



The design of an automatic vision switching system
by Chunsheng Feng

A thesis submitted in partial fulfillment Of the requirements for the degree Of Master of Science In
Industrial and Management Engineering
Montana State University
© Copyright by Chunsheng Feng (1998)

Abstract:

The importance of Robotics has been recognized by the manufacturing industry. The introduction of Computer Vision has greatly increased the versatility and application domain of robots such as the interaction between the robot and its changing environment. From previously developed vision position correction methods and currently used visual servo-feedback techniques, computer vision systems present many possibilities in improving the quality and productivity of an industrial product.

But computer vision systems also have limits in dealing with some robotic tasks such as Storage Battery Cap Installation. Random geometric distortion exists, which results from the characteristics of battery material or from the positioning error of fixtures. Robotic teaching-playback method can not guarantee the assembly precision when deviation happens. A vision system introduced into the robotic system allows it to respond to an uncertain environment. Conventional vision correction methods only drive robots to the final position without considering robotic trajectory control. They reduce the robotic operation speed compared with stable conditions. Visual servo-feedback techniques can generate the optimal path and assembly precision, but the calculation of visual servo control algorithm is time consuming. It also reduces robotic execution time. In addition, random assembly tasks sometimes do not need vision correction. So the justification of this robotic vision system is improved robotic assembly accuracy that does not affect other robot performance.

In this thesis a new robotic vision control system, Automatic Vision Switching System (AVSS), was designed. It was derived from the concept of Just-in-Time and dealt with the installation of storage battery caps. It demonstrated a vision system that functions to correct just when it is needed. Image processing techniques for solving the battery hole's centroid values were included in AVSS design. The experiment was carried out based on an AdeptOne robot, a Vision-EZ system, a camera with eye-in-hand configuration, and a battery model. The experimental results showed that the AVSS can integrate a vision system with a robotic system efficiently. The AVSS identified the deviation of battery holes' locations and generated the corrected values in real time. The control accuracy, operation speed and optimal path of robotic manipulator were obtained. The AVSS can also be used in other industrial tasks where random variation is present.

THE DESIGN OF AN AUTOMATIC VISION SWITCHING SYSTEM

by

Chunsheng Feng

A thesis submitted in partial fulfillment
Of the requirements for the degree

Of

Master of Science

In

Industrial and Management Engineering

MONTANA STATE UNIVERSITY-BOZEMAN
Bozeman, Montana

January 1998

© COPYRIGHT

by

Chunsheng Feng

1998

All Rights Reserved

N378
F356

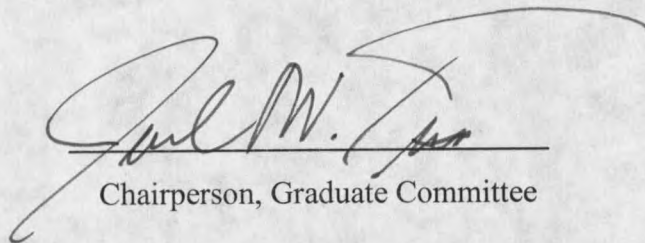
APPROVAL

of a thesis submitted by

Chunsheng Feng

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

Dr. Joel W. Troxler

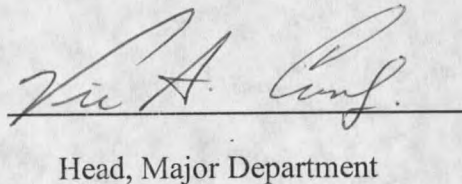

Chairperson, Graduate Committee

1-15-98

Date

Approved for the Department of Mechanical and Industrial Engineering

Dr. Vic Cundy

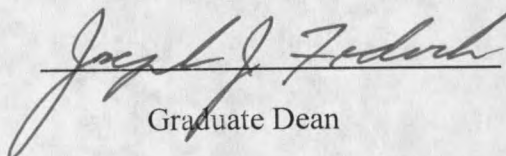

Head, Major Department

1/15/98

Date

Approved for the College of Graduate Studies

Dr. Joseph Fedock


Graduate Dean

1/20/98

Date

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University-Bozeman, I agree that the Library shall make it available to borrowers under rules of the Library.

If I have indicated my intention to copyright this thesis by including a copyright notice page, copying is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for permission for extended quotation from or reproduction of this thesis in whole or in parts may be granted only by the copyright holder.

Signature

Chunsheng Feng

Date

January 14, 1998

ACKNOWLEDGEMENT

I would like to use this opportunity to express my sincere appreciation to my advisor, Dr. Joel W. Troxler for his teaching, guidance and support. Although I did not list all knowledge I have learned from Dr. Troxler, that, as well as his guidance in research will help me to meet future challenges. I also would like to thank other members of the committee, Dr. Donald W. Boyd and Dr. Paul Schillings for their continued encouragement and help.

I want to thank my wife and my son for their understanding and support during my study in Montana State University. They helped me to understand the meaning of my work and my study.

TABLE OF CONTENTS

	Page
LIST OF TABLES.....	vii
LIST OF FIGURES.....	viii
ABSTRACT.....	x
1. INTRODUCTION.....	1
Development of Robotic Vision System.....	1
Vision Adjustment.....	3
Vision Tracking.....	4
Review of Selected Robotic Vision Literature.....	6
Justification of Robotic Vision System.....	8
2. PRESENTATION OF PROBLEM.....	10
3. DESIGN OF AUTOMATIC VISION SWITCHING SYSTEM.....	13
AVSS Theory.....	14
Design of AVSS.....	16
Threshold and Image Filter.....	16
Geometry Measurement of Holes.....	19
Vision Correction.....	21
4. DESIGN OF EXPERIMENT.....	23
Configuration of Hardware.....	23
Adept Robot.....	23
Vision-EZ System.....	23
Other Equipment.....	24
Design of Experiment.....	24
Experimental Procedures.....	25
Results and Discussion.....	32
Boundary Points	32

TABLE OF CONTENTS—Continued

	Page
Edge Detection.....	33
Identification of Distortion.....	33
5. CONCLUSION AND RECOMMENDATION.....	35
LITERATURE CITED.....	38
APPENDICES.....	41
Appendix A: Calculation of Arc Center Coordinates Based on Boundary Points.....	42
Appendix B: The Solving of Inverse Kinematics for Two-joints Robot.....	45
Appendix C: Robotic Control Software Source Codes.....	48
Appendix D: Image Processing Software Source Codes.....	53

LIST OF TABLES

Table		Page
1.	Standard Positions and Orientation of Holes.....	26
2.	Image Position (pixel number) of Standard Hole Locations.....	28
3.	Comparison of the Two Methods for Boundary Point Search.....	33

LIST OF FIGURES

Figure		Page
1.	Robotic Vision Adjustment Process.....	4
2.	Robotic Vision Tracking Process.....	5
3.	Battery Cap Assembly with Robot.....	11
4.	Inconsistent Features and Locations.....	11
5.	Configuration of Robotic Control System with AVSS.....	13
6.	Location of Battery Holes for Vision Correction.....	15
7.	Path Correction.....	16
8.	Definition of Pixels within a Window.....	17
9.	Definition of Processing Window for Sobel Operator.....	18
10.	Scanning Lines for Solving Centroid Values of Holes.....	20
11.	Distortion Situation of Holes.....	20
12.	Middle-value Method for Boundary Points Search.....	21
13.	Configuration of Kinematics for Robot.....	22
14.	Configuration of Adept Robot Manipulator.....	24
15.	Configuration of Robotic Vision Assembly System.....	25
16.	Battery Model and Gripper-Camera Configuration.....	26
17.	Relationship of Camera Coordinate and Image Coordinate.....	27

LIST OF FIGURES—Continued

	Page
18. Location Adjustment of Standard Holes.....	28
19. Flowchart of Robotic Control Software.....	30
20. Flowchart of Vision System Software.....	31
21. Deviation Limit of a Hole Location.....	32
22. Robotic Motion path with AVSS.....	34
23. Scanning Line Method for Solving Centroid Value.....	43
24. Configuration of Kinematics for Two-Joint Robot.....	46
25. Robotic Teaching Points.....	49

ABSTRACT

The importance of *Robotics* has been recognized by the manufacturing industry. The introduction of *Computer Vision* has greatly increased the versatility and application domain of robots such as the interaction between the robot and its changing environment. From previously developed vision position correction methods and currently used visual servo-feedback techniques, computer vision systems present many possibilities in improving the quality and productivity of an industrial product.

But computer vision systems also have limits in dealing with some robotic tasks such as *Storage Battery Cap Installation*. Random geometric distortion exists, which results from the characteristics of battery material or from the positioning error of fixtures. Robotic teaching-playback method can not guarantee the assembly precision when deviation happens. A vision system introduced into the robotic system allows it to respond to an uncertain environment. Conventional vision correction methods only drive robots to the final position without considering robotic trajectory control. They reduce the robotic operation speed compared with stable conditions. Visual servo-feedback techniques can generate the optimal path and assembly precision, but the calculation of visual servo control algorithm is time consuming. It also reduces robotic execution time. In addition, random assembly tasks sometimes do not need vision correction. So the justification of this robotic vision system is improved robotic assembly accuracy that does not affect other robot performance.

In this thesis a new robotic vision control system, *Automatic Vision Switching System* (AVSS), was designed. It was derived from the concept of Just-in-Time and dealt with the installation of storage battery caps. It demonstrated a vision system that functions to correct just when it is needed. Image processing techniques for solving the battery hole's centroid values were included in AVSS design. The experiment was carried out based on an AdeptOne robot, a Vision-EZ system, a camera with eye-in-hand configuration, and a battery model. The experimental results showed that the AVSS can integrate a vision system with a robotic system efficiently. The AVSS identified the deviation of battery holes' locations and generated the corrected values in real time. The control accuracy, operation speed and optimal path of robotic manipulator were obtained. The AVSS can also be used in other industrial tasks where random variation is present.

CHAPTER 1

INTRODUCTION

In recent years, robotic technology was demanded not only in the manufacturing industries, but also in the non-manufacturing industries such as construction, ocean development, nuclear power, and medicine. Robotics has gained wide recognition within manufacturing industries for three main reasons: robots characterize the trend toward automation of manufacturing processes, enhance efforts to reduce rework times and costs, and therefore increase productivity and improve product quality. Quality control plays an important part in industrial production. Robots can perform repetitive industrial tasks more accurately and faster than human workers, hence quality, as well as productivity, is improved [1]. Robotics is a multidisciplinary field that combines control theories, computer techniques, industrial measurement technologies, and mechanical engineering design. Improvement in any one of these areas can result in the improvement of product quality.

Development of Robotic Vision System

Acquisition, processing, and interpretation of sensory information are important activities in a robotic system. At lower levels such as internal position sensing, sensory information is used to derive control signals to drive the robots. At higher levels such as

vision and other external sensors, sensory information is used to create models of the robotic system and its environment. Integration of sensory information increases the versatility of robots and extends their domain of application. Moreover, robots have made far less impact on applications where the work environment and object placement cannot be accurately controlled [2], so higher level sensor-based control is essential if robots are to perform adequately in the real world. Through the use of sensors such as computer vision, conventional machines are able to react and decide how to adapt to changing circumstances [3].

Computer vision, particularly in the context of sensor-driven robotic systems, is an extensive, expanding and hugely exciting field to study and, not surprisingly, one with a great deal of potential [4]. Vision systems have undergone dramatic changes in the last few years due to the rapid advances in microprocessors and specialized digital signal image processing electronics. It has substantially expanded the capacity and flexibility of robotic applications such as assembly, arc welding, and others [3]. Furthermore, as the power of vision systems steadily increases, their role in the industrial production will grow more prevalent and become an integral part of robotic controllers and CAD/CAM systems [5].

Internal sensors, such as encoders, are widely used in a conventional robotic system. The internal position error of robotic manipulators can be corrected using feedback information from internal sensors. In order for robots to satisfactorily fulfill many potential missions and applications, it is necessary to incorporate advanced technologies including external sensors. External sensory information is critical in many

inspection and assembly applications, for operations in unknown or changing environment, and for maintaining robot safety [6].

Complex robotic tasks such as automated assembly, clearly depend on the vision system to pick/place objects, install components, and monitor product quality. For example, applications may include compensating for small positioning errors to grasp objects moving on a conveyor belt, tracking a seam in arc welding, or, more generally, adapting to environmental uncertainties. Typically, visual sensing and manipulation are combined in an open-loop fashion: "looking" then "moving". Visual sensing is primarily used for recognizing, locating, and inspecting stationary parts. A recent trend is to integrate vision sensors into the servo control loops instead of using vision sensors as merely sources of data. Currently, image-processing equipment has now reached the stage in which vision can be used to generate a feedback signal to control position and orientation of the manipulator end-effector in real time. Visual servo techniques allow the robots to manipulate and track a randomly moving part without any previous knowledge of the part's placement or motion. It increases the overall accuracy of the system and makes it possible to design intelligent control systems for robotic manipulators [2]. Therefore, a robot can update its world model by continuously analyzing visual information and immediately generating the robotic control commands. Finally, the precise manipulation with imprecise models of both robots and environment is achieved. Two applications of a robotic vision system follow.

Vision Adjustment

Based on internal sensors, robotic manipulators are driven to the teaching position

(playback procedure and point-to-point movement) and the vision system is only implemented to adjust the final position. The process is depicted in Figure 1. The system obtains actual world positions of the object and the robotic end-effector and compares them with previously taught positions. The deviations of world coordinates are transferred to the values of robotic joint coordinate system. The joint controllers drive the robotic manipulator and makes the end-effector move to the position where the object is located. The internal and external position errors are corrected, but robotic operation speed is reduced.

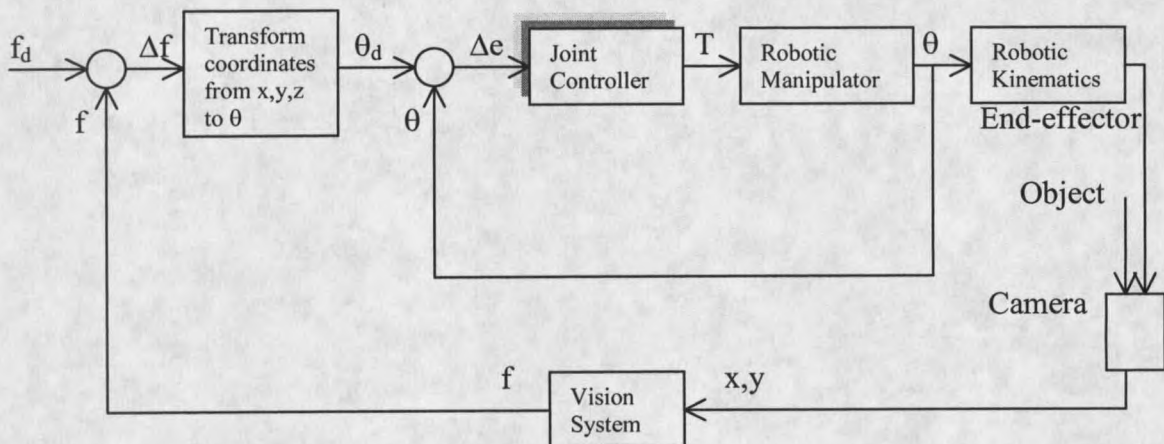


Figure 1. Robotic Vision Adjustment Process

Vision Tracking

With vision tracking the system functions throughout the whole procedure (real-time continuous path tracking, i.e. visual servo control system). For an illustration of the process see Figure 2. The system obtains the actual world positions of the object and robotic end-effector and compares them with the desired moving path. Visual servo control depends on the deviation and corrects the moving path on time. An optimal

path is achieved.

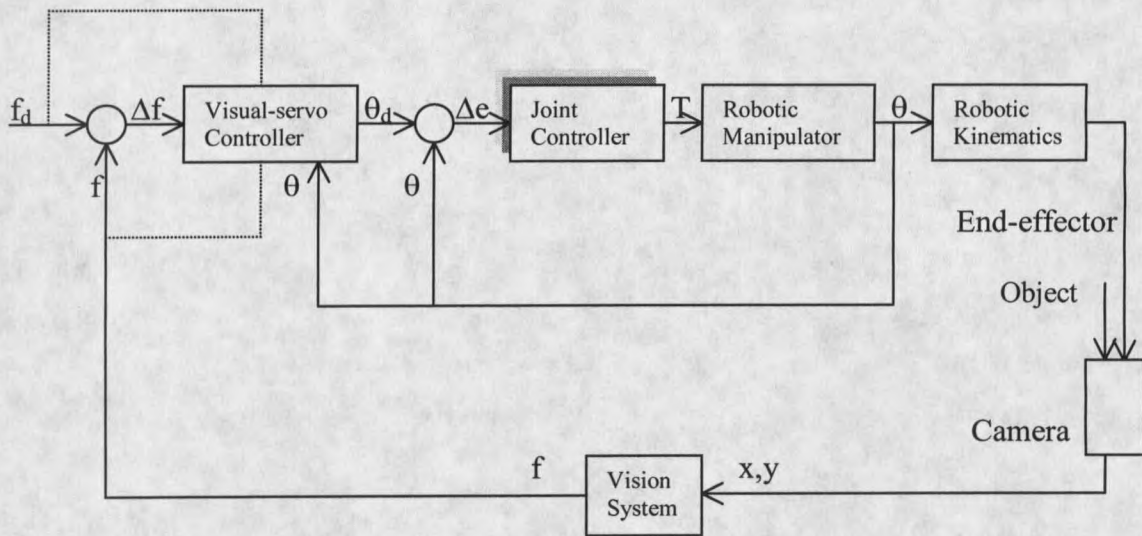


Figure 2. Robotic Vision Tracking Process

To smooth the manipulator's motion, a trajectory planner is required to alter the trajectory of the robot as it receives updated vision information, requiring solution of robotic inverse kinematic equations. For the vision subsystem, image feature extraction time and image interpretation time have been identified as two major limiting factors affecting the robotic operation speed and the performance of most visual servo control systems. Coherence between the sampling frequency at the image level and the bandwidth of the system must be controlled. In addition, application of visual servo, to some extent, depends on image processing algorithms such as filtering, on dedicated real-time control architectures, and on camera calibration.

Review of Selected Robotic Vision Literature

Human beings are capable of assembling a group of diverse parts to produce either a finished product or a subassembly because of their ability to utilize good eye-hand coordination in conjunction with the sense of touch. However, these jobs may be extremely tedious because of their repetitious nature. As such, assembly operations represent an attractive application of robots [7]. Unfortunately, it takes more to solve a manufacturing problem with the help of a robot than to merely purchase the device and expect it to do meaningful work on the factory floor [8]. Accessory systems are also involved. A vision system can be introduced, but the integration of the vision system should not affect the robotic execution time and increase the possibility of demanding more external equipment (such as fixtures) because it will affect the flexibility of assembly line.

Since the early work of Shirai and Inoue [9], who described how a visual feedback loop can be used to correct the position of a robot to increase task accuracy, considerable effort has been devoted to the visual control of robot manipulators [2]. Conventional visual sensing and manipulation are only built in an open-loop fashion. Introducing a visual servo technique will further improve the accuracy of robotic system.

In addition, vision-based path planning is a well-established discipline, but the idea of combining image space feature for path planning with visual feedback has not been adequately explored [2]. Susan Gottschlich et al. [10] presented the assembly planning problem. They pointed out when the clearances allowed between parts in an

assembly are small relative to the uncertainties, which is often the case, it is necessary to develop a fine motion plan for assembling the parts. Fine motion planning relies on motion with sensory feedback to overcome the uncertainties as the assembly operation proceeds. Furthermore, these planners often either rely on models that are too simplistic or use reasonable models but are too computationally burdensome. Of particular importance is how best to represent model uncertainty.

Vision is also popularly used in quality assurance. Billibon H. Yoshimi et al. [11] conducted research on visually-guided grasping and manipulation. They emphasize that visually monitoring a task will give us the feedback necessary both to perform the task, as well as to gauge how well the robot performed the task, or if an error has occurred.

A vision system is required to extract the information needed to perform the servo tasks [2]. Some features such as "corners" or "edges" are used to indicate the presence of object boundaries or surfaces in making an image. However, not all pixels in the image are of interest, and computation time can be greatly reduced if only a small region around each image feature is processed. So a promising technique for making vision cheap and tractable is to use window-based tracking techniques [12].

W. Hattich [13] discussed model-directed recognition. Recognition was based on a comparison of contour elements of an image with a reference contour which corresponds to a geometrical model of object data for a given image. Contours were described in terms of straight lines. Comparison was done by iterative construction. Therefore, only the contour lines of an object are usually analyzed.

Expert systems were introduced by Suresh K. Devarajan [14] to fulfill object

identification for robotic applications. An object identification expert system was developed to classify the objects. The expert system employed a database of classification spectra as a basis of reference. The reference database was built using a variety of known objects. The images of target objects were captured by digital camera and processed to remove signal noise and to extract the contour of the object. So the image processing was not affected seriously by environmental conditions.

Much of the visual servo control literature deals with the kinematics of visual control. One of the important futures was solving the robotic inverse kinematics problem. Novakovic and Nemeč [15] presented a new closed-loop algorithm for solving the inverse kinematics problem. It was easy to achieve coordinate transformation using the sliding mode principle and a Lyapunov-like concept. An attractive feature of this algorithm over the Newton-Raphson method is that its computational burden is reduced. It requires no Jacobian matrix inversion and thus avoids the numerical instabilities associated with matrix singularities.

Cooke and Good [16] also introduced the concept of visual dynamic control which is concerned with dynamic effects due to the manipulator and machine sensor which limit performance. They also mentioned feedback control issues such as choice of compensation, axis position, and velocity of torque controlled inner-loops within the visual servo system.

Justification of Robotic Vision System

Justification of a robotic vision system should be based on the fact that the vision

system can improve selective areas of robot performance without affecting other areas. Vision adjustment and tracking methods improve robotic performance to adapt to changing environment, but they may omit the randomness of the external environment. During the robotic work cycle, sometimes vision system is needed; sometimes it is not needed for stable conditions. Vision tracking is based on the visual servo controller and execution of its control algorithms creates a computation burden for the robotic system and therefore reduces its operation speed.

In this thesis, a new design for a robotic vision system, the automatic vision switching system (AVSS), is presented. The design concept was derived from the concept of Just-in-Time that calls for the vision system to function just in the time of need. Design of a parallel control structure guaranteed that robotic operation and image processing were completed at the same time. AVSS considered the randomness of some situations and avoided the computation burden that vision tracking brings. So it overcame the side effects of vision adjustment and tracking methods in robotic assembly where random variation is present.

CHAPTER 2

PRESENTATION OF PROBLEM

Many robotic assembly tasks are attempted without external sensing, assuming an absolute world model that never changes. For example, in many pick-and-place operations, objects are always in a previously known absolute position and orientation. However, many objects are normally presented to the robots in an uncertain manner and their positions and orientations in the robotic world space are rather random and must be measured on-line [17,18]. Applications of robotic vision systems often encounter problems of inconsistent work pieces.

In one storage battery production line, installation of caps on a storage battery was a manual operation several years ago. Repetitive long-time work caused workers to become bored, with the result that installation speed and the work consistency were reduced. Manufacturing engineers turned to automatic installation. They introduced a SCARA type robot, as well as a material feeder and a work piece fixture. The assembly workstation is displayed in Figure 3. Manual teaching and automatic playback were used as the programming medium. After a series of pre-installation tests, they found that results were not as good as anticipated. Further study disclosed that, due to attributes of plastic storage batteries, deviation of a hole's position on a storage battery happened randomly, the result of inconsistent geometry or fixture/battery deflection. The situation

is depicted in Figure 4. A hard coded robot would fail miserably for this kind of random installation [10]. By implementing a robotic vision system, this problem can be addressed, but the execution time of the robotic assembly cycle is adversely increased.

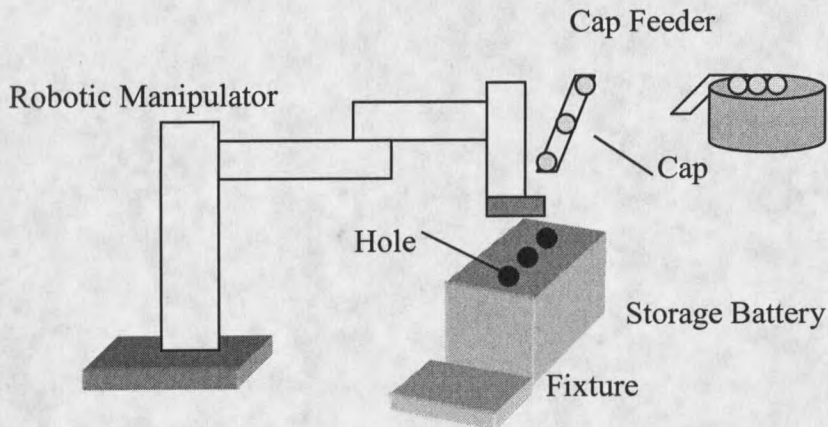


Figure 3. Battery Cap Assembly with Robot

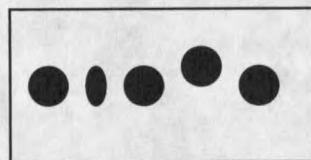


Figure 4. Inconsistent Features and Locations

Vision adjustment cannot deal with on-line path planning problems. The path of the robotic gripper is not optimal. Vision tracking is a multi-faceted process that includes high-speed image processing, robotic kinematics and dynamics, control theory, and real-time computing [2]. Extensive calculations burden the robotic system and challenge efficiency, especially for the above mentioned battery cap installation. With a vision

adjustment system, vision correction for the final position is made without considering the optimal path of motion. Designing and testing a better vision control system to reduce robotic execution time and increase control precision, thus increasing quantity and quality, is the focus of this research.

CHAPTER 3

DESIGN OF AUTOMATIC VISION SWITCHING SYSTEM

The primary objective of this research was to design an Automatic Vision Switching System (AVSS) for battery cap assembly. The system changed currently used vision-based assembly procedures, i.e. vision adjusting and tracking methods. Due to the randomness of inconsistent geometry or deflection of the storage battery, vision induced correction was not needed in all situations. Based on the concept of Just-in-Time, the AVSS functions just at the time of need as depicted in Figure 5.

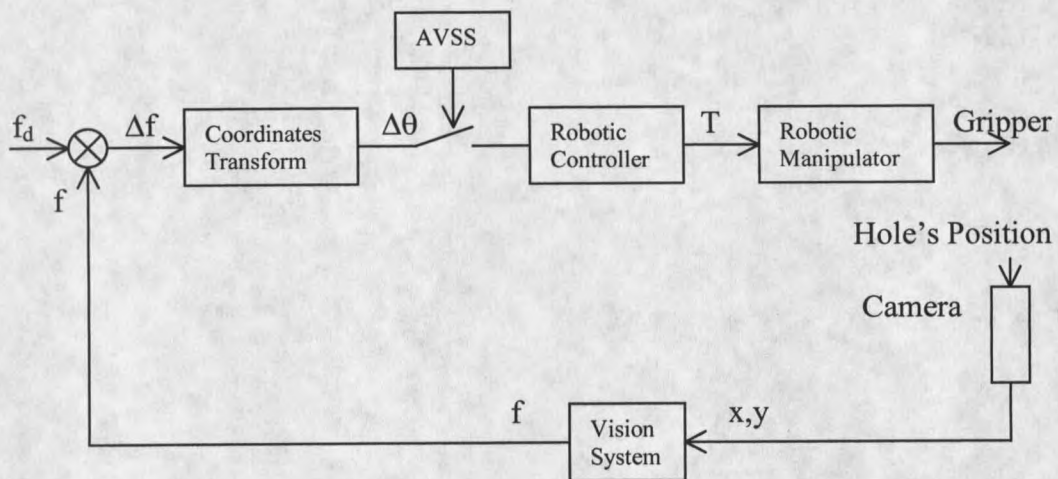


Figure 5. Configuration of Robotic Control System with AVSS

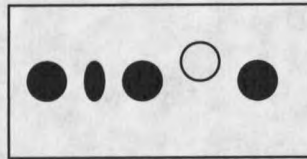
AVSS Theory

AVSS is a control system that integrates a vision system into a robotic assembly system. It consists of an interface between the robot and the vision system, the control algorithms of the robotic system, and the image processing techniques.

AVSS commands the robot to complete the off-line teaching procedures and records the battery's hole locations and the standard feature characteristics of the hole's image. During routine operations, when the AVSS finds that the battery's hole locations need correction, it automatically integrates the vision system with the robotic system and provides the position correction. When the robotic controller receives the commands from AVSS, it will change its previously taught position and the path to get to the next hole's deviated position. The process is explained below:

- 1) Based on placing caps on a standard battery, the robot is manually programmed for a standard configuration (feature location and distortion).
- 2) Based on the standard configuration, the image of holes on the battery is captured by the AVSS through a camera and stored in the frame grabber.
- 3) When the robotic installation is started, the AVSS will be used to determine if the next cap installation needs vision correction. If the AVSS finds a significant difference between the taught position and the required position of the hole's image, a vision-based path for the robot is generated automatically. If the geometric distortion of the hole exceeds the installation range, the AVSS will invoke an emergency alarm. Otherwise, the installation is

accomplished without vision correction. The arrangement is depicted in Figure 6.



- Vision correction not required.
- Vision correction required
- ◐ Distortion requiring emergency alarm

Figure 6. Location of Battery Holes for Vision Correction

Integration of the AVSS will further improve efficiency in the computer vision system in robotic assembly application by reducing computing requirements and execution time. In other words, the AVSS is a trade-off between full-time vision-adjustment and vision-tracking methods.

A robotic system with vision performs many calculations for image processing, robotic dynamics and kinematics, servo control, etc. For traditional vision-adjustment procedures, the vision system is used in the final position adjustment without considering deviation from the taught path. If correction is required, the vision-adjustment system and the AVSS will provide a different corrected path to the next hole's location, as shown in Figure 7. AVSS achieves an optimal path and therefore reduces the execution time of the robotic work cycle. Although the vision tracking technique can guarantee an optimal path, the visual servo control algorithm requires many calculations. On the other hand,

vision tracking increases execution time of the robotic work cycle as that of the vision adjustment system.

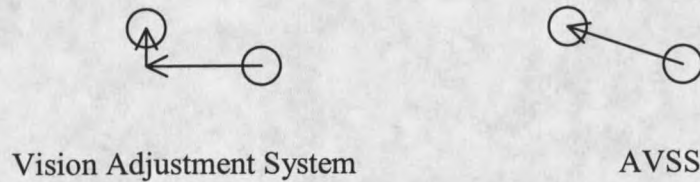


Figure 7. Path Correction

Design of AVSS

Threshold and Image Filter

This system incorporates image processing techniques based on pixel gray levels. The pixel and its intensity values, the gray level values, are the important representations of a digitized image on a computer monitor. In battery cap installation, it is desired to segment a hole's image from its background regions through threshold techniques. Based on the standard features of holes, the threshold value was set as T (a histogram can be used to find this value). The constant number of pixels for a standard hole (N_C) can be used to identify deviation and image noise.

Let $f(x,y)$ represent the gray level value of pixel (x,y) within a window of the window-based tracking technique [12], as shown in Figure 8. Then, within the hole $f(x,y) \geq T$ is the necessary condition. Two points (x_1,y_1) and (x_2,y_2) are used to define the window, hence the total number of pixels within the window is $(x_2-x_1)*(y_2-y_1)$. The ratio $R_C, N_C / (x_2-x_1)*(y_2-y_1)$, can be also used to represent a standard measure of

this hole's image and to identify deviation.

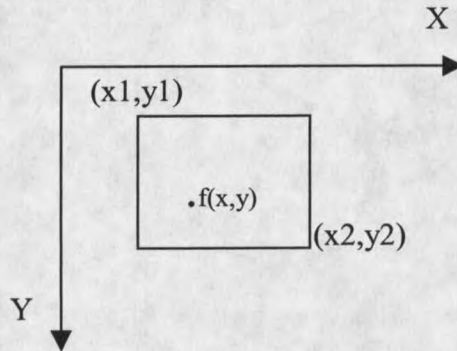


Figure 8. Definition of Pixels within a Window

If noise is present in the image capturing process, N_C or R_C can be used to define the noise level and whether or not the image should be filtered. For example, with error level ϵ and a hole with N pixels, the comparison between the actual number of pixel (N) and the standard number of pixels (N_C) is:

If $|N - N_C| \leq \epsilon$ then filtering is not needed;

If $|N - N_C| > \epsilon$ then filtering is needed.

The noise comes from many external factors such as lighting conditions, camera vibration, or robotic movement (eye-in-hand system). Noise makes it difficult for the AVSS to identify the boundary points of holes, so edge-detection filters are used.

The edge detector described by Michael C. Fairhurst [4] utilizes a window which is moved across successive image points computing a value at each point which, based on local neighborhood information, represents some measure of the probability that the point in question is, in fact an edge point. An advantage of this method is that it can detect

where the edge is most likely located and also can distinguish where a discontinuity is caused by the presence of an edge or by noise-induced intensity change.

In the AVSS, a Sobel operator [4] was used to implement edge-detection filtering. The Sobel operator accomplishes some degree of inherent neighborhood smoothing in addition to its primary function, returning a gradient value. The gradient value calculation is given below. Figure 9 describes the processing window.

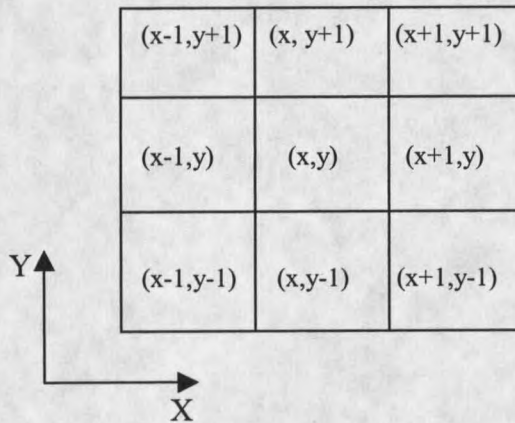


Figure 9. Definition of Processing Window for Sobel Operator

The gradient value is given as follows:

$$G(x,y) = \sqrt{\left[\left\{ (f(x+1,y+1) + 2f(x+1,y) + f(x+1,y-1)) - (f(x-1,y+1) + 2f(x-1,y) + f(x-1,y-1)) \right\}^2 + \left\{ (f(x-1,y-1) + 2f(x,y-1) + f(x+1,y-1)) - (f(x-1,y+1) + 2f(x,y+1) + f(x+1,y+1)) \right\}^2 \right]}$$

The gradient value must be tested against an operator-chosen 'edge threshold', δ :

$G(x,y) > \delta \Rightarrow \text{point}(x,y)$ is an edge point

$G(x,y) \leq \delta \Rightarrow \text{point}(x,y)$ is not an edge point

For practicality, the choice of δ must effect a compromise between ensuring that all real edges are identified, while rejecting false edges caused by noise, spurious contrast changes, etc. [4].

Geometry Measurement of Holes

The centroid of an object is an important geometric parameter. The centroid equations are given below [19]:

$$X_C = (1/A) \sum_{i=1}^N X_i \quad Y_C = (1/A) \sum_{i=1}^N Y_i$$

Where X_C and Y_C are the centroids, A is the area of one hole, N is the number of pixels within the hole, and X_i and Y_i are the coordinates of the i th pixel.

In the battery cap installation, the purpose of image processing is to find the centroid value of each hole, and to determine if distortion is serious. All pixels within a hole are processed by the centroid equations.

Two arbitrary scanning lines, $L1$ and $L2$, are used to define four boundary points of the hole (P_1 , P_2 , P_3 , and P_4) as shown in Figure 10. From points P_3 , P_1 and P_2 , center coordinates ($P_{c1}(x_{c1}, y_{c1})$) of the arc are calculated. From points P_1 , P_2 , and P_4 , center coordinates ($P_{c2}(x_{c2}, y_{c2})$) of this arc are determined. Detailed calculations are given in Appendix A.

The next step is to set the hole's distortion limit, ϵ_d as seen in Figure 11. The ϵ_d is defined as the distance between the two centers (P_{c1} and P_{c2}). If $|P_{c1} - P_{c2}| > \epsilon_d$, then

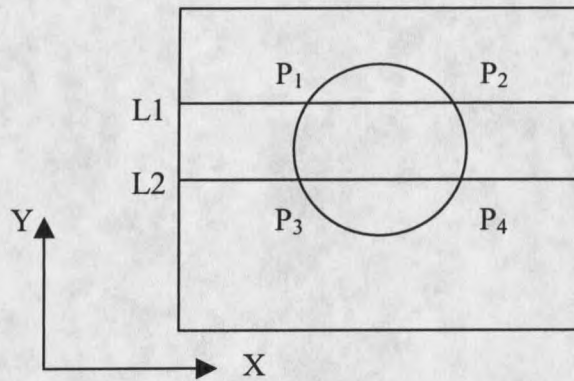


Figure 10. Scanning Lines for Solving Centroid Values of Holes

robotic movement is stopped and the alarm is triggered. If $|P_{c1} - P_{c2}| < \epsilon_d$, then the centroid coordinates of the hole ($P_c = (P_{c1} + P_{c2})/2$) are used to direct the robot to this location.

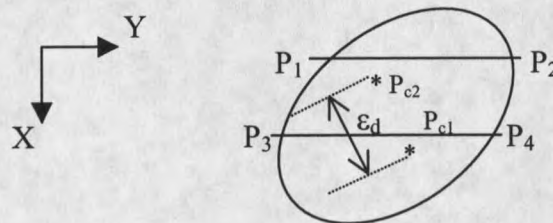


Figure 11. Distortion Situation of Holes

The scanning line method for boundary points proved to be slow because it processes more than 100 pixels that comprise the scanning line. A simpler method, middle value calculation, was designed and implemented in the AVSS method. Use of this method reduced the search to find the four boundary points in Figure 12. Suppose P_1 and P_3 are the end points of a scanning line, then the middle point $P_2 = (P_1 + P_3)/2$. Scan

is started from P_1 , P_2 and P_3 to search for boundary points. For the left boundary point, $P_4 = (P_1 + P_2)/2$. If P_4 is within the hole (decided by gray level value), then choose P_1 and

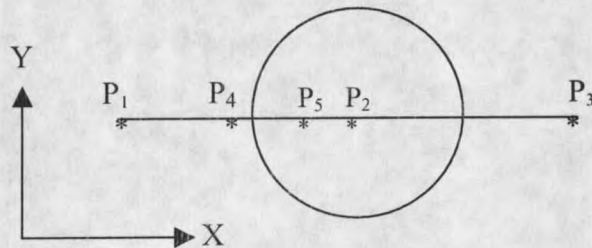


Figure 12. Middle-Value Method for Boundary Points Search

P_4 as the next two points to solve $P_5 = (P_1 + P_4)/2$. If P_4 is outside the hole, choose P_4 and P_2 as the next two points to solve $P_5 = (P_4 + P_2)/2$. In this example, P_4 is located outside the hole and P_5 is inside the hole, hence P_4 and P_5 are chosen next and $P_6 = (P_4 + P_5)/2$. Repeat the same procedures until $(P_{n+1} - P_n) = 1$, then P_{n+1} is set as the left boundary point on the chosen scanning line. For the right boundary point, the scan is started with point P_2 and P_3 and repeats the procedure.

Vision Correction

If the hole is not excessively distorted, but requires vision correction, the robotic control system is implemented. For SCARA-type robots, two joints decide the Cartesian coordinates of the gripper center (i.e., set the gripper's orientation perpendicular to Y axis of world coordinate system) depicted in Figure 13.

The forward kinematics equations are:

$$X=(L1+L2\cos(\theta_2))\cos(\theta_1)$$

$$Y=(L1+L2\cos(\theta_2))\sin(\theta_1)$$

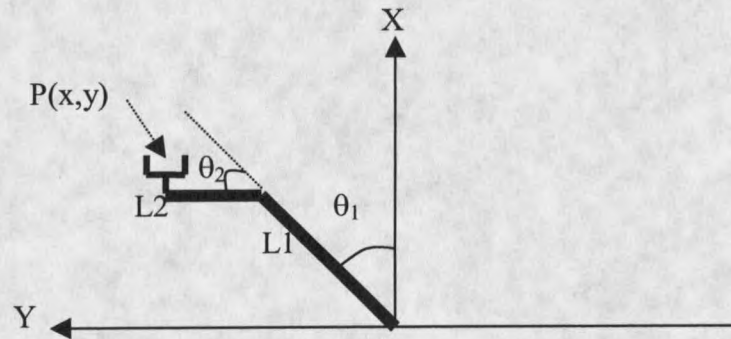


Figure 13. Configuration of Kinematics for Robot

Position $P(x,y)$ of the manipulator in terms of world coordinates is normally known. However, the joint coordinates must be obtained in order to make the move. An inverse kinematic procedure is used to find the coordinates; equations are given in Appendix B.

After obtaining deviation values of holes in world coordinates, the robotic controller transforms these values into robotic joint values. Then the robotic controller drives the manipulator and makes the gripper move to the actual holes' locations. Effectiveness of the AVSS was demonstrated through experiment carried out in the CIM Lab of the Department of Mechanical and Industrial Engineering.

CHAPTER 4

DESIGN OF EXPERIMENT

An AdeptOne Robot and Vision-EZ system, as well as other equipment, were used to demonstrate the effectiveness of the Automatic Vision Switching System.

Configuration of Hardware

AdeptOne Robot

Adept Robot is a SCARA type robot with four servo control axes and pneumatic control gripper. It includes manipulator, robot controller, MCP (manual control pendent), and VAL-II real-time operation system.

The robot controller includes many functions such as teaching, playback, interpolation or trajectory generation, world coordinates and joint coordinates control methods, etc. It also provides many parallel and serial I/O ports for user interfaces. Figure 14 displays the configuration of the robotic manipulator. Joint 1, Joint 2 and Joint 4 control the x and y coordinates of the gripper center; Joint 3 controls the z coordinates of the gripper center.

Vision-EZ System

Vision-EZ vision system is produced by Data Translation Inc. It includes a frame grabber (DT55), and image processing software libraries. It can complete image

capturing, storing, and displaying through an ordinary video camera. It grabs images

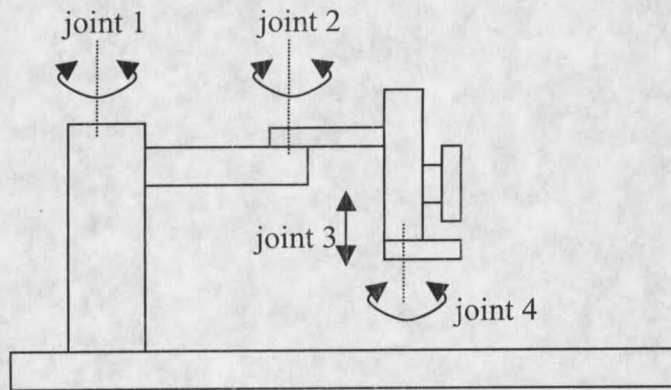


Figure 14. Configuration of Adept Robot Manipulator

with the speed 1/30 seconds and supports square pixel resolution and 256 gray levels. Due to its powerful software functions, Vision-EZ can fulfill real-time image processing and provide user-friendly accessible interfaces.

Other Equipment

Solid Camera: JE2362A

Personal Computer

Robotic Working Table

Storage Battery Model

Design of Experiment

AVSS theory was described in Chapter 3. The robotic vision system, used for installation of storage battery caps, is given in Figure 15.

The camera is installed parallel to the gripper. When the robot is installing a cap on a given hole, the vision system captures and analyzes the next hole's position to

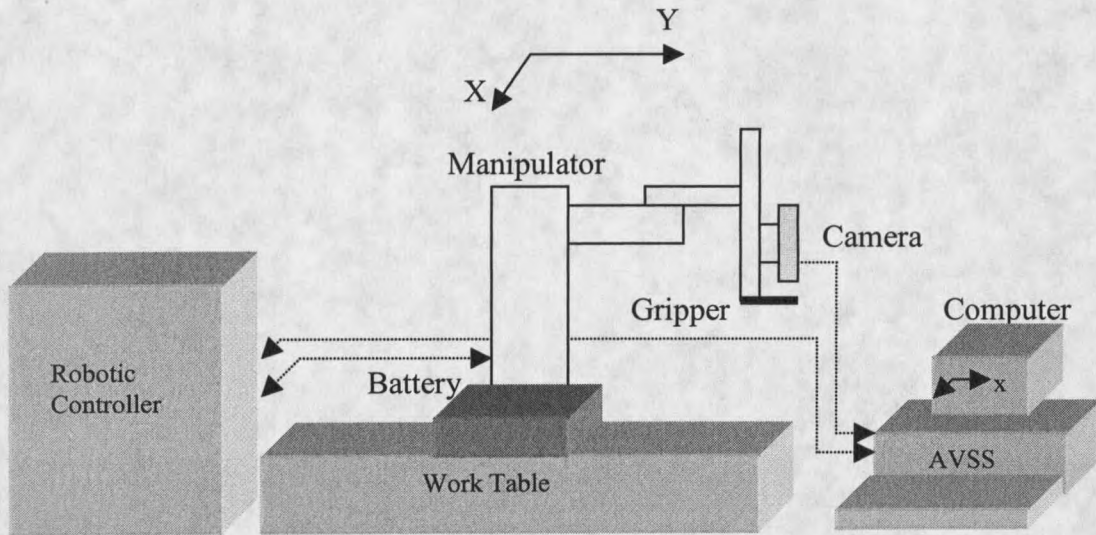


Figure 15. Configuration of Robotic Vision Assembly System

decide if the hole needs vision correction because of its deviation. If so, the AVSS automatically generates compensation for the hole's position and path. The motion path to the hole will be along the newly corrected trajectory. If not, the robot will keep the previously taught path to get to the next hole's location.

Experimental Procedures

- Step 1: Place a standard storage battery model on the work table. Top view of the battery model is depicted in Figure 16.
- Step 2: Use MCP to move robotic gripper to different holes' and reference point's locations and record the gripper center coordinates as the standard

reference coordinates listed in Table 1.

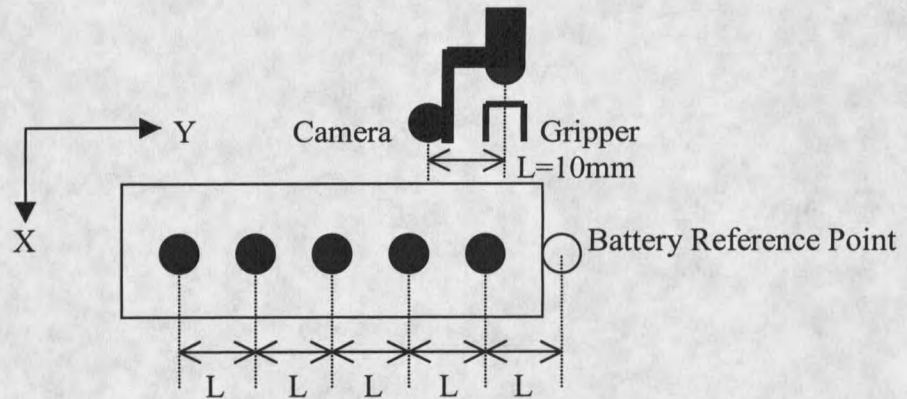


Figure 16. Battery Model and Gripper-Camera Configuration

Table 1. Standard Positions and Orientation of Holes

Holes	X(mm)	Y(mm)	Z(mm)	y(degree)	p(degree)	r(degree)
Hole1	526.09	245.89	809.89	0	180	46.62
Hole2	526.09	137.42	809.89	0	180	46.62
Hole3	527.31	28.44	809.89	0	180	46.63
Hole4	527.65	-80.32	809.89	0	180	46.60
Hole5	529.59	-190.01	809.89	0	180	46.62

Step 3: Camera calibration involves finding the relationship between robotic world coordinates and image pixel coordinates on the computer monitor. If a hole is positioned on the robotic X axis, the hole's image is positioned on the X axis of the monitor coordinate system; if a hole is positioned on

the robotic Y axis, the hole is positioned as a mirror image on the negative Y axis of the monitor coordinate system. So if the hole's image deviates Δx from its standard position, the robot should move Δx to correct the deviation; if the hole's image deviates Δy , the robot should move $-\Delta y$ to correct the deviation. The relationship between the two coordinate systems is given in Figure 17.

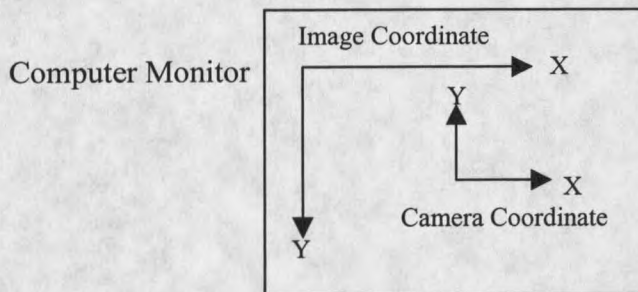


Figure 17. Relationship of Camera Coordinate and Image Coordinate

Because the camera is installed parallel to the gripper, the camera center's coordinates can represent the gripper center's coordinates in the world coordinate system. Robotic world coordinates and the hole's image position on the monitor are obtained through scanning and image processing. The resolution of the image on the monitor (640*480 pixels) is obtained as:

$$\text{Ratio}_x = 0.251 \text{ mm/pixel}$$

$$\text{Ratio}_y = 0.234 \text{ mm/pixel}$$

Step 4: Record center coordinates for each battery hole obtained from image

coordinates on the monitor for future reference. The data are given in Table 2.

Table 2. Image Position (pixel number) of Standard Hole Locations

Holes	X(pixel)	Y(pixel)
Hole1	327	249
Hole2	325	245
Hole3	329	245
Hole4	324	246
Hole5	334	249

After applying power to the robotic controller, the actual values may change, as the result of camera vibration, battery model deviation, and other factors. Consequently, the reference values must be adjusted by a correction factor if needed. To determine if adjustment is needed, the gripper is moved with MCP to the reference point. The current gripper center coordinates are compared with previously recorded standard values. If there are deviations with Δx and Δy , all holes' standard values are corrected with Δx and Δy respectively. Figure 18 describes the situation.

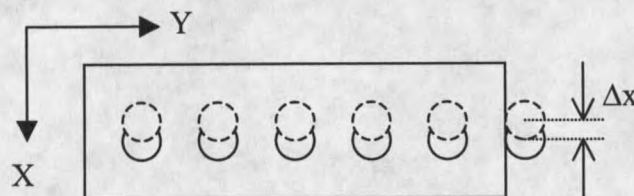


Figure 18. Location Adjustment of Standard Holes

Step 5: Apply the robotic control and image processing software. A parallel processing technique was introduced for the robotic installation and image processing. Whenever vision correction is needed, the corrected position values are transmitted to the robotic controller.

The robotic control software completes a "playback" process on the corrected locations of holes. During battery cap installation, the robotic controller communicates with the computer vision system and identifies if next hole's location needs vision correction. The robotic control program was developed with VAL-II robotic language. Figure 19 gives the flowchart of the program. Program source codes are attached in Appendix C.

The image processing program implements the capturing and processing of each hole's image. The deviation values of a hole's location are obtained and transmitted to the robotic controller. The program was developed with Microsoft Visual C++ 1.0. Figure 20 gives the flowchart of program. Program source codes are attached in Appendix D.

To accommodate image processing, limiting values must be established for deviation identification criteria. Deviation of the hole's location should not exceed the feature-based window. The limit value was set at ± 10 mm as described in Figure 21. Because the AVSS captured the present hole's location based on the previous hole's location, if the previous hole deviated from the standard location, that deviation value was deducted from the coordinates of the present hole's image. If the deviation value

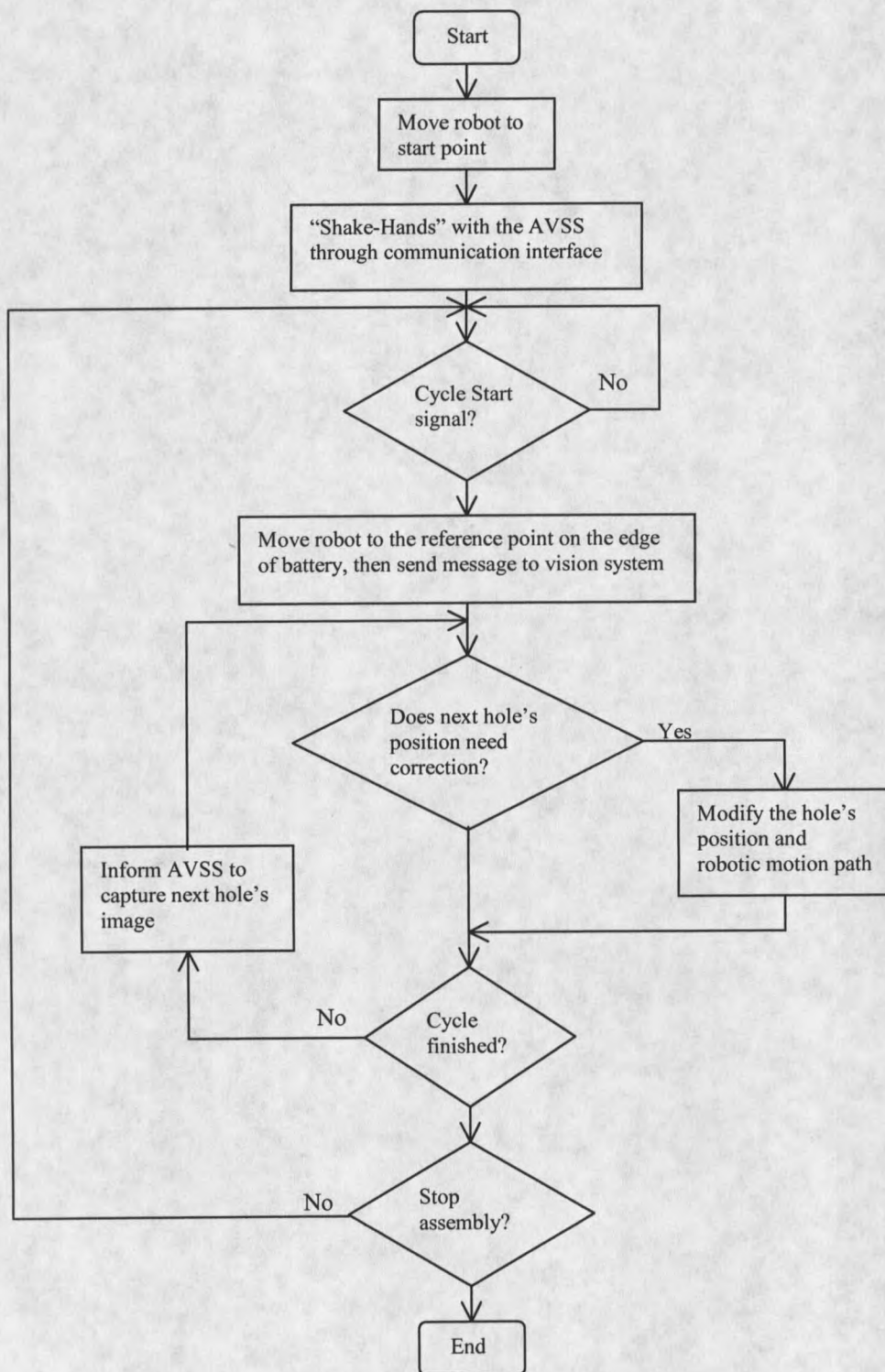


Figure 19. Flowchart of Robotic Control Software

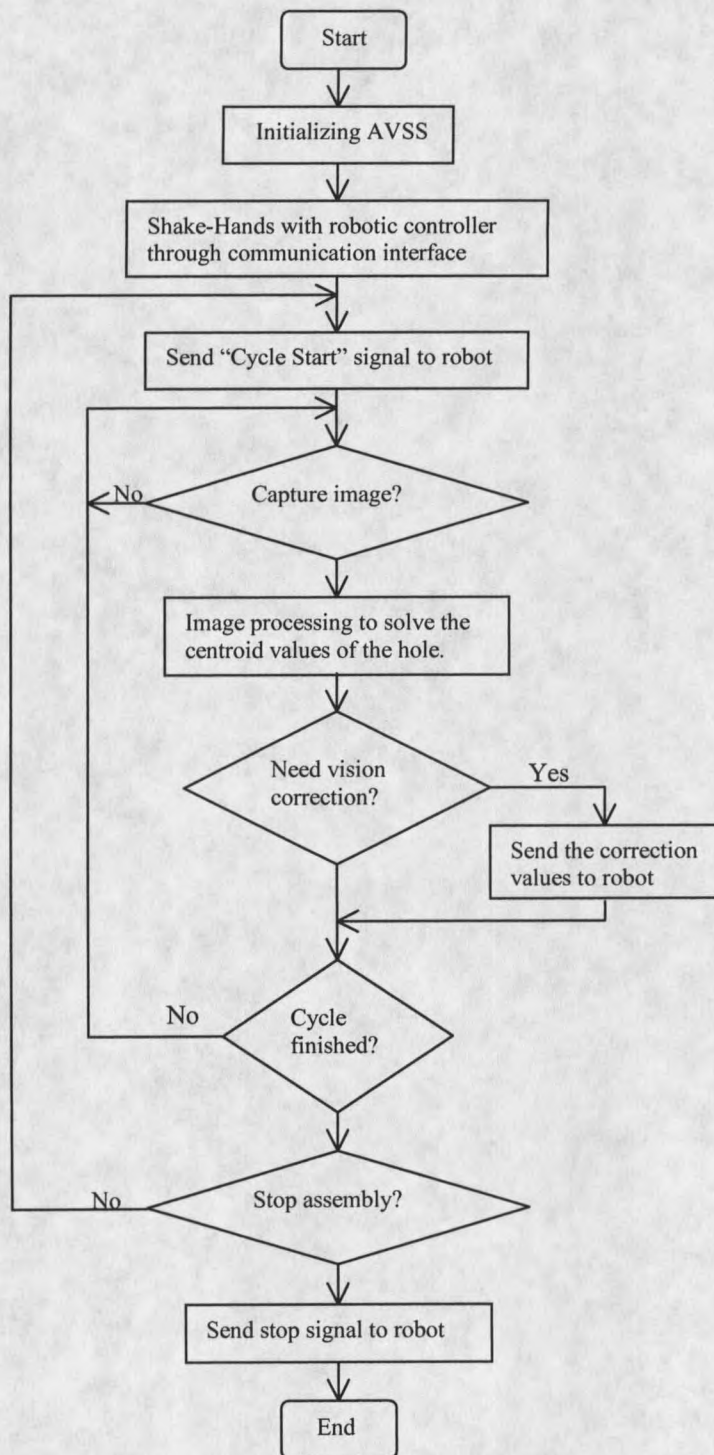


Figure 20. Flowchart of Vision System Software

exceeded the limit, an alarm was generated.

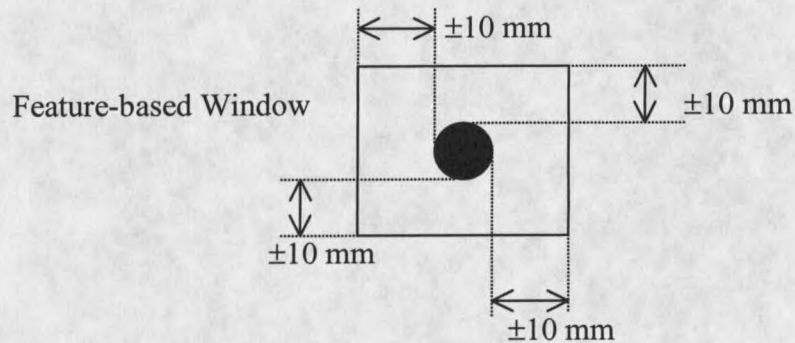


Figure 21. Deviation Limit of a Hole Location

A limiting value for a hole's geometry distortion was also set at ± 10 mm, which corresponds to the distance between two arc's centers P_{c1} and P_{c2} (see Figure 10 for scanning line method for solving centroid value). If $|P_{c1}-P_{c2}| > \epsilon_d = 10$ mm, the AVSS triggered an alarm and stopped the installation.

It was found through experimentation that a threshold value of 150 (0 is white and 255 is black) for binary image edge detection gave the best results.

Results and Discussion

Boundary Points

The purpose of image processing in battery cap installation is to identify the position of a hole and its deviation. The scanning line method and middle value calculation method were used to find boundary points. The feature-based window defined on the computer monitor was (220, 170) to (400,330). Table 3 gives the comparison of the calculation results for the above two methods.

In comparing the two methods, it was found that the middle value calculation method greatly reduced the number of iterations in searching for boundary points; although it did have one plus/minus pixel deviation corresponding to ± 0.25 mm error in terms of robotic world coordinates.

Table 3. Comparison of the Two Methods for Boundary Point Search

	Pixel Location (x axis)		Searching Pixels' numbers	
	<u>Scanning</u>	<u>Middle</u>	<u>Scanning</u>	<u>Middle</u>
Hole1	266	267	180	6
Hole2	271	271	180	6

Edge Detection

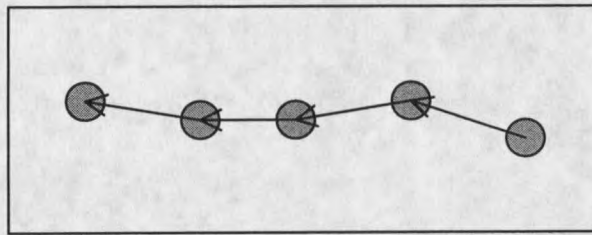
As previously mentioned, a Sobel operator was used to verify boundary points. The 'edge threshold' value, δ , was chosen as 150 (255 gray levels), the same as the threshold value set in the binary image processing. If the gradient value, $G(x,y)$, on a boundary point is greater than 150, the boundary point was correctly identified.

Identification of Distortion

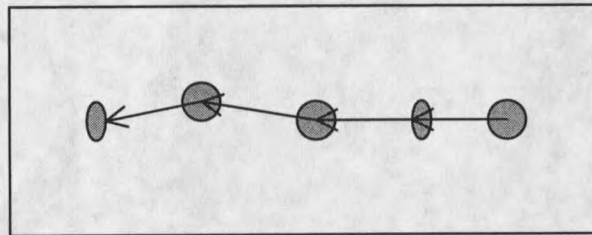
Two different methods were used to detect the distortion of a hole. If a hole's image was beyond the feature-based window, then the AVSS triggered an alarm. The ratio R_C mentioned in Chapter 3 was used to find this distortion. Distortion may also occur in a hole's geometry. This distortion was identified with the value of $|P_{c1} - P_{c2}|$ and its limiting value, ϵ_d .

During the experiment, many different instances of distortion were set manually.

Figure 22 displays the results. The actual path is also depicted. In actual operation, if an alarm signal is triggered, the operator must stop the installation process and readjust the fixture's position to remove or reduce the deviation and then continue the process. In the experiment, this procedure was omitted.



(a)



(b)

Figure 22. Robotic Motion Path with AVSS

CHAPTER 5

CONCLUSION AND RECOMMENDATION

Due to randomness in some kinds of robotic assembly tasks, previously developed vision adjustment methods and currently used visual tracking techniques don't adapt robots to uncertain external environment efficiently. Integrating these methods with the robotic system reduced the operation speed and/or placed excessive computation burden on the robotic system. Designing a vision system to countermand this difficulty should improve random assembly accuracy without affecting other robotic performance.

Through design and implementation of an Automatic Vision Switching System (AVSS), many advantages were obtained. For installation of storage battery caps, the AVSS guaranteed assembly accuracy and further improved the efficiency of computer vision in robotic assembly applications by reducing the computing requirements and the execution time that vision adjustment and vision tracking methods normally bring. In other words, the AVSS is a trade-off between full-time vision adjustment and vision tracking methods. It generated an optimal motion path and increased operation speed in a manner by which the vision adjustment method is not capable. Furthermore, AVSS omitted the visual servo control algorithm which vision tracking requires, so that execution time was reduced.

Many image processing methods were included in AVSS. For battery cap

installation, one purpose was to find centroid values of holes on a battery. Within the feature-based window that a hole is located in, the image processing time was reduced greatly. Standard features of a hole's image were used to detect if noise was serious and if filters should be introduced to remove noise. Two kinds of deviation were addressed: (1) deviation of the hole's locations outside the feature-based window as the ratio (R_c) of hole's pixel number over window's pixel number, and (2) geometric distortion of the hole as the distance of two arc centers ($|P_{c1} - P_{c2}|$).

In solving for centroid values of holes, two methods were compared to search for boundary points of holes: a scanning line method and a new middle value calculation method. The latter method needed only about 6 steps to find a boundary point compared with 280 steps for the scanning line method. Considering the effect of noise in image capturing, a Sobel edge detection operator was introduced to guarantee the correct solution for boundary points of the holes. The arc center method was used to solve centroid coordinates based on four boundary points and mathematical calculations instead of the pixel summarization method.

The experiment was implemented based on an AdeptOne robot and Vision-EZ system. The gripper-camera parallel structure and eye-in-hand configuration made it possible for the robot to be installing a cap and for the AVSS to be processing the next hole's image simultaneously. The parallel processing structure guaranteed improvement in real-time operation. During the installation process, when deviation happened, the AVSS provided deviation values and transmitted them to the robotic controller. The robotic gripper was moved to the next hole along the optimal path. Each time a deviation

value exceeded the limit, the AVSS triggered an alarm to stop the robot. The random deviation of the hole's locations were perturbed manually and the corrected path and alarm system demonstrated the effectiveness of AVSS.

In short, AVSS integrates a vision system with an assembly robot and functions just at the point in time that vision system is needed for robotic movement and assembly. Thus, the AVSS provides an additional choice as a method of vision application in assembly in a cost-effective manner.

Due to complexity of battery cap installation, force during installation should also be considered. Future experiments will include a force sensor in the gripper and building a knowledge-based software for integration into the AVSS. Thereby, caps can be installed correctly without damaging the battery case.

With development of computer technologies, visual servo control will be improved and the calculation of control algorithms will become faster and faster. The development of AVSS has contributed to the body of knowledge in this field.

LITERATURE CITED

LITERATURE CITED

- [1] Stan Gibilisco, Editor in Chief, *The McGraw-Hill Illustrated Encyclopedia of Robotics & Artificial Intelligence*, McGraw-Hill, New York, 1994.
- [2] Seth Hutchinson, Gregory D. Hager and Peter I. Cork, "A Tutorial on Vision Servo Control," *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 5, pp. 651-670, October 1996.
- [3] Christer U. Peterson, "An Integrated Robot Vision System for Industrial Use," *Proceedings of the Third International Conference on Robot Vision and Sensory Controls*, pp. 241-247, Cambridge, Massachusetts, November, 1993.
- [4] Michael C. Fairhurst, *Computer Vision for Robotic System*, Prentice Hall International (UK) Ltd, 1988.
- [5] Spyros G. Tzafestas, *Intelligent Robotic System*, Marcel Dekker Inc., New York, 1991.
- [6] James H. Graham, "Special Computer Architectures for Robotics: Tutorial and Survey," *IEEE Transactions on Robotics & Automation*, Vol. 5, No. 5, pp. 543-554, October 1989.
- [7] Richard D. Klafater, Thomas A. Chmielewski and Michael Negin, *Robotic Engineering: An Integrated Approach*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1989.
- [8] Ulrich Rembold, *Robot Technology and Applications*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1989.
- [9] Y. Shirai and H. Inoue, "Guiding a Robot by Vision Feedback in Assembly Tasks," *Pattern Recognition*, Vol. 5, pp.99-108, 1973.
- [10] Suan Gottschlich, Carlos Ramos and Damian Lyons, "Assembly and Task Planning: A Taxonomy," *IEEE Robotics & Automation Magazine*, Vol. 1, No. 3, pp. 4-12, September 1994.
- [11] Billibon, H. Yoshimi and Peter K. Allen, "Visual Control of Grasping and Manipulation Tasks," *IEEE International Conference on Multisensor Fusion and Integration for Intelligent System*, Las Vegas, NV, Oct. 2-5, 1994.

- [12] G. D. Hager, "The "X-vision" System: A General Purpose Substruct for Real-time Vision-based Robotics," *Proc. Workshop on Vision for Robots*, pp. 56-63, 1995.
- [13] W. Hattich, "Recognition of Overlapping Workpieces by Model-Directed Construction of Object Contours," *Artificial Vision for Robots*, Edited by Professor I. Aleksander, pp. 77-92, Chapman&Hall, 1984.
- [14] Suresh K. Devarajan, *Object Identification for Robotic Applications using Expert Systems*, Master's Thesis, Robotics Laboratory, Dept. Of Mechanical Engineering, University of Nevada Las Vegas, 1994.
- [15] Zoran R. Novakovic and Bojan Nemeč, "A Solution of the Inverse Kinematics Problem Using the Sliding Mode," *IEEE Transactions on Robotics & Automation*, Vol. 6, No. 2, pp.247-252, April 1990.
- [16] Peter I. Cork and Malcolm C. Good, "Dynamic Effects in Visual Closed-Loop Systems," *IEEE Transactions on Robotics & Automation*, Vol. 12, No. 5, pp. 671-683, October 1996.
- [17] Peter K. Allen, *Robotic Object Recognition Using Vision and Touch*, Cluwer Academic Publishes, 1987.
- [18] Y. F. Li and M. H. Lee, "Applying Vision Guidance in Robotic Food Handling," *IEEE Robotics & Automation Magazine*, Vol. 3, No. 1, pp.4-12, March 1996.
- [19] Harley R. Myler & Arthur R. Weeks, *Computer Imaging Recipes in C*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1993.

APPENDICES

APPENDIX A

Calculation of Arc Center Coordinates Based on Boundary Points

Suppose two lines L_1 and L_2 are parallel to X-axis. They intersect with a circle at point $P_1(x_1,y_1)$, $P_2(x_2,y_2)$, $P_3(x_3,y_3)$, and $P_4(x_4,y_4)$, respectively. The circle center is $P_c(x_c, y_c)$.

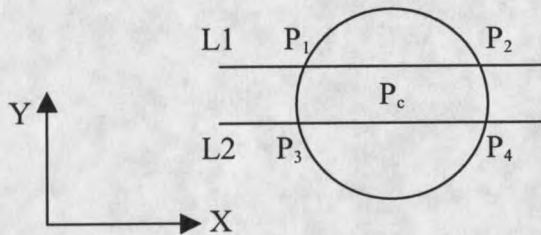


Figure 23. Scanning Line Method for Solving Centroid Value

Use $P_3, P_1,$ and P_2 to solve the first center values ($P_{c1}(x_{c1},y_{1c2})$) and use $P_1, P_2,$ and P_4 to solve the second center values ($P_{c2}(x_{c2},y_{c2})$).

The equation of the line that is perpendicular to chord P_3P_1 is given below:

$$y = k_1x + b_1 \quad \dots\dots\dots(1)$$

Here $k_1 = -(x_1 - x_3)/(y_1 - y_3)$, and $b_1 = (y_1 + y_3)/2 + (x_1 - x_3)(x_1 + x_3)/(2(y_1 - y_3))$.

The equation of the line that is perpendicular to chord P_1P_2 is given below:

$$x = (x_1 + x_2)/2 \quad \dots\dots\dots(2)$$

The equation of the line that is perpendicular to chord P_2P_4 is given below:

$$y = k_3X + b_3 \quad \dots\dots\dots(3)$$

Here $k_3 = -(x_4 - x_2)/(y_4 - y_2)$, and $b_3 = (y_2 + y_4)/2 + (x_4 - x_2)(x_4 + x_2)/(2(y_4 - y_2))$.

Solving Equations (1) and (2), yields the first center values (P_{c1}):

$$x_{c1} = (x_1 + x_2)/2$$

$$y_{c1} = k_1(x_{c1}) + b_1$$

Solving Equations (2) and (3), produces the second center values (P_{c2}):

$$x_{c2} = (x_1 + x_2) / 2$$

$$y_{c2} = k_3(x_{c2}) + b_3$$

Because of the randomness of the battery holes, the position of the holes will deviate from its standard position. During the calculation, if coincidentally $y_3 = y_1$ and $y_2 = y_4$, then $(y_1 - y_3)$ and $(y_4 - y_2)$ will become zero. In this situation, it follows that $y_{c1} = y_{c2} = (y_3 + y_1) / 2$ or $(y_4 + y_2) / 2$.

Based on results of the above calculation, center coordinates of the battery hole are given as the average of the first and second center values:

$$x_c = (x_{c1} + x_{c2}) / 2$$

$$y_c = (y_{c1} + y_{c2}) / 2$$

The distance between P_{c1} and P_{c2} is given below:

$$| P_{c1} - P_{c2} | = \sqrt{[(x_{c1} - x_{c2})^2 + (y_{c1} - y_{c2})^2]}$$

APPENDIX B

The Solving of Inverse Kinematics for Two-Joint Robot

The configuration of two-joint robot in Figure 13 is redrawn in Figure 24.

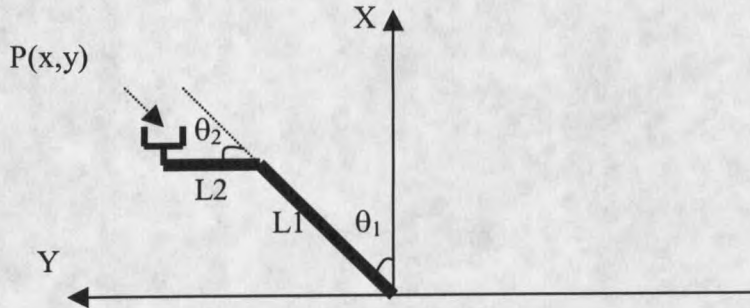


Figure 24. Configuration of Kinematics for Two-Joint Robot

The kinematics equations are:

$$X = (L1 + L2 \cos(\theta_2)) \cos(\theta_1) \dots\dots(1)$$

$$Y = (L1 + L2 \cos(\theta_2)) \sin(\theta_1) \dots\dots(2)$$

For the two-joint robot, solution of the inverse kinematic equations is relatively simple.

The possibility of multiple solutions poses no difficulty:

Equation (2) divided by Equation (1) yields:

$$Y/X = \tan(\theta_1)$$

Then the value of angle θ_1 is given below:

$$\theta_1 = \tan^{-1}(Y/X) \quad -150^\circ < \theta_1 < +150^\circ$$

The value of θ_2 can be obtained from Equation (1) or Equation (2). These results are given below:

$$\theta_2 = \cos^{-1}[(X/(\cos(\theta_1)) - L1)/L2] \quad \text{or}$$

$$\theta_2 = \cos^{-1}[(Y/(\sin(\theta_1)) - L1)/L2] \quad -147^\circ < \theta_2 < +147^\circ$$

Due to the multiple solutions, θ_1 and θ_2 , rules, such as dealing with continuity, must be used to find the correct solution. Many methods for solving inverse kinematic equations are presented in the literature.

APPENDIX C

Robotic Control Software Source Codes

Figure 25 defines the position of teaching points for robotic installation control.

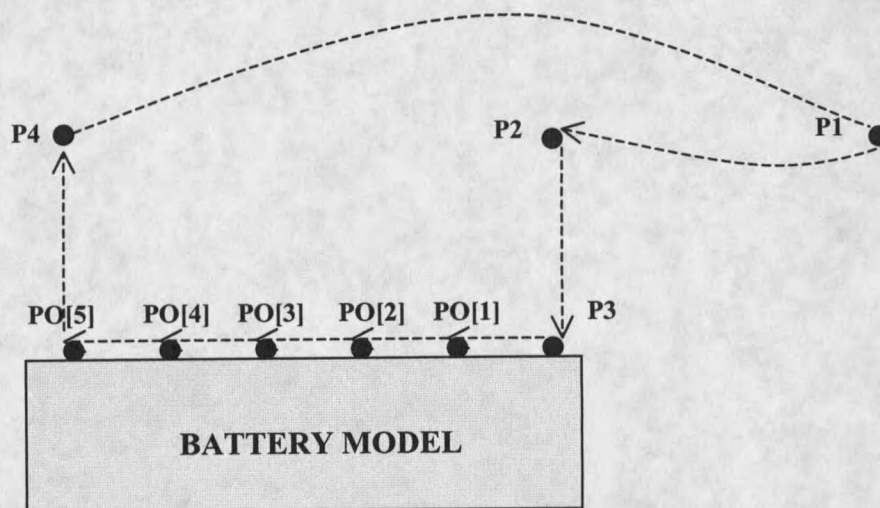


Figure 25. Robotic Teaching Points

P1: Robotic Starting Point

P2: Intermediate point

P3: Battery Reference Point

P4: Intermediate point

PO[1]: Hole 1 Point

PO[2]: Hole 2 Point

PO[3]: Hole 3 Point

PO[4]: Hole 4 Point

PO[5]: Hole 5 Point

PROGRAM LIST

```

1. Program Auto29( )
2;
3;   AVSS Robotic Control Program
4; This program is a semi-automatic control process. AVSS is implemented through
5; manual commands from MCP or keyboard. The communication between the robot
6; and image processing system is not used.
7;
8   SPEED 10 MMPS ALWAYS
9   ACCEL 8, 8
10;
11  DECOMPOSE X1[ ] = PO[1]
12  DECOMPOSE X2[ ] = PO[2]
13  DECOMPOSE X3[ ] = PO[3]
14  DECOMPOSE X4[ ] = PO[4]
15  DECOMPOSE X5[ ] = PO[5]
16  XX[1] = X1[0]
17  YY[1] = X1[1]
18  ZZ[1] = X1[2]
19  RR[1] = X1[5]
20  XX[2] = X2[0]
21  YY[2] = X2[1]
22  ZZ[2] = X2[2]
23  RR[2] = X2[5]
24  XX[3] = X3[0]
25  YY[3] = X3[1]
26  ZZ[3] = X3[2]
27  RR[3] = X3[5]
28  XX[4] = X4[0]
29  YY[4] = X4[1]
30  ZZ[4] = X4[2]
31  RR[4] = X4[5]
32  XX[5] = X5[0]
33  YY[5] = X5[1]
34  ZZ[5] = X5[2]
35  RR[5] = X5[5]
36;
37  10   ATTACH(1)
38      $$ = "move to robotic starting point"
39      WRITE(1) $$
40      SPEED 30 ALWAYS

```

```

41      MOVE P1
42  20   $$ = "START A WORK CYCLE(1/0)?"
43      WRITE(1) $$
44      KEYMODE 43, 44 = 0
45      key = PENDANT (0)
46      IF key == 43 GOTO 20
47      $$ = "move to battery reference point"
48      WRITE (1) $$
49      MOVE P2
50      DELAY 1
51      MOVE P3
52      IF HAND == 1 THEN
53          $$ = "place pen and close gripper with 1"
54          WRITE(1) $$
55          DO
56              KEY = PENDANT (0)
57          UNTIL KEY == 44
58          CLOSEI
59      ELSE
60      END
61      DETACH (1)
62;
63      FOR I=1 TO 5
64          ATTACH(1)
65          $$ = "next hole needs vision correction(1/0)?"
66          WRITE(1) $$
67          DELAY 2
68          KEYMODE 43, 44 = 0
69          S1 = PENDANT(0)
70          IF S1==44 THEN
71              $$ = "input offset of x and y from keyboard"
72              WRITE(1) $$
73              PROMPT "input corrected offset of x and y:", X, Y
74              TYPE /B, /F10.3, "X=", X
75              TYPE /B, /F10.3, "Y=", Y
76              TYPE "move to next hole with vision correction"
77              MOVE TRANS(XX[I]+X,YY[I]+Y,ZZ[I],0,180,RR[I])
78          ELSE
79              MOVES PO[I]
80      END
81      END
82;

```

```
83      DO  
84          KEY = PENDANT(0)  
85      UNTIL KEY == 43  
86      DETACH (1)  
87      MOVE P4  
88      DELAY 1  
89      GOTO 10  
90. END
```

APPENDIX D

Image Processing Software Source Codes

MAIN CONTROL PROGRAM

```
#include "c:\qcap\pedefs.h"
#include "c:\qcap\peerrs.h"
#include <conio.h>
#include <stdio.h>
#include <malloc.h>
#include <math.h>

#define UINT    unsigned int
#define UCHAR  unsigned char
#define B_PIX  0
#define I_PIX  1

int v,vvv,status,v10,j5=0;
float ox,oy,dpixel,derror,mx,my;

init();
comm();
image();

main()
{
    int i,i1,j;
    float ox5,oy5,derr,x[5],y[5],fx,fy;
    char hole;

    /* menu(); */
    init();          /* initial DT55 */

    // [Shaking hands test with adept robot] //

    v10=0;
    /*receive data from robot*/
    comm();
        if (v==10)
        {
            printf("communication is ok\n");
            v10=10;
            /*transfer received value '10' to robot*/
            comm();
        }
}
```

```

else
{
    printf("communication failed\n");
    return 0;
}

v10=0;    /*receive the teaching data of start
           point and five holes from robot*/
for(i=0;i<12;i++)
{
    comm();
    x[i]=v;
}

v10=10;   /*start adept robot for installation*/
v=1;
comm();
v10=0;
comm();
if(v==1) { printf("robot is running\n");}
else
{
    printf("communication failed\n");
    return 0;
}

// [Manual image capturing control ]//

mx=0.; my=0.;
do
{
    do{
        printf("press 'y' key to start image processing
               : ");
        scanf("%s",&hole);
    }while(hole!='y');
    printf("input hole number : ");
    scanf("%d",&j5);

    if(j5==1) { mx=0; my=0; }
    ox5=0; oy5=0;

    for(i=0;i<5;i++)

```

```

{
    image();
    ox5=ox+ox5; oy5=oy+oy5;
}

if((dpxel==0)&&(derror==0))
{
    ox5=ox5/5.0; oy5=oy5/5.0;
    fx=ox5+mx; fy=oy5+my;

// [correction because of former hole deviation ] //

    if((fx>2.0)|| (fx<-2)) { mx=fx; }
    else { fx=0; mx=0;}
    if((fy>2.0)|| (fy<-2)) { my=fy; }
    else { fy=0; my=0;}

    printf("THE HOLE NUMBER IS:   %d\n",j5);
    printf("offset value:  x=%6.3fmm
            y=%6.3fmm      \n", fx,fy);
}
else
{
    printf("\nDeviation of geometry exceed limits,
            stop installation!!!\n\n");
    dpxel=0; derror=0;
    mx=0; my=0;
}

}while(1);

// [Decide if next hole needs correction? ]//

derr=sqrt(ox5*ox5+oy5*oy5);
if(derr>5.0)
{
    printf("vision correction for this hole");
    v10=10; /*transfer corrected position to robot*/
    v=x[j5]+ox5;   comm();
    v=y[j5]+oy5;   comm();
}
else
{

```

```

        v10=10;    /*do not need vision correction*/
        v=8;
        comm();
    }

    return 0;
}

```

INITIALIZATION PROGRAM

```

#include<stdio.h>
#include"c:\qcap\pedefs.h"
#include"c:\qcap\peerrs.h"

init()
{
    extern int v, vvv, status;

// [Initialization DT55 procedure]//

    status = pe_open ();
    status = pe_reset ();
    vvv = inpw(0x320);
    printf("vvv=%x\n", vvv);
    /*check the "video control/status register=200H */
    status = pe_init_luts ();
    /*initializing the look-up-table*/
    vvv = vvv&(0xfffc);
    outpw(0x320,vvv);
    /* select ILUT 0 again */
    status = pe_set_const (0);
    printf("status=%d\n",status);
    /* set the value of pixel gray level as 0 */
    vvv = inpw(0x320);
    printf("vvv=%x\n", vvv);
    /* video control/status register=200H */

// [Display Image Operation ]//

    status = pe_set_sync (1);

```

```

printf("status=%d\n",status);
    /* set external sync signal because the external
       camera is used */
    /* enable displaying the board memory */
    vvv = vvv|(0x40);
outpw(0x320,vvv);
    /* enable display board memory again*/
    vvv = vvv|(0x10);
outpw(0x320,vvv);
    /*select video timing source as external camera */
    vvv = inpw(0x320);
printf("vvv=%x\n", vvv);
    /* video control/status register=250H */
status = pe_set_const (0);
printf("status=%d\n",status);
status = pe_acquire();
printf("status=%d\n",status);
    /*acquire a image from camera, store it in onboard
       fram and display it in external monitor */
    // status = pe_set_sync (0);
printf("status=%d\n",status);
    /* video signal is in field change state */
    vvv = vvv|(0x8050);
outpw(0x320,vvv);
    vvv = inpw(0x320);
printf("vvv=%x\n", vvv);
    /* video control/status register=350H */

//[Initialization serial communication interface COM2]//

outp(0x3fb,0x80);
outp(0x3f9,0);
outp(0x3f8,0x60);          /* 1200 baud rate*/
outp(0x3fb,0x1a);
    /* 7-bit length, 1-stop bit, even parity */
outp(0x3f9,0);
    /* disable all interrupts */

//[Shake signal with robot controller//

v=inp(0x3fd);
if((v-30)!=0)
{

```

```

do{v=inp(0x3fd);
    printf("communication interface error\n");
    if(v==200) return 0;
    }while(1);
return 0;
}
}

```

IMAGE PROCESSING PROGRAM

```

#include<stdio.h>
#include<conio.h>
#include<math.h>
#include"c:\qcap\pedefs.h"
#include"c:\qcap\peerrs.h"

#define UINT    unsigned int
#define UCHAR   unsigned char
#define B_PIX   0
#define I_PIX   1

int vvv,v,v10,status;
extern int j5;
extern float ox,oy,dpixel,derror; /* return status */
int offx,offy;

image()
{
    int i,i1,j,k,kk,kkmax,jj; /* loop index */
    int x1,y1,x2,y2,x3,y3,x4,y4;
    int xlim1,xlim0,xlim2,xlim,snum;
    int xx0,yy0;
    int num=0;
    int f[231][3];
    int fx,fy;
    double gg;
    double k1,b1,k2,b2;
    double x0,y0,r,x01,y01,x02,y02;
    float xratio,yratio;
    UCHAR *line_array;
    UINT count;
}

```

```

status = pe_reset ();
vzv = inpw(0x320);
/*check the video control/status register=200H */
status = pe_init_luts ();
/* initializing the look-up-table */
vzv = zvz&(0xffff);
outpw(0x320,vzv); /* select ILUT 0 again */
status = pe_set_const (0);
/* set the value of pixel gray level as 0 */
vzv = inpw(0x320);
/* video control/status register=200H */

// [Display Image Operation] //
status = pe_set_sync (1);
/* set external sync signal because
the external camera is used */
vzv = zvz|(0x40);
outpw(0x320,vzv);
/* enable display board memory again*/
vzv = zvz|(0x10);
outpw(0x320,vzv);
/*select video timing source as external camera */
vzv = inpw(0x320);
/* video control/status register=250H */
status = pe_set_const (0);
status = pe_acquire();
/*acquire a image from camera, store it in onboard
fram and display it in external monitor */
vzv = zvz|(0x8050);
outpw(0x320,vzv);
vzv = inpw(0x320);
/* video control/status register=350H */

// [Binary Image and solving the standard pixel number in
one hole] //

for(j=170;j<=330;j++)
{ outpw(0x32a,j);
for(i=220;i<=400;i++)
{ outpw(0x328,i);
vzv=inpw(0x32c); zvz=vzv & (0x00ff);
if(vzv<=150)
{
//outpw(0x32c,0); /*set background as "0"*/
num++;
}
}
}

```

```

    }
    //else outpw(0x32c,255);    /*set hole as "255"*/
}
}
printf("pixel number is %d\n\n", num);

// [Boundary points solving (1): conventional method] //
k=0;
j=245;kk=0;
  outpw(0x32a,j);
  for(i=220;i<=400;i++)
  {
    outpw(0x328,i);
    vvv=inpw(0x32c); vvv=vvv&(0x00ff);
    if(k==0)
    {
      if(vvv<=150)
      {
        k=1; //outpw(0x32c,0);
        x1=i; y1=j;
        kk++;
      }
      //else outpw(0x32c,255);
    }
    else
    {
      if(vvv>150)
      {
        k=0; jj=i-1;
        outpw(0x328,jj);
        //outpw(0x32c,0);
        x2=jj; y2=j;
      }
      //else outpw(0x32c,255);
    }
  }

j=265;kk=0;
  outpw(0x32a,j);
  for(i=220;i<=400;i++)
  {
    outpw(0x328,i);
    vvv=inpw(0x32c); vvv=vvv&(0x00ff);
    if(k==0)
    {

```

```

        if (vzv<=150)
        {
            k=1; //outpw(0x32c,0);
            x3=i; y3=j;
            kk++;
        }
        //else outpw(0x32c,255);
    }
    else
    {
        if (vzv>150)
        {
            k=0; jj=i-1;
            outpw(0x328,jj);
            x4=jj; y4=j;
        }
    }
}

```

// [Boundary points solving (2): simple method] //

```

    outpw(0x32a,245);
    snum=0; y1=245;
    xlim1=220;xlim2=400;xlim0=(xlim2+xlim1)/2;

    outpw(0x328,xlim0);
    vzv=inpw(0x32c); vzv=vzv&(0x00ff);
    if (vzv==255)
    {
        derror=1;
        /* geometry of hole deviate outside feature
        window */
        return 0;
    }

    do
    {
        xlim=(xlim1+xlim0)/2;
        outpw(0x328,xlim);
        vzv=inpw(0x32c); vzv=vzv&(0x00ff);
        if (vzv>=150) {xlim1=xlim;}
        else if (vzv<=150) {xlim0=xlim;}
        snum++;
    }while( (xlim0-xlim1)!=1);
    printf("xlim=%d          snum=%d\n",xlim,snum);

```

```

printf("x1=%d\n\n",x1);
x1=xlim;

snum=0; y2=245;
xlim1=220; xlim2=400; xlim0=(xlim2+xlim1)/2;
do
{   xlim=(xlim0+xlim2)/2;
    outpw(0x328,xlim);
    vvv=inpw(0x32c); vvv=vvv&(0x00ff);
    if(vvv>=150){xlim2=xlim;}
    else if(vvv<=150){xlim0=xlim;}
    snum++;
}while((xlim2-xlim0)!=1);
printf("xlim=%d      snum=%d\n",xlim,snum);
printf("x2=%d\n\n",x2);
x2=xlim;

outpw(0x32a,265);

snum=0;y3=265;
xlim1=220;xlim2=400;xlim0=(xlim2+xlim1)/2;

outpw(0x328,xlim0);
vvv=inpw(0x32c); vvv=vvv&(0x00ff);
if(vvv==255)
{
    derror=1;
    /* geometry of hole deviate outside feature
       window */
    return 0;
}

do
{   xlim=(xlim1+xlim0)/2;
    outpw(0x328,xlim);
    vvv=inpw(0x32c); vvv=vvv&(0x00ff);
    if(vvv>=150){xlim1=xlim;}
    else if(vvv<=150){xlim0=xlim;}
    snum++;
}while((xlim0-xlim1)!=1);
printf("xlim=%d      snum=%d\n",xlim,snum);
printf("x3=%d\n\n",x3);
x3=xlim;
snum=0;y4=265;
xlim1=220;xlim2=400;xlim0=(xlim2+xlim1)/2;

```

```

do
{
  xlim=(xlim0+xlim2)/2;
  outpw(0x328,xlim);
  vvv=inpw(0x32c); vvv=vvv&(0x00ff);
  if(vvv>=150){xlim2=xlim;}
  else if(vvv<=150){xlim0=xlim;}
  snum++;
}while((xlim2-xlim0)!=1);
printf("xlim=%d      snum=%d\n",xlim,snum);
printf("x4=%d\n\n",x4);
x4=xlim;

```

```
// [Solving coordinates of the center] //
```

```

if((y4-y2)!=0)
{
  k2=-(x4-x2); k2=k2/(y4-y2);
  /*solve the center p01 from p1, p2, p4*/
  b2=(y2+y4)/2-k2*(x2+x4)/2;

  x01=(x1+x2)/2;
  y01=k2*x01+b2;
}
else
{
  x01=(x1+x2)/2;
  y01=(y1+y2)/2;
}

if((y1-y3)!=0)
{
  k1=-(x1-x3); k1=k1/(y1-y3);
  /*solve the center p02 from p1, p2, p4 points*/
  b1=(y3+y1)/2-k1*(x3+x1)/2;

  x02=(x1+x2)/2;
  y02=k1*x02+b1;
}
else
{
  x02=(x1+x2)/2;
  y02=(y1+y2)/2;
}

```

```

// [ Identify if the deviation of hole's geometry is
serious ]//

dpxiel=sqrt((x02-x01)*(x02-x01)+(y02-y01)*(y02-y01));
printf("dpxiel=%3.0f,\n", dpxiel);

if(dpxiel<20)
{
    xx0=(x01+x02)/2.; yy0=(y01+y02)/2.;
    /*solve the center of circle*/
    printf("xx0(pixel)=%d    yy0(pixel)=%d\n\n",
        xx0,yy0);
    xratio=0.253; yratio=0.234;

    switch(j5)
    {
        case 1: { offx=xx0-327; offy=yy0-249; break; }
        case 2: { offx=xx0-325; offy=yy0-245; break; }
        case 3: { offx=xx0-329; offy=yy0-245; break; }
        case 4: { offx=xx0-324; offy=yy0-246; break; }
        case 5: { offx=xx0-334; offy=yy0-249; break; }
        /*standard centroid values are measured based on
        standard holes*/
    }

    printf("offx=%d    offy=%d    \n", offx,offy);

    x0=(float)offx*xratio;
    /*calculate camera coordinates with mm*/
    y0=(float)offy*yratio;

    ox=x0; oy=-y0;

    derror=sqrt(ox*ox+oy*oy);
    if(derror>10.0)
    {
        printf("\nDeviation of geometry exceeds
            limit and stop installation!!!\n\n");
        derror=1;
        return 0;
    }

    printf("x0=%6.3fmm    y0=%6.3fmm\n", x0,y0);
    status = pe_close ();
}

```

```
// [Sobel Operator for Edige Detection] //
```

```

for(j=244;j<=246;j++)
{  outpw(0x32a,j);
   for(i=220;i<=400;i++)
   {  outpw(0x328,i);
      f[i][j]=inpw(0x32c);
      /*if(f[i][j]==0) { f[i][j]=1; }
      else { f[i][j]=0; }*/
   }
}

j=245;
for(i=221;i<400;i++)
{gg=sqrt(((f[i+1][j+1]+2*f[i+1][j]+f[i+1][j-1]) -
          (f[i-1][j+1]+2*f[i-1][j]+f[i-1][j-1]))
          *((f[i+1][j+1]+2*f[i+1][j]+f[i+1][j-1]) -
            (f[i-1][j+1]+2*f[i-1][j]+f[i-1][j-1]))
          +((f[i-1][j-1]+2*f[i][j-1]+f[i+1][j-1]) -
            (f[i-1][j+1]+2*f[i][j+1]+f[i+1][j+1]))
          *((f[i-1][j-1]+2*f[i][j-1]+f[i+1][j-1]) -
            (f[i-1][j+1]+2*f[i][j+1]+f[i+1][j+1]))));

   if(gg!=0)
   { printf("point is edge=%d      %d\n",
            i,j); }
}

for(j=264;j<=266;j++)
{  outpw(0x32a,j);
   for(i=220;i<=400;i++)
   {  outpw(0x328,i);
      f[i][j]=inpw(0x32c);
   }
}

j=265;
for(i=221;i<400;i++)
{gg=sqrt(((f[i+1][j+1]+2*f[i+1][j]+f[i+1][j-1]) -
          (f[i-1][j+1]+2 *f[i-1][j]+f[i-1][j-1]))
          *((f[i+1][j+1]+2*f[i+1][j]+f[i+1][j-1]) -
            (f[i-1][j+1]+2*f[i-1][j]+f[i-1][j-1]))
          +((f[i-1][j-1]+2*f[i][j-1]+f[i+1][j-1]) -
            (f[i-1][j+1]+2*f[i][j+1]+f[i+1][j+1]))
          *((f[i-1][j-1]+2*f[i][j-1]+f[i+1][j-1]) -
            (f[i-1][j+1]+2*f[i][j+1]+f[i+1][j+1]))));
}

```

```

        * ((f[i-1][j-1]+2*f[i][j-1]+f[i+1][j-1]) -
          (f[i-1][j+1]+2*f[i][j+1]+f[i+1][j+1]));
        if(gg!=0)
            {printf("point is edge=%d      %d\n", i,j); }
    }
    dpixel=0; derror=0;
}
else
{    dpixel=1; }
return 0;
}

```

COMMUNICATION PROGRAM

```

#include<stdio.h>
#include<conio.h>

extern int v,v10;

comm()
{
    int i;
    if(v10==0)          /*receive data from robot*/
    {
        if((v-30)!=0)
        {
            printf("communication error\n");
            return 0;
        }
        while((v-1)==0);
        v=inp(0x3f8);
    }
    if(v10==10)        /*transfer data to robot controller*/
    {
        if((v-30)!=0)
        {
            printf("communication error\n");
            return 0;
        }
        while((v-32)==0);
        outp(0x3f8,v);
    }
}
}

```

MONTANA STATE UNIVERSITY LIBRARIES



3 1762 10298991 8