1995 NASA/ASEE SUMMER FACULTY FELLOWSHIP PROGRAM
JOHN F. KENNEDY SPACE CENTER
UNIVERSITY OF CENTRAL FLORIDA

S7-61
1757
P. 27

INVESTIGATION OF REGISTRATION ALGORITHMS FOR THE AUTOMATIC TILE PROCESSING SYSTEM

Dr. Dan E. Tamir
Associate Professor
Computer Science Department
Florida Institute of Technology
Melbourne, Florida

KSC Colleagues - Peter Engrand and Todd Graham
Artificial Intelligence/Expert Systems

Contract Number NASA-NGT-60002
Supplement 19

August 4, 1995

Acknowledgments

I would like to thank Peter Engrand, Todd Graham, and Terry Ross from NASA, as well as, Charlie Goodrich, Rich Bennett, and Bob Merchant from I-Net, for their cooperation and technical support. I would like also to extend compliments and thanks to Gregg Buckingham, Dr. Hosler, and Kari Stiles for their professional management of the summer faculty program.

Abstract

The Robotic Tile Inspection System (RTPS), under development in NASA-KSC, is expected to automate the processes of post-flight re-water-proofing and the process of inspection of the Shuttle heat absorbing tiles. An important task of the robot vision sub-system is to register the "real-world" coordinates with the coordinates of the robot model of the Shuttle tiles. The model coordinates relate to a tile data-base and pre-flight tile-images. In the registration process, current (post-flight) images are aligned with pre-flight images to detect the rotation and translation displacement required for the coordinate systems rectification.

The research activities performed this summer included study and evaluation of the registration algorithm that is currently implemented by the RTPS, as well as, investigation of the utility of other registration algorithms.

It has been found that the current algorithm is not robust enough. This algorithm has a success rate of less then 80% and is, therefore, not suitable for complying with the requirements of the RTPS.

Modifications to the current algorithm has been developed and tested. These modifications can improve the performance of the registration algorithm in a significant way. However, this improvement is not sufficient to satisfy system requirements.

A new algorithm for registration has been developed and tested. This algorithm presented very high degree of robustness with success rate of 96%.

Summary

This report summarizes the findings of the summer research performed at NASA-KSC. The research has concentrated on the registration algorithms for the Robotic Tile Processing System (RTPS).

First the image registration algorithm that is currently implemented by the RTPS is described and evaluated. Second, modifications to the existing algorithm are outlined. Next, a new approach is proposed. Results of experiments to evaluate the utility of the registration algorithms are presented and evaluated. Finally, conclusions that are based on the results of the experiments are drawn.

The registration procedure is divided into three phases; The first, referred to as center-to-tile is responsible for placing the camera so that one tile is completely included in an image frame. The second phase, performs coarse registration. The last phase, is responsible for fine registration.

The current system does not include a utility for tile centering.

The algorithm for coarse registration implemented by the current system is relatively simple. It applies one dimensional projections to an edge-enhanced tile-image. These projections are used for the identification of the tile vertices.

The fine registration phase relies on identification of tile "blobs"[1].

On-line and off-line experiments with the current algorithm has raised doubts about its adequacy for complying with the system requirements. On one hand, in many cases the algorithm has failed to identify the tile vertices. On the other hand, there have been concerns that the "blobs" can change significantly during the landing process, thus, confusing the fine registration procedure.

Modifications to the current algorithm has concentrated on the coarse registration process. Two modifications has been considered; First, it has been proposed to binarize the edge-enhanced image before obtaining the projections. If done properly, the binarization process can further emphasizes the tile edges and increase the rate of successful edge identification by the projections. The second modification includes the division of the tile into zones obtaining per-zone projections of the edge-enhanced image. The results of edge identifications per individual zone are then interpolated on the entire image to obtain a better approximation of the tile edges. These modifications have improved significantly the performance of the registration algorithm.

The modifications described above has prompted a probe of a new registration procedure. The idea has been to try to interpolate the edge lines directly from the binarized edge-enhanced image rather then using the zone projections for the interpolation. This idea has lead to the implementation of the Hough transform as an instrumental phase of the entire registration procedure.

In the new registration procedure, Hough transform is used to detect one edge, as well as, the circle that marks the re-water-proofing hole. This can be used for tile centering. Next, Hough transform is applied to the binarized, edge-enhanced, tile-image to identify the tile edges and vertices. Finally, cross-correlation of a small area around the tile vertices taken from the post-flight and pre-flight images is used for fine registration.

Experiments with the new coarse registration algorithm has yielded more than 96% success rate with an error of less than 0.4 degrees in the detected rotation and less than 0.1" in translation.

1. A "blob" is an homogeneous, continuous, region in the image.

Table of Contents

List of Figures

List of Tables

Introduction

NASA-KSC is developing a Robotic Tile Processing System (RTPS) for automatic post-flight position calibration, re-water-proofing hole verification, and visual inspection of the Shuttle tiles for anomaly detection.

The re-water-proofing and the inspection processes involve acquisition of pre-flight and post-flight images of the tiles, registration of these images, identification of the tile location for re-water-proofing, and detection of significant changes in the tile surface as a part of the tile inspection.

Image registration is one of the important tasks of the vision sub-system of the RTPS. Currently, the RTPS is applying two different registration algorithms. The first is used for the re-water-proofing hole verification. The second is a part of the automatic tile inspection.

A prototype of RTPS delivered to NASA-KSC has been tested and found deficient in its performance with respect to the image registration for re-water-proofing hole verification.

This summer research has concentrated on investigation of image registration techniques. In specific, the focus has been on the registration of the tile images for the re-water-proofing hole verification.

1.1 Background

Each Shuttle contains between 20,000 to 30,000 heat-absorbing tiles. Most of the tiles are square tiles with dimensions of 6"x6". Some tiles, however, has different sizes or shapes.

The robot, developed by CMU, is guided through a laser system. A second laser system on the robot arm places the arm in a given distance from the Shuttle surface and aligns the arm with the surface (three degrees of freedom, z, yaw, and pitch).

The robot vision sub-system, developed by SRI, is expected to take care of the other three degrees of freedom (x,y, and, roll) through the image registration process.

The system has been designed under the assumption that the total error in the identification of the robot location is relatively small (up to 1"). The data acquisition system has been designed to cover an area of 8"x8" with a resolution of 128 pixels/inch. Hence, under these assumptions, one entire tile is within one camera frame (one image).

It is currently assumed that due to changes in the Shuttle parking location and due to accumulated errors in the robot location system, there might be a discrepancy of up to 2" between the physical and the computed location of the robot. This new assumption emphasizes the need for a utility to center the RTPS camera around examined tiles.

1.2 The SRI Registration Algorithm

In addition to the data acquisition unit, the vision sub-system includes a CPU board (Motorolla 68040), a hard drive, and an image-processing board (Data-Cube). The system runs under a "UNIX-like" real-time operating system (VXWORK). Development tools include a C-compiler, a debugger (VXGDB), and a monitor. The system is accessible as an internet node.

The Shuttle surface is logically divided into zones of about 100 tiles per zone. Each zone is expected to include 3- 6 registration tiles. The process of registering the system to any of these registration tiles can be divided into three main procedures: Center_to_Tile, Coarse_Registration, and Fine-Registration. These procedures are described briefly in the following section. A more detailed description is included in Appendix B of this document.

Center_to_Tile

Center_to_Tile is supposed to take care of the case where due to errors in the robot location sub-system the camera is not centered at a tile so that one or more of the edges of the tile are out of view. This module have not been fully developed by SRI. A module by this name exists, however, it fails if one or more of the tile edges is missing.

Coarse_Registration

This module attempts to identify the tile vertices through a process of edge enhancement and one dimensional projections. A tile vertex is defined to be the point where two tile edges coincide.

Edge enhancement is accomplished through convolution of the tile-image with an horizontal and a vertical gradient operator mask. Since the convolution requires extensive CPU time, this module has been ported by SRI to the Data-Cube. This is, however, the only computational intensive module ported to the Data-Cube.

After identifying the tile edges, the next modules of the registration software are applied only to the area enclosed within the edges. This has the effect of coarse registration.

Fine_Registration

Sometimes, it is difficult to determine whether a detected edge belongs to the tile in hand, or to the adjacent tile. Hence, a module for fine registration has been developed by SRI. This module attempts to identify continuous homogeneous regions in the tile (referred to as "blobs") and use these blobs for fine registration.

Segmentation is achieved through binarization of the images using a dynamic threshold. A connected component labeling algorithm identifies blobs. The blobs are classified according to their area, shape, and proximity to other blobs. Only blobs that comply with size shape and proximity restrictions are retained as "good" blobs. The centers of these blobs are used for the fine registration stage.

1.3 Modifications to the SRI Registration Algorithm

The modifications to the current algorithm has concentrated on the coarse registration process. The first modification is concerned with the edge detection part and the second attempts to on improve the procedure of obtaining the projections.

Edge Enhancement

Generally, the pixels that belong to the tile edges are darker than the rest of the image pixels. As a result, in many cases it is possible to skip the edge-enhancement phase and search for local minimums rather then local maximums in the projections. The advantage of this approach is that it reduces the computation time. For this reason, it has been investigated by KSC stuff. On the other hand, this method is the least robust of all the methods investigated so far. Moreover, given the available high-performance hardware, the reduction in computational complexity is irrelevant

Nevertheless, it is implemented (referred to as "ksc") as a part of the Off-line system described later in this paper.

Another modification relates to the edge enhancement procedure. In this version of the registration process, after the convolution the image is binarized using K-means clustering [3]. This approach has proved to be very robust with respect to identifying the edge vertices, however, it is inferior to the method described in the next section.

Per-zone Projections

The second modification to the SRI algorithm includes the division of the tile into zones obtaining per-zone projections of the edge-enhanced image. The results of edge identifications per individual zone are then interpolated on the entire image to obtain a better approximation of the tile edges. These modifications have improved significantly the performance of the registration algorithm. Moreover, this modification has prompted a probe of a new registration procedure. The idea has been to try to interpolate the edge-lines directly from the binarized edge-enhanced image rather than using the zone projections for the interpolation. This idea has lead to the implementation of the new algorithm for registration described in the next section.

A New Registration Algorithm

In the new registration procedure, Hough transform is used to detect one edge, as well as, the circle that marks the re-water-proofing hole. This is used for tile centering. Next, Hough transform is applied on the binarized, edge-enhanced, tile image to identify the tile edges and vertices. Finally, cross-correlation of a small area around the tile vertices of the post-flight and pre-flight images is used for fine registration. The new algorithm is described briefly in the following sections. A detailed description is available in Appendix B.

2.1 Center-To-Tile

A module for centering the camera around one tile has been developed. Two versions of the module exist. The first is using a circular Hough transform to identify the circle marking the re-water-proofing hole [1]. This module has been tested on a set of tile images that is currently available. It has been found to identify the circle in 63 out of 66 tile images.

A second version for tile centering procedure relies on the linear Hough transform [2]. It attempts to identify at least one vertex and use this vertex for centering the camera. This version has not been tested yet.

2.2 Coarse-Registration

The module Center_to_Tile, as well as, the module Coarse_Registration are using Hough transform. The transform is applied to a binary image obtained from edge enhancement and line enhancement kernels. The difference between these two modules is that Center_to_Tile is using edge enhancement and circular Hough transform, while Coarse_Registration is using line enhancement and linear Hough transform. These operation are described below.

In the future, experiments can be conducted to examine the utility of applying other convolution masks including noise reduction operators.

After the convolution the image is binarized using K-means clustering [3]. The algorithm is executed with K=3, hence, it partitions the edge-enhanced/line-enhanced pixels into three classes. It has been found that class three of the edge-enhanced pixels are best for the circle identification while class two of the line-enhanced pixels are best for the tile edges detection.

The Hough transform is considering every pixel in the appropriate cluster. The circular transform attempts to draw a circle around every pixel from class three of the edge enhancement image. It measures how many of these circles coincide. The linear Hough transform is attempting to draw as many lines as possible through each pixel in class two of the line-enhanced image. It measure how many of these lines coincide. It is assumed that most of the circles with the same radius as the radius of the circle that marks the re-water-proofing hole coincide at the center of this circle. The lines going through tile edges share the same slope and intercept. Hence, most of the lines coincide in slopes and intercept that correspond to the slope and intercept of the tile edges. This can enable the detection of the tile vertices.

The linear Hough transform has been applied to 156 tile images and has identified the lines in 96.5% of these images with an accuracy of about 0.1" in detecting translation and 0.4 degrees in detecting rotation.

2.3 Fine-Registration and AMDF

One concern with the fine registration method developed by SRI is that the blobs, used as a part of the fine registration, are susceptible to changes between flights.This is due to the effect of the heating of the tiles when the Shuttle returns to the atmosphere and the collision of gases and dust with the tiles during the landing. One assumption has been that the tile vertices are less sensitive to these problems, hence, they can be used for a more robust fine registration. This assumption has lead to a modified fine registration algorithm.

The Average Magnitude Difference Function can be described as a brute-force approach for registration [4]. This method ranks every possible rotation and translation and selects the best. Being a brute-force method, it can not be considered for implementation on large images, however, after the detection of the tile vertices, AMDF can be applied to a set of small images centered around the vertices. This procedure has been implemented, but, has not been fully tested yet.

Experiments, Results, and Evaluation

An off-line system for experiments with the different algorithms has been developed. The Off-line system is a set of C procedures. It is further described in appendixes A and B.

The experiments, the results, and result evaluation are given below.

3.1 Experiments

Two sets of images have been acquired using the SRI camera and acquisition software. The first set consists of 66 images of 22 tiles mainly from one zone of the Shuttle Endeavor. The second set includes 90 images of 30 tiles acquired from the Shuttle Columbia. This set includes "tough" tiles from different zones of the Shuttle. They are considered "tough" since according to our subjective judgement each of them contains several artifacts that might impose difficulty in the process of identifying the tile-edges for one or more of the tested algorithms. Each tile has been acquired with three different lighting set-ups. In addition, a few more images of some of the tiles have been acquired with some degrees of rotation and translation.

All the algorithms excluding the per-zone method have been applied to the first set of tiles. Only the new algorithm has been applied to the second set. The results of these experiments are summarized in the next section.

3.2 Results

The results of the experiments are summarized in Table 1. The entries of the table list the percentage of correct identification of the tile edges and the average CPU time (seconds). In this table, "SRI" refers to the method implemented by SRI and "KSC" refers to a modification to the SRI method. Under this modification, edge-enhancement is not applied to the tile-image. Instead, the algorithm obtains the projections of the tile-image and searches for minimums. The method "Trin" refers to the case in which the image goes through edge-enhancement, and binarization using 3-means clustering before the one dimensional projections are obtained. The method "Line" refers to the application of the linear Hough transform, and the method "Circle" refers to the application of the circular Hough transform.

Set/Method	SRI	KSC	Trin	Line	Circle
First Set	81% / 190	77% / 10	100% / 250	100% / 310	95% /310
Second Set	n/a	n/a	n/a	93% / 310	n/a
Total	n/a	n/a	n/a	96.5% / 310	n/a

Table 2: Experiments and Results

3.3 Evaluation

The evaluation of these methods analyses two aspects of performance: identification accuracy and computational complexity. The identification accuracy is evaluated using the error rate of individual method and the accuracy of identifying the edge parameters. The computational complexity is evaluated based on the average CPU time obtained in these experiments.

Identification Accuracy

The experiments include three methods for identifying the vertices using one dimensional projections. Among this three, the method referred to as "Trin" is a clear winner.

One problem with these three methods is that they are sensitive to rotation. Moreover, in their current form, they do not give a good indication of the edge parameters. That is the distance of the tile vertices from the origin of the coordinate system[1] and the angles between each of the tile vertices and the X -axis of the coordinate system. Hence, they require an additional step.

The procedure "Line" does this additional step at the same time that it identifies the tile edges. This procedure identifies the edges and supplies their distance from the origin and angle with the X -axis of the model coordinates. The accuracy of "Line" is about 0.4 degrees and 0.1". The method "Line" has the same or better error rate compared to "Trin", this is done with additional, relatively accurate, information. For this reasons, the method "Line" is considered the best method tested so far.

The method "Circle" has identified the circle around the re-water-proofing hole in 95% of the times. There are, however, two concerns with respect to the performance of this method; First, it relies on an artifact (the circle) that can change significantly between flights. Second, it does not identify the center of the circle accurately enough (an error of about 0.25" is expected). This method, however, has been designed for the module center-to-tile and not for the coarse registration. Therefore, it should be compared with other methods for centering the tile-image. This is one of our recommendations for future research.

Computational Complexity

Based on the information in Table 1, the algorithms are ranked according to their complexity in the following ascending order (the lower the better):

1) "KSC"
2) "SRI"
3) "TRIN"
4) "Line" and "Circle"

These figures are based on running the methods off-line on a Sun Sparc -10. However, with respect to the RTPS, the issue of complexity should not be considered in the process of selecting an algorithm since the actual system contains a very powerful pipeline-processor, the Data Cube. Actually, the procedure "SRI" is already implemented on the Data-Cube and runs in less than 5 seconds. Moreover, the Data-Cube contains a module for Hough transform which can be used for implementing the procedure "line".

1. The coordinate system referred to here, is the coordinate system of the shuttle model stored in the robot.

Conclusions and Recommendations for Future R&D

4.1 Conclusions

Based on the evaluation of the results of the experiments., the following conclusions are drawn:

- Edge-enhancement is an essential operation. It should not be skipped (as in the "KSC" algorithm). The current edge-enhancement masks give reasonable results, however, there is still need for further research into other edge-detection and edge-enhancement methods.
- After edge-enhancement, the image has to be binarized in order to further emphasize the edge information. The binarization has to be performed in a method similar to the one investigated in this research.
- Among the methods investigated so far, the linear Hough transform is the most appropriate method for identifying tile-edges in the process of coarse registration.
- Identifying the circle using the circular Hough transform has some potential for the procedure center-to-tile. However, it is possible that using the linear Hough transform in an attempt to identify one tile vertex will have better utility for center-to-tile.

4.2 Recommendations for Future R&D

It is recommended that future research and design concerning the registration system will include the following components:

- Experiments to examine the utility of applying other convolution masks including noise reduction operators.
- Experiments to determine the utility of identifying at least one vertex using this vertex for centering the camera on a tile.
- Experiments to asses the utility of cross-correlation or AMDF in the process of fine registration [3].
- It is recommended to implement the new registration algorithm on the robot vision system. The development should include three main phases: testing, implementation on the Motorolla 68K board, and implementation using the Data-Cube.
    - The testing process can be done both off-line (on a Sparc work-station) and on-line (using the robot vision system). The main goals are to determine the best light configuration, and to finalize the details of the algorithms to be implemented on-line. It is proposed to obtain about 150 tile images from the Shuttle Columbia, as well as, about 150 tile images from the Shuttle Endeavor. Part of the Endeavor images will cover the same tiles obtained during this summer. These images will be used as "post-flight" images.
    - The second phase is to integrate the new algorithms with the current robot vision software on the 68K processor.
    - Finally, some of the modules of the new algorithm should be implemented on the Data-Cube. For example, the AMDF and the clustering algorithms are relatively time consuming. These algorithms can be ported to the Data-Cube.

Bibliography

[1] R. C. Gonzalez and P. Wintz, *Digital Image Processing,* (2'nd Ed.*)*, Addison Wesley, Reading, MA, 1987.

[2] S. A. Dudani and A. L. Luk, "Detecting Straight Line Segments on Outdoor Scenes", *Pattern Recognition*, vol. 10, 1987: 145- 157.

[3] L.R., Rabiner and R. W. Schafer, *Digital Processing of Speech Signals,* Prentice Hall, Englewood Cliffs, NJ, 1978.

[4] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles,* Addison Wesley, Reading, MA, 1974.

[5] Dan E. Tamir, Chi-Yeon Park and Book-Sung Too, "Vector Quantization and Clustering: A Pyramid approach", *IEEE Data Compression Conference,* Utah, March, 1995.

Appendix A: The Off-line Programs - A User Manual

Appendix A contains instructions for the user of the Off-line programs. These instructions include installing, compiling and running the Off-line programs.

A.1 Installing the Off-line programs

The source code of the Off-line programs and a "make-file" are stored in an archive in the Unix "tar- file" "off-073095.tar". To install the programs, place the file "off-073095.tar" in any directory and apply the command - "tar xvf off-073095.tar". This command extracts the following set of files into the directory:

1) Makefile
2) main.c
3) image_read.c
4) image_process.c
5) trinarize2.c
6) trinarize3.c
7) hconv.c
8) vconv.c
9) line_enhance.c
10) point_enhance.c
11) image_write.c
12) hproj.c
13) vproj.c
14) image_add.c
15) image_sub.c
16) circle_find.c
17) line_find.c
18) register.h

A.2 Compiling the Off-line program

The Off-line program consists of one user interactive program ("register") that is geared to enable the user to test several registration algorithms, and six individual programs each testing a specific registration algorithm. These programs are 1) "org" - a program to generate a sun-raster-file from a file acquired by the SRI camera, 2) "sri" - a program to test the coarse registration algorithm developed by SRI, 3) "ksc" - a modification to the SRI algorithm that has been developed by KSC stuff, 4) "trin" - a modification to the SRI algorithm that binarizes the edge-enhanced image, 5) "line" - a registration algorithm that is based on a linear Hough transform, and 6) "circle" - a frame centering algorithm that is based on a circular Hough transform.

The file "Makefile" is used to compile the Off-line programs. The command "make" or "make all" creates the entire set of programs ("register", "org", "sri", "ksc", "trin", "line", and "circle"). This command invokes the Sun "cc" compiler and linker on the source-code and generates object files and executable files.

To compile a specific program such as "register" or "sri" use the command "make register" or "make sri".

In addition, the "make-file" contains an instruction to create an archive of the source-code (C code, Makefile, and include files). To activate this command use the instruction "make tar". Another instruction available in the "make-file" is "make clean". This instruction removes all the object files and the executable files generated in the compilation stage.

A.3 Running the programs

The set of programs Off-line requires more than 8MB of stack space to run. Thus, before running the programs the user should check his stack space using the command "limit". If needed, the size of the stack space can be increased to 16384k Bytes using the command "limit stacksize 16384k".

Detailed description of the algorithms along with flow-charts of the programs are available in appendix B.

The program "register" is a menu driven program. It prompts the user for input files which are either Sun-rater-files or camera acquired files of the tiles. It produces an output file that contains the result of applying a specific algorithm to the input image. The output file is in Sun-raster format. The file name is given by the user. The following algorithms are implemented in "register":

1) An algorithm to create a Sun Raster File.

2) An algorithm for Line Enhancement.

3) An algorithm for Point Enhancement.

4) An algorithm to binarize an image using 3-means clustering and choosing cluster - 2.

5) An algorithm to binarize an image using 3-means clustering and choosing cluster - 3.

6) An algorithm to implement SRI method for detecting the tile edges.

7) An algorithm to implement KSC modification to the SRI method.

8) An algorithm to implement the binarization modification to the SRI method.

9) An algorithm to detect the circle around the re-water-proofing hole.

10) An algorithm to detect the tile edges using a linear Hough transform.

11) A procedure to add two images.

12) A procedure to produce an error image using the absolute difference between the images.

Some of these algorithms requires pre-processing by other algorithms in the set. For example, before applying the algorithm to detect the tile edges the user has to run the algorithm for line enhancement and the algorithm for binarization with cluster two. The menu instructs the user on the proper way of activating each of these routines.

In addition to the interactive program "register", a few programs that can be run from the Unix command line or from shell files are included in the Off-line programs. Unlike the interactive program "register", where some algorithms require pre-processing, the line-command programs are self-contained. That is, they operate directly on the input image and do not depend on the result of pre-processing of the input image. The following sections explain how to execute these programs.

The program "org" is executed using the command "org in out" where "in" is the name of an existing tile-image acquired from the SRI camera and "out" is a new name. The program converts the image stored in "in" into Sun raster format and stores it in the file "out".

The program "sri" is executed using the command "sri in out" where "in" is the name of an existing tile-image in Sun raster format and "out" is a new name. The program applies the SRI algorithm to the image "in" and presents the results in an image stored in Sun-raster format in the file "out".

The program "ksc" is executed using the command "ksc in out" where "in" is the name of an existing tile-image in Sun raster format and "out" is a new name. The program applies the KSC modification to the SRI algorithm to the image "in" and presents the results in an image stored in Sun-raster format in the file "out".

The program "trin" is executed using the command "trin in out" where "in" is the name of an existing tile-image in Sun raster format and "out" is a new name. The program applies edge-enhancement, binarization and one dimensional projections to the image "in" and presents the results in an image stored in Sun-raster format in the file "out".

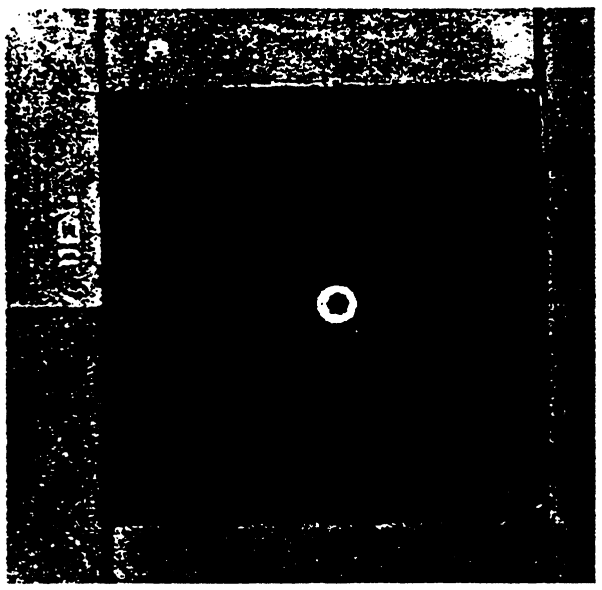
The program "circle" is executed using the command "circle in out" where "in" is the name of an existing tile-image in Sun raster format and "out" is a new name. The program applies a circular Hough transform to the image "in" and attempts to identify the circle around the re-waterproofing hole. The program "circle" presents the results in an image stored in Sun-raster format in the file "out".

The program "line" is executed using the command "line in out" where "in" is the name of an existing tile-image in Sun raster format and "out" is a new name. The program applies a linear Hough transform to the image "in" and attempts to identify the tile edges. The program "line" presents the results in an image stored in Sun-raster format in the file "out".

The output of the Off-line modules is an image in Sun raster-file format. Image display is accomplished through the shareware XV.

The following Figures illustrate the results of applying these algorithms to a tile-image. Figure 1a, contains the original image. Figure 1b, contains the results of applying edge-enhancement and projections (SRI method) on the image of Figure 1.a. Figure 1.c, is the result of superimposing the Figures 1.a and 1.b. Finally, Figure 1.d. is the result of applying the program "ksc" on the image of 1.a.

Figure 2, contains the line-enhanced image along with the results of running "trin" "circle" and "line".


a) The Tile-Image

b) Result of Convolution and
Projections

c) Output of “sri”

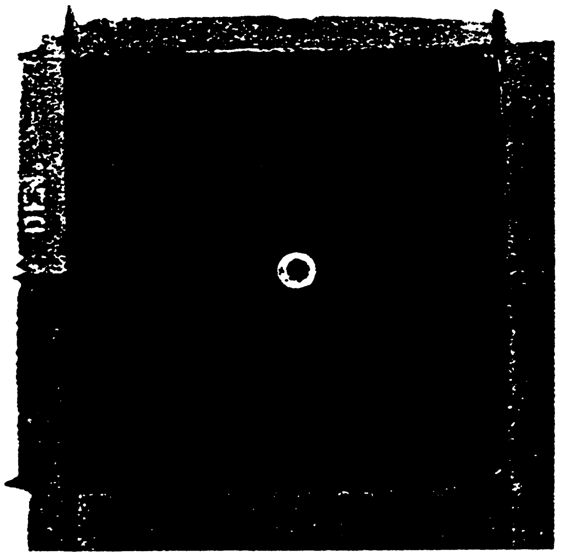
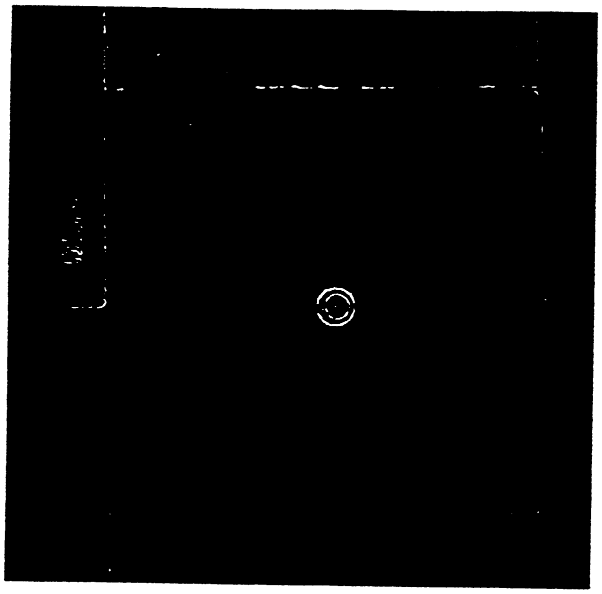
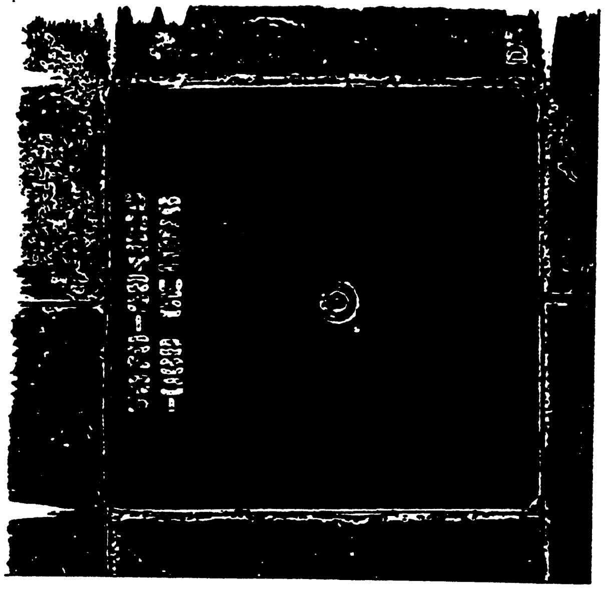

d) Output of “ksc”

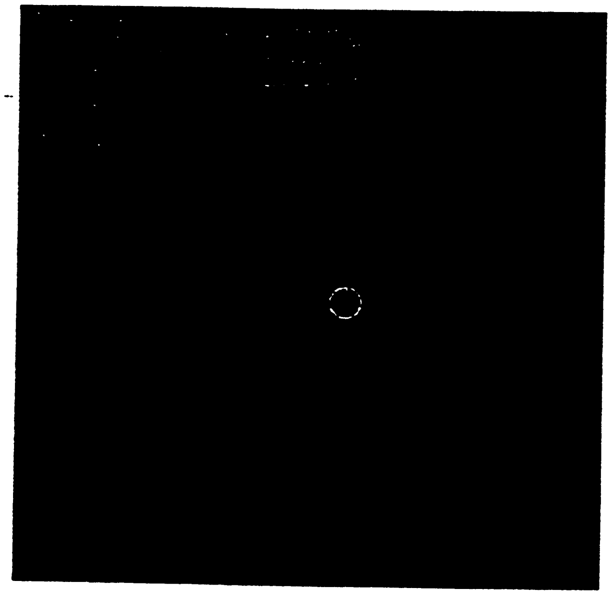
FIGURE 1. **Output of “ksc” and “sri”**



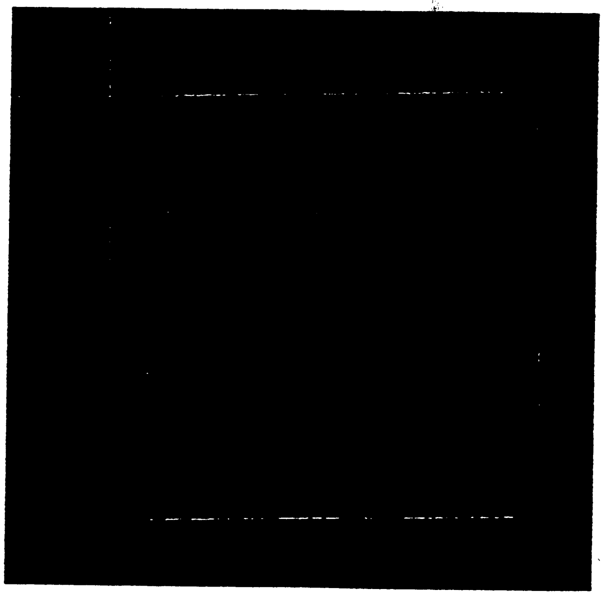
a) Output of "line_enhance"



b) Output of "trin"



c) Output of "circle"



d) Output of "line"

FIGURE 2. Output of "line_enhance", "trin", "circle", and "line"

Appendix B: The Off-line Programs - A Programmer Manual

Appendix B contains instructions for a programmer, with basic knowledge in image processing and C, who has to modify or maintain the Off-line system. The manual explains the function of the important modules included in the Off-line programs.

B.1 The program main.c

This program checks the command line arguments. If the first argument is "register" then the program enters into a switch loop where a menu is printed and selection of different items from the menu activates the program module image_process.c with an indication of the menu item selected. Image process activates specific image processing modules according to the flow diagrams of Figures 3, and 4.

If the command is "org", "sri", "ksc", "trin", "circle", or "line", then the program checks that the number of arguments is three and activates the appropriate algorithm according to Figures 3 and 4. Any other activation of the main program terminates the program.

B.2 The program image_process.c

The module image_process.c has three parameters, an integer which denotes the algorithm to be activated an input matrix that is filled by image_read.c and an output matrix that is filled by the activated algorithm. A switch statement selects the algorithm to be activated according to the parameter transferred from main.c.

B.3 The program image_read.c

The input to image_read.c is a string that contains the name of an existing file. The output is a "1024 by 1024" matrix of unsigned characters. This program opens the input file (no error checking) and checks the first 4 Bytes. If these Bytes corresponds to the Sun-raster "magic" number then the program skip the next 766 characters which are assumed to belong to the header and color-map of the file. The program then reads the next "1024 by 1024" pixels into the output array. If the first four Bytes does not match "magic", then it is assumed that it is a file in "SRI" format. That is, an image file obtained and saved through the SRI system. In this case, the program skips the next 1020 Bytes and reads "1024 by 1024" pixels into the output matrix.

B.4 The program image_write.c

The input to image_wirte.,c is a string that represents a name of a file to be created and a "1024 by 1024" matrix of unsigned characters. The program creates a file in Sun raster format (no error checking). The file contains "1024 by 1024" one-Byte pixels. Before writing the image pixels, an appropriate Sun raster header and a color-map that contains identity mapping are written into the file. By identity mapping it is meant that the (R,G,B) vector in location "i" of the color map contains the values (i, i, i). Hence a pixel with the value "i" is mapped into the gray level "i".

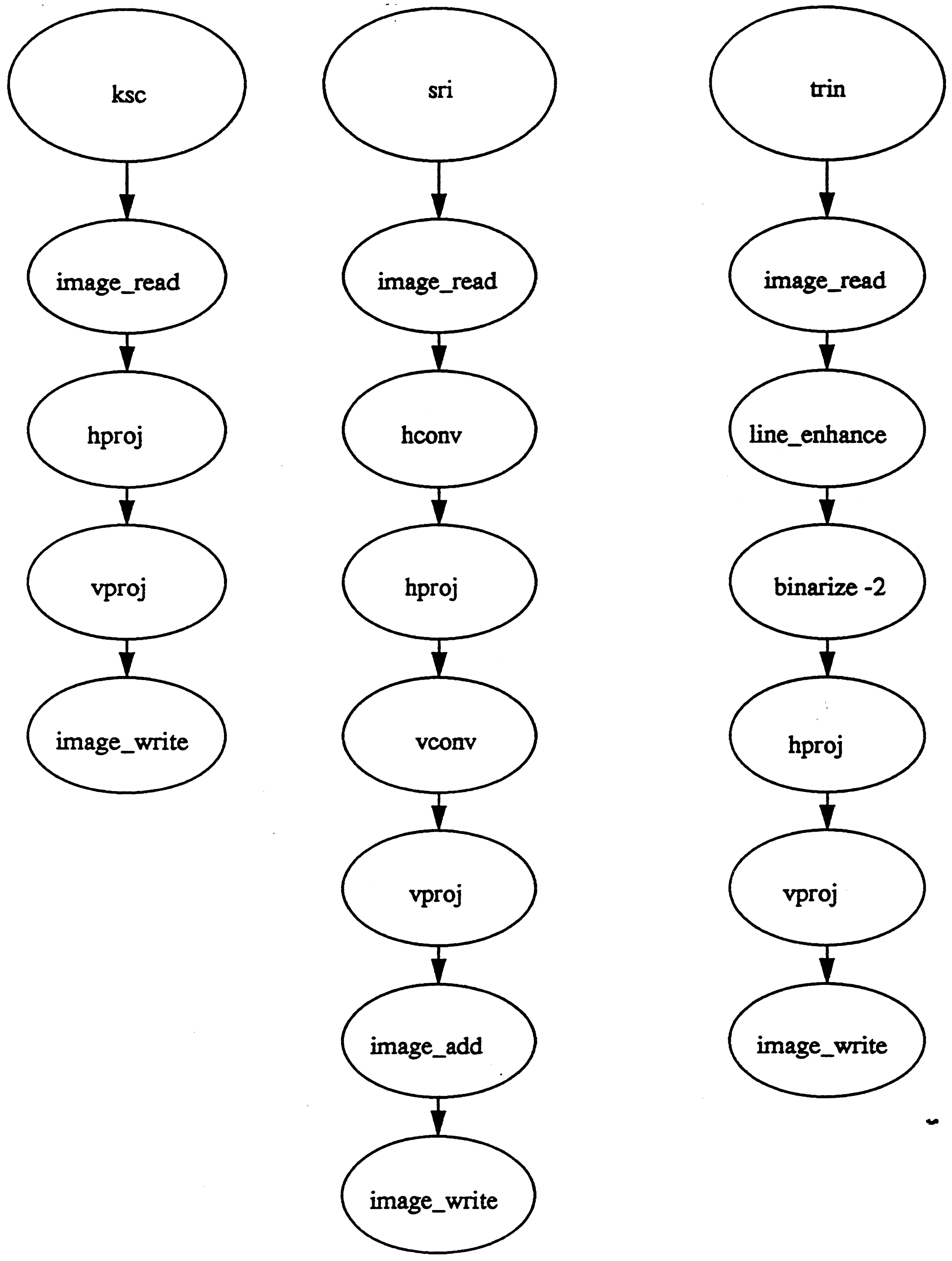


FIGURE 3. The nodule "ksc", "sri", and "trin".

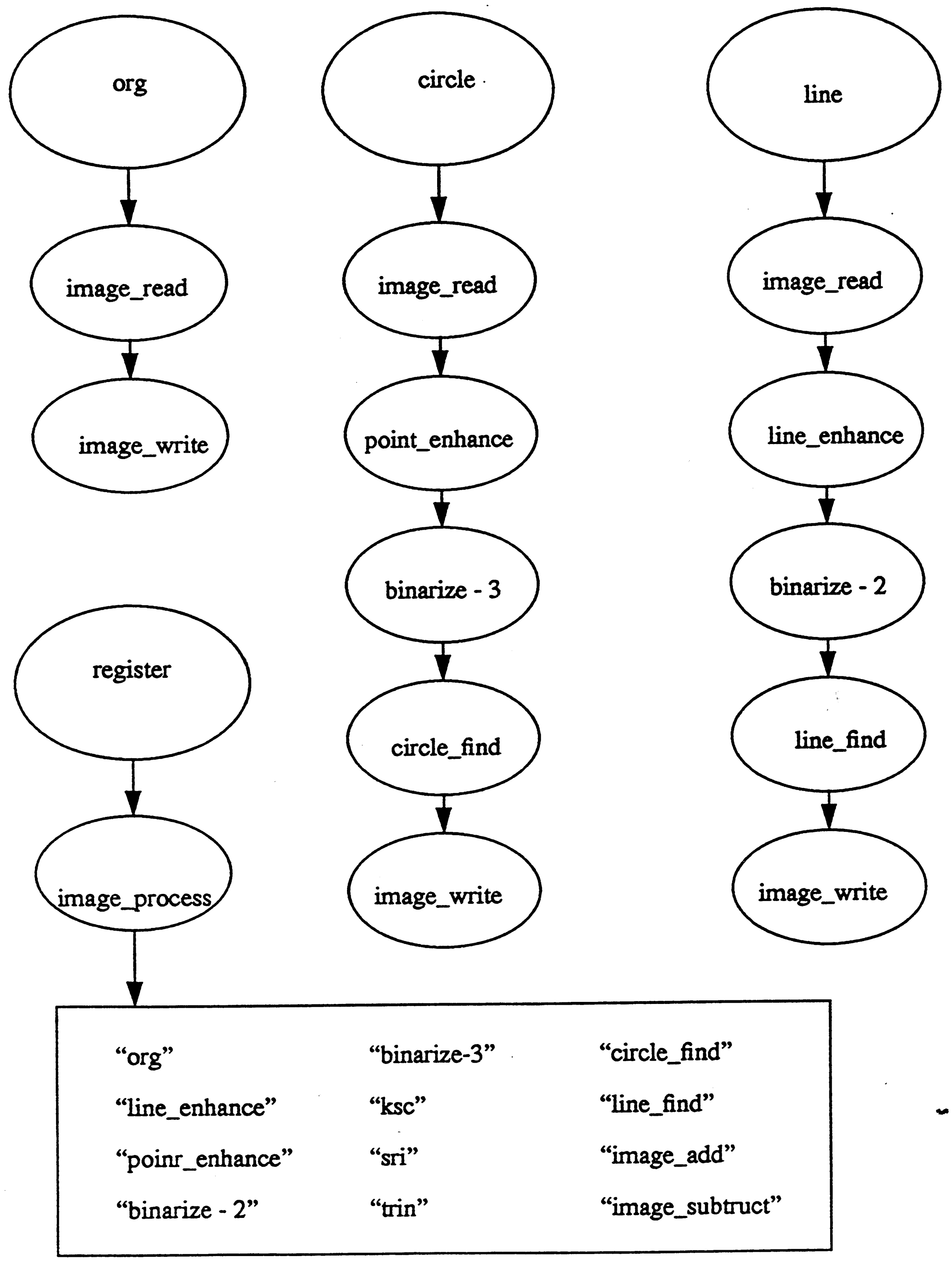


FIGURE 4. The Modules: "org", "circle", "line", and "register"

B.5 The programs hconv.c, vconv.c, line_enhance.c, and point_enhance.c

These modules implement convolution with “7 by 7” masks. The only difference between them is the masks applied. The input to these modules is a “1024 by 1024” matrix of unsigned characters. The modules apply the convolution, scale the results to the range [0,255], and write them into a “1024 by 1024” matrix of unsigned characters.

As with any convolution, a decision has to be made about the boundaries of the image. In the implementation reported here, the first and last three columns and rows of the matrix are considered to be the boundaries. These pixels are not processed by the module, effectively, they are replaced by zeros. Thus, these modules implement the following formula:

$$new(i,j) = \sum_{k=-3}^{3} \sum_{l=-3}^{3} old(i+k, j+l) \times mask(k,l)$$
$$(i,j) = 3, 4, \ldots, 1020$$

Where “old” is the matrix holding the image to be processed, “new” is the matrix holding the convolution results, and “mask” is the matrix that holds the convolution mask. It is assumed the rows and columns of the matrices “old” and “new” are indexed from 0 to 1023, and that the rows and columns of “mask” are indexed from -3 to 3.

The individual masks applied by the different modules can be found in the source code. In general, the module hconv.c is applying an horizontal mask, the module vconv.c is applying a vertical mask, the module line_enhance.c applies the mask of hconv.c and the mask of vconv.c in one path, and the module point_enhance.c is applying in one path a modified version of the horizontal and vertical masks. The main difference between the line enhancement masks and the point enhancement masks is that the point enhancement masks include additional weight for the center pixels. This tends to emphasize a small local change and not just horizontal/vertical edges. It has been found that the point enhancement masks produce better pre-processing for the identification of the circles around the re-water-proofing holes.

B.6 The programs hproj.c vproj.c

These two modules compute the horizontal and vertical projections of an image. The input to the modules is a “1024 by 1024” matrix of unsigned characters. The programs generates an array (hproj or vproj respectively) of 1024 elements according to the following formulae:

$$\left(hproj(i) = \sum_{j=0}^{1023} image(j,i) \right) \; ; i = 0, 1, 2, \ldots, 1023$$

$$\left(vproj(i) = \sum_{j=0}^{1023} image(i,j) \right) \; ; i = 0, 1, 2, \ldots, 1023$$

Where “image” is a “1024 by 1024” matrix, “hproj”, and “vproj” are arrays of rank “1024”. The modules hconv.c and vconv.c scale the arrays “hconv” and “vconv” to the range [0,127] and then generate an output matrix in which the first 128 rows/columns contain an "histogram" representation of the results of the projections. Figure 1.b from page 19 illustrates this histogram.

B.7 The programs trinarize2.c and trinarize3.c

These programs implement the K-means algorithm with the LBG termination condition on the one dimensional data set obtained form the gray-level values of the pixels in an input image [4], [5]. In this application "k" is selected to be 3 and the initial cluster centers are 0, 127, and 255, respectively. It is assumed that after convergence, the first cluster center represents the background, the second cluster center correspond to weak edges, and the third cluster center corresponds to strong edges. This assumption has been confirmed through a large set of experiments.

Strong edges correspond mainly to the white characters, white circles, and other white marks, drawn on the tile by the Shuttle maintenance crew. Therefore, applying point enhancement and 3-means clustering to the image and then picking cluster 3, filters out most of the information from the image and leaves just the characters, circles, and other white marks. This is exactly what trinarize3.c is doing. It is generating a binary output image in which every pixel that has been clustered into cluster 3 is assigned the value 255. Other pixels are assigned the value 0. This image can be used for the circle identification (see Figure 2.c).

Weak edges correspond mainly to the edges of the tile. Therefore, applying line enhancement and 3-means clustering on the image and then picking cluster 2, filters out most of the information from the image and leaves just the tile edges. This is exactly what trinarize2.c is doing. It is generating a binary output image in which every pixel that has been clustered into cluster 2 is assigned the value 255. Other pixels are assigned the value 0. This image can be used for the line identification (Figure 2.d).

The current version of trinarize2.c and trinarize3.c have not been optimized for performance. If these modules are to be used in the on-line system, it is recommended to either use the Data-Cube hardware for the implementation, or to use a fast version of K-means developed by Tamir [5].

B.8 The program circle_find.c

This program operates on a binary image. It is modeling the operation of "drawing" a circle around every non-zero pixel in the input image and uses the results of this modeling to update a scratch-pad matrix. Let (y,x) be the coordinates of a non-zero pixel in the original image. The coordinates (y_i,x_i) of the points that lie on a circle with radius "R" around (y,x) are computed using the following formula:

$$(y_i = y + \lfloor R \times \sin(i) \rfloor)\ (x_i = x + \lfloor R \times \cos(i) \rfloor)\ i = 0, 1, 2, \ldots, 180$$

In this formula, $\lfloor x \rfloor$ represents the floor of "x". The radius of most of the circles around the re-water-proofing holes is approximately 28 pixels. Hence, "R" is set to 28 in the module circle_find.c.

The scratch-pad matrix "accum" is a "1024 by 1024" short integers matrix. It is initialized to contain zeros in all the places. The module circle_find.c computes the coordinates of all the points (y_i,x_i) that lie on circle with a radius of 28 pixels around every non-zero pixel in the input image. For every computed point (y_i,x_i) the module circle_find.c increments the entry (y_i,x_i) in the matrix "accum" by one. In the next stage, the function circle_find.c searches for the maximum value in "accum" assuming that the location with maximum value corresponds to the center of the re-water-proofing hole verification circle. The program then dims the original image by dividing every gray level by two and draws a white (gray-level of 255) circle with a radius of 28 pixels in the original image around the identified center. The resultant image is the output of the program. Figure 2.c from page 19 illustrates the output of circle_find.c with respect to the image from Figure 1.a.

B.9 The program line_find.c

This program operates on a binary image. It is modeling the operation of "drawing" lines that pass through non-zero pixel in the input image and uses the results of this modeling to update a scratch-pad matrix. In this application, a line is represented using the normal representation [1]. Under this representation the (y,x) coordinates of the points that lie on the line correspond to the following equation:

$$x \times \cos(\theta) + y \times \sin(\theta) = \rho$$

Where θ is the angle between the x-axis and a vector perpendicular to the line, and ρ is the distance of the line from the origin.

Let (y,x) be the coordinates of a non-zero pixel in the original image. The program line_find.c evaluates several lines that pass through (y,x) these lines correspond to the equation:

$$x \times \cos(\theta_i) + y \times \sin(\theta_i) = \rho_i$$

Under the current assumptions about the accuracy of the robot location system, the possible range for ρ in pixels is [0,255] and [768,1023]. The possible range for θ in degrees is $[-5^\circ, 5^\circ]$ and $[85^\circ, 95^\circ]$. However, to increase accuracy, the angle θ is measured with an accuracy of 0.2 degrees. Thus, given a non-zero pixel in location (y,x), the program line_find.c computes the integer values of all the ρ_i and θ_i, such that the point (ρ_i, θ_i) complies with the accuracy constraints and satisfies the equation $x \times \cos(\theta_i) + y \times \sin(\theta_i) = \rho_i$.

The values of ρ_i in the range [768,1023] are mapped to the range [256,511]. The values of θ_i in the range [-5,5] (with accuracy of 0.2) are mapped into the range [0,49] and the values of θ_i in the range [85,95] (again, with accuracy of 0.2) are mapped into the range [50,99].

The scratch-pad matrix "accum" is a "512 by 100" matrix of short integers. It is initialized to contain zeros in all the places. For each non-zero pixel at location (y,x) the module line_find.c computes the (ρ_i, θ_i) parameters of all the lines that pass through (y,x) and comply with the distance and angle constraint. Let (ρ_i, θ_i) be such a pair of parameters. The module line_find.c increments the corresponding entry in the matrix "accum" by one. In the next stage, the function line_find.c searches for maximum values in four ranges of "accum" assuming that the locations with maximum values corresponds to the individual edges of the tile. The ranges are:

1) $((0 \le \rho_i \le 255), (0 \le \theta_i \le 49))$,
2) $((256 \le \rho_i \le 511), (0 \le \theta_i \le 49))$,
3) $((0 \le \rho_i \le 255), (50 \le \theta_i \le 99))$, and
4) $((256 \le \rho_i \le 511), (50 \le \theta_i \le 99))$

The program then dims the original image by dividing every gray level by two and draws inside the dimmed image four white (gray-level of 255) lines corresponding to the (ρ_i, θ_i) of the four maximum values. The resultant image is the output of the program. Figure 2.d from page 20, illustrates the output of line_find.c with respect to the image from Figure 1.a.

B.10 The program image_add.c

The input to this program is two “1024 by 1024” matrices of unsigned characters. It is applying the following formula:

$$new(i, j) = max(255, (old1(i, j) + old2(i, j)))$$

where “old1” and “old2” are the two input matrices, “max” stands for maximum, and “new” is the output matrix.

B.11 The program image_sub.c

The input to this program is two “1024 by 1024” matrices of unsigned characters. It is applying the following formula:

$$new(i, j) = abs(old1(i, j) - old2(i, j))$$

where “old1” and “old2” are the two input matrices, “abs” stands for absolute value, and “new” is the output matrix.

B.12 The header file register.h

This file contains the following definitions:

```
# include <stdio.h>
# include <math.h>
# include <memory.h>
# include <string.h>
# include <pixrect/pixrect_hs.h>
# define PI 4 * atan(1.)
# define RAS_MAGIC 0x59a66a95
# define MAGIC 0xa659956a
# define DEPTH 8
# define TYPE 1
# define MAPTYPE 1
# define MLENGTH 768
#define IMAGE_X 1024
#define IMAGE_Y 1024
```