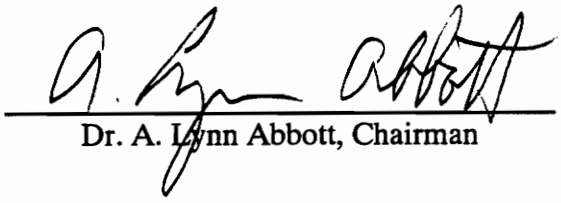
A Recognition System for Rectangular Components Based on Structural Decomposition

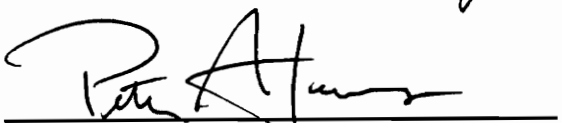
by
Christopher John Del Gigante

Thesis submitted to the Faculty of the
Bradley Department of Electrical Engineering
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
Master of Science
in
Electrical Engineering

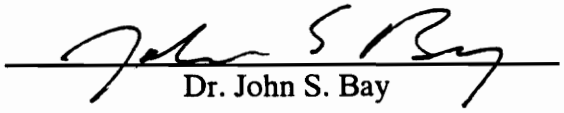
APPROVED:



Dr. A. Lynn Abbott, Chairman



Dr. Peter Athanas



Dr. John S. Bay

July, 1995
Blacksburg, Virginia

A Recognition System for Rectangular Components Based On Structural Decomposition

by

Christopher John Del Gigante

Dr. A. Lynn Abbott, Chairman

Electrical Engineering

Abstract

The practical implementation of nontrivial machine vision algorithms requires a balance of efficiency, flexibility, and cost. This paper discusses the development and implementation of an industrial system that recognizes scanned rectangular kitchen cabinet frames. By utilizing a configuration of modular algorithms that reduce the image into structural primitives, accurate recognition is possible at a relatively high speed using limited hardware. The system essentially decomposes a silhouette of the frame into a border representation, extracting corner-containing regions which, in turn, yield vertices. Global information about the frame is then used to convert the vertices into usable features. This thesis discusses the motivation, development, and implementation of this system. Recognition tests were performed successfully on several thousand frame samples.

Acknowledgments

I would like to express special thanks to the following people for providing me with support during the development of the frame recognition system and the writing of this thesis:

- My thesis committee for their contributions and support. I would especially like to thank Dr. A. Lynn Abbott for his suggestions and his leadership as Principal Investigator for this project.
- Alfred Foster of American Woodmark Corporation for being the factory-side coordinator of the frame-recognition project and providing assorted background material.
- Allen Pulsifer of Bitflow, Inc. for his invaluable technical support regarding the Data Raptor-VL Frame Digitizer.
- Angela Nobili and Nathan Carpenter for various aid in testing the system.
- My family for their support.

Table of Contents

1. Introduction and Background

1.1 Industrial Machine Vision

For factories to streamline their manufacturing processes, they must increase production efficiency. One way to do this is to implement automatic inspection and quality control. Without automation, inspection must be done manually--a tedious, time-consuming task that requires a broad familiarity with the factory's products. Even with trained personnel, errors can still be made. Through automation, errors can be reduced, the quality of inspection can be consistently maintained, and human resources can be devoted to less automatic portions of production.

Many inspection tasks are vision-oriented. Examples of this are inspection of a part's finish, checking for surface defects, sorting components by general type, and comparing parts against some ideal. A computer that processes visual information can be used to perform these tasks.

This processing is called machine vision. In the case of sorting or part matching, the general procedure is to identify several *features*, and perform matches based on them. A *feature* is any measurement that is made on an image or a region [PAR94], such as length or position. This thesis discusses a system that identifies parts based on a database of the factory's product line.

1.2 American Woodmark Corporation (AWC)

American Woodmark Corporation, based in Winchester, Virginia, is a manufacturer of kitchen cabinets and other vanities. They have several plants distributed about the nation where each performs some portion of the manufacturing process. Their overall process can be divided into four steps:

1. Piece manufacturing
2. Piece assembly
3. Finishing
4. Vanity system assembly

At one plant, small pieces of wood are cut to size, drilled, and routed as necessary. These components are shipped by truck to a second plant, where the pieces are joined by peg and wood staple to make the front parts of the cabinets and vanities, primarily frames and doors. After assembly, these components are finished by sanding and staining to specification. The assembled parts are then shipped to another plant, where complete cabinet units are assembled.

During the finishing operation at the second plant, sprayers create a "curtain" of varnish. As the parts move through this curtain on a conveyor, they are coated with varnish, then later dried. Since the sprayers operate continuously, the parts must be packed very closely to prevent waste. In order for this packing to be efficient, parts are mixed up, with small doors often being placed in the openings of larger frames. Since this effectively scrambles the order of the parts on the line, the parts must be recognized manually before they are inventoried.

American Woodmark has identified this recognition task as a critical step in the manufacturing operation. The task is very tedious and requires a high level of familiarity with the parts. Ultimately, mistakes cost the company. When a manufactured part is mislabeled, it is placed into inventory incorrectly. Inventory errors cause the delayed or

incorrect filling of customer's orders, which could affect business. For the company to develop and expand, an automatic recognition system is needed, which should reduce errors and facilitate an inventory tracking system.

The frame recognition system described in this thesis is one of four separate systems, that when combined, will provide complete recognition for the majority of AWC's product line. The four systems are: Door Recognition, Frame Recognition, Color Classification, and a Bar Code Printer/Database. Figure 1.1 below shows how these systems interrelate.

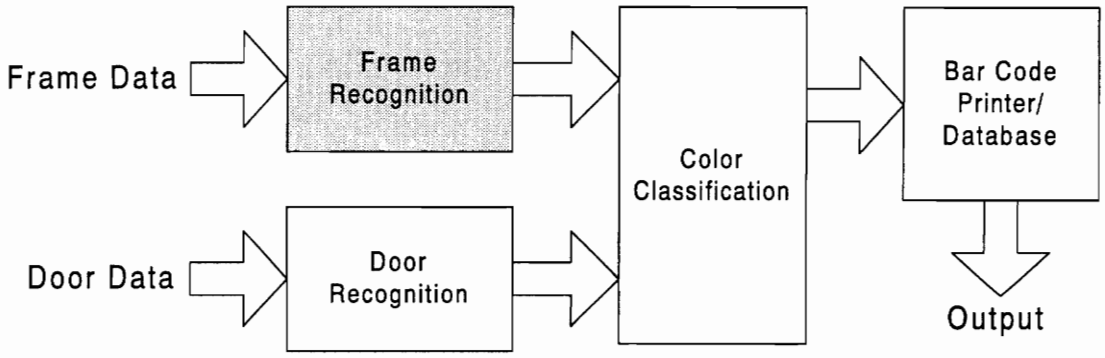


Figure 1.1: Overall automatic recognition system. Frames and doors are recognized by two separate systems. After recognition, the color of the part is classified, using the same system regardless of type. The bar code printer/database system collects all of the recognition and classification data and prints out a label to be placed on the part. This thesis discusses the shaded part of the diagram.

This system configuration will permit a user to access up-to-the-minute information about rejects, distributions, and inventory amounts through a central system, aiding in quality control.

1.3 Representation of Rectangular Frames

Frames serve as the fronts of cabinets, providing the mounting points for doors and cabinet hardware. Figure 1.2 shows a frame mounted on a vanity cabinet.

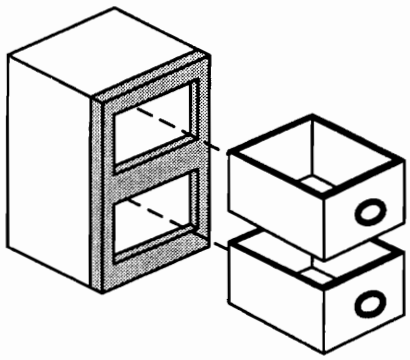


Figure 1.2: A vanity cabinet assembly with drawers. The shaded part is the cabinet frame.

Many different frame styles are made to serve in different roles, and support a wide variety of drawer types and doors. Yuan [YUA94] characterizes the differences between door styles and sizes in developing a door recognition system. [BAR94] developed a description of the frame styles. In general, frames are flat, approximately 1/2" thick, rectangular, and subdivided into smaller rectangular openings for doors and drawers. Two typical frames are shown in Figure 1.3.

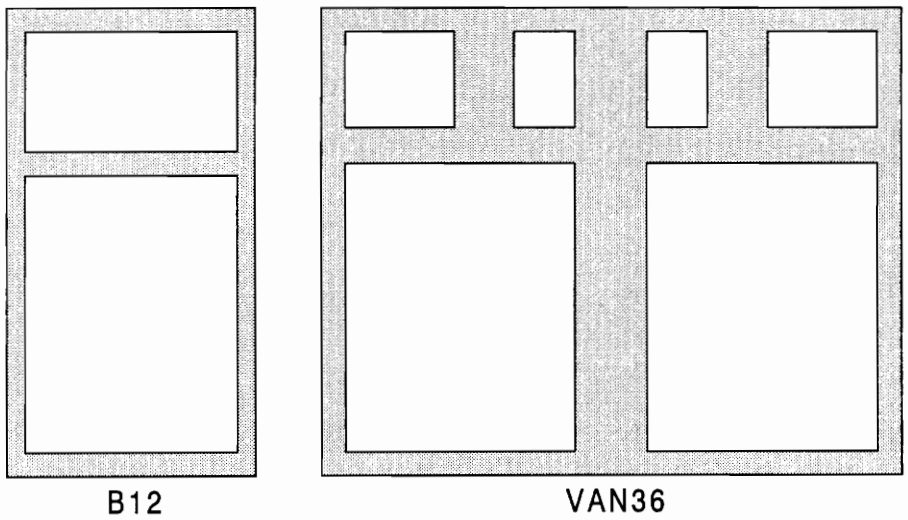


Figure 1.3: Typical frames (not drawn to scale). Frame style B12 has two openings, while VAN36 has six.

Frames can vary in the following ways:

1. Number of Openings

   To satisfy the different needs of customers, frames have from one to eight interior openings for the mounting of drawers and doors. This the most recognizable difference between the different frames. However, many frames have the same number of openings, so this measure cannot exclusively be used to differentiate them. Let the number of interior openings in a frame be n. For the purposes of this thesis, the borders of the openings will be numbered 1 to n. The outside border of the frame will informally be called an opening and will be assigned the index 0.

2. Size of Openings

   The sizes of frame openings vary from style to style to fit assorted drawers and doors. In the current frame product line, there can be up to four different opening sizes in a six opening frame. The outside border of the frame also varies in size. The size of the outside border is typically reflected at least approximately in the part name. For example, UT3696, a tall frame used in making narrow utility cabinets, is approximately 36" wide by 92" high. Frame openings and outside borders are measured horizontally and vertically in inches. Given n openings, the dimensions D_k of the kth opening are represented by the ordered pair

$$D_k=(c_k, r_k) \tag{1.1}$$

   where k is the opening number, $k\in\{0, 1, 2, ..., n\}$, c_k the width of the kth opening in inches, and r_k is the height of the kth opening in inches. By this notation, the areas of all of the openings can be represented by a list of ordered pairs.

3. Displacement of Openings

Additionally, the frames' openings are placed at various positions relative to the top and left edges of the frame. Just as with the size of the openings, displacement is measured horizontally and vertically in inches. Defining the horizontal as the x direction, and vertical as the y, let this x and y displacement be represented by the ordered pair (i_k, j_k), where k is the opening number.

Combining the displacement and opening area measures into a matrix λ, yields the compact notation

$$\lambda = \begin{bmatrix} i_0 & j_0 & c_0 & r_0 \\ i_1 & j_1 & c_1 & r_1 \\ . & . & . & . \\ . & . & . & . \\ i_n & j_n & c_n & r_n \end{bmatrix} \tag{1.2}$$

All of these values in λ can be easily obtained through measurements of the frames or from production blueprints. Graphically these four parameters are depicted in Figure 1.4. Through variation of these parameters, American Woodmark produces a total of 253 different frame styles.

Bari determined that, in almost all cases, the smallest size increment for (c_0, r_0) between different frame styles is 0.5" in width, and 1.5" in height [BAR94]. As a consequence, the first two features, the number of openings and their size, are sufficient to discern more than 98% of the frames. The displacement will be used primarily to order the openings in a consistent way based on their distance from the frame origin, O (see Figure 1.4).

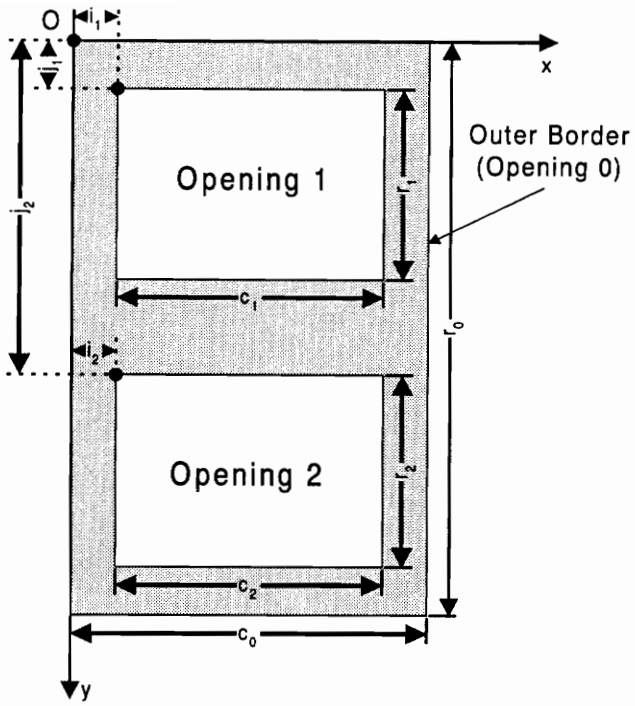


Figure 1.4: Graphical depiction of the three frame parameters. All displacements are measured relative to the frame origin (i_0, j_0)=(0, 0), the upper left-hand outside corner of the frame.

1.4 Unusual Frame Styles

In addition to these 253 purely closed, planar, rectangular frames there exist seven frame styles that cannot be specified exclusively by these three parameters. They are as follows. (Figures 1.5 through 1.8 are not drawn to scale.)

1. AGS24/AG2418

   The AGxx series of frames are used as add-ons to other frames, allowing frames to be placed in unusual positions such as corners. The only difference between them and

other frames is that they are one piece, but open on one end, like a "C". Figure 1.5 shows a diagram of an AGxx series frame.

Figure 1.5: AGxx series frame schematic.

2. BCxx Series

Whereas all of the standard frame types are strictly rectangular, the BCxx series are one-opening frames with a decorative French curve. There are three frames in the BCxx series: BC18, BC24, BC30. Figure 1.6 shows a representative diagram of a BCxx series frame.

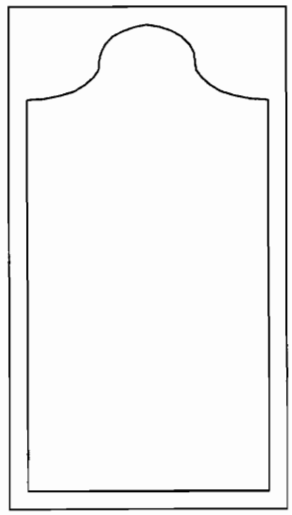


Figure 1.6: BCxx series frame schematic.

3. HU36

HU36 is a combination of the AGxx series and the BCxx series. Figure 1.7 shows a schematic of this frame:

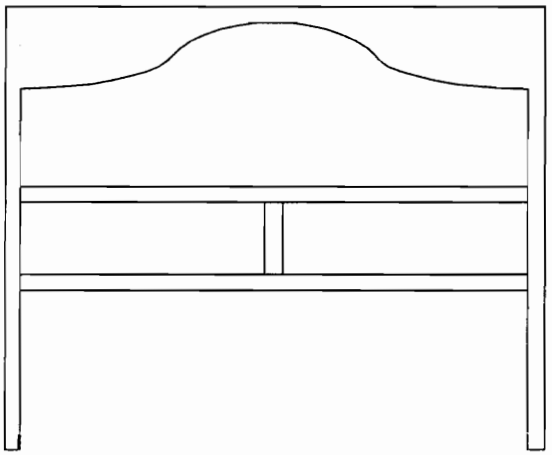


Figure 1.7: HU36 Frame Schematic.

4. WER/O and CAR/O

WER/O and CAR/O are identical to their "normal" counterparts, WER and CAR, save that they each have two cutouts in the left hand side of the interior of the frame opening for recessed hinges. Figure 1.8 shows a schematic of a WER/O frame.

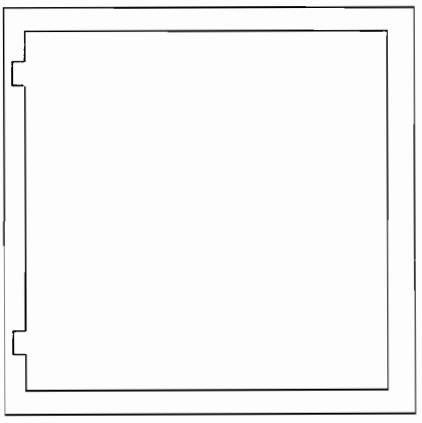


Figure 1.8: WER/O frame schematic.

One other item of note is that frame style pairs (WER, CAR) and (WER/O, CAR/O) are identical in all parameters save that the CAR series is approximately 3/8" taller than the WER series. This very slight height difference occurs with a few of the other frames in production. However, since this situation occurs rarely, it will be treated as an exception.

Since these exceptions make up such a small portion of annual production, they will all be treated as special cases, not bound by the recognition system this thesis discusses. However, to show the flexibility of the system, the AGxx series will be implemented using a extension to the implemented algorithm (see Appendix B).

1.5 Mathematical Specification of a Cabinet Frame

It is possible to model a boundary representation of a cabinet frame as a ***graph***. A graph G is an ordered triple (V, A, ϕ), where V is a arbitrary set, A is a set of ordered pairs of the elements of V, and ϕ is a set of incidence functions with a one-to-one correspondence to the members of A [ROB65, BON76]. In a graphical depiction of G, V is typically represented using a set of arbitrarily placed points, A is a set of edges spanning the members of V, and ϕ is a set of values that correspond to those edges. Figure 1.9 depicts examples of several graphs.

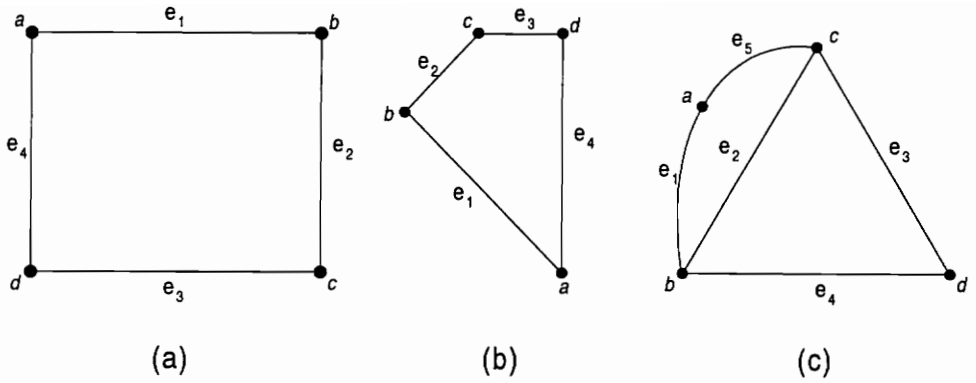


Figure 1.9: Sample Graphs. In the case of all three graphs $V=\{a, b, c, d\}$. In (a) and (b) $A=\{(a, b);(b, c);(c, d);(d, a)\}$. Since the two graphs have equivalent Vs and As, they are isomorphic--they represent the same information. Although (c) shares the same V, it is not isomorphic to (a) and (b) because it does not have the same A as they do. e_n are the incidence functions relating the elements of V to one another in A for each of the graphs.

Graphs can also possess several properties, some of which are listed below.

- *Isomorphic.* If two graphs G_1 and G_2 exist where $G_1=G_2$, the G_1 is said to be isomorphic to G_2. For instance, Figures 1.9(a) and 1.9(b) are isomorphic to one another.
- *Finite.* A graph is finite if A and V are not infinitely large sets. All of the graphs in Figure 1.9 are finite.
- *Planar.* A graph G is planar if and only if it can be drawn so that no edge defined in A crosses over another edge in A. In other words, G can be depicted graphically in two dimensional space.
- *Subdividable (into subgraphs).* A valid graph $G'=(V', A', \phi')$, is a subgraph of a graph G if $V' \subseteq V, A' \subseteq A$, and $\phi' \subseteq \phi$.

Let H be the set of $4(n+1)$ corner vertices for a frame W (where n is the number of interior frame openings). To locate the corners of H in $\Re^2$, let each corner be individually specified by the ordered pair $K_{op}=(x_{op}, y_{op})$, where o and p are index numbers. The index o designates the border that the corner belongs to, starting with the outside border (which is

labeled o=0). The index p is a base four variable that numbers the corners clockwise around the border. The corner with the minimum Euclidean distance to the origin of coordinate axis for the frame (the upper-left hand corner of the outside border) is designated p=0. The length of a side D_{oij} of an opening border can be represented by the Euclidean distance between the two corners in the ordered pair (K_{oi}, K_{oj}), where i=(p)MOD(4) and j=(p+1)MOD(4). Note that based on this definition, D_{oij}= D_{oji}.

Using this notation, a graph G_W=(V_W, A_W, ϕ_W) represents a frame W where

$$V_W=\{K_{op}\}; \quad \forall o, \text{ and } p\in\{0, 1, 2, 3\} \tag{1.3}$$
$$A_W=\{(K_{oi}, K_{oj})\}; \quad \forall o, \text{ and } i,\ j\in\{0, 1, 2, 3\} \tag{1.4}$$
$$\phi_W=\{D_{oij}\}; \quad \forall o, \text{ and } i,\ j\in\{0, 1, 2, 3\} \tag{1.5}$$

Figure 1.10 depicts an example of a frame represented as a graph. Note that the graphs of the interior borders are surrounded by the graph of the outside border, and the graphs of all openings o are closed rectangles.

Since a frame can be represented as a graph, frames inherit some of the properties and characteristics of graphs. First of all, note that if the graph of the entire frame is rotated at any arbitrary angle, the graph is the same as the original. That is, all rotations of the graph are isomorphic to the original. This implies that a frame has the same representation at any rotational alignment. Second, since the graph is a finite planar graph, the frame is as well. This is because the G_W contains only the 4(n+1) vertices, and two edges from each one. Finally, each frame opening, including the outside border, is a subgraph of the graph for the entire frame.

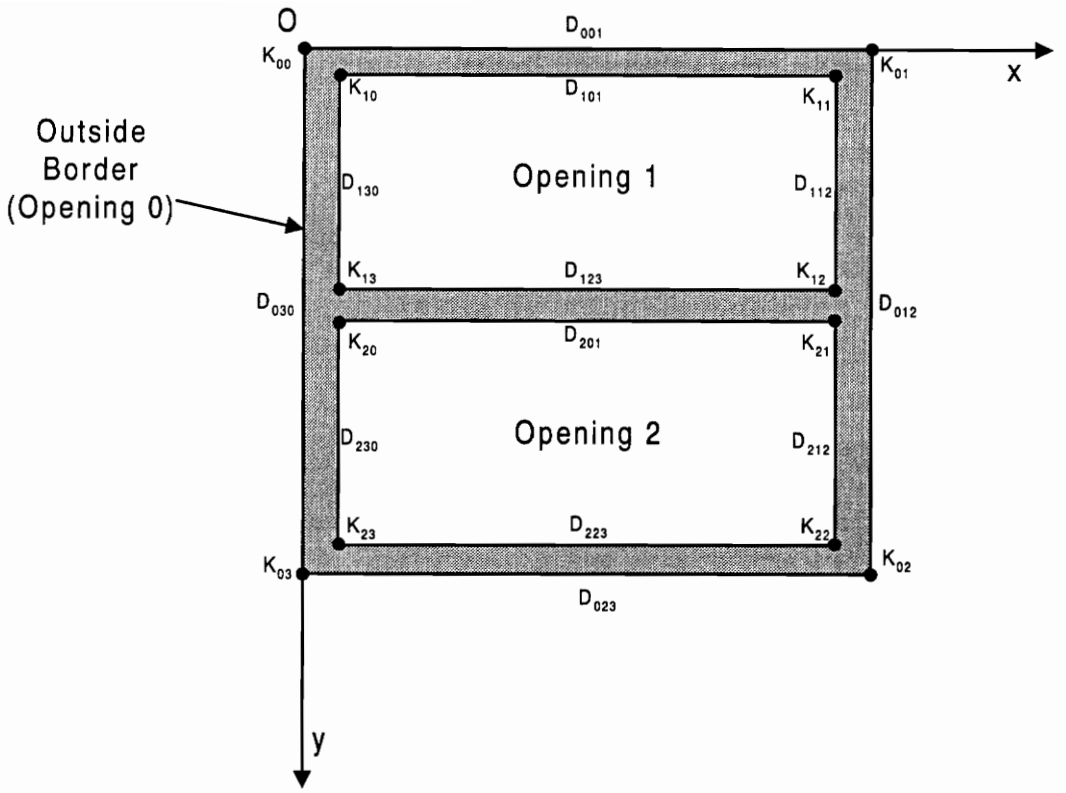


Figure 1.10: A two-opening frame represented as a graph. The outside border is designated Opening 0 for the purposes of notation. The dimension D_{oij} of each side is the incidence function assigned to the ordered pair in A that defines the side. See text for further details.

The dimension matrix λ (Equation 1.2), in reality, is a simplified form of G_W based on the geometry of the frame. Since each opening is rectangular, it can be represented by two side lengths and a vertex. The two sides specify the area of the opening, and the vertex places it in $\Re^2$. In this case, since opposite sides are equal,

$$D_{o01}=D_{o23}; \quad \forall o \tag{1.6}$$
$$D_{o30}=D_{o12}; \quad \forall o \tag{1.7}$$

If S_o represents opening o, selecting K_{o0} for the single vertex permits the opening to be represented with the ordered triple,

$$S_o=\{K_{o0}, D_{o01}, D_{o30}\} \tag{1.8}$$

Since an opening is a subgraph of the frame, where the union of all openings (and outside border) compose the frame, a frame W can be represented as the set of ordered triples

$$W=\{(K_{o0}, D_{o01}, D_{o30})\}; \qquad \forall o \tag{1.9}$$

By representing each opening as a row in a matrix and listing each in order of K_{o0}'s increasing distance from the origin of the coordinate system, the following form can be obtained:

$$W=\begin{bmatrix} K_{00} & D_{001} & D_{030} \\ K_{10} & D_{101} & D_{130} \\ K_{20} & D_{201} & D_{230} \\ \vdots & \vdots & \vdots \\ K_{n0} & D_{n01} & D_{n30} \end{bmatrix}. \tag{1.10}$$

This is equivalent to the λ-matrix in Equation 1.2.

1.6 Design Guidelines

Pugh [PUG83] discusses the essential requirements for success in industrial machine vision, and they coincide with the factory's goals:

1. Fundamental simplicity
2. Low cost
3. Reliable operation
4. Fast image processing
5. Ease of scene illumination

Fundamental simplicity means that the system is only as complex as necessary to perform the task at hand. This simplicity nurtures easier future expansion of the system, and aids in debugging. By keeping the system as simple and generic as possible, new frames can be added to the system with a minimum of work. Furthermore, this simplicity can aid the factory in meeting OSHA regulations that require a secure, safe work environment for employees when implemented. It is more difficult to seal components in enclosures if they require many complex adjustments often.

In addition, the factory desires the system to be as cost efficient as possible. The primary consideration is that the system should interfere minimally with the regular operation of the plant. As a consequence, the system must be compatible with their existing floor layout, which is conveyor belt based. Also, the system should cost as little as possible, so a return on the investment can be seen quickly. The illumination system and host computer selection were performed with this as the foremost consideration.

Reliable operation means that the system is error-resistant. Although the factory specifies that the frames must be aligned with a guide bar so that the bar code labeling machine can affix the labels properly, a realistic assumption would be that the frames would not be aligned with a very high degree of accuracy. As a consequence, the recognition algorithm detailed in this thesis supports a high degree of rotational invariance. Furthermore, misclassifications are highly undesirable, since a significant number could defeat the purposes of recognition in the first place. It its far more desirable to assign an "UNKNOWN" label to the frame if its type cannot be readily determined, and have the entire system reject the frame at a point further down the line.

Fast image processing can be ensured in two ways, by careful design of the recognition algorithm, and by careful selection of hardware for execution. By performing time-consuming operations on as little data as possible, efficient recognition can be

achieved. Also, the system can be improved if designed to take advantage of more efficient computer architectures, such as higher speed buses and processors, as well as imaging processing functions directly built onto image acquisition boards.

Ease of scene illumination draws on all of these concepts. By using a simple illumination system, such as a the backlight used in the presented system, costs can be reduced and fewer complications can result due to improper lighting. This simplicity reduces the processing burden on the recognition algorithm by reducing the side effects of lighting, such as shadows, that must be removed by the algorithm before proceeding.

One additional requirement, not included in Pugh's requirements but specified by the factory, is expandability. As new frames are designed and incorporated into the product line, they will have to be added into the system. The addition of a frame should be possible with very little effort. One way to do this is to develop a central database for frame data, using an efficient search routine to locate matches.

This thesis describes an automated method of distinguishing rectangular kitchen cabinet frames in a production environment. The emphasis has been to develop a reliable system that provides a high level of accuracy and stability. Since the frames are all made of the same material, it is natural to distinguish their geometries and dimensions visually. When the information about a frame is obtained, it is compared against the entire line of frames manufactured, and a best match is reported.

Bari presents an experimental system that also recognizes frames [BAR94]. The work presented here is different in that it uses alternate hardware and an entirely new recognition algorithm. The system discussed here is also approximately three to ten times faster in recognizing the frames. Also, the system discussed here has explicit low level

support for frame rotations of extreme angles, as well as generally more accurate corner detection.

1.7 Contributions of this Research

This thesis describes a system that achieves all of these requirements. The major contributions of this research are shown below.

- This thesis presents a representation format for flat rectangular components with rectangular openings.
- An automatic system for acquiring an image of a silhouette of a frame is presented based on back lighting and a line scan camera. The system implements new hardware that has several advantages in this type of application.
- Several features are examined for their suitability in cabinet frame recognition.
- Several prototype algorithms are discussed which can be used to extract these features from an image of a frame.
- A fast modular algorithm for recognizing cabinet frames is then presented based on the structure of a frame.
- A novel corner detection method is described, involving extrapolation of a corner location from a region of high corner likelihood on a curve.
- A novel database search technique is presented, based on a multilevel mean squared error heuristic, which supports frames scanned at arbitrary orientations.
- Successful test result data of the complete frame recognition system (30 feet/minute conveyor belt speed) on a set of 39 frame styles scanned at various orientations is discussed.
- The results of the successful factory-based test (50 feet/minute conveyor belt speed) are also detailed, which include over 6,600 frame samples.

- Time benchmarks for the complete system are presented, providing an estimate of system throughput.
- Special support for the AGxx series of atypical frame styles is detailed, serving to generalize the frame recognition algorithm to polygons with extreme vertex angles.

1.8 Organization

This chapter has discussed American Woodmark, their product line, and their motivation to automate the frame recognition process. Chapter 2 describes the acquisition subsystem, and the provisions necessary in the software for the hardware. Chapter 3 details several feature-based algorithms for frame recognition, including prior research in this area. Additionally, Chapter 3 discusses the performance, advantages, and shortcomings of all of these algorithms, and suggests a compromise based on the structure of a frame. Chapter 4 details this compromise, the final recognition algorithm implemented in the frame recognition system. Finally, Chapter 5 discusses the performance of the system in both laboratory and factory environments.

In addition, this thesis contains three appendices. Appendix A contains the numerical results of the laboratory test. Appendix B contains a discussion of implemented support for the AGxx frame style series (Section 1.4). Finally, Appendix C contains assorted documents necessary for the factory to set-up, calibrate, and use the system.

2. The Acquisition Subsystem

2.1 Overview

The frame recognition system consists of two separate subsystems: acquisition and recognition. The acquisition subsystem's primary function is to capture a video image of a frame moving on a conveyor belt. After an image is acquired, the recognition subsystem then identifies the frame. By distinguishing the two subsystems as separate entities that share information, they can be developed and optimized independently. This chapter details the development and implementation of the acquisition subsystem.

The acquiring of a frame's image, is not necessarily straightforward. Since they are always in motion on the conveyor belt (Figure 2.1), the acquisition subsystem must have additional provisions for sensing the location of the frame and for ensuring proper timing of the whole system at any conveyor belt speed. Furthermore, acquisition must be controlled primarily through hardware. Careful considerations must be made to keep the system cost-efficient while keeping the necessary functionality. The acquisition system is based primarily on work by Bari [BAR94]. Further details are given where modifications to his work were made.

The acquisition subsystem, in turn, can further be subdivided into six areas: Sensor Control, Illumination, Camera, Video Digitizer, Host Computer, and Substructure. Each of these areas is described below.

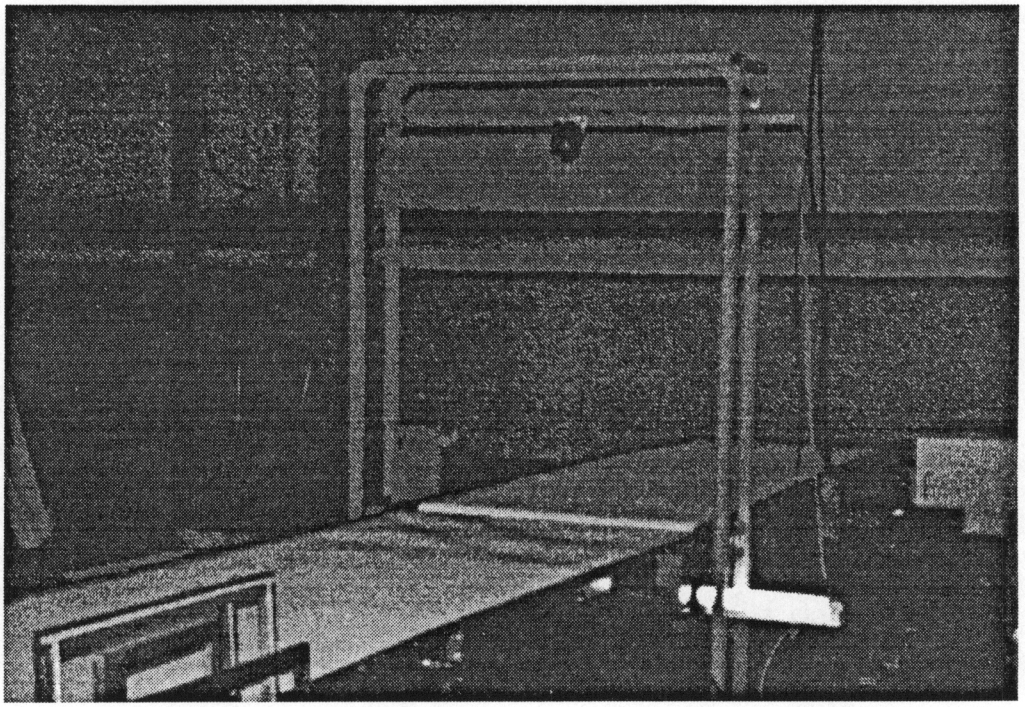


Figure 2.1: The complete acquisition subsystem. See Figure 2.9 for schematics.

2.2 Sensor Control

Since the conveyor belt carrying the frames is constantly in motion, the system must have information about the frame's position on the belt. The acquisition subsystem uses an infra-red emitter and detector pair placed on the belt just before the site of image acquisition. As a frame approaches this site, the infrared beam is interrupted and the sensors notify the rest of the subsystem to begin acquisition. Since the sensor pair is just before the acquisition site, the time the frame takes to cover the distance between the sensors and the site gives the rest of the subsystem ample time to initialize. After the frame clears the sensor pair, however, image acquisition cannot immediately halt--it must wait for the frame to clear the acquisition site. While the sensors directly control the start of acquisition, the system must delay after the sensors report that the frame has cleared the beam.

The sensor's output signals are compatible with the RS-232C standard [INT69] (±12V), and as a consequence, can be directly interfaced to the host computer's serial port (shown in Figure 2.2). The sensor status is polled through the use of a interrupt service routine, outlined in Figure 2.3. Essentially, the service routine polls the sensors at every tick of the computer's system clock (typically 18.2 times/second). As a consequence, the sensors have a finite resolution, and if a frame crosses the sensors in between queries, the system will only be able to respond at the next query. To compensate for this, the system clock's speed was increased by approximately three times. The higher rate of query provides for better sensor response.

Furthermore, the sensors must be mounted slightly above the conveyor belt. This is to prevent sensor noise from the irregular surface of the conveyor belt and the mounting slots on the underside of the frame. Using a narrow beam filter on the sensors also aids in preventing misstarts of acquisition.

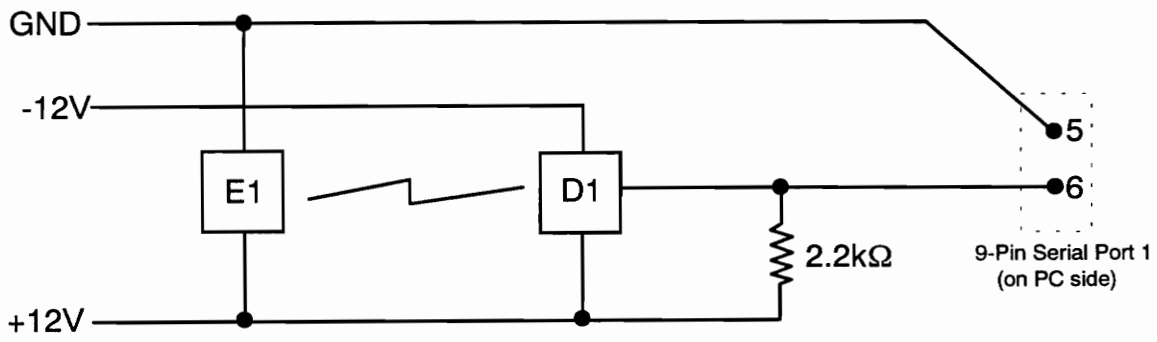


Figure 2.2: Infrared sensor circuit layout for frame recognition system. The resistor provides current limiting. (E1,D1) are an infrared sensor pair (Honeywell CP18RDND2 and CP18ECX2). All resistors have 5% tolerance, 1/4 Watt. Use a metal housing for the 9-pin serial port.

```
Initialize_ISR:
        Initialize Serial Communication
        Reprogram Timer for Higher Rate
        Install Sensor_ISR
        Cascade Sensor_ISR with original Timer 08h ISR
end Initialize_ISR

Sensor_ISR:
        Read Serial Port 1
        Check CTS status from serial status register
        Alter global variable picture_flag, based on CTS status
        Reset Serial Port 1 Status
        Chain to former Timer ISR
end Sensor_ISR
```

Figure 2.3: Pseudocode of sensor interrupt service routine. The sensor pair is wired directly to CTS (pin 6) on Serial Port 1. Sensor control routines are based off of the Host Computer system clock interrupt at 08h.

The sensor control system relays information to the rest of the acquisition subsystem when a frame is in the acquisition site. The next section discusses the subsystem's source of illumination, which is essential for high quality images.

2.3 Illumination

As in Bari's experimental system, the acquisition subsystem uses back lighting techniques to illuminate the frame for acquisition. Basically, it involves placing the frame between the light and the camera, as in Figure 2.4.

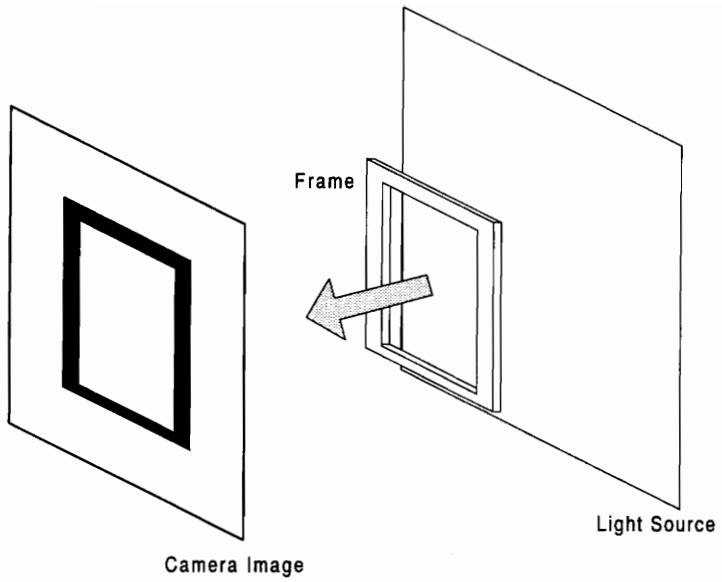


Figure 2.4: Back lighting technique. The gray arrow shows the direction of emitted light.

As mentioned in the previous chapter, the frame's overall geometry--the number, size, and placement of a frame's openings--allows one to distinguish one frame from another. All of this information is contained in the frame's silhouette. In addition, using back lighting in this sort of application provides several advantages over other illumination techniques. These advantages are listed here:

1. Produces high contrast images of the perimeter of opaque objects [GAL90]. This is the primary reason for using a back light in the frame acquisition subsystem. Since all frames are opaque, and all of the parameters mentioned in Chapter 1 can be obtained from the perimeters of the openings and their relative positions, it provides information for recognition in a readily usable form. Also, the high contrast minimizes the imaging processing task.
2. Resistance to light intensity changes. As light sources age, their luminescent outputs change, typically decreasing unless special provisions are made. Since the back light

provides such a high contrast, as long as the light intensity is sufficient for the camera's optics, the frame openings can be easily emphasized. The high contrast reduces the sensitivity of the system to illumination source variations [GAL90].

3. Resistance to exterior ambient light. By aiming the camera directly at the light source, and making the back light encompass the camera's entire field of vision, the effects of stray light are reduced. Any glare that does occur will be minimal because of the high contrast.

As Galbiati notes, the image obtained is essentially binary in that a high-contrast silhouette of the frame is obtained [GAL90]. However, the edges of the frame will actually be intermediate gray-level values. This variation of gray level occurs for two reasons: First, subpixel registration can result in extraneous gray levels. That is, the frame edge occurs between two pixels, and the gray levels represent the fractional distance. Second, the clear varnish on the frames can produce highlights. When these gray levels are thresholded, as a consequence, the theoretical "true" edge of the frame might deviate slightly. This deviation is worst at the interior corners of the frames, where the corners appear rounded off after processing.

Finally, the intensity of the back light must not overwhelm the camera. While modern cameras are not likely to be damaged by extremely bright light, the excess intensity will cause an optical effect called *blooming*. Blooming is when a bright region of an image "bleeds" into a dark section because of the properties of the camera's imaging array. Blooming can be easily compensated for by reducing the light level intensity until it no longer occurs, or closing the camera's aperture slightly. The acquisition subsystem is configured to minimize these potential problems of back lighting to ensure that a relatively clear, crisp image is passed on to the recognition subsystem. Furthermore, the recognition subsystem is designed to compensate for the few errors that do appear in the frame image.

. Beyond these few difficulties, the only other complication is that the back light must have a relatively even intensity over its area. The size of the largest frame, UT3696, which is 36" by 92", cannot be evenly illuminated by any cost-effective method. Since the frame is constantly moving on the belt, careful selection of a camera can simplify lighting requirements greatly. A line scan camera can reduce this large area to a narrow strip, which can be easily be lit by a fluorescent tube. If a common household fluorescent light is used, however, line scan cameras operating at their default sample rates would perceive the 60 Hz "flicker" of the light. Higher frequency florescence are available, but extremely expensive. If special provisions are made in camera timing, not only is this illumination source realizable, but cost-effective. By altering the camera's sample rate, the acquisition subsystem uses a common 8 foot 60 Hz fluorescent light as its illumination source.

2.4 Camera

The fundamental difference between line-scan cameras and their area-scan counterparts is the differences in the geometries of their CCD imaging arrays. Whereas area-scan cameras possess a rectangular array, line scan cameras have an array that is very wide, but only one row high. As a consequence, a line scan camera perceives only a narrow one-dimensional line. Figure 2.5 graphically compares the two.

Through the use of a line scan camera, the lines of projection are constrained to lie on a plane between the camera and the back light. This plane is perpendicular to the motion of the conveyor belt, but parallel and aligned to the fluorescent tube.

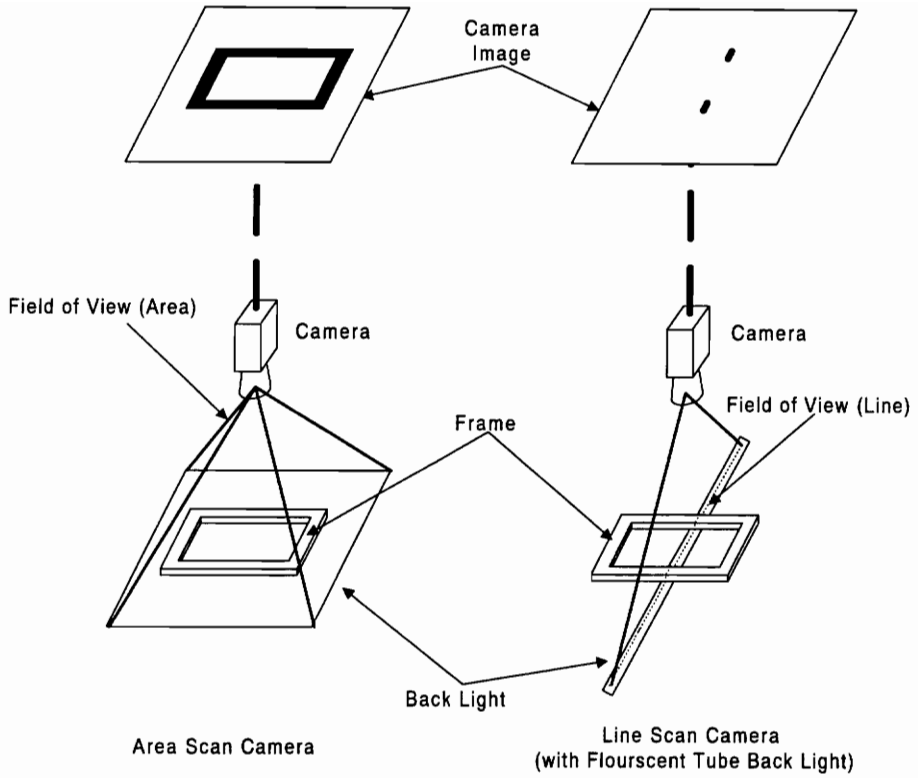


Figure 2.5: Comparison of area scan vs. line scan acquisition.

Since a line scan camera acquires only one line of an image at a time, there is great flexibility in how images are obtained. In the case of the frame acquisition subsystem, a line is acquired approximately every 18 ms, and each line is placed below the previous one in a buffer. Since the frame is moving through the acquisition plane on the conveyor belt, a complete silhouette of the frame is constructed in the buffer. An image generated in this way is called a "waterfall" image. The start and end of an image frame is communicated to the subsystem through the sensors.

The line timing, also known as the line clock, is a parameter that is easily adjustable in a line scan camera. Bari computed that a camera with a line width of 1024 pixels would sufficiently cover all frame widths. As a consequence, the DALSA CL-C4-

1024-M camera was selected for this subsystem. By placing a clock signal, CCLK, of the desired frequency on the EXSYNC control pin of the DALSA camera, CCLK will act as the line clock. This clock can come from a custom built circuit [BAR94], a timing generation board, or the video digitizer board itself.

By careful adjustment of the line timing, the camera image can be synchronized to the conveyor belt speed. With a conveyor belt speed of 30 feet/minute, the CCLK should have a frequency of 55.9 Hz [BAR94]. The closest frequency obtainable from the timing generation board used by the system was 55.56 Hz, which was adequate for the system. The mismatch between the ideal and the actual frequency used causes a slight stretching of the image in the direction perpendicular to the acquisition plane.

This low sampling frequency is the reason that an inexpensive household florescent tube can be used as the system's back light. A line scan camera operates on a principle similar to a film picture camera. The camera is instructed to take a picture, opening the shutter for the film to be exposed for a short period of time, then closes the shutter. In the case of the line scan camera, the amount of time the CCD imaging array is exposed is called the integration time, controlled by the time between CCLK pulses. Since the line frequency is so low, the interval between rising edges of CCLK is very high, resulting in a long integration time. In the case of the frame acquisition subsystem, the integration time, or exposure time, is set to the maximum allowable by the video digitizer that does not cause image blur, approximately 18 ms. This long exposure time averages out the 60 Hz "flicker" of the fluorescent, leaving a relatively even background. Thus, the requirement for an even intensity light is met by using a long fluorescent bulb that lacks "hot-spots" in the horizontal direction, and by maximizing the exposure time in the vertical over time. "The line scan camera is the choice for part or inventory inspection stations on an assembly line" [DAL94].

The camera is controlled by the video digitizer, which provides assorted control signals, as well as truly being the part of the subsystem that directly controls acquisition. Furthermore, the video digitizer also contains RAM for the buffer of the line scan cameras, and is where the "waterfall" image is built. Since it is the interface between the camera and the host computer, speed and flexibility is critical. The next section discusses the video digitizer selected for the acquisition subsystem.

2.5 Video Digitizer

The video digitizer is the interface between the camera and host computer. Not only does it provide control directives to the camera, but it also buffers the image data received from the camera in on-board RAM. Typically, the video digitizer is a card that slides into the computer bus, and connects via cables to the camera.

For the frame acquisition subsystem, the video digitizer board must be capable of supporting a line scan camera. This means that the board must provide the signals for CCLK and exposure time, as well as the ability to generate "waterfall" images, as detailed in the previous section. In addition, the board must have a large, fast image buffer, at least one megabyte in size.

In Bari's research, he selected the Advanced Frame Grabber (AFG) by Imaging Technologies, Inc. [IMA92]. With some minor modifications this board could support line scan cameras, as long as CCLK was generated externally. After his implementation of the system was developed, however, several problems became apparent. First of all, software support was limited--the board only used libraries for compilers that were at least four years old. As a consequence, the features of the newer compilers could not be used to their fullest. Second, although the board claimed to support line scan cameras, Bari experienced many problems, including random "drop-outs" of lines and noise. This

is most likely attributable to an inconsistent exposure time. Finally, there were limitations in acquisition. An image has a fixed length on the AFG, whether or not a line scan camera is used. Ideally, when the sensor control system reports that a frame has cleared the sensor beam, there should be a brief delay, and acquisition should immediately stop. By the design of the AFG, it will not release the host computer to perform another acquisition until 1024 lines are acquired for an image pseudo-frame, regardless of the actual frame length in lines. This means that a 15" frame would take the same length of time to acquire as a 92" frame. If only the longest frames were being regularly produced this would be acceptable, but in the factory, frames of various sizes will be passing through the system at random. Because of these limitations, a new video digitizer had to be selected for use in the acquisition subsystem.

The video digitizer selected for the system was the Data Raptor-VL by Bitflow, Inc. [BIT94]. This board offered numerous features that not only solved the problems encountered with the AFG, but added several features. These features are as follows:

- VESA-Local Bus Interface. This fast interface allows data to be transferred between the Raptor and the host computer at a much higher rate than the AFG (approximately three times faster).
- 4MB Dual Port RAM. The Raptor was designed to be almost completely autonomous from the host. The buffer on the video digitizer can be read by the computer while the camera is still writing to it. This increases throughput.
- True support for line scan cameras. The Data-Raptor has hardware support for line scan cameras, and the board is able to generate CCLK internally, or accept it from an external source. Also, the board provides complete control over exposure time.
- Asynchronous acquisition. Typical video digitizers place their host computer into a wait cycle while they are acquiring images. The Raptor can free the computer after a slight synchronization delay.

- A high degree of acquisition control. The Raptor is programmed with lower-level image acquisition than the AFG. Where the AFG comes with many prepackaged libraries for image processing, the Raptor does not. The Raptor, however, allows very simple register-level control. For instance, by altering the registers, it is possible to develop a command "on trigger, start and end acquisition", like a stopwatch. This stopwatch acquisition is desired for the frame system, where the trigger would be the sensor signal.
- True frame reset. The Raptor is capable of changing the source of CCLK on the fly. Because of this, CCLK can be increased to 40MHz and video can be shut off, finishing the pseudo-frame practically instantly. Alternately, the image buffer can be simply cleared and the buffer memory pointer can be set back to zero for a very fast reset. The latter is used by the frame acquisition subsystem to forcibly end acquisition.

The functional block diagram of the Data Raptor-VL is shown in Figure 2.6. In addition, the Raptor contains look-up table support, like the AFG. This is a hardware table that can perform a real-time thresholding operation, which is required by the recognition subsystem. Since the timing generation board can produce a great many more frequencies than the Raptor can internally, the timing board was connected directly into the Data Raptor. The schematic diagram for this interconnection is shown below in Figure 2.7. Finally, Figure 2.8 outlines the pseudocode for the acquisition subsystem on the Raptor.

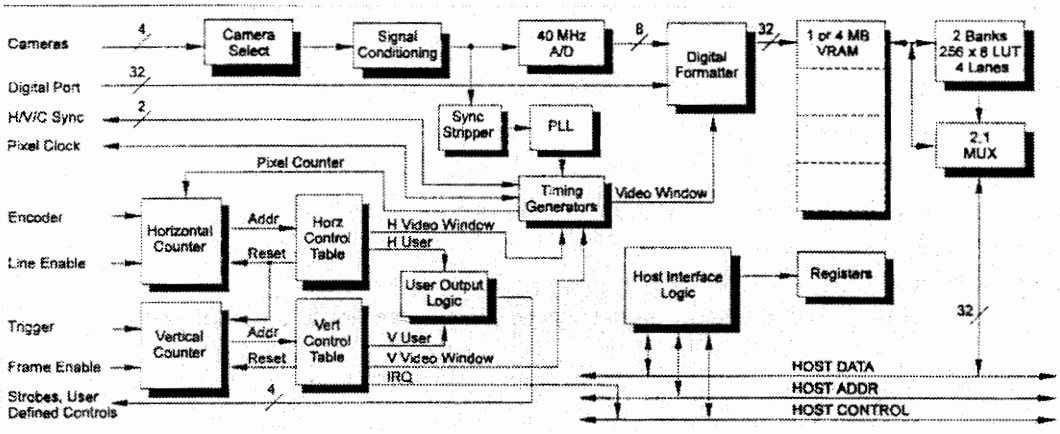


Figure 2.6: Data Raptor-VL functional block diagram [BIT94].

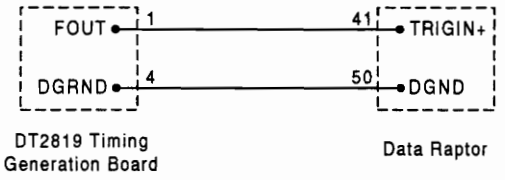


Figure 2.7: Interconnect for Raptor to timing generation board.

All of the information gathered by the camera from the rest of the acquisition subsystem is collected into the video digitizer. While it is possible to operate directly on the Data Raptor's image buffer, a far simpler and more efficient approach is to transfer it into program memory. This is because the command set to operate on the image buffer is very limited, and the program memory is significantly faster. The host computer acts as the home of the recognition subsystem, the interface to users, and the base of the entire system.

```
(on sensor beam break):
    ACQUIRE:
        configure asynchronous acquisition
        soft trigger acquisition
        synchronize host PC to Raptor
        until Raptor buffer full or sensor beam closed
            while new lines are in Raptor buffer
                copy lines over to host buffer
            end while
        end until
    end ACQUIRE

    GET_SLACK:
        if current frame length < Raptor Buffer Length-(44)
            for count=1 to 11,
                while new lines are in Raptor buffer
                    if count=11, break
                    copy lines over to host buffer
                end while
                break if Raptor buffer full
                if sensor beam broken, break and report error
                    (frames too close together, insufficient time for processing)
            end for
        end if
    end GET_SLACK

    RESET_GRABBER:
        configure Raptor for vertical line reset on trigger
        soft trigger Raptor
        disable vertical line reset on trigger
        clear Raptor VRAM
    end RESET_GRABBER
```

Figure 2.8: Pseudocode for frame acquisition system. ACQUIRE compiles an image from the start of sensor crossing to when the frame clears the sensors. GET_SLACK then acquires additional lines to complete the image of the frame as it clears the acquisition plane. RESET_GRABBER then forcibly ends acquisition. Typically the Raptor acquires four lines every time it loops through the "while" structures in ACQUIRE and GET_SLACK.

2.6 Host Computer

Since the factory's primary directives included keeping costs at a minimum and the Data Raptor can only interface with a IBM-compatible Personal Computer (PC), a PC is the base platform for the frame recognition system. This allows the factory to take advantage of rapidly lowering personal computer prices, and their affordable maintenance. Another advantage of using PCs is that they are familiar to end users. No appreciable learning curve is necessary.

The recognition subsystem was written in "C" using DOS. However, PCs, specifically ones running DOS, offer several limitations that must taken into account. One of the most glaring of these is memory limitations. Without additional provisions, no program can exceed 640K. In most applications, this limitation is easily avoidable, but efficient machine vision is very memory intensive. Given a typical scanned image of 1024 pixels by 1024 pixels with 256 shades of gray (8 bits), the memory that must be allotted is approximately one megabyte. A PC could not even hold a single image of this size in memory. However, the system discussed in this thesis switches the computer into Protected Mode, where the program memory space is equal to amount of available RAM, similar to a UNIX machine. Furthermore, the Data Raptor-VL video digitizer port addresses are very high in the PC's memory space. Consequently, the board requires Protected Mode to be accessed at all.

An IBM-compatible computer is an excellent base for the overall acquisition and recognition subsystems. It provides an excellent balance between cost, usability, support, and using Protected Mode, functionality. The next section details how the various system components are mounted and connected together to provide a complete image for the recognition subsystem to process.

2.7 Substructure

Figure 2.9 details the final imaging geometry. Note that two conveyor belts, moving at 30 feet/minute, are placed so that there is a slight gap between them. This gap contains the acquisition line, which is back lit from the fluorescent tube.

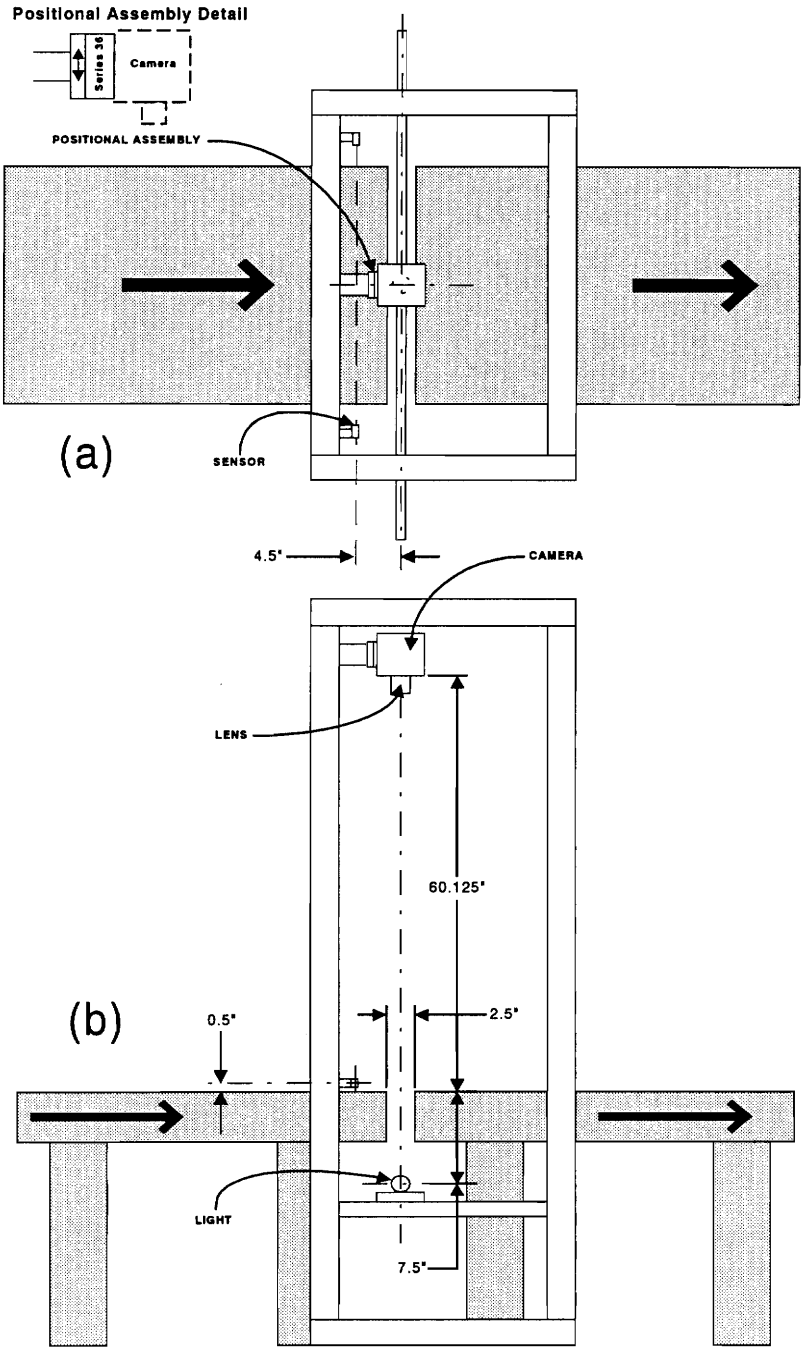


Figure 2.9: Acquisition subsystem blueprint with dimensions. (a) top view. (b) side view. The positional assembly is a tilt/pan mount with 1-D translation, permitting fine adjustments to the camera position relative to the back light.

Figure 2.10 details the cable interconnects between the host computer, sensor system, video digitizer, and camera.

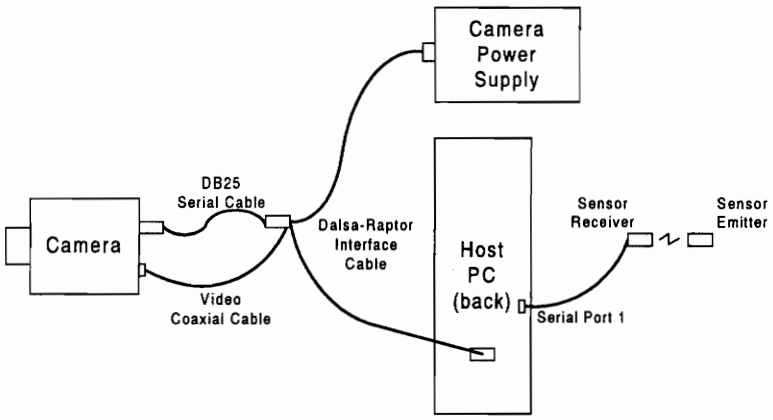


Figure 2.10: Interconnect schematic for the acquisition subsystem.

2.8 Summary

This chapter has discussed the frame acquisition subsystem, one of the two parts of the frame recognition system. It converts a moving frame on a conveyor belt into an image that is capable of being processed on the host computer by the recognition subsystem. By using carefully selected components, acquisition of high-quality images is cost-efficient. Fundamentally, the acquisition system is simple, using very basic lighting, altering the timing of the camera to compensate for the 60Hz flicker. Figures 2.11 and 2.12 show images of samples frames obtained by the acquisition subsystem.

The following chapters discuss the recognition subsystem, which was implemented in Protected Mode Watcom "C" v10a on a IBM-compatible personal computer. Several potential algorithms are reviewed and discussed before the development of the final recognition subsystem.

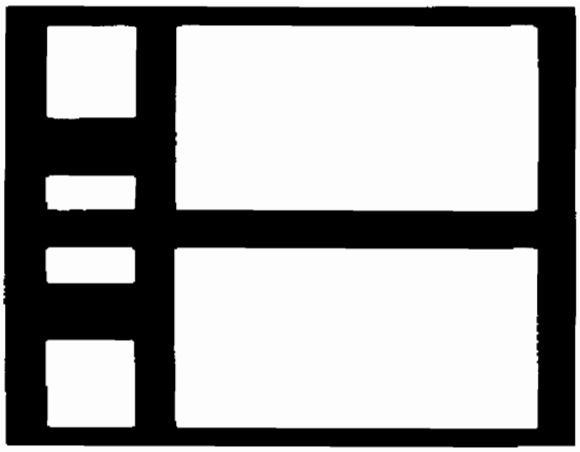


Figure 2.11: Image of VAN36 frame style from acquisition subsystem. Frame size is 27.125" by 36". Frame is aligned against an alignment bar. Note the edge noise effects.

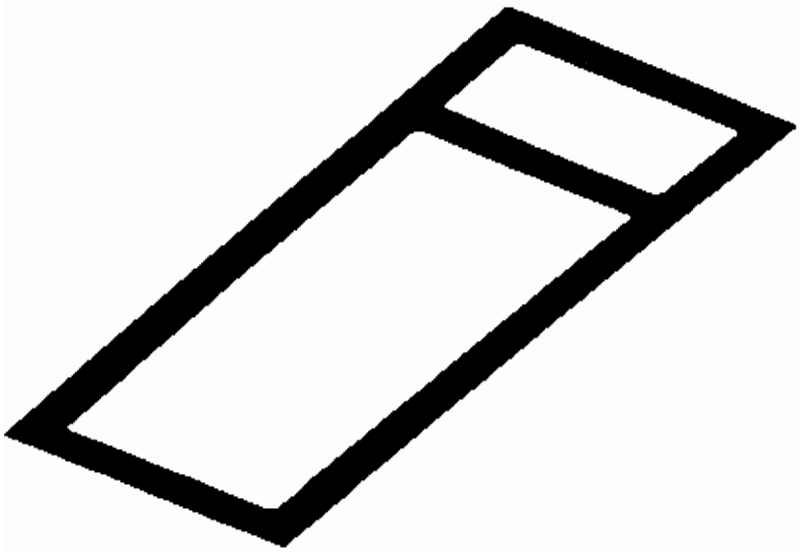


Figure 2.12: Image of B12 frame style from acquisition subsystem. Frame size is 30.5" by 12". Frame is unaligned at approximately a 45 degree angle. The distortion in the shape is due to the differences between the sampling rate of the camera and the speed of the conveyor belt. Also note that the corners are slightly rounded off--this is accentuated because of the limitations in sample rate.

3. Feature Extraction

3.1 Overview

Through the use of the acquisition subsystem discussed in the previous chapter, cabinet frames can be imaged and analyzed. By applying machine vision techniques, the recognition subsystem attempts to assign frame style labels to those images.

Conceptually, the high-level recognition algorithm is very simple. The procedure is as follows:

1. Extract a set of significant features F from the scanned image of the frame.
2. Compare F to each representative frame in $\mathbf{F}=\{F^{(k)} \mid k=1, 2, 3, ..., N\}$, where N is the number of frames known to the system, $F^{(k)}$ is the ideal representation of the kth frame.
3. Assign style k to F, where $F^{(k)}$ is the closest match to F.

Several issues must be addressed for this algorithm to produce reliable results. First is the proper selection of the features that compose F and the members of $\mathbf{F}$. Second, the F-features must be distinct enough to be obtained even from a slightly noisy and corrupted image. Although the image acquisition subsystem is designed to produce clear images, allowances must be made for slight errors due to edge noise (Section 2.3) and because the frame can shift slightly during acquisition. Third, the features extracted from the scanned image must be reasonably accurate.

There are many valid selections for the features to include in F. This chapter discusses several choices. The first is image moment invariants, and the remainder are

based directly on the λ-matrix (Equation 1.2). These methods are Bari's blob analysis, rectangle fitting, and boundary-based corner detection. A final recognition algorithm is then proposed which combines the benefits of all of these in an easily realizable structure.

3.2 Image Moment Invariants

Image invariants are properties of an image that are independent to given image transformations [REI93, DAE93]. Typical transformations are rotation, translation, and scale, shown in Figure 3.1. Several invariant operations exist for these transformations, based on moments. The regular moment m_{pq} of an image I of size $X \times Y$ is defined as:

$$m_{pq} = \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} x^p y^q I(x,y), \text{ where } p,q = 0,\ 1,\ 2,\ \ldots \tag{3.1}$$

The central moment μ_{pq} can then be represented as

$$\mu_{pq} = \sum_{x=0}^{X} \sum_{y=0}^{Y} (x-\bar{x})^p (y-\bar{y})^q I(x,y) \tag{3.2}$$

where $\bar{x} = m_{10}/m_{00}$ and $\bar{y} = m_{01}/m_{00}$.

Since $(\bar{x}, \bar{y})$ is the centroid of the image, central moments are independent of image translations, as the image centroid is shifted to the origin [REI93].

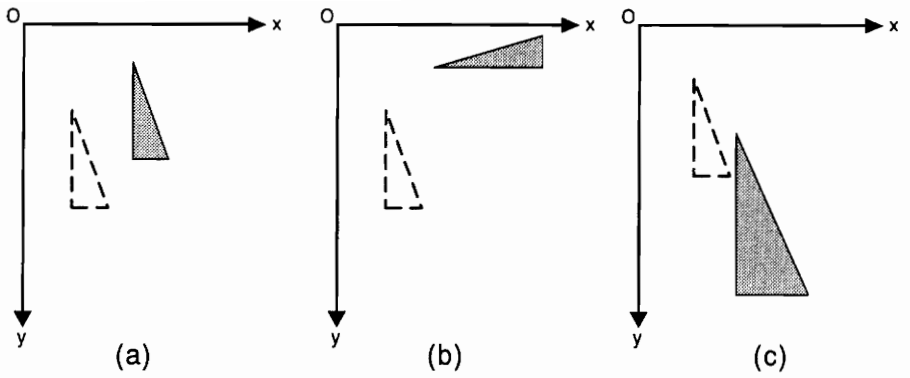


Figure 3.1: Three typical transformations about the origin O. The dashed triangle is the original position, the shaded triangle is the new position. (a) shows translation, (b) shows rotation, and (c) shows scaling.

Consequently, Hu's first two translation and rotation invariants can be computed as [HU62]

$$\Phi_1=\mu_{20}+\mu_{02} \tag{3.3}$$
$$\Phi_2=(\mu_{20}+\mu_{02})^2+4\mu_{11}^2 \tag{3.4}$$

Finally, central moments in (3.3) and (3.4) can be replaced with the scale invariance

$$\eta_{pq}=\frac{\mu_{pq}}{\mu_{00}^{\gamma}}\text{, where }\gamma=\frac{p+q}{2}+1\text{, }p+q=2,3,... \tag{3.5}$$

This substitution forms the following first and second order equations, which are translation, rotation, and scale invariant:

$$\Psi_1=\eta_{20}+\eta_{02} \tag{3.6}$$
$$\Psi_2=(\eta_{20}+\eta_{02})^2+4\eta_{11}^2 \tag{3.7}$$

The invariant values determined by (3.6) or (3.7) should not change regardless of the placement of the frame silhouette in the image, its rotation, or its absolute scale. Invariance to translation permits the frame to be placed at any lateral position on the belt, not in just some predetermined position. Scale invariance effectively makes recognition independent of the camera's distance to the belt (resistant to "zooming"). Finally, rotation invariance permits unaligned frames to be properly recognized.

By computing an invariant for all three transformations simultaneously, it is possible to reduce the image of a frame down into a single global feature value for F [HAR93a]. The recognition algorithm is then based on determining the invariants for all of $\mathbf{F}$, and searching it for the closest match to F.

Although image invariants sound like ideal features for the recognition problem, their use introduces several difficulties. One of these is noise susceptibility. Image noise, whether due to edge noise (Section 2.3) or otherwise, corrupts invariant values dramatically [HAR93a]. Moments become more susceptible to noise as their order and number of terms increases [ABU84]. However, better discrimination is obtained by using higher order moments. As a consequence, there is a limit to the discretionary power of moment-based invariants [REI93]. Another problem is the fact that invariants, especially those expressed in (3.6) and (3.7), are very abstract--they do not have any immediate physical counterpart. As a consequence, if the system misclassifies a frame using invariants, the problem cannot be easily visualized. Finally, generating $\mathbf{F}$ for the entire product line of frames is very time consuming. While image invariants can be used with good results for near-ideal images [DAE93, REI93], the factory recognition subsystem needs a more error-tolerant, intuitive method of recognition.

3.3 Corner-based Methods

As detailed in Section 1.3, the three features that distinguish all of the frames are number of openings, the size of the openings, and their displacement from the frame origin. All of these parameters can be quantified and combined into one feature matrix, λ_k for each frame style k.

In the terms of the algorithm in Section 3.1, let F equal λ^*, the λ-matrix of the scanned frame, and $\mathbf{F}$ then equals $\{\lambda\}$, the set of λ_k for all frame styles k. While the values in each λ_j can easily be obtained by physically measuring the frames or from manufacturing schematics, the values in λ^* must be obtained through some form of image processing. Upon further examination, λ^* can be generated if all of the corner vertices can be determined for each opening of the frame. Since corners are so visualizable, and there are also a wide variety of methods to detect and locate them, they are a good candidate for use in frame recognition [BAR94]. Several corner-based methods follow below.

3.3.1 Blob Analysis

As mentioned previously, Bari developed an experimental system for frame recognition where region properties are used to estimate the corners in the image [BAR94]. These corners are then used in turn to formulate the values in λ^* for the frame.

A region is a set of 8- or 4-connected pixels in the image, all having the same value. As a collective group, the pixels in the region can also be seen as an arbitrary "blob", which has properties of area, perimeter, and so on. For the purposes of his work, the blobs are the area of each frame opening, including the outside region. These blobs are individually numbered for reference.

Bari's algorithm bases corner registrations on the fact that the frames are all rectangular. A labeled blob in the image can be then inscribed in a rectangle, called its bounding box. Figure 3.2 shows the bounding boxes for a one opening frame. If the frames are aligned to the bounding box, the generation of corner points is trivial; they are the vertices of the bounding box. However, if the frame is misaligned, further computations must be performed. One approach Bari uses is a least-squares line fit to the a side of the outer region, using the points where the lines cross the bounding box as two of the corners. Symmetry and geometry are then used to determine the rest.

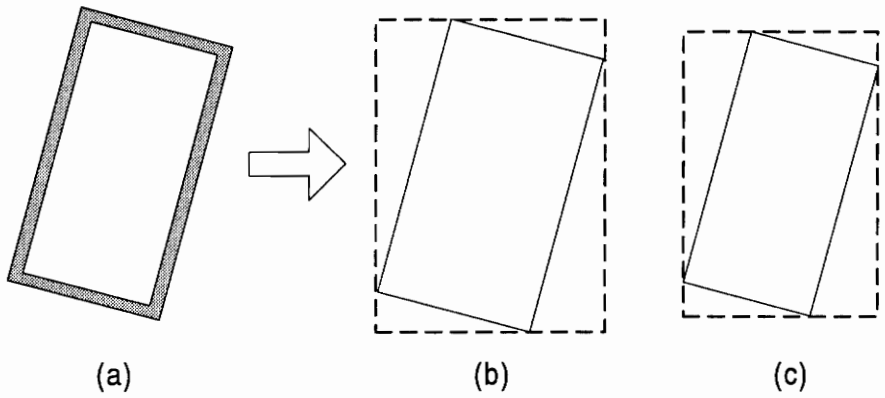


Figure 3.2: Bounding box decomposition of a one opening frame. (a) Original silhouette of the frame. (b) Solid bounding rectangle of the outer opening and its dashed bounding box. (c) Rectangle and the bounding box for the interior opening. Note that the frame is unaligned for bounding box clarity.

Although the blob analysis method generally locates the corners accurately when the frame is aligned, it does not perform as well for frames which are not. For example, when blob analysis is applied to the frame style W3615 when it is misaligned over fifteen degrees, 33% of the corners generated for the entire frame are significantly misplaced. There are several possible reasons for this. First of all, the blob analysis method implicitly assumes that the outlines of the frame openings and the outside border are perfect rectangles with right angle corners. As discussed in Section 2.3, corners appear

rounded off in the image, and edge noise appears when edge gray-levels are thresholded. Both of these can cause the fitted lines to deviate from their correct positions and consequently cause corners to be misplaced. When the frames are aligned, the noise and other effects are minimized, and the results are subsequently better.

Furthermore, the blob analysis method is undesirable because of its execution time. For instance, the system requires 27.4 seconds to register the corners in the frame style W3615--too long for factory operation. Another problem occurs when the frame is jostled or misfed through the acquisition plane. This algorithm has no provisions quantifying the quality of the frame image, and will continue to assume they are four-sided rectangular shapes, even if badly corrupted.

Although the region analysis method is conceptually clear and functions extremely well in aligned cases, it has insufficient support for misaligned frames and feeding errors. Furthermore, its execution time is prohibitive for plant operation.

3.3.2 Rectangle Fitting

Another approach to frame recognition is to fit a rectangle to each of the openings and the outside border outlines, and to use the vertices of this rectangle as the corners of the opening [PAR94]. Assuming that the openings are approximately rectangular (allowing for corner rounding and edge noise), the orientation of a particular opening can be depicted via its *principal axis*. The principal axis is defined as a line passing through the object's center of mass having a minimum total distance from all pixels belonging to the object [PAR94]. Once this axis is known, there is sufficient information to construct a rectangle that attempts to fit the shape relatively closely.

The algorithm for the rectangle fitting method based on the principal axis is shown in Figure 3.3 below.

for each opening/rectangle:

GET_PRINCIPAL_AXIS:

Determine the centroid, α, of the opening.

Let δ={all boundary pixels that lie above α}.

For each member of δ (call it ϕ),

Generate equation of line between ϕ and α.

Compute and store total perpendicular distance between line and all boundary points in rectangle.

end for

Line with minimum total perpendicular distance is the principal axis, ε.

end GET_PRINCIPAL_AXIS

GENERATE_FITRECTANGLE:

Let (x_1,y_1) and (x_2,y_2) be the intersections ε makes with the boundary.

Given the slope of ε, generate 2 lines, A and B, through (x_1,y_1) and (x_2,y_2), respectively.

Let ζ, the minor axis, be perpendicular to ε.

Let (x_3,y_3) and (x_4,y_4) be the intersections ζ with the boundary.

Given the slope of ζ, generate 2 lines, C and D, through (x_3,y_3) and (x_4,y_4), respectively.

The intersections of AB, BC, CD, and DA are the vertices of the rectangle.

end GENERATE_FITRECTANGLE

Figure 3.3: Algorithm for rectangle fitting method of corner generation. GET_PRINCIPAL_AXIS generates the equation for the principal axis. GENERATE_FITRECTANGLE constructs an idealized rectangle about the opening with orientation specified by the principal axis.

At first glance, this algorithm seems to be ideal. First and foremost, it models the openings at a very high level, relying only on general trends in shape to construct the rectangle. This is in contrast to the blob analysis method, which uses bounding boxes to aid in determining the corners, which are based on the openings' extremes. As a consequence, rectangle fitting is very resistant to edge noise, which is very localized. Second, since the fitting algorithm returns the vertices of the fitted rectangle, which are not even necessarily on the opening border, rounded off corners (Section 2.3) are typically interpolated back into reconstructed corners (Figure 3.4).

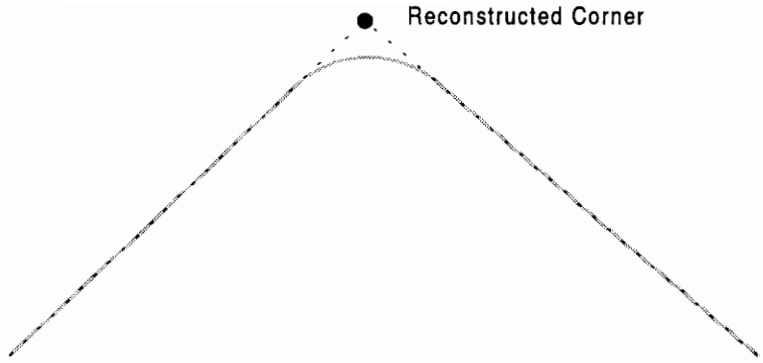


Figure 3.4: Rectangle fitting technique corner reconstruction. Since this method returns the vertices of a fitting rectangle whose sides are best fits to the sides of the opening border, a rounded corner can effectively be reconstructed. The solid gray line is the scanned opening border line, and the dashed lines are the sides of the fitted rectangle.

While this method appears at first more suitable than the blob analysis method, it has one disadvantage which makes it undesirable as well. This is the issue of the rectangularity of the scanned openings. Due to mismatches in the line clock and conveyor belt speed (see Section 2.4), the frame heights can be stretched or compressed due to the sample rate, as in Figure 2.12. Fine tuning of the sample rate is very difficult as there are frequency resolution limits in the timing controller, and the conveyor belt's speed cannot be controlled precisely. This makes the openings appear closer to parallelograms than rectangles. As a consequence, the rectangle fit cannot be used when the image is distorted.

3.3.3 Boundary-Based Corner Detectors

Boundary-based corner detectors define corners one of two ways: 1) as points where the object boundary makes discontinuous changes in direction, or 2) where the curvature of the boundary exceeds a certain specified threshold. The confidence level that a particular curve point is a corner is called that point's cornerity. Unlike the other

corner based methods discussed previously, boundary-based detectors work at a very low level in the image, examining the outline of the opening at a very high resolution. There are many boundary-based corner detectors available, and each follows different techniques to detect corners.

For instance, the Freeman-Davis method [FRE77] represents the opening's outline as a long list or chain, where each element contains the relative direction to the next pixel rather than the location of the current one. Chain coding is discussed further in Section 4.4.1. Let the chain be represented as $\{a_i\}^n$. It is scanned with a sliding window of size s. For each window, or node, its endpoints a_{j-s+1} and a_j, are connected with a line segment L_j^s. As this window passes over the chain, the angular differences between the successive positions of L_j^s are recorded and then smoothed. A corner is then characterized by three parts: two for which the smoothed angle lies within narrow limits, separated by a third region of exactly $s+1$ nodes with a total angular difference exceeding a critical value. Freeman and Davis use a heuristic to quantify this definition of a corner.

Rosenfield and Johnson propose another method [ROS73]. Given a point on the border $i=(x_i, y_i)$, define two k-length vectors projecting from i to both sides. Mathematically, these two vectors are represented as the *k-vectors*: $a_{ik}=(x_i-x_{i+k}, y_i-y_{i+k})$ and $b_{ik}=(x_i-x_{i-k}, y_i-y_{i-k})$. Corners are determined using the heuristic *k-cosine* of the angle between the two vectors, which is determined by:

$$c_{ik} = \frac{(a_{ik} \bullet b_{ik})}{|a_{ik}|\,|b_{ik}|} \tag{3.8}$$

Therefore, as c_{ik} approaches zero, the angle between a_{ik} and b_{ik} approach a right angle. Corners in this algorithm are defined by i's which have a minimum c_{ij} under a certain threshold in a predefined neighborhood.

The major advantage of these two algorithms is their flexibility. In both of these algorithms there exists many tunable parameters, such as s in the Freeman-Davis algorithm, which aids in recognizing exactly the right level of cornerity. Even the heuristic used to quantize cornerity is readily changeable. In fact, many variations of the two algorithms listed above, such as those in [BEU87, ROS75, KOP92], use the same basic method with a modified heuristic.

These methods, however, are prone to misregistering corners if the parameters are not set correctly, if corners are too rounded, if corners are too close together, or if the edge noise exceeds a certain level. In addition, these algorithms do not take into account the geometry of the shape. For example, since the frame openings are approximately four-sided figures, there should be precisely four corners. None of the examined boundary-based methods exploit the known overall geometry of the shape.

3.4 A Compromise

If the image of the frame were ideal, all of the methods in this chapter could obtain usable features for use in recognition. An ideal image of the frame has precisely straight edges, with all corners well defined at right angles. However, due to hardware constraints, edge noise, and other mechanical problems, the recognition subsystem must be able to process a nonideal image. By designing the recognition algorithm to support nonideal images, a degree of error-resistance is added that is very desirable for factory floor operation.

Each of the methods discussed previously have inherent advantages and disadvantages. These are summarized below:

Table 3.1: Comparison of the different feature based methods.

Method	Advantages	Disadvantages
Image Invariants	• Fast--one pass over image • Not sensitive to image rotation, translation, or scaling • Returns single unique feature for entire image	• No visualizable counterpart • Sensitive to noise • Not adjustable--manufacturing deviations not supportable
Blob Analysis	• Visualizable • Uses fact that openings are approximately rectangular	• Susceptible to edge noise • No support for poorly scanned frames • Cannot reconstruct rounded corners
Rectangle Fitting	• Visualizable • Rounded corners can be reconstructed • Resistant to edge noise • Uses fact that openings are approximately rectangular	• No support for timing-based distortion
Boundary-Based Corner Detectors	• Adjustable parameters • Visualizable • Many different methods available	• Sensitive to local errors like edge noise • Do not take into account shape of opening • Cannot reconstruct rounded corners

An ideal recognition algorithm for frame recognition would have the benefits of all of these algorithms, while minimizing the effects of their disadvantages. Since image moment invariants are too inflexible and abstract, the λ-matrix was selected as F. Consequently, the corners must be obtained from the image of the frame and processed.

By integrating all of the corner-based methods together, a recognition algorithm is possible which is error resistant, visualizable, and fast. By basing the algorithm directly on the structural decomposition of a frame (Figure 3.5), the frame image at the highest level is segmented into smaller and smaller units, which can be processed using self-

contained specialized algorithms. Since the raw quantity of data is being reduced by each step, the more computationally-intense algorithms can be designed to perform on the minimum amount of information necessary. This results in a speed increase as compared to processing the image as a whole. Furthermore, each step can report local errors in their processing, causing recognition to abort immediately, rather than waiting for the entire image to finish processing. Once the image is reduced to the desired features, the corners, this information can be abstracted into dimensions to complete λ^* based on the general characteristics of a frame. The final structure-based recognition algorithm is detailed in the next chapter.

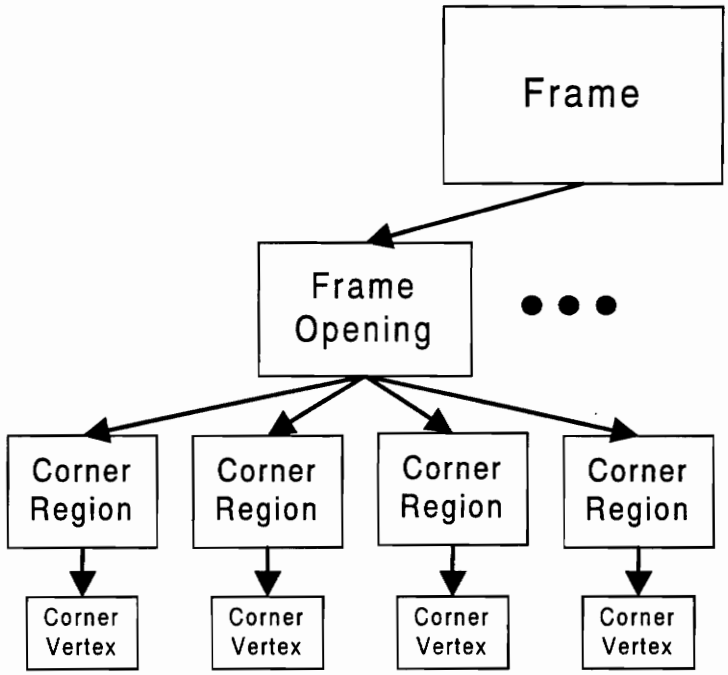


Figure 3.5: A structural decomposition of a cabinet frame. Each step down in the graph reduces the information that the algorithm module must process (represented by the box size). A corner region is a region of an opening boundary that contains a corner, discussed in the next chapter. Note that the diagram above is only of one branch; there would be several branches, one for each opening, and a subbranch for each corner region.

4. The Recognition Subsystem

4.1 Overview

The recognition subsystem accepts the scanned image of a frame, processes the image to determine the λ*-matrix, and finds the frame style code that corresponds to it. The algorithm is modeled after the structural decomposition of a frame, where the parts of a frame image are subdivided further and further until λ* can be obtained. This chapter discusses the development and implementation of the recognition algorithm.

4.2 A Structure-Based Approach

The recognition algorithm is composed of a network of modular steps that generally follows the structural decomposition of a frame (Figure 3.5). Each step, or *node*, of the recognition algorithm performs a specialized operation on the image information it is given, and passes its results on to a group of successor nodes. A block diagram of a generic node in the algorithm is shown in Figure 4.1.

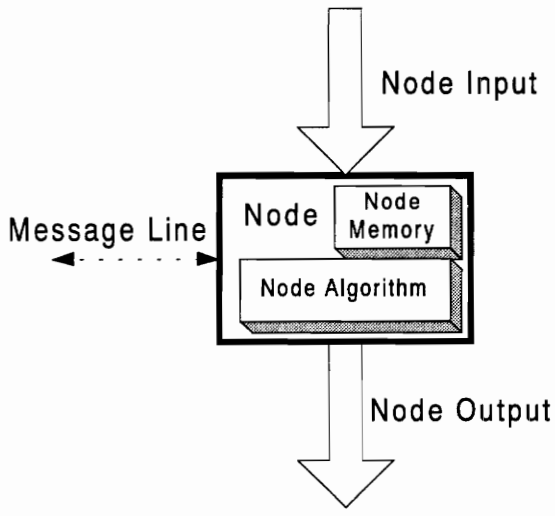


Figure 4.1: A generic node in the frame recognition algorithm. The regions of the node are labeled.

The operation performed by the node is called its *node algorithm*, and its input and output data and parameters are called the *node input* and *node output*, respectively. A node typically has its own local memory for operations, which is called the *node memory*. Finally, the *message line* communicates with the global environment of the program, which contains global variables, the user interface, and error handlers. The node input, node output, and message line all comprise the *node interface*. A node is referred to by the node algorithm it performs.

By basing the algorithm directly on the structure of a frame, the frame recognition subsystem has several benefits over other types of industrial image processing systems. Examples are those developed by [BRÜ93] and [WAY93]. These benefits are as follows.

1. Modular design. Since a node is inherently designed to be as independent as possible, the recognition subsystem can benefit from modular design. For instance, by isolating the nodes, a node algorithm can be upgraded or tested independently of its neighbors. In the case of upgrading, the new node algorithm can directly replace the old providing the node interface is the same. A node can be tested simply by controlling its node input and intercepting its node output. Through the use of the message line, the recognition subsystem has a high level of error support. Since the nodes are independent, when one returns an error, the system can stop recognition immediately, and report the errors from the failing node. This is much more efficient than collecting error messages and waiting until the algorithm completes normally. Once the initial error occurs, all subsequent information is for all practical purposes useless. As a final note, this modularity helps the algorithm to map very easily into software, where each node can be its own subroutine.
2. Data reduction. Since the host computer is an IBM-compatible personal computer, efficient memory usage is one of the primary considerations. If the algorithm is ordered so that successive nodes operate on a smaller and smaller set of data, memory can be freed as soon as possible. Processing speed is also

improved if the node algorithms are placed so that the more intricate algorithms operate on the smallest data set possible.

3. Parallel execution. Although parallel operations are extremely difficult to implement on a personal computer with the DOS operating system, the hierarchical structure of a frame lends itself very well to parallelization. Each opening can be treated as an individual process, returning its corner-based dimensions to some central program for further processing.

The overall algorithm was designed to maximize these benefits, with the exception of parallel execution. An example flow graph is shown below in Figure 4.2.

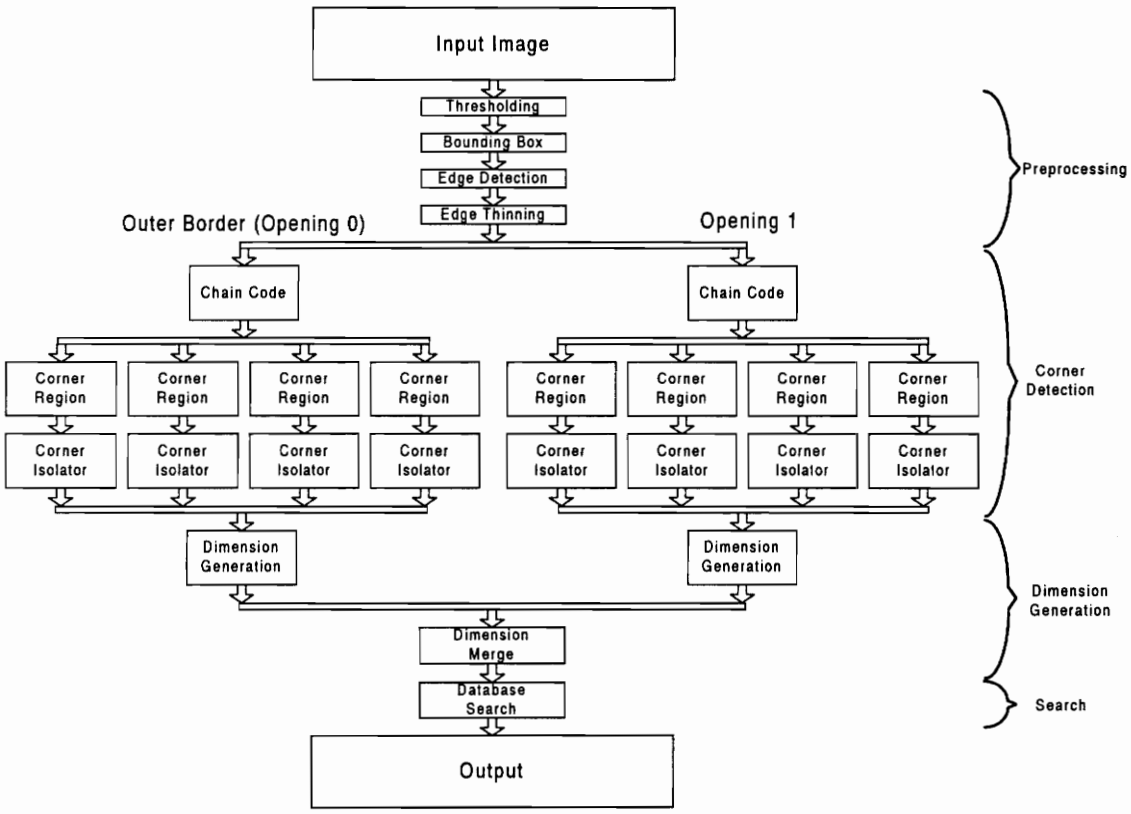


Figure 4.2: Graph of the recognition algorithm/subsystem for a 1-opening frame. There are five classifications of nodes total, where each node contributes towards a specific goal. There is one main branch for each opening in the frame, including the outside border. Each opening in turn has four subbranches, one for each corner to be generated. After corner generation, the information is integrated together into dimensions for that opening. Finally, the information for each opening is combined into the λ^*-matrix for the database search.

As can be seen in Figure 4.2, there are several classes of nodes. These are:

- Preprocessing Nodes
- Corner Detection Nodes
- Dimension Generation Nodes
- Search Nodes

Each opening region is processed completely into dimensions before the next opening is started. Once all openings are processed, the information for the entire scanned frame is compiled together and searched for in a database of frame styles. Each of the nodes in these classes are discussed in the following sections.

4.3 Preprocessing Nodes

The purpose of the preprocessing nodes is to reduce the image of the scanned frame from the acquisition subsystem into a compact image that contains only border information for the frame. This reduction process is accomplished through four nodes that process the image sequentially. These nodes are shown below:

1. Thresholding
2. Bounding Box
3. Edge Detection
4. Edge Thinning

Thresholding binarizes the gray level image, partially filtering out noise. The bounding box node extracts the region containing the silhouette of the frame from the complete image. Edge detection is then used to produce a border representation of the frame. Finally, edge thinning reduces all of the edges to one pixel width, in preparation for chain coding and segmentation by the corner detection class of nodes.

4.3.1 Thresholding

As mentioned previously, thresholding is the process that reduces an gray scale image to binary, that is, pure black and white. The function used to threshold a image pixel $I(x, y)$ in an image I is as follows:

$$I(x,y) = \begin{cases} 255, & \text{if } I(x,y) < 91 \\ 0, & \text{if } I(x,y) \geq 91 \end{cases} \qquad \forall x, \forall y \tag{4.1}$$

That is, for each pixel in the image, if the pixel intensity is less than 91, it is set to the foreground value of 255, if not, it is set to the background of 0. The threshold level of 91 was determined experimentally to minimize gray level edge noise effects. The image is inverted for convenience so the frame silhouette appears as a light region with a dark background. Figures 2.11 and 2.12 show thresholded images of frames.

The thresholding is performed in hardware by the Data Raptor video digitizer. A look-up table, or LUT, is integrated into the circuit that transfers data from the image buffer into the host computer's memory. By programming the look-up table with the appropriate step function, the thresholding node can be implemented in real time with no loss of performance.

4.3.2 Bounding Box

The Data-Raptor video digitizer is designed to acquire an image of certain fixed dimensions. While this is not a problem with area-scan cameras, linescan cameras like the one used in the acquisition subsystem do not return a fixed size image but rather a "waterfall" of single lines (Section 2.4). As a consequence, to support linescan cameras, the digitizer enforces a "pseudo-frame". This pseudo-frame, not to be confused with a cabinet frame, is a structure in the Raptor's image buffer with static width and height. The width of the buffer is set to 1024 pixels, the line size of the camera, and the height is set to 1024 lines. This is the default image size. By placing acquired lines in the pseudo-

frame and automatically resetting back to the beginning when it is completed, the video digitzer emulates waterfall operation.

The pseudo-frame, by design, is more than adequate to contain even the largest frame. In fact, a great majority of frame styles will encompass only a small percentage of the overall image. The remainder should be discarded as it contains no meaningful information and wastes memory. One way to extract the silhouette's area from the image is to use a bounding box. Essentially, a bounding box is the smallest rectangle aligned with the image coordinate axes that circumscribes the area of interest (AOI), which in this case is the image of the frame itself and which is aligned vertically and horizontally. The bounding box of frame in an binary image I can be determined from the algorithm shown in Figure 4.3. An example of a bounding box is shown in Figure 4.4.

```
FIND_BBOX:
        for all x in I,
                for all y in I,
                        if (I(x, y)=foreground) then
                                if (x<min_x), min_x=x
                                if (x>max_x), max_x=x
                                if (y<min_y), min_y=y
                                if (y>max_y), max_y=y
                        end if
                end for
        end for
        bb_ulcorner=(min_x, min_y)
        bb_urcorner=(max_x, min_y)
        bb_llcorner=(min_x, max_y)
        bb_lrcorner=(max_x, max_y)
end FIND_BBOX
```

Figure 4.3: Bounding box algorithm. bb_ulcorner, bb_urcorner, bb_llcorner, and bb_lrcorner contain the upper-left, upper-right, lower-left, and lower-right vertices of the bounding box respectively.

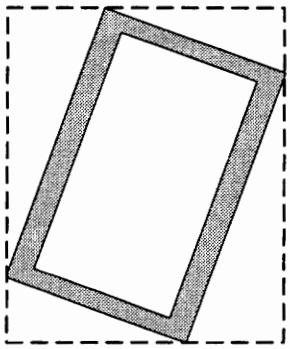


Figure 4.4: Bounding box of a cabinet frame. The dashed rectangle is the bounding box of the frame. The frame is unaligned for bounding box clarity.

Once the vertices of the bounding box are known, the subimage B defined by the box is copied into an optimally sized memory structure in the host computer's memory. The original structure is then discarded. The remaining nodes in the algorithm then operate on this smaller subimage.

4.3.3 Edge Detection

After the image area is optimized, the silhouette is reduced to a boundary representation through the use of an edge detector. While there are many edge detectors and algorithms, such as those by Prewitt [PRE70], Roberts [ROB65], and Canny [CAN86], the Sobel edge detector [DUD73] produces a balance between performance and speed [PIT93, DAV90]. In fact, for its complexity, it is one of the most accurate edge detectors [DAV90]. The Sobel technique consists of applying two templates to an image, shown in Figure 4.5.

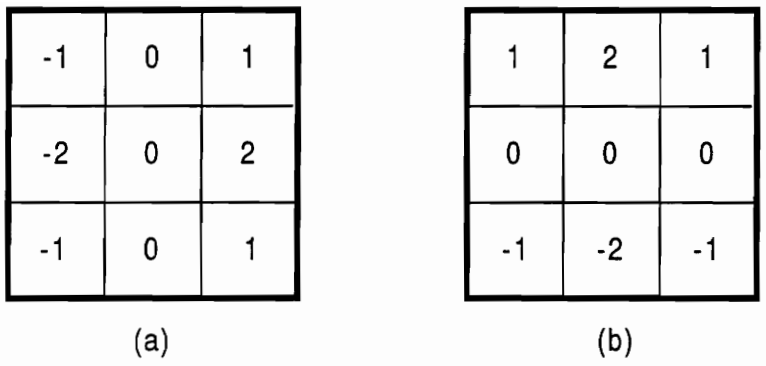


Figure 4.5: Sobel edge detector templates. (a) Horizontal edge detection. (b) Vertical edge detection.

Each of the Sobel templates is sensitive to edges with a single orientation. By applying both templates to each pixel in the image, the magnitude of two together form a heuristic that represents the edge likelihood for that pixel. Mathematically, this is represented as:

$$E(P) = \sqrt{H(P)^2 + V(P)^2}; \quad \forall P, \tag{4.2}$$

where for a pixel P, $E(P)$ is the edge likelihood heuristic, $H(P)$ is the result of applying the Sobel horizontal edge detection template, and $V(P)$ is the result of applying the vertical template. If $E(P)$ is close to zero, P is not on an edge. A simple threshold can then be used to define, based on the likelihood values, which image pixels are edges. Figure 4.6 shows an a frame with edge detection applied. For more details on template-based edge detection, consult [HAR93b] and [PRA78].

Since edge thickness is proportional to the template size [PIT93], and a single pixel width boundary representation is desired, the edges extracted through Sobel edge detection must next be thinned.

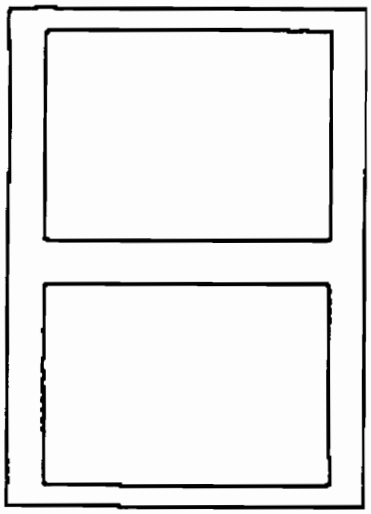


Figure 4.6: Edge detected image of frame style W3615.

4.3.4 Edge Thinning

The edge thinning node reduces the thick edges produced by the edge detection node into one-pixel thick boundaries. Thinning is defined basically as a set of successive erosions of the outermost layers of a shape, until a connected unit-width set of lines, or skeleton, is obtained [DAV81]. Thinning should also comply, as closely as possible, with the constraints listed below [PAR94, PIT93]:

1. Skeletal pixels should maintain connectivity; that is, skeletal pixels should be connected to one another to form the same number of regions as the original image.
2. The pixels comprising the skeleton should lie as close to the center of the cross-section of the original region as possible.
3. Thinned shape limbs are not shortened when the algorithm is applied (see Figure 4.7).

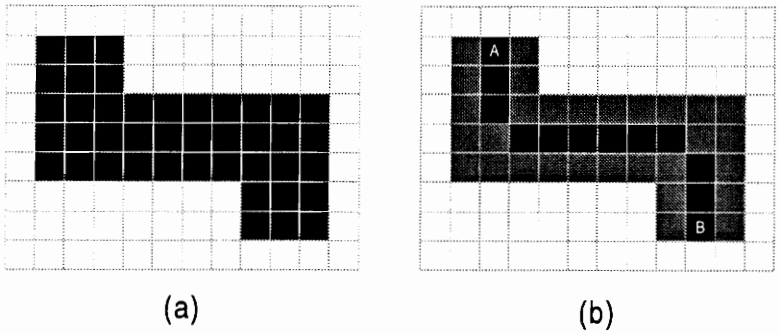


Figure 4.7: An ideal example of thinning. (a) original unthinned shape. (b) shape after thinning. The black pixels are the thinned result, the gray pixels are those that are removed. Note that the three constraints are satisfied, especially #3 which prohibits pixels A and B from being removed, which would shorten the overall shape.

No precise mathematical definition exists for the thinning operation [PIT94]--it can best described as a morphological algorithm [GIA88]. As a consequence, there are many different thinning techniques, such as those in [DAV90, TAM78]. One popular thinning algorithm is the Zhang-Suen method [ZHA84]. The Zhang-Suen method uses multiple iterations over an image in a technique called *successive deletion*, where at each iteration, pixels are marked to be deleted and then removed in a second pass. For speed, the determination of whether or not a pixel is deleted is based solely on its 8-connected neighbor pixels. The process is complete and the image is thinned when no more pixels can be deleted from the image. The Zhang-Suen method uses the following rules to mark pixels that can be deleted [PAR94].

- A pixel can be deleted only if it has more than one or fewer than seven (8-connected) neighbor pixels. This maintains the first constraint listed above.
- A pixel can be deleted only if its *crossing index* is one. The crossing index is the number of individual subregions that a single image region would divide into if the pixel was removed. For example, removing a pixel with a crossing index of two from a region would cause the image region to break into two separate subregions. In the case of the Zhang-Suen technique, only the 8-connected neighborhood of a pixel is evaluated to determine the counting index. The counting index is a simplified form of the counting number in [DAV90]. Figure 4.8 demonstrates the concept of the counting index. This maintains the third constraint listed above.

To comply with the second constraint, each pass to mark pixels is done both forwards and backwards over the image. This reduces pixel bias based on the pass direction. For efficiency, both passes are merged into one by looking at different sides of a pixel's neighborhood separately. The Zhang-Suen algorithm is shown below in Figure 4.9.

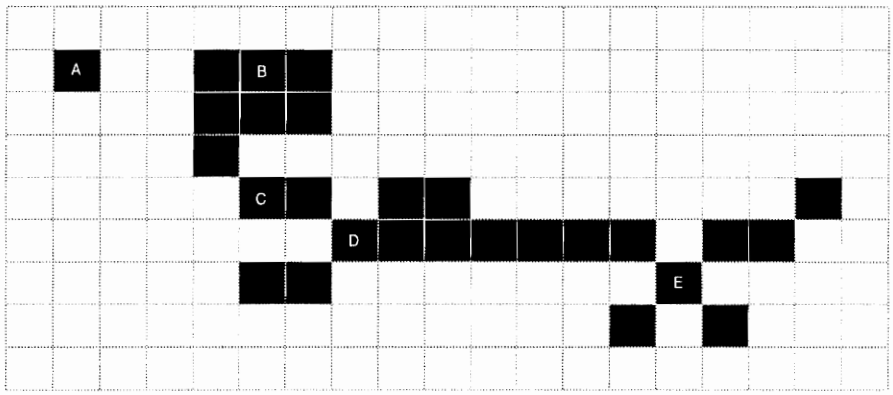


Figure 4.8: Crossing index examples. Pixel A has a crossing index of 0--it does not connect a region to any other. Pixels B, C, D, and E have crossing indices of 1, 2, 3, and 4, respectively.

```
THIN:
    copy image A into B

    repeat
        ImageChanged=FALSE
        for all x in B,
            for all y in B,
                if B(x, y) is a foreground pixel (i.e., B(x, y)=255),
                    if B(x, y) has more than one or less than seven 8-connected neighbors,
                        if B(x, y) crossing index=1,
                            if there is a neighbor pixel in the 0, 2, or 6 directions*
                              and in the 0, 4, or 6 directions*,
                                delete the pixel in A (i.e., A(x, y)=0 )
                                ImageChanged=TRUE
                            end if
                        end if
                    end if
                end if
            end for
        end for

        copy image A into B

        for all x in A,
            for all y in A,
                if A(x, y) is a foreground pixel (i.e., A(x, y)=255),
                    if A(x, y) has more than one or less than seven 8-connected neighbors,
                        if A(x, y) crossing index=1,
                            if there is a neighbor pixel in the 0, 2, or 4 directions*
                              and in the 2, 4, or 6 directions*,
                                delete the pixel in B (i.e., B(x, y)=0 )
                                ImageChanged=TRUE
                            end if
                        end if
                    end if
                end if
            end for
        end for

        copy image B into A
    until ImageChanged=FALSE

    return image A
end THIN
```

Figure 4.9: Zhang-Suen thinning algorithm pseudocode. Image A is input, and image A is returned thinned. The copying of the images back and forth makes prevents the current working image from being altered before the pass is complete. *Numeric directions for neighbor pixels are based on 8-connected neighbors of the current pixel at (x, y), where the rightmost pixel is 0 and counts up to 7 counterclockwise.

As [PAR94] observes, thinning methods are not perfect. Because of the limitations of the Zhang-Suen method, a few pixels that are not part of the thinned skeleton remain. In addition, it may occasionally violate the third constraint of thinning listed above, eroding the tips of an image line segment slightly. This, however, is not a problem for the frame recognition system because the image at this point should consist of closed curves only. Generally, the Zhang-Suen method thins the outlines of frame openings relatively well. Figure 4.10 shows a thinned boundary representation of a frame.

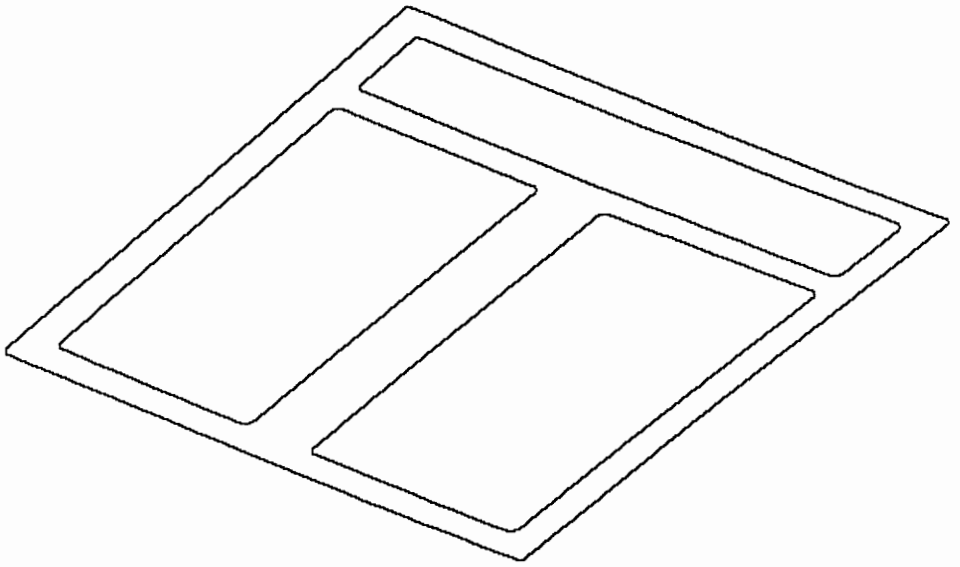


Figure 4.10: Edge thinned image of unaligned frame style B27.

4.4 Corner Detection Nodes

The corner generation class of nodes is where the algorithm truly follows the structure of a frame, segmenting the edge-thinned frame image into compact border representations for each opening, and then extracting four corners from each. The process is similar in concept to the multiscale techniques discussed in [LIN94, BAU92], and the vectorization techniques shown in [GIL93, HOR94]. Corner generation basically considers each opening border, locates regions where corners are extremely likely, and interpolates the exact corner vertices. Corners are isolated from an opening before the

next opening is processed. Extraneous information is discarded after each node in the interest of efficiency and memory conservation.

4.4.1 Chain Coding

Given an edge thinned image, the next step is to distinguish the opening borders from one another so each can be processed individually. Region labeling algorithms are image processing operations that can be used to assign a unique label to each opening. [HAR93b] details several methods for region labeling. One of these algorithms, which was implemented in [BAR94], assigns each pixel a label according to a list of rules, and then merges equivalent labels together. Since this algorithm may produce thousands of labels for an edge-thinned frame image, where almost all are equivalent, this method is computationally inefficient. Furthermore, use of this algorithm requires extensive additional memory to represent those thousands of labels. An alternate solution is available, however.

Through the use of chain codes [FRE61, FRE74], not only can opening borders be reduced into a compact, meaningful form for corner detection, they can also be labeled. Typically in image processing applications, pixels are referred to by their row and column position in an image. A frame boundary in this case would just be a large table of pixel locations, which would not contain any information about the "lay" of the border. Chain codes, however, are based on direction rather than position. Each element of a chain code consists of an octally-encoded direction vector from the current pixel to an adjacent neighbor. Figure 4.11 shows the direction codes from an arbitrary pixel to a neighbor.

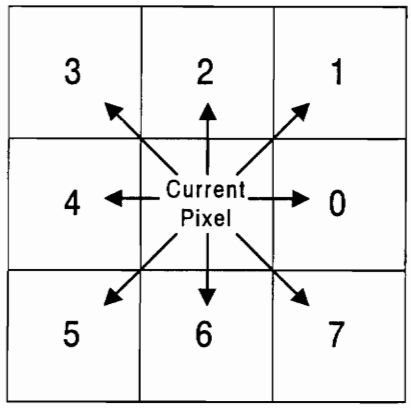


Figure 4.11: Direction codes for an 8-connected chain code. The current pixel is positioned in the center, and the direction to each of its neighbors is an octal digit. Another valid, yet nonstandard coding scheme is for the digits to increment clockwise.

A frame boundary can be modeled as a closed curve. Given a closed curve $B=\{(x_i, y_i) \mid i=0,1, ..., M\text{-}1\}$, where (x_i, y_i) is connected to both (x_{i-1}, y_{i-1}) and (x_{i+1}, y_{i+1}), a chain code $C(B)$ is a circular list of the form

$$C(B)=\{d_{0 \text{ MOD } M}, d_{1 \text{ MOD } M}, d_{2 \text{ MOD } M}, d_{3 \text{ MOD } M}, ..., d_{(M-1) \text{ MOD } M}\}, \quad (4.3)$$

where $d_{i \text{ MOD } M}$ is the direction code from curve pixel (x_i, y_i) to (x_{i+1}, y_{i+1}), and M is $\#B$, the number of pixels in B.

A direction code d_i from the pixel (x_i, y_i) to the next is determined by performing a counterclockwise search around the current pixel. The first foreground neighbor pixel encountered is in the direction to take. Typically, the search starts in the pixel direction $(d_{i-1}+5)$ MOD 8, and ends at $(d_{i-1}+3)$ MOD 8, where d_{i-1} is the direction code from the previous pixel to the current one. Care must be taken not to create an infinite loop between two pixels, and consequently the previous pixel represented by the chain must not be searched. If no neighbor pixel is found in this range, the chain is considered "lost"

and the chain code generation node returns an error over its message line. Because of this search method, opening borders are traced in a clockwise fashion. Figure 4.12 shows the chain code generation of a sample image.

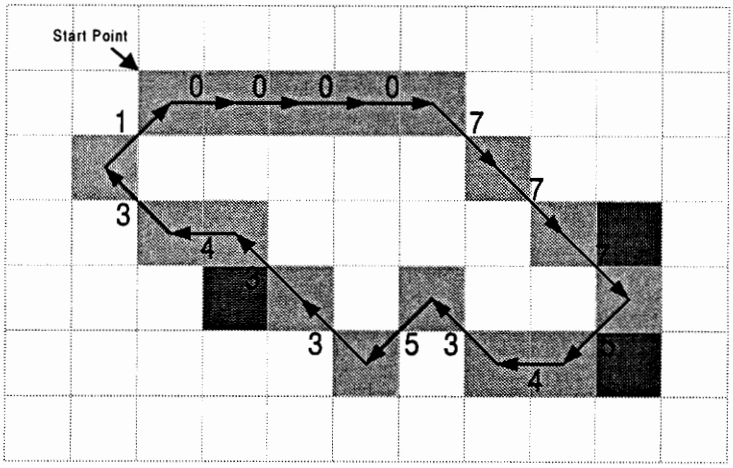


Figure 4.12: Chain coding of a sample outline. With the start point specified, the chain code would be "0000777543533431". The darkened pixels are not included in the chain code even though they are present in this example because of the counterclockwise direction of the neighbor search. These excess pixels could exist because of the limitations of the Zhang-Suen thinning method. By starting at another (non-darkened) foreground pixel, the chain would be the same, only circularly shifted. Also note that the chain code contains no information about the outline's position in 2-D space.

Figure 4.12 also implies two other facts about chain codes. First, a chain code is translation invariant, like the moment invariants in Equations 3.2 through 3.7, since the position of every pixel is only defined relative to another. Second, the overall chain code depends on its starting point on the boundary. By selecting another pixel in the image to start at, the chain code would merely be a circularly shifted version of the original. For the chain code generation node developed here, the starting point of the chain, or *chain seed*, is the pixel that is returned when the chain neighbor search is performed on the first point on the opening border encountered in a raster search of the edge-thinned image. Furthermore, by maintaining the row and column of the chain seed, the chain code can be translationally placed in the overall image. This is useful for returning the coordinates of

points of interest in the chain code. For computational speed, the chain code generation node generates a structure for each pixel that contains the direction code as well as its position in the image, rather than recalculating the coordinates of pixels at a future time.

As mentioned at the beginning of this section, it is possible to combine region labeling and chain coding with some modifications to the basic chain coding algorithm. The combined algorithm is shown below in Figure 4.13.

```
PROCESS_IMAGE:
        label=0
        for all x in I,
                for all y in I,
                        if I(x, y) is a foreground pixel  (I(x, y)=255),
                                label++
                                cccode_and_label with SEED=I(x, y) and OPENINGLABEL=label
                        end if
                end for
        end for
end PROCESS_IMAGE

CCODE_AND_LABEL:
        generate chain[0], the direction code to next foreground pixel, based on previous pixel SEED
        label the pixel coordinate representing chain[0] OPENINGLABEL
        last_pixel=chain[0]
        i=0
        do
                i++
                generate chain[i], the direction code to next foreground pixel, based on previous pixel
                        last_pixel

                label the pixel coordinate representing chain[index] OPENINGLABEL
                if (i>2)
                        delete all neighbors of last_pixel except chain[i] and chain[i-2] (last_pixel's
                        last_pixel)
                end if
                last_pixel=chain[i]
        until chain completely represents the closed boundary

        return chain
end CCODE_AND_LABEL
```

Figure 4.13: Pseudocode for labeling and chain code generation algorithm. The chain is returned in the form of an array *chain*, indexed with the variable *I*. Circularity is implied. PROCESS_IMAGE performs a raster scan over an image *I* and stops to perform the chain coding labeling operation CCODE_AND_LABEL when a foreground pixel is encountered. Foreground pixels are defined as pixel value 255. The labeling automatically alters the value of the foreground pixels to the value of *OPENINGLABEL*. The maximum legal value for *OPENINGLABEL* is 9, which is for an 8-opening frame plus 1 for the outside borders. As a consequence, the pixel value after labeling will not conflict with the foreground value. Excess neighbor pixels before the third element in the chain are retained so that the chain is not inadvertently broken.

Each chain is generated using only the foreground pixels. As a chain is generated, its pixels are labeled, and other neighbors not selected for the chain are deleted. As a consequence, when the first chain is completed and labeled, the raster scan continues uninterrupted until the first pixel of another unlabeled opening border is reached.

With the algorithm listed above there is only one very rare arrangement of pixels that can occur which can cause the chain to be "lost", as shown in Figure 4.14. This pixel arrangement is caused by the limitations of the Zhang-Suen thinning algorithm.

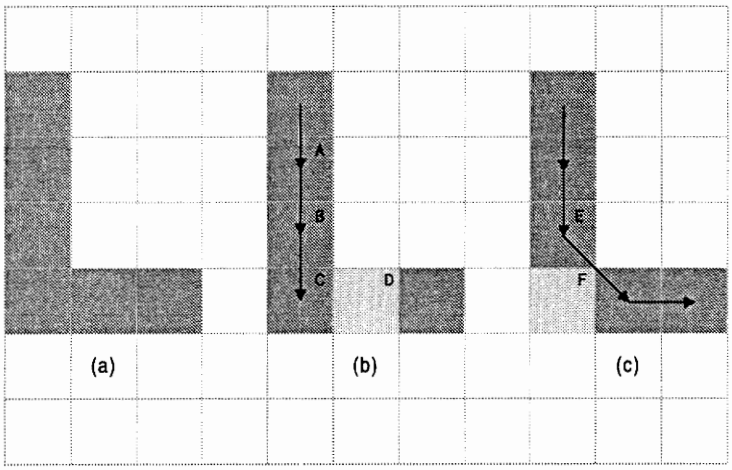


Figure 4.14: A pixel configuration that requires a neighbor history to be maintained. Notice that a perfect thinning method that produces 8-connected curves would delete the lower left pixel. However, this situation can occur due to the limitations of the Zhang-Suen thinning method. (a) shows the original boundary. In (b), when the chain reaches point B and performs a counterclockwise search for the next pixel, C is selected. However, all other neighbors except A and B are deleted, removing D. The chain is lost on the next iteration--it can no longer continue. (c) To compensate for this, when the chain is lost, the neighborhoods of E and F are restored, and F is then deleted. The algorithm then continues as normal from E.

To compensate for this, the neighborhoods of the two previous pixels in the chain are saved before any deletions. When this error occurs, the algorithm backtracks two

pixels, deleting pixel F in Figure 4.14c. The algorithm then proceeds normally. Since this situation occurs rarely for rectangular frames when edge-noise is overly pervasive, the overall performance loss by adding this support is negligible.

This algorithm was tested in isolation in a laboratory environment before implementing it in the node for the recognition subsystem. It produced excellent results, and runs in at most one third of the time required for the previous algorithm implemented by Bari [BAR94]. The significant speedup is due to the fact that the labeling is directly integrated into the chain code generation, rather than being performed separately. The chain code representation provides useful information for corner region isolation, detailed in the next section.

4.4.2 Corner Region Generator

After the chain codes for each border have been extracted from the image, the directional information stored by the chain code can be used to locate corners. Chain codes are used in corner detection algorithms such as those discussed in Section 3.3.3. Whereas those methods seek to extract corner points directly from the chain, the corner generation algorithm presented here uses the chain code only loosely to approximate corners. This node identifies regions of the opening border curve that have a high curvature, or *corner regions,* based on the chain code.

The first step is to convert the chain code representation of a curve to a *difference chain code.* This is the first differential between successive elements in the 8-connected chain code. Each differential step is computed by counting in a counterclockwise manner the number of left-hand turns (in increments of 45 degrees) between a chain code element d_i and its preceding element d_{i-1} [PIT93]. Based on this, a difference chain code step has a value h ranging from h=-4 to h=+4, shown in Figure 4.15, where the angle change between the pixels is $45° \cdot h$. Figure 4.16 demonstrates difference chain coding.

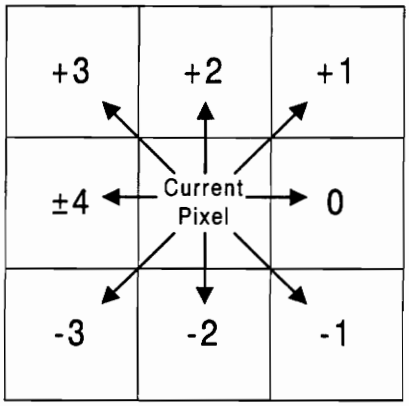


Figure 4.16: Direction codes for an 8-connected difference chain code. The current pixel is positioned in the center, and the direction change to the next pixel of the chain is represented by a value from -4 to +4. Note that +4 is equivalent to -4.

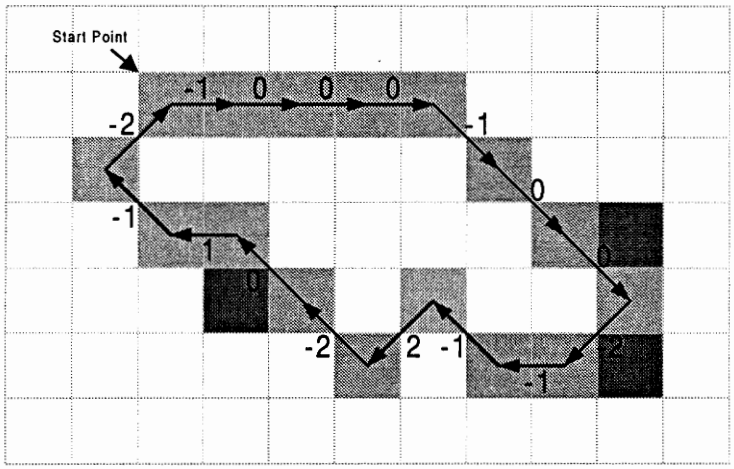


Figure 4.15: Difference chain coding of Figure 4.12. Each number represents the angle change from one pixel to the next. Straight runs of pixels are denoted by a series of zeros. Note that the chain code generation technique is identical to that of regular chain coding, and the same pixels are disregarded.

With this alternate coding scheme, the chain code for a perfectly aligned and noise-free rectangular curve would ideally consist of long runs of 0's, separated by pairs

of -1's. The zeros result from the straight sides, where no angle changes occur, and the pairs of -1s at the right angle corners are due to the chain code's counterclockwise search pattern. In this case, the corner regions would be defined by the -1s.

However, as discussed previously, the outline of an opening border is not ideal. Edge noise causes the border to be slightly jagged. The chain code reflects this noise through spurious value changes. Corner rounding additionally serves to create less distinct corner regions. Furthermore, unaligned frames have sides that are not truly straight due to pixel quantization. This causes the representation of the straight sides in difference chain code to deviate from all 0's. Figure 4.17 shows examples of these imperfections.

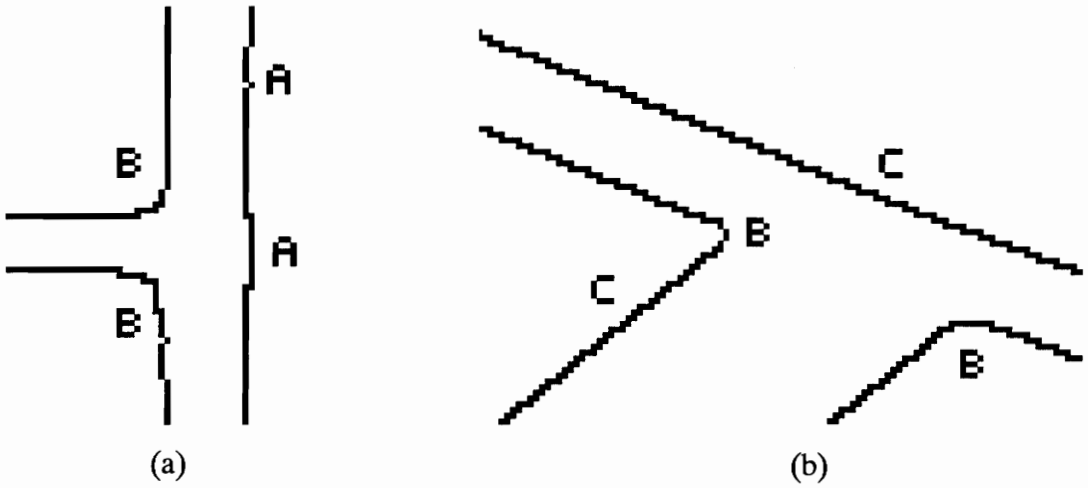


Figure 4.17: Imperfections in a boundary representation that affect difference chain codes. Images (a) and (b) are fragments of actual output from the edge thinning node. (A) indicates spurious noise in the border due to edge noise, (B) rounded corners, and (C) quantization effects of an unaligned side. When difference chain coded, a side like this contains many tiny direction changes (Figure 4.18).

To compensate for these problems, the difference chain code values are smoothed and mode filtered before corner regions are extracted. Mathematically, the smoothing in the corner region generation node is represented as

$$w_n = \text{ROUND}((h_n+h_{n-1}+h_{n-2}+h_{n-3}+h_{n-4}+h_{n-5}+h_{n-6}+h_{n-7}+h_{n-8}+h_{n-9})/10), \qquad (4.4)$$

where h_n is the nth difference chain code value, and ROUND is "round to the nearest integer."

This heavy smoothing serves to make the chain code resistant to sudden changes in value, such as those caused by edge noise. Only persistent changes, such as those where the chain code responds to an actual corner on the border, will remain. Also, note that only the difference chain code values are smoothed, not the pixel coordinates they are associated with. During corner region detection, the processed chain code values are treated as a mapping describing the "lay" of the border separate from its pixel coordinates. Since smoothing essentially "smears" the chain, the direction differences will only approximately correspond to actual image pixel locations. This is intentional because if the image coordinates were smoothed as well, the scanned border would be changed. The node algorithm compensates for the "smearing" when the corner regions are extracted from the chain.

Next, mode filtering is performed on the smoothed chain to remove any remaining spurious values, typically caused by rounding errors. The statistical mode of a set is the member of that set that occurs most frequently. Mode filtering, in this case, takes the form of a "vote" between the previous value in the chain w_{n-1} and the next value in the chain w_{n+1} to nondestructively determine the current chain value k_n. If there is no conclusive mode between the three values, the value is left unchanged. Mathematically this is represented below.

$$k_n = \begin{cases} w_{n-1}, & \text{if } w_{n-1} = w_{n+1} \\ w_n, & \text{if } w_{n-1} \neq w_{n+1} \end{cases} \tag{4.5}$$

At this point, the processed difference chain k should have very few spurious values. In fact, due to the chain code generation method (Figure 4.13) and the shape of the openings, the only values that occur after processing are -1, 0, and +1. With this in mind, the corner regions are extracted from the border in three steps. First, the algorithm

must compensate for the presence of +1s in the difference chain. Due to quantization, an angled line is represented in a difference chain code by alternating positive and negative values with some intervening 0s (Figure 4.18). If a -1 is the nearest nonzero value in either direction to a +1, it is deleted because the direction change is effectively canceled. Second, the five neighborhood chain values on each side of the remaining -1s, as well as the position of the -1s, are marked. Third, the chain is searched for marked areas. Since the true corner areas of the border will withstand processing since they are so pronounced, there will be some -1 activity near them. Also, since each -1 produces 11 marked chain values, and the -1s of the corner regions should be near one another, many marked areas will overlap and be merged together. Each of these continuous areas are recorded in clockwise order around the chain, noting their beginning and ends as well as the pixel coordinates they originally described before processing. Since "smearing" causes the chain code values to no longer exactly follow the pixel positions, several chain elements before and after the areas are included to compensate. These areas are the corner regions of that opening.

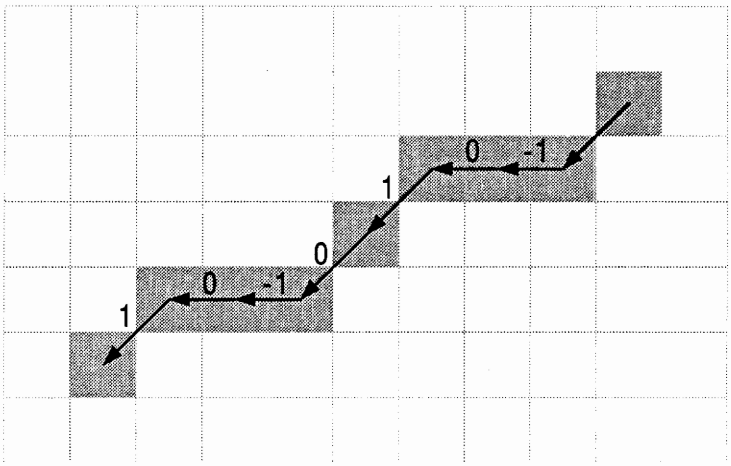


Figure 4.18: A difference chain coded quantized line. A quantized line when difference chain coded consists of alternating -1s and 1s separated by some number of 0s. Each ±1 direction change serves to cancel out one another. By disregarding the nearest -1s to a 1, the corner region generation node treats the above image as a straight line. Since the chain is smoothed and filtered before the marking occurs, many sources of spurious noise are treated the same way.

Based on the fact that each opening is rectangular, if four corner regions are not returned, the corner region generation node is designed to return an error and halt further recognition efforts. This reduces the risk of misclassification in a factory environment. However in an experimental environment, it would be very easy to measure the cornerity of corner regions by the number of marks they contain, including overlapping marks. If more than four regions were discovered, the four regions with the highest scores that are maximally distant from one another could then be used as the corner regions of the shape. If only three were detected, they could be used to triangulate the last region based on their positions and the shape of the border.

Extensive tests of this algorithm on frame boundaries have produced excellent results--four corner regions have always been detected, with each region encompassing an actual corner on the opening border. Sample results are shown in the next section.

The corner regions are extracted using a processed chain code as an indicator of border curvature. Once these regions are obtained, they are the focus of the corner isolation node--the rest of that opening's border and chain code is extraneous. The corner isolation node is detailed below.

4.4.3 Corner Isolation

A corner region is a set of pixel coordinates which represent a portion of a curve that contains a corner. The task of this node is to detect the location of the corner as a function of the pixel locations in the corner region. Since the outlines of the openings are likely to have rounded corners (Figure 4.17), it is possible that the best corner location will not lie on the curve (Figure 3.4). Through the use of a line fitting technique, the corner location can be estimated.

Let a corner region β be represented as the set of ordered pairs,

$$\beta = \{(x_0, y_0); (x_1, y_1); (x_2, y_2); \ldots ; (x_{n-1}, y_{n-1})\}, \tag{4.6}$$

where (x_i, y_i) is a pixel coordinate, in columns and rows.

The approach developed here is to use the two endpoints of the corner region as pivot points for lines that are fit to the remaining curve points. More formally, line ℓ_1 is chosen to pass through (x_0, y_0) and it is the best fit to points

$$\beta_1 = \{(x_0, y_0); (x_1, y_1); \ldots ; (x_p, y_p)\}, \tag{4.7}$$

where p=ROUND((n-1)/2), and ROUND is defined in Equation 4.4. Similarly, line ℓ_2 is chosen to pass through (x_{n-1}, y_{n-1}) with the best fit through

$$\beta_2 = \{(x_p, y_p); \ldots ; (x_{n-1}, y_{n-1})\}. \tag{4.8}$$

The general equation of a line ℓ_i in $\Re^2$, is

$$\ell_i = ax + by + c = 0 \tag{4.9}$$

where $a, b, c \in \Re$. [PAR94] observes that, given a point (x, y) on ℓ_i, and the angle θ of ℓ_i from the horizontal in radians, the use of the coefficient values

$$a = -sin(\theta) \tag{4.10a}$$

$$b = \sqrt{1 - a^2} \tag{4.10b}$$

$$c = -(ax + by) \tag{4.10c}$$

yield the normal form of ℓ_i, where $a^2 + b^2 = 1$.

Based on this information, ℓ_1 can be generated with the algorithm described in Figure 4.19 below. ℓ_2 can be generated in a similar manner.

```
FIT_LINE1:
        startx=x0
        starty=y0
        bestscore=9E9

        for line_angle=-90° to +89°, step +1°,
                score=0
                convert line_angle to radians
                compute a, b, c (Equations 4.10abc) based on angle line_angle and (startx, starty)
                for each member of β1 (call it (xi, yi)),
                        if (abs(axi+byi+c)<=SQRT(2)/4)
                                score-=1
                        end if
                end for
                if score<bestscore,
                        bestscore=score
                        bestline.a=a
                        bestline.b=b
                        bestline.c=c
                end if
        end for

        return bestline
end FIT_LINE1
```

Figure 4.19: Algorithm for line fitting. "*bestline*" is a structure containing the a, b, and c coefficients for the best line that characterizes β_1 . Note that the algorithm evaluates the line only at the pixels in β_1 for efficiency. See the text for further clarifications.

The heuristic for determining the quality of the fit lines is based on the equation for the perpendicular distance d_i of an arbitrary point (x_i, y_i) in unit pixels from ℓ_i, shown below:

$$d_i = \frac{ax_i + by_i + c}{\sqrt{a^2 + b^2}} \tag{4.11}$$

Since the potential fit lines are in normal form, this can be reduced to

$$ax_i+by_i+c=d_i \tag{4.12}$$

The algorithm determines the number of pixels that are within $d_i \leq \sqrt{2}/4$ unit pixels of the line. This value is one-half the distance from the center of a unit square pixel to its corner. The line with the most number of sufficiently close pixels is considered the best fit. Note that this criterion is slightly different than the fitting methods suggested by [PRA78, DUD73], which seek to determine a fit line that minimizes the distance to all of the points in the data set. The method adopted here disregards isolated pixels that deviate significantly from the general orientation of the curve due to noise or other factors.

The corner vertex $V=(x_c, y_c)$ of the corner region is defined to be the intersection of ℓ_1 and ℓ_2. For additional error support, errors are returned and recognition fails if the lines do not intersect at all, or intersect at a point beyond a certain predefined distance from the corner region. Once the coordinates of V are obtained for a corner region, the corner region can then be discarded. Each of the remaining three corner regions for the opening border are then processed in the same way as listed here. All four corner vertices are stored in the same order as their corner regions.

When tested, this structure-based approach to determining the corner vertices performed very well. The table below shows the results of corner isolation on several different frame styles. If a corner vertex is placed within three pixels of an actual (manually-located) corner, it is considered accurate. Frames designated with *'s are misaligned by more than 15°.

Table 4.1: Corner isolation node output result accuracy for several test frame styles. (All images of frame styles were obtained from the acquisition subsystem.)

		Distance from actual corner position (# pixels)		
Frame Style	Total corner vertices	Within 1 pixel	Within 3 pixels	Greater than 3 pixels
B27*	16	12	4	0
B12*	12	9	3	0
B12	12	10	2	0
OV3084	24	18	6	0
W3615*	12	10	2	0
UB60	36	31	5	0
VAN36	28	24	4	0

The table can be compared directly with Bari's results. The results here are greatly improved over the blob analysis technique (Section 3.3.1), especially regarding unaligned frames.

Figure 4.20 shows a completely processed border representation of frame style VAN36, containing the final output from of the corner generation phase and its nodes. Each opening is labeled with a different gray level to indicate its region number. Also, the break in the outlines of the openings indicate the start and end of corner regions. Finally, the individual dark gray pixels in the image are the isolated corner vertices determined for each opening. Note that all of the corners were properly detected, and they are very close to their correct positions, even if their corner region is very curved.

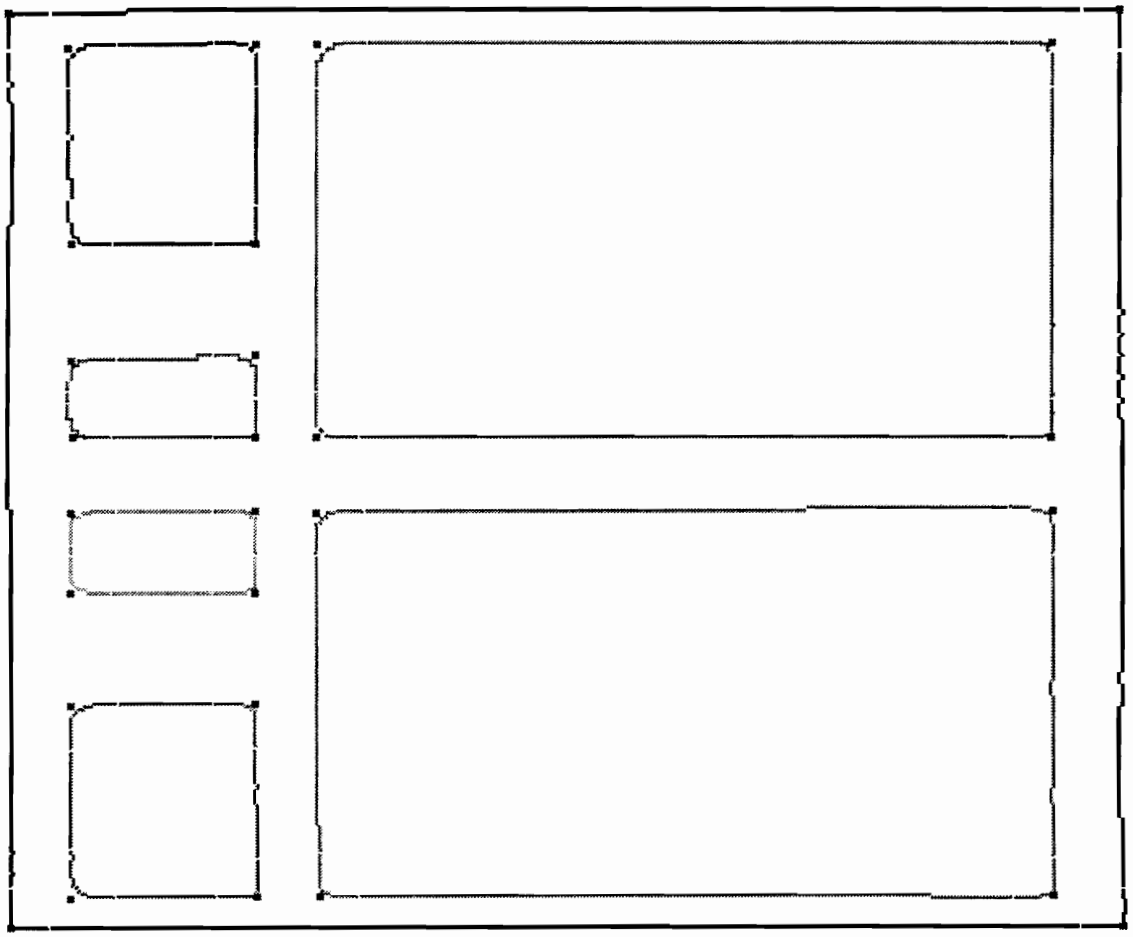


Figure 4.20: Processed image of frame style VAN36. Each region is shaded according to its assigned label. The breaks in the borders represent the corner regions, and the dark gray dots are the interpolated corner vertices determined by the algorithm. Frame size is 27.125" by 36".

Note that this image is for demonstration purposes only; in reality, only the corner vertices with their opening numbers would be retained, to be forwarded to the dimension generation class of nodes.

4.5 Dimension Generation Nodes

4.5.1 Overview

Once corners are determined for each opening border, the dimension generation nodes compiles the dimensions of each opening into a compact form suitable for use by the search node. The dimensions necessary to completely specify the frame can be represented compactly by the λ^* matrix for the scanned frame (Section 1.3). The λ^* matrix requires three pieces of information for each opening (including the outside border) to be complete. This information is listed below.

1. Each opening must be uniquely labeled, so the rows of λ^* can be formed.
2. The length of the sides of an opening. Since the frame openings are rectangular, only the width and height of the opening need to be determined.
3. The distance of the origin of the opening to the origin of the outer opening border. The origin of an opening is defined arbitrarily as its upper left-hand corner vertex. This serves the purpose of placing the opening translationally in the frame.

Only information concerning the corner vertices is needed to obtain this information. The first piece of information is obtained when each opening is initially processed by the chain code generation node--the corner vertices for an opening inherit the opening's label. The second piece of information, the actual dimensions of the opening, are obtained very easily from the corner vertices if the frame is aligned. But, if the frame is unaligned, a more complex technique must be used. Finally, as discussed in Section 1.3, the distances between the opening origins opening serve to consistently order the rows of the λ^*. Both the dimensions and the origins of the openings are computed by the dimension generation node detailed below.

4.5.2 Dimension Generation

The size of each opening is determined by the distances between corners. For example, if the corner regions R_1, R_2, R_3, and R_4 interpolate to form the vertices V_1, V_2, V_3, and V_4 respectively, the sides of the openings would be defined by the line segments V_1V_2, V_2V_3, V_3V_4, and V_4V_1. By computing the distance between each corner, using the formula,

$$D_{ij} = \sqrt{\left(V_{ix} - V_{jx}\right)^2 + \left(V_{iy} - V_{jy}\right)^2}, \tag{4.13}$$

where D_{ij} is the pixel distance between V_i and V_j, and V_{ix} is the x-coordinate of V_i, etc., the length of each side of the opening can be determined. There is, however, one complication with the above formula--it returns the distance in pixels. This must be converted to inches for comparison with frame dimensions. But, the pixels are not square, which Equation 4.13 implicitly assumes. To account for nonsquare pixels, Equation 4.14 shows the revised formula.

$$D_{ij} = \sqrt{\alpha\left(V_{ix} - V_{jx}\right)^2 + \tau\left(V_{iy} - V_{jy}\right)^2}, \tag{4.14}$$

where α is the horizontal (x-direction) inches per pixel conversion factor, τ is the vertical (y-direction) inches per pixel conversion factor, and all other variables are defined the same as in Equation 4.13.

Using same techniques as [BAR94], α was determined for the laboratory system (with a conveyor belt speed of 6 inches/second) to be 0.0752 pixels/inch and τ was determined to be 0.1080 pixels/inch. These values were determined using the camera's characteristics (line size, timing, and size of field of view), as well as the conveyor belt speed. These values were changed slightly in the actual system. In implementation, the actual values used by the frame recognition system are within 13% of these values in the

horizontal (actual value used 0.0657) and within 6% in the vertical (actual value used 0.1137). The horizontal change is due to the fact that the field of view for a linescan camera is very difficult to measure accurately, and the vertical change is due to roundoff. It was also determined that the α and τ values for the outside border of the frame had to be scaled slightly larger (4% in the horizontal and 2% in the vertical). This is attributed to the optical camera distortion and distortion of the original border due to lighting conditions.

Since the frames may be possibly unaligned, however, it is not readily apparent which side lengths represent the observed width and height. By computing the slopes of the sides, it is possible to determine which is which. The algorithm is shown below in Figure 4.21.

```
GET_WIDTH_OR_HEIGHT:
        (performed only for outside border opening)
        locate lowest corner (greatest row value) on the outside border, call it V_A
        locate second lowest corner (2nd greatest row value) on the outside border, call it V_B
        let s be the slope of the line segment between V_A and V_B

        (performed for all openings, including outside border opening)
        for every side of the opening being examined,
                let m=slope of that side
        end for
        return as the "widths" the two sides that have a slope m closest to s
        return as the "heights" the other two
end GET_WIDTH_OR_HEIGHT
```

Figure 4.21: Algorithm for determination of which sides of an opening are widths or heights. The first part of the algorithm determines the orientation of the entire frame, and defines the bottom edge of the outside border as the width. The slopes of all other openings' sides are compared to it. Note that an opening is processed completely before the next is started.

To obtain a single horizontal and vertical dimension for each opening, the two lines designated as "widths" are averaged together, and the two "heights" are averaged together as well. This averaging provides a degree of error tolerance, in case one of the sides of a pair is slightly different from the other. Note that the observed widths and

heights will be interchanged if the frame rotates 90°. This designation change will be supported by the search node, detailed in Section 4.5.

The second task of the dimension generation node is to determine which of the corner vertices is the opening border's origin. It can be determined by computing the two highest corner vertices (smallest and 2nd smallest row values), and selecting the leftmost one (smallest column value) for the origin. Call the origin of an opening border O_V. Note that this computation returns the origin measured relative to the entire image rather than the O_V of the outer border of the frame. By subtracting the coordinate values for the O_V for the outside border from the origin for each opening, the O_V for each opening is adjusted to be relative to the outside borders origin. The origin of the outside border is then changed to (0,0) to reflect this normalization.

4.5.3 Dimension Merge

After all of the borders of the frame have been processed, the dimensions and origins for each are collected together in the dimension merge node. This node essentially completes the parts of the λ*-matrix necessary to perform recognition by the search node.

All of the information for the openings (origin coordinate in pixels, width and height in inches) is paired with its opening label, determined by the chain code generation node. As a consequence, all of the completed rows of λ* are available, but not in any particular order. Since the opening borders are scanned in the chain code generation node in raster order, the outer border is always the first outline encountered and processed as it encompasses all the others. The processing order of the remaining openings depends on the orientation of the scanned frame. The technique used to sequence all of the opening data consistently is to sort the openings by their origin's distance (in inches) to the origin of the outer border, based on the method used by Bari [BAR94].

Once this sort is performed and λ^* checked for obvious recognition errors (such as frames with too many openings), the data is forwarded to the final node in the recognition subsystem, the search node.

4.6 Search Node

The search node performs a database search for the closest frame that matches the data compiled by the dimension merge node. The two primary parts of the search node are the database and the search engine.

The database contains a record for every closed rectangular frame manufactured by American Woodmark. There are 253 different frame styles manufactured. For convenience and efficiency, the database is divided into eight separate text files, each one containing only the records of frame styles with a certain number of interior openings. There are eight files total, representing frames with one interior opening to eight. Each record consists of the items listed below.

- The frame style name.
- The internal product item number, if known.
- The drawing number of the schematic of the frame style, if known.
- Number of openings in the frame, for reference.
- A table that duplicates the entries of the λ-matrix for the frame style derived from an engineering schematic. There is one line for each opening (including the outside border) which contains its origin position and opening size dimensions. The origin position is not listed for the outside border, since it is always (0,0).

All database parameters are given in inches where applicable. Figure 4.22 depicts a sample database entry for the frame style BLB42/45.

```
STYLE                  BLB42/45
ITEM_NUM                  ???
DRAWING_NUM            03-1101B
NUM_OPENINGS                 4
SIZE       39.000     30.500
OPENING     1.625      1.625     14.750     21.250
OPENING    22.625      1.625     14.750     21.250
OPENING     1.625     24.500     14.750      4.375
OPENING    22.625     24.500     14.750      4.375
```

Figure 4.22: Sample database entry for frame style BLB42/45. In this case, the frame item number is not known and is therefore marked "???". SIZE designates the width and height of the outside border. Based on NUM_OPENINGS and the number OPENING lines, the frame has 4 interior openings. The elements of each line of the OPENING statement are the origin's x dimension, the origin's y dimension, the opening's width, and the opening's height, in that order. The SIZE and OPENING lines contain the λ-matrix for the frame.

The database as a whole has no limitations on the number of records it can contain, and can be expanded by simply adding an additional record for each new frame style. Also, it is not necessary for the interior frame openings be listed in any particular order.

Once all desired frame styles are entered into their proper text files and before the frame recognition system is executed, the database is "compiled." This compilation performs several tasks. First, the database is processed with a program developed by Bari that converts each record to a single line where the interior openings are sorted based on the origin distances. By applying the same sorting method to the scanned frame and to the database entries, their λ-matrices will be ordered in the same fashion. Second, each frame record is morphologically transformed to form the λ-matrix for the same frame rotated 90°, 180°, and 270° from the original. All four orientations are then saved back to the database file with the same frame name, item number, and drawing number.

As mentioned previously in Section 4.5.2, the same scanned frame can have different λ*-matrices depending on the rotational alignment of the frame as it is being

scanned. By performing and recording these three additional rotations, all possible valid permutations of heights and widths for each are stored in the database. This permits the search engine to find a match to a frame, if it exists, regardless of alignment.

When the recognition system initializes, the compiled database files are read. The search engine is the interface to the database that finds a match to the scanned frame. Since external file accesses are relatively slow operations, the database files are cached in memory.

After the proper database file has been loaded, the λ-matrices of each of the records are compared to λ*. For a frame with n interior openings, there are $2(n+1)$ dimensions to be compared (height and width for all opening borders). Only the opening sizes, dimension by dimension, are compared. The method of comparison used is an error measure called the mean-squared-error, or MSE, expressed by the formula below.

$$MSE = \left(\frac{1}{2(n+1)}\right) \sum_{k=1}^{2(n+1)} \left[D_k^* - D_k\right]^2, \tag{4.15}$$

where n is the number of interior openings in the scanned frame, k is an index variable representing the different sides of the frame, D_k^* is an opening dimension in inches from λ*, and D_k is D_k^* 's corresponding opening dimension from λ for the database record currently being examined.

Essentially, the difference between each opening dimension in the database and the scanned frame is squared and averaged with the rest. The squaring of each error serves to bias the MSE measure, reducing dimensional differences less than one inch, and accentuating errors greater than an inch. The averaging serves to combine all of the errors together, reducing the effects of spurious individual squared errors on the total error measure. The style of the scanned frame is reported as the database frame style that results in the lowest MSE. However, to prevent misclassifications, the MSE for the pair

must be below 1.0 before a match is declared and the style name is output. If it is not, the MSE is deemed too significant and the node reports an unknown frame style. Additionally, the search engine is designed to report an error if two frames in the database with different style names report the same MSE during the search. Of the 253 frame styles in production, only 10 have dimensions sufficiently close to one another to cause problems with this search technique. These have been identified, and are treated by the system as exceptions.

The next chapter reports overall test results for the frame recognition system in a laboratory and factory environment. In the laboratory, 39 frames styles were tested between 4 and 7 times each at various orientations. In the factory, over 6,600 frame samples were tested. Performance benchmarks for the overall recognition subsystem are also provided.

5. Results

5.1 Results of Laboratory Test

This chapter discusses the results of a test of the complete frame recognition system in the laboratory. Two hundred fifty-six samples were scanned and processed by the system. Each sample was one of 39 different frame styles at 4 to 7 different orientations. Special support (Appendix B) was added to the system so it could recognize the AGxx class of exceptions. The frame styles were selected to be representative of the entire frame product line. The results of this test are shown below.

- 0 test samples were misclassified.
- 0 test samples were reported unknown.
- 256 test samples were classified correctly.

All frame styles were recognized correctly. Appendix A lists the exact results for each of the test samples, with various parameters obtained from the recognition algorithm, including their MSE values. The MSE error for the samples ranged from 0.00002 (for a frame style AG2418) to 0.54637 (for an frame style W1242).

The time benchmarks for each of the samples are also shown in Appendix A. There are two time measures, *PTIME* and *FTIME*, graphically shown in Figure 5.1. *PTIME* is the time in seconds to recognize the frame, from completed image to database match. Generally, *PTIME* is proportional to the area of the bounding box of the frame. This is logical, considering the fact that the larger the bounding box, the larger the frame, and consequently the more pixels in the boundary that must be processed. The small jumps at the ends of the curves for *FTIME* and *PTIME* in Figure 5.1 are due to the fact that the frames with bounding box areas that are larger than the laboratory conveyor belt. This causes the

edges of the opening borders to be severely distorted as the frame wobbles. As a consequence, the system must perform additional computations to compensate.

FTIME is the amount of time in seconds from the point where the frame being scanned clears the sensor pair to the end of recognition. Rather than being a direct measure of system performance, *FTIME* can be used to determine the minimum distance between frames on the conveyor belt for proper recognition in the factory. This distance represents, in turn, the throughput of the system. As discussed in Section 2.2, after the frame clears the sensors of the acquisition subsystem, the system continues acquisition until the frame clears the linescan camera's field of view. Therefore, *FTIME* equals *PTIME* plus some time for the additional acquisition. While the system is running, if two frames are too close together on the conveyor belt, the second one might not be scanned properly. This is because the system cannot acquire a frame image while it performing the recognition algorithm. Once recognition of the first frame is completed, if the sensor beam is still broken by the second frame, acquisition will begin instantly. Since the frames are constantly in motion, it is possible that the second frame may already be partly past the acquisition plane, resulting in the second frame image being truncated. To prevent misclassifications, if two frames are too close together, the system is designed to report an error message and disregard the image of the second frame.

Since *FTIME* is a function of the currently processing frame's *PTIME*, the minimal allowable distance between a pair of frames is a function of the first frame's *PTIME* as well. Given a constant conveyor belt speed, the minimum distance D in inches after a frame F before the next frame can begin is approximately expressed by:

$$D \approx FTIME(F) \cdot C \tag{5.1}$$

where $FTIME(F)$ is the *FTIME* for a frame style F, and C is the conveyor belt speed in inches/second.

In the case of the laboratory system, the conveyor belt speed is 6 inches/second. The minimum distance is also proportional to the bounding box areas since *FTIME* is a function of *PTIME*. A graph of the bounding box area versus the minimum distance is shown in Figure 5.2 below.

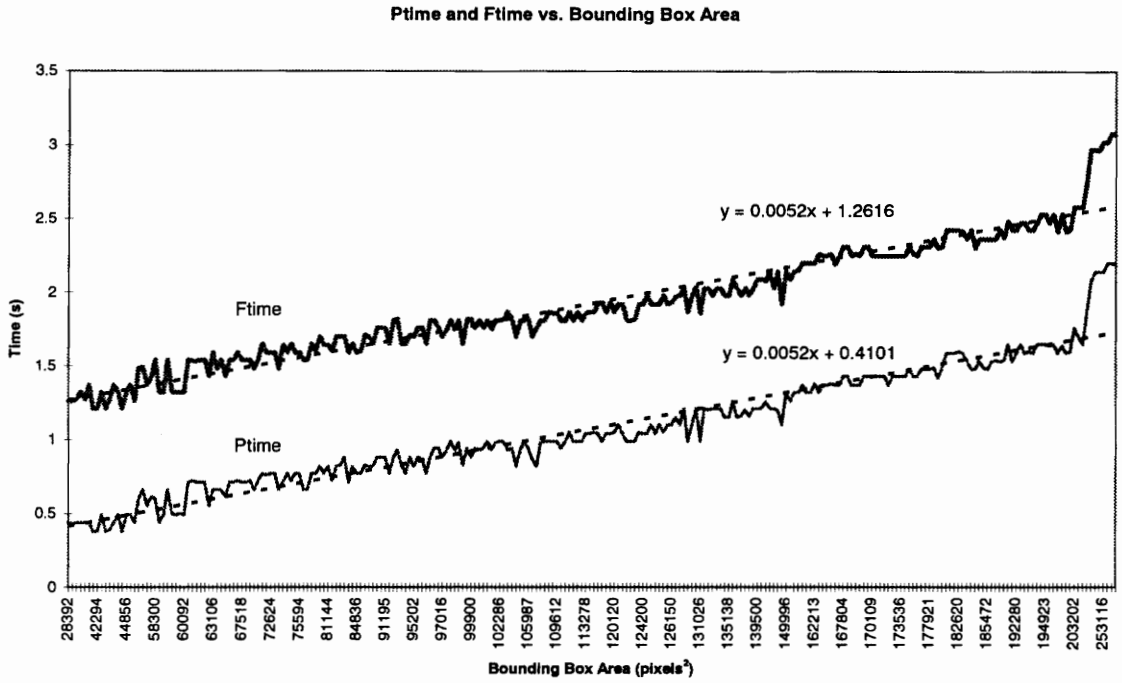


Figure 5.1: Graph of *PTIME* and *FTIME* vs. bounding box area. Datasets are shown as continuous lines for clarity. *PTIME* is proportional to the bounding box area in pixels. *FTIME* is a function of *PTIME*. The trend lines for the two times are designated by the dashed lines. Equations for the linear regressions of the trend lines are also listed above.

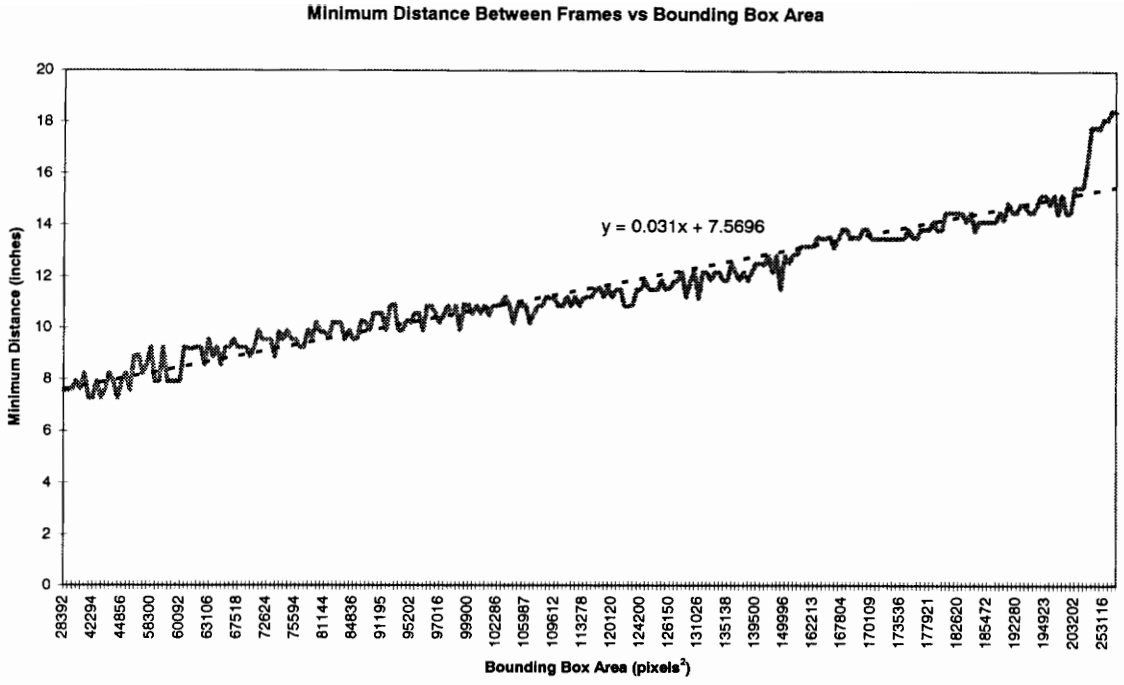


Figure 5.2: Graph of minimum distance vs. bounding box area. The dataset is shown as a continuous line for clarity. As in Figure 5.1, the dashed trend line and its linear regression equation are superimposed on the graph.

5.2 Results of Factory Test

In addition to the laboratory test, the frame recognition system was set up in the factory and tested. The primary difference between the laboratory system and the factory system is that in the factory, the conveyor belt moves at 10 inches per second and is twice as wide, permitting the recognition of extremely large frames. At the time this is being written, 6,600 frames samples have been tested from the entire frame product line at nearly aligned orientations. (As mentioned in Section 1.6, the factory is primarily concerned with aligned frames, but unaligned frame support was added for error resistance.) In addition, several defective frames were passed through the system, in an attempt to cause misclassifications. These defects included mislabeling as well as physical damage. Over 99% of the samples classified correctly, and the remaining ones were classified as unknown or mislabeled due the exceptions and defective frames being included in the test. The defective frames were identified properly, or labeled unknown,

unless they exactly matched the structure of another frame being produced. Since the exceptions make up a small minority of the factory's annual production, the support for these exceptions is a target of future research.

6. Conclusion

This thesis has described a machine vision based system which can identify planar rectangular wooden cabinet frames. The system has been tested successfully in the laboratory and in the factory.

The system is divided into two nearly independent subsystems: the acquisition subsystem, and the recognition subsystem. The acquisition subsystem's primary task is to acquire the silhouette image of a frame, using infrared sensors to start and stop acquisition. A linescan camera is used to build the image of the silhouetted frame line by line, using back light illumination. Through the use of a back light, many typical problems in machine vision applications are avoided. Furthermore, with careful control of camera timing, an inexpensive back light such a long fluorescent tube can be used. The acquisition subsystem has been designed to be cost-efficient and reliable.

Once the image is completely acquired, it is the task of the recognition subsystem to extract the features that uniquely identify the scanned frame. Several different techniques were explored to extract a number of different features. All were found to have significant disadvantages, and individually, were not usable in the system.

A recognition algorithm was developed that efficiently combines the benefits of all the techniques explored. This algorithm, based on the structural decomposition of a frame, subdivides the frame image into smaller and smaller geometric and conceptual primitives. The algorithm is configured as a network of modular nodes, with each performing a different operation on the image. Additionally, the nodes were carefully ordered so that the most complex operation was performed on the smallest image area possible. To conserve memory and promote algorithmic efficiency, only the necessary information is passed from one node to the next for processing. Furthermore, when an

error occurs, the algorithm's modular design permits the recognition subsystem to immediately halt, returning an meaningful error message from the offending node. This is in contrast to waiting until recognition completes before announcing that an error has occurred partway through.

Since the dimensions of each frame can be obtained if the corner vertices of each opening are known, the algorithm's primary task is to extract the corners from each frame opening. This is performed by a novel technique that initially approximates the regions of the opening borders that contain a corner. Once approximated, the corner vertices for each are interpolated based on the curvature of these corner regions. These corners, in turn, are then used to generate the dimensions of the opening. Each opening, including the outside border, is processed completely before the next is started.

Also, a novel technique for a database search is also presented. The dimensions of the scanned frame are compiled into a feature matrix and compared to a database containing the feature matrices of the entire frame product line. The search method identifies the scanned frame in two steps, ending with a comparison to each of the frames in a subset of the database using a multilevel mean-squared-error heuristic. By rotating the frames in the database by 90°, 180°, and 270° and including them in the database as well, the frames can be scanned and recognized at virtually any orientation.

As the factory's product line increases and changes, so can the recognition system. New frames can be supported by the system with very little effort. Also, the system can be adapted to support shapes having more than four sides with extreme angled corners. This system satisfies the requirements of the factory so that they may increase production.

Bibliography

[ABU84] Abu-Mostafa, Y.S., and D. Psaltis. "Recognitive Aspects of Moment Invariants." *IEEE Trans. Pattern Anal. and Machine Intell.* 6 (1984): 698-706.

[BAR94] Bari, Farooq. "A Machine Vision System for Classifying Rectangular Cabinet Frames." Master's Thesis, Virginia Polytechnic Institute and State University, 1994.

[BAU92] Baugher, Sam, and Azriel Rosenfeld. "Corner Detection and Localization in a Pyramid." *Computer Vision and Image Processing.* Boston: Academic Press, Inc., 1992. 103-121.

[BEU87] Beus, H. L., and S. S. H. Tiu. "An Improved Corner Detection Algorithm Based on Chain-Coded Plane Curves." *Pattern Recognition* 20 (1987): 291-296.

[BIT94] Bitflow, Inc. *Raptor Series Installation and User's Manual.* Massachusetts: Bitflow, 1994.

[BON76] Bondy, J. A., and U. S. R. Murty. *Graph Theory with Applications.* New York: North-Holland, 1976.

[BRÜ93] Brühl, Wolfgang. "Standard Machine Vision Systems Used in Different Industrial Applications." *Computer Vision for Industry.* Proc. SPIE 1989 (1993). 300-302.

[CAN86] Canny, J. "A Computational Approach to Edge Detection." *IEEE Trans. Pattern Anal. and Machine Intell.* 8 (1986): 679-698.

[DAE93] Daemi, M. F., H. K. Sardana, and M. K. Ibrahim. "Recognition of Line Patterns Using Moments." *Applications of Digital Image Processing XVI.* Proc. SPIE 2028 (1993). 236-247.

[DAL94] Dalsa, Inc. *CL-Cx Camera User's Manual.* Ontario: Dalsa, 1994.

[DAV90] Davies, E. R. *Machine Vision: Theory, Algorithms, Practicalities.* Microelectronics and Signal Processing Series. London: Academic Press, 1990. 100-167.

[DAV81] Davies, E. R., and A. P. N. Plummer. "Thinning Algorithms: A Critque and a New Methodology." *Pattern Recognition* 14 (1981): 53-63.

[DUD73] Duda, Richard O. and Peter E. Hart. *Pattern Classification and Scene Analysis.* New York: John Wiley and Sons, 1973. 271-272, 328-339.

[FRE77] Freeman, H., and L. S. Davis. "A Corner Finding Algorithm for Chain-Coded Curves." *IEEE Trans. Comput.* 26 (1977): 267-278.

[FRE61] Freeman, H. "On the Encoding of Arbitrary Geometric Configurations." *IRE Transactions on Electronic Computers* 10 (1961): 260-268.

[FRE74] Freeman, H. "Computer Processing of Line-Drawing Images." *Computing Surveys* 6 (1974): 57-97.

[GAL90] Galbriati, Louis J., Jr. *Machine Vision and Digital Image Processing Fundamentals.* New Jersey: Prentice-Hall, 1990. 20-30.

[GIA88] Giardina, Charles R., and Edward R. Dougherty. *Morphological Methods in Image and Signal Processing.* New Jersey: Prentice Hall, 1988. 116-125.

[GIL93] Gilliot, Jean-Marc, and Jean-Louis Amat. "Artificial Intelligence for Networks Recognition in Remote Sensing Images." *Computer Vision for Industry.* Proc. SPIE 1989 (1993). 275-286.

[HAR93a] Haralick, Robert M., and Linda G. Shapiro. *Computer and Robot Vision, Volume II.* Massachusetts: Addison-Wesley, 1993. 427-430.

[HAR93b] Haralick, Robert M., and Linda G. Shapiro. *Computer and Robot Vision, Volume I.* Massachusetts: Addison-Wesley, 1993. 28-48, 337-354.

[HOR94] Hori, Osamu, and Satohide Tanigawa. "Line Fitting Method for Line Drawings Based on Contours and Skeletons." *IEICE Trans., Inf. and Syst.* E77-D.7 (1994): 743-748.

[HU62] Hu, M. K. "Visual Pattern Recognition by Moment Invariants." *IRE Transactions on Information Theory* 8 (1962): 179-187.

[IMA92] Imaging Technologies, Inc. *AFG Hardware Reference Manual.* Massachusetts: Imaging Technologies, Inc., 1992.

[INT69] *Interface Between Data Terminal Equipment and Data Communications Equipment Employing Serial Binary Data Interchange.* Washington, D.C.: Electronic Industries Association, 1969.

[KOP92] Koplowitz, Jack, and Stephen Plante. "Corner Detection for Chain Coded Curves." *Vision Geometry.* Proc. SPIE 1832 (1992). 23-33.

[LIN94] Lindeberg, Tony. *Scale-Space Theory in Computer Vision.* Boston: Kluwer Academic Publishers, 1994. 328-339.

[PAR94] Parker, J. R. *Practical Computer Vision Using C.* New York: John Wiley and Sons, Inc., 1994.

[PIT93] Pitas, Ioannis. *Digital Image Processing Algorithms.* Prentice Hall International Series in Acoustics, Speech, and Signal Processing. New York: Prentice Hall, 1993.

[PRA78] Pratt, William K. *Digital Image Processing.* New York: John Wiley and Sons, 1978. 471-525.

[PRE70] Prewitt, J. M. S. "Object Enhancement and Extraction." *Picture Processing and Psychopictorics.* Eds. B. Lipkin and A. Rosenfeld. New York: Academic Press, 1970. 75-149.

[PUG83] Pugh, Alan, ed. *Robot Vision.* International Trends in Manufacturing Technology. Berlin: Springer-Verlag, 1983. 4-6.

[REI93] Reiss, Thomas H. *Recognizing Planar Object Using Invariant Image Features.* Lecture Notes in Computer Science. Berlin: Springer-Verlag, 1993. 1-22, 61.

[ROB84] Roberts, Fred S. *Applied Combinatorics.* New Jersey: Prentice-Hall, 1984. 80-102.

[ROB65] Roberts, L. G. "Machine Perception of Three-Dimensional Solids." *Optical and Electrooptical Information Processing.* Eds. J. Tippet, et al. Massachusetts: MIT Press, 1965. 159-197.

[ROS73] Rosenfeld, A., and E. Johnson. "Angle Detection on Digital Curves." *IEEE Trans. Comput.* 22 (1973): 875-878.

[ROS75] Rosenfeld, A., and J. S. Weszka. "An Improved Method of Angled Detection on Digital Curves." *IEEE Trans. Comput.* 24 (1975): 940-941.

[TAM78] Tammura, H. "A Comparison of Line Thinning Algorithms from a Digital Geometry Point of View." *Proc. 4th Int. Conf. on Pattern Recognition.* Kyoto, 1978. 715-719.

[WAY93] Norton-Wayne, L., M. Bradshaw, and C. Sanby. "Machine Vision for the Automated Inspection of Web Materials." *Computer Vision for Industry.* Proc. SPIE 1989 (1993). 2-13.

[YUA94] Yuan, Bin. "Automatic Classification of Wooden Cabinet Doors Using Computer Vision." Master's Thesis, Virginia Polytechnic Institute and State University, 1994.

[ZHA84] Zhang, T. Y., and C. Y. Suen. "A Fast Parallel Algorithm for Thinning Digital Patterns." Communications of the ACM 27.3 (1984): 236-239.

Appendix A: Laboratory Test Result Data

This appendix lists in tabular form the results of the laboratory test of the frame recognition system. All test samples were identified correctly, with 39 frames tested between 4 and 7 times each at various orientations. In addition, the AGxx series of exception frames were included in the test, which are further detailed in Appendix B. Each row in the table is for one test sample, where the information in a row is in the order listed below.

- Style name of test sample as reported by frame recognition system
- Number of borders
- *MSE*, or mean-squared-error, between the scanned frame and the frame style listed
- *BBR*, the row-size of the bounding box, in pixels
- *BBC*, the column-size of the bounding box, in pixels
- The *PTIME* for the scanned frame
- The *FTIME* for the scanned frame
- *BBAREA*, the bounding box area in pixels^2, *BBAREA=BBR*BBC*
- *FDIST*, the minimum distance separating this frame and the following frame on the conveyor. Equivalent to *D* in Section 5.1.

For additional information on each of these items, and an analysis of the results, consult Chapter 5.

Frame Name	# Borders	MSE	BBR	BBC	PTIME	FTIME	BBAREA	FDIST
AG2418	1	0.0000194	161	269	0.380	1.210	43309	7.260
AG2418	1	0.0101073	163	272	0.380	1.210	44336	7.260
AG2418	1	0.0111908	161	270	0.390	1.270	43470	7.620
AG2418	1	0.013433	159	265	0.380	1.210	42135	7.260
AG2418	1	0.0305418	159	266	0.380	1.210	42294	7.260
AG2418	1	0.0580711	179	298	0.440	1.260	53342	7.560
AGS24	1	0.0038824	164	362	0.490	1.320	59368	7.920
AGS24	1	0.0044238	165	363	0.490	1.320	59895	7.920
AGS24	1	0.0170514	163	362	0.440	1.320	59006	7.920
AGS24	1	0.0182389	165	364	0.500	1.320	60060	7.920
AGS24	1	0.0182389	165	362	0.500	1.320	59730	7.920
AGS24	1	0.0396543	166	362	0.490	1.320	60092	7.920
B12	3	0.018956	279	242	0.710	1.540	67518	9.240
B12	3	0.0328899	273	196	0.600	1.480	53508	8.880
B12	3	0.0356003	285	259	0.720	1.650	73815	9.900
B12	3	0.0455313	274	217	0.660	1.540	59458	9.240
B12	3	0.0496325	278	227	0.660	1.590	63106	9.540
B12	3	0.0942214	287	255	0.710	1.640	73185	9.840
B12	3	0.106966	280	220	0.710	1.530	61600	9.180
B27	4	0.0736047	285	442	1.050	1.980	125970	11.880
B27	4	0.0853304	268	409	0.990	1.860	109612	11.160
B27	4	0.1263765	278	425	1.040	1.870	118150	11.220
B27	4	0.149652	270	412	0.990	1.810	111240	10.860
B27	4	0.1537029	285	442	1.100	1.920	125970	11.520
B27	4	0.1903395	290	435	1.100	1.930	126150	11.580
B27	4	0.2104151	280	429	1.050	1.870	120120	11.220
B45	7	0.066533	398	459	1.590	2.410	182682	14.460
B45	7	0.0727925	397	460	1.600	2.420	182620	14.520
B45	7	0.0736958	396	459	1.590	2.420	181764	14.520
B45	7	0.1273783	399	478	1.650	2.480	190722	14.880
B45	7	0.1344522	405	488	1.640	2.530	197640	15.180
B45	7	0.1399177	395	462	1.590	2.420	182490	14.520
B45	7	0.1434928	394	459	1.590	2.420	180846	14.520

B48	5	0.1013433	420	460	1.600	2.420	193200	14.520
B48	5	0.1341316	420	460	1.590	2.420	193200	14.520
B48	5	0.1597874	422	462	1.640	2.470	194964	14.820
B48	5	0.1721423	418	460	1.590	2.420	192280	14.520
B48	5	0.1741839	419	459	1.590	2.470	192321	14.820
B48	5	0.1805749	429	485	1.700	2.580	208065	15.480
BLB18	2	0.0184847	291	341	0.830	1.650	99231	9.900
BLB18	2	0.0294395	279	306	0.770	1.600	85374	9.600
BLB18	2	0.0409419	272	283	0.660	1.540	76976	9.240
BLB18	2	0.0410014	272	281	0.660	1.540	76432	9.240
BLB39/42	5	0.0467716	310	494	1.320	2.140	153140	12.840
BLB39/42	5	0.0799619	292	459	1.210	2.030	134028	12.180
BLB39/42	5	0.1129494	269	495	1.210	2.030	133155	12.180
BLB39/42	5	0.155542	278	504	1.260	2.080	140112	12.480
BLB39/42	5	0.1593316	299	472	1.210	2.140	141128	12.840
BLB39/42	5	0.160939	308	487	1.320	2.140	149996	12.840
BLW24/2730	3	0.0439744	268	362	0.940	1.760	97016	10.560
BLW24/2730	3	0.0554075	269	374	0.940	1.760	100606	10.560
BLW24/2730	3	0.0558304	293	429	1.100	1.920	125697	11.520
BLW24/2730	3	0.0687695	286	415	1.040	1.930	118690	11.580
BLW24/2730	3	0.0705278	269	372	0.940	1.760	100068	10.560
BLW24/2730	3	0.0805821	285	410	0.990	1.930	116850	11.580
BLW24/2730	3	0.1253145	298	436	1.100	1.980	129928	11.880
BLW36/3936	3	0.2131778	318	540	1.430	2.250	171720	13.500
BLW36/3936	3	0.2192145	312	545	1.430	2.310	170040	13.860
BLW36/3936	3	0.2768374	317	542	1.370	2.250	171814	13.500
BLW36/3936	3	0.3049077	318	548	1.430	2.250	174264	13.500
BLW36/3936	3	0.3092729	322	551	1.480	2.310	177422	13.860
BLW36/3936	3	0.318629	320	548	1.430	2.260	175360	13.560
BLW42/2742	3	0.0688605	373	360	1.150	1.980	134280	11.880
BLW42/2742	3	0.0713228	375	365	1.200	2.030	136875	12.180
BLW42/2742	3	0.0736284	375	373	1.210	2.090	139875	12.540
BLW42/2742	3	0.0762712	389	422	1.320	2.260	164158	13.560
BLW42/2742	3	0.078136	387	398	1.320	2.150	154026	12.900
BLW42/2742	3	0.0820495	375	372	1.210	2.090	139500	12.540
BLW42/2742	3	0.0866658	388	411	1.320	2.200	159468	13.200

BLW42/2742	3	0.0875659	389	417	1.380	2.200	162213	13.200
BLW42/2742	3	0.0929092	376	370	1.210	2.030	139120	12.180
BLW42/2742	3	0.0940025	374	364	1.150	2.030	136136	12.180
BLW42/2742	3	0.1167173	380	399	1.260	2.090	151620	12.540
BLW42/2742	3	0.3101752	392	411	1.320	2.200	161112	13.200
BLW42/4530	4	0.0762639	372	458	1.430	2.250	170376	13.500
BLW42/4530	4	0.0912257	370	461	1.430	2.250	170570	13.500
BLW42/4530	4	0.1046749	368	457	1.370	2.260	168176	13.560
BLW42/4530	4	0.1088565	366	455	1.370	2.190	166530	13.140
BLW42/4530	4	0.1209905	369	457	1.370	2.250	168633	13.500
BLW42/4530	4	0.1215225	369	454	1.370	2.250	167526	13.500
CSF36	3	0.1399375	269	270	0.770	1.590	72630	9.540
CSF36	3	0.1618887	278	303	0.820	1.650	84234	9.900
CSF36	3	0.1766469	287	330	0.880	1.710	94710	10.260
CSF36	3	0.1897955	276	294	0.820	1.640	81144	9.840
CSF36	3	0.2045636	295	354	0.930	1.810	104430	10.860
CSF36	3	0.2154813	267	272	0.770	1.590	72624	9.540
CW2430	2	0.0240991	283	324	0.770	1.650	91692	9.900
CW2430	2	0.0286883	275	296	0.720	1.600	81400	9.600
CW2430	2	0.0356138	287	335	0.770	1.650	96145	9.900
CW2436	2	0.0650153	325	291	0.770	1.650	94575	9.900
CW2436	2	0.1173204	315	267	0.710	1.590	84105	9.540
CW2436	2	0.1239426	337	362	0.990	1.810	121994	10.860
CW2436	2	0.1270868	327	315	0.820	1.700	103005	10.200
CW2436	2	0.1394119	331	334	0.940	1.810	110554	10.860
CW2436	2	0.1458302	343	382	0.990	1.860	131026	11.160
CW2436	2	0.2314049	337	361	0.990	1.810	121657	10.860
DB12	4	0.0293885	283	273	0.770	1.600	77259	9.600
DB12	4	0.0369002	285	286	0.820	1.700	81510	10.200
DB12	4	0.0439736	277	231	0.660	1.480	63987	8.880
DB12	4	0.0486945	275	212	0.600	1.540	58300	9.240
DB12	4	0.0543428	287	306	0.820	1.650	87822	9.900
DB12	4	0.0633566	282	251	0.720	1.540	70782	9.240
TSF24	3	0.0493959	270	370	0.880	1.820	99900	10.920
TSF24	3	0.0582142	280	390	0.990	1.860	109200	11.160
TSF24	3	0.0620357	303	446	1.150	1.980	135138	11.880

TSF24	3	0.0667985	271	368	0.940	1.820	99728	10.920
TSF24	3	0.0868048	288	402	1.050	1.870	115776	11.220
TSF24	3	0.1087979	266	361	0.880	1.760	96026	10.560
TSF24	3	0.1413201	270	366	0.930	1.750	98820	10.500
TSF30	4	0.0546667	266	462	1.050	1.920	122892	11.520
TSF30	4	0.0614814	268	466	1.100	1.920	124888	11.520
TSF30	4	0.1241467	274	458	1.040	1.920	125492	11.520
TSF30	4	0.1386282	267	455	1.040	1.920	121485	11.520
TSF30	4	0.1630924	270	460	1.040	1.920	124200	11.520
TSF30	4	0.1765145	274	465	1.100	1.980	127410	11.880
TSF42	5	0.0594882	373	478	1.490	2.360	178294	14.160
TSF42	5	0.0679748	376	487	1.480	2.420	183112	14.520
TSF42	5	0.1140158	373	477	1.480	2.310	177921	13.860
TSF42	5	0.1270452	366	461	1.430	2.310	168726	13.860
TSF42	5	0.1289205	364	461	1.430	2.310	167804	13.860
TSF42	5	0.1367674	364	461	1.430	2.310	167804	13.860
TSF42	5	0.1381652	370	471	1.480	2.310	174270	13.860
TSF48	7	0.0766944	421	463	1.650	2.530	194923	15.180
TSF48	7	0.0775248	421	462	1.650	2.530	194502	15.180
TSF48	7	0.0809583	419	459	1.650	2.470	192321	14.820
TSF48	7	0.1235382	421	465	1.650	2.530	195765	15.180
TSF48	7	0.1255739	426	477	1.760	2.580	203202	15.480
TSF48	7	0.1326818	442	524	1.870	2.740	231608	16.440
TSF48	7	0.1518101	422	460	1.650	2.470	194120	14.820
UB60	9	0.0878505	529	465	2.140	2.960	245985	17.760
UB60	9	0.0911938	536	488	2.200	3.020	261568	18.120
UB60	9	0.1236177	530	463	2.140	2.970	245390	17.820
UB60	9	0.126771	540	494	2.190	3.070	266760	18.420
UB60	9	0.1552941	538	488	2.200	3.080	262544	18.480
UB60	9	0.1767892	531	459	2.080	2.970	243729	17.820
UB60	9	0.1794071	534	474	2.140	3.020	253116	18.120
VAN36	7	0.0351516	317	413	1.210	2.040	130921	12.240
VAN36	7	0.043898	336	460	1.370	2.200	154560	13.200
VAN36	7	0.0452333	319	415	1.210	2.030	132385	12.180
VAN36	7	0.0580247	324	419	1.260	2.090	135756	12.540
VAN36	7	0.1004916	318	407	1.210	2.030	129426	12.180

VAN36	7	0.1007332	320	415	1.210	1.980	132800	11.880
VDB15	4	0.0303893	258	293	0.770	1.590	75594	9.540
VDB15	4	0.0308203	260	306	0.770	1.640	79560	9.840
VDB15	4	0.0327546	240	228	0.660	1.490	54720	8.940
VDB15	4	0.0376392	249	258	0.660	1.540	64242	9.240
VDB15	4	0.0519203	254	269	0.720	1.540	68326	9.240
VDB15	4	0.0620634	254	268	0.710	1.540	68072	9.240
VKD36	2	0.0274525	314	98	0.430	1.260	30772	7.560
VKD36	2	0.0511762	315	105	0.440	1.270	33075	7.620
VKD36	2	0.0566851	316	178	0.550	1.370	56248	8.220
VKD36	2	0.0597089	312	91	0.440	1.260	28392	7.560
VKD36	2	0.0833363	313	106	0.440	1.320	33178	7.920
VKD36	2	0.0889944	315	200	0.550	1.430	63000	8.580
VSDB24L	6	0.0465505	259	395	0.990	1.820	102305	10.920
VSDB24L	6	0.0646287	258	397	0.930	1.870	102426	11.220
VSDB24L	6	0.0666982	282	451	1.150	1.970	127182	11.820
VSDB24L	6	0.0740993	264	413	0.990	1.870	109032	11.220
VSDB24L	6	0.0750809	252	386	0.930	1.750	97272	10.500
VSDB24L	6	0.080473	257	398	0.990	1.810	102286	10.860
VSDB24L	6	0.1205319	259	388	0.930	1.810	100492	10.860
VSDB24R	6	0.0603998	281	444	1.040	1.980	124764	11.880
VSDB24R	6	0.0615908	256	394	0.990	1.810	100864	10.860
VSDB24R	6	0.0693704	218	418	0.880	1.760	91124	10.560
VSDB24R	6	0.0718028	247	377	0.880	1.810	93119	10.860
VSDB24R	6	0.075866	259	392	0.980	1.810	101528	10.860
VSDB24R	6	0.0824732	271	418	1.040	1.860	113278	11.160
VSDB24R	6	0.1022737	261	393	0.940	1.820	102573	10.920
W1236	2	0.0575014	332	322	0.870	1.700	106904	10.200
W1236	2	0.0709506	321	218	0.660	1.480	69978	8.880
W1236	2	0.1110545	329	222	0.660	1.480	73038	8.880
W1236	2	0.1230251	322	180	0.610	1.420	57960	8.520
W1236	2	0.1587169	338	317	0.820	1.760	107146	10.560
W1236	2	0.2948088	334	254	0.770	1.590	84836	9.540
W1242	2	0.0486456	379	247	0.830	1.650	93613	9.900
W1242	2	0.0744841	383	319	0.990	1.820	122177	10.920
W1242	2	0.0980167	385	384	1.100	1.920	147840	11.520

W1242	2	0.0991635	385	387	0.990	1.870	129745	11.220
W1242	2	0.4209417	390	377	1.200	2.140	147030	12.840
W1242	2	0.5463757	387	246	0.820	1.700	95202	10.200
W2718_SPC	3	0.0159947	240	278	0.710	1.540	66720	9.240
W2718_SPC	3	0.0203781	240	280	0.720	1.590	67200	9.540
W2718_SPC	3	0.0547511	269	368	0.980	1.810	98992	10.860
W2718_SPC	3	0.0616421	268	362	0.880	1.700	97016	10.200
W2718_SPC	3	0.0659169	264	354	0.930	1.820	93456	10.920
W2718_SPC	3	0.071009	270	362	0.990	1.810	97740	10.860
W3318	3	0.0424957	312	358	0.990	1.810	111696	10.860
W3318	3	0.0426594	309	343	0.940	1.820	105987	10.920
W3318	3	0.047435	304	316	0.870	1.760	96064	10.560
W3318	3	0.0495588	294	291	0.830	1.710	85554	10.260
W3318	3	0.0602986	316	364	1.040	1.870	115024	11.220
W3318	3	0.0659836	299	305	0.880	1.760	91195	10.560
W3318	3	0.0753897	291	272	0.820	1.700	79152	10.200
W3324	3	0.0707072	288	367	0.990	1.810	105696	10.860
W3324	3	0.087258	316	431	1.160	1.980	136196	11.880
W3324	3	0.0877251	291	370	0.990	1.810	107670	10.860
W3324	3	0.0999478	320	444	1.210	2.030	142080	12.180
W3324	3	0.1022753	303	398	1.100	1.920	120594	11.520
W3324	3	0.1167931	299	390	1.040	1.920	116610	11.520
W3324	3	0.1639603	298	374	1.050	1.870	111452	11.220
W3342	3	0.194027	372	499	1.530	2.360	185628	14.160
W3342	3	0.2330556	369	499	1.490	2.360	184131	14.160
W3342	3	0.2351428	376	509	1.530	2.420	191384	14.520
W3342	3	0.2774896	368	504	1.480	2.360	185472	14.160
W3342	3	0.3160961	372	506	1.540	2.360	188232	14.160
W3342	3	0.3183655	367	499	1.480	2.300	183133	13.800
W3612	3	0.0412988	333	303	0.930	1.750	100899	10.500
W3612	3	0.0487687	318	191	0.720	1.530	60738	9.180
W3612	3	0.0560808	325	237	0.770	1.650	77025	9.900
W3612	3	0.0635716	319	190	0.710	1.540	60610	9.240
W3612	3	0.0675229	320	193	0.710	1.540	61760	9.240
W3612	3	0.0682577	321	207	0.710	1.540	66447	9.240
W3612	3	0.1073219	321	192	0.710	1.540	61632	9.240

W3615	3	0.0148578	316	234	0.770	1.590	73944	9.540
W3615	3	0.0420516	314	230	0.770	1.650	72220	9.900
W3615	3	0.0425264	324	267	0.820	1.700	86508	10.200
W3615	3	0.0439185	316	229	0.760	1.590	72364	9.540
W3615	3	0.0466007	321	254	0.830	1.700	81534	10.200
W3615	3	0.081438	318	231	0.770	1.590	73458	9.540
W3615	3	0.1335921	346	383	1.200	2.030	132518	12.180
W3636	3	0.1909577	319	543	1.430	2.250	173217	13.500
W3636	3	0.1935738	319	544	1.430	2.250	173536	13.500
W3636	3	0.1952881	321	552	1.480	2.310	177192	13.860
W3636	3	0.196032	343	587	1.600	2.420	201341	14.520
W3636	3	0.2381339	321	548	1.430	2.250	175908	13.500
W3636	3	0.2560094	327	559	1.540	2.360	182793	14.160
W3915_3VT	3	0.0265632	351	276	0.940	1.810	96876	10.860
W3915_3VT	3	0.0308928	344	243	0.880	1.700	83592	10.200
W3915_3VT	3	0.0486481	356	301	0.980	1.810	107156	10.860
W3915_3VT	3	0.0721627	349	259	0.880	1.760	90391	10.560
W3915_3VT	3	0.0846239	365	378	1.160	1.980	137970	11.880
W3915_3VT	3	0.0921133	359	313	0.990	1.870	112367	11.220
W3915_3VT	3	0.1075403	352	275	0.880	1.810	96800	10.860
W3915_3VT	3	0.1154465	358	314	0.990	1.810	112412	10.860
W4230	3	0.1195539	369	461	1.430	2.250	170109	13.500
W4230	3	0.1196011	375	476	1.420	2.300	178500	13.800
W4230	3	0.1218357	365	455	1.380	2.260	166075	13.560
W4230	3	0.1300629	370	466	1.430	2.250	172420	13.500
W4230	3	0.1302731	366	459	1.370	2.250	167994	13.500
W4230	3	0.141762	363	453	1.370	2.250	164439	13.500
W4230	3	0.1449255	364	455	1.370	2.250	165620	13.500
W4530	4	0.0974749	396	463	1.540	2.360	183348	14.160
W4530	4	0.1000682	408	491	1.590	2.410	200328	14.460
W4530	4	0.1093002	400	472	1.530	2.420	188800	14.520
W4530	4	0.1105192	401	472	1.540	2.370	189272	14.220
W4530	4	0.1188649	416	513	1.650	2.580	213408	15.480
W4530	4	0.1321493	407	482	1.590	2.410	196174	14.460
W4530	4	0.1353124	396	453	1.490	2.310	179388	13.860
W930	2	0.0137441	275	170	0.490	1.370	46750	8.220

W930	2	0.0177761	274	236	0.610	1.430	64664	8.580
W930	2	0.0288823	273	160	0.440	1.370	43680	8.220
W930	2	0.0308915	270	163	0.490	1.320	44010	7.920
W930	2	0.0369647	273	152	0.440	1.370	41496	8.220
W930	2	0.0407615	267	168	0.490	1.320	44856	7.920
W930	2	0.0480998	267	161	0.490	1.320	42987	7.920
W930	2	0.0492983	262	140	0.440	1.270	36680	7.620
AVERAGE	**3.69**	**0.10202**	**321.18**	**371.65**	**1.075**	**1.926**	**122388.67**	**11.55**
MAX	**9.00**	**0.54638**	**540.00**	**587.00**	**2.200**	**3.080**	**266760.00**	**18.48**
MIN	**1.00**	**0.00002**	**159.00**	**91.00**	**0.380**	**1.210**	**28392.00**	**7.26**
STD. DEV.	**1.74**	**0.07369**	**69.11**	**106.20**	**0.395**	**0.395**	**51884.50**	**2.37**

Appendix B: AGxx Frame Series Support

Section 1.4 lists several unusual frame styles that do not fit the general specifications of a frame. These include frames with unusual crossbar designs, and frames that are not based on closed rectangles. In other words, these frame styles cannot be easily specified by a λ-matrix. For these frame styles to be properly recognized by the frame recognition system, they must be implemented as exceptions to the regular algorithm, requiring special treatment. This appendix discusses the additions to the algorithm necessary to support the AGxx series of frame styles. The algorithm described here can be extended to any multisided polygon with discernable corners, limited only by camera resolution.

The AGxx series consists of two different frame styles, AGS24 and AG2418. Both of these frames are shown below in Figure B-1.

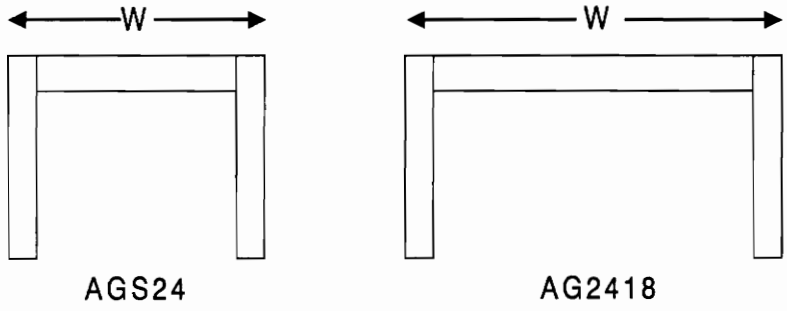


Figure B-1: Schematics of frame styles AGS24 and AG2418. Dimension "W" is the primary difference between the two frame styles.

The frames are practically identical, both being 3 sides of a rectangle. The primary difference between them is that AG2418 has a "*W*" dimension of 24 inches, and AGS24 has a "*W*" dimension of 18 inches.

With a few simple additions to the current frame recognition algorithm, it is possible to include support for the AGxx series. The original algorithm proceeds without error through the chain code generation node, since the AGxx series' outside borders are closed shapes. When corner regions are extracted, however, several will be missed by the current algorithm. Right-handed corner regions are the only type of regions detected by the corner region node, depicted in the chain code by clusters of -1s. Figure B-2(a) illustrates the right-hand corners of an AGxx series frame.

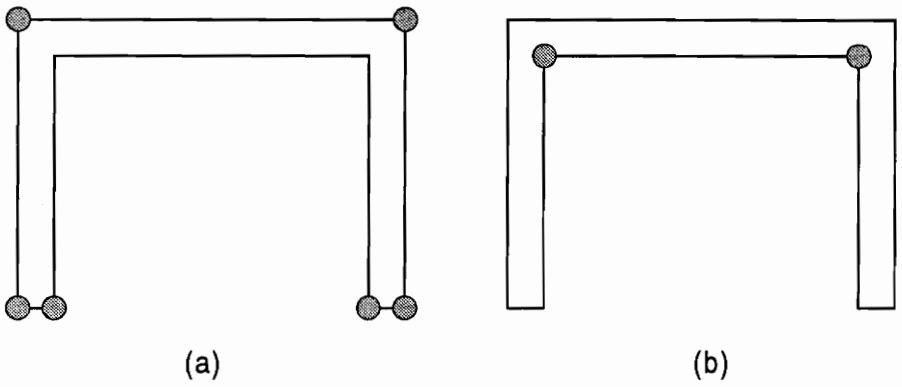


Figure B-2: Examples of right-hand and left-handed corners in AGxx series frames.

However, as can be seen in Figure B-2(b), there are two corners of AGxx frame styles that would not be sensed by the node as is. When the corner region generation node senses that there are six corner regions, rather than four, an alternate form of the corner region algorithm is also performed. This algorithm is practically identical to the node specified in Section 4.4.2. The primary difference is that when regions are extracted from the processed difference chain code, regions are defined by clusters of +1s. In terms of the modifications to the current algorithm, all instances of -1s are interchanged with +1s and visa versa when marking and extraction occurs. Additionally, the number of

marks in these new corner regions are used to filter out some potential misdetections. This new algorithm is designed to be sensitive to the "left-handed" corner regions. If two left-handed corner regions are detected, they are inserted into the list of the right hand corners, ensuring the corners are ordered properly around the border. If not, recognition aborts and an error is reported. Since the corner regions are then isolated, due to the structure of the algorithm, they are all treated equally. Corner vertices are then interpolated from each of the regions as for a normal frame. The dimensions are not generated since there technically is no height and width, in terms of the dimension generation node for the AGxx series frames.

Since each opening of a frame is processed from beginning to end before the next is started, the algorithm will halt at this point for the single border AGxx series frames. If only one border region is detected, with 2 left-hand and 6 right-hand corner regions, the frame is identified as a potential AGxx frame. The side dimensions spanning each of the corner vertices are then generated, and the longest side dimension is compared to the longest dimensions of the AGS24 and the AG2418 frame styles (18" and 24", respectively). The two comparisons are expressed in terms of squared error, and the frame that produces the lowest squared error is a match for the scanned frame. As an additional check, the squared error must be below 0.5, or the system reports an unknown frame style. This is more rigorous than the search node of the frame recognition system to compensate for the fact that only one dimension is checked.

This supplemental algorithm was implemented successfully in both the laboratory and factory systems. All test samples of AGxx series frames were recognized correctly with the above modifications, providing the frames were nearly aligned. The alignment was necessary so that the frames could clear the acquisition plane without falling through the gap to the back light (Figure 2.9).

Appendix C: Supporting Documentation

This appendix contains various documents that are related to the implementation, installation, and usage of the frame recognition system. These documents are listed below.

- Equipment List for Frame Recognition System (2 pages)
- Command Line Parameters for the Frame Recognition System's Executable Modules (4 pages)
- Configuration File Parameters and Syntax (1 page)
- Installation and Calibration Procedure for Frame Recognition System (2 pages)
- Troubleshooting Guide for the Frame Recognition System (2 pages)
- How to Add a New Frame Style to the Database (1 page)

Equipment List for Frame Recognition System

Equipment	Quantity	Description	Manufacturer
Line Scan Camera	1	CL-C4-1024, with multiplexer option and C-mount lens adapter (1K x 1)	Dalsa, Inc. 605 McMurray Road Waterloo, Ontario Canada N2V 2E9 (519)886-6000
Camera Lens	1	C21212 Computar C-mount lens with f:1.8, 12.5mm focal length (TV)	Computar
Power Supply for Line Scan Camera	1	Uniforce PS/IF-2 Power Supply/Interface for DALSA, CCD Cameras	Uniforce Sales & Engineering 536A Valley Way Milpitas, CA 95035 (408)946-3864
Photoelectric Sensor System	-	CP Series 18mm diameter microswitches, +/- 12VDC:	Honeywell, Inc. 11 West Spring St. Freeport, IL 61032 (815)235-6600
	1	CP18RDND2 photoelectric receiver	
	1	CP18ECX2 photoelectric emitter	
	2	CP218B03 swivel bracket	
	2	CP218RAW Right Angle Adapter	
Positional Adjustment Mount	-	Positional Mounts with mounting hardware	Newport Corporation Irvine, CA 800-222-6440
	1	Series 36 (XZ mount with micrometers)	
	1	1D translational mount (with micrometers) compatible with Series 36	
Frame Grabber	1	RAP-VL-VS-U-4 Data Raptor-VL Camera Controller and Frame Grabber w/4MB RAM	Bitflow, Inc. 21-G Olympia Ave #80 Woburn, MA 01801 (617)932-2900
DALSA-Frame Grabber Adapter	1	CAB-DALSAN Cable for DALSA Camera	Bitflow, Inc.
Timing Generator	1	DT2819 Timing Board w/PACER library	Data Translation, Inc. 100 Locke Drive Marlboro, MA 01752-1192 (508)481-3700
Computer (IBM-Compatible)	1	Dell 486/100 16MB RAM VL-Local Bus (See enclosed invoice for specifications) (ADD CDROM)	Dell 2214 W. Braker Lane, Bld 3 Austin, TX 78758-4063 (512)338-4400
Software	1	Watcom C/C++ v10a CDROM (optional: Paper Manuals)	Watcom, Inc. 415 Phillip St. Waterloo, Ontario Canada N2L 3X2 (519)747-4971

Software (continued)	1	SW-BASEDEV Raptor Development Kit v1.2a	Bitflow, Inc.
	1	DOS v6.2 (or later)	Microsoft
	1	MS Windows v3.1	Microsoft
Miscellaneous	1	10ft (Maximum) 25pin Male to 25pin Female Quad Insulated Serial Cable	General
	1	8ft Fluorescent Light Source w/wall plug (60W)	General
	-	Video Cable with BNC Connectors (with barrel)	General
	-	Camera/Light Mountings/Structure	General
	1	Power Strip	General
	1	Timing Board-to-PC Breakout Board	CUSTOM
	1	Uniforce Pwr Supply-to-Raptor Adapter	CUSTOM
	1	Sensor-to-Serial Port Conversion Circuit	CUSTOM

Command Line Parameters for the Frame Recognition System's Executable Modules

USAGE NOTES:

Parameters in brackets ([...]) are optional, items in carets (< ...>) are required
Items in *italics* are variables, items in **bold** are to be typed "as is" or critical information.
Underlined text is critical information or notations.

setfclk (SET Frequency CLocK)

Sets the clock frequency of the Data Translation's DT2819 FCLK pin. Used internally by the other programs in the suite.

USAGE:

setfclk [*clockfreq*]

INPUT:

clockfreq, the desired clock frequency, in Hz. If clock frequency parameter not selected, defaults to 55.56 Hz. For 30ft/min speed should be 55.56 Hz, and for 50 ft/min. should be 92.6 Hz.

OUTPUT:

FCLK set to closest frequency to *clockfreq* that can be output by the DT2819 Timer Board FCLK. Also outputs frequency information on screen.

raptest (RAPtor TEST)

Displays raw waterfall input from linescan camera. First display is without threshold, second is with threshold. Press any key to change from first to second, then again to quit. Used in calibration.

USAGE:

rap30 (use for 30ft/min conveyor belt speed)
rap50 (use for 50ft/min conveyor belt speed)

INPUT:

No command line parameters.

OUTPUT:

Graphical output on screen. Be sure to select 1024x768 graphics.

makedb (MAKE DataBase)

Compiles the *vtf_?.dat* database files into *vtf_?_rt.dat* files, where the *?* is the number of internal openings in the frames. Compresses the database and rotates each frame through 360 degrees. Must be in the same directory with the database files and the files *VTFCJD*, *LINEC*, and *CVRROT*.

USAGE:

makedb

INPUT:

vtf_?.dat files. A single execution processes all of the *vtf_?.dat* files

OUTPUT:

vtf_?_rt.dat files (Used by framesys). Reports status and error messages to screen

frcalib (Frame CALIBrate)

Displays (and optionally saves) thinned-edge silhouette of frame being scanned. Also tests sensor control of acquisition. Can also display saved *.RAW images from framesys and frcalib. A "Q" or "q" quits the program when the message "Waiting for Frame" is displayed. Used in calibration.

USAGE:

frcal30 [-SHOWIM *fname.RAW*] [-DUMPIM] [-SLACK *svalue*] [-?]
(use for 30ft/min conveyor belt speed)

frcal50 [-SHOWIM *fname.RAW*] [-DUMPIM] [-SLACK *svalue*] [-?]
(use for 50ft/min conveyor belt speed)

INPUT:

-? displays the usage of *FRCALIB*

-SHOWIM *fname.RAW* will display the image contained in *fname.RAW*

-DUMPIM saves the current scanned frame into *calib.raw*

-SLACK *svalue* will adjust the scan slack (the number of lines scanned after the frame clears the sensor pair) where svalue is the number of lines scanned divided by 4. *svalue* defaults to 11 if the slack parameter is not specified.

OUTPUT:

Graphical output on screen. Be sure to select 1024x768 graphics.

framesys (FRAME SYStem)

The actual frame recognition engine. Processes the linescan image from the camera controlled by the sensors. Produces text output on the screen. *FRAMESYS* is a batch file that sets up the screen properly and disk caching. "Q" or "q" quits the program when the green message "Waiting for Frame..." is displayed.

USAGE:

framesys <-CFG *cfile.cfg*> [-DEBUG] [-GETIMAGES] [-TIMING] [-DUMPDATA] [-?]

INPUT:

-? display the usage of framesys

-CFG *cfile.cfg* loads the specified configuration detailed in *cfile.cfg*. The three configuration files included in the initial release are:

- *vtlab.cfg*, the Virginia Tech Laboratory configuration (experimental)
- *awc30.cfg*, the AWC factory 30ft/min configuration
- *awc50.cfg*, the AWC factory 50ft/min configuration

THIS IS A REQUIRED PARAMETER. See the document "Configuration File Parmeters and Syntax" for more details on the mechanics of this file.

-DEBUG suppresses all screen output, but appends all of the screen output to the text file *debug.out*, along with almost all of the intermediate computations performed by the program. The only messages displayed on the screen are: "Waiting for frame", "Frame is OK", and "Frame is NOT OK", which represents idle state, last frame recognized, and last frame not recognized, respectively. Useful for debugging.

-GETIMAGES saves images of the current frame scanned to the **.raw* files:

- *othresh.raw*: The thresholded image (akin to the *RAPTEST* 2nd screen image)
- *oedge.raw*: Edge detected thresholded image
- *othin.raw*: Thinned version of *oedge.raw* (image is like *calib.raw* from *FRCALIB*)
- o*process.raw*: Chain-coded and region labeled version of *othin.raw*.

An "**I**" is displayed in the upper-left hand corner of the GUI when this parameter is in effect. The program will automatically exit after a key press after an acquisition, so the files can be moved.

-TIMING computes timing benchmarks for the system and displays them on the screen at the bottom of the scrolling data window, and in the *dumpfile.out* if the -DUMPFILE command line parameter is specified. Two times are computed: ptime, which is the raw computation time for the scanned frame (in seconds), and ftime, which is a representation of time from the start of processing to the beginning of acquisition of the next frame (can be used to compute distance between frames) (in seconds). Note that this parameter decreases the sensor resolution in order to maintain timing accuracy, and the system must be rebooted **before** this parameter is used. While this flag is in effect a "**T**" is displayed in the upper-left hand corner of the GUI.

-DUMPDATA appends the text file *dumpdata.out* a line with various information about the frame. Only known frames are written out. Each line in *dumpdata.out* has the following format:

FRNAME MSE BBR BBC PTIME FTIME

Where:

FRNAME is the name of the scanned frame from the database *MSE* is the reported mean-squared-error of the scanned image to the database match

BBR is the row-size of the bounding box of the scanned frame image

BBC is the column-size of the bounding box of the scanned frame image

PTIME and *FTIME* are defined as ptime and ftime, if the -TIMING flag is also specified. Otherwise, these values are each set to 0.0.

While this flag is in effect a "**D**" is displayed in the upper-left hand corner of the GUI.

OUTPUT:

Varies, depending on command line parameters. Text mode should be 80x50.

Configuration File Parameters and Syntax

(used by <-CFG *cfile.cfg*> parameter of FRAMESYS executable)
(all must be included in a raw text file, space delimited, no formatting)

Keyword	Default Settings (if any)	Description
slackval	11	Number of scan lines divided by 4 to be grabbed after the frame clears the sensor beam
maxmsescan	1.0	Maximum MSE value between a scanned frame and a database entry where the system will display the database frame name
maxmsepass	5.0	Maximum MSE value between a scanned frame and a database entry before the processing of that database entry is automatically aborted, and the next is started
maxcornerrange	100	The maximum row and column distance that a reconstructed corner vertex can be from the midpoint of the corner region. **Do not reduce below 50.**
minchainlength	120	The minimum valid length of a chain code. **Do not reduce below 120.**
scalex	---	Inches/pixel in the x-direction for all interior frame openings. **Obtain through calibration.**
scaley	---	Inches/pixel in the y-direction for all interior frame openings. **Obtain through calibration.**
outsidescalefx	---	Scale factor applied to scalex to obtain inches/pixel for outside border of frame in x direction. **Obtain through calibration.**
outsidescalefy	---	Scale factor applied to scaley to obtain inches/pixel for outside border of frame in y direction. **Obtain through calibration.**

Installation and Calibration Procedure For Frame Recognition System

(References to schematics have been changed to reflect figures in this thesis)

(NOTE: BEFORE BEGINNING ENSURE ALL POWER IS OFF)

Additional Equipment Needed for Installation:
Tape Measure
Torpedo Level

Installation Procedure:

1. Install Raptor board and DT2819 Timing Board in PC.
2. Connect Timing Board to Raptor with circuit in Figure 2.7.
3. Install all software in PC (Watcom C, Raptor Development Kit, etc.) and configure the Raptor as per its documentation with the *CLC1KB1.CAM* camera file.
4. Set up the two conveyor belts, ensuring that both are level, straight, and moving at the same speed. This can be tested by running a VAN42 frame over the gap and ensuring that the feed is smooth.
5. Set up the substructure and lighting assembly as specified in the overall schematic (Figure 2.9) over the belts. Use the level to make sure the framework is suspended parallel over the gap.
6. Mount the translation unit on the frame, then mount the tilt-pan unit on it with the micrometers on the left top side of the mounting surface, with that surface facing you. Make sure the mounts are at midrange (no adjustment).
7. Mount the camera on the framework pointing down at gap between the two conveyor belts, as shown in Figure 2.9. Adjust the camera with the micrometers to fine tune the height of the camera over the belt. (Note that the dimension in the diagram is to the body of the camera, not the lens.) Level the camera so it is pointing straight down and the optical plane of the camera is approximately parallel to the gap.
8. **ENSURE CAMERA POWER SUPPLY IS OFF!** Connect camera data connection to Raptor, using a video grade coaxial cable, and camera control to the Raptor with a ten foot quad-insulated 25-pin serial cable. Connect the power connection of the camera to the camera power supply. Connect entire cable assembly to Raptor. (Figure 2.10)
9. Set the camera focus to approximately five feet and the aperture to approximately 5.6.
10. Mount the sensors on the framework as depicted in Figure 2.9, and connect to the host PC's serial port with the circuit shown in Figure 2.2.
11. Power on the computer, camera, light, and sensors.

Calibration Procedure:

1. Use the *RAPTEST* program to view the raw video output. The image should be an even white "waterfall" down the entire screen. If not, adjust the camera position, camera settings, and/or light position to make this so. If the screen consists of multiple horizontal bars of white, gray, and black, check the timing circuit. Using this program, scan a B12 frame to make sure the system is timed properly.
2. Use the *FRCALIB* to test acquisition and the sensors by scanning a B12 frame. Scan it oriented straight and at 45 degrees. If the bottom edge of the frame is cut off, adjust sensors closer to gap. Check several other frames to ensure the sensors are triggered properly and are not too high or too low. Adjust height of sensors if necessary. Also, ensure that the dashed bounding box only contains the frame image, with a small background margin separating the two. If there is a problem, ensure that no foreign object is obstructing the camera's field of view.
3. Execute *FRAMESYS* with the command:

```
framesys -cfg vtlab.cfg
```

This configuration file contains preliminary values for the system calibration. then execute the following algorithm to calibrate the inch/pixel values which are necessary to finalize the system.

Inch/Pixel Calibration for *FRAMESYS*:

Phase 1 Calibration.

For all of the Phase 1 steps, use an aligned W1242 single interior opening frame centered in the linescan camera's field of view.

A) Obtain basic row dimension.
   Make three measurements of the frame row (Y) dimension for the interior opening border (Region 1), and average them together. Express it as a ratio of the average to the true dimension of the frame. Update the configuration file based on this ratio. Repeat once.

B) Obtain outer border (Region 0) row ratio.
   Repeat (A) for the row dimension of Region 0. (Note that the configuration file parameter requires the ratio only.)

C) Repeat (A) and (B), except in the column (X) dimensions.

D) Scan the frame oriented straight. The MSE error should be less than 0.1.

E) Test the system with the calibration frame at several orientations.

Phase 2 Calibration.

A) Test frame style VAN36 aligned in all 4 orientations

B) Test frame style BLB18 aligned, with the wide part on the left of the center line of the conveyor belt.

C) Test AGS24 properly aligned.

Troubleshooting Guide for the Frame Recognition System

(Table assumes proper knowledge of operation of *RAPTEST*, *FRCALIB*, and *FRAMESYS* executables)

Problem	Possible Solution
FRAMESYS	
1. Framesys reports "can't see the frame" or similar message.	1. Ensure system is powered up properly 2. Ensure camera/sensor system is aligned properly
2. Framesys consistently reports "too few corners found for Region xxx"	1. Check conveyor belt speed 2. Ensure system is providing a smooth feed over the imaging gap 3. Ensure sensors are in the correct position (via *FRCALIB*)
3. Framesys consistently reports "too many corners found for Region xxx"	1. Ensure system is providing a smooth feed over the imaging gap 2. Ensure sensors are in the correct position (via *FRCALIB*) 3. Inspect frame itself for quality/gaps in frame
4. Framesys reports "chain code too short for Region xxx" or similar message	1. Ensure field of vision of camera is not blocked and noise-free.
5. Frame consistently not recognized or misclassified	1. Ensure that frame is represented properly in the system, check for exception if misclassified 2. Recalibrate system 3. Ensure feed across gap is smooth 4. Make sure camera is level 5. Replace backlight
6. VKD36 not recognized consistently	1. Reduce aperture slightly
7. Frame stops acquisition before the frame is clear of the sensors	1. Check sensors to make sure they are triggering properly 2. Check for proper feeding of the frame 3. If frame is very long, consult technical support for aid. The pseudo frame length must be increased and can only be done by a modification of the source code.
8. System reports cannot find <file>. Applies to database, cfg files, and raptor control files	1. Make sure, in *config.sys* menu, Raptor Development Mode is selected 2. Make sure relevant files are in current directory (for database/cfg files) or in the path for raptor files (check with the "set" command)

	3. If a database file is missing be sure it should contain data. Empty database files have not been supplied and are not needed.
9. System immediately exits to DOS with a malloc or calloc error	1. Reboot system 2. Remove memory resident routines (run bare-bones) 3. Add more system memory to system (last resort)
FRCALIB	
1. Frame corner cut off in image	1. If bottom corner, adjust slack value in command line parameters, if other adjust camera position
2. Noisy image	1. Ensure no foreign light are blocking the gap 2. Adjust the backlight 3. Replace the backlight
3. Dashed box does not contain the scanned frame image with a tight fit	1. Check the system for a noisy image 2. Adjust distance from camera to belt (closer)
4. Frame distortion/slip significant	1 Ensure entire system (camera, framework and belt) is level in all relevant directions. 2. Adjust conveyor belts 3. Make sure feed across gap is smooth
5. Frame image unpredictably truncated	1. Adjust sensors (height) 2. Maximum length of frame supported by system is 1024 lines, make sure this is not exceeded
RAPTEST	
1. Image #2 experiencing drop-outs (solid horizontal bars of various gray shades)	1. Shorten length of cables from camera to Raptor 2. Check cable quality 3. Adjust backlight 4. Replace backlight
2. In Image #2, when an object is place in the plane of vision, there should be a solid stripe down the screen. This stripe has noisy edges (extra pixels/pixels dropped)	1. Adjust aperture 2. Adjust focus 3. Adjust backlight 4. Ensure system is level/stabily mounted 5. Replace backlight
3. No image/Improper Image	1. Ensure system is properly installed/powered 2. Ensure camera is properly aligned 3. Adjust backlight 4. Replace camera with known working camera and try again

How to Add a New Frame Style to the Database

1. Determine the number of interior openings in the new frame. For example, frame style W1242 has one interior opening.

2. In the file *vtf_?.dat* in the *FRAMESYS* directory, where "*?*" is the number of interior openings in the new frame, create a record for the new frame style with the format shown below:

```
STYLE                BLB42/45
ITEM_NUM                  ???
DRAWING_NUM           03-1101B
NUM_OPENINGS                 4
SIZE      39.000    30.500
OPENING    1.625     1.625    14.750    21.250
OPENING   22.625     1.625    14.750    21.250
OPENING    1.625    24.500    14.750     4.375
OPENING   22.625    24.500    14.750     4.375
```

Where:

"**STYLE**" is the style name for the frame, less than 12 characters.

"**ITEM_NUM**" is the style name for the frame, less than 12 characters. If unknown, designate as "**???**".

"**DRAWING_NUM**" is the drawing number of the frame, less than 12 characters. If unknown, designate as "**???**".

"**NUM_OPENINGS**" is the number of interior openings in the frame

"**SIZE**" is the x and y outer border size of the frame, respectively (in decimal inches, to three places)

"**OPENING**" contains the information: x and y displacement of the upper left hand corner of the interior opening from the upper left hand corner of the outer border, and the x and y size of the interior opening, in that order from left to right. All dimensions are in decimal inches, to three places. Each interior opening has its own "**OPENING**" entry.

Be sure to delimit different entries in each row with spaces (NOT TABS), and end each line with carriage return. Resave the altered *vtf_?.dat* file as raw text (DOS's EDIT is recommended).

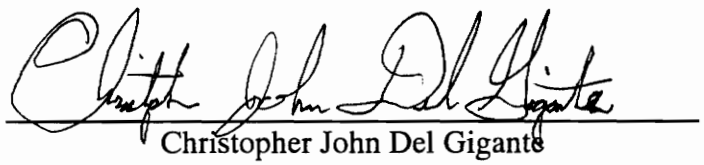
3. In the *FRAMESYS* directory, execute "*makedb*". Many messages will scroll on the screen as the database is recompiled, and will return the DOS prompt when finished

4. The database addition is finished. Test the new frame with the frame recognition system to ensure that it works properly.

Vita

Christopher John Del Gigante was born in Bayonne, New Jersey on March 5, 1971. In May, 1993, he obtained his Bachelor of Science degree from Virginia Polytechnic Institute in Electrical Engineering with a Computer Option and a Mathematics Minor. He joined Virginia Polytechnic Institute and State University as a Master of Science student in Electrical Engineering in August, 1993. During his Master's program, his areas of study included computers, digital signal processing, and robotics. His research interests include the industrial applications of machine vision, computer architecture, and software engineering.

Signed,



Christopher John Del Gigante