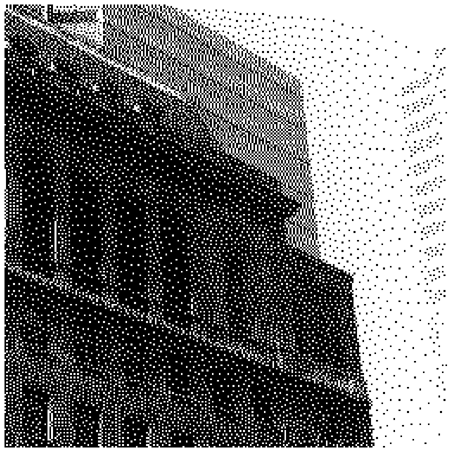
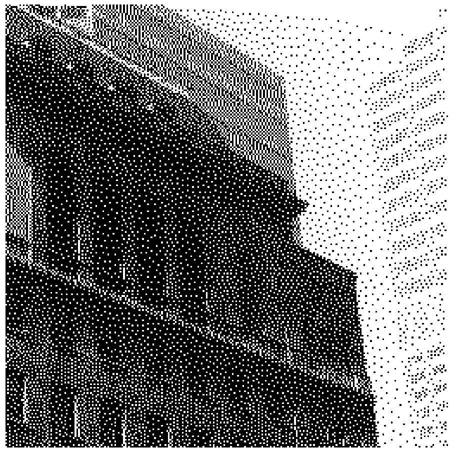
(o) (p)

Figure 6.1. (o) 3-D edges originally, (p) 3-D edges after tests.

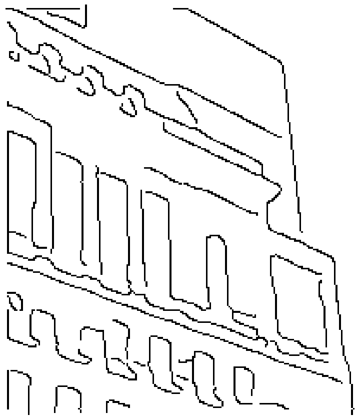
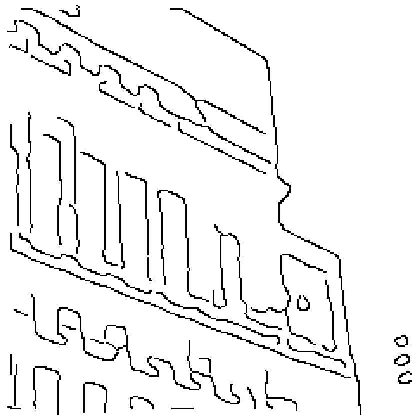
(q) (r)

Figure 6.1. (q) Original disparity, (r) Disparity after tests.

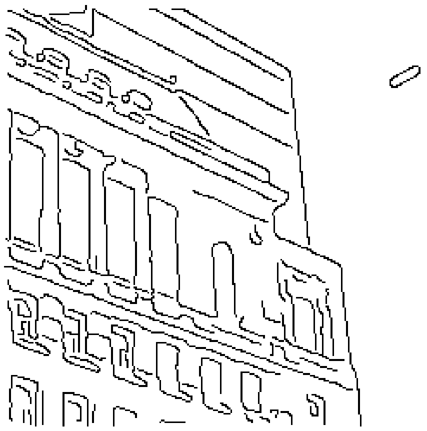
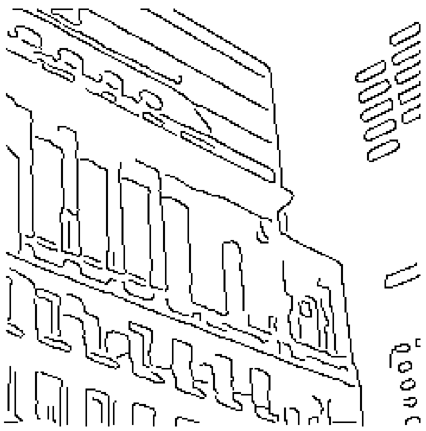
(a) (b)

Figure 6.2. (a) Right image, (b) Left image.

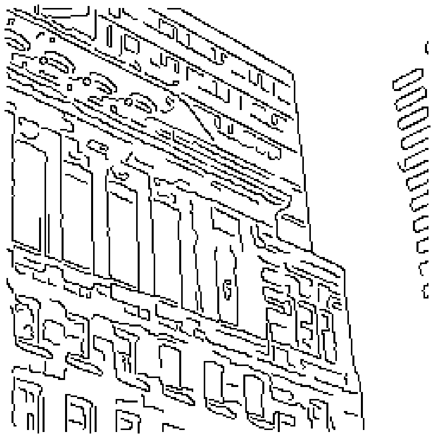
(c) (d)

Figure 6.2. (c) Original right channel 1, (d) Original left channel 1.

(e) (f)

Figure 6.2. (e) Original right channel 2, (f) Original left channel 2.


(g)

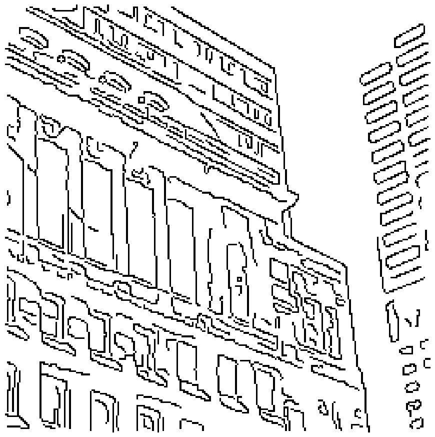
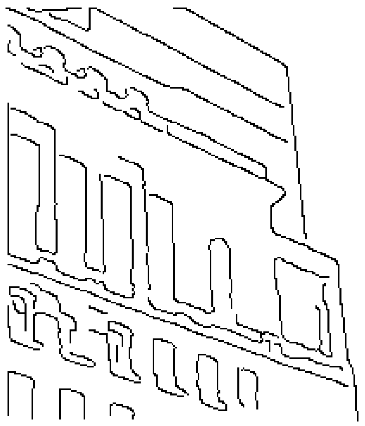

(h)

Figure 6.2. (g) Original right channel 3, (h) Original left channel 3.


(i)

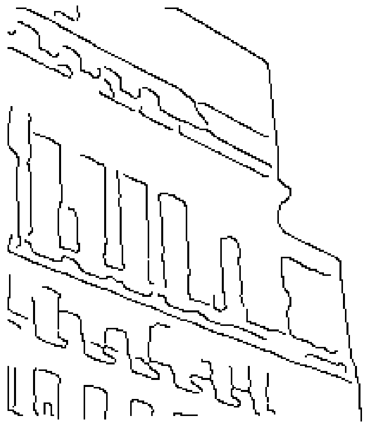
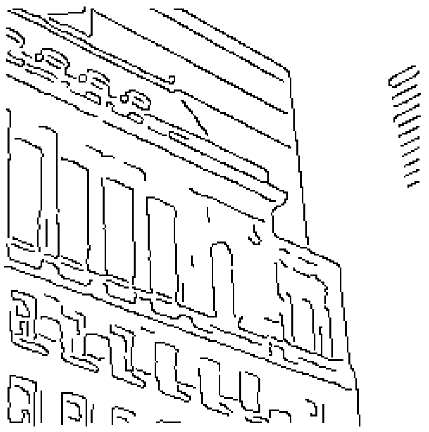

(j)

Figure 6.2. (i) New right channel 1, (j) New left channel 1.


(k)

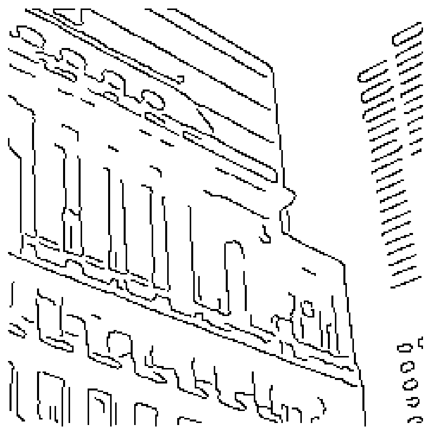
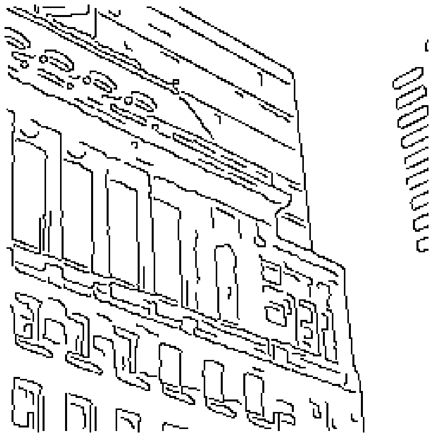
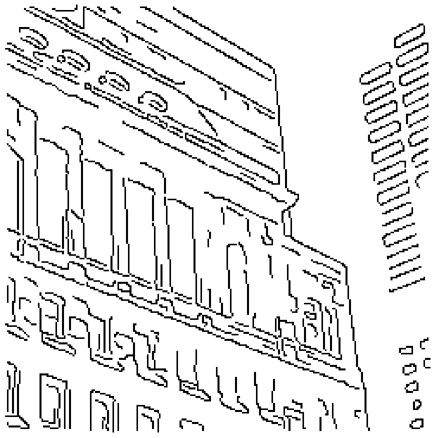
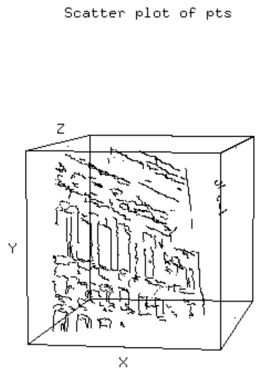

(l)

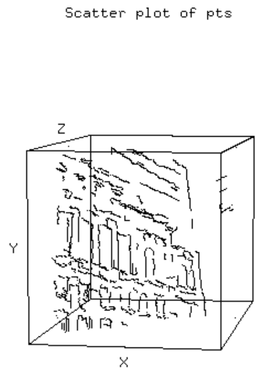
Figure 6.2. (k) New right channel 2, (l) New left channel 2.

(m) (n)

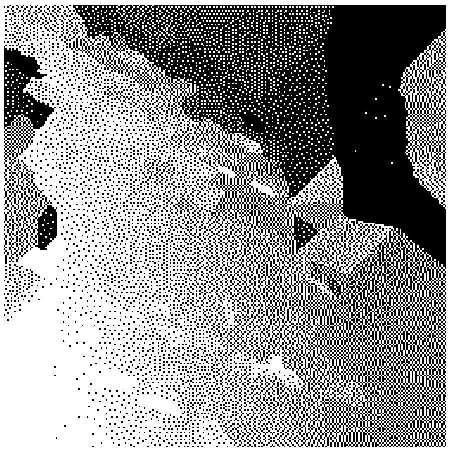
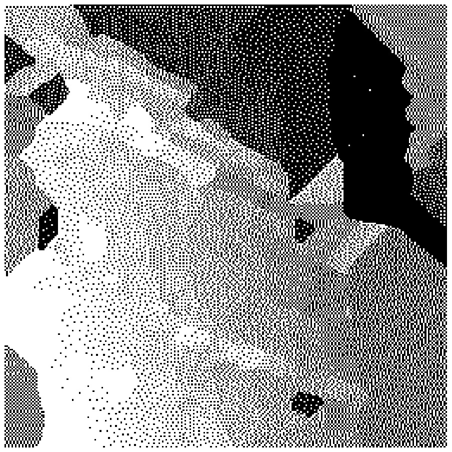
Figure 6.2. (m) New right channel 3, (n) New left channel 3.





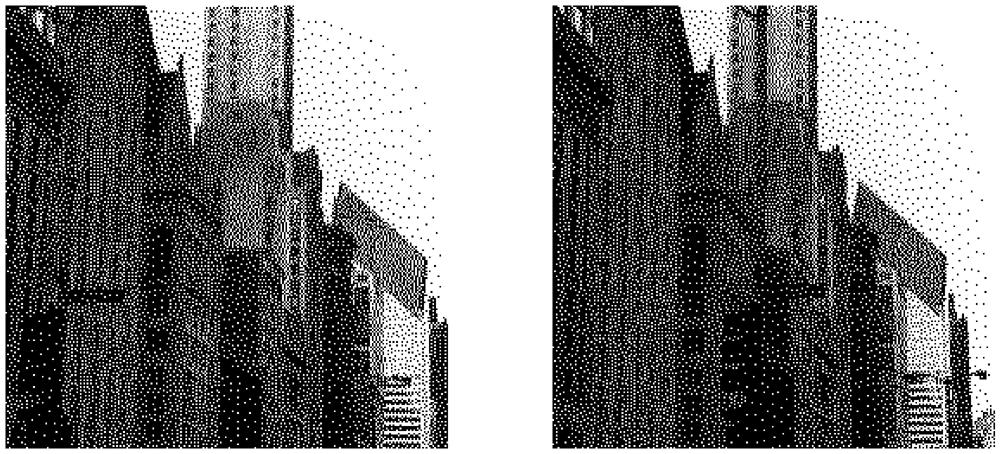
(o) (p)

Figure 6.2. (o) 3-D edges originally, (p) 3-D edges after tests.

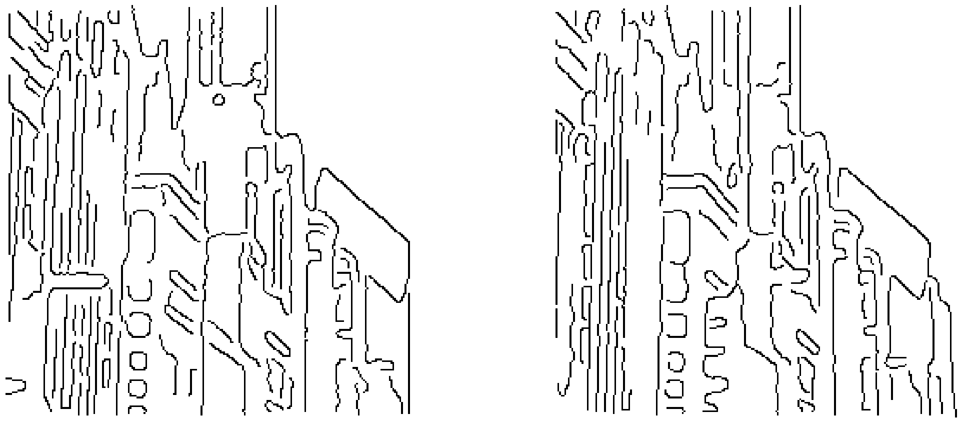
(q) (r)

Figure 6.2. (q) Original disparity, (r) Disparity after tests.



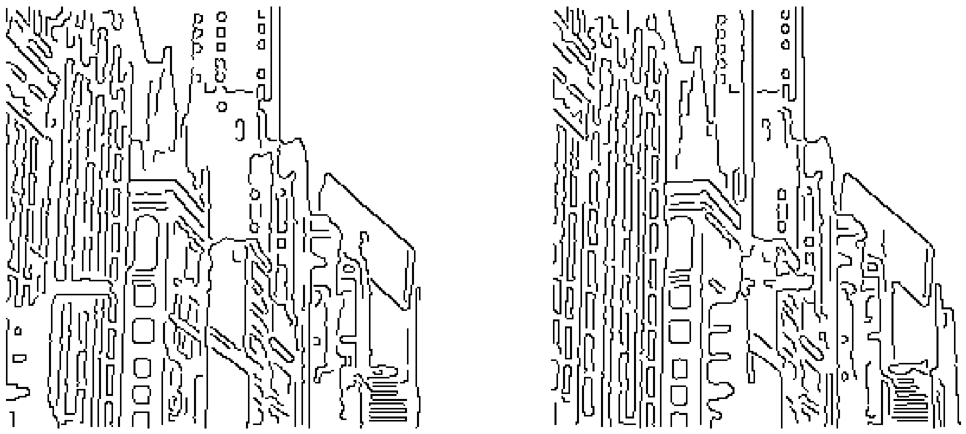
(a) (b)

Figure 6.3. (a) Right image, (b) Left image.



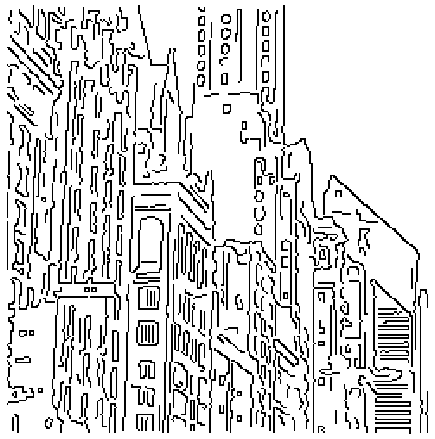
(c) (d)

Figure 6.3. (c) Original right channel 1, (d) Original left channel 1.

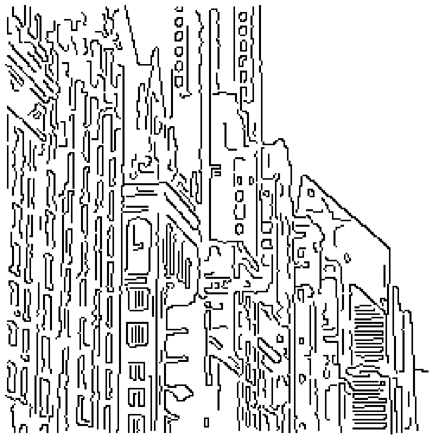


(e) (f)

Figure 6.3. (e) Original right channel 2, (f) Original left channel 2.

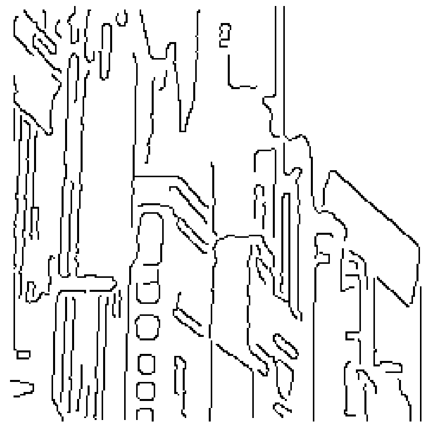


(g)

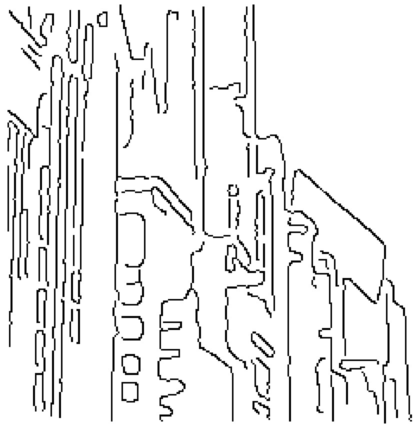


(h)

Figure 6.3. (g) Original right channel 3, (h) Original left channel 3.

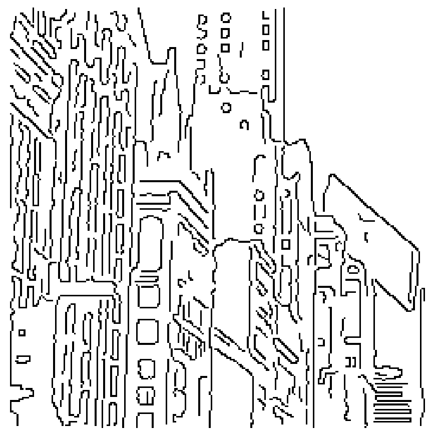


(i)

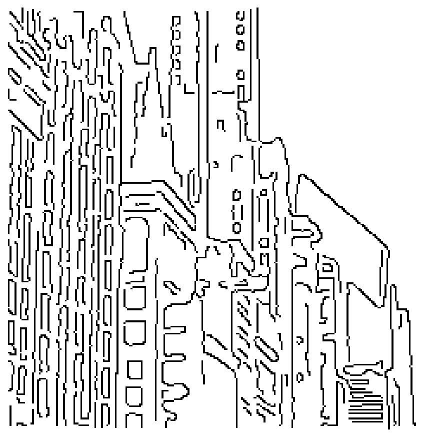


(j)

Figure 6.3. (i) New right channel 1, (j) New left channel 1.

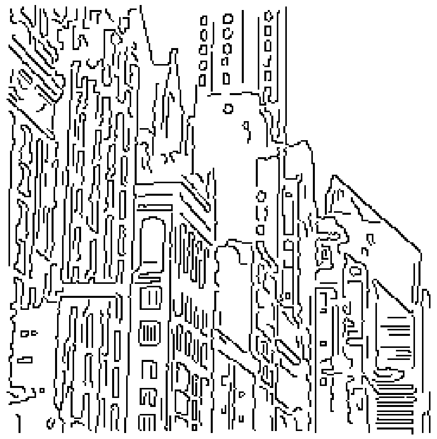
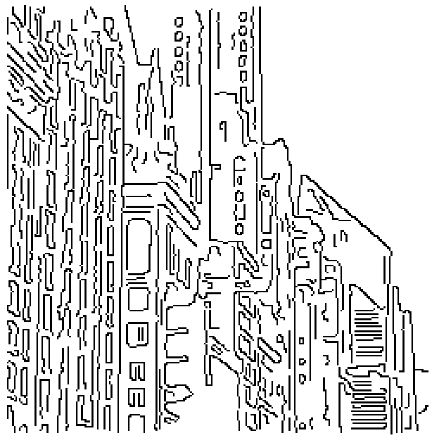


(k)



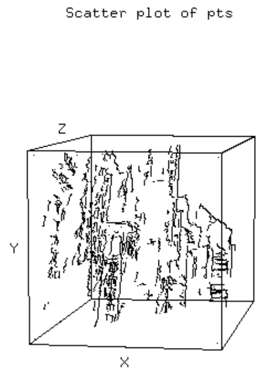
(l)

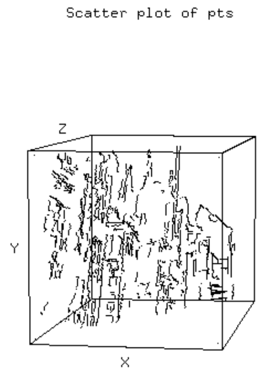
Figure 6.3. (k) New right channel 2, (l) New left channel 2.

(m) (n)

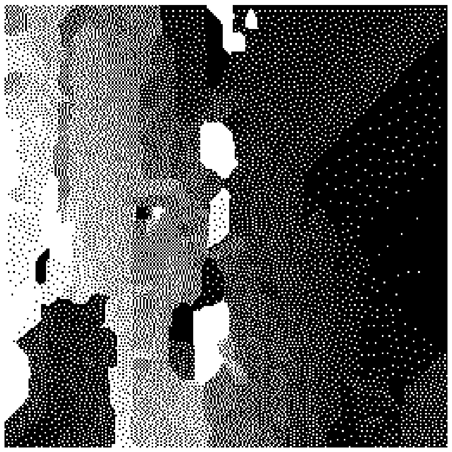
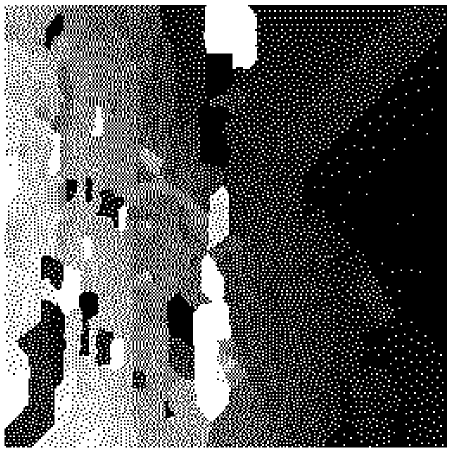
Figure 6.3. (m) New right channel 3, (n) New left channel 3.





(o) (p)

Figure 6.3. (o) 3-D edges originally, (p) 3-D edges after tests.

(q) (r)

Figure 6.3. (q) Original disparity, (r) Disparity after tests.

CHAPTER 7

POSSIBLE FUTURE WORK

The area of automated parameter tuning is open to a variety of future work. Not only can progress be made on the stereo vision front, but also on the statistical theory and system implementation fronts as well. In the vision area, possible future work includes extending the test images to span a variety of image domains and lighting conditions. This would further test the adaptability of the system. A great deal of work could also be performed in refining the candidate generation rules. Although these rules work fairly well currently, there is still much that can be done to reduce greatly unnecessary search of the parameter space. This includes interpreting much of the intermediate data from the token-matching process. Statistics such as the number of tokens found, how many were unmatched due to no similarity, and how many were unmatched due to ambiguity could be very useful for refining the candidate parameters. Other improvements on the stereo side include extending the number of parameters tuned by the system. Example parameters would include the similarity thresholds for the edgel orientation and magnitude used for the matching process.

There is also a wide variety of open problems left in the statistical guidance strategy area. The biggest open problem lies in the analysis of multistage strategies such as round-robin/greedy and round-robin/minimum-risk. A succinct analysis for these strategies could provide the optimal guidance parameters for a given domain. This could possibly result in a great time savings. Another open area includes the analysis of strategies having three or more stages, and other dynamic strategies. As these have not been fully analyzed, it is still not certain what type of gains may be possible.

To make the system more effective on real-world problems, it would be beneficial to modify the analysis to cope with situations in which individual tests take a variable amount of time. This more closely models the real world, but presents new difficulties of its own. Finally, it would also be of great use to extend the analysis to the realm of parallel computing. New difficulties encountered here would include accounting for communication overhead and potentially incomplete information for both the guidance strategy and candidate generation.

CHAPTER 8

CONCLUSIONS

This thesis presented a generate-and-test system for tuning the parameters of a general binocular stereo vision system. This was accomplished by systematically generating new parameters by analyzing the results of previous tests and by performing limited and controlled tests on the candidates. The generate-and-test framework has been modeled as a statistical selection problem operating under a given time constraint. A multistage framework for selecting the candidate with the best population has been developed, evaluated, and tested. Methods for selecting appropriate parameters for these multistage selection strategies were also presented.

The complete system is able to propose new parameter sets that in some cases are better than the ones originally found by extensive hand-tuning and commonly used heuristics. The experiments show that different parameter values may be required under different objectives and performance constraints. This system provides an automated tool for generating new parameters that can be part of an adaptive stereo vision system.

As vision applications become more in demand, it will become increasingly difficult to construct and install individual systems for the myriad of specific domains. With the automated system proposed here, however, most of the burden lies on the machine. The system has the capability to adapt to changing algorithm specifications as well as changing goals and target domains.

[1] N. Ayache and O. D. Faugeras, *Building, Registrating, and Fusing Noisy Visual Maps*, pp. 73-82, IEEE, 1987.

[2] R. E. Bechhofer, C. W. Dunnett, and M. Sobel, ‘‘A Two-sample Multiple Decision Procedure for Ranking Means of Normal Populations with a Common Unknown Variance,’’ *Biometrika*, vol. 41, pp. 170-176, 1954.

[3] R. E. Bechhofer, ‘‘A Single-Sample Multiple Decision Procedure for Ranking Means of Normal Populations with Known Variances,’’ *Ann. Math. Statist.*, vol. 25, no. 1, pp. 16-39, Institute of Mathematical Statistics, Ann Arbor, MI, March 1954.

[4] J. O. Berger and J. Deely, ‘‘A Bayesian Approach to Ranking and Selection of Related Means With Alternatives to Analysis-of-Variance Methodology,’’ *J. of the American Statistical Association*, vol. 83, no. 402, pp. 364-373, American Statistical Association, June 1988.

[5] J. Canny, ‘‘A Computational Approach to Edge Detection,’’ *Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679-698, IEEE, Nov. 1986.

[6] J. L. Devore, *Probability and Statistics for Engineering and the Sciences*, Brooks/Cole Pub. Co., Monterey, CA, 1982.

[7] S. E. Dreyfus and A. M. Law, in *The Art and Theory of Dynamic Programming*, Academic Press, New York, NY, 1977.

[8] E. J. Dudewicz and J. O. Koo, *The Complete Categorized Guide to Statistical Selection and Ranking Procedures*, American Sciences Press, Inc., Columbus, OH, 1982.

[9] D. E. Goldberg, ‘‘Probability Matching, the Magnitude of Reinforcement, and Classifier System Bidding,’’ *Machine Learning*, vol. 5, pp. 407-425, Kluwer Academic Pub., Boston, MA, 1990.

[10] W. E. L. Grimson, in *From Images to Surfaces: A Computational Study of the Human Early Visual System*, MIT Press, Cambridge, MA, 1981.

[11] S. S. Gupta and D. Y. Huang, ‘‘Subset Selection Procedures for the Means and Variances of Normal Populations: Unequal Sample Sizes Case,’’ *Sankhya*, vol. 38B, no. 2, pp. 112-128, 1976.

[12] S. S. Gupta and S. Panchapakesan, *Multiple Decision Procedures Theory and Methodology of Selecting and Ranking Populations*, John Wiley & Sons, Inc., New York, NY, 1979.

[13] S. S. Gupta and H. Yang, ‘‘Bayes-P* Subset Selection Procedures for the Best Population,’’ *J. of Statistical Planning and Inference*, vol. 12, pp. 213-233, Elsevier Science Pub., 1985.

[14] S. S. Gupta and K. J. Miescke, ‘‘On the Problem of Finding the Largest Normal Mean Under Heteroscedasticity,’’ in *Statistical Decision Theory and Related Topics IV*, ed. J. O. Berger, vol. 2, pp. 37-49, Springer-Verlag, New York, NY, 1988.

[15] S. S. Gupta and S. Panchapakesan, ‘‘Sequential Ranking and Selection Procedures,’’ in *Handbook of Sequential Analysis*, ed. P. K. Sen, pp. 363-380, Dekker, NY, 1991.

[16] R. V. Hogg and E. A. Tanis, *Probability and Statistical Inference*, 3rd Edition, Macmillan Pub. Co./Collier Macmillan Pub., New York, NY/London, England, 1988.

[17] B. K. P. Horn, in *Robot Vision*, MIT Press, Cambridge, MA, 1986.

[18] A. Ieumwananonthachai, A. Aizawa, S. R. Schwartz, B. W. Wah, and J. C. Yan, ‘‘Intelligent Process Mapping Through Systematic Improvement of Heuristics,’’ *J. of Parallel and Distributed Computing*, vol. 15, pp. 118-142, Academic Press, June 1992.

[19] H. Kriplani, ‘‘Resource Constrained Design of Artificial Neural Networks,’’ *M.S. Thesis, Dept. of Electrical and Computer Engineering*, pp. 1-148, Univ. of Illinois, Urbana, IL, May 1990.

[20] M. B. Lowrie and B. W. Wah, ‘‘Learning Heuristic Functions for Numeric Optimization Problems,’’ *Proc. Computer Software and Applications Conf.*, pp. 443-450, IEEE, Chicago, IL, Oct. 1988.

[21] D. Marr, in *Vision*, W. H. Freeman and Co., New York, NY, 1982.

[22] J. E. W. Mayhew and J. P. Frisby, ‘‘Psychophysical and Computational Studies toward a Theory of Human Stereopsis,’’ *Artificial Intelligence*, vol. 17, no. 1-3, pp. 349-385, North-Holland, Amsterdam, Aug. 1981.

[23] R. S. Michalski, ‘‘Understanding the Nature of Learning: Issues and Research Directions,’’ in *Machine Learning: An Artificial Intelligence Approach*, ed. R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, vol. II, pp. 3-25, Morgan Kaufmann, Los Altos, CA, 1986.

[24] J. Pearl, ‘‘On the Discovery and Generation of Certain Heuristics,’’ *The AI Magazine*, pp. 23-33, AAAI, Winter/Spring 1983.

[25] C. P. Shapiro, ‘‘Sequential Allocation and Optional Stopping in Bayesian Simultaneous Estimation,’’ *J. of the American Statistical Association*, vol. 78, no. 382, pp. 396-402, American Statistical Association, June 1983.

[26] M. Sitek, ‘‘Application of the Selection Procedure R to Unequal Observation Numbers,’’ *Zastosowania Matematyki Applicationes Mathematicae*, vol. 12, no. 4, pp. 355-363, 1972.

[27] J. Sklansky, ‘‘Bottlenecks to Effective Application of Machine Vision,’’ in *Machine Vision: Algorithms, Architectures, and Systems*, ed. H. Freeman, pp. 187-192, Academic Press, Inc., San Diego, CA, 1988.

[28] C. V. Stewart and C. R. Dyer, *The Trinocular General Support Algorithm: A Three-Camera Stereo Algorithm for Overcoming Binocular Matching Errors*, pp. 134-138, IEEE, 1988.

[29] H. Takahashi and F. Tomita, *Self-Calibration of Stereo Cameras*, pp. 123-128, IEEE, 1988.

[30] S. Tanaka and A. C. Kak, ‘‘A Rule-Based Approach to Binocular Stereopsis,’’ TR-EE 88-33, Purdue Univ., West Lafayette, IN, July 1988.

[31] B. W. Wah and H. Kriplani, ‘‘Resource Constrained Design of Artificial Neural Networks,’’ *Proc. Int’l Joint Conf. on Neural Networks*, vol. III, pp. 269-279, IEEE, June 1990.

[32] J. Weng, N. Ahuja, and T. S. Huang, *Two-View Matching*, pp. 64-73, IEEE, 1988.

[33] J. C. Yan, *Post-Game Analysis--A Heuristic Resource Management Framework for Concurrent Systems*, Ph.D. Dissertation, Dept. Elec. Eng., Stanford Univ., Dec. 1988.

[34] J. C. Yan and S. F. Lundstrom, ‘‘The Post-Game Analysis Framework--Developing Resource Management Strategies for Concurrent Systems,’’ *Trans. on Knowledge and Data Engineering*, vol. 1, no. 3, pp. 293-309, IEEE, Sept. 1989.

[35] C. F. Yu and B. W. Wah, ‘‘Learning Dominance Relations in Combinatorial Search Problems,’’ *Trans. on Software Engineering*, vol. SE-14, no. 8, pp. 1155-1175, IEEE, Aug. 1988.