INNOVATIVE APPROACH TO CONTROL OF AN EMULSION LOADER

J. TODD ENGRAM

Innovative Approach to Control of an Emulsion Loader

A thesis submitted to the
School of Graduate Studies in partial
fulfillment of the requirements for the degree of
Master of Engineering

Faculty of Engineering and Applied Science
Memorial University of Newfoundland

September 11, 2007

St. John's Newfoundland

Abstract

In this thesis, the work to develop the vision system for the Emulsion Loading Automation Project (ELAP) is presented. The ELAP project was a collaborative effort among various team members to develop an intelligent system to automate the complex task of an emulsion loader used in underground mining operations. The vision system was tasked with the challenging imaging requirements of locating and identifying drill-holes in the rock face of an underground mine, and to provide visual guidance for the hose guide to load the drill-holes with emulsion. The first part of this thesis outlines the development of image processing algorithms to segment and identify drill-holes present in an image acquired from a video stream. This involved applying thresholding and morphological techniques to preprocess the image to improve contrast, separate touching objects, and fill any holes. A pattern recognition model was then developed using drill-hole feature data to classify segmented objects as either drill-holes or background artifacts. The second part of this thesis presents the work performed for visual guidance to position the robotic boom in front of a drill-hole for loading. Using a camera mounted to the end of a robotic boom, camera and hand-eye calibration routines were developed to transform the drill-hole image object to both the camera and end-effector reference frames. A visual guidance algorithm was then developed using the calibration parameters to visual servo the robotic boom to a drill-hole for loading. Finally, testing performed in an underground mine after the critical subsystems were integrated and operational, verified the vision system operation as designed.

Acknowledgements

I would like to thank my supervisor, Dr. Raymond Gosine, for his encouragement and assistance throughout my studies.

I am very grateful to C-CORE for providing me the opportunity to pursue graduate studies and allowing me the time to do so.

I would like to thank Mr. Al Akerman for permitting me to use the research project work for my thesis.

I especially thank Mr. Baxter Smith who assisted me to develop the camera calibration algorithms, Mr. Jason Mills for integrating the algorithms into the software application, Mr. Stefan Tarrant for sharing his GA implementation, Mr. Carl Howell for his assistance with feature selection, and other members of the Intelligent Systems team at C-CORE for their suggestions and comments.

Finally, I would like to thank my lovely wife, Joanne, and my daughter, Ally, for their patience, understanding, and support.

Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations and Symbols

ELAP	Emulsion Loading Automation Project
VS	Vision System
ISE	International Submarine Engineering Ltd.
LPS	Loading Process Supervisor
RBC	Robotic Boom Controller
ROI	Region Of Interest; an area located on the image plane
JPEG	Joint Photographic Experts Group
FSS	Feature Subset Selection
GA	Genetic Algorithm
SFS	Sequential Forward Selection method
ES	Exhaustive Search technique
LOOCV	Leave-One-Out Cross-Validation
CCD	Charged Coupled Device; image sensor for a camera
RGB	Red-Green-Blue signal format from a camera
TCP/IP	Transmission Control Protocol / Internet Protocol
OpenCV	Open Source Computer Vision Library
MIL	Matrox Imaging Library
pose	Position and orientation of an object
MLC	Maximum Likelihood Classifier
PCI	Peripheral Components Interconnect; a computer data bus
DOF	Degrees Of Freedom
PBVS	Position-Based Visual Servoing
IBVS	Image-based Visual Servoing
$P_f(f)$	Cumulative probability distribution
P^*x, y	Weighted-average of pixels in a specified neighborhood

T	Image threshold value
h(b)	Image histogram
R_i	Image region i
$P_{\phi d}(a, b)$	Texture description operator
$\sum$	Covariance matrix
$C_{ff}(p, q)$	Texture autocorrelation function
F(u,v)	Image transform; Fourier, Hadamard, or Cosine function
d	Column vector dimension
ω_i	Set of predefined classes
n	Number of classes
ω_h	Drill-holes class
ω_b	Background objects class
$P(\omega_i\|\mathbf{x})$	Class posterior (*a posteriori*) probability
$p(\mathbf{x}\|\omega_i)$	Class-conditional probability function
$P(\omega_i)$	Prior (*a priori*) probability
$p(\mathbf{x})$	Scale factor
$g_i(\mathbf{x})$	MLC discriminate function
$\boldsymbol{\mu_i}$	The d dimensional mean vector of class ω_i
Σ_i	The $d \times d$ covariance matrix of ω_i
r^2	Squared Mahalanobis distance
g_h	MLC discriminate function for class ω_h
g_b	MLC discriminate function for class ω_b
n_p	Initial population of chromosomes (GA)
g_0	Initial generation of chromosomes (GA)
p_c	GA crossover rate
p_m	GA mutation rate
n_g	Maximum number of generations (GA)
f	GA fitness function
F_s	Feature set
f_i	Specific feature for an object
d	Class dimensionality, or number of features
$\mathbf{x}$	Feature vector, a fixed set of elementary features

${}^{x}\mathbf{P}$	Coordinates of a point P with respect to a coordinate frame x
${}^{x}\mathbf{R}_{y}$	Rotation matrix representing the orientation of frame y with respect to frame x
${}^{x}\mathbf{T}_{y}$	Location vector of the origin of frame y with respect to frame x
${}^{x}\mathbf{X}_{y}$	Pose of the coordinate frame
$\mathbf{X}$	Represents the transformation between reference frames
$\mathbf{B}$	Base reference frame of the robot boom
$\mathbf{E}$	Robotic boom end-effector reference frame, located at the boom-tip
$\mathbf{C}$	Camera reference frame located inside the camera housing
$\mathbf{S}$	Sensor reference frame located at the front of the sensor housing
u, v	Computer image coordinates
p	Image principal point
f	Effective focal length
p_x, p_y	x and y components of the principal point in pixels
f_x, f_y	x and y components of the focal length in pixels
$\mathbf{A}$	Camera intrinsic parameters matrix
k_1, k_2, p_1, p_2	Lens distortion coefficients
$P(x_w, y_w, z_w)$	3D world coordinate of the object point
$P(x, y, z)$	Coordinates of a object point
O_c	Optical center of the 3D camera coordinate system
O_i	Image coordinate system origin
$P_u(u_u, v_u)$	Image coordinate of the object point $P(x, y, z)$
$P_d(u_d, v_d)$	Image coordinate of the distorted object point $P(x, y, z)$
E_i	Robotic boom at position i
OBJ_i	Position i of a fixed object relative to the camera frame
dx, dy	Delta move in x and y directions
Z	Real world distance referenced from the lens to the object in mm

Chapter 1

Introduction

1.1 Automation in the Mining Industry

Much of the mineral resources in this country are accessible through underground mining operations. Underground mining can be simply described as two main activities: drift development (extending an underground tunnel toward or through an ore body), and production (removal of the ore to bring to the surface for processing). However, these activities involve manually controlled, complex pieces of equipment which are used to:

- Provide ground support (i.e. spraying Shotcrete) to protect miners operating equipment in the drift.
- Drill holes in the rock face (at front of a drift), the drift back (ceiling), or floor.
- Insert blasting material (emulsion) into the drilled holes.

- Insert primers and detonators into the emulsion prior to blasting.
- Muck (remove) the rock or ore after blasting using LHDs (Load-Haul-Dump vehicles).

The ground support/drill/load/blast/muck (GS/D/L/B/M) cycle is repeated as required to develop a drift and produce the available ore [1].

Given the nature of the GS/D/L/B/M cycle and a global economy, the Canadian mining industry faces serious obstacles with their current operations in the long term. The best and most accessible deposits in Canada have already been developed, and the costs of further exploration and development are greater than those in less developed countries. The Canadian mining industry also has no control over world prices, and thus is adversely affected by the decline in metal prices caused by lower cost foreign producers selling in a global marketplace. Also, rising expectations for social and environmental responsibility have resulted in additional costs borne by producers, which do not reduce production costs and are very difficult to recover. These obstacles have placed the Canadian mining industry at a competitive disadvantage relative to countries with more accessible mineral deposits, lower labor costs, and less restrictive environmental regulations [2].

Given the current trend, mining companies whether underground or open pit, are beginning to develop programs for both equipment and process automation, similar to those in the manufacturing industry with the anticipation that it will offset present obstacles. Automation has the potential to improve the Canadian mining industry's competitive position through increased productivity and reduced costs, while protecting the safety of workers and the environment. Productivity could be improved by reducing the amount of machine idle-time caused by operator absence

during work breaks, meals, and shift changes. Reduction in labor costs could be achieved by having only one operator operating multiple pieces of equipment from a remote location, or having fully autonomous machinery requiring no operator at all. Remote machine operation helps to protect the workers by removing them from the hazardous work environment of the mine, which will also reduce workers compensations costs incurred from accidents and injuries. Automated solutions to reflect strict environmental regulations, such as diesel emission controls, will help protect the environment [2].

The focus toward mining automation has already been realized as technology development programs have already been initiated. In 1996, companies Dyno Nobel, Inco Ltd., Tamrock, and Natural Resources Canada agreed to a five-year joint research and development plan, the Mining Automation Program, to investigate technology solutions for automating mining operations [3]. Inco Ltd. has implemented a vision-based autonomous guidance system called the Light Rope System, whereby a mining vehicle follows a specific path of the lighted rope to perform the required task. Implementation of this system has proved that a single miner could operate three machines at the same time. Recognizing the interest in automation, Laurentian University located in Sudbury, ON has established the Laurentian University Mining Automation Laboratory (LUMAL) with the objective to provide both educational and research expertise to develop, enhance, and support automation in the mining sector [4].

1.2 Emulsion Loading Process

In recent years, companies have been actively involved in automating the various components of GS/D/L/B/M mining cycle. However, attempts to automate the emulsion loading and blasting process (ELBP) have had no commercial success. The ELBP is practically the same for both development holes (12 - 15' long) and production holes (up to 150' long). The following outlines the steps of the ELBP which are performed manually with the assistance of emulsion pumping equipment:

- Emulsion hose is pushed to toe of drill-hole.
- Emulsion is pumped into the hole as the hose is withdrawn at a rate which deposits a specified amount of emulsion per foot of hole.
- Emulsion loading is completed before the front (collar) of the hole.
- A detonator and primer assembly is inserted to initiate the blast.

With manual development of a drift (all equipment is operator controlled and the emulsion and detonators are loaded manually), the GS/D/L/B/M cycle delivers 10 feet of drift per day. Using an automated cycle (all equipment is tele-remote controlled and emulsion and detonators are loaded automatically), advancing a drift 40 feet or more per day is entirely feasible. However, this level of productivity is not achievable due to the lack of technology for automated emulsion loading and blasting [1].

1.3 Emulsion Loading Automation Challenges

Automating the emulsion loading and blasting process is a rather challenging task. The automated system would first require a machine vision system to find the drill-holes. Given the underground mining environment is extremely harsh, where the atmosphere is moist and dusty, the use of a vision system would be very challenging. In addition, the vision system would have difficulty in locating drill-holes that are often obscured by overhanging rock, drill-holes that have been filled in by falling debris, or partial drill-holes resulting from a previous blast. Secondly, the automated system would also require a robotic boom and controller to move the emulsion hose to the drill-hole for loading. The robotic boom would also need to interact with the vision system to receive the appropriate guidance. A detonator loading and management system would load the detonator and primer components after loading the drill-holes with emulsion. Finally, a high level loading process control system would be required to supervise the entire emulsion loading process to ensure all required tasks are performed in the appropriate sequence and to deal with any problems that occur.

Given the challenging tasks of each system, and their mutual dependence, it is clear how difficult it is to automate the emulsion loading and blasting process for underground mining.

1.4 Objectives

The objective of this research is to develop a vision system that is tasked with the challenging imaging requirements of locating and identifying the difficult to

see drill-holes in the rock face that make up the drill plan in an underground mine. In addition, the vision system is also required to interact with a robot boom controller to provide visual guidance. This includes the development of drill-hole segmentation and classification algorithms using image processing techniques to find the drill-holes. In addition, visual guidance algorithms are developed to position a robotic boom in front of a drill-hole using processed image data. The data used for this research was collected from an underground research mine located in Sudbury, Ontario.

1.5 ELAP System Overview

The work in this thesis has been done as a part of a project called the Emulsion Loading Automation Project (ELAP). ELAP was a collaborative effort among team members to develop an intelligent system to automate the complex task of an emulsion loader used in underground mining. The team members included Inco Ltd. who provided funding and access to a research mine; Maclean Engineering, who developed the low level controller and human machine interface; C-CORE, who developed the vision system and supervisory controller; International Submarine Engineering Ltd. (ISE), who developed the robotic boom and controller; and ORICA Ltd., who developed the detonator handling and management system. Figure 1.1 shows the overall system diagram for the ELAP system.

The work in this thesis focuses on the development of the vision system component for the ELAP system. Figure 1.2 shows the vision system diagram. Upon request from the supervisory controller, an image is acquired from a video stream and image analysis is performed to find a drill-hole. Once an object has been correctly

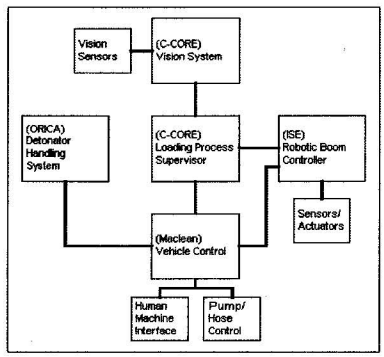


Figure 1.1: ELAP System components.

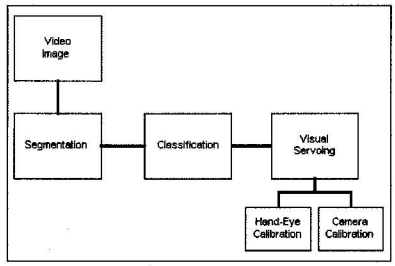


Figure 1.2: ELAP Vision system components.

identified as a drill-hole, image, sensory, and calibration data are used to guide a robotic boom to the drill-hole. This process continues until the robotic boom has been positioned in front of the drill-hole within a specified tolerance.

1.6 Document Organization

The first part of this thesis focuses on the primary requirement of the vision system which is the ability to correctly identify and locate a drill-hole within a digital image. A segmentation algorithm is first developed to isolate drill-hole candidates within the image. A classification technique using feature data is applied to identify which candidates are actually drill-holes and those that are background artifacts.

The second part of this thesis outlines the second requirement of the vision system: to provide visual guidance for positioning the robotic boom in front of a drill-hole for loading. Using a camera attached to the end of a robotic boom, the position and orientation of a drill-hole with respect to the camera frame must be determined, and then mapped to the robot boom frame to move the boom to the drill-hole. This overall mapping is achieved using both camera and hand-eye calibrations, which are presented in detail. A visual guidance algorithm is then used to visual servo the robotic boom to a drill-hole for loading.

Chapter 2 provides background information of image segmentation and analysis techniques. Chapter 3 presents the drill-hole segmentation and classifications algorithms developed to isolate and identify a drill-hole in an image. Chapter 4 outlines the camera and hand-eye calibrations used to calculate the mapping from image frame to robotic boom frame. It also presents the algorithm developed to visually guide the robotic boom to a drill-hole. Finally, the conclusions and recommendations for this work are given in Chapter 5.

Chapter 2

Background

In this chapter, many fundamental techniques used in this work for image pre-processing, image segmentation, shape representation, and blob analysis are presented. A brief explanation of typical morphological operations used in this work are also described. For a more in depth explanation of the various techniques presented, please consult the reference material as indicated.

2.1 Image Pre-processing

Image pre-processing is often the first stage in image analysis. Pre-processing does not increase information content but rather provides a means of suppressing information that is not relevant to the specific image processing or analysis task, and to enhance features or areas of concern. These operations include improving contrast, background isolation, smoothing, etc. This section provides an overview of various methods used for image pre-processing.

2.1.1 Window Leveling

If the contrast or brightness range within an image is very small, the information in the image may not be interpreted well by an observer or a program. Typical digitizers convert the analog voltage range from the camera or other sensor to numbers from 0 to 255 (8 bits). If the range of variation in brightness of the image is much smaller than the dynamic range of the camera or digitizer, then the actual range of values will be smaller than the full range of 0 to 255. This results in low image contrast and can be observed from the image histogram where the number of pixels at each of the 256 intensity levels is displayed. A narrow peak in the histogram indicates only a small range of intensities are represented.

However, the visibility of the image can be improved by performing window leveling, or contrast stretching. Window leveling takes a range of pixel values and maps them to a different range of pixel values. In essence, the input range is linearly "stretched" or "contracted" to the desired output range. By stretching the entire range or a sub-range of values present in an image to the full dynamic range, window leveling permits the use of all available intensities in the image to improve interpretation.

2.1.2 Histogram Equalization

Using histogram equalization can also enhance image contrast. This technique improves the image by equally distributing the intensity levels over the whole output intensity scale. Contrast is enhanced for intensity values close to the histogram maximal, and decreases contrast near minima. This is achieved by using the cu-

Table 2.1: Histogram equalization formulations.

Method	Output Probability Density Model	Transfer functions
Uniform	$P_g(g) = \frac{1}{g^{max} - g_{min}}$	$g = [g_{max} - g_{min}]P_f(f) + g_{min}$
Exponential	$P_g(g) = \alpha \left(e^{[(-a)(g-g_{min})]}\right)$	$g = g_{min} - \frac{1}{\alpha}\ln[1 - P_f(f)]$
Rayleigh	$P_g(g) = \frac{g-g_{min}}{\alpha^2}\left[e^{\left[\frac{(g-g_{min})^2}{2\alpha^2}\right]}\right]$	$g = g_{min} + \frac{1}{2}\left[2\alpha^2 \ln\left\{\frac{1}{1-P_f(f)}\right\}\right]$

mulative image histogram to find the pixel intensity transformation:

$$P_f(f) = \sum_{m=0}^{i} H_F(m) \tag{2.1}$$

where $P_f(f)$ is the cumulative probability distribution [5]. Several equalization formulations are used including a uniform histogram equalization, which results in a more uniform distribution of your image's pixel values, as well as exponential and Rayleigh equalization. These redistribution functions enhance contrast and increase dynamic range in a non-linear manner. Table 2.1 provides a summary of some of the more common histogram equalization formulations.

2.1.3 Neighborhood Averaging

Digital images often have high frequency spatial noise present that causes a reduction in image quality. Noise gets introduced into the data via any electrical system used for storage, transmission, and/or processing. To remove noise from digital images, low pass filtering techniques, or "smoothing" can be employed. One such

smoothing technique is neighborhood averaging. This method can be accomplished by replacing each pixel by a weighted-average of the pixels in some neighborhood around it.

$$P^*x, y = \frac{\sum_{i,j=-m}^{+m} W_{i,j} P_{x+i,y+j}}{\sum_{i,j=-m}^{+m} W_{i,j}} \tag{2.2}$$

where $W_{i,j}$ is the specified kernel weights, and P is the pixel intensity value. The weights are non-negative, and if all the weights are equal, then this is a mean filter [6].

2.1.4 Median Filtering

A median filter is another smoothing technique used to reduce noise in an image. It is similar to the neighborhood averaging process, but it preserves edges and other sharp features in the image. For each pixel being considered, its value is replaced with the median or middle value of its neighboring pixels when sorted. This linear filtering technique is very effective for removing shot noise. However, due to the large amount of sorting that is required, it is computationally expensive.

2.2 Segmentation Techniques

Image segmentation is a crucial step in the process of reducing raw image data into meaningful components or objects. There are a wide variety of segmentation techniques employed such as thresholding, edge-based, region-based, and textural-based operations. Generally, segmentation techniques can be grouped into two

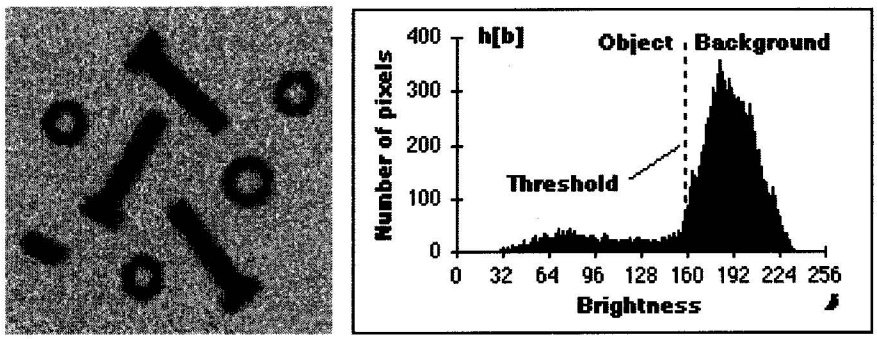
basic categories: discontinuity, and similarity. Edge-based segmentation belongs to the first category since this method focuses on the detection of lines, edges, and other discontinuities in the image. Thresholding, region-based, and textural-based methods belong to the second category where areas of similarity in the image data are to be exploited. The following is a brief outline of the various segmentation techniques mentioned above.

2.2.1 Thresholding

Thresholding is often the first technique applied to extract information from raw image data. Since images are characterized by intensity levels over a given dynamic range, a brightness level or threshold can be used to segment objects and the background. Thresholding is a computationally inexpensive and fast method of image segmentation that can easily be performed in real time using specialized hardware. Typically, a gray scale input image f is transformed to an output (segmented) image g by:

$$\begin{aligned} g(i,j) &= 1 \; for \; f(i,j) \geq T \\ &= 0 \; for \; f(i,j) < T \end{aligned} \tag{2.3}$$

where T is the threshold, $g(i,j) = 1$ represents objects, and $g(i,j) = 0$ represents the image background [7]. This method can be further extended to multi-band images whereby a threshold is selected for each image band (ie. r,g,b).



(a) Image to threshold (b) Image brightness histogram

Figure 2.1: Pixels below the threshold will be labeled as object pixels, those above will be labeled as background pixels.

2.2.1.1 Global Thresholding

In some instances, separation between the object and the background is significant enough to permit the use of a simple global threshold to segment the whole image. The intensity histogram of the image as shown in Figure 2.1, is analyzed for features such as peaks that often identify various homogeneous regions within the image with approximately the same gray-level. Depending on the gray-level distribution, there may be more than one peak representing a multi-modal histogram. Using the mode method, a threshold is found by selecting the histogram minima between the two highest local histogram maxima and apply it to the image for segmentation. However, this method does not guarantee correct segmentation in cases where objects are located on a background of varying gray levels [7].

2.2.1.2 Local Thresholding

In only very controlled situations may a simple threshold prove successful for the entire image. Often, gray-level variations in both objects and background resulting

from such factors as non-uniform lighting prevent the use of a single threshold for segmentation. As a result, segmentation using variable thresholds or adaptive thresholding in which the threshold varies over the entire image as a function of local image characteristics. This can be achieved by dividing the image f into sub-images f_c and determine the threshold for each using the corresponding subimage histogram:

$$T = T(f, f_c) \tag{2.4}$$

where T is the image or sub-image threshold.

2.2.1.3 Optimal Thresholding

Correct threshold selection for successful image segmentation is not easily achieved. As a result, various approaches have been developed to improve threshold selection for segmentation.

Isodata Method

This iterative technique for choosing a threshold was proposed by Ridler and Calvard [8]. First, the histogram is separated into two parts using a starting threshold value such as:

$$\theta_0 = 2^{B-1} \tag{2.5}$$

which is half the maximum dynamic range, where B is the image resolution. The sample mean $(m_f, 0)$ of the gray levels associated with the foreground pixels, and

the sample mean $(m_b, 0)$ of the gray levels associated with the background pixels are computed. A new threshold value θ_1 is then computed as the average of the two sample means.

$$\theta_k = (m_{f,k-1} + m_{b,k-1})/2 \ \ until\ \theta_k = \theta_{k-1} \tag{2.6}$$

Using the new threshold, the process is repeated until the threshold value does not change.

Background-Symmetry Method

This technique assumes a distinct and dominant peak in the image histogram for the background that is symmetric about its maximum. The maximum peak (max_p) is first found by searching for the maximum value in the histogram. The algorithm requires prior information for the image to search on the non-object side of the maximum to find a chosen $p\%$ point. Because of the assumed symmetry, the threshold is computed as a displacement to the left of the maximum (max_p), which is equal to the displacement to the right where $p\%$ is found.

$$\theta = max_p - (p\% - max_p) \tag{2.7}$$

This technique can easily be adapted to the opposite case where there are light objects on a dark dominant background.

Triangle Method

This technique presented by [9] is illustrated in Figure 2.2. A line is drawn between

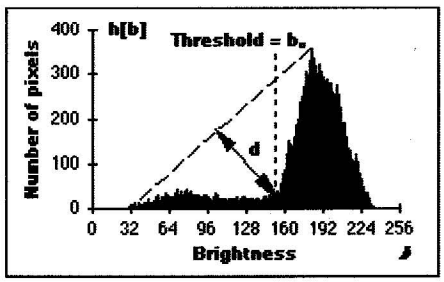


Figure 2.2: The triangle method is based on finding the value of b that gives the maximum distance d.

the maximum of the histogram at brightness b_{max} and the lowest level b_{min} for the image. The distance d, between the line and the histogram $h[b]$, is computed for all values of b from $b = b_{min}$ to $b = b_{max}$. The brightness value b_o where the distance between $h[b_o]$ and the line is maximal is selected as the image threshold value.

Otsu's Method

This method of thresholding separates the pixels into two classes C_0 and C_1 (objects and background) using a threshold t. C_0 represents pixels with intensity levels $0, 1, 2, \ldots, t$, and C_1 represents pixels with intensity levels $t+1, t+2, \ldots, L-1$, with L being the number of gray levels. Let σ_b^2 and σ_d^2 be the between-class variance and the within-class variance, respectively. An optimal threshold can be obtained by maximizing the separability of the histogram, that is, the ratio of the between-class variance and the within-class variance with respect to t,

$$\eta(t) = \frac{\sigma_b^2}{\sigma_d^2} \tag{2.8}$$

The approach performs well in situations where there is little contrast between background and the object of interest. However, this method assumes a bi-modal

histogram, and is very sensitive to noise which can often produce a biased result for t [53].

2.2.2 Edge-based Segmentation

Thresholding techniques segment the image to yield all the pixels that belong to the objects of interest in the image. An alternative to thresholding is to identify those pixels that belong to the borders or edges of the objects. Techniques that follow this approach result in a edge-based segmentation of the image. Edge-based segmentation relies on the presence of edges in the image that are identified as local gray-level intensity changes, or discontinuities in the gray-levels, color, or even texture. These discontinuities are represented in images where step-edges, lines, or stripes occur. In fact, it is the discontinuities that are used as visual cues when we interpret and understand image content.

Given the nature of discontinuities, most edge detectors facilitate the use of the gradient or first derivative, which simplifies to a difference operator across pixels. One of the early edge operators was developed by Roberts [6]. This involved the use of two 2×2 image masks applied to the image to calculate the gradient magnitude. Other edge operators such as the Prewitt and Sobel [6] use two 3×3 masks to obtain the gradient magnitude, but also incorporate the gradient direction to find edges. In instances where edges in an image have ramp intensity profiles rather than step-like, gradient detectors have erratic performance, and thus edge detectors using the second derivative, or Laplacian are more suited [6]. Still, other methods such as the Canny edge detector provides a more advanced approach to edge detection [10]. This method included the use of Gaussian smoothing filters to alleviate the

presence of noise, non-maximal suppression to find local maxima in the direction perpendicular to the edge, and hysteresis thresholding to identify which edges are significant and form a part of the edge contour.

The effectiveness of edge-based segmentation techniques depends on the nature of the edge information present in the image. It is difficult to design a general edge detection algorithm which will perform well in many applications. Usually, edges segmented rarely form closed contours around objects and thus subsequent algorithms such as edge linking must be applied to aid segmentation. In addition, noisy images containing speckles, or other high frequency data will obscure the edges of interest, making the use of edge-based techniques unsuitable in these conditions.

2.2.3 Region-based Segmentation

The aim of edge-based segmentation techniques is to find borders between regions or objects; however, in instances where borders are extremely difficult to detect, particularly due to noise, region growing methods may give better results. Region growing techniques focus on isolating homogeneous regions in the image, using a homogeneity criteria such as gray-level, color, shape, or texture. In general, the objective of segmentation is to divide an image R into a finite set of regions $R_1, ..., R_S$,

$$R = \bigcup_{i=1}^{s} R_i \qquad R_i \cap R_j = 0 \quad for\ i \neq j \tag{2.9}$$

For region-based segmentation, the following additional conditions must also be satisfied:

$$H(R_i) = TRUE \quad i = 1, 2, ..., S \tag{2.10}$$

$$H(R_i \cup R_j) \quad i \neq j \quad R_i \textit{ adjacent to } R_j \tag{2.11}$$

where S is the total number of regions in the image and $H(R_i)$ is the homogeneous binary result for region R_i. Regions R_i must also be maximal such that the homogeneity criterion would not hold true after being merged with an adjacent region [7]. Using these conditions, various techniques have been developed which employ region-based analysis.

2.2.3.1 Region Merging and Splitting

The most direct method of segmenting an image into regions is to begin by letting each individual pixel represent a single region. Each pixel is then compared to its neighbors and if the merging criterion is satisfied, the two regions are merged. The process continues between all neighboring regions, including newly formed ones, until the regions cannot be merged with any of its neighbors and it is marked as "final". The merging process stops when all regions have been so marked. The starting regions and merging criteria can be defined by any suitable method to begin the process. The result of region merging usually depends on the order in which regions are merged, and thus segmentation results will probably be different if merging begins in different image locations.

Region splitting is the opposite of region merging. It begins with the whole image represented as a single region, which does not usually satisfy the condition of homo-

geneity. The existing image regions are sequentially split to satisfy all conditions of homogeneity. Region splitting does not result in the same segmentation even if the same homogeneity criteria are used. Some regions may be homogeneous during the splitting process and therefore are not split any more [7].

A better segmentation may result by using a combination of splitting and merging methods. This approach uses a pyramid structure to represent the image, whereby regions are in the shape of squares and correspond to elements at a particular level. If any region at any pyramid level (excluding the start) is not homogeneous, it is split into four sub-regions to create a new level. If four regions with the same parent node exist at any pyramid level with approximately the same value of homogeneity measure, they are merged into a single region in an upper pyramid level. The segmentation process is similar to that of a quad tree where each leaf node represents a homogeneous region. Splitting and merging corresponds to removing or adding parts to the quad tree. The number of leaf nodes corresponds to the number of segmented regions after the split and merge process is over [11]. Unfortunately, this method assumes a fixed square shape for each region, which imposes a restriction on the true shape of a particular region.

2.2.3.2 Seeded Region Growing

The seeded region growing algorithm is based on the conventional approach of region growing where the image is segmented into homogeneous regions using a selected similarity measure. However, the mechanism for growing regions is controlled by choosing a selected number of pixels, referred to as seeds. The seed pixels have been isolated by segmentation techniques already applied and are grouped into

sets: $A_1, A_2, ..., A_n$. For each set of pixels representing a region, the immediate neighbors of each pixel in the set are compared to the region using the selected similarity measurement criteria. If the similarity measure is met, the pixels are labeled as part of the region. The process continues until all pixels in each set, including those newly added, have their neighboring pixels analyzed to maximize each region limits. This technique does not have a shape restriction, since each region is grown one pixel at a time.

2.2.4 Background Subtraction

The application of a background subtraction algorithm is a useful technique for segmenting images that have nonuniform brightness. Variations in image contrast and brightness can seriously affect the ability to segment objects in an image. The basis of this technique involves comparing an observed image with an estimate of the image if it contained no objects of interest (ie. the background). As the name implies, the technique subtracts the estimated background image from the observed image, and thresholds the resultant image to segment the objects of interest.

To apply the background subtraction algorithm, an estimate of the image background must be generated. This is accomplished by dividing the gray scale image into non-overlapping sub-images, or blocks of a specified size. The average of each of these blocks will then correspond to one pixel of a newly created block-averaged image [12, 13]. For example, a gray scale image of size 640 $\times$ 480 pixels and using a block size of 32 pixels, the resulting image has dimensions of 20 $\times$ 15. The block-averaged image produces an estimate of the overall lighting profile for the image.

The block-averaged image has an 8-bit dynamic range (i.e. each pixel has an intensity range from 0 to 255) which is then normalized between 0 and 1. The normalization process results in dark areas with values close to zero to become darker, whereas brighter areas with values close to one will change very little. The end result is an image where the separation between the lighter and darker areas is increased. Repeating the normalization process will further increase the separation between dark and light areas. A threshold is selected that provides optimal separation between the objects and the background. The image is rescaled to the full dynamic range, and resized to the original dimensions using cubic interpolation.

2.2.5 Texture Segmentation

Many images contain regions that are not characterized by a unique value of brightness, but rather a variation of brightness that is often referred to as texture. Generally, texture refers to properties that represent the surface or structure of an object. Formulating a precise definition for textural objects is hard to achieve due to its wide variability.

However, given that textural features may be used to identify objects, texture is often used during image segmentation and object classification stages. In image processing, texture may be defined as a group of mutually related pixels often referred to as a texture primitive or texture element having particular gray-level properties and spatial organization [7]. Various methods have been used to estimate and describe texture.

2.2.5.1 Gray Level Co-occurrence

The gray level spatial dependence approach of texture description characterizes texture by the co-occurrence of its gray levels. The distribution varies rapidly with distance in fine textures, and slowly in coarse textures. Te gray level co-occurrence can be specified in a matrix of relative frequencies $P_{\phi d}(a, b)$, which describes how frequently two pixels with gray-levels a and b appear in a region separated by a distance d in direction f. For angles quantized to 45 intervals, the non-normalized frequencies are defined as:

$$P_{0,d}(a,b) = |\{[(k,l)\,(m,n)] \in D : k-m = 0, |l-n| = d, f(k,l) = a, f(m,n) = b\}|$$

$$P_{45,d}(a,b) = |\{[(k,l)\,(m,n)] \in D : k-m = d, |l-n| = -d, f(k,l) = a, f(m,n) = b\}|$$

$$P_{90,d}(a,b) = |\{[(k,l)\,(m,n)] \in D : k-m = d, |l-n| = 0, f(k,l) = a, f(m,n) = b\}|$$

$$P_{135,d}(a,b) = |\{[(k,l)\,(m,n)] \in D : k-m = d, |l-n| = d, f(k,l) = a, f(m,n) = b\}|$$

where $|\{...\}|$ refers to set cardinality, $D = (M \times N) \times (M \times N)$, and $M \times N$ is a rectangular sub-image of size M by N. Texture classification can then be achieved by calculating features from the co-occurrence matrix [14].

2.2.5.2 Joint probability Distribution

A generalization of the gray level co-occurrence approach is to consider more than two pixels at a time. Given a local neighborhood (i.e a 3×3 neighborhood) and a subimage, a parametric estimate of the joint probability distribution of the gray levels over the neighborhoods in the subimage can be calculated. For parametric estimation, the multivariate normal is used. If $\mathbf{x_1}, ..., \mathbf{x_N}$ represent the N K-normal

vectors coming from neighborhoods in a subimage, then the mean vector $\boldsymbol{\mu}$, and covariance matrix $\boldsymbol{\Sigma}$ can be estimated by:

$$\boldsymbol{\mu} = \boldsymbol{\mu_0}\mathbf{1}, \quad where: \quad \boldsymbol{\mu_0} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{1}\mathbf{x_n} \tag{2.12}$$

$$and \quad \Sigma = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x_n} - \boldsymbol{\mu})(\mathbf{x_m} - \boldsymbol{\mu})' \tag{2.13}$$

Where $\mathbf{1}$ is a column vector whose components have the volume 1 [15].

2.2.5.3 Autocorrelation Texture Description

Measurement of spatial frequencies provides a means for texture recognition. Textural character is directly related to the spatial size of texture primitives. Primitives of large size are indicative of coarser textures with lower spatial frequency, while primitives of smaller size are indicative of finer textures with higher spatial frequency. The autocorrelation function is a method to describe the size of textural primitives. Using this model, a single pixel is considered a texture primitive and the correlation coefficient evaluates linear spatial relationships between primitives. If the textural primitives a relatively large, the autocorrelation function will drop off slowly with distance; if the primitives are small, the autocorrelation function will drop off quickly with distance. If primitives are spatially periodic, the autocorrelation function will drop off and rise again in a periodic manner. The autocorrelation coefficients can be evaluated for different values of p, q by

$$C_{ff}(p,q) = \frac{MN}{(M-p)(N-q)}\frac{\sum_{i=1}^{M-p}\sum_{j=1}^{N-q} f(i,j)f(i+p,j+q)}{\sum_{i=1}^{M}\sum_{j=1}^{N} f^2(i,j)} \tag{2.14}$$

where p, q is the position difference in the i, j direction, and M, N are the image dimensions [7].

2.2.5.4 Discrete Image Transform

The application of an image transform may also be used for texture description. Typically, an image is divided into a set of non overlapping square sub-images. If the subimage size is $n \times n$, the gray-levels of its pixels may be thought of as an n^2-dimensional vector, and an image can be represented as a set of vectors. These vectors are then transformed using various techniques into a new coordinate system where they relate to spatial frequencies of texture present in the image and can be used for texture description. The following transforms methods are often used.

Fourier Transform

$$F(u,v) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) exp\left[-2\pi i \left(\frac{mu}{M} + \frac{nv}{N}\right)\right] \tag{2.15}$$

$$u = 0, 1, ..., M-1 \quad v = 0, 1, ..., N-1$$

Hadamard Transform

$$\mathbf{F} = \mathbf{H}_{MM} f \ \mathbf{H}_{NN} \quad f = \frac{1}{MN} \mathbf{H}_{MM} \mathbf{F} \ \mathbf{H}_{NN} \tag{2.16}$$

Cosine Transform

$$F(u,v) = \frac{2c(u)c(v)}{N} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m,n) cos\left(\frac{2m+1}{2N} u\pi\right) cos\left(\frac{2n+1}{2N} v\pi\right) \tag{2.17}$$

$$u = 0, 1, ..., N-1 \quad v = 0, 1, ..., N-1$$

where

$$\begin{aligned} c(k) &= \frac{1}{\sqrt{2}} \; for \;\; k = 0 \\ &= 1 \quad otherwise \end{aligned} \tag{2.18}$$

2.2.6 Shape-based Segmentation

If an image contains objects of known shape and size, segmentation can be viewed as the process of finding these objects in the image. A common method to find specific objects is to use a mask of appropriate shape and size, and look for correlation between it and the image. However, mask performance is greatly reduced if the objects are partially occluded, or distorted.

2.2.6.1 Hough Transform

These issues may be overcome by using the Hough transform, which is tolerant to these problems. The Hough transform is a technique which can be used to isolate features of a specified shape within an image. It requires that the desired features be represented in some parametric form, i.e. a curve. The classical Hough transform is most commonly used for the detection of regular curves such as lines, circles, ellipses, etc. For more complicated shapes, the Generalized Hough transform is used since a simple analytic description of a feature is not possible. In this case, instead of using a parametric equation of the curve, a pre-defined look-up table representing the feature boundary description is used.

During implementation, the algorithm uses an accumulator array A, with bin dimensions corresponding to the number of unknowns in the parametrized curve. For example, to find line segments the equation $d = r\sin\theta + c\cos\theta$ can be used, where d is the perpendicular distance from the line to the origin, θ is the angle the perpendicular makes with the x-axis, and (r, c) are the corresponding constant (x, y) image pixel coordinates known. The accumulator array has quantized values of d and θ. Using edge pixels of candidate objects resulting from prior image processing, the parameters of the specified curve are calculated. The parameters for each pixel are then quantized to the appropriate values d and θ and the accumulator $A(d, \theta)$ is incremented. After all pixels are processed, the accumulator array is searched for peaks. These peaks indicate the parameters of the most likely lines occurring in the image [15]. For circles, the equation $(x_1 - a)^2 + (x_2 - b)^2 = r^2$ is used where the circle has center (a, b) and radius r. In this case, the accumulator array is three dimensional, and the cell $A(a, b, r)$ is incremented.

The main advantage of the Hough transform technique is that it is tolerant of gaps in feature boundary descriptions and is relatively unaffected by image noise. However, the Hough transform is only efficient if a high number of votes fall in the correct bin, such that the bin can be easily detected amid the background noise. Therefore, bin size must be carefully chosen to ensure votes do not fall in neighboring bins, thus reducing the visibility of the main bin. Also, with a high parameter count (¿3), the vote count per bin will be low, which again causes problems during detection. As a result, the Hough transform must be used carefully when detecting curves with complicated shapes.

2.2.7 Conclusion

The preceding material provided a brief overview of the common segmentation techniques of thresholding, edge-based, region-based, background, texture-based, and shape-based segmentation. As stated, the techniques can generally be divided into two categories: discontinuity or similarity. Edge-based techniques attempt to exploit the discontinuities present in an image. These techniques rely on the presence of edges in the image that are identified as local gray-level intensity changes resulting from lines or stripes. The effectiveness of the these techniques is limited in noisy images containing random frequency data that obscures the edges of interest.

Image thresholding is typically the first technique used to process an image, and belongs to the second segmentation category. A brightness level is often determined either on a global or local basis using various methods, and applied to the image respectively. Thresholding requires minimal processing overhead, and can be performed in real time when using specialized hardware. However, determining a threshold value to achieve accurate segmentation results is rather challenging, and thus it is often combined with other techniques.

Region-based and textural-based techniques focus on areas of similarity within the image. Region-based methods attempt to isolate homogeneous regions using a selected criteria such as gray-level, or color. However, the process of identifying a region is often computationally expensive. In contrast, texture-based methods attempt to isolate regions by their variation in brightness. Formulating a texture definition for the textural objects is difficult due to high variability within the region.

For shape-based segmentation, a prior information such as shape and size about

the object, can be used to isolate from the image. Rudimentary objects such as squares, rectangles, circles, and lines can be easily defined and applied to an image using a mask. However, mask performance is often reduced if the objects are partially occluded, or distorted, or if complex object shapes are being sought.

2.3 Morphological Operations

Morphological operators in image processing are used to modify the shape or form of a region or object. A morphological operation is a type of neighborhood operation that determines the new value for a pixel based on the relationship between its neighborhood and a given set of numbers. The numbers are often represented in a matrix known as a structuring element. The effect of a morphological operation depends on the algorithm used for the operation type and the composition of the structuring element being used [5]. The basic operations of binary morphology are erosion, dilation, opening, and closing [14].

2.3.1 Binary Erosion and Dilation

Erosion is the morphological operation that applies the structuring element to the image, such that at each position where every 1-pixel of the structuring element covers a 1-pixel of the binary image, the binary image pixel corresponding to the origin of the structuring element is ORed to the output image [14]. The erosion of binary image X by structuring element B is denoted by $X \ominus B$ and is defined by

$$X \ominus B = \{x |\ x + b \in X\ \forall b \in B\} \tag{2.19}$$

Erosion, as the name implies, reduces or shrinks the object or region, and is often used to remove extraneous pixels and small particles from the image.

Dilation is the morphological dual of erosion. Dilation applies the structuring element to the image, such that each time the origin of the structuring element touches a binary 1-pixel, the entire translated structuring element shape is ORed to the output image [14]. The dilation of binary image X by structuring element B is denoted by $X \oplus B$ and is defined by

$$X \oplus B = \bigcup_{b \in B} S_b \tag{2.20}$$

Dilation adds layers to objects or particles, thereby enlarging the objects.

2.3.2 Binary Opening and Closing

Erosion followed by dilation performs a morphological operation called opening. The opening of an image X by a structuring element B is denoted by $X \circ B$ and is defined as

$$X \circ B = (X \ominus B) \oplus B \tag{2.21}$$

Opening removes small particles and breaks isthmuses or connections between touching objects.

Dilation followed by erosion performs a morphological operation called closing. The closing of an image X by a structuring element B is denoted by $X \bullet B$ and is

defined as

$$X \bullet B = (X \oplus B) \ominus B \tag{2.22}$$

Closing fills holes in objects and connects close objects.

Although the morphological operators presented have been applied to binary images, the operations can also be applied to gray scale images [7]. Refer to the cited material for a more in-depth study.

2.4 Blob Analysis

The objective of the segmentation process is to isolate objects of interest within the image and discard the background. Once this is achieved, information about the objects is often required and is usually determined by performing blob analysis. Blob analysis is a branch of image analysis that provides the means to identify connected regions of pixels, referred to as blobs, within an image. Blobs are areas of touching pixels that are in the same logical pixel state. This pixel state is called the foreground state, while the alternate state is called the background state. Typically, the background has the pixel value 0 and the foreground is everything else, usually 255 for binary images, or vice versa.

In many applications, we are only interested in blobs whose features satisfy a certain criteria. Since feature computation can be time-consuming, blob analysis is often used as an elimination process whereby only blobs of interest are considered for further analysis. By calculating selected features for each blob, feature criteria

is used to automatically discard blobs that are not of interest, and calculate the required information for those remaining.

The Matrox Imaging Library (MIL Ver. 6.1) [5] software package includes a blob analysis module that can extract a wide assortment of blob features, such as the blob area, perimeter, extract holes from blobs, Feret diameter at a given angle, minimum bounding box, compactness, as well as many others. A typical blob analysis procedure is as follows:

1. Grab or load an image that was captured under good lighting conditions
2. Process the image to reduce the amount of noise present.
3. Segment the image to separate blobs from the background and each other to produce a binary blob identifier image. The original gray scale image is required to perform gray scale calculations as well.
4. Select a result buffer and feature list applicable to the objects of interest.
5. Calculate the features and analyze the results. Exclude those blobs that do not meet the specified feature criteria.
6. Repeat the calculation of features and blob exclusion until only the desired blobs remain.

2.5 Conclusion

This chapter provided a brief review of the many techniques used for image pre-processing, image segmentation, shape representation, and blob analysis. The in-

tent of this review was to provide a brief explanation of some techniques that may be used during image processing algorithm development.

Drill-hole images typically have poor contrast and will require application of pre-processing techniques before segmentation can be performed. The rather noisy, coarse nature of a rock face excludes the use of edge-based techniques, as the resulting image would contain thousands of edges that would provide little assistance during drill-hole segmentation, and therefore was not used. As a result, a combination of thresholding and region-based techniques were employed to isolate the typical dark region of a drill-hole. Using the Matrox Imaging Library [5], morphological operations can be applied to fill holes in objects, and remove small objects. The blob analysis component may be used to calculate features, and remove or select blobs meeting the selected feature criteria.

Chapter 3

Hole Segmentation: Image Analysis and Drill-Hole Classification

One of the main tasks in the development of the ELAP vision system is to identify the difficult to see holes in the rock face that make up the blast pattern in an underground mine. A video camera attached to the end of a boom provides digital images of a rock face that are to be analyzed to determine the presence of a hole. Once the analysis is complete, information collected from the image is used to calculate coordinate transformations to position the camera/boom assembly in front of the hole for loading. Therefore, the first task in developing the ELAP vision system is the development of image processing and classification algorithms to segment and identify holes present in an image of an underground rock face.

3.1 Image Analysis

Figure 3.1 contains an assortment of 640 × 480 color JPEG images of holes drilled in various underground rock faces. These images illustrate the challenging imaging requirements of the vision system to operate in an underground mine. For instance, support structures such as wire screen, Shotcrete (mixture of metal fibers and concrete sprayed on the rock wall), and rock bolts are often used to improve the mines integrity, and can obscure the locations of drill-holes. Rock and other fallen debris often occupy the drill-hole collar, and again obscure its location. The rock type and surface also makes it difficult to locate drill-holes. The surface is typically non-planar which often causes shadows to be cast in the vicinity of a drill-hole. The rock surrounding a drill-hole can be very different in color, wet from ground water, or exhibit glare when exposed to a light source, making it difficult to locate in an image due to lack of contrast. In addition, attempting to locate the hole when the camera is off center from the hole's axis can exacerbate the factors previously described. At first glance, the suggestion of using simple thresholding may be a sufficient means for segmenting the drill holes in the image. However, a review of the database of underground images collected shows the difference both within and between images, as well as lighting conditions and the other factors, would not make this approach adaptive to these variations. Therefore, the drill-hole images will first undergo pre-processing to improve the quality of the image such that successive techniques can achieve better performance when applied. Techniques such as window leveling or histogram equalization will be required to improve the image contrast so that the image content may be better segmented and interpreted.

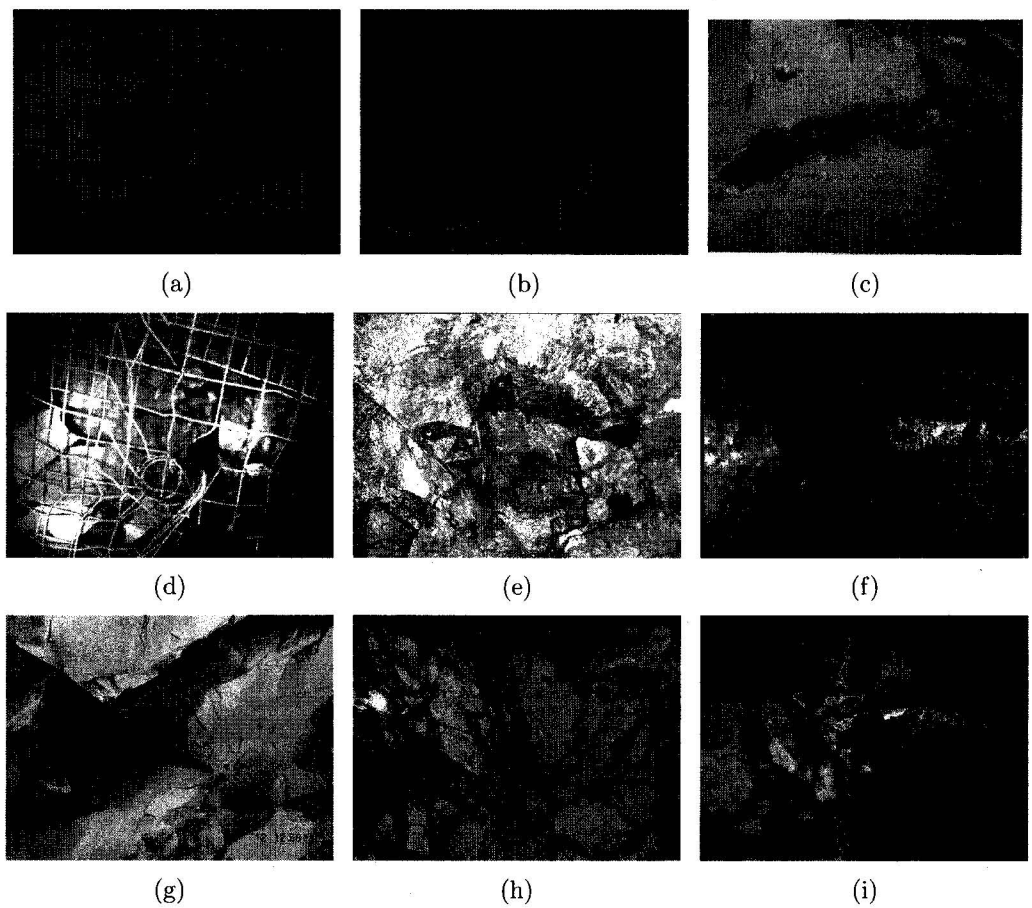


Figure 3.1: Typical images of underground rock faces. (a), (b), and (c) are Shotcrete with water and rust. (d) shows wire screen on rock face. (e) exhibits glare. (f) shows a dark image. (g) holes hidden behind a rock brow. (h) and (i) show holes obscured by debris.

Due to the challenging imaging requirements, the provision of adequate uniform lighting will play a critical role in effectively and reliably segmenting a hole from an image. Artificial lighting located next to the camera will attempt to provide an optimum quality image for analysis and subsequent segmentation of drill holes. However, the image processing algorithms developed must be able to cope with an image acquired under less than ideal conditions.

3.1.1 Segmentation Scheme Selection

As stated in Chapter 2, segmentation methods can be separated into two categories: discontinuity and similarity, due to the dominant techniques employed. The first group relies on the presence of edges in the image that are identified as areas with local gray-level discontinuities. However, they have poor response in images containing significant background noise. This is particularly true for images of an underground rock face. As shown in Figure 3.1, the non-planar surface of the rock face resulting from blasting would produce an image entirely composed of edges if such a technique was employed in an attempt to find the edges of the drill-hole. Isolating the edges of the drill-hole from that of the surrounding rock surface would be very challenging. As a result, edge-segmentation methods would further complicate the segmentation of drill-holes from the image, and thus are not appropriate for this application.

The second group comprising of threshold-based, region-based, and textural-based techniques, focus on utilizing image content that is similar or common in the image or a portion of it. Threshold-based methods are effective when objects are characterized by having a different homogeneous gray level as compared to that of

the background. Using the brightness histogram, a peak corresponding to the gray level intensity of the object sought may be used to segment it. Given the variation in brightness levels for the images shown in Figure 3.1, the image will first undergo pre-processing to improve image contrast, and then use a suitable thresholding technique for segmentation. Given the limited information from the image color space in drill-hole images, the ELAP Vision System (VS) will use monochrome images when applying various methods during segmentation.

Due to the limited contrast in drill-hole images and irregular textures of a rock face, both region and textural-based methods will not be used for primary segmentation. Rather, a region-based technique will be used to define the boundaries of possible drill-holes after they have been segmented during the initial stages to see if they are in fact a drill-hole or a background object. Textural features will be used to classify the candidates segmented as a drill-hole or a background artifact, rather than a direct method for segmentation.

3.1.1.1 Image Segmentation Process

As stated previously, image analysis will be performed using monochrome images since there is little information in the original color image that could aid in drill-hole segmentation. Thus, the original color image is converted to a monochrome image and is input to the segmentation process.

However, it is worth noting that before any segmentation is performed on the image, an undistortion process must first be applied to remove any distortions resulting from the lens characteristics. This lens distortion causes the image contents to be skewed from where they should actually be located in the image. Given that the

hole center is used during the visual servoing process, an accurate projection must be provided to ensure correct guidance. A more detailed explanation of the lens distortion issue is provided in Chapter 4.

The first step in image segmentation is to apply pre-processing techniques to improve image contrast. Given the irregular shape of the rock face and lack of uniform lighting, many dark shadowy areas exist in the image and create a low contrast effect that must be improved. Next, using the resulting pre-processed image, a threshold is determined and applied to binarize the image to produce an image composed of foreground objects (drill-holes), and background artifacts (shadows, water, etc). Finally, morphological techniques are applied to remove small objects and fill holes. Figure 3.2 illustrates the segmentation process for the drill-hole image. The steps required for image segmentation are presented and discussed in the following sections.

3.1.1.2 Image Pre-processing

As stated in Chapter 2, image pre-processing is often the first stage in image analysis. Pre-processing operations help to improve image contrast by suppressing information that is not relevant, and enhancing areas of interest. From the images shown in Figure 3.3, non-uniform lighting in combination with the rock surface can create shadow areas that make it hard to discern a drill-hole from the rest of the background. In addition, thresholding in these areas may cause false objects to be segmented and confused with an actual drill-hole. Thus, pre-processing techniques need to be applied first in order to improve contrast and image clarity.

To improve image contrast, a window leveling technique is applied. Referring to

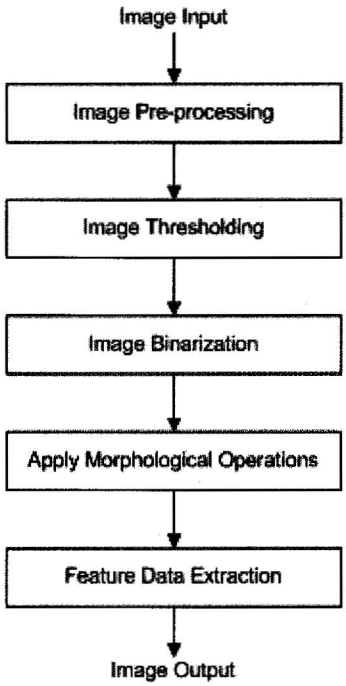


Figure 3.2: Flow chart illustrating image segmentation process.

the images in Figure 3.3, drill-holes typically have a small brightness range consisting of low intensity values $(0 - 30)$ while the remaining intensities correspond to the background. This range represents only a small portion of the full dynamic range of an image $(0 - 255)$, and is observed at the left of the histogram as shown in Figure 3.4. The range of intensities to re-map starts from 10, the lower value, to the intensity value that corresponds to a percentage (30%) of the area under the histogram curve starting from 0. Intensities greater than the 30% value are all set to the maximum intensity value (255). These values were determined after investigating an extensive database of drill-hole images of actual underground mines, and provide very good results.

To ensure optimal response, this process is performed at a local level using a moving window of size 160×120 pixels, where the range for re-mapping is determined

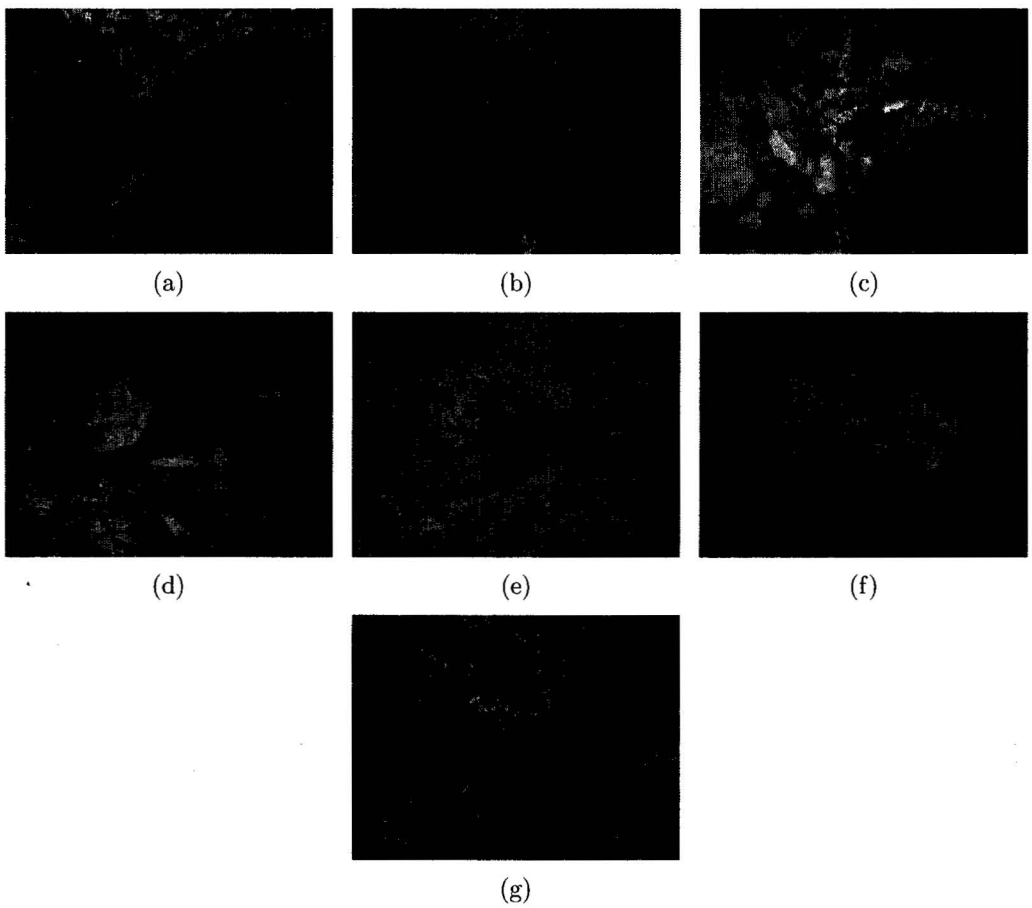


Figure 3.3: Examples of images with poor lighting. Images indicate difficulty in discerning the drill-hole from the background.

dynamically for each sub-image. After the window leveling process is performed, the image shows an improved contrast compared to that of the original by maximizing the dynamic range. This process retains all dark image content and brightens the remaining background to provide improved object segmentation. The window leveling result is demonstrated using the image in Figure 3.3(f) as shown in Figure 3.5.

However, in images with poor contrast, drill-holes appear 'washed out' and result in poor segmentation after window leveling is applied. This is particularly evident with simulated drill-hole images captured in the presence of background ambient lighting. Also, images of drill-holes where the surrounding area is wet tend to reflect light, which again, results in reduced contrast and sub-optimal drill-hole segmentation. As a result, additional techniques to process images where the rock face is wet, may need to be explored, and adequate lighting conditions are critical to ensure adequate and reliable drill-hole segmentation is achieved.

The sample image in Figure 3.3(f) is further used to illustrate the results from the subsequent image segmentation processes applied. The segmentation results for all the images in Figure 3.3 are included in Appendix A, and on the accompanying CD-ROM at the end of this document.

3.1.1.3 Thresholding and Morphology

After the image has been pre-processed, the resulting image is then binarized using a thresholding operation. Thresholding results in a binary image (0/1) where white pixels represent the background, and black pixels represent objects that may be drill-holes since their intensity value is lower than the selected threshold value. The

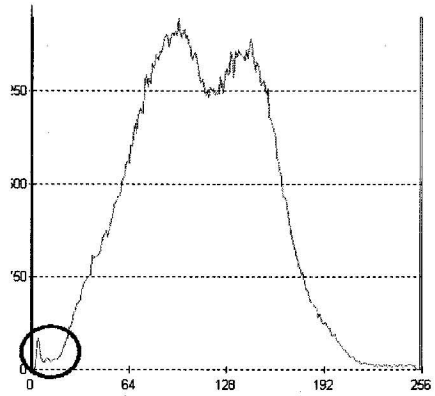
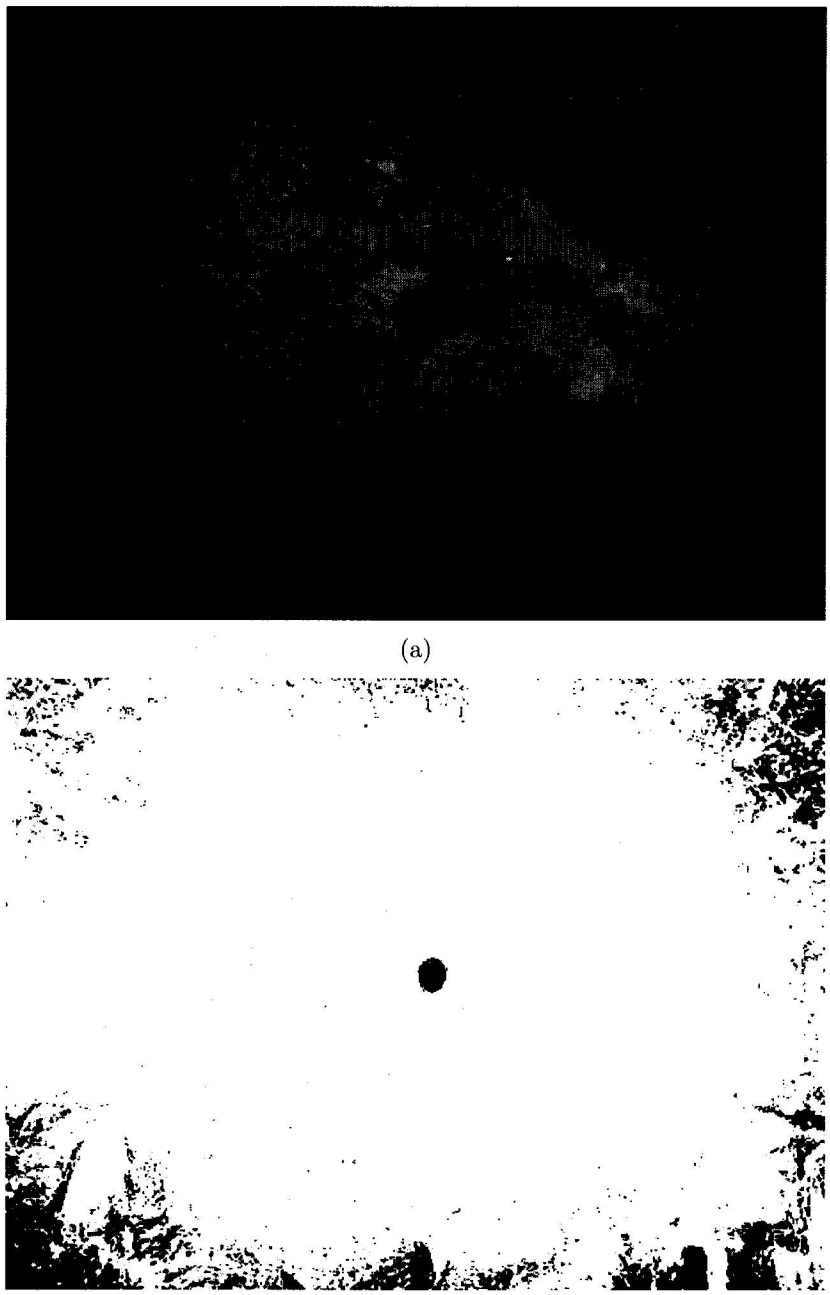


Figure 3.4: Sample drill-hole image identifying intensity range of interest at left of the histogram as indicated.

threshold value is determined using the respective pre-processed image histogram. The binary image after the threshold is applied is shown in Figure 3.6(a). Image thresholding is then followed by morphological opening and closing operations that remove small image objects, connections between touching objects, and fill holes within objects. Figure 3.6(b) shows the resulting image after morphological operations are applied to Figure 3.6(a). Small objects have been removed leaving only a few objects to remain for further processing and identification.

3.1.1.4 Feature Data Extraction

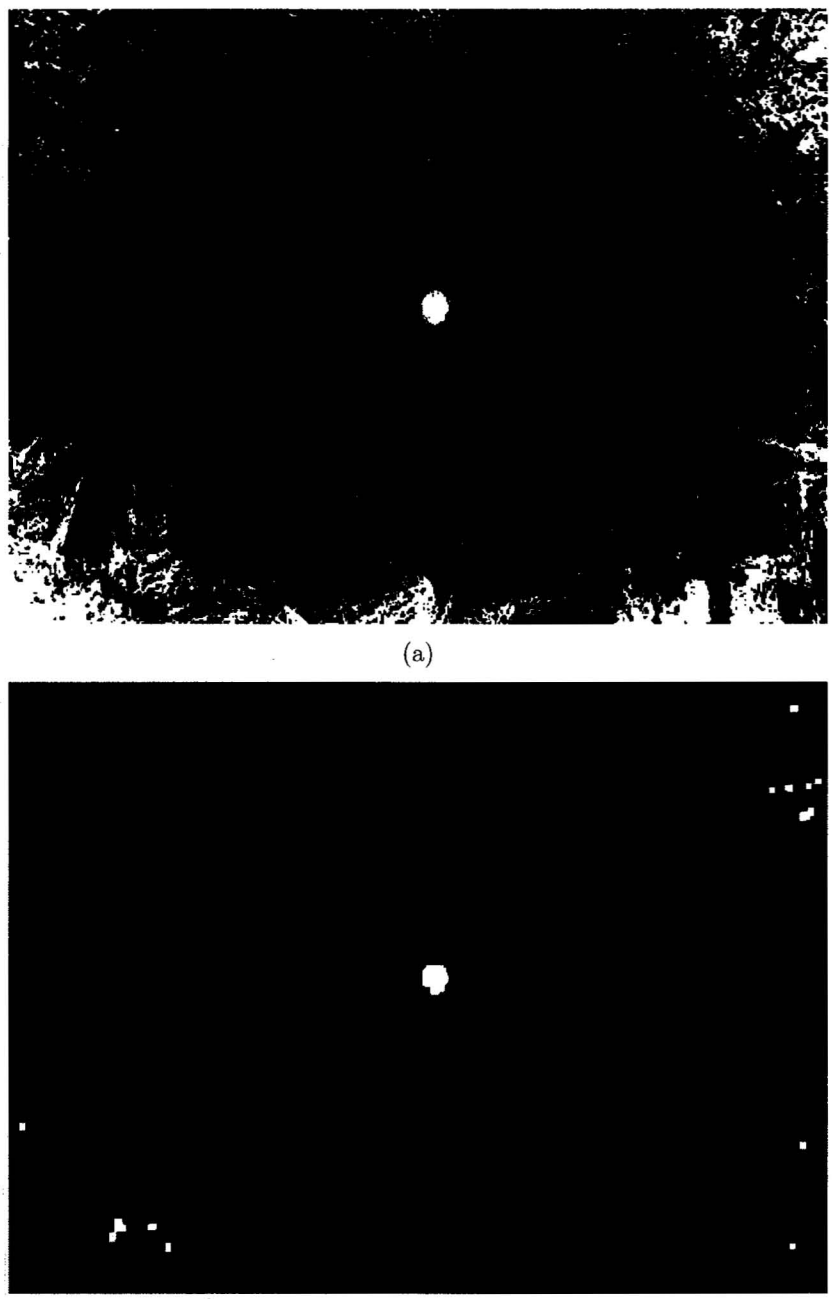
The resulting segmented image in Figure 3.6(b) contains potential drill-holes objects that must be identified as such. This is achieved by performing blob analysis on the image to calculate object feature data. The analysis data is first used to further exclude small objects with $area < 10$, and objects touching the image border. Selected features are then calculated for the remaining blobs (objects) and input to a classification process discussed in Section 3.2, to identify them as either drill-holes or background objects.



(a)

(b)

Figure 3.5: Drill-hole with window leveling applied. (a) original image, (b) result after window leveling.



(a)

(b)

Figure 3.6: Drill-hole image after binarizing and morphology.

3.1.2 Conclusion

The preceding image segmentation scheme resulted in reliable isolation of potential drill-hole objects. A computationally intensive image processing algorithm could not be afforded since the overall segmentation process is required to be sufficiently fast to facilitate efficient visual servoing of the end-effector to a drill-hole. Thus, a simplified segmentation approach was taken to fulfill this requirement given that, (1) the camera will be positioned approximately in front of the drill-hole by using recorded drill plan information, thus reducing the search area; (2) a pattern recognition model will be applied to the segmented objects to assist in identifying which objects are drill-holes and which are background artifacts; and (3) the drill plan information will also be used to determine which of the classified objects is of interest using location information. Thus, a simplified segmentation approach, along with classification and location information provides the system response required for visual servoing. The next section outlines the development of the pattern recognition model used to classify segmented image objects, including feature selection, classifier design, and optimization.

3.2 Drill-Hole Classification

Once the drill-hole image has been segmented, potential candidates are isolated. However, not all objects segmented are drill-holes and thus a standard drill-hole pattern or model must be compared to each segmented object for classification. The model is designed using image features of segmented drill-hole objects from a large number of samples. Image features of new drill-hole candidates are compared

to the feature values of the training set to see if they are similar. Based on the result of the particular metric used for comparison, the object can be labeled as belonging to one of two classes: drill-holes or background objects. The following sections present the work performed to develop the pattern recognition model used to classify segmented image objects.

3.2.1 Classification Model

The classical pattern recognition approach as shown in Figure 3.7, is that of taking raw data and assigning the data to one of several potential classes. As shown in the previous section, raw image data is processed to segment potential drill-hole objects. The next step is feature extraction, where sensed properties or features describing the segmented objects are calculated. For each object, a pattern or feature vector $\mathbf{x}$, is a fixed set of elementary features of length d, where $\mathbf{x}$ is the d dimensional column vector. Using this approach, an object can be abstractly represented as a point in d dimensional feature space as shown in Figure 3.8 [16]. In practice, different objects input to the feature extractor will produce different feature vectors. However, the hope is that the within-class variability is small relative to the between-class variability [16].

The next step is classification, where $\mathbf{x}$ is evaluated and assigned to one class, ω_i, from the set of predefined classes $\{\omega_1 \cdots \omega_n\}$, where $n = 2$ for this work. For this work, the statistical maximum likelihood classifier (MLC), based on Bayesian decision theory was used. This classifier accounts for differences in scale between features, compensates for highly correlated features, and permits curved as well as linear class decision boundaries [16]. The classifier is trained using feature data

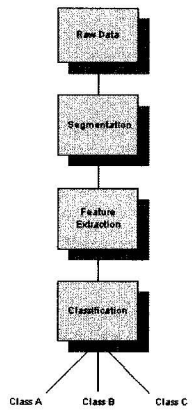


Figure 3.7: Classical model for pattern recognition.

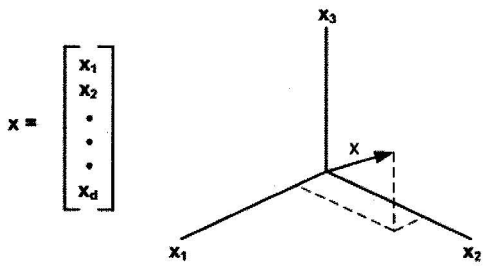


Figure 3.8: Feature vector in d-dimensional feature space.

from known objects belonging to each class, and new objects are then compared to the training set to determine if they can be labeled as belonging to one of the two classes.

To achieve optimal classification performance, the classifier normally requires optimization. This is often accomplished by selecting the best classifier parameters, or more commonly, by feature subset selection (FSS). FSS consists of choosing a subset from the initial set of features used to describe the target objects. Ideally, the feature subset would provide high separability for the target objects, and any redundant or irrelevant features are removed. In this work, FSS is used for classifier optimization and three selection methods are investigated, namely: a genetic al-

Table 3.1: The initial 19 features proposed for classification.

(1) Minimum Pixel	(11) Binary Central Moment X1Y1
(2) Mean Pixel	(12) Energy
(3) Pixel Standard Deviation	(13) Entropy
(4) Compactness	(14) Homogeneity
(5) Feret Elongation	(15) Contrast
(6) Central Moment X0Y2	(16) Inverse Diff. Moment
(7) Central Moment X2Y0	(17) Correlation
(8) Central Moment X1Y1	(18) Elongation
(9) Binary Central Moment X0Y2	(19) Roughness
(10) Binary Central Moment X2Y0	

gorithm (GA), sequential forward selection (SFS), and an exhaustive search (ES). Although the exhaustive search method would undoubtedly provide the optimum feature subset, the GA and SFS methods were investigated to compare their performance given their reduced computational time. Finally, the classifier performance for each of the FSS techniques is presented, including a summary of the results.

3.2.2 Feature Selection

The basic problem that pattern classification is concerned with is the assignment of a given object to one of n known classes. An object is represented by a pattern of features which, presumably, contains sufficient information to distinguish among the classes. Instead of performing a thorough investigation of features that would be most suitable for drill-hole objects, this work focused on FSS. Typically, an initial set of features describing relevant object characteristics is considered, and then a subset is selected that maximizes the classifiers performance. As previously stated, the subset selection methods investigated include a GA, SFS, and ES.

The initial set of features included those commonly used as a starting point for

image analysis, and were applicable to the target object being sought. In addition, several textural features were included that produced encouraging results during the initial feature investigation. Features that were irrelevant, indicated a possible bias, or are inefficient or difficult to calculate, were removed. The idea was to include only relevant features while trying to keep the feature count to a minimum in order to achieve system robustness and minimize the computational overhead. Having a high feature count would make it difficult to isolate an optimum subset within the feature space, and thus decrease the classification performance.

Thus, the initial 19 features calculated for each object are shown in Table 3.1 (refer to Appendix B for feature definitions). The numbers in brackets are used to reference the specific features from this point onward. The final feature subset is determined during classifier optimization.

3.2.3 Feature Vector Extraction

To develop the recognition model, drill-hole images were segmented and features extraction algorithms were employed to produce a feature vector for each of the selected drill-hole and background objects. The classifier then processed the feature vector, and assigned the object to the class that has similar feature space characteristics. A total of 1170 drill-hole and background objects were gathered and used for both classifier training and testing. The sample object data is divided between the object classes as described in Section 3.2.4.

Table 3.2: Summary of the data set samples for the classes.

Class	No. of Samples
ω_h	520
ω_b	650
total samples	1170

3.2.4 Object Classes

The 1170 feature vector samples of both drill-hole and background objects were gathered from image ground truth data. As a result, two object classes were created

1. *drill-holes*; hereafter indicated by the subscript h, as in ω_h
2. *background*; hereafter indicated by the subscript b, as in ω_b

The drill-hole feature data was selected from sample images captured in a research mine under average lighting conditions. The diffuse lighting provided good separation between the drill-holes and their surroundings. The lighting conditions also caused the drill-hole region to have an homogeneous appearance in the image, with intensity values less than 30. Also, the drill-holes are relatively centered in the image, and captured on-axis with the drill-hole to provide a more circular shape. The background object feature data was taken from the same set of images, and is comprised of background image artifacts segmented during analysis. These background objects represent shadows created due to rock texture, or areas of wet rock. These objects vary in size and shape, with mean intensity values covering the full dynamic range. Table 3.2 shows how the 1170 data objects are grouped for each class.

3.2.5 Classifier Design

For this work, the statistical MLC was selected which is based on Bayesian decision theory [17]. Bayesian classification is based on Bayes formula given by

$$P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})} \tag{3.1}$$

where $P(\omega_i)$ is the prior (*a priori*) probability, $P(\omega_i|\mathbf{x})$ is the posterior (*a posteriori*) probability that the object belongs to class i given the sampled feature vector $\mathbf{x}$, $p(\mathbf{x}|\omega_i)$ is the class-conditional probability function for $\mathbf{x}$ given that the class is ω_i, and $p(\mathbf{x})$ acts as a scale factor that ensures the posterior probabilities sum to one, which is usually dropped for classification.

For classification, the approach is then to minimize the probability of classification error by choosing the class that maximizes the posterior probability $P(\omega_i|\mathbf{x})$. Bayes decision rule is then to sample $\mathbf{x}$, calculate $P(\omega_i|\mathbf{x})\forall i \in 1 \ldots N$ and select the class ω_i for which $P(\omega_i|\mathbf{x})$ is greatest. The Bayesian classifier results in the maximum theoretical classification accuracy, however it requires knowledge of $p(\mathbf{x}|\omega_i)$ which is typically unavailable.

Following the maximum-likelihood approach, $p(\mathbf{x}|\omega_i)$ is modeled as a Gaussian distribution, and the MLC can be formulated using the discriminate function

$$g_i(\mathbf{x}) = P(\omega_i|\mathbf{x}) = p(\mathbf{x}|\omega_i)P(\omega_i) \tag{3.2}$$

If f is a monotonically increasing function, $g_i(\mathbf{x})$ can be replaced with $f(g_i(\mathbf{x}))$ without changing the classification results. Hence, the discriminate function is

written in logarithm form as

$$g_i(\mathbf{x}) = \ln p(\mathbf{x}|\omega_i) + \ln P(\omega_i) \tag{3.3}$$

Therefore, using the Gaussian for ω_i, $p(\mathbf{x}|\omega_i)$ can be formulated as

$$p(\mathbf{x}|\omega_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \, exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right] \tag{3.4}$$

where $\mathbf{x}$ is the d dimensional feature vector of an sample object, $\boldsymbol{\mu}_i$ is the d dimensional mean vector of class ω_i, and Σ_i is the $d \times d$ covariance matrix of ω_i calculated from the training data set. The MLC discriminate function for ω_i can be derived by substituting Equation 3.4 into Equation 3.3 to give the following

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) - \frac{d_i}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i) \tag{3.5}$$

In the above equation, $(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}) = r^2$ is known as the squared Mahalanobis distance. The Mahalanobis metric is a non-linear distance measurement that compares a sample feature vector $\mathbf{x}$, to the mean $\boldsymbol{\mu}$ of class ω_i. The use of Σ in the Mahalanobis classifier accounts for differences in scale between features, compensates for highly correlated features, and permits curved as well as linear class decision boundaries [16].

Recall from Section 3.2.4, two object classes were created, thus $i \in \{h, b\}$. The sample data for ω_i is used to calculate the classifier parameters, specifically, the mean feature vector $\boldsymbol{\mu}_i$, the covariance matrix Σ_i, and the determinate $|\Sigma_i|$. The term $d_i/2\ln(2\pi)$ is a constant because the dimensionality of both classes is fixed: $d = 19$. $P(\omega_i)$ is the MLC prior probability of class i, and is intended to improve

classification results by helping to resolve confusion among classes that are poorly separable. However, reliable class prior probabilities are difficult to determine, and thus for this work they were set equal.

$$P(\omega_h) = P(\omega_b) = 0.5 \tag{3.6}$$

Hence, the final MLC discriminate functions are

$$g_h(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_h)^T \Sigma_h^{-1}(\mathbf{x} - \boldsymbol{\mu}_h) - \frac{1}{2}\ln|\Sigma_h| + \ln P(0.5) \tag{3.7}$$

$$g_b(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_b)^T \Sigma_b^{-1}(\mathbf{x} - \boldsymbol{\mu}_b) - \frac{1}{2}\ln|\Sigma_b| + \ln P(0.5) \tag{3.8}$$

The above equations formulate the MLC design, such that a new sample feature vector $\mathbf{x}$ is classified as ω_h if $g_h > g_b$, otherwise its classified as ω_b.

3.2.6 Classifier Optimization

In this work, classifier optimization is accomplished by FSS. The objective of FSS is to determine the set of features which provide the greatest separation between the different classes. Initially, all relevant features shown in Table 3.1 are included to describe the objects characteristics. However, some features may be biased, noisy, or highly correlated with other features. The number of features should also be kept to a minimum without affecting classifier performance. Minimizing the feature number will also improve the software speed by reducing the calculation overhead during classification.

Mathematically, FSS is a search problem. Depending on which search method is used, FSS methods can be categorized into three techniques: (a) random search

(e.g. genetic algorithms), (b) heuristic search (e.g. sequential selection), and (c) optimal search (e.g. exhaustive search algorithms). While random searches are computationally more efficient, optimal search strategies guarantee an optimum solution [18]. To perform FSS, the three selection strategies listed above are investigated using the MLC determinant as the evaluation criteria to measure the accuracy of the feature subset.

3.2.6.1 FSS: Genetic Algorithm

Genetic algorithms are a stochastic optimization method inspired by the theory of natural selection and evolutionary processes [18]. GA's are an efficient method of iteratively searching a large sample space and arriving at near-optimal solutions based on some fitness function. GA-based feature selection was first introduced by Siedlecki & Sklansky [19], and has been actively studied by numerous researchers such as Yang & Honavar [20], Richeldi & Lanzi [21], and Sun et al. [22]. In GA-based feature selection, the n potential features are mapped onto an n bit chromosome where each bit (or gene) represents a specific feature. If the gene = 1 then that feature is used, otherwise it is not. For example, in the chromosome 1001111111111111111, all 19 features as enumerated in Table 3.1 are included in the classifier instance, except for the features 2 and 3.

The algorithm commences by randomly selecting an initial population of chromosomes, n_p for the initial generation g_0. Each chromosome is then assessed by the fitness function (MLC for this work) to calculate its fitness f. The chromosomes are ranked in order of fitness, the higher the chromosome fitness, the higher the probability it will be selected to mate. The mating is performed using the sin-

gle point crossover operator, where the crossover point is randomly selected. The crossover rate, p_c, determines if the chromosomes undergo mating or if they simply migrate to the next generation, g_{i+1}, unchanged. The mutation operator is then applied to generation g_i of chromosomes where each bit has a probability, p_m, of being mutated. Finally, the chromosomes are evaluated using the corresponding feature subset. The generated classifier is then trained and tested using the fitness function (MLC) and the results are compared to determine if the terminating criteria (acceptable fitness value or maximum number of generations, n_g) is satisfied. If the terminating criteria is satisfied, the algorithm returns the chromosome with the highest fitness, otherwise any duplicates in the population are removed, and the process repeats. The GA implementation is summarized in Algorithm 1.

The GA algorithm parameters were not investigated specifically, to determine the optimal values for this application. Rather, standard GA parameter values were used that have been derived from recognized sources [23, 24, 21] and were set to the following values: $n_p = 10$, $p_c = 0.7$, and $p_m = 0.02$.

Algorithm 1 FSS using a Genetic Algorithm (GA)

Require: initial population of chromosomes of size n_p
1: evaluate fitness of each chromosome using fitness, f
2: **for** $g = 1$ to n_g **do**
3: chromosomes selected from P_{g-1} based on fitness
4: create P_g using crossover operator, p_c
5: mutate each chromosome in P_g using p_m
6: evaluate fitness of each chromosome in P_g using f
7: **end for**

3.2.6.2 FSS: Sequential Forward Selection

The SFS algorithm begins with an empty set of features, F_s, and sequentially evaluates and adds features to F_s. Once a feature f_i, has been identified as the best feature for the feature space being evaluated, F_e, it is permanently added to F_s, and removed from the list of features, F_T, available for subsequent selection. The SFS algorithm is listed in Algorithm 2. However, SFS cannot guarantee the optimality of the subset found since the sequential nature of searching may result in it getting stuck in a local minima [25]. This also applies to GAs, however the GA mutation (random) operator attempts to overcome this possibility.

Algorithm 2 FSS using Sequential Forward Selection (SFS)

Require: $FS = \{\}, ES = \{\}, TS = \{f_1, f_2, \ldots, f_n\}$
Ensure: $FS \subseteq TS\,\{\}$
1: **for** $i = 1$ to n **do**
2: $accuracy = 0$
3: $bestFeature = \{\}$
4: $bestAccuracy = 0$
5: **for** $j = 1$ to length(TS) **do**
6: $ES = TS(j) \cup FS$
7: train discriminate functions (MLC) for feature space TS
8: calculate accuracy from discriminate functions ES
9: **if** (accuracy > bestAccuracy) **then**
10: $bestAccuracy = accuracy$
11: $bestFeature = TS(j)$
12: **end if**
13: **end for**
14: $bestFeature$ assigned to FS
15: $bestFeature$ removed from TS
16: **end for**

3.2.6.3 FSS: Exhaustive Search

The exhaustive search strategy is a brute force technique that examines all possible subsets 2^n (where n is the number of features), and finds the optimal one. It starts with an empty set, then considers all possible subsets containing one feature, two features, and so on, up to the entire set of n features. For each subset containing i features, the MLC discriminate functions are trained and the accuracy is calculated. The combination of i features having the best accuracy is then stored in a feature subset accuracy buffer. The process is then continued for the remaining subsets of $\{i+1, i+2, \ldots, n\}$ features. Once complete, the feature subset buffer is searched to find the subset of i features with the best accuracy. This subset of i features is then the optimum set for this sample data. This can be computationally expensive especially for a large initial feature set. Thus, for $n = 19$ features, a total of 524, 288 subsets of features were examined for each classifier (ω_h, ω_b). The algorithm was implemented in Matlab, running on a P4 3.2GHz computer, and took just under two days to complete. This was much longer that the GA and SFS approaches which took on the order of hours to complete. The exhaustive search implementation is shown in Algorithm 3.

3.2.7 Classifier Performance

During FSS optimization, the classifier performance for both classes, (ω_h, ω_b), was determined implicitly. For each feature subset selected, the classifier was trained and tested using the leave-one-out cross-validation method (LOOCV) [26]. The sample data for each class containing 520 drill-holes, and 650 background objects was divided such that a single observation from each sample set, as well as the

Algorithm 3 FSS using an Exhaustive search strategy.

Require: $FS = \{f_1, f_2, \ldots, f_n\}, ES = \{\}$
Ensure: $FS \subseteq TS\,\{\}$
1: **for** $i = 1$ to n **do**
2: $accuracy = 0$
3: $BS(i) = \{\}$
4: $BSAccuracy[i] = 0$
5: **for** $j = 1$ to Perm(i, n) **do**
6: $ES(j) \subset FS$
7: train discriminate functions (MLC) for feature space $ES(j)$
8: calculate accuracy from discriminate functions $ES(j)$
9: **if** (accuracy > BSAccuracy[i]) **then**
10: $BSAccuracy[i] = accuracy$
11: $BS(i) = ES(j)$
12: **end if**
13: **end for**
14: **end for**
15: optimum $FS = BS(i)$ with $max(BSAccuracy[i])$

sample data for the other class, was used for testing, and the remaining samples were used to train each classifier. The process was repeated until each sample is used for testing. Using the MLC classifier discriminants, each test sample feature vector $\mathbf{x}$ is classified as ω_h if $g_h > g_b$, otherwise its classified as ω_b. The average error rate was then calculated for each feature subset identified using Equation 3.9.

$$E = \frac{1}{n}\sum_{i=1}^{n} E_i \tag{3.9}$$

The classifier results for each of the FSS optimization techniques is summarized in the corresponding confusion matrices as illustrated in Table 3.3 and Table 3.4. The accuracy for each optimized feature space identified using each FSS strategy is shown in Figure 3.9. The classifier accuracy found for each FSS strategy was very similar differing only by 0.5%. Also, each strategy resulted in approximately the same number of features, 8 for the GA, and 9 for both the SFS and ES clas-

Table 3.3: Classifier performance for each FSS optimization strategy illustrated in a confusion matrix. ω_b samples = 650, ω_h samples = 520.

(a) GA

	ω_h	ω_b
ω_h	516	48
ω_b	4	602

(b) SFS

	ω_h	ω_b
ω_h	506	32
ω_b	14	618

(c) ES

	ω_h	ω_b
ω_h	504	30
ω_b	16	620

sifiers. Five features were also found common to all three classifier configurations, $(4, 9, 14, 18, 19)$. As shown in Figure 3.9, after feature space of size 7, the accuracy for each optimization strategy tended to level off and only begin to worsen for the GA configuration. Thus, indicating that the addition of more features did little to improve the accuracy, and only increased the computational overhead.

All but one of these features are shape based, while the other $feature(14)$ is based on gray level intensity. An initial thought might be to exclude all but shape based features from the feature set, but the exhaustive search clearly indicates that the inclusion of this feature does increase the classification performance for this training data set. Another training set may show otherwise, and a somewhat different optimum feature subset altogether.

Arguably, any of the three classifier configurations could be used for the final design given their similar accuracies, with only a slight increase in computation for the SFS and ES configurations resulting from the extra feature. Although, the ES strategy identified the optimum feature subset, the time required for training makes it somewhat impractical if re-training is needed and the number of features or samples in the training set is increased. However, for this work, re-training the classifier was not required, and thus the feature subset found using the ES strategy was used in the final classifier design.

Table 3.4: Optimized feature space and accuracy for each FSS optimization strategy (Feature subset enumerated as in Table 3.1).

FSS Strategy	Accuracy	No. of Features	Opt. Feature Subset
GA	0.9556	8	4,8,9,11,13,14,18,19
SFS	0.9607	9	4,5,8,9,12,14,16,18,19
ES	0.9607	9	4,5,7,9,13,14,17,18,19

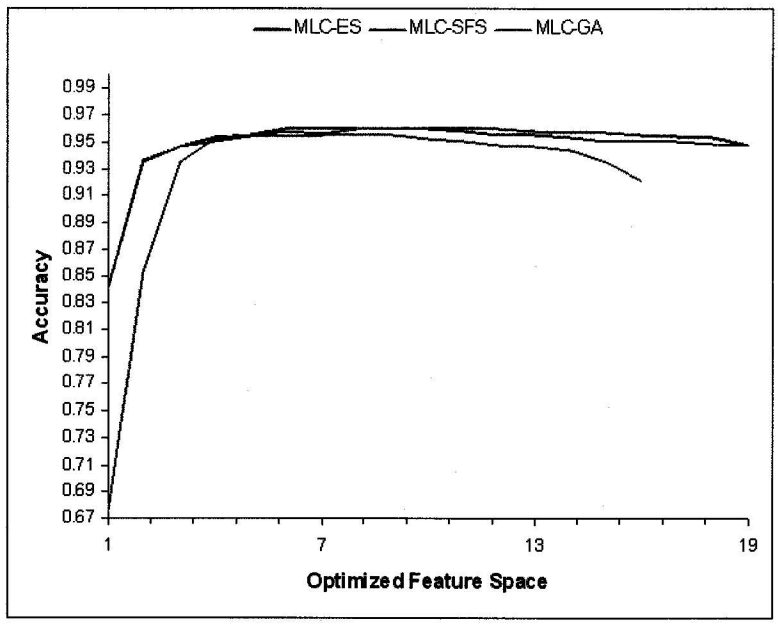


Figure 3.9: Classifier performance for FSS optimization strategies: GA, SFS, and ES.

3.3 Conclusion

The first task in developing the ELAP VS was the development of the image processing algorithms to segment and identify holes present in an image of a underground rock face. An underground mine is a dynamic environment where non-uniform lighting and drill-hole obscurities challenge the imaging requirements. The image processing algorithms developed attempted to mitigate these conditions for successful segmentation of drill-hole objects. Initial image pre-processing techniques such as window leveling were employed to improve image contrast. Thresholding and morphological operations were then applied which remove small image objects, connections between touching objects, and fills any holes within objects.

However, not all objects segmented are drill-holes and thus a classification model was developed using image features calculated with blob analysis, of actual drill-hole objects. The classifier was designed using the MLC discriminate and image features of new drill-hole candidates are compared to the feature values of the training model. The candidate object is then classified as either a background artifact and the object is discarded, or as a drill-hole where the results are returned to the calling function for further consideration. FSS selection techniques were employed to both reduce and find an optimum feature subset for classification. This resultant feature set achieved an accuracy of 96.07% for drill-hole and background objects.

Overall system performance is further increased as the camera is always positioned in front of the drill-hole at start, and drill plan location information is used to determine which of the classified objects is of interest. However, the resulting image segmentation and object classification performance is dependent on image

quality which requires suitable lighting, and how representative the classification model is compared to new data input.

Chapter 4

Vision System Design

4.1 System Overview

The requirements of the ELAP vision system (VS) is to first identify the difficult to see drill-holes in the rock face that make up the drill plan of in an underground mine (Chapter 3), and secondly, use the corresponding image data to guide a robot boom to position an emulsion tube for insertion into the hole for loading with emulsion (i.e. an explosive). Using a vision system to accomplish this type of objective has already been implemented successfully in the manufacturing setting [27, 28, 29, 30]. However, the dynamic nature of the underground mining environment adds an extra level of complexity to the overall system design that must be mitigated. As a result, the vision system must be sufficiently robust to operate in the underground environment without compromising the required functionality.

The proposed vision system conceptual design is illustrated in Figure 4.1. It involves ridgedly mounting a camera/lens assembly to the end-effector of a robotic

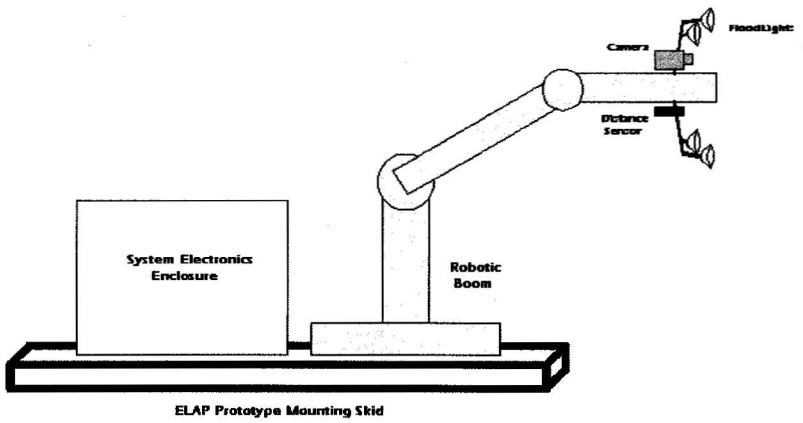


Figure 4.1: ELAP prototype concept illustrating vision system components. Supporting electronics including industrial computer, framegrabber, and other required hardware and software are housed in the system electronics enclosure.

boom. This location is required to provide optimum viewing capabilities when imaging the rock face. The design also includes rigidly mounting a distance sensor, and a light source to the robotic boom's end-effector. The distance sensor serves to provide the depth information lost during image capture, which is required when performing image analysis and visual servoing. The halogen lighting is needed to illuminate the viewing area with a diffuse light pattern to provide optimum imaging without restricting the working area of the robotic boom. An industrial computer on-board the ELAP unit houses a framegrabber that samples the cameras video signal and presents the image data in digital format for analysis. Installed application software developed, incorporates the image processing algorithms used to analyze image data to identify drill-holes, and communicates with the Loading Process Supervisor (LPS) control software. The drill-hole image data and previously determined calibration parameters will be used to transform the image information between the camera and end effector reference frames to perform visual servoing of the robotic boom to a drill-hole for loading. The LPS controller terminates the visual servoing process once the robotic boom-tip is within a specified tolerance of

the drill-hole collar.

The following outlines the various vision system components, both hardware and software, used in the design. Also, a detailed description of the visual servoing control process including established reference frames, required calibrations, and their corresponding parameters is discussed. Finally, results from both lab testing and field trials are presented.

4.2 System Components

The ELAP Vision System (VS) is comprised of both hardware and software components as illustrated in Figure 4.2. The system hardware consists of a Pulnix TMC-7DSP progressive scan CCD color camera, a Pentax 3.5mm wide angle lens, and a Matrox 4-Sight II Pentium class industrial PC. The camera output is an analog RGB-Sync signal and is input to the PC via a Matrox Meteor-II/4MB PCI frame grabber for digitizing. Distance measuring is performed using a Massa M-5000 ultrasonic distance sensor over a serial communication link, and polled by the VS software during visual servoing. The VS is connected via Ethernet to the LPS over TCP/IP, and is configured as a server to perform its specified tasks during the emulsion loading process.

The VS software, $VisSys$, is a command-line based application developed using Microsoft Visual C++ 6.0, the Intel Open Source Computer Vision Library (OpenCV) Beta 2.0, and the Matrox Image Processing Library (MIL) 6.1 running under the Windows NT 4.0 operating system. The $VisSys$ software consists of two main components: system software, and the analysis/control software. The system software

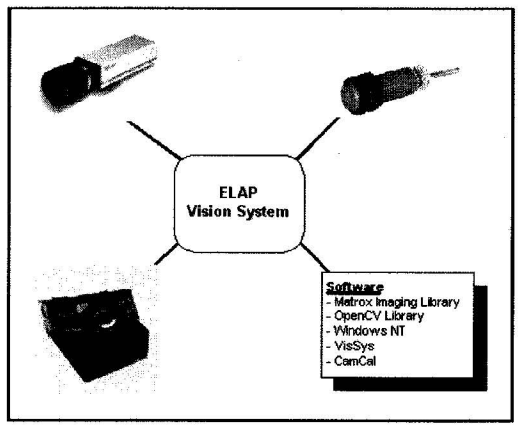


Figure 4.2: ELAP vision system hardware and software components.

component is responsible for configuring the system components, monitoring system hardware and its resources, image acquisition, and subsystem communications. The analysis/control software encapsulates the image processing and classification algorithms used to locate and identify drill-holes in captured images. It also contains the control process used for visual servoing the robotic boom once a drill-hole has been identified.

A separate supporting software application, *CamCal*, performs both the camera and hand-eye calibrations required by the VS. This software is a Windows-based application developed using Microsoft Visual C++ 6.0, MIL 6.1, and OpenCV Beta 2.0. The *CamCal* software is used to determine the camera and hand-eye calibration parameters required to visual servo a robotic boom when using image based techniques.

4.3 Visual Servoing

Vision is a useful robotic sensor since it simulates the human sense of vision and allows for non-contact measurement of the environment [31]. By using visual information, the *pose* (position and orientation) of a robotic manipulator may be controlled relative to a target object. This type of control is referred to as *visual servoing.*

Visual servoing is accomplished by first knowing the different transformations (i.e. translations and rotations) between the various system reference frames. The coordinates of a point P with respect to a coordinate frame x is noted by ${}^{x}\mathbf{P}$. Given two coordinate frames, x and y, the rotation matrix representing the orientation of frame y with respect to frame x is denoted by ${}^{x}\mathbf{R}_{y}$, while the location of the origin of frame y with respect to frame x is denoted by the vector ${}^{x}\mathbf{T}_{y}$. Together, the position and orientation represent the pose of the coordinate frame, and is denoted by ${}^{x}\mathbf{x}_{y} = ({}^{x}\mathbf{R}_{y}, {}^{x}\mathbf{T}_{y})$ [31].

The robot used for the ELAP system was required to have 6 joints, providing it with 6 degrees-of-freedom (DOF) to move (Figure 4.3). The robot's reference frame, **B**, is located at the base of the robot and its working area is defined with reference to this point. The end-effector reference frame (interchangeably referred to as boom-tip frame), **E**, located at the center of the boom-tip, is referenced to the robots base frame. It is the end-effector frame origin that is used to visual servo the emulsion hose to a hole for loading.

The ELAP VS reference frames and corresponding transformations are illustrated in Figure 4.4. The camera reference frame **C**, cannot be accurately measured since

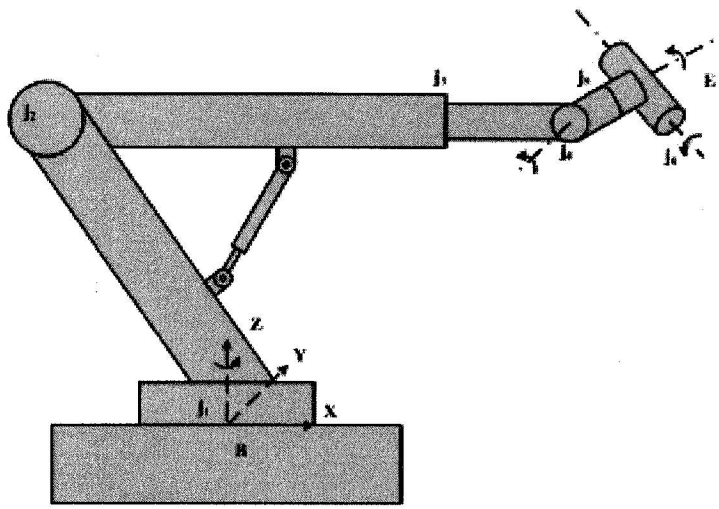


Figure 4.3: ELAP robotic boom conceptual design. 6-joints to provide 6-degrees of freedom (6-DOF). **B**, robot base frame; **E**, robot end-effector frame.

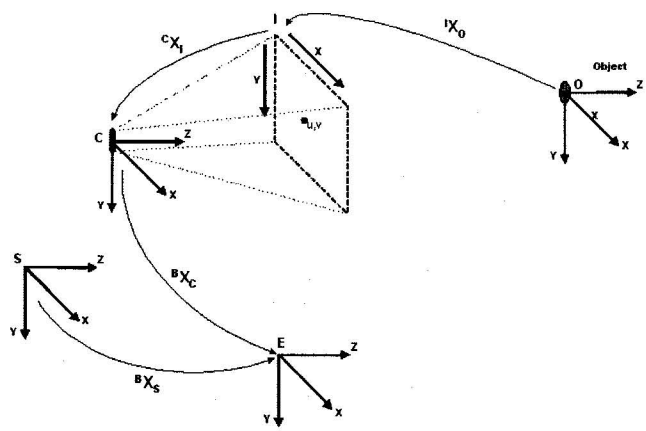


Figure 4.4: Vision System reference frames and transformations. The reference frames include: **E** – robot end-effector frame; **C** – camera frame; **I** – image frame; **S** – sensor frame; **O** – object frame.

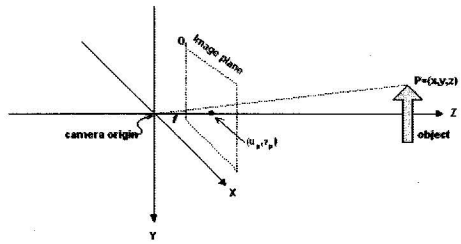


Figure 4.5: Camera geometry with perspective projection.

it is located inside the camera housing, and is specific to each camera-lens configuration. It is the geometric relationship between the camera coordinate frame, **C**, and the end-effector frame, **E**, that is critical to perform visual servoing, and therefore must be determined.

The geometry model for a camera/lens system is shown in Figure 4.5. The x and y axis form a basis for the image plane, the z-axis is perpendicular to the image plane along the optical axis, and intersects the image plane at the principle point $\mathbf{p} = [u_p, v_p]$. The cameras origin is located at a distance f, behind the image plane, where f is the effective focal length of the camera lens.

The camera lens forms a 2D projection of the scene on the image plane where the camera sensor is located. This projection causes direct depth information to be lost as each point on the image plane corresponds to a ray in 3D space. The projective geometry of the camera is modeled by perspective projection, where a 3D object point ${}^c\mathbf{P} = [x, y, z]^T$, whose coordinates are expressed with respect to the camera coordinate frame, will project onto the image plane with coordinates $\mathbf{p} = [u, v]^T$, given by

$$\begin{bmatrix} u \\ v \end{bmatrix} = A\,[RT]\,M \tag{4.1}$$

where **A** is the camera intrinsic (ie. internal) parameters:

$$\mathbf{A} = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

and **R,T** corresponds to the pose of the camera referenced to the 3D object frame where **R** is the rotation matrix, and **T** is the translation vector,

$$\mathbf{T} = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T \tag{4.3}$$

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{4.4}$$

Since the camera sensor can only provide a 2D reference for the target object, depth information must be obtained by other means to complete the cameras pose with respect to an object. For this work, it is accomplished by mounting an ultrasonic sensor next to the camera to provide distance measurement. The sensor reference frame **S**, has its origin located at the end of the sensor housing. Therefore, when an object is imaged, the sensor performs a distance measurement and the relationship between the camera frame **C**, and the sensor frame **S** (Figure 4.4) transforms the distance measurement to provide object depth information, and thus completing the reference of the object with respect to the camera frame.

Thus, to perform visual servoing, the remaining transformation required is between the camera frame **C**, and the boom-tip frame **E**. This transformation completes the mapping from image units (pixels) to real world units (mm). A summary of

the required transformations to reference a object point **P** from the camera frame **C**, to the boom-tip frame **E**, is given by the following transformation equation:

$$^{E}\mathbf{P} = {}^{E}\mathbf{X}_{C}\ {}^{C}\mathbf{X}_{S}\ {}^{C}\mathbf{X}_{I}\ {}^{I}\mathbf{P} \tag{4.5}$$

where **X** represents the transformation between reference frames.

However, before the required transformations between the different reference frames can be applied, various parameters must be determined, such as the camera parameters discussed previously. These parameters are determined by performing two calibrations: a camera and a hand-eye calibration. The following will discuss these calibrations outlining the required parameters calculated for each.

4.3.1 Vision System Calibrations

Using a camera/lens assembly attached to the end of a robot's end-effector, the pose of the drill-hole with respect to the camera frame must be determined, and then mapped to boom-tip reference frame to move the boom-tip to the drill-hole. To accomplish this overall mapping, a camera calibration and hand-eye calibration must be performed. The camera calibration is required to calculate the camera model parameters and the location of the hole with respect to the camera frame; while the hand-eye calibration is required to determine the mapping from robot boom-tip frame to camera frame. Once both calibrations are determined, visual servoing the end-effector to a drill-hole for loading with emulsion can be achieved.

4.3.1.1 Camera Calibration

Camera calibration in the context of three-dimensional (3D) machine vision is the process of determining the internal camera geometric and optical characteristics (intrinsic parameters) and/or the 3D position and orientation of the camera frame relative to a certain world coordinate system (extrinsic parameters) [32]. For this application, a camera calibration is required in order to infer the 3D information using computer image coordinates $[u, v]$, from a moving camera held by a robotic boom relative to the target world coordinate system. As a result, both intrinsic and extrinsic camera parameters must be calculated.

Camera Calibration Parameters

The parameters for the transformation from 3D object coordinates in the camera coordinate system to computer image coordinates are called the intrinsic parameters. These parameters are specific to each camera and lens setting. Changing either the camera type, lens, or lens setting will require a new calibration to be performed since the parameters will be different in each case. The intrinsic parameters are:

- f_x, f_y : effective focal length; the distance from the optical center to the image plane in the x and y directions.
- p_x, p_y : principal point; the intersection of the optical axis with the image plane.
- k_1, k_2, p_1, p_2 : lens distortion coefficients.

Once the intrinsic parameters are determined, the extrinsic camera parameters [**R**,**T**] which describe the pose of the camera system in the 3D world frame can then be calculated.

The camera geometry model with perspective projection and radial lens distortion is shown in Figure 4.6. Point $P(x_w, y_w, z_w)$ is the 3D coordinate of the object point in the 3D world coordinate system. Point $P(x, y, z)$ is the same object point relative to the 3D camera coordinate system, which is centered at point O_c, the optical center, with z-axis the same as the optical axis. (u_i, v_i) is the image coordinate system with origin at O_i and parallel to the cameras x and y axes. f is the effective focal length which is the distance between the front image plane and the optical center. Point $P_u(u_u, v_u)$ is the image coordinate of the object point $P(x, y, z)$ projected on the image frame if a perfect pin-hole camera model is used [32]. However due to lens distortion, primarily radial, the object point $P(x, y, z)$ is located at the distorted image coordinate $P_d(u_d, v_d)$. Due to this error in projection resulting from lens distortion, the distortion parameters must be determined to compensate for this offset when calibrating the camera. This is particularly critical for this work, since a wide angle lens is required for this application which exhibits considerable lens distortion that must be handled.

Extensive research has been performed to develop an effective and efficient means of performing camera calibration for 3D computer vision. The classical approach [33] originating from the field of photogrammetry performs a calibration by observing a calibration object whose geometry in 3-D space is known with very good precision. The calibration object usually consists of two or three planes that are mutually orthogonal. This approach requires sophisticated precision equipment and an elaborate setup that can be rather expensive. Other techniques employ a

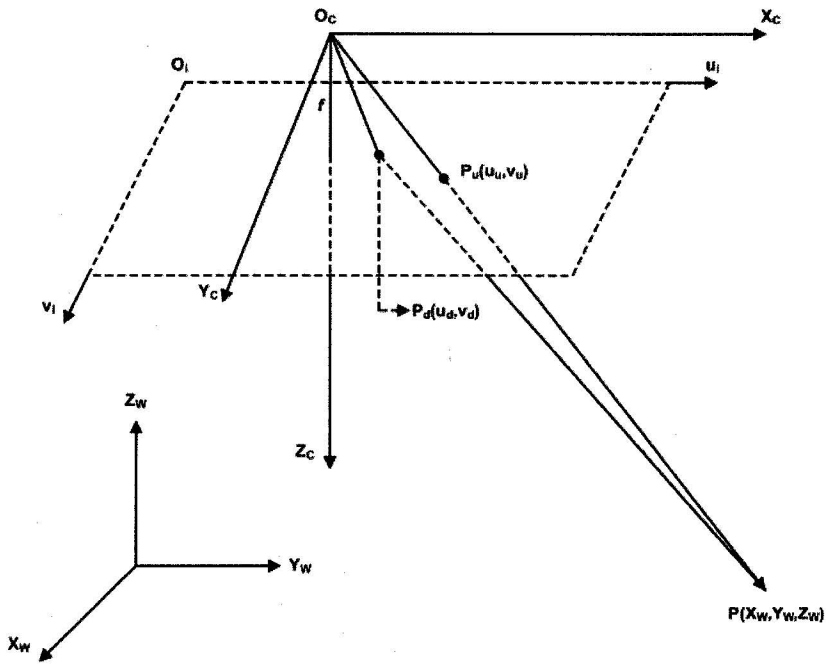


Figure 4.6: Camera geometry with perspective projection and radial lens distortion.

means of self-calibration which do not use a calibration object [34]. By moving a camera in a static scene, the rigidity of the scene permits using image information to identify correspondence between images (at least three) to allow for the reconstruction of a 3-D structure. This technique requires estimating many of the critical calibration parameters, and thus does not provide reliable results. Recent work has focused on developing flexible and robust methods that use off the shelf cameras and equipment, and can be performed by individuals who have minimal experience in computer vision [32].

The method proposed by Zhang [35] was developed with the aforementioned considerations in mind. The technique only requires a camera to observe a planar pattern or object from several views (at least two). The pattern is of known geometry and should be printed at high resolution using a laser printer and attached

to a reasonable planar surface such as plexiglass or a hard book cover. Either the camera or the planar pattern can be moved since the motion need not be known. For each view, the points on the object or model plane are projected onto the image plane and the homography matrix for all points in the series of images is found. The procedure developed is outline in Algorithm 4 [36].

Algorithm 4 Camera calibration

1: calibration object images, $I = \{I_1, I_2, \ldots, I_n\}$
2: calibration points per image, p
3: **for** $g = 1$ to I_n **do**
4: find homography (ie. chessboard corners) for all points, p
5: initialize intrinsic parameters, $\mathbf{A}$; distortion is set to 0
6: find extrinsic parameters for each image of the object or pattern, [$\mathbf{R}$,$\mathbf{T}$]
7: **end for**
8: make main optimization by minimizing error of projection points with all parameters

Results generated using both computer simulation and real world data have proved the effectiveness of this technique for camera calibration. A recent study conducted by Sun and Cooperstock [37] compared the conventional world-reference based (3-D object) approach to that proposed by Zhang. Their investigation identified that the conventional approach does achieve higher accuracy when trained on data with low measurement error. However, the calibration requires expensive equipment, and a time-consuming measurement process that is not always easily afforded or accessible. While the method proposed by Zhang, requires neither a laborious measuring task nor specialized equipment, and can produce better results with less accurate data. Improving the sensitivity of Zhang's algorithm to pixel noise can simply be overcome by increasing the number of grid corner points on the chessboard pattern. This study demonstrated that Zhang's planar approach is

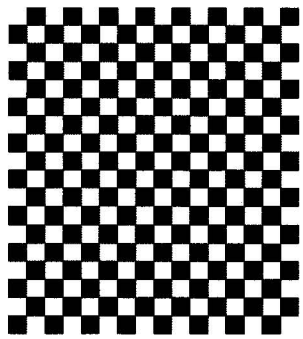
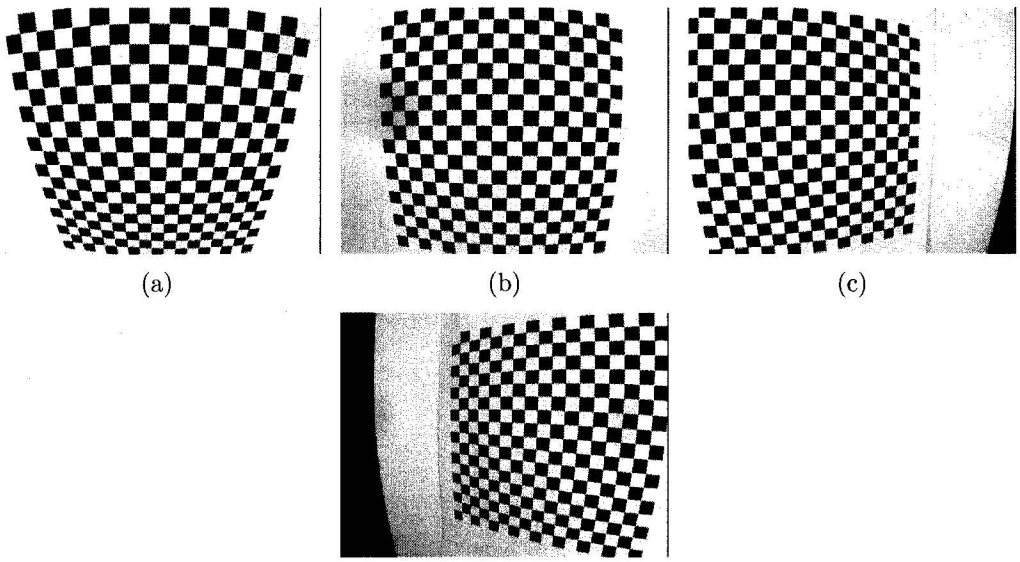


Figure 4.7: Chessboard pattern used for camera calibration.

both flexible and suitable for performing calibrations in dynamic environments making it ideally suitable for this application. This approach developed by Zhang has been incorporated into OpenCV [36], is free for use, and thus was used to perform the required camera calibration for this work.

The camera to be calibrated was a Pulnix TMC-7DSP 1/2" CCD progressive scan camera with a 3.5mm wide angle lens, and input to a Matrox Meteor-II/4MB PCI frame grabber. The image resolution was 640 × 480 pixels. The calibration object model was a chessboard pattern of 16 × 18 squares, with a square size of 10.159mm each. The chessboard pattern was printed at high resolution using a laser printer and attached to a 1/8" white aluminum plate as shown in Figure 4.7. The *CamCal* camera calibration software (GUI) application, as shown in Figure 4.9, was developed and included the OpenCV camera calibration routines. The camera was mounted rigidly to the end-effector of the robotic boom as illustrated in Figure 4.1 and four images of the chessboard object at different orientations were acquired (Figure 4.8). Each image was then processed to locate the internal object points (ie. chessboard corners) used to generate the homography matrix.

After the object points are extracted as shown in Figure 4.9, the user would deter-



(a) (b) (c)

(d)

Figure 4.8: Example of four chessboard images used to perform camera calibration. Images are taken at different distances and orientations.

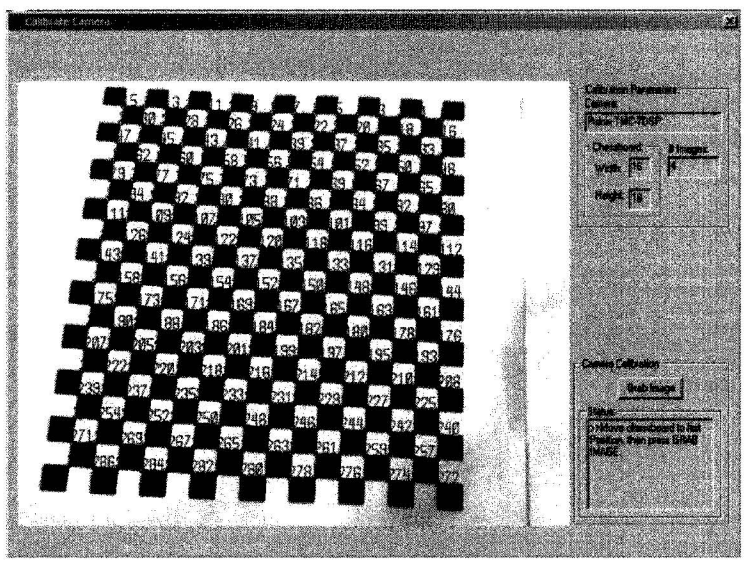


Figure 4.9: Camera calibration software GUI illustrating chessboard object corners extracted.

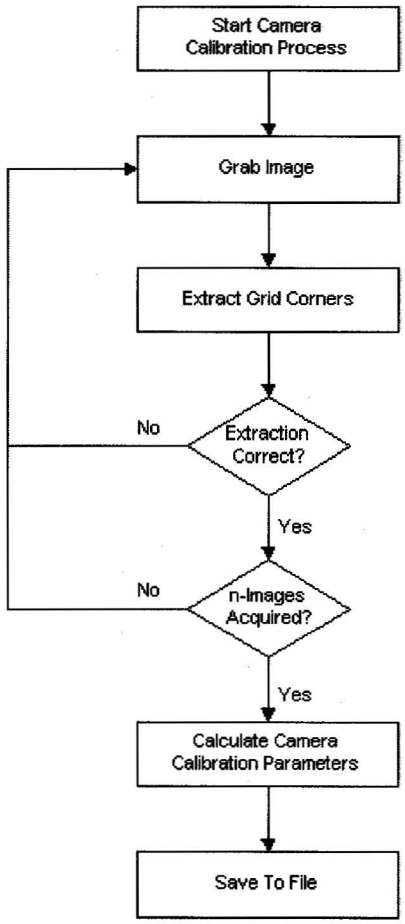


Figure 4.10: Camera calibration process.

mine if they (i.e. chessboard internal corners) have been labeled correctly to accept the image for calibration. If the labeling was incorrect, the extracted object points and image would be discarded and the process would continue until at least four images were captured with correct labeling. The process of performing a camera calibration is illustrated in Figure 4.10.

Although only two images are sufficient to perform a calibration, more images are suggested to improve the accuracy of the calibration parameters. Therefore, four images with the chessboard corners extracted and correctly identified are used.

Table 4.1: Camera calibration parameters calculated. Calibration used a chessboard size of 16 × 18, and a square size 10.0159mm (calc.).

Parameter	Test 1	Test 2	Test 3
FocalLength Xpixels	347.2757	353.9817	366.2279
FocalLength Ypixels	346.3365	353.2321	366.1256
PrincPointX	321.3295	322.6051	327.3821
PrincPointY	251.6542	250.4295	256.7823
CameraMatrix0	347.2757	353.9817	366.2279
CameraMatrix1	0.0000	0.0000	0.0000
CameraMatrix2	321.3295	322.6051	327.3821
CameraMatrix3	0.0000	0.0000	0.0000
CameraMatrix4	346.3365	353.2321	366.1256
CameraMatrix5	251.6542	250.4295	256.7823
CameraMatrix6	0.0000	0.0000	0.0000
CameraMatrix7	0.0000	0.0000	0.0000
CameraMatrix8	1.0000	1.0000	1.0000
Distortion1	-1.9551e-01	-2.2248e-01	-2.0364e-01
Distortion2	5.0969e-02	8.4222e-02	4.956e-02
Distortion3	8.4428e-04	1.1911e-03	9.1433e-05
Distortion4	2.9581e-03	2.8967e-03	-1.3251e-03

For every image view, the extracted internal corner points on the model plane (ie. chessboard) and their projections onto the image are passed to the calibration routine to calculate the camera calibration parameters. The calibration parameters calculated, are automatically saved to the file *ELAPCalibration.cal* to be used for image distortion correction and visual servoing. The camera calibration parameters calculated are shown in Table 4.1 for three calibrations performed.

Once a camera calibration is performed, it is then reviewed to determine the quality of the results. The quality of calibration parameters calculated is dependent on how well the calibration grid corners were segmented using the OpenCV calibration function `FindChessboardCorners()`. Poor or uneven lighting may cause blurring at the grid corners, which would result in errors for their extracted locations. To evaluate the calibration performed, the calibration parameters are applied to an

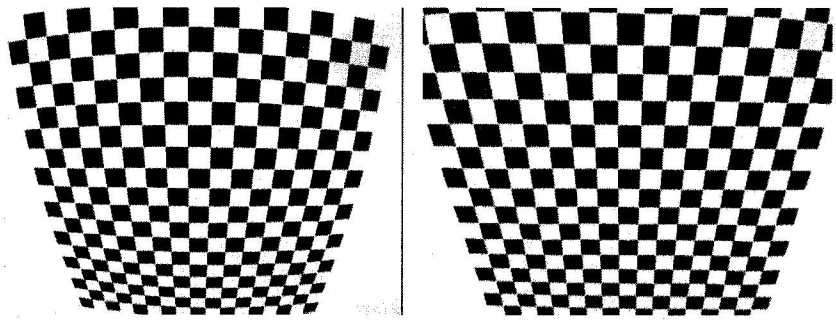


Figure 4.11: Correcting image distortion after camera calibration parameters determined. (a) is the original image of the chessboard grid, and (b) shows the corrected image after distortion parameters are applied.

image as illustrated in Figure 4.11. The original chessboard image (Figure 4.11(a)) appears curved toward the image border and is a direct result of radial lens distortion. The image shown in Figure 4.11(b) is the original image after calibration is applied. The lens distortion has been removed and the chessboard has regained its orthogonal characteristics. Depending on the quality of the calibration parameters, this distortion may not be sufficiently estimated, and thus the calibration must be repeated until the results are satisfactory. The calibration parameters determined for each of the tests provided in Table 4.1 were evaluated to determine the best calibration. The first distortion coefficient for each test resulted in a negative value. The negative represents barrel-type distortion which was clearly observed in all images. However, once the distortion parameters for tests 2 and 3 were applied to the calibration images, the lens distortion was not fully corrected as compared to that of test 1. Thus as a result, the parameters from Calibration 1 in Table 4.1 provided the best overall results and were used for this work.

4.3.1.2 Hand-Eye Calibration

In order to use a gripper mounted sensor such as a camera, for a robotic task, the relative 3D position and orientation between the camera and the gripper must be known. The problem of determining this relationship is referred to as the *hand-eye* calibration problem [38]. More specifically, its the relative rotation and translation between the two coordinates frames, one centered at the camera coordinate frame, **C**, and the other at the end-effector, **E**, as shown in Figure 4.4. The end-effector coordinate frame is typically centered on the last link of the robot manipulator (end effector/boom-tip frame for this work).

Direct measurements between coordinate frames are difficult as there may be obstacles to obstruct the measurement path, the points of interests may be inside equipment and therefore unreachable, or the coordinate frames may differ in their orientation. The measurement path can be obstructed by the geometry of the sensor or robot, the sensor mount, cables, etc. The camera frame is unreachable since it is inside the camera housing, however, the robotic end-effector frame has been provided by the robotic arm design team and is located at the center of the end-effector hose guide, mounted at the end of the robotic boom, which is accessible for measurement [39]. Rather than direct measurement, this relationship can be determined by displacing the robot and observing the resulting motion in the image frame.

The hand-eye problem has been formally described as the solution to the homogeneous transform equation of the form:

$$\mathbf{AX} = \mathbf{XB} \tag{4.6}$$

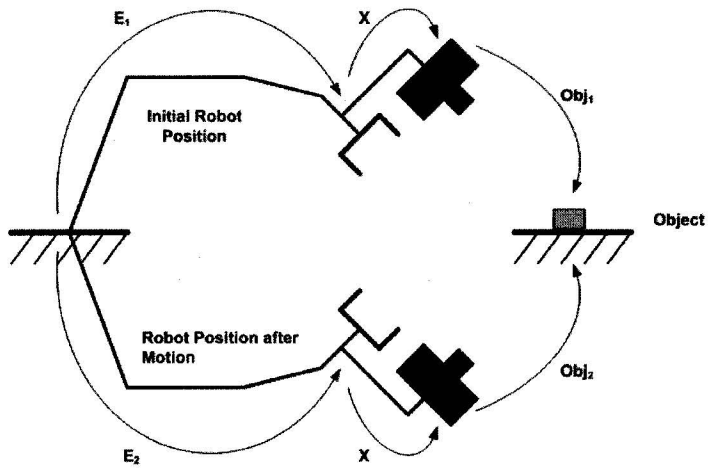


Figure 4.12: Finding the mounting position of a camera with respect to the robotic arm end-effector frame by solving a homogeneous transform equation of the form $\mathbf{AX} = \mathbf{XB}$, where $\mathbf{A}$ is the robot motion, $\mathbf{B}$ is the resulting camera motion, $\mathbf{O}_{bj}$ is the object referenced to the camera frame, and $\mathbf{X}$ is the camera mounting position.

where $\mathbf{A}$ is the known change in the end-effector position, $\mathbf{B}$ is the known resulting sensor displacement, and $\mathbf{X}$ is unknown transformation of the camera relative to the end-effector frame as shown in Figure 4.12.

Considerable research has been performed to develop a procedure for hand-eye calibration with much success, each with varying observations and assumptions to formulate a solution. Practically all methods attempt to solve a homogeneous transform equation of the form in Equation 4.6. Tsai and Lenz [40], Shiu and Ahmad [39], Chou and Kamel [41], Chen [42], and Wang [43] all use a closed form solution to solve the system of equations developed. Tsai and Lenz, and Shiu and Ahmad decouple the hand-eye calibration from the robot model calibration and use linear rather than high dimensional non-linear optimization methods which require a good initial guess and accurate data for convergence as proposed by Dornaika and Horaud [38]. They formalize the system of homogeneous equations of the form $\mathbf{AX} = \mathbf{ZB}$, where $\mathbf{X}$ is the end-effector to camera transformation as before, and $\mathbf{Z}$

is the robot to world transformation. This technique does not require the camera intrinsic and extrinsic calibration parameters to be known, but does require a good initial guess for a solution. Another self-viewing hand-eye calibration is proposed by Meinicke and Zhang [44] where the camera directly observes the end-effector. Being purely vision based, this calibration method is not restricted by the precision in the robots kinematics, and can provide an on-line updated calibration to compensate for any disturbance in the hand-eye geometry. However, the end-effector calibration points for this setup may be few in number, collinear or concentrated in a small region, or not easily segmented from the image accurately.

For this work, the hand-eye calibration was performed using a variation of the method outlined by Shiu and Ahmad [39]. Shiu and Ahmad formulate the homogeneous transform equation using Figure 4.12. By moving the robot from position $\mathbf{E}_1$ to $\mathbf{E}_2$, and the position of the fixed object relative to the camera frame is found to be $\mathbf{O}_{bj1}$ and $\mathbf{O}_{bj2}$ respectively; then the following equation can be obtained:

$$\mathbf{E}_2^{-1}\mathbf{E}_1\mathbf{X} = \mathbf{X}\mathbf{O}_{bj2}\mathbf{O}_{bj1}^{-1} \tag{4.7}$$

where $\mathbf{E}_2^{-1}\mathbf{E}_1$ is the relative motion made by the robot and which is denoted by $\mathbf{A}$ as in Equation 4.6, thus

$$\mathbf{A} = \mathbf{E}_{-1}^{2}\mathbf{E}_1 \tag{4.8}$$

Similarly, $\mathbf{O}_{bj2}\mathbf{O}_{bj1}^{-1}$ can be denoted by $\mathbf{B}$, again as in Equation 4.6 which is the relative motion of the camera frame.

$$\mathbf{B} = \mathbf{O}_{bj2}\mathbf{O}_{bj1}^{-1} \tag{4.9}$$

The transform matrices $\mathbf{A}$ and $\mathbf{B}$ are known, since $\mathbf{E}_1$ and $\mathbf{E}_2$ can be obtained directly from the robot controller to provide the 3D points in world coordinates; and $\mathbf{O}_{bj1}$ and $\mathbf{O}_{bj2}$ can be found relative to the camera frame after a camera calibration has been performed. In addition, the robot manipulator must possess enough degrees-of-freedom (DOF) to be able to rotate the camera around two different axes while keeping the camera focused on the calibration object.

However, a complete hand-eye calibration was not performed as a part of this project. It was decided that by mounting the camera and gripper in an attempt to align their coordinate frames (i.e. no rotation, only translation) would reduce the complexity of the calibration without jeopardizing the usefulness and quality of the solution. Thus, this specification reduced the hand-eye calibration to one equation:

$$\mathbf{X} = {}^{E}\mathbf{T}_{O}\ \mathbf{O}_{bj1}^{-1} \tag{4.10}$$

where $\mathbf{T}$ represents the translation from the object frame relative to the end-effector frame $\mathbf{E}$, and $\mathbf{O}_{bj}$ is the object projected onto the image frame measured relative to the camera frame. Again, all frames were aligned as closely as possible, including the object frame, to reduce the requirement of knowing the rotation component. The hand-eye transformation $\mathbf{X}$, can then be found once the other transformations have been determined.

4.3.1.3 Hand-Eye Calibration Setup

The hand-eye calibration setup involved ridgedly mounting the same camera and lens configuration used for the camera calibration to the gripper of a RT200 robot

arm, as illustrated in Figure 4.13. The camera and lens were aligned with the gripper frame to minimize the rotation component of the calibration. A chessboard calibration object of size 8×10 squares measuring 20mm each, was aligned with the gripper frame to minimize the rotation component between the respective reference frames.

The process of performing a hand-eye calibration is illustrated in Figure 4.15 using the *CamCal* software. The *CamCal*, includes a software component developed to determine the hand-eye calibration by capturing images of the chessboard calibration object, as shown in Figure 4.14. For each image captured, the camera calibration parameters previously calculated are applied to compensate for lens distortion, and subsequently undistort the image. The extracted chessboard corner points are then used to calculate the extrinsic parameters for each image. The extrinsic parameters provide the object transformation $\mathbf{O}_{bj}^{-1}$, relative to the camera frame for the corresponding gripper position. Once three chessboard object images are acquired (one image would suffice, however, three images are captured and the average result is used), the object frame translation ${}^{E}\mathbf{T}_{O}$ relative to the gripper frame is measured and input to the software. Using the $\mathbf{O}_{bj}^{-1}$ and ${}^{E}\mathbf{T}_{O}$ transformations determined, the hand-eye calibration was calculated using Equation 4.10.

To verify this approach for calculating the hand-eye calibration, the calibration object was moved to several locations with respect to the RT200 robot's gripper frame. At each location, alignment was maintained between the object and gripper reference frames and the hand-eye calibration was calculated. The results were validated by taking an approximate measurement of the translation between the gripper and camera frames, noting the gripper as the reference frame. The measured and calculated translations for the three object locations are listed in Table

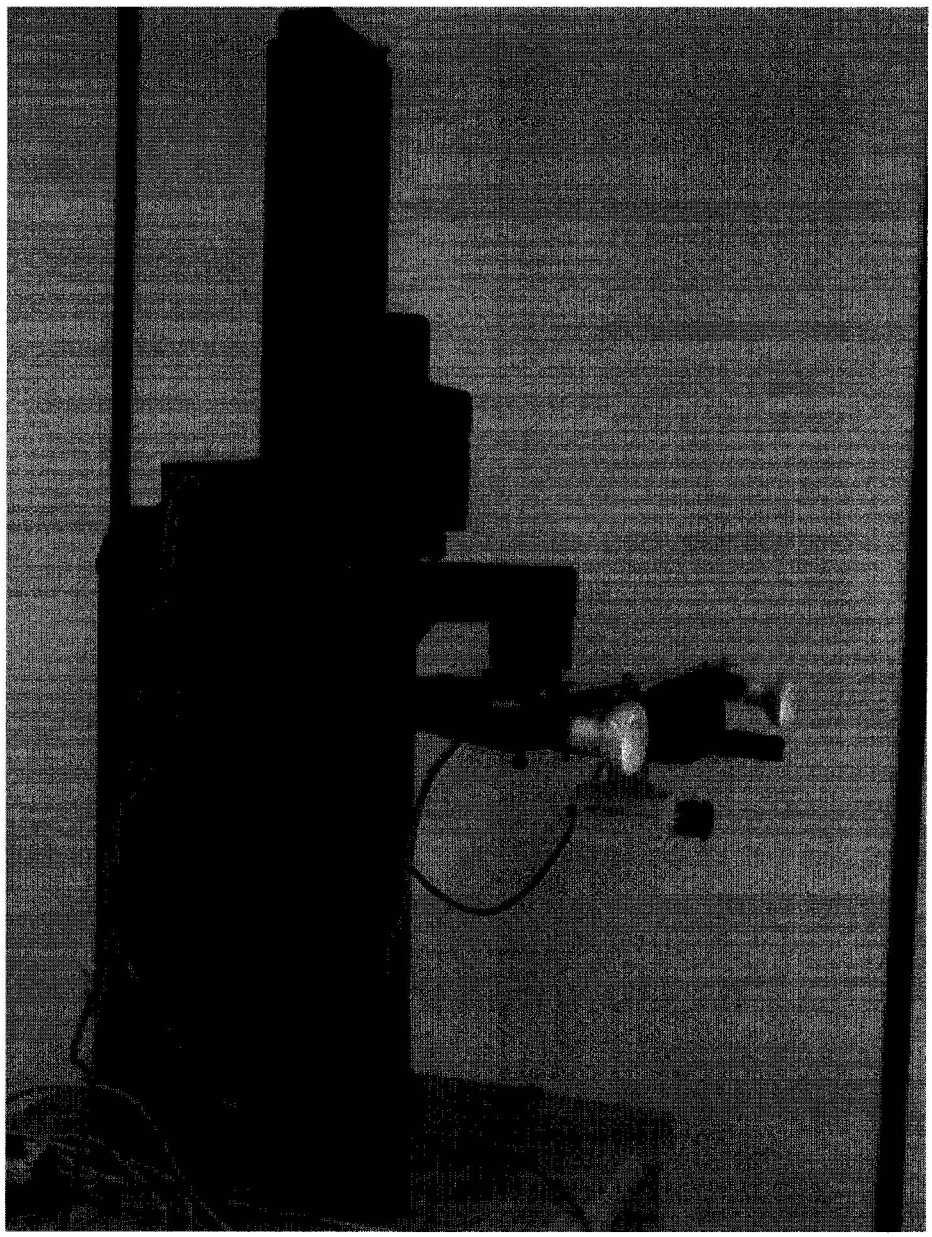


Figure 4.13: Setup for hand-eye calibration using the RT200 robot arm. Equipment setup includes: a RT200 robot arm, a gripper mounted camera, and diffuse lighting using two frosted 40W bulbs as shown.

Table 4.2: Hand-Eye calibration results for the RT200 robot arm.

Translation	Measured (mm)	Location 1 (mm)	Location 2 (mm)	Location 3 (mm)
X	-165	-162.54	-162.2	-161.66
Y	-315	-317.72	-309.4	-314.85
Z	-80	-92.59	-91.10	-78.8

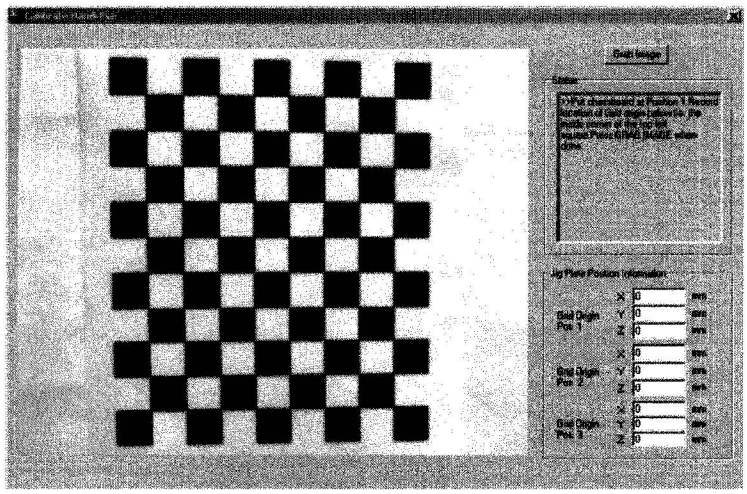


Figure 4.14: Hand-Eye calibration GUI window.

4.2 and represent the hand-eye calibration parameter values for this configuration. As shown in Table 4.2, the calculated results for the hand-eye calibration using the RT200 robot arm and the gripper mounted camera were very close to the expected values, thus verifying the approach used.

The same approach was used to perform the hand-eye calibration between the camera and robotic boom end-effector for the ELAP vision system, as shown in Figure 4.16. The calibration object was aligned with the end-effector coordinate frame and the translation measurements between them were recorded. The *CamCal* software was then used to capture images of the calibration object, and subsequently calculate the hand-eye calibration using the input object to end-effector translation measurements. Results of the hand-eye calibration were output to the same cal-

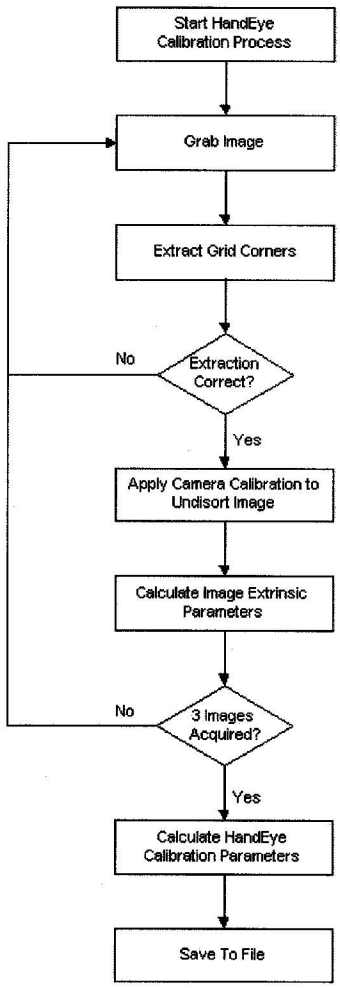


Figure 4.15: Hand-Eye calibration process.

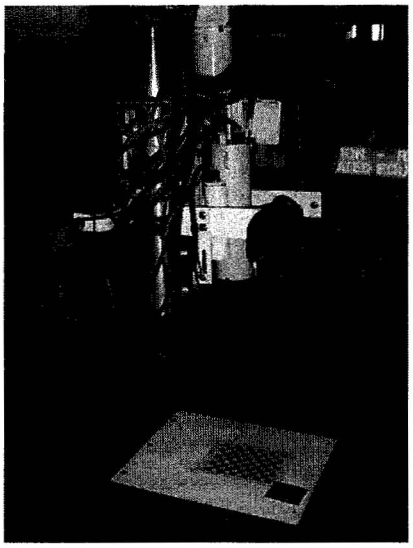


Figure 4.16: Hand-Eye calibration setup for the full-scale ELAP robotic boom assembly. Equipment shown includes: end-effector hose guide, tracker lights for drill-hole lighting, camera enclosure, distance sensor enclosure, and calibration jig temporarily mounted to end-effector.

Table 4.3: Hand-Eye calibration results for the translation component for the full-scale ELAP unit.

Parameter	Measured (mm)	Test 1 (mm)	Test 2 (mm)
X	-65	-63.50	-69.66
Y	-202	-206.375	-209.55
Z	-163	-161.925	-163.32

ibration file, *ELAPCalibration.cal*, containing the camera calibration parameters. The ELAP hand-eye calibration results are provided in Table 4.3.

The hand-eye parameters are then used during visual servoing to transform object (i.e. drill-hole) measurements in the camera frame to the gripper frame for guidance purposes. The effects of not including the rotation component for the coordinate transformations is investigated during visual servoing, which is discussed in Section 4.5.

4.4 Control Scheme Selection

Visual servoing has proved to be a highly effective means to control a robot manipulator through the use of visual data. Visual servo methods have classically been of the look-and-move type which can be divided into two approaches: position-based, and image-based control systems. In position-based visual servoing, features are extracted from the image and used in conjunction with a geometric target model and the camera model (ie. camera calibration) to estimate the pose of the target with respect to the camera. Using the feature values, an error between the current and desired pose of the robot is estimated in pose space. However, in image-based visual servoing the control values are computed using the image features directly [31]. The control scheme for each servoing approach is illustrated in Figure 4.17 (a) and (b). For this application, an image-based Look-and-Move approach is used, Figure 4.17 (c), whereby a gripper-mounted sensor (i.e. a camera) is used to visually guide a robot gripper to move to different locations within its workspace. To do this, both the relationship between the gripper and camera (hand-eye calibration), and between the object and camera (camera calibration) must be known, and subsequently applied to each image captured. These calibrations and their corresponding parameters were discussed in Section 4.3.1.

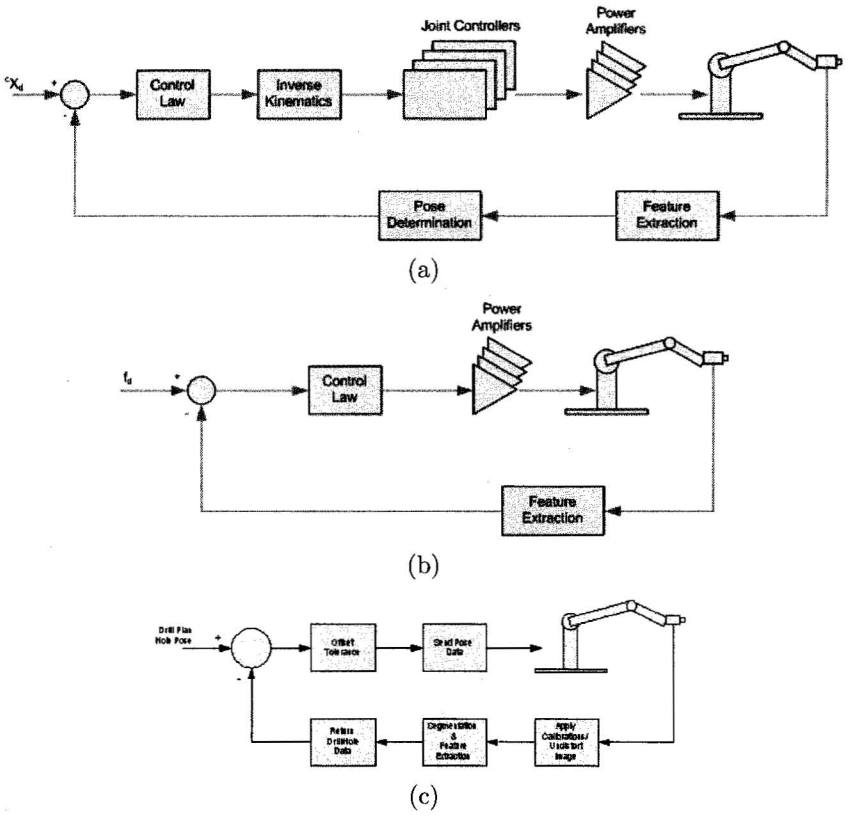


Figure 4.17: Visual servoing control schemes. (a) Position-based visual servoing (PBVS), (b) Image-based visual servoing (IBVS), and (c) Proposed ELAP Look-and-Move visual servoing.

4.5 Vision System Implementation

As previously stated, the second objective of the ELAP vision system is to visual servo the robotic boom to each drill-hole for loading with emulsion. To achieve this objective, a review of the existing manual process of loading drill-holes was performed, and the process was then incorporated into the LPS as the main high level logic controller as illustrated in Figure 4.18. The implementation of the LPS controller into software was a separate development project and is not included as a part of this thesis. The next section provides an overview of the LPS control process, and outlines the vision system implementation for visual servoing the robotic boom to each drill-hole for loading with emulsion.

4.5.1 LPS Control Process

During drift development, holes are drilled into the rock face, the back, or the floor according to the specified drill plan diagram. The drill plan diagram is produced by mine personnel and is based on the outline of the ore-body. Holes are drilled at specified locations to develop the ore-body, and maximize the amount of raw material produced from each blast. The drilling machine is setup in the area to be drilled, and information such as drill-hole depth, size, and hole collar coordinates are recorded.

The LPS begins by selecting a drill-hole from the drill plan diagram recorded during the drilling process. After drilling, the coordinates of each drill-hole is known only in mine space, and thus are transformed to end-effector space which is controlled by the ELAP Robotic Boom Controller (RBC). Once transformed, an approach pose

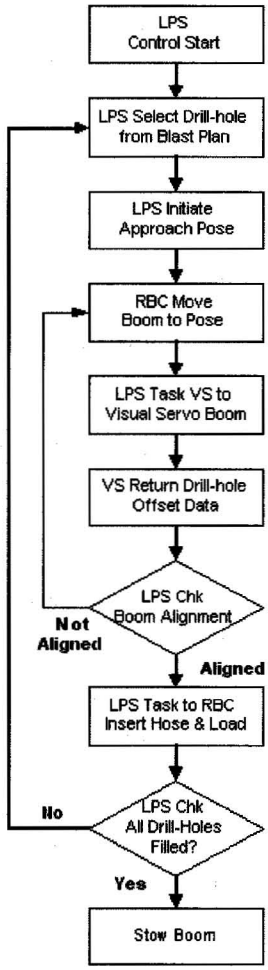


Figure 4.18: LPS control procedure for loading drill-holes for a specified drill plan.

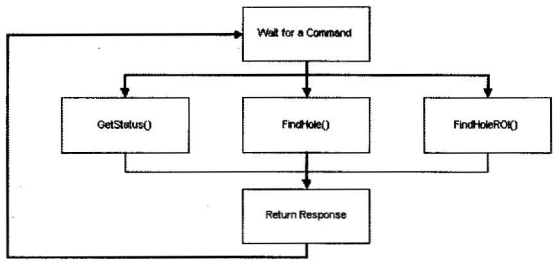


Figure 4.19: VS communication process as tasked by LPS.

is calculated which positions the robotic boom such that the drill-hole is in view of the end-effector mounted camera. This initial approach pose reduces the search area for locating the target drill-hole in the image, and thus enables the use of a simplified segmentation algorithm as previously discussed. After the camera/boom assembly has been positioned using the approach pose, the visual servoing process first aligns the camera with the drill-hole and then visual servos the hose guide to the drill-hole. After all drill-holes have been loaded, the boom is stowed to prepare for blasting.

4.5.2 VS Commands

The VS software is a server-based command line application that continuously waits for a command to be sent from the LPS controller (the VS only communicates with the LPS sub-system) over Ethernet communications. The commands issued by the LPS to the VS are `GetStatus`, `FindHole`, and `FindHoleROI` as shown in Figure 4.19. The `GetStatus` command is sent to inquire about the operational state of the VS which performs diagnostic checking (ie. valid video stream, sensor communications, and temperature) to ensure the VS is operating normally. `FindHole` and `FindHoleROI` are the main LPS commands transmitted to the VS to both identify

and visual servo to drill-holes after the approach pose is complete. Figure 4.20 outlines the process details for these three commands.

4.5.3 Visual Servoing Process

With reference to Figure 4.18, after the LPS positions the camera at the approach pose for a drill-hole, the LPS begins by first sending the `FindHole` command to the VS. As shown in Figure 4.20, upon receiving this command, an image is acquired from the video stream and the camera calibration parameters previously calculated are applied to correct for lens distortion. Image processing algorithms developed are then used to segment drill-holes from the image, and classify each using the corresponding feature data. Figure 4.22 shows the approach pose image acquired by the `FindHole` command with found drill-holes indicated by a cross-hair.

For visual servoing to occur, each drill-hole found must be referenced to the camera frame. The cameras center of projection intersects the image frame at the principal point, and thus aligning this point with drill-hole image coordinates (u, v), effectively aligns the camera with the drill-hole. This is accomplished by calculating a delta or offset from the drill-hole (u, v) relative to the principal point, (p_x, p_y). However, this delta move must be calculated in real world units (mm) for positioning of the end-effector. This transformation is illustrated in Figure 4.23, and the delta move in pixels (dx, dy) was calculated in real world units (mm) using Equations 4.11 and 4.12.

$$dx = \frac{u - p_x}{f_x} Z \tag{4.11}$$

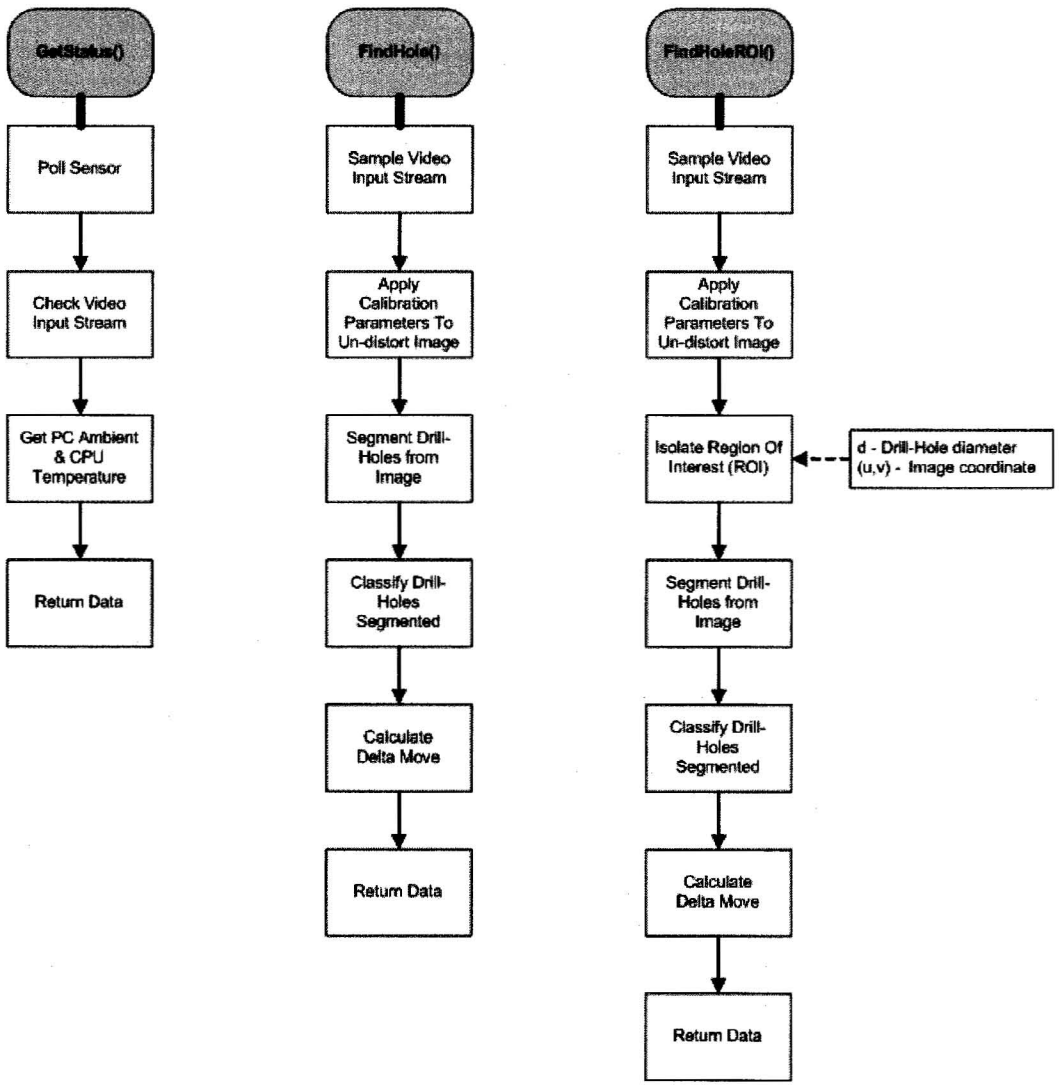


Figure 4.20: VS process commands issued by LPS.

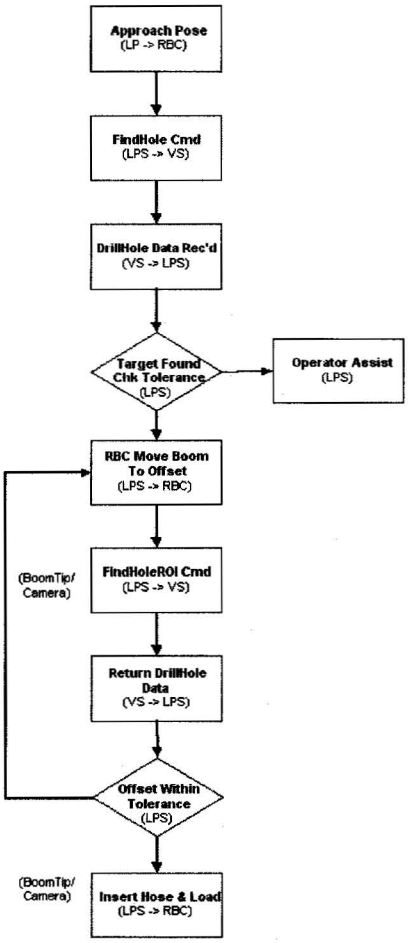
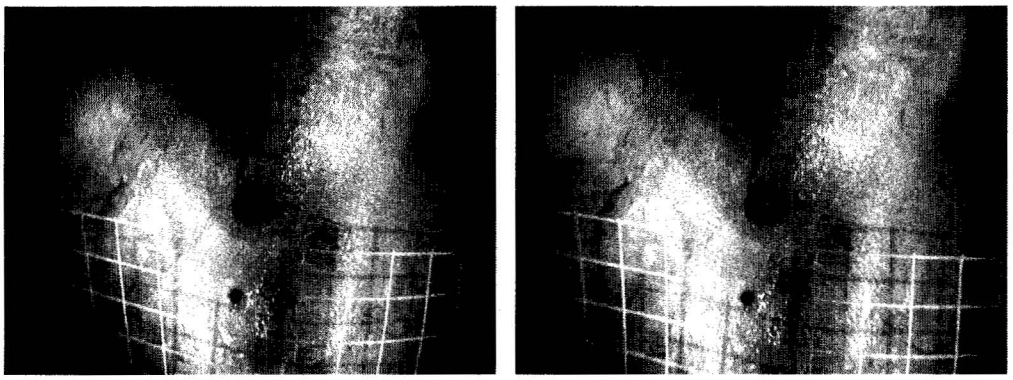


Figure 4.21: Visual servoing process developed for ELAP system. Brackets indicate communication between sub-systems (i.e. $sender \rightarrow receiver$).



(a) Actual rock face image captured from video stream.

(b) Corresponding image with found drill-holes indicated.

Figure 4.22: Actual approach pose image and corresponding found drill-hole after image processing.

$$dy = \frac{v - p_y}{f_y} Z \tag{4.12}$$

dx, dy : required delta move in mm.

u, v : image coordinates of the object (i.e. drill-hole) that the camera is to be aligned with in pixels.

p_x, p_y : coordinates of the principal point in the image frame (i.e. the intersection of the cameras optical axis with the image frame) in pixels. Calculated during camera calibration.

f_x, f_y : effective focal length calculated (i.e. the distance from the cameras center of projection to the principal point) in pixels. Calculated during camera calibration.

Z : real world distance referenced from the lens to the object in mm. Provided by the boom mounted ultrasonic distance sensor.

The effective focal length, f_x, f_y, and the principal point coordinates p_x, p_y were calculated during camera calibration. The Z distance is measured using the ultrasonic sensor and is referenced to the camera frame. These parameters are loaded from the calibration file *ELAPCalibration.cal*, a sample of which is provided in Appendix C.

After the delta move is calculated, the `FindHole` command is completed, and the VS returns the required information for each potential drill-hole identified in the image to the LPS for processing, specifically: the delta move (x, y, z) in mm referenced to the camera frame, and the cameras principal point (p_x, p_y). The LPS then compares the location of each potential drill-hole using the delta move information, to the expected location as specified in the drill plan. The drill-hole that is within a specified tolerance of the expected location is selected for visual servoing. If no drill-holes are within the tolerance, operator assistance is requested to locate and visual servo to the drill-hole.

Once a drill-hole has been selected for visual servoing, the LPS begins the process of aligning the cameras reference frame (ie. the principle point) with the drill-hole image center (u, v) using the corresponding delta move information provided. The LPS sends a move command to the RBC to offset its current location in end-effector space using the delta move information. Once the robotic boom has reached its new location, the LPS sends the `FindHoleROI` command to begin visual servoing. The `FindHoleROI` command is very similar to the `FindHole` command as shown in Figure 4.20. However, the `FindHoleROI` command is used to take advantage of the information found after issuing the `FindHole` command by focusing the attention to a region of interest (ROI) in the image where the drill-hole is expected to be located after performing the delta move. This is accomplished by passing the expected

(u, v) coordinate of the drill-hole along with the drill-hole diameter, as command arguments. The drill-hole diameter is recorded in the drill plan and is a standard size depending on the type of holes drilled; production (4-inch) or development (2-inch) holes. The expected (u, v) coordinate of the drill-hole can be referenced to either the camera frame or the boom-tip frame by passing either *CAMERA* or *BOOM-TIP* as the parameter for visual servoing. Visual servoing with respect to the camera will align the drill-hole with the cameras principle point (p_x, p_y), while visual servoing with respect to the boom-tip will align the boom-tip to the drill-hole, since the hand-eye calibration parameters are used to calculate the delta move image plane coordinates (u, v) corresponding to the boom-tip. Depending on the reference frame selected, the (u, v) coordinate locates the ROI or subimage center while the drill-hole diameter is used to determine its size (ie. width and height). By obtaining a distance measurement at the current position referenced to the camera frame, the size of the drill-hole in image pixels can be determined using Equations 4.13 and 4.14. The dimensions are increased by 10% to ensure that the drill-hole does not lie on the ROI subimage border.

$$ROI_w = \frac{d * f_x}{Z} \tag{4.13}$$

$$ROI_h = \frac{d * f_y}{Z} \tag{4.14}$$

ROI_w, ROI_h : delta move in pixels.

d : drill-hole diameter in mm.

f_x, f_y : focal length calculated (i.e. the distance from the cameras center of projection to the principal point) in pixels.

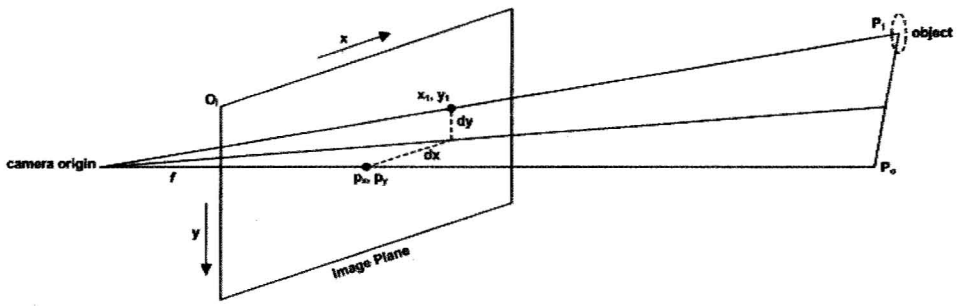


Figure 4.23: Delta move calculation for visual servoing.

Z : real world distance from the lens to the object in mm.

The ROI subimage then undergoes the same image processing as before with the `FindHole` command, to segment and classify any drill-holes in view, typically only one for a ROI subimage. The delta move to align the camera or boom-tip with the drill-hole, along with the (u, v) coordinate of the expected image location of the drill-hole after the delta move is performed, is returned to the LPS for processing. Visual servoing continues until it is determined by the LPS that the calculated delta move is within a specified tolerance. If the tolerance criteria is not met, subsequent `FindHoleROI` commands are sent to the VS until it is achieved. If the tolerance is met, the LPS terminates the visual servoing as the camera or boom-tip is now aligned with the drill-hole. The proposed ELAP visual servoing process is summarized in the flow chart shown in Figure 4.21.

4.5.4 Visual Servoing Results

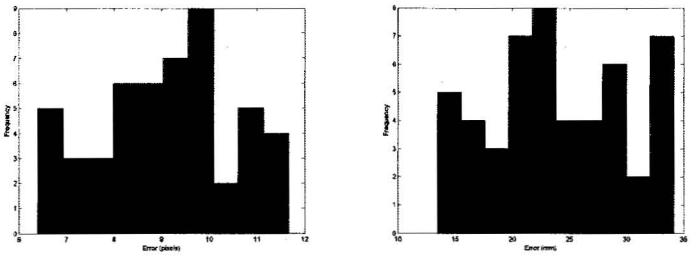
4.5.4.1 Lab Testing

To verify the visual servoing technique proposed, a similar setup for the hand-eye calibration was used consisting of the RT200 robot arm, an Opticom 1/2" CCD

(a) (b)

Figure 4.24: Visual servoing test setup. (a) shows the equipment used for testing (RT200 robotic arm, camera, and simulated underground rock face containing drill-holes), and (b) shows the VS and LPS software, and the current image grabbed and processed from the video stream.

color camera, a simulated underground rock face containing drill-holes, and versions of the VS, LPS, and RT200 control software installed on a desktop computer as shown in Figure 4.24. The RT200 robot base was positioned at a fixed location and maintained an offset distance of $Z = 1000mm$ from the simulated rock face. An approximate measurement of the locations (i.e. the hole collar) of five drill-holes in the simulated rock face was recorded with respect to the RT200 robot's base frame. The location information recorded was then used to provide the respective approach pose for each of the drill-holes. The LPS software then tasked the robot arm to position the camera at the approach pose for each drill-hole. The LPS then began the visual servoing process, by sending the required commands to the VS and robot arm controller software until the delta move tolerance was met, indicating that the camera is aligned with the drill-hole. The offset (i.e. error) of the cameras principal point with respect to the drill-hole center was measured and recorded once the camera was aligned. The visual servoing process of aligning the camera to each drill-hole continued for the remaining holes. The same process was also performed to test the visual servoing of the robot gripper to each of the five drill-holes. The visual servoing error for the gripper was measured from the gripper tip



(a) (b)

Figure 4.25: Visual servoing error (rms) using simulated rock face and RT200 robot for five drill-holes. (a) error measured when visual servoing the camera to each drill-hole, and (b) error measured when visual servoing the gripper to each drill-hole.

center to the center of the drill-hole. The visual servoing error (rms) recorded for both the camera and gripper frames is shown in Figure 4.25.

Based on the measured data, an approximate offset (i.e. to achieve alignment) of $\{dx = -9, dy = -5\}$ pixels was required and used to improve the camera alignment. This offset referenced to the camera frame, represented moving the camera left and upward to achieve alignment. However, adding this offset did not eliminate the visual servoing error completely. A small misalignment continued to remain that may be resultant from errors introduced during camera calibration, inaccurate segmentation of the drill-hole center during image segmentation, the considerable amount of 'slack' in the RT200 robot arm joints, or a combination thereof. Attempts to minimize this offset further by reducing the tolerance during visual servoing were unsuccessful, and only increased the number of delta move iterations to achieve the same alignment accuracy.

Similar performance was observed when visual servoing the RT200 robot arm gripper to each drill-hole, which required an offset of $\{dx = -19, dy = -13\}$ mm for

alignment. This error may be the result of the errors in the hand-eye calibration, particularly the absence of the rotational component, and the sources of error already mentioned. The testing was repeated 10 times for each simulated drill-hole, and produced similar results. Given that the results could be repeated with a small error present, the testing demonstrated that the proposed visual servoing approach was successful and it could be implemented as the design for the full scale ELAP system.

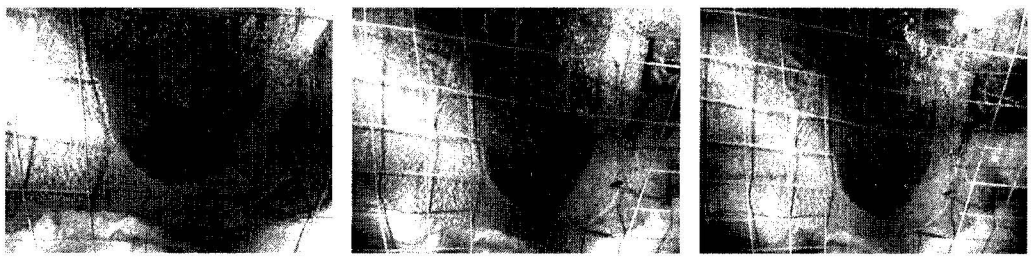
4.5.4.2 Full Scale System Testing

Full scale system testing using the ELAP robotic boom and VS components, were conducted in an underground research mine after the required ELAP subsystems were integrated and operational. Drill plan data recorded for a ring of drill-holes was used to test the VS, LPS, and RBC subsystems to verify the process control, image capture, drill-hole identification, and the visual servoing capability for the ELAP system prototype. The images in Figure 4.26 are a subset of an actual mission, and represents the images of a drill-hole that were captured and processed to first visual servo the camera and then the boom-tip for loading. This image set is also found on the CD-ROM located at the end of the document. The first image in Figure 4.26(a) represents the initial approach pose. This image was processed and three objects were identified as indicated by the cross and returned to the LPS. Using drill plan information, the LPS selected the target object and proceeded to visual servo the camera to the drill-hole using the `FindHoleROI` command by passing the *CAMERA* parameter first to indicate alignment with the camera reference frame as shown in images (b) through (d). Once the camera was aligned, the LPS proceeded with visual servoing the boom-tip to the drill-hole. The remaining im-

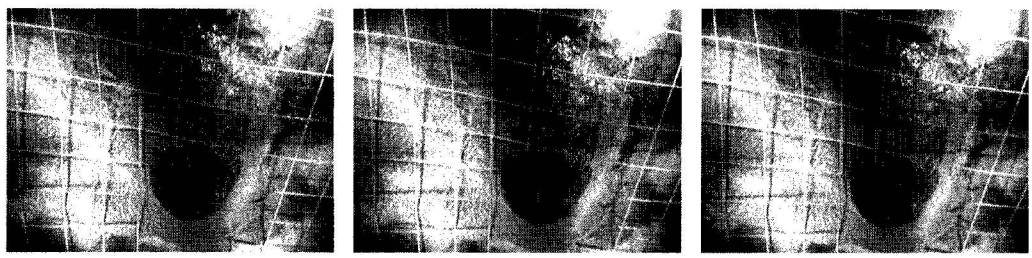
ages (e) through (k) represent the visual servoing of the boom-tip to the drill-hole by sending the `FindHoleROI` command and passing the *BOOM-TIP* parameter to indicate the specified reference frame.

Despite the reliability of the visual servoing process with respect to the boom-tip frame, tests performed on various drill-holes identified that a regular offset of $\{dx = -25mm, dy = 17mm\}$ referenced to the end-effector frame was required to dock the hose guide with the drill-hole collar. This reproduced the same operational performance observed with the prototype setup during lab testing. The error may be a result of the limited accuracy of the transformation measurements recorded during initial setup, required calibrations performed, or the accuracy of the drill plan data recorded during surveying. During full-scale testing, the offset was approximated only, by looking down the hose guide and observing if the hose guide was centered with the drill-hole. By including this offset when docking, aligning the hose guide with the drill-hole could be achieved, and the hose could be successfully inserted into the drill-hole for subsequent loading with emulsion. Visual servoing to other drill-holes within the mine was successfully performed to validate the identified offset. The results for each drill-hole servoed, verified the visual servoing capability as designed, with the successful insertion of the end-effector's hose into each drill-hole for loading with emulsion.

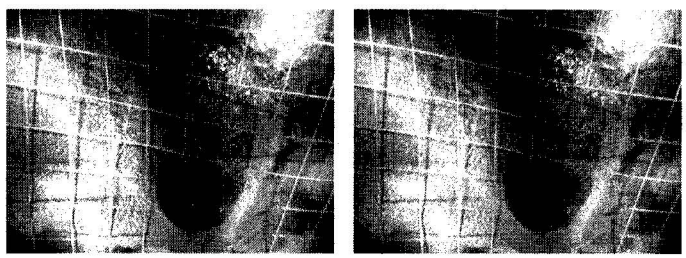
(a) Approach pose (b) (c)



(d) Camera aligned (e) (f)



(g) (h) (i)



(j) (k) Boom-tip aligned

Figure 4.26: ELAP full scale visual servoing tests in an underground mine. (a) is the approach pose image, (b)-(d) represent images grabbed during visual servoing the camera to the drill-hole, and (e)-(k) represent images grabbed during visual servoing the boom-tip to the drill-hole using hand-eye calibration information. Full size images of this set are included on the accompanying CD-ROM.

Chapter 5

Conclusions and Recommendations

5.1 Summary of Results and Conclusions

The use of vision equipped robots has been a proven tool in the manufacturing setting for a long time. However, the underground environment of an operating mine cannot be closely controlled, and thus adapting vision guided systems to this dynamic environment is no trivial task. With these challenges in mind, the ELAP project forged ahead with the goal of automating an emulsion loader used in underground mining operations. This thesis focused on the vision system component for this project, and thus organized the work into two main parts: to fulfill the challenging imaging requirements of locating and identifying drill-holes in the rock face of an underground mine, and secondly, to provide visual guidance to the hose guide for loading identified drill-holes with emulsion.

The first task in developing the ELAP VS was the development of the image processing algorithms to segment and identify drill-holes present in an image of

a underground rock face acquired from the video stream. An underground mine is a dynamic environment where non-uniform lighting, dust, water, and drill-hole obscurities challenge the imaging requirements. However, a key operation of the proposed ELAP system assists to reduce the drill-hole search task by using drill plan information to provide an initial pose for the robotic boom which positions the camera in front of the drill-hole. Rather than having to process an image of an entire rock face, the search area is greatly reduced to that of the drill-hole vicinity, provided that the blast plan data recorded is accurate. This approach reduced the image data to process, reduced the amount of background clutter, and permitted the use of simplified segmentation techniques to provide greater system response.

The image processing algorithms developed attempted to mitigate the adverse conditions for successful segmentation of drill-hole objects. Initial image pre-processing techniques such as window leveling were employed to improve image contrast. Thresholding and morphological operations were then also applied to remove small image objects, connections between touching objects, and fill any holes within objects. Blob analysis was then performed on the segmented image objects to calculate feature data to formulate a criteria for drill-hole selection. Over 1100 images were evaluated to determine the robustness of the algorithm. Using the specified lighting configuration with a diffuse pattern, the algorithms would perform sufficiently well to segment the drill-hole from image.

However, not all segmented objects were drill-holes and thus a classification model was developed using image features calculated during blob analysis. The classifier was designed using the MLC discriminate, and image features of new drill-hole candidates sampled were compared to the feature values of the training model. The candidate objects were then classified as either a background artifact and

discarded, or as a drill-hole where the results were returned to the calling function for further consideration. The FSS techniques which included a genetic algorithm, a sequential forward selection, and an exhaustive search were employed to both reduce and find an optimal feature subset for classification from the initial set of 19 features. Due to the small initial feature count, the ES strategy could be used. This resulted in a subset of only 9 features with an accuracy of 96.07%. The GA and SFS techniques, although non-optimal, were also tested to compare performance. For this data set, both the GA and SFS methods had comparable performance with a difference of only 0.5%. However, using a different or modified training and testing set will likely widen the performance result between the ES method and the other techniques.

The second part of this thesis focused on the overall vision system design to provide visual guidance to position the robotic boom's hose guide in front of a drill-hole for loading with emulsion. This included the selection of system components both hardware and software, necessary to fulfill the operational requirements. Using the camera and distance sensor mounted to the end of the robotic boom, camera and hand-eye calibration procedures were developed and implemented into a GUI software package. The two procedures determined the required calibration data to transform the drill-hole appearing in an image to both the camera and end-effector reference frames. A visual guidance algorithm included into the developed *VisSys* application software, communicates with the LPS to permit visual servoing of the robotic boom's hose guide to a drill-hole for loading. Visual servoing testing was conducted both in the lab using a RT200 robot using a simulated rock face, and in an underground mine with the full-scale ELAP unit. Both levels of testing validated the overall system design with the successful visual servoing to a drill-hole.

As a result, for optimum performance during drill-hole segmentation and classification, the following conclusions were drawn:

- Images of drill-holes captured where the surrounding area is wet, cause drill-holes to appear 'washed out' and result in poor segmentation after window leveling is applied. Additional techniques to process images containing water should be investigated further.

- Diffuse lighting is required to provide optimum contrast between drill-holes and the background rock face. The lighting should have no distinct pattern such that it provides an even distribution of light for the viewable area.

- The pattern recognition model and subsequent classification performance is dependent on how representative the training data is in comparison to new data being classified. Once the lighting type and configuration has been finalized, training data should then be gathered for different rock types and conditions, and updated regularly to sufficiently train the recognition model to ensure optimum performance.

- The ES feature subset selection strategy should be used when practical, to provide the optimum classification performance. However, if results are required in a more timely manner or if a large number of features are used, the GA and SFS techniques should be investigated to provide comparable performance.

5.2 Recommendations

The work presented in this thesis represents the initial research and development phase for the ELAP vision system. Further development will be necessary to mitigate deficiencies and issues identified during system integration, and field testing. The following recommendations are suggested for subsequent development of the ELAP vision system.

- Improve image segmentation. In cases where lighting is sub-optimal, suggest selecting a series of threshold values and combine the results from each in an attempt to increase the robustness of the segmentation process. Also, investigate techniques that may be more effective in segmenting images of reduced contrast, particularly, those that contain wet or watery areas.

- Review the foreground lighting design. The halogen tractor lights used, do not provide the flat diffuse lighting required to provide good contrast, and thus should be replaced with a more suitable lighting source. Also, the lighting mounting arrangement should be streamlined such that they do not limit the maneuverability and work space of the end-effector.

- Position the camera such that the end-effector can been observed in the cameras field of view. This will serve as a constant reference point between the camera and the end-effector, and will attempt to improve the visual servoing accuracy of the end-effector to the drill-hole.

- Update the drill-hole training model data online, as each new drill-hole is identified. This will provide a better representation for the drill-hole training model to help assist in reducing the false positive rate.

- Improve the camera calibration process. Correctly and accurately isolating the corners on the calibration object (i.e. checkerboard) proved to be difficult if lighting conditions were less than ideal. Substituting the checkerboard with a similar grid of dots has been successfully demonstrated, and can be segmented more easily under adverse lighting conditions.

- Modify the hand-eye calibration to include the rotation component.

- Re-design the imaging system to incorporate a 3D laser scanner for imaging the rock face rather than video. A laser scanner would eliminate the dependency on recorded drill plan data, since the drill-hole locations will now be identified using the same equipment within the same reference plane, and thus minimizing the errors introduced.

References

[1] "Emulsion loader automation project (elap): A precarn proposal," July 2000.

[2] H. A. Ltd., "Market analysis of automation technology opportunities in the mining sector," National Research Council (NRC), Industry and Science Canada (ISC), and CANMET, Tech. Rep., Dec. 1993.

[3] Mining automation program. Natural Resources Canada, Minerals and Metals Sector. [Online]. Available: http://www.nrcan.gc.ca/mms/prod-serv/map_e.htm

[4] Laurentian university mining automation laboratory (lumal). Laurentian University, School of Engineering. Sudbury, Ont. Canada, P3E 2C6. [Online]. Available: laurentian.ca/lumal/

[5] *Matrox Imaging Library MIL 6.1*, Matrox Electronic Systems Ltd., 2000.

[6] J. C. Russ, *The Image Processing Handbook*, 3rd ed. Boca Raton, Florida: CRC Press, 1999.

[7] V. H. Milan Sonka and R. Boyle, *Image Processing, Analysis, and Machine Vision*, 2nd ed. Pacific Grove, California: PWS, 1999.

[8] T. W. Ridler and S. Calvard, "Picture thresholding using an iterative selection method," in *Proc. IEEE Transactions On Systems, Man, and Cybernetics*, vol. SMC-8, Aug. 1978, pp. 630–632.

[9] W. R. G.W. Zack and S. Latt, "Automatic measurement of sister chromatid exchange frequency," *Journal of Histochemistry and Cytochemistry*, vol. 25, no. 7, pp. 741–753, July 1977.

[10] J. Canny, "A computational approach to edge detection," in *IEEE Transactions On Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, Nov. 1986, pp. 679–698.

[11] S. L. Horowitz and T. Pavlidis, "Picture segmentation by a directed split-and-merge procedure," in *2nd International Joint Conference on Pattern Recognition*, Copenhagen, Denmark, 1974, pp. 424–433.

[12] A. B. Yuri Ivanov and J. Liu, "Fast lighting independent background subtraction," in *International Journal on Computer Vision*, vol. 2. MIT Media Laboratory, June 2000, pp. 199–207.

[13] T. Yoshida, "Background differencing technique for image segmentation based on the status of reference pixels," in *ICIP04*, 2004, pp. V: 3487–3490.

[14] L. G. Shapiro and G. C. Stockman, *Computer Vision*, 1st ed. Upper Saddle River, New Jersey: Prentice Hall, 2001.

[15] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision: Volume 1*, 1st ed. Addison Wesley, 1992.

[16] R. Duda. (1997, Oct.) Pattern recognition for hci. Department of Electrical Engineering, San Jose State University. [Online]. Available: http://www.engr.sjsu.edu/~knapp/HCIRODPR/PR_home.htm

[17] P. H. R.O. Duda and D. Stork, *Pattern Classification*, 2nd ed. John Wiley and Sons, 2001.

[18] W. Yan and K. Goebel, "Feature selection for partial discharge diagnosis," ser. SPIE: Health Monitoring and Smart Nondestructive Evaluation of Structural and Biological Systems IV, vol. 12, 2005, pp. 166–175.

[19] W. Siedlecki and J. Sklansky, "A note on genetic algorithms for large-scale feature selection." in *Proc. IEEE Transactions On Computers*, no. 10, 1989, pp. 335–347.

[20] J. Yang and V. Honavar, "Feature subset selection using A genetic algorithm," in *Genetic Programming 1997: Proceedings of the Second Annual Conference.* Stanford University, CA, USA: Morgan Kaufmann, 1997, p. 380. [Online]. Available: citeseer.ist.psu.edu/article/yang97feature.html

[21] M. Richeldi and P. Lanzi, "Improving genetic-based feature selection by reducing data dimensionality," 1996. [Online]. Available: citeseer.ist.psu.edu/richeldi96improving.html

[22] Z. Sun, X. Yuan, G. Bebis, and S. Louis, "Genetic feature subset selection for gender classication: A comparison study," Aug. 2002. [Online]. Available: citeseer.ist.psu.edu/sun02genetic.html

[23] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1st ed. Addison-Wesley, 1989.

[24] K. D. Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ann Arbor, Tech. Rep., 1975.

[25] H. Liu and H. Motoda, *Feature Selection: For Knowledge Discovery and Data Mining*, 1st ed. Kluwer Academic Publishers, 1998.

[26] J. Schneider. Cross validation. School of Computer Science, Carnegie Mellon University. [Online]. Available: http://www.cs.cmu.edu/~schneide/tut5/node42.html

[27] (2001, Nov.) Vision-guided robots for inspection, material handling. Society of Manufacturing Engineers. [Online]. Available: http://www.findarticles.com/p/articles/mi_qa3618/is_200111/ai_n9008431

[28] A. Wilson. Machine vision targets industrial automation. PennWell Publishing. [Online]. Available: http://cgw.pennnet.com/Articles/Article_Display.cfm

[29] C. Scott. (2004, Jan.) Vision-guided robot dispenses liquid gasket. Vision Sytems Design Magazine. [Online]. Available: http://vsd.pennnet.com/Articles/Article_Display.cfm?Section=Articles&Subsection=Display&ARTICLE_ID=196815

[30] R. Zens. (2006, Aug.) Vision-guided robot system trims parts. Vision Sytems Design Magazine. [Online]. Available: http://vsd.pennnet.com/Articles/Article_Display.cfm?Section=ARTCL&ARTICLE_ID=261912&VERSION_NUM=2&p=19

[31] G. H. Seth Hutchinson and P. Corke, "A tutorial on visual servo control," in *Proc. IEEE Transactions On Robotics and Automation*, vol. 12, Oct. 1996, pp. 651–670.

[32] R. Y. Tsai, "A versatile camera calibration technique for high accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *IEEE Electron Device Lett.*, 1985.

[33] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," Infotech Oulu and Depeartment of Electrical Enginnering University of Oulu FIN-90570 Oulu, Finland.

[34] S. Maybank and O. Faugeras, "A theory of self-calibration of a moving camera," *The International Journal of Computer Vision*, vol. 8, pp. 123–152, Aug. 1992.

[35] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," *IEEE Electron Device Lett.*, 1999.

[36] *Open Source Computer Vision Library: Reference Manual*, Intel Corporation., 2001. [Online]. Available: http://developer.intel.com

[37] W. Sun and J. R. Cooperstock, "Requirements for camera calibration: Must accuracy come with a high price?" Department of Electrical and Computer Engineering, McGill University, Montreal, Quebec H3A 2A7 Canada.

[38] F. Dornaika and R. Horaud, "Simultaneous robot-world and hand-eye calibration," in *Proc. IEEE Transactions On Robotics and Automation*, vol. 14, no. 4, Aug. 1998, pp. 617–622.

[39] Y. C. Shiu and S. Ahmad, "Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form ax = xb," in *Proc. IEEE Transactions On Robotics and Automation*, vol. 5, no. 1, Feb. 1989, pp. 16–29.

[40] R. Y. Tsai and R. K. Lenz, "A new technique for fully autonomous and efficient 3d robotics hand/eye calibration," in *Proc. IEEE Transactions On Robotics and Automation*, vol. 5, no. 3, June 1989, pp. 345–358.

[41] J. Chou and M. Kamel, "Finding the position and orientation of a sensor on a robot manipulator using quaternions," in *International Journal of Robotics Research*, vol. 10, no. 3, 1991, pp. 240–254.

[42] H. Chen, "A screw motion approach to uniqueness analysis of head-eye geometry," in *In IEEE Conf. Computer Vision and Pattern Recognition*, Maui, Hawaii, June 1991, pp. 145–151.

[43] C. Wang, "Extrinsic calibration of a robot sensor mounted on a robot," in *Proc. IEEE Transactions On Robotics and Automation*, vol. 8, no. 2, 1992, pp. 161–175.

[44] P. Meinicke and J. Zhang, "Calibration of a self-viewing eye-on-hand configuration," Technical Computer Science, Faculty of Technology, University of Bielefeld, 33501 Bielefeld, Germany. [Online]. Available: http://citeseer.ist.psu.edu/660760.html

[45] J. G. Prasantha Guda, Jin Cao and E. L. Hall, *Handbook of Industrial Automation*, R. Shell and E. Hall, Eds. Falls Church, Virginia: Marcel Dekker, New York, 2000.

[46] Vision systems. IAS Corp., Industrial Automation Specialists. 17 Reserach Drive, Hampton VA, 23666. [Online]. Available: http://www.iascorp.net/vision.htm

[47] G. B. Ross Poole, Peter Golde and M. Scoble, "Implementation of communications and machine teleoperation in mine automation."

[48] L. Bloomquist and E. Hinton, "Towards an infrastructureless guidance system: A proposed guidance system for autonomous underground vehicles," *Canadian Institute of Mining Journal*, Jan. 2001.

[49] P. LeFeuvre, "Fish species identification using image analysis of echo sounder images," Master's thesis, Memorial University of Newfoundland, St. John's, NL Canada, 2004.

[50] X. Zhao, "Automated image analysis for petrographic image assessments," Master's thesis, Memorial University of Newfoundland, St. John's, NL Canada, 2000.

[51] J. Byrne and S. Singh, "Precise image segmentation for forest inventory," 5000 Forbes Ave. Pittsburgh, PA 15213, Tech. Rep., May 1998.

[52] G. Cooper, "The textural analysis of gravity data using co-occurrence matrices," in *Computers and Geosciences*, vol. 30, 2004, pp. 107–115.

[53] N. Otsu, "A threshold selection method from gray-level histograms," ser. IEEE Trans. Sys., Man., Cyber., vol. 9, 1979, pp. 62–66.

[54] P. F. Sturn and S. J. Maybank, "On plane-based camera calibration: A general algorithm, singularities, applications," *IEEE Electron Device Lett.*, 1999.

[55] T. A. Clarke and J. G. Fryer, "The development of camera calibration methods and models," ser. Photogrammetric Record, vol. 16, no. 91, Apr. 1998, pp. 51–66.

[56] C. C. Slama, *Manual of Photogrammetry*, 4th ed. Falls Church, Virginia: American Society of Photogrammetry, 1980.

[57] J.-Y. Bbouguet. Camera calibration toolbox for matlab. California technical University. [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html#own_calib

[58] M. Sallinen and T. Heikkila, "A simple hand-eye calibration method for a 3d laser range sensor," VTT Automation, PO Box 13023, FIN-90571 Oulu, Finland. [Online]. Available: http://www.ele.vtt.fi/aut/

[59] H. Zhuang and Y. C. Shiu, "A noise-tolerant algorithm for robotic hand-eye calibration with or without sensor orientation measurement," in *Proc. IEEE Transactions On System, Man and Cybernetics*, vol. 23, no. 4, 1993, pp. 1168–1175.

[60] C. Garcia, "Fully vision-based calibration of a hand-eye robot," in *Autonomous Robots*, vol. 6. PO Box 1385, GR 711 10 Herakliom, Crete, greec: Institute of Computer Science - Foundation of Research and Technology-Hellas (FORTH), 1999, pp. 223–238.

[61] R. H. Nicolas Andreff and B. Espiau, "On-line hand-eye calibration," INRIA Rhone-Alpes/ GRAVIR-IMAG, 655 av de l'Europe, 38330 Montbonnot Saint Martin, France.

[62] H. Malm and A. Heyden, "A direct method for hand-eye calibration and depth reconstruction," Department of Mathematics, Lund Institute of Technology, PO Box 118, S-22100 Lund.

[63] F. Dornaika and R. Horaud, "Hand-eye calibration," in *Proc. Workshop on Computer Vision for Space Applications*, 1993, pp. 369–379.

[64] Z. R. H. Zhuang and R. Sudhakar, "Simultaneous robot-world and tool/flange calibration by solving homogeneous transformation of the form ax=yb," in *Proc. IEEE Transactions On Robotics and Automation*, vol. 10, Aug. 1994, pp. 549–554.

[65] F. Park and B. Martin, "Robot sensor calibration: Solving ax=xb on the euclidean group," in *Proc. IEEE Transactions On Robotics and Automation*, vol. 10, Oct. 1994, pp. 717–721.

[66] K. W. H. Zhuang and Z. Roth, "Simultaneous calibration of a robot and a hand mounted camera," in *Proc. IEEE Transactions On Robotics and Automation*, vol. 11, Oct. 1995, pp. 649–660.

[67] F. C. Ezio Malis and S. Boudet, "2-1/2-d visual servoing," in *Proc. IEEE Transactions On Robotics and Automation*, vol. 15, no. 2, Apr. 1999, pp. 238–250.

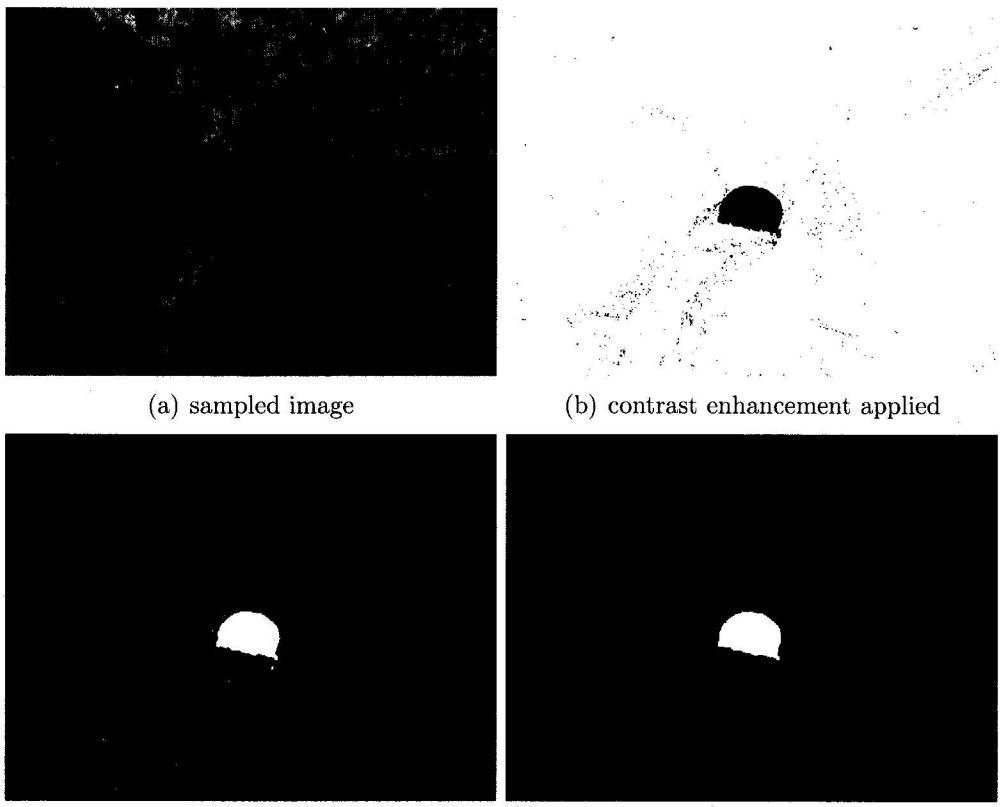
[68] N. R. Gans and S. A. Hutchinson, "An experimental study of hybrid switched system approaches to visual servoing," in *Proc. IEEE Transactions On Robotics and Automation*, Sept. 2003, pp. 3061–3068.

[69] F. C. Bernard Espiau and P. Rives, "A new approach to visual servoing in robotics," in *Proc. IEEE Transactions On Robotics and Automation*, vol. 8, no. 3, June 1992, pp. 313–326.

[70] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The Confidence of Vision and Control*, ser. Lecture Notes in Control and Information Systems. Springer-Verlag, 1998, vol. 237, pp. 66–78.

Appendix A

Image Segmentation Results For Sample Drill-Hole Images



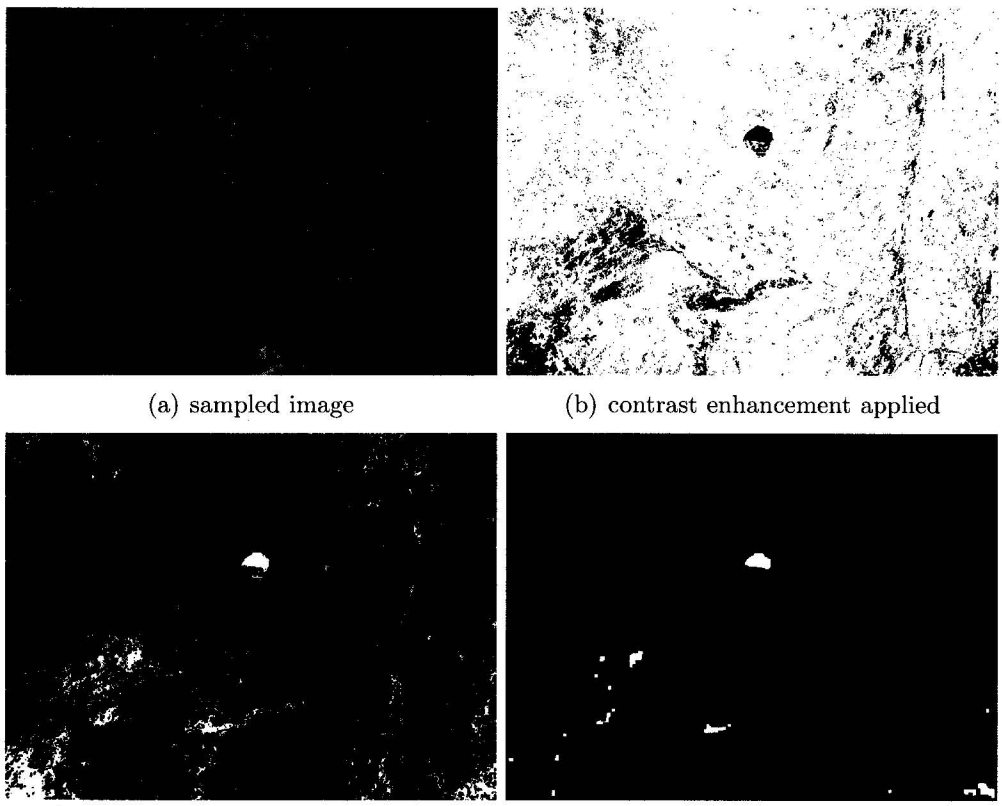
(a) sampled image

(b) contrast enhancement applied

(c) morphology applied

(d) segmented image

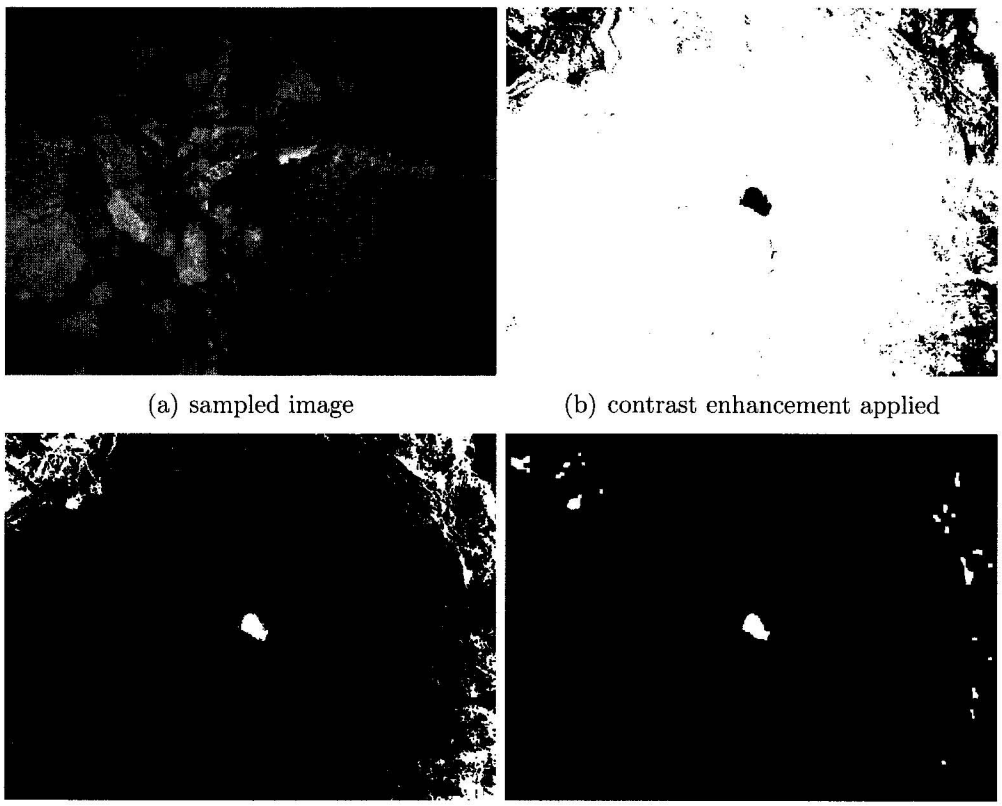
Figure A.1: Example #1 - Drill-hole image segmentation results.



(a) sampled image (b) contrast enhancement applied

(c) morphology applied (d) segmented image

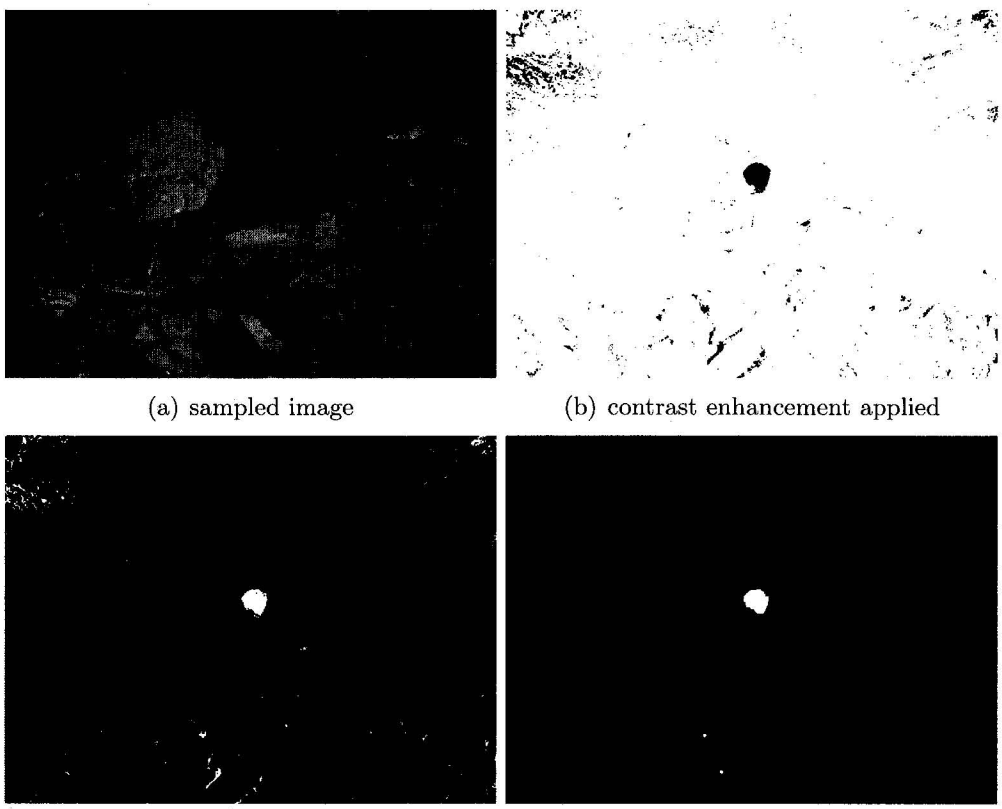
Figure A.2: Example #2 - Drill-hole image segmentation results.



(a) sampled image (b) contrast enhancement applied

(c) morphology applied (d) segmented image

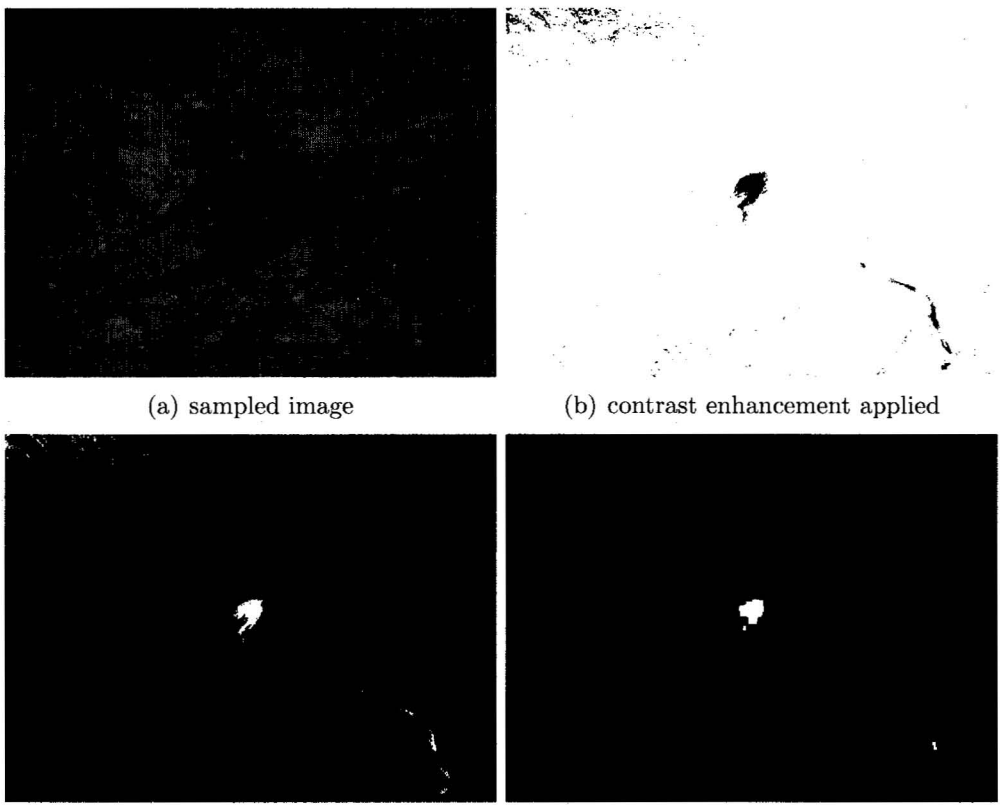
Figure A.3: Example #3 - Drill-hole image segmentation results.



(a) sampled image (b) contrast enhancement applied

(c) morphology applied (d) segmented image

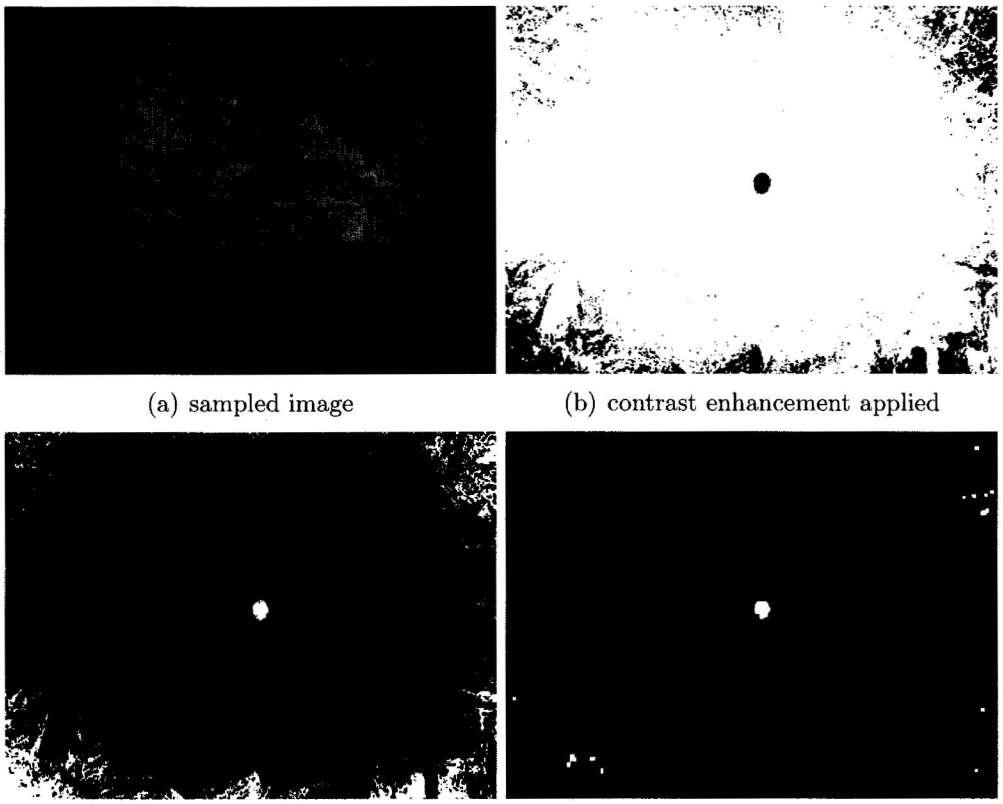
Figure A.4: Example #4 - Drill-hole image segmentation results.



(a) sampled image (b) contrast enhancement applied

(c) morphology applied (d) segmented image

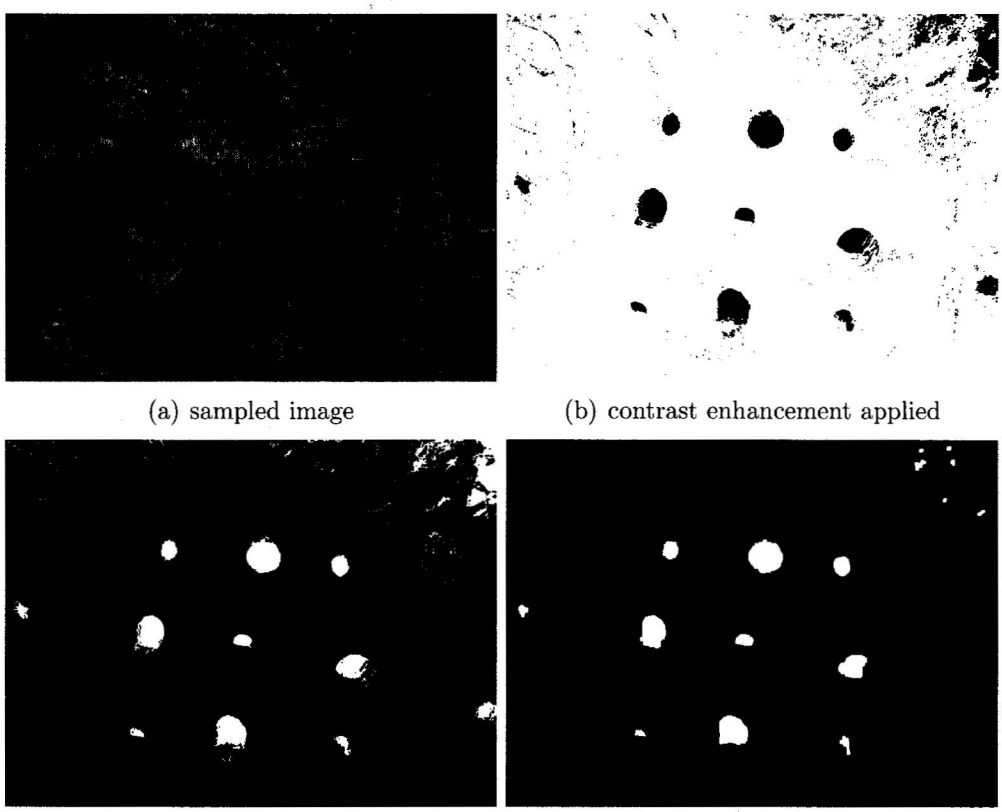
Figure A.5: Example #5 - Drill-hole image segmentation results.



(a) sampled image (b) contrast enhancement applied

(c) morphology applied (d) segmented image

Figure A.6: Example #6 - Drill-hole image segmentation results.



(a) sampled image (b) contrast enhancement applied

(c) morphology applied (d) segmented image

Figure A.7: Example #7 - Drill-hole image segmentation results.

Appendix B

Features Used For Classification

The following provides a description of the original 19 features selected for classification of segmented image objects. The final set of features used are a subset as selected by the FSS optimization techniques discussed in Chapter 3. The features are enumerated for identification as in Table 3.1.

- **Minimum Pixel(1)** The minimum intensity value in a blob.
- **Mean Pixel(2)** The average pixel intensity value in a blob: $\frac{\sum p_i}{A}, p_i$ is the intensity of the i^{th} pixel.
- **Standard Deviation of Pixels(3)** The standard deviation of the pixel values in a blob.

$$\sqrt{\frac{\sum p_i^2 - (\sum p_i)^2/N}{N}} \tag{B.1}$$

  where: N= the number of pixels, and p_i= the intensity of the i^{th} pixel.
- **Compactness(4)** Derived from the perimeter P and area A, which has the minimum value of 1 for a circle. The more convoluted the shape, the greater the value.
- **Feret Elongation(5)** A measure of the shape of the blob. A Feret diameter is calculated by measuring the diameter of the blob at a certain angle measured from the horizontal. For minimum Feret diameter, f, n Feret diameters are measured at $n/180$ angles and the minimum diameter is selected. The maximum Feret diameter, F, is calculated the same way, but the maximum diameter is selected. Feret elongation is equal to F/f.

- **Central Moment X0Y2(6)** Normalized gray scale second order central moment. It is a measure of how horizontally dispersed the object's pixels are from its centroid.

$$CMX0Y2 = \frac{\sum y_i^2 p_i}{A^2} \tag{B.2}$$

where:
y_i= horizontal distance from the i^{th} pixel to the centroid of the object.
p_i= the intensity of the i^{th} pixel.
A= the area of the object in pixels.

- **Central Moment X2Y0(7)** Normalized gray scale second order central moment. It is a measure of how vertically dispersed the object's pixels are from its centroid.

$$CMX2Y0 = \frac{\sum x_i^2 p_i}{A^2} \tag{B.3}$$

where:
x_i= vertical distance from the i^{th} pixel to the centroid of the object.
p_i= the intensity of the i^{th} pixel.
A= the area of the object in pixels.

- **Central Moment X1Y1(8)** Normalized gray scale second order central moment. It is a measure of how horizontally and vertically dispersed the object's pixels are from its centroid.

$$CMX1Y1 = \frac{\sum x_i y_i p_i}{A^2} \tag{B.4}$$

where:
x_i= vertical distance from the i^{th} pixel to the centroid of the object.
y_i= horizontal distance from the i^{th} pixel to the centroid of the object.
p_i= the intensity of the i^{th} pixel.
A= the area of the object in pixels.

- **Binary Central Moment X0Y2(9)** Normalized binary second order central moment. It is a measure of how horizontally dispersed the object's pixels are from its centroid.

$$BCMX0Y2 = \frac{\sum y_i^2 p_i}{A^2} \tag{B.5}$$

where:
y_i= horizontal distance from the i^{th} pixel to the centroid of the object.

p_i= the intensity of the i^{th} pixel.
A= the area of the object in pixels.

- **Binary Central Moment X2Y0(10)** Normalized binary second order central moment. It is a measure of how vertically dispersed the object's pixels are from its centroid.

$$BCMX2Y0 = \frac{\sum x_i^2 p_i}{A^2} \tag{B.6}$$

where:
x_i= vertical distance from the i^{th} pixel to the centroid of the object.
p_i= the intensity of the i^{th} pixel.
A= the area of the object in pixels.

- **Binary Central Moment X1Y1(11)** Normalized binary second order central moment. It is a measure of how horizontally and vertically dispersed the object's pixels are from its centroid.

$$BCMX1Y1 = \frac{\sum x_i y_i p_i}{A^2} \tag{B.7}$$

where:
x_i= vertical distance from the i^{th} pixel to the centroid of the object.
y_i= horizontal distance from the i^{th} pixel to the centroid of the object.
p_i= the intensity of the i^{th} pixel.
A= the area of the object in pixels.

- **Energy(12)** An homogeneity measure — the more homogeneous the object, the larger the value [7].

$$Energy = \sum_{a,b} P_{\phi d}(a,b)^2 \tag{B.8}$$

where:
$P_{\phi d}(a,b)$ determined from the Grey Level Co-occurrence Matrix (GLCM) and represents the probability that two pixels with a specified separation have gray levels a and b. The separation is specified by a displacement d, and and angle ϕ. For this work, the values used for d and ϕ were d(1), $\phi(0, 45, 90, 135)$. Refer to Section 2.2.5.1 for further explanation.

- **Entropy(13)** Measure of the homogeneity of an object.

$$Entropy = -\sum_{a,b} P_{\phi d}(a,b) \log P_{\phi d}(a,b) \tag{B.9}$$

See Energy definition for parameter explanation.

- **Homogeneity(14)** Measure of local image variations.

$$Homogeneity = \sum_{a,b} \frac{P(a,b)}{1+|a-b|} \tag{B.10}$$

See Energy definition for parameter explanation.

- **Contrast(15)** A measure of local image variations; typically k= 2, l=1 [7].

$$Contrast = \sum_{a,b} |a-b|^k \left(P_{ab}\right)^l \tag{B.11}$$

See Energy definition for parameter explanation.

- **Inverse Difference Moment(16)** Determined from the GLCM and attains a maximum value when all the image pixels that are compared have the same value [52].

$$IDM = \sum_{a,b} \frac{\left(P_{ab}\right)^l}{|a-b|^k} \tag{B.12}$$

See Energy definition for parameter explanation.

- **Correlation(17)** Measure of image linearity; linear directional structures in direction ϕ result in large correlation values [7].

$$Correlation = \sum_{a,b} \frac{(a-\mu)(b-\mu)P_{ab}}{\sigma^2} \tag{B.13}$$

and

$$\mu = \sum_{ab} iP_{ab} \tag{B.14}$$

where:
i is the gray level intensity. See Energy definition for parameter explanation.

- **Elongation(18)** A measure of the shape of an object and is derived from area A and perimeter P,

- **Roughness(19)** A measure of the roughness of a blob's surface. A smooth convex object will have the minimum roughness of 1 [5]. It is calculated using P/C; where P is the blob's perimeter (i.e. the total length in pixels of the edges in a blob, including the edges of any holes), and C is the convex perimeter (i.e. the length of the perimeter of the convex hull of the blob).

Appendix C

ELAP Vision System Calibration File Data

[Camera Calibration Parameters]
Board_Width = 16
Board_Height = 18
SquareSize = 10.159
FocalLength_X_pixels = 347.2757
FocalLength_Y_pixels = 346.3365
PrincPoint_X = 321.3295
PrincPoint_Y = 251.6542
CameraMatrix0 = 347.2757
CameraMatrix1 = 0.0
CameraMatrix2 = 321.3295
CameraMatrix3 = 0.0
CameraMatrix4 = 346.3365
CameraMatrix5 = 351.6542
CameraMatrix6 = 0.0
CameraMatrix7 = 0.0
CameraMatrix8 = 1.0
Distortion1 = -1.9551e-1
Distortion2 = 5.0969e-1
Distortion3 = 8.4428e-4
Distortion4 = 2.9581e-3

[Hand-Eye Calibration Parameters]
HandEyeX[0][0] = 9.997477531433106e-001

HandEyeX[0][1] = 1.547639910131693e-002
HandEyeX[0][2] = 1.612677238881588e-002
HandEyeX[0][3] = -60
HandEyeX[1][0] = -1.527536660432816e-002
HandEyeX[1][1] = 9.998015165328980e-001
HandEyeX[1][2] = -1.256382465362549e-002
HandEyeX[1][3] = -18
HandEyeX[2][0] = -1.631853543221951e-002
HandEyeX[2][1] = 1.231363229453564e-002
HandEyeX[2][2] = 9.997899532318115e-001
HandEyeX[2][3] = -115
HandEyeX[3][0] = 0.0
HandEyeX[3][1] = 0.0
HandEyeX[3][2] = 0.0
HandEyeX[3][3] = 1.0

[Boom To Sensor Transform]
TranslationX = -65.50
TranslationY = -206.375
TranslationZ = -161.925