Teaching Robot Vision in Manufacturing Technology

Zhongming Liang
Purdue University Fort Wayne

Abstract

This paper discusses a number of experiments developed for teaching robot vision. The experiments help students with fundamental theories of machine vision and its applications in robotics.

Introduction

With machine vision playing an increasingly important role in areas of robotics such as inspection, identification, and visual servoing and navigation,[1] the manufacturing technology department sees the importance of teaching fundamentals of machine vision. It has been a difficult topic to teach since it involves a number of concepts that many students in manufacturing technology programs are not familiar, especially when laboratory support was not completely ready.

In the spring and the summer of 1995, with help of a student majoring in electrical engineering technology, the author used the basic vision system to develop a number of experiments for robot vision. They include thresholding, image binarization, edge detection, object recognition, image feature extraction and random object picking.

This paper will briefly discuss all the experiments listed above. Computer programs will be available at the presentation. In addition, video showing the experiments will also be included in the presentation.

The Vision System

The vision system includes a Sony XC-57 monochrome CCD (Charge-Coupled Device) camera, a Sony professional video monitor, and a Data Translation DT 2851 frame grabber, which is installed in a 486 IBM-compatible personal computer. The camera has 512 by 492 photosites and outputs analog signals in the EIA RS-170 format. The video monitor can display original image from the camera or processed image from the frame grabber in either the RS-170 standard or the NTSC standard.[2] The frame grabber, shown in Figure 1, grabs one monochrome video frame from the camera within 1/30 second, converts it to a digital frame of 512 by 512 pixels of eight bits each, and stores it in either buffer 0 or buffer 1 on the board. The digital frames in the buffers can be transferred to the internal memory of the computer for further processing, analysis, and storage. Each look-up table (LUT) stored in the on-board memory is a conversion table of light intensity values for altering the image in a frame. For example, the LUT

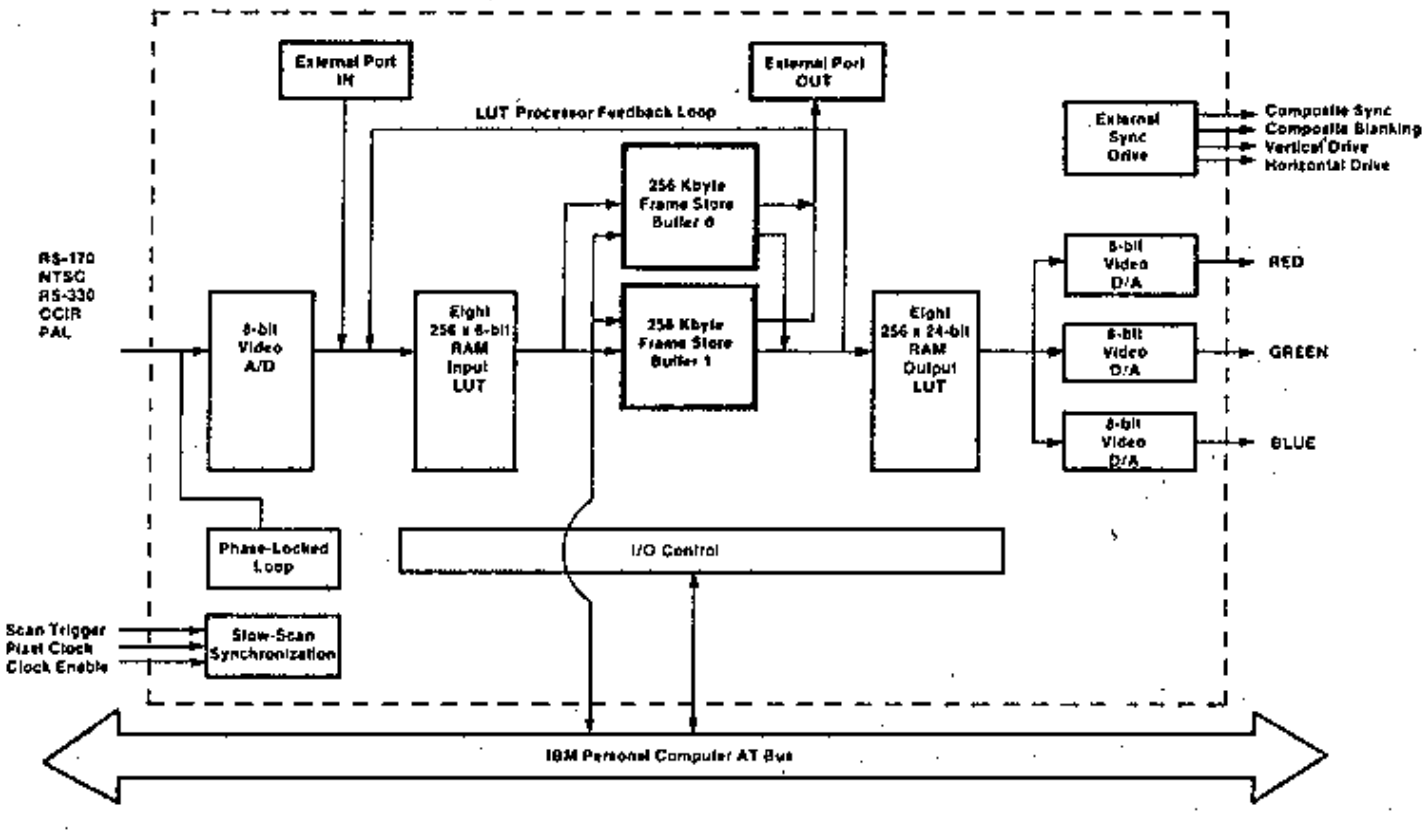

Fig. 1 The frame grabber block diagram

... ...	... ...
(row 100)	0
(row 101)	255
... ...	... ...
(row 256)	255

maps the intensity values to 0 for all those below 100 and to 255 for all the others and thus converts a gray image to a black-and-white binary image with a threshold value of 100.

The basic vision system can be used in either the interactive mode or the programming mode. In the interactive mode a user issues vision commands such as

- Acquire 1 Frame to Buffer 1
- Place_Cursor At Row 60 Column 200
- Select Input Table 3
- Convolve Buffer 1 with Laplacian

In the programming mode, a user places vision subroutine calls in a BASIC or C program. Examples of the vision subroutines in a vision library are[3]:

- ISACQ Bb%, Nn% -- acquiring a number of image frames to a buffer
- ISREPC R%, C% -- reporting the current location of the cursor
- ISGETP Bb%, R%, C%, N%, D%(1) -- copying the light-intensity values of a number of pixels in an on-board buffer to a program array stored in the internal memory

In the programming mode vision commands are integrated with robot commands, programming loops, and arithmetic and logic commands, which allows input from the vision system to be further analyzed and utilized

for various applications. The vision experiments discussed in the next session are performed in the programming mode.

Vision Experiments

In this session, the main ideas of several vision experiments will be presented. Complete QBASIC programs will be available at the presentation.

Converting a gray image to a binary image

The conversion is performed in the following steps.

- Acquire 1 frame to buffer 1 and display buffer 1.
- Load the function values of the gray-scale histogram of the image to a program array HISTOGRM$(256).
- Array HISTOGRM$(256) is converted to BARGPH%(64) to fit the computer screen in graphical display.
- Draw vertical bars to show the array BARGPH%(64). Figure 3 is the graphical gray-scale histogram of the image in figure 2, which shows the university logo in a rectangular piece.

Fig. 2 A gray image

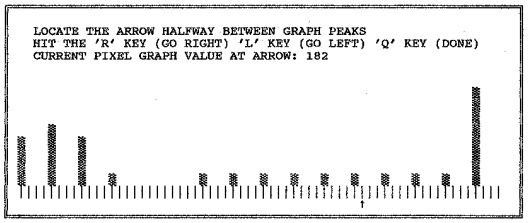

Fig. 3 The histogram of the image

- An arrow beneath the abscissa indicates the current threshold value for gray-to-binary conversion. The vision monitor concurrently shows the resulting binary image. For example, with the threshold value shown in figure 3, the binary images in figure 4 is displayed.
- The user moves the arrow horizontally to try different threshold values. When the arrow is moved
  - The computer screen indicates the current light-intensity threshold value.
  - The vision monitor displays the updated binary image.
- When the user exits at the best threshold value, the system saves the input look-up table file and the associated binary image on a disk.

Extracting geometric properties of an object

- Locating the centroid of an object

- Converting the image into binary such that the object and only the object is black.
- Computing the means of the row numbers and the column numbers of all the pixels whose light-intensity values are zeros.
- The means are the row and column coordinates of the centroid.

- Locating the principal axes of a rectangular object
  - Computing the covariance matrix of the coordinate vector population of all the black pixels. The covariance matrix is

$$\mathbf{C}_x = E\{(\mathbf{x} - \mathbf{m}_x)(\mathbf{x} - \mathbf{m}_x)^T\}$$

    where **x** represents a population of two-element column vectors of the row and column coordinates of all the black pixels, $\mathbf{m}_x$ is the mean vector of the population, and the operation $E\{\text{matrix}\}$ computes the expected value of each matrix element.[4]
  - Computing the two eigenvalues and eigenvectors of the covariance matrix.
  - The eigenvector corresponding to the eigenvalue larger in magnitude is the vector of the longitudinal sides of the rectangular object. Compute the angle of the vector based on its row and column coordinates and the aspect ratio of the pixels.

Detecting the edge of an object

- The edge of an object can be detected from a binary image or directly from a gray image.
- Reading three rows of pixels at a time and convoluting the middle row using the Laplacian filter

$$\begin{vmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{vmatrix}$$

Figure 5 shows the result of convoluting the image in figure 4.

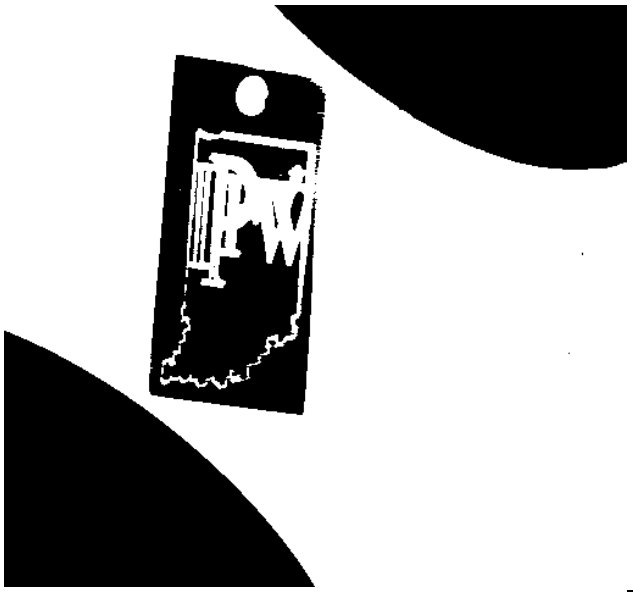

Fig. 4 The binary image from the gray image in Fig. 2 using the threshold value shown in Fig. 3

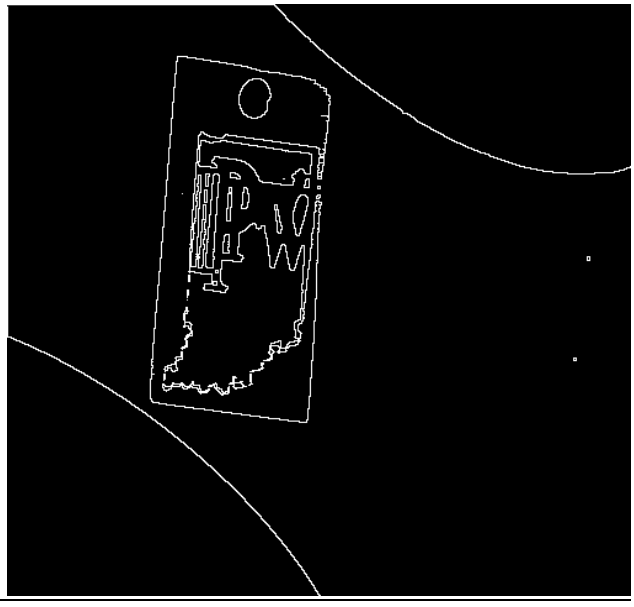

Fig. 5 The convolution result of the image

Object recognition

For example, the vision system can help the robot separate objects of two different groups. The objects in one group each has a hole at a certain location. The objects in the other group do not have the feature.

- Placing an object from the first group at the reference location and adjusting the illumination such that there is a clear contrast in light intensity between the hole area and its surrounding area.
- Recording the row and column coordinates of the hole center and the average light-intensity value of the hole area.
- In the robot program, reading the pixels at the location of a possible hole. If the average light-intensity value is high, then here is a hole in the object and the robot moves the object to the location for the group.

Guiding the robot to pick randomly placed rectangular objects

- Computing the coordinates of the centroid of the object, and converting them to the true horizontal distances from the object to the camera based on the magnification of the camera lens.
- Converting the horizontal distances from the camera to the object center to the required movement of the robot, and computing the rotation angle of the robot base joint for the movement, say θ_b.
- Computing the angle of the rectangular object with respect to the camera, and converting it to the angle of the object with respect to the robot gripper jaws, say θ_1.
- Angle $(\theta_1 - \theta_b)$ is the required roll rotation of the robot wrist for aligning the gripper jaws with the longitudinal sides of the object.

Conclusions

Relatively simple and interesting experiments for teaching robot vision in the course of robotics applications for students in the department of manufacturing technology have been presented. These experiments help students with understanding fundamental concepts and techniques of robot vision.

The vision system is relatively slow. It takes approximately fifteen seconds to locate the centroid, and another fifteen seconds to determine the orientation of a rectangular object. Together it is approximately half a minute from the time the robot sees a randomly placed object to the time it catches the object. It is because on one hand we have only limited hardware and on the other hand we use QBASIC language, with which the students in the course are familiar.

References

1. Goover, M.P. et al., "Industrial Robotics: Technology, Programming, and Applications," McGraw-Hill, 1986.
2. "User Manual for DT2851 High Resolution Frame Grabber for the IBM Personal Computer AT," Data Translation, Inc., 1991.
3. "R700 Series Robotic Systems Course Manual," JZC and Rixan Associates, 1992.
4. Gonzales, R.C. and Woods, R.E., "Digital Image Processing," Addison-Wesley Publishing Company, 1992.

ZHONGMING (WILSON) LIANG
graduated from South-China Univ of Science & Technology in 1966. He earned an MS degree from Huazhong Univ of Science & Technology in 1981 and an ME degree from the City College of New York in 1982. He made very good progress toward his Ph.D. degree at Stevens Institute of Technology from 1983 to 1987. He was with a company as a design engineer for ten years and is a registered professional engineer in Indiana.