IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Android Based Visual Product Identification For The Blind

MAHESH KUMAR SUTHAR

PARUL UNIVERSITY,VADODARA (GUJARAT) ,INDIA

DEPERATMENT OF COMPUTER SCIENCE AND ENGINEERING

Abstract:

This project is developed to make the life of blind people easy. This is a camera based system to scan the barcode behind the image and read the description of the product with the help of Id stored in the barcode. This is very beneficial in case of finding out the description of packaged goods to the blind people and thus helping them in deciding to purchase a product or not especially which are packaged. This is because it becomes very difficult for the blind people to distinguish between the packaged goods. In order to use this system, all the user needs to do is capture the image on the product in the mobile phone which then resolves the barcode which means it scans the image to find out the Id stored. Thus this application really benefits blind and visually impaired people and thus making their work of identifying products easy. This is very easy to use and affordable as it requires a scanner to scan the barcode and a camera phone to take the picture of the image containing the barcode. This is now easy to implement as most of the mobile phones today have the required resolution in order to scan the barcode to identify the Id stored in it and read out the product description. This project can be implemented in any shopping mall, supermarket, Book stores, Medical stores etc.

1.3 FEATURES:

- Supports most of barcode types
- Quick text to speech translation
- Easy to use
- Affordable

1.4 APPLICATIONS：

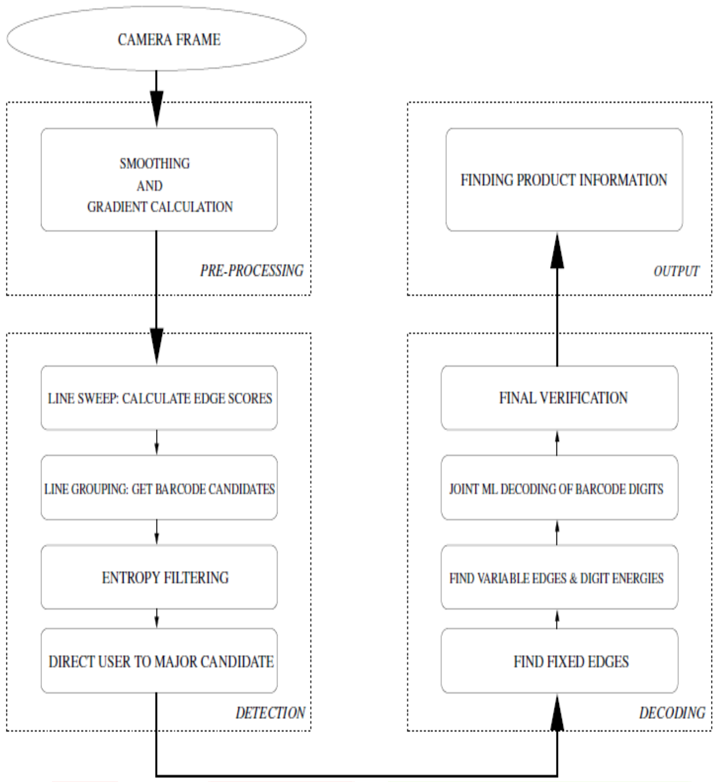
- Medical Stores
- Electronic devices
- Book Stores
- Shopping Malls

LITERATURE REVIEW

In the past, most research on reading barcodes focused on the problem of decoding barcode signals from a laser scanner, which are 1D waveforms (time series) representing barcode slices. One work [4] focuses on an optimal procedure for deblurring the waveforms to restore the locations of closely spaced edges in order to improve the decoding process. Another paper [6] uses an HMM model to decode a barcode signal in the presence of spurious edges. More recent work has addressed the problem of reading barcodes from images acquired by a camera. Most of this work has simplified the problem of locating and segmenting the barcode in the image by assuming certain constraints hold. In, the barcode is assumed to be horizontal in the image and viewed close enough for the long width of the barcode to subtend about two-thirds of the image width; although the authors point out that the method can be easily extended to the case of unknown orientation, it is unclear how well the algorithm would work with barcodes viewed from farther away. Other work assumes that the barcode can be detected from the morphological structure of binarized regions of the image, but the binarization procedure may fail on images that are noisier than the crisp example shown in the paper; similarly, assumes that a barcode can be detected by searching for a black bar using a spiral searching method, which scans in a spiral outward from the center of the image, but it is unclear that an individual black bar can be sufficiently well resolved at a distance. In addition to the commercial products mentioned in Sec. 1 (Red Laser, Nokia Point and Find, and Realeyes3D), which use proprietary algorithms, there have been some publications describing research on reading 1D barcodes using camera phones. In ,a barcode reading algorithm assumes that “a horizontal scanline in the middle of the image will cover the barcode,” and exploits the opportunity to capture multiple frames each second until this constraint is satisfied. (Our proposed system is similar in that it captures multiple frames until one of them can be read, but we detect the barcode from a distance and guide the user towards it before attempting to read it.) Finally, specifically addresses the problem of detecting and localizing 1D barcodes at unknown orientation in cluttered, noisy images, but there is no mention of how far away the barcodes can be detected (In the examples shown in the paper, the barcodes appear close enough to be at moderately high resolution). We build on our recent work focusing on reading a barcode (assuming it has already been segmented) using a Bayesian deformable template algorithm that combines a prior model of the geometric shape of the barcode pattern with a likelihood model that evaluates evidence in the image for edges. Such an approach is a principled technique of decoding noisy barcode images that contain spurious (or missing) edges. A related approach, also based on deformable templates, can successfully decode barcodes from extremely blurry and noisy pictures. Finally, we note that there is growing interest [11, 1] in the use of 2D barcodes, which are better suited to acquisition by camera than 1D barcodes, and encode information more densely. While 2D barcodes will one day supplant their 1D predecessors, at present almost all packaged goods are still labeled with 1D barcodes, which is why this paper focuses on them.

EXISTING SYSTEM

Before we go into the details of our algorithms, we give a brief overview of the major steps, shown schematically in Fig. 1. The system can be broken down into four main sub-systems: a detection part that looks for evidence of a



Barcode System

barcode in the image, a direction system that guides the user to a barcode if one is found, a decoding step that decodes the actual UPC-A code from the barcode once all the edges are seen, and the final stage which matches the UPC-A code to a product descriptions and outputs this information. Below is a summary of these steps:

1. **Detection:**

(a) Lines in 4 different orientations swept to determine collection of edge points with alternating polarities.

(b) Line scores tallied in direction perpendicular to sweep direction to get 2D representation of possible barcode areas.

(c) Orientation entropy used to eliminate false positives (e.g. dense text).

2. **Direction:**

(a) A maximal bounding box to enclose the detected barcode is calculated.

(b) The user is directed to the barcode by voice commands until enough edges are seen.

3. **Decoding:**

(a) Slices with maximum number of edges are found and edges localized with sub-pixel accuracy.

(b) Maximum likelihood (ML) estimation of the fundamental width and fixed edges.

(c) ML estimation of the barcode digits using the check bit.

(d) Detection attempted both right side up and upside down.

4. **Output:**

(a) Product information retrieved from database and read out.

PROPOSED SYSTEM

Algorithm for Finding Barcodes

1D barcode patterns are characterized by a rectangular array of parallel lines. The particular symbology we focus on in this paper is UPC-A (Fig. 2), which is widespread in North America and encodes a total of 12 decimal digits (one of which is a checksum digit that is a function of the preceding eleven digits). The UPC-A pattern contains a sequence of 29 white and 30 black bars, for a total of 60 edges of alternating polarity.

UPC-A barcode, encoding 12 digits

The code axis runs left to right in this image and the bar axis runs vertically upwards. Note that the bar patterns representing any specific digit have opposite polarity on the left and right sides of the barcode.

Any algorithm for finding a 1D barcode will conduct some sort of search for linear edge features in an image. While simple pre-processing steps such as intensity binarization and line extraction may be useful for identifying these features when they are clearly resolved, these steps may fail when the barcode is viewed from a distance. Instead, we decided to begin our detection algorithm by drawing on a simple, local image cue: the direction of the image gradient. The important signature of a barcode region is that, among pixels where the image gradient is significantly above zero, nearby pixels in the region have gradient directions that are either roughly aligned (corresponding to edges of the same polarity) or anti-aligned (corresponding to edges of opposite polarity). Thus, in the first stage of our detection algorithm, we calculate the image gradient everywhere in the image, and at all locations where the gradient magnitude is above a threshold (which we refer to as *edge pixels*) we calculate the gradient direction as an angle from 0 to 2_. Next we scan the image in four different orientations: horizontal, vertical, and both diagonals ($\pm 45^{\circ}$). Let us consider the horizontal orientation first. The scan is conducted in raster order (top row to bottom row, and left to right within each row), and we search for edge pixels whose orientation is consistent with vertical bars. For each such edge pixel, we search for a nearby "mate" pixel with the opposite polarity. Once a sufficient number of these pixels are found close by on aline segment, this segment is saved for the next step which sweeps the lines in a direction perpendicular to the first sweep direction to see if there are any approximately consecutive segments that have similar beginnings and ends. If a number of candidate line segments with similar beginnings and ends are found in this manner, this area is saved as a possible barcode candidate and passed on to the next stage which eliminates false positives that may arise, such as dense text when seen from a distance. These algorithms are summarized in Figures 3 and 4. The gradient angles which were quantized into 16 bins are histogrammed into 8 bins by combining pixels whose directions are 180 degrees apart. We then calculate the entropy of the resulting distribution, and compare it to a maximum threshold. Since a barcode is expected to only have lines of a single orientation, we expect a low entropy value. This stage eliminates false positives from the previous stage such as text, which has more orientations. As we direct the user to the barcode by giving directional feedback, the localization accuracy also increases.

Algorithm for Reading Barcodes

This part is based on a previous publication by the authors, , that models a barcode as a deformable template. We start with an initial estimate of the fundamental width, X, of the barcode (i.e. the width of the narrowest black or white bar) using the end points of the barcode bounding box from the previous stage. We first model the "fixed edges" of a UPC-A barcode, which are shown in Figure 2 as the guard-band edges and the digit boundaries shown in red. We model these fixed edges and digits conditioned on the barcode slice as obeying a Gaussian distribution centered around their expected geometric locations (which consists of their expected absolute distance from the left barcode edge and their relative distance from the previous fixed edge), and an exponential distribution in terms of their gradient strengths as given below: P(E,D|S) / e−L(E,S)−G(E,D) (1) where L(E, S) is the (log) likelihood term that rewards edge locations lying on high-gradient parts of the scanline, and G(E,D) is the geometric term that enforces the spatial relationships among different edges given the digit sequence.
By assuming conditional independence of a fixed edge from the previous fixed edges given the immediately prior edge, we can come up with a Markovian description of the fixed edges. This allows us to the find the maximum likelihood estimate of these locations efficiently using the Viterbi algorithm. We then iteratively refine this estimate and the fundamental width until we are satisfied with our estimate.
Once we find the fixed edge locations, we calculate the probabilities of the "in-digit" edges for each barcode digit, which gives us a distribution on the probabilities of each digit 0, . . . , 9 for this location. These are then used in conjunction with fixed edge estimates to get an overall estimate of the barcode. Since the digits are not conditionally independent due to the check bit, we use an auxiliary variable that is a running parity and preserves these probabilities as well as obeying the Markovian property. Hence, we can once more use the Viterbi algorithm to efficiently calculate the maximum likelihood estimate of the barcode. We use a multi-candidate Viterbi algorithm to ensure that the probability of our estimate is sufficiently larger than the probability of the second best ML estimate. We also ensure that the estimate is at most 1 digit away from the individually most likely digit estimates, since the parity digit is only guaranteed to find single-digit errors. This algorithm is summarized in Figure 5.

INITIALIZATION:
τ_G = minimum gradient threshold
n_E = minimum # of edges required
d_E = maximum distance between consecutive edges
SWEEP:
for orientation $t = 0, 45, 90, 135$ **do**
  for line $l = 1, \ldots, lastLineInThisOrientation$ **do**
    $count \leftarrow 0$
    for pixel $i = 1, \ldots, lastPixelOnThisLine$ **do**
      Let j be the last pixel on this line that was counted.
      if $|\nabla I_i| > \tau_G$ **then**
        //Gradient above threshold and angle approx. perpendicular to sweep line
        if $\angle \nabla I_i \approx \perp$ orientation **then**
          if $|\nabla I_i| \geq \max\{|\nabla I_{i-1}|, |\nabla I_{i+1}|\}$ **then** //non-maximum suppression
            if $\angle \nabla I_i$ is $\approx$ 180 degrees out of phase with $\angle \nabla I_j$, and $d_{ij} < d_E$ **then**
              $count \leftarrow count + 1$ //count this pixel
        else
          $count \leftarrow count - 1$ //pixel with strong gradient at wrong orientation
      else if $d_{ij} > d_E$ **then** //no edge pixel seen in a while
        $count \leftarrow count - 1;$
        if $d_{ij} > 2 * d_E$ **then** //no edges in a long while
          $count \leftarrow 0$ //end of candidate segment
      if $count = 0$ **then** //see if end of segment has been reached
        $score \leftarrow \max_{i \in lastSegment} count(i)$ //score is the max count for this segment
        if $score > n_E$ **then** //if the minimum # edges has been seen
          Record this segment as a barcode candidate segment for this line
        else
          Discard this segment

Figure 6.2 Line Scan Algorithm

probabilities of the "in-digit" edges for each barcode digit, which gives us a distribution on the probabilities of each digit 0, . . . , 9 for this location. These are then used in conjunction with fixed edge estimates to get an overall estimate of the barcode. Since the digits are not conditionally independent due to the check bit, we use an auxiliary variable that is a running parity and preserves these probabilities as well as obeying the Markovian property. Hence, we can once more use the Viterbi algorithm to efficiently calculate the maximum likelihood estimate of the barcode.

We use a multi-candidate Viterbi algorithm to ensure that the probability of our estimate is sufficiently larger than the probability of the second best ML estimate. We also ensure that the estimate is at most 1 digit away from the individually most likely digit estimates, since the parity digit is only guaranteed to find single-digit errors. This algorithm is summarized in Figure 5.

PROPOSED SYSTEM'S WORKFLOW:

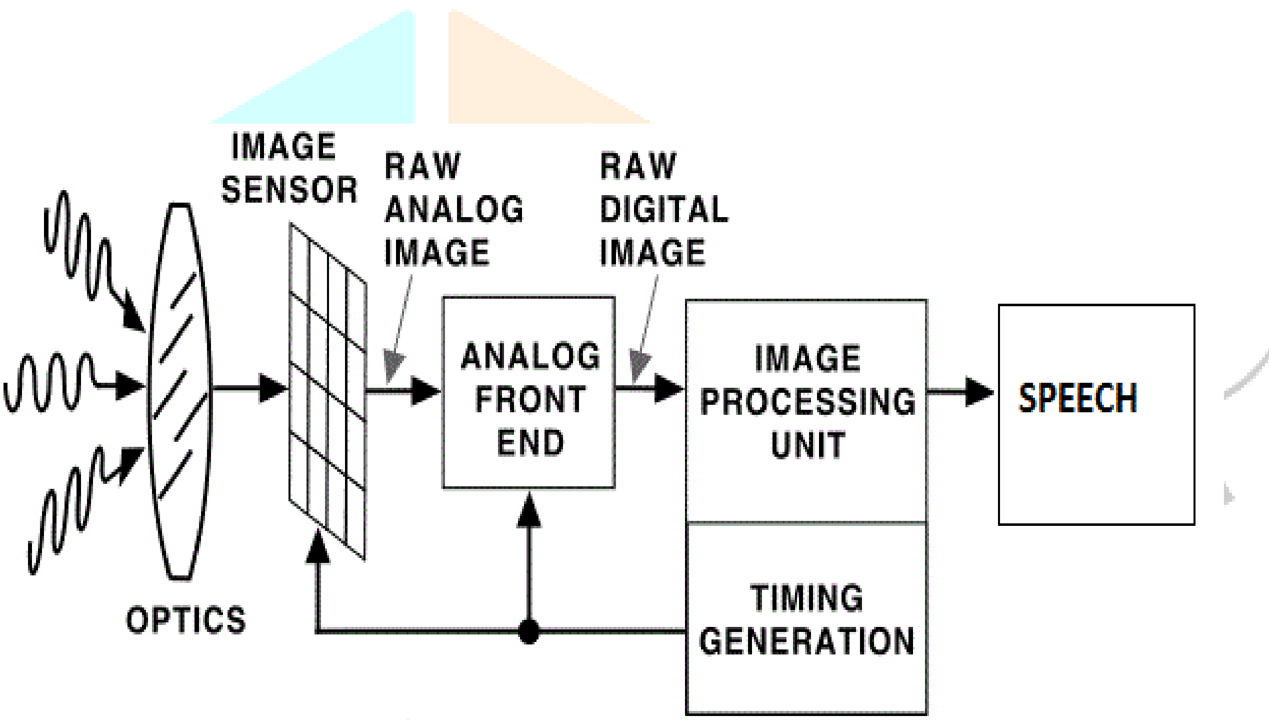


Figure 6.3 Workflow

Chapter 7 SCOPE

We experimented with the webcam focus to see how far we could detect a barcode. By setting the focus near infinity, the algorithm was able to detect barcodes at distances of up to 12 inches. Unfortunately, this focus setting made close-up views of barcodes too blurry to be read. In the future we hope to use an autofocus lens to be able to focus well at all distance ranges, assuming it can be made to focus fast enough for real-time use.
In the future we plan to port our system to a camera phone, and to extend our system to symbologies other than UPC-A, such as the EAN-13 (which is widespread in Europe). With the use of electronics further improvement can me made such as if a person wants to know only particular information about the product, we can provide few buttons having name written in brallie of details that will be provided after selection of that button. Eg. Button 1 for product name, 2 for price of product, 3 for mfg. and expiry date, etc.

Chapter 8 Methodology

8.1. Color Conversion

Using video capture from the board, the image is taken from the camera to Simulink and is converted from YCrCb to RGB for better processing in Simulink. The conversion requires taking the YCrCb and splitting it into the three color signals of Y, Cr, and Cb. After the split, since the Cr and Cb are smaller in dimension than Y, the Cr and Cb are upsampled using chroma resampling and transposed to match the dimensions of RGB from the 4:2:2 to 4:4:4. The three color signals are transposed again before sending them to the color space conversion from YCrCb to RGB still in three separate signals. The separate RGB signals are concatenated with a matrix concatenate for one to use as display, and for another line, it is sent to convert from RGB to intensity. The grayscale version of the image will be inserted to the feature calculations.This process of color conversion is also reversed before sending to output of board, except in this case, it will be from RGB to YCrCb.

8.2. Feature Calculations

The feature calculations module of the algorithm creates 3 scanlines for scanning barcodes as well as calculating the pixel values from the barcode intensity image in a given row to a vector. First a Gaussian filter is implemented to smooth out the image gradient identified as the barcode region. The gradient of the scanlines are set and validated so that the scanlines are inside the appropriate range. Then, the mean and standard deviation of the pixel intensities are calculated for the barcode area. The range of pixel parameters, f_low and f_high, for setting the color is determined. Pixels on the scanlines are compared to the f_low and f_high intensity values. A pixel is considered black if its value is less than f_low, and it is considered white if its value is f_high or larger. The remaining pixels are proportionally set between white and black. Black pixels are set to 1 and white pixels are set to -1. From the calculations, the vector of pixels from the scanlines is inputted to the barcode recognition. The scan lines are also sent to display to be added to the real time video.

8.3. Barcode Recognition

The barcode recognition module consists of three parts: bar detection, barcode detection, and a barcode comparison block. The bar detection block detects bars from the barcode feature signal. First, it tries to identify a black bar, if it is not there, then the first bar has zero width. If there is a black bar, then it calculates the pixels of the black bar. For the white bars, it does the same. After the bar detections, the barcode detection begins with the beginning bars and calculates all the possible values of barcode values that may form a valid string with all the possible separators. This function returns sequence of indices to barcode guard bars. The barcode comparison block takes in the codebook for all the encoded GTIN 13 barcode values. It also reverses it for determining the last 6 digits of the GTIN 13 barcode. The barcode recognition block takes in the barcodes and tries to match up the barcode with the numbers of pixels generated from the bar detection. In order to ensure better accuracy, the values are calculated from the left to right and right to left. The normalized confidence is calculated. The barcode recognition block set returns the barcode and the normalized confidence.

8.4. Barcode Validation

In the barcode validation stage of the algorithm, the simple calculation is used to determine whether the barcode is valid or not. It is calculated by taking the even elements and multiplying them by three. Then, add the sum of the odd elements with the sum of the even elements. Take 10 mod the sum and subtract 10. If the answer is the same as the check digit, which is the last digit, then the barcode is valid. This validation along with a confidence level higher than the threshold allows the barcode to be displayed on the screen.

8.5. Display

The display adds the scan-lines to the real time video and displays the barcode only if it is_validated and has a high enough confidence level to enable the switch for display. All the information is sent to the module to convert the 3 dimensional matrices back to 2D matrices. Then, RGB is converted to YCrCb format to display through the board.

8.6 System Implementation

After designing and testing the algorithms primarily in Matlab, the entire code base was ported to C++ for speed. The system was executed on a desktop computer with an inexpensive webcam, and the manual focus of the webcam was set to an intermediate focal distance: far enough for the webcam to resolve barcodes sufficiently well to be detected at a distance, but close enough for the webcam to resolve the barcode clearly enough to read properly at close range. We also experimented with autofocus webcams, but the time lag due to the autofocus feature proved to be impractical for a real-time system. Microsoft Speech API was utilized for the oral directional feedback. We devised a simple acoustic user interface to guide the user to the barcode. For each image frame, if a candidate barcode is detected then the system issues directional feedback instructing the user to move the camera left, right, up or down to better center the barcode in the field of view. If the barcode is oriented diagonally then the user is instructed to rotate the barcode, to allow the barcode to be aligned either approximately horizontally or vertically with the pixel lattice; this was done because the square pixel lattice maximally resolves the 1D barcode pattern when the code axis is perfectly horizontal or vertical, whereas barcodes oriented diagonally are harder to resolve. (Note that it is unnecessary to tell the user which direction to rotate in, since the user need only align the barcode to the pixel lattice modulo90◦.) If the barcode is close enough to detect but too far to read then the system tells the user to bring the camera closer, or if the barcode covers a very big portion of the webcam, the user is instructed to move farther to ensure the whole barcode is captured. Once the barcode is sufficiently close and well centered, the system attempts to read the barcode repeatedly (sounding a beep each time to inform the user) until the barcode is decoded with sufficiently high confidence. The barcode digit string read by the algorithm is looked up in a UPC code database (freely available online at http://www.upcdatabase.com/); if the string exists in the database then the corresponding descriptive product information is read aloud (e.g. “Walgreens Fancy Cashew Halves with Pieces. Size/Weight: 8.5 oz. Manufacturer: WALGREEN CO.”). If the string is not present in the database then the system alerts the user to this fact and outputs the barcode string. Even though the detection stage worked well at 320x240 resolution at around 15fps, for our experiments we used 640x480 resolution to be able to resolve more lines and read the barcode when it is not exactly aligned. In this mode, using a 2.4Ghz Intel Pentium processor with 2GB of RAM, our algorithm ran at up to 7fps (detection and decoding) without sound. However, due to the lag caused by the TTS (text-to-speech) system, in normal circumstances we are limited to only a few frames a second, which seemed to be sufficient for this experiment.

n_L = minimum # of lines required
d_L = maximum distance between matching lines
τ_S = minimum score required to declare barcode candidate
$\mathcal{B} = \{\}$ //list of candidate areas
SWEEP:
for orientation $t = 0, 45, 90, 135$ **do**
  for line $l = 0, \ldots, lastLineInThisOrientation$ **do**
    for barcode segment candidate $c = 1, \ldots, lastCandidateOnThisLine$ **do**
      if $\exists b \in \mathcal{B}$: $begin_b \approx begin_c, end_b \approx end_c$ and $d_{lb} < d_L$ **then**
        //There was a previous barcode area candidate with similar beginning and end a little earlier
        $count_b \leftarrow count_b + 1$
      else
        $\mathcal{B} \leftarrow \mathcal{B} + c$ //Add this segment as the beginning of a new candidate area
        $count_c \leftarrow 1$
      for $b \in \mathcal{B}$ **do** //check that the candidates are still valid
        if $d_{lb} > d_L$ **then** //have not seen a match in a while
          $count_b \leftarrow count_b - 1$
          if $d_{lb} > 2 * d_L$ **then** //have not seen a match in a long while
            $count_b \leftarrow 0$
        if $count_b = 0$ **then** //end of candidate
          $score_b \leftarrow max_{l \in b} count(l)$ //score is the max count over all lines in this area
          if $score_b > \tau_S$ **then** //if the minimum # lines has been seen
            Record b as a barcode area segment
          else
            $\mathcal{B} \leftarrow \mathcal{B} - b$ //Discard this candidate

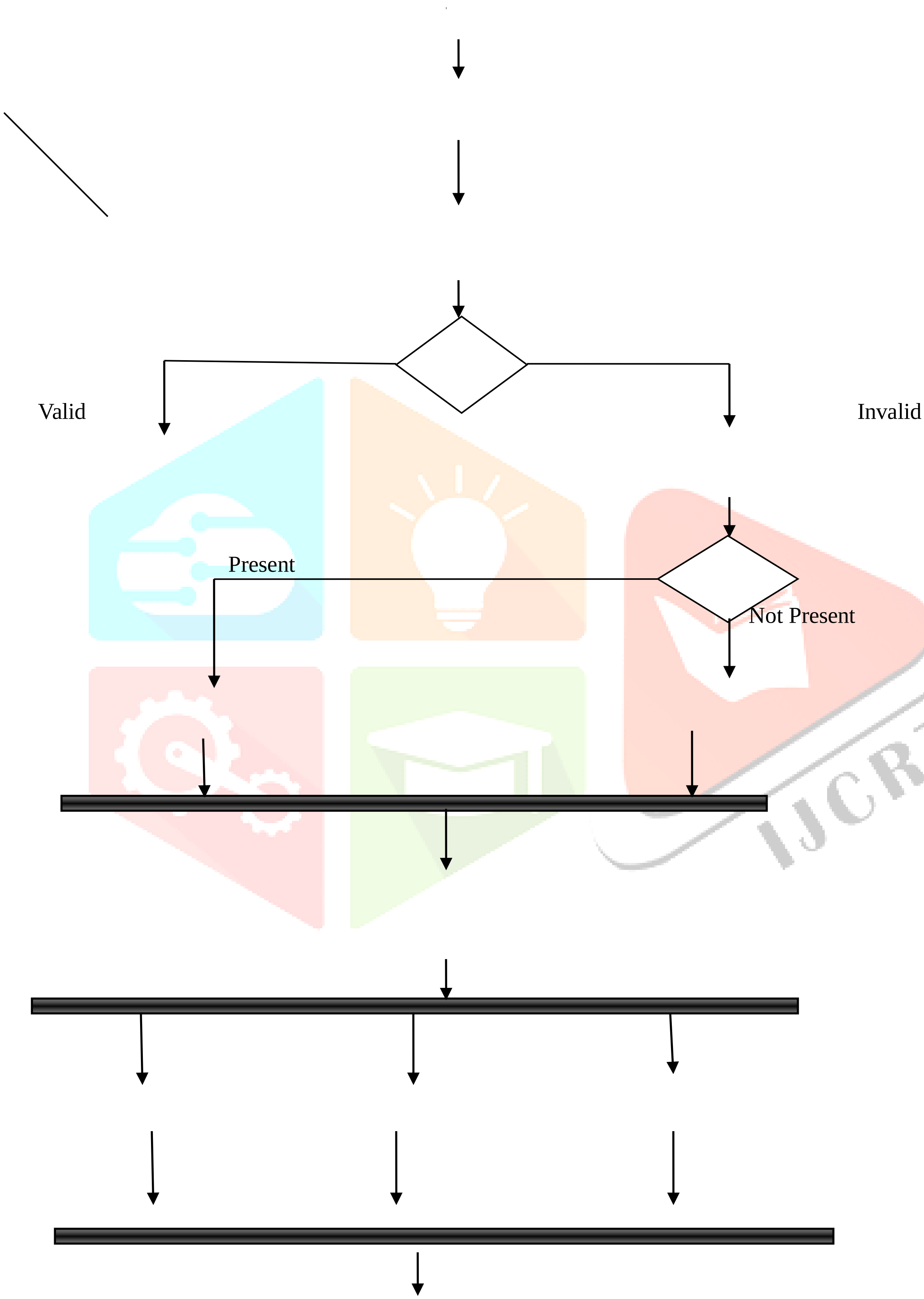
Figure 8.1 Line Tally Algorithm

INITIALIZATION:
$X_{initial} = \frac{lastEdge - firstEdge}{95}$
FIND EDGES AND DIGIT PROBABILITIES:
Find the N_{slices} lines in the barcode with the highest edge count
for Slice $i = 1, \ldots, N_{slices}$ **do**
  Estimate $N_{fixedEdgeEstimates}$ fixed edges
  for Fixed Edge Estimate $j = 1, \ldots, N_{fixedEdgeEstimates}$ **do**
    for Barcode digit $d = 1, \ldots, 12$ **do**
      Get digit probabilities for each numeric digit $0, \ldots, 9$
  Marginalize digit probabilities over all fixed edge estimates
Marginalize digit probabilities over all slices
BARCODE ESTIMATION:
for Barcode digit $d = 1, \ldots, 12$ **do**
  Calculate auxiliary running parity check digit probabilities.
ML Estimation of the 2 most likely sequence of auxiliary random variables
Convert auxiliary random variables back to barcode digits
BARCODE VERIFICATION:
if Probability of most likely sequence $> K \times$ Probability of the second most likely sequence **then**
  if Most likely sequence differs from individually most likely digits by at most 1 digit **then**
    OUTPUT BARCODE
Get new frame

Figure 8.2 Barcode Decoding Algorithm

Chapter 9 Design Detail

9.1 Activity Diagram



TASK		August			September			October			November			December			January			February			March		
		1	15	30	1	15	30	1	15	30	1	15	30	1	15	30	1	15	30	1	15	30	1	15	30
1	EXISTING SYSTEM STUDY																								
2	PLANNING PHASE																								
3	PROBLEM DEFINATION																								
4	INVESTIGATION OF SYSTEM REQUIREMENTS																								
5	APPENDING KNOWLEDGE BASE OF SYSTEM																								
6	DESIGN OF BARCODE READER IN MATLAB																								
7	COLLECTION OF PRODUCT INFO i.e (CREATE DATABASE)																								
8	DESIGN OF SPEECH SECTION IN MATLAB																								
7	IMPLEMENTATION																								
8	UNIT AND SYSTEM TESTING																								
9	SYSTEM ANALYSIS FOR IMPORVEMENT																								
10	FINAL PROJECT DEPLOYMENT																								

10.2HARDWARE AND SOFTWARE REQUIREMENT

10.2.1 SOFTWARE

- **MATLAB 7.0**

MATLAB® is a high-level language and interactive environment for numerical computation, visualization, and programming. Using MATLAB, you can analyze data, develop algorithms, and create models and applications. The language, tools, and built-in math functions enable you to explore multiple approaches and reach a solution faster than with spreadsheets or traditional programming languages, such as C/C++ or Java™.

You can use MATLAB for a range of applications, including signal processing and communications, image and video processing, control systems, test and measurement, computational finance, and computational biology. More than a million engineers and scientists in industry and academia use MATLAB, the language of technical computing.

MATLAB lets you manage, filter, and preprocess your data. You can perform exploratory data analysis to uncover trends, test assumptions, and build descriptive models. MATLAB provides functions for filtering and smoothing, interpolation, convolution, and fast Fourier transforms (FFTs). Add-on products provide capabilities for curve and surface fitting, multivariate statistics, spectral analysis, image analysis, system identification, and other analysis tasks.

10.2.2 HARDWARE

Processor	:	Intel Processor IV and above
RAM	:	4 GB
Hard disk	:	40 GB
Monitor	:	Any Branded Monitor having resolution of 1024*768

Chapter 11 CONCLUSION

We have described a novel algorithm for finding and reading 1D barcodes, intended for use by blind and visually impaired users. A key feature of the algorithm is the ability to detect barcodes at some distance, allowing the user to rapidly scan packages before homing in on a barcode. Experimental results with a blindfolded subject demonstrate the feasibility of the system. In the future we plan to port our system to a camera phone, and to extend our system to symbologies other than UPC-A, such as the the EAN-13 (which is widespread in Europe).

Chapter 12 REFERENCES

[1] A. Adelmann, M. Langheinrich, and G. Floerkemeier. A toolkit for bar-code-recognition and resolving on camera phones - jump starting the internet of things. In Workshop on Mobile and Embedded Interactive Systems (MEIS'06) at Informatik 2006, 2006.
[2] D. Chai and F. Hock. Locating and decoding EAN-13 barcodes from images captured by digital cameras. In Information, Communications and Signal Processing, 2005 Fifth International Conference on, pages 1595–1599, 2005.
[3] O. Gallo and R. Manduchi. Reading challenging barcodes with cameras. In IEEE Workshop on Applications of Computer Vision (WACV) 2009, Snowbird, Utah, December 2009.
[4] E. Joseph and T. Pavlidis. Deblurring of bilevel waveforms. IEEE Transactions on Image Processing, 2, 1993.

Chapter 4 AIM AND OBJECTIVE

The aim of this project is to develop an algorithm that will enable blind users to get information about a particular product.

We build on our recent work focusing on reading a barcode (assuming it has already been segmented) using a Bayesian deformable template algorithm that combines a prior model of the geometric shape of the barcode pattern with a likelihood model that evaluates evidence in the image for edges. Such an approach is a principled technique of decoding noisy barcode images that contain spurious (or missing) edges. A related approach, also based on deformable templates, can successfully decode barcodes from extremely blurry and noisy pictures.

Finally, we note that there is growing interest in the use of 2D barcodes, which are better suited to acquisition by camera than 1D barcodes, and encode information more densely. While 2D barcodes will one day supplant their 1D predecessors, at present almost all packaged goods are still labeled with 1D barcodes, which is why this paper focuses on them.

Chapter 5 PROBLEM STATEMENT

In India blind, visually impaired and old peoples with low vision tend to rely on others to go out and buy their essential products. The busy schedules won't allow people to help them out every time. Hence there is need of fast efficient and inexpensive way of scanning large number of barcodes and read out the information of corresponding product.