AUTOMATED PARAMETER TUNING IN STEREO VISION UNDER TIME CONSTRAINTS

Steven R. Schwartz Motorola, Inc.

Benjamin W. Wah University of Illinois

ABSTRACT

This paper presents a method for tuning parameters under a fixed time constraint for a general binocular stereo-vision algorithm. A major difficulty in stereo vision, as well as in other vision algorithms, lies in adjusting the large variety of parameters for maximizing performance. This effort is usually performed by human experts with a minimum of formal guidelines. To automate this process, we develop TEACHER 4.2, a generate-and-test system that systematically generates new parameter values by analyzing the results of previous tests, and that performs limited and controlled tests on the candidates generated using high-speed computers. The system is modeled as a statistical selection problem operating under a given time constraint. It divides the time allowed into stages, where promising parameter-value sets found in one stage are passed to the next stage for further testing, and selects the parameter-value set deemed best by the final stage as the result. We show experimentally that our system can find new parameter-value sets which in some cases are better than the ones originally found by extensive hand-tuning and commonly used heuristics. Our experiments further show that different parameter values may be required under different objectives and performance constraints. Our system provides an automated tool for generating new parameters that can be part of an adaptive stereo-vision system, capable of adapting to changing algorithm specifications as well as changing goals and target domains.

KEYWORDS AND PHRASES. Depth perception, generate-and-test, time constraint, statistical allocation, stereo vision.

1. INTRODUCTION

Stereo vision entails the extraction of 3-D knowledge of a scene from the corresponding 2-D projections. Algorithms for stereo vision generally requires the tuning of a large number of parameters, serving purposes such as detection of thresholds and the configuration of image preprocessing stages. Traditionally, these parameters are tuned by designers using specific training images: the experts propose new parameter-value sets to be tested on training images until satisfactory performance is obtained. This process can be automated by designing an intelligent parameter generator and by relying on high-speed computers to test the parameter-value sets proposed on selected training images.

Steven R. Schwartz is with Motorola, Inc., Mail Drop IL 27 G79, 1501 W. Shure Drive, Arlington Heights, IL 60004 (schwartz@marble.rtsg.mot.com). Benjamin W. Wah is with the University of Illinois, Center for Reliable and High Performance Computing, Coordinated Science Laboratory, 1308 West Main Street, MC255, Urbana, IL 61801 (b-wah@uiuc.edu).

This research was supported partially by National Aeronautics and Space Administration Contract NCC 2-481, National Science Foundation Grant MIP 88-10584, and Sumitomo Electric Industries, Yokohama, Japan.

Proceedings of 4th International Conference on Tools with Artificial Intelligence, November 10-13, 1992.

This paper presents an intelligent parameter-tuning framework and analyzes a statistical method for systematically exploring the parameter space. The objective is to find the stereo-vision parameters that maximize the average performance over a database of test images. There are three main reasons for this approach. 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. 3) Last, there is little knowledge available for creating new parameter-value sets. As it is difficult to evaluate all possible parameter-value sets, our method proposes a limited set of parameter values and selectively evaluates them by a given deadline. As the process is incremental, it allows the results from previous tests to be used as the starting point for further parameter tuning.

The central aspect of our approach is a *statistical* method for trading between the number of new parameter-value sets to generate, and the number of tests to perform on the existing ones. Our approach consists of some initial empirical tests for determining the scheduling of the time allowed, the division of the total time into stages, and the assignment of the testing strategy in each stage. The testing strategy is used to determine the parameter-value set to test next based on results of previous tests. A rule base creates new parameter values to be tested, using knowledge on past test results to determine how specific parameter values should be modified.

The results presented in this paper extends our previous work on population-based learning of heuristics using a learning-by-example paradigm [9]. TEACHER (which stands for TEchniques for Automated Creation of HEuRistics) is a system we developed for population-based learning, and has been applied for finding new heuristics for process mapping, load balancing, and combinatorial search.

2. BACKGROUND

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 [4]. 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 or image understanding.

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 for use in matching. (This information is known as the primal sketch.) [4] 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.

Using the general algorithm as outlined above, the steps most suited for computer tuning are (2) and (3). Accordingly, this paper focuses on this area.

Under the above general view of the stereo matching procedure, the computational goals fall into three conflicting areas: 1) speed of matching, 2) density of matches, and 3) accuracy of matches. Note that in most vision applications the objective is *ill-defined*; that is, the user may not know the precise performance trade-off that is necessary, but rather that certain requirements must be met.

Related work that attempts to address some of the above problems falls under the category of adaptive stereo vision. Two prominent approaches in this area 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 [10]. 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. First, the tunable parameters are merely heuristic parameters abstracted by one level and must be set through experimentation. Second, it is necessary to gather statistics during run-time in order to use meta-parameters, and the statistical distribution is assumed constant or is changing slowly.

Another example of metaparameters is seen in Tanaka and Kak's work on rule-based stereo matching [8]. Here, the control parameters are embedded in the system in the form of rules. The matching strategy dynamically shifts among four different approaches as necessary. Although the combination of these methods under central control improves robustness, there is now the overhead of coordinating them. The basic problems of meta-parameters are again present here, namely, run-time decision overhead and the need for tuning the metaparameters.

Another area of adaptive stereo vision is in the iterative refinement of the parameters. An example is the work of Takahashi and Tomita [7]. 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 periodically, it still requires run-time computation. Additionally, it depends on the existence of an analytically correct solution. This is appropriate in the case of finding the camera parameters, but oftentimes the search methods for these parameters will be ill-defined. In this case, intelligent or expert experimentation is the best available tool.

The implementation of the above approaches raises a variety of questions. First, we need to determine the heuristic components of the vision algorithm. Questions that we need to answer include the following. How much detail should be extracted for each token in a 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 may involve explicit assumptions about the characteristics of the tokens and the specific implementation.

Second, we need to determine the parameter values used in the vision algorithm. Table 1 lists some typical parameters that must be tuned in a general token-based stereo matching algorithm. The last column shows sample values considered in the specific algorithm implemented in this paper (to be described in Section 5.1). It is obvious that it is impossible to test all combinations of parameter values, as they have a continuous range.

Third, the application domain is knowledge-lean and has little information available for guiding the search of the appropriate parameters. Hence, it is difficult to design an expert system for guiding the testing of alternative parameter-value sets.

Table 1. Stereo Algorithm Parameter List

Parameter	Range
Number of Channels — aids matching density by refining the search region and by obtaining increasingly dense depth estimates.	≥1
Blurring Kernel Size (in pixels) — 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.	0.1–5.0
High Threshold (in ∂ *intensity*/∂ *pixel*) — upper limit of gradient strength for hysteresis of edge thresholding.	1 – 255
Low Threshold (in ∂ *intensity*/∂ *pixel*) — lower limit on gradient strength allowed for edge classification.	0 – 255
Initial Search Window (in pixels) — size of region to initially consider for a match.	1 – 200
Similarity Thresholds — used to determine if tokens are a match	
Gradient (in $\Delta\partial$ *intensity*/∂ *pixel*) — difference in gradient magnitude	1 – 50
Orientation (in degrees) — difference in edgel orientation	$\lvert\bullet\rvert \le 30$
Vertical Position (in pixels) — difference in vertical position	$\pm$ 1 – 3

To address the issues raised above, our focus in this paper is on testing alternative parameter-value sets in a controlled manner, without relying on specific domain knowledge of the application. 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 the available testing time based on a statistical model. It is important to note that, as the available time is finite, it is necessary to schedule tests appropriately. Details of this approach are presented in the following sections.

3. GENERATE-AND-TEST METHOD

In this study, we assume that the *objective function* of the vision application is specified by the users, and is expressed as a heuristic parametric function of the time taken to match the tokens in the scene and the quantity and accuracy of the matched tokens. By varying the objective function and by finding parameter values that optimize the objective function, we attempt to find a good combination that results in the ''best'' performance.

The parameter-tuning problem we study is unique because the parameter values are continuous and unbounded. Hence, it is impossible to test the performance for all possible combinations of parameter values. In tuning the performance of the vision algorithm, we propose a generate-and-test method that searches the domain of parameter values and finds parameter-value sets that improves the objective function. Such a method has been found to be useful for discovering new heuristics in knowledge-lean application domains [9]. To make the search more efficient, we can use expert knowledge for generating new parameters and high-speed computers for evaluating the quality of the proposed parameters against test images from the target domain. As the domain searched is unbounded, we can only measure relative improvement of the parameter-value sets found but cannot pinpoint the optimal set.

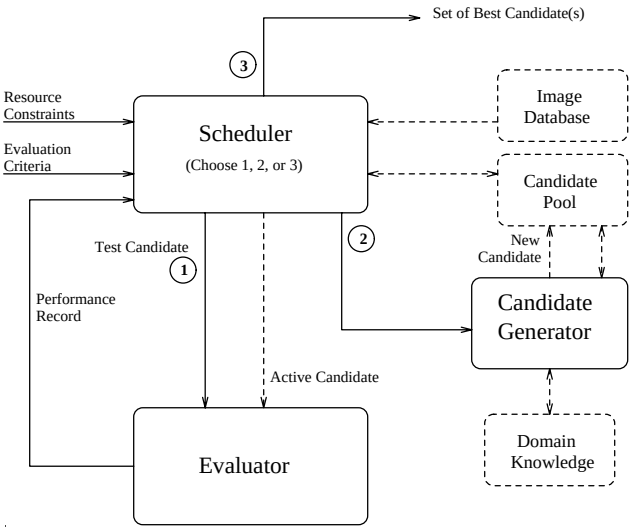


Figure 1. Generate-and-test framework

We apply a generate-and-test framework for searching the ill-defined space under a given time constraint (see Figure 1). As mentioned earlier, a candidate refers to a particular parameter-value 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*.

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. Our focus in this paper is placed on the resource scheduler.

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, with a goal of finding the candidate with the greatest average performance. Of course, the ideal situation 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 set of test images representing the application domain. This would imply that all candidates (including the worst) must be evaluated to an equal degree.

At the end of each quantum, the scheduler selects one of the following actions to perform (see Figure 1). 1) Select the next candidate to test from the candidate pool. 2) Generate a new candidate to be placed in 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 deciding when to generate new candidates is to generate them 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*, consisting of 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 discussed in Section 4.

If the decision is made to pursue choice (2), then a new candidate must be generated. The generator should be intelligent in creating new, and ideally better, candidates. To this end, it should use the past performance of existing candidates as well as any available domain knowledge.

4. STATISTICAL GUIDANCE STRATEGIES

In the generate-and-test framework described in Section 3, we are 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 the images from the database. A statistical formulation of this problem can be expressed as follows: *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*.[1] In this case, the performance values of a parameter-value set are associated with applying the candidate to the given test images.[2] 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[3] within a fixed and known number of tests.

The goal of selecting the best candidate can be reduced to designing a *guidance strategy* that tells which candidate to test next, so that the likelihood of selecting the best candidate is maximized. Of course, the ideal goal is to find a guidance strategy that always finds populations with 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 only on this information.

Existing guidance strategies can be classified as *static* or *dynamic*. Static guidance strategies have a selection sequence that is determined ahead of time. One simple strategy of this type is the *round-robin* strategy which takes samples from each population in turn. Its drawback is that it tests the worst candidate as much as the best, ignoring how initial tests may quickly demonstrate the disparity in performance between the two. In contrast, dynamic guidance strategies select the candidate for

1. 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.

2. A test image is randomly drawn from the image database.

3. The highest mean is used as the objective here. Other objectives, such as the maximum of a population, may be used instead.

testing based on known sample values. An example of this is the *greedy* method which samples the population that currently has the maximum sample mean. The problem with dynamic strategies is that they might be misled by dynamic information obtained early in the search process, and discard the best candidate at an early stage.

4.1. Statistical Selection Strategies

The problem of finding the best candidate by performing a series of tests is known traditionally as the *selection problem* in statistical inference. These problems can be classified into two types. The first type was first studied by Stein in 1948 [6] and is called the *stopping problem*. It deals with finding the minimum number of tests in order to know which population among a finite number of populations is the best to within a certain degree of confidence. This result and subsequent extensions cannot be applied in our case because they considered a finite number of populations and an unbounded amount of time.

The second type was pioneered by Bechhofer in 1954 [1] and is called the *allocation problem*. It focuses on allocating a fixed number of tests among a given finite number of populations so that the probability of selecting the population with the maximum mean is maximized. Recent work in this area are predominantly on multistage methods in which the number of tests allowed are divided into stages, and promising populations selected in one stage are passed to the next stage for further testing. The allocation of tests can be static, which is determined a priori based on statistical distributions of populations, or can be dynamic, which is based on means and variances estimated during testing.

Results in the second type do not apply in our case because they deal with a finite and given number of populations before selection begins; our objective in this paper, on the other hand, is to find a population with a large mean value but not necessarily the one with the largest mean, as there are infinitely many populations to start. However, existing statistical algorithms can be applied if the time allowed were divided into two parts: the first part is used for determining a finite number of populations to be tested in the second part, and the second parts assigns tests according to algorithms developed in statistical allocation.

In the next section, we present a multistage testing procedure that uses the first stage to determine a finite number of populations to be tested in the remaining stages. This procedure has been applied successfully in TEACHER 4.1 for finding better heuristics for mapping a set of communicating processes on a distributed-memory multicomputer system [3].

4.2. Multistage Testing

A good guidance strategy 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 at the end.) We formulate a general guidance strategy, $G(T)$, composed of a series of stages. 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 characterized by a 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 2) can accommodate both static and dynamic guidance strategies. Under this method, the early stages correspond to coarse initial testing used to weed out unworthy candidates, and the latter stages correspond to more careful scrutiny of the better candidates. 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. Also, only the top candidate is selected at the end of the last stage.

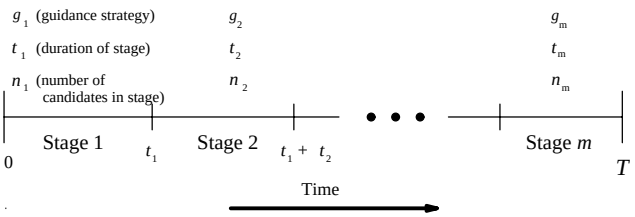


Figure 2. The multistage testing procedure

The goal now is to find the strategy parameters comprising the best guidance strategy. The parameter values must be selected 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 testing time, 4) the number of possible populations, and 5) the distribution of the population means of the populations as a whole.

In our previous work, we had analyzed a two-stage testing procedure for finding new heuristic parameter-value sets for mapping a set of communicating processes on a distributed-memory computer system [3]. In analyzing the testing procedure there, we made three general assumptions. 1) Each population was assumed to be normally distributed, and that sample values in a population were independent and identically distributed (i.i.d.). 2) The mean of each population was assumed to be drawn from a known distribution f_μ, and that the values of all population means were i.i.d. Each population was also assumed to have an infinite number of samples. 3) The standard deviation of each population was assumed to be drawn from a known distribution f_σ, and that the values of all population standard deviations were i.i.d. We further assumed that the mean and standard deviation of each population were independent.

The same assumptions are made in the stereo-vision problem studied in this paper. To justify these assumptions, we verified them using the empirical cumulative distribution function for the population means and the population variances of 85 candidate parameter-value sets of the vision problem. In all these tests, a normal distribution was found to have a good fit.

We have studied and evaluated four testing procedures [3]:

- single-stage round-robin algorithm, where a predetermined number of populations are sampled a constant number of times in the time[4] allowed;
- two-stage round-robin algorithm, where analysis aims at finding the division of time between the first and the second stage and the parameters of the round-robin algorithm in each stage;
- two-stage round-robin/greedy algorithm, which uses (as heuristics) the same division of time found analytically for the two-stage round-robin algorithm; and
- two-stage round-robin/minimum-risk algorithm, which is similar to the last algorithm except that it selects populations for testing in the second stage based on a risk function (defined as the expected squared-error loss of the estimated mean of each population).

For the last two testing procedures, performance analysis can be formulated but is too difficult to be solved in closed forms.

4. The time constraint is expressed as the total number of samples drawn, assuming that each draw takes unit time.

We have applied the four testing procedures described above for the stereo-vision application. To estimate the statistical parameters of the vision application, we assume that the initial 10% of the total time allowed is assigned to presample the populations. During this period, the system evaluates as many parameter-value sets as possible and collect performance data in terms of sample means and sample standard deviations. Each parameter-value set is evaluated on four different test images to assure a reasonable confidence in its statistical performance values. The parameters of the selection process are then determined based on our previous analysis [3].

5. IMPLEMENTATION DETAILS

The stereo-vision 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 by partitioning them into smaller ones so that their processing times are less than one quantum. 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 [2]. This algorithm employs hysteresis for greater edge continuity and retains mid-processing 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 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 is discussed in Section 2. For this, the necessary parameters include the edge-detection parameters for each channel as well as the number of channels.

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 1 shows the value ranges for the parameters. 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 manually, and held constant throughout the experimentation.

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

5.1. 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 desired form of the objective function should only be a function of the stereo-vision algorithm parameters and should 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. Other 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*, on the other hand, is measured by probing the resulting disparity map and by comparing the values with hand-calculated disparities. For the test images in our database, ten points were measured per image. The accuracy used in 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 as a function t, the time for the test, t_0, the time limit, and q, a parameter of the penalty function, is shown as follows.

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

The final objective function is the average quality of the candidate measured over each test image in the database. The quality measure over one image, i, with parameter-value 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 $d(i, P)$, the density of disparity measurements, $a(i, P)$, the average accuracy of disparity measurements, and $t(i, P)$, the time to process the image, where

$$Q(i, P) = d(i, P) \times a(i, P) \times p(t(i, P)). \tag{2}$$

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

$$O(P, B) = Q_{avg} = \frac{1}{N} \sum_{i \in B} Q(i, P). \tag{3}$$

The objective function stated here is really one in a family of possible objectives. By varying components such as t_0, q, and $e_{\max}$, users can make the system search for particular parameter-value sets for different applications. This kind of information is of interest because it shows how the system focuses towards different classes of candidates under different system requirements. Further, users may not be very clear in the beginning on the quality of results they wish to get. By finding new parameter-value sets based on different objectives, users can determine subjectively the best parameter-value set when testing is completed.

5.2. 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. Accuracy points were measured by finding correspondences manually on magnified copies of the images. Accuracy measurements made this way are precise to within one pixel.

5.3. Candidate Generation

Initially, some sample parameter-value 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 this area. Due to the limited time available for testing, the candidate generator should refer to the performance of previous candidates when generating new ones. Currently, two methods have been integrated: 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. This is employed as a hedge against stagnation in a local maximum on the objective-function surface. 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. This allows existing good transformations to be followed. The direction is expressed as a vector of the delta values between the parameters of the two candidates. Details of the rules used are described in the reference [5].

6. EXPERIMENTAL RESULTS

This section presents a verification of the effectiveness of the methods described in the previous sections. 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 seven candidates that were generated manually (see Table 2). All experiments started with these candidates to show the different types of candidates that could be discovered as the available time and the objective function were modified. From these initial candidates, the rules in the candidate generator were applied to generate new candidates as necessary. At the end of each run, all of the candidates were 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. This fraction was determined heuristically from experiments using a simulator on a variety of problems, and appears to be fairly robust. Presampling was performed for 10% of the available time to determine the mean and standard deviation of the population means, and the standard deviation of each population (assumed constant for all populations). These parameters were then used to determine the number of candidates to be tested in each of the two stages. 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.

Table 2. Hand-tuned Seed Candidates (numbers in each row indicate parameters used for the channel).

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

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 on a Sun 4 IPC workstation. 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 simulation results are close to the observed ones. 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 candidates are not generated during testing but rather selected from the existing pool of candidates already generated beforehand. 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 in Table 3 for real tests (those with candidate generation) and the results in Table 4 (those with 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 3. Each entry of this table lists the identity of the best parameter-value set that was found, as well as its performance, *i.e.*, the value of the objective function after 30 tests. The parameter-value set for each channel is encoded in the form {channel width}-{low threshold}-{high threshold} with separate channels separated by a "/". The number in parentheses at the end of the candidate name indicates that this was the n^{th} candidate generated to be tested.

Simulation results for the case of a fixed duration and different objectives (modeled using different e_{max}'s and t_0's) are shown in Table 4. 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 duration for these tests was 400 quanta. Due to the limited space, other simulation results are not presented [5].

Table 3. Performance from Actual Tests

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

Table 4. Performance from Simulations with Various Objectives

t_0	Error Cut-off (e_{max})		
	2.5	**5**	**10**
5	2.4-2.0-5.0/ 1.7-2.0-5.0/ 1.0-2.0-5.0 (22) 0.0163	1.5-2.3-2.7 (24) 0.0225	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	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) 0.0408
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	2.7-2.7-7.2/ 1.7-2.0-5.0/ 1.0-2.0-3.7 (85) 0.0424
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	2.8-2.0-5.0/ 1.3-3.5-4.6/ 0.6-2.5-6.1 (39) 0.0456

As seen in Table 4, the objective function is important in determining the solution found. The difference in the results for various t_0 and e_{max} demonstrates the purposes of the objective function: the ability to express the designer's goals and to guide the search in an automated fashion. As expected, spending more time generally produces better solutions.

As stated in Section 5, it was assumed that the populations were normally distributed. Although this cannot be guaranteed for all possible candidates, the performance values of typical candidates fit the normal assumption well. To verify this fact empirically, one hundred different candidate parameter-value 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. Normality for a total of 327 candidates, each with 30 points, t_0 = 30 seconds, and e_{max} = 2.5, were then verified at 95% confidence level (not shown due to space limitation).

It was also assumed in Section 5 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.

Figure 3 shows a coke-can scene obtained from NASA, Ames Research Center. It is shown here to demonstrate sample performance on a relatively simple scene. The set of images consists of the stereo pair, the left and right edge-maps found on the original final channel, and those found on the discovered final channel after 200 tests with t_0 = 30, and e_{max} = 5 (which define the objective function). The original parameter-value setting processed the scene in 62.23 seconds, using candidate number 5 in Table 2, whereas the discovered parameters took only 51.64 seconds, using the parameter-value set 2.2-2.0-7.0/1.6-3.8-3.8/1.3-3.0-3.6 (41). In addition, it is noticeable from Figures 3e and 3f that the edges matched are slightly improved: the string in front of the coke can was better identified by the new parameter-value set learned.

A number of other stereo images have been tested; they are not shown due to space limitation. Our experiments generally found parameter-value sets that improve both the accuracy of the tokens matched as well as the computational time for processing the images. In all cases, the computer-tuned parameters performed faster and resulted in a 15% to 30% improvement as measured by the objective function.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

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

[5] S. R. Schwartz, *Resource Constrained Parameter Tuning Applied to Stereo Vision*, M.Sc. Thesis, Dept. of Electrical and Computer Engineering, Univ. of Illinois, Urbana, IL, Aug. 1991.

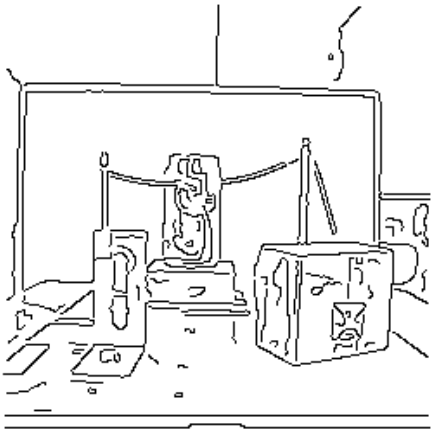
[6] C. Stein, ''The Selection of the Largest of a Number of Means, Abstract,'' *Ann. Math. Statist.*, vol. 19, p. 429, 1948.

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

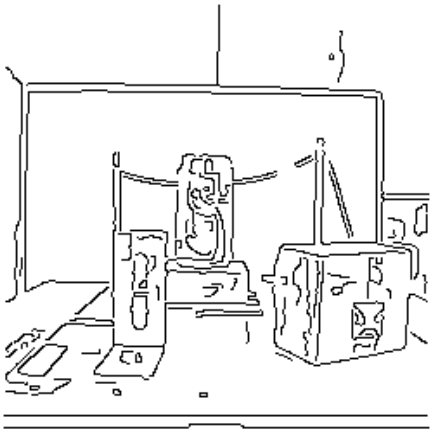
[8] S. Tanaka and A. C. Kak, ''A Rule-Based Approach to Binocular Stereopsis,'' TR-EE 88-33, Purdue Univ., West Lafayette, IN, July 1988.

[9] B. W. Wah, ''Population-Based Learning: A New Method for Learning from Examples under Resource Constraints,'' *Trans. on Knowledge and Data Engineering*, vol. 4, no. 5, pp. 454-474, IEEE, Oct. 1992.

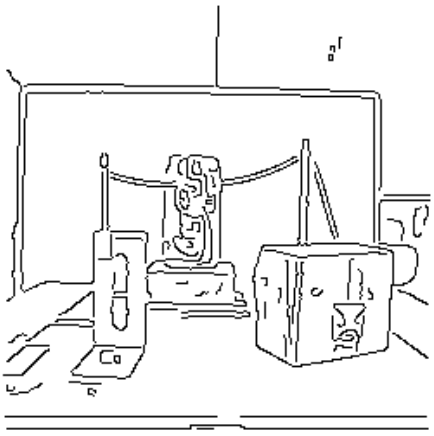
[10] J. Weng, N. Ahuja, and T. S. Huang, *Two-View Matching*, pp. 64-73, IEEE, 1988.



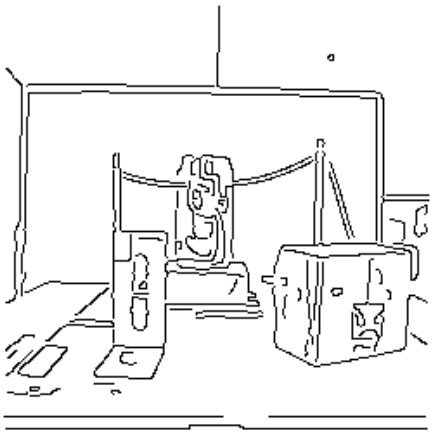
(a) Original right channel 3.



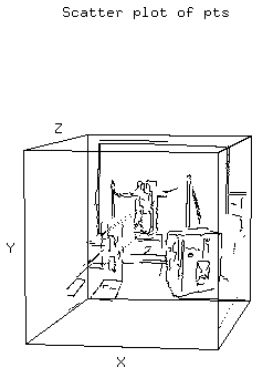
(b) Original left channel 3.



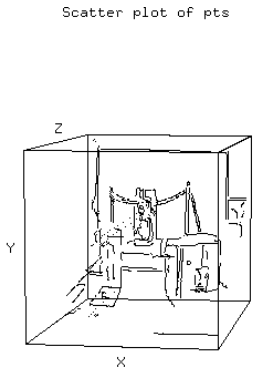
(c) New right channel 3.



(f) New left channel 3.



(e) 3-D edges originally.



(f) 3-D edges after tests.

Figure 3. Example for illustrating improvements found by the automated parameter tuning method. (The original pair of stereo images were processed using parameter-value set 5 defined in Table 2; the new pair of images were processed using the parameter-value set 2.2-2.0-7.0/1.6-3.8-3.8/1.3-3.0-3.6. The authors would like to thank Dr. B. Sridhar at NASA, Ames Research Center, for providing the original test images.)