Hardware Implementation of the Hough Technique for Irregular Colour and Grey-level Pattern Recognition

Witold ŻORSKI, Andrzej ŻAK, Martin TURNER*

Abstract.

Keywords

1. Introduction to the Hough Transform
The Radon Transform is equivalent to the Hough Transform when considering binary images (i.e. when the function $I(y,x)$ takes values 0 or 1). The Radon Transform for shapes other than straight lines can be obtained by replacing the delta function argument by a function, which forces integration of the image along contours appropriate to the shape.

Using the Radon Transform to calculate the Hough Transform is simple (almost intuitive) and is often applied in computer implementations. We call this operation **pixel counting**.

An (alternative) interpretation of the Hough Transform is the so-called **backprojection** method. The detection of analytical curves defined in a parametrical way, other than straight lines is quite obvious. Points $(y,x)$ of image lying on the curved line determined by $n$ parameters $n_1, a_2, \ldots, a_m$ may be presented in the form:

$$\{(g: \mathbb{R})y,x \mid \lambda_1 = \lambda_2 = \cdots = \lambda_n\},$$

where $\{(g: \mathbb{R})y,x \mid \lambda_1 = \lambda_2 = \cdots = \lambda_n\}$ describes the given curve.

By exchanging the meaning of parameters and variables in the above equation we obtain the backprojection relation (mapping image points into parameter space), which may be written down in the following way:

$$\{(a_1, a_2, \ldots, a_m, y, x) \mid \lambda_1 = \lambda_2 = \cdots = \lambda_n\}.$$

From equation (3) the Hough Transform $H(a_1, a_2, \ldots, a_m)$ for the image $I(y,x)$ is defined as follows: $\lambda = \epsilon = \cdots = \epsilon$, $\lambda = \epsilon = \cdots = \epsilon$, $\lambda = \epsilon = \cdots = \epsilon$.

In order to calculate the Hough Transform digitally an appropriate representation of the parameter space $H(a_1, a_2, \ldots, a_m)$ is required. In a standard
implementation, any dimension in the parameter space is subject to quantisation and narrowing to an appropriate range. As a result, an array is obtained where any element is identified by the parameters \( a_1, ..., a_n \). An element in the array is increased by 1 when the analytical curve, determined by coordinates \( (a_1, ..., a_n) \), passes through point \( (y, x) \) of the object in image I. This process is called accumulation and the array used is called an accumulator (usually marked with a symbol A).

Thus, we may assume that the Hough Transform is based on a representation of the image I into the accumulator array A, which is defined as follows:

\[
\forall p \in \mathbb{P}, A(p_1, ..., p_d) = \sum_{i=1}^{d} p_i
\]

The symbol \( N_{p_i} \subset \mathbb{P} \) determines the range of \( i \)-parameters of a \( p \)-dimensional space \( \mathbb{P} \). Determining array A is conducted through the calculation of partial values for points of an object in image I and adding them to the previous ones (see 4) which constitutes a process of accumulation. Initially, all elements of array A are set to zero.

This paper presents a hardware implementation of the Hough technique to the tasks of irregular colour and grey-level pattern recognition. It is based on the Hough Transform with a parameter space defined by translation, rotation and scaling operations. A fundamental element of this method is the generalisation of the Hough Transform for grey-level and colour images. The Generalised Hough Transform has been previously described in detail in [19]. Nevertheless a short introduction to the technique is given in this paper.

2. The Binary Hough Transform for irregular objects

Basic definitions

digital images

\[
\rightarrow = \times \times \ldots \times \subset \mathbb{P}
\]

object

\[
\epsilon = \mathcal{E} =
\]
The Binary Hough Transform

\{ \alpha = \} 

\alpha

Figure 1. Scaling, rotation and translation of pattern with respect to an arbitrary point

\alpha = , ,

\{ \alpha - \alpha + \alpha - \alpha + \}

\alpha

\alpha

\alpha

\alpha

\alpha

\alpha
Hardware Implementation of the Hough Technique for ...

\[ \alpha = \sum \alpha \]

\[ \alpha = \begin{cases} \in & \text{otherwise} \\ 0 & \end{cases} \]

\[ = x x = x x - \Delta \alpha = \frac{\pi}{L} \]

Figure 2. Hough Transform for complex image

3. Generalisation of the Hough Transform for grey-level images
The process of binarisation can lose important information. The problem lies in the process of accumulation. At the beginning let us try to write the equation (12) in a different variation:

\[
\alpha = \begin{cases} 
\alpha'''' = \\
\alpha'''' - \end{cases}
\]

Equation (14) suggests the idea of modifying (12) in the following way:

\[
\alpha = \alpha'''' - \alpha''''
\]

This form tells us what to do in the case of grey-level images. However, we must first define the concept of a grey-level image, an object appearing in such an image and the concept of a grey-level pattern in a computer vision system.

Definitions

- **image with 256 grey levels**
  
  \[
  \rightarrow = \times \subseteq
  \]

- **Object**
  
  \[
  \rightarrow K \subseteq = \times \subseteq
  \]

- **Pattern**
  
  \[
  \rightarrow \times = \times \subseteq
  \]

Generalisation of the Hough Transform

\[
\alpha = \sum_c \alpha
\]

\[
\alpha = \alpha'''' - \alpha''''
\]
and the values \( y', x', \) are calculated from

\[
\begin{align*}
\theta &= + - \alpha - - \alpha + \\
\phi &= + - \alpha + - \alpha +
\end{align*}
\]

Application of the histogram function

\[
\Phi \rightarrow \Phi
\]

**ALGORITHM** – histogram analysis

\[
\begin{align*}
\Phi &= \Phi \\
\Phi &= - + = - + \\
\Phi &= \Phi \\
\Phi &= - \Phi \\
\pi &= \frac{4}{\pi} \sum_{k=0}^{255} |\Phi - \Phi|
\end{align*}
\]
4. The Hough Transform and the scaling problem

\[ \xi \in \xi \]

\[ = \times \times \times = \times \times - \times \xi \xi \Delta \alpha = \frac{\pi}{\pi} \]

\[ \xi \xi \]

\[ \_ \_ \_ \_ \times \]

\[ \_ \_ \_ \_ \times \]

\[ \_ \_ \]
5. Generalisation of the Hough Transform for Colour Images [18]

The first approach
The second approach

The second approach to the problem of a colour pattern location in a colour image is based on calculating the distance between \((y,x)_{P}\) and \((y,x)_{I}\) with the following formula (see 20):

\[
|\sqrt{(b-b')^2 + (g-g')^2 + (r-r')^2}| \quad (27)
\]

i.e. Euclidean distance between two points \((b,g,r)_{III}\) and \((b,g,r)_{MMM}\) in the RGB cube. Unfortunately, this is often not acceptable due to its high computational complexity.

The third, final approach

The third approach assumes that an input image and a pattern are given with 256-colour depth. Then it is easy to calculate the distance between \((y,x)_{P}\) and \((y,x)_{I}\) by pre-calculating a square matrix \(d_{M}\), that includes all possible distances between any two of 256 base colours. The size of the matrix is 65,536 integer cells, which is acceptable in terms of memory requirement. Such a matrix solves the problem of calculating a distance as well as accelerates the calculations considerably. It allows us to obtain a distance in the following way (compare 20 and 27):

\[
|\sqrt{(b-b')^2 + (g-g')^2 + (r-r')^2}| = d_{M}
\]

Nevertheless another problem appears, i.e. the problem of creating matrix \(d_{M}\). There are many colour models for example: RGB cube, CMY or single-hexcone HSV. Thus it is necessary to choose the colour model first and then establish the colour quantisation method, and then finally calculate all possible distances to be stored in matrix \(d_{M}\).

Figure 5. Object location in a colour image obtained from HST
6. Hardware implementation of the Hough technique [23]

Introduction to the Hardware [22]

Figure 6. Altera SOPC board (the APEX device is zoomed in)
The board includes physical interfaces for widely used standard interconnects as follows:

• 10/100 Ethernet with full and half duplexing
• peripheral component interconnect (PCI) mezzanine connector
• high and low-speed universal serial bus (USB) host supporting
• IEEE Std. 1394a FireWire interface at 100, 200 and 400 Mbps
• IEEE Std. 1284 parallel interface and two RS-232 ports (DCE and DTE)
• custom interface, i.e. 50 user I/O pins connected directly to the APEX device.

In order to support processor functions implemented in the APEX device, the board includes a memory system consisting of the following parts:

• volatile 64MB of SDRAM memory
• non-volatile 4MB of Flash memory and 256KB of EPROM memory
• pipelined 1MB of cache memory with burst SRAM.

The board also supports IEEE Std. 1149.1 Joint Test Action Group (JTAG) for system testing, as well as Extended JTAG (EJTAG) for development and debugging of MIPS-like microprocessor functions. For additional analysis, the JTAG port can be used with the SignalTap embedded logic analyser available with the Quartus™ development software.

The APEX 20K device is programmed via the JTAG interface. It may be programmed directly using the Quartus or the MAX+PLUS ® software using either the MasterBlaster™ or ByteBlasterMV™ dedicated cable.

Technical Assumptions for the Project

- the input image and the patterns are 256 grey levels
- the size of the patterns is set at 25x25 pixels
\[ \Delta \alpha = 0^\circ \]

The most important problem in the presented scenario is the implementation of the function \( s, y, x (\alpha) \) in the APEX device. In practice it means that "the sealing method" must be implemented. Thus, the difference between the fragment (indicated by \( y, x \)) of the input image and the pattern (indicated by \( s, \alpha \)) must be calculated by the hardware.

Implementation of function \( s, y, x (\alpha) \) in the APEX device is presented. Figure 7 shows the block diagram of the structure implemented in the APEX device.

As previously described the SOPC board receives the input image and the patterns from the PC at the beginning. The implemented structure has only one aim; to calculate the difference between the given fragment of the input image and the given pattern. The image fragment and the pattern are indicated by the arguments of function \( s, y, x (\alpha) \). This means that the PC sends to the SOPC board co-ordinates \( s, y, x (\alpha) \) of the accumulator array and waits for the result, i.e. value of function \( s, y, x (\alpha) \). This is a single cycle of the whole process of the accumulator array calculation which starts by sending signal RESET. Signal RESET clears the main elements of the implemented structure: COUNTER, BUFFER, SUBTRACTER and COMPARATOR. This is the initial step for the process and after that the device is ready to calculate accumulator array cells one by one. In order to describe precisely the basic cycle for the implemented structure the following algorithm will be useful.
Algorithm

Step 0: The APEX device receives (subsequent) co-ordinates \( s, y, x(\alpha) \) and indicates adequate fragment of the input image (by \( y, x \)) and adequate pattern (by \( s, \alpha \)).

Step 1: At the \( j \)-th step the BUFFER receives the \( j \)-th pixel (i.e. 8 bits) of the input image fragment and the SUBTRACTER receives the \( j \)-th pixel (i.e. 8 bits) of the pattern.

Step 2: The COUNTER sends the SUBTRACTER signal that begins the calculation process of the difference between the pixels. The obtained difference is added to the previous one stored in the SUBTRACTER.

Step 3: If there is any pixel of the pattern left, then \( 1 + j = j \) and go to step 1. If there is no more pixels of the pattern, the COMPARATOR receives value \( s, y, x(\alpha) \) from the SUBTRACTER and compares it with the previous one.

Step 4: The COMPARATOR sends out value \( s, y, x(\alpha) \) to the PC with information whether it is the temporary minimum or not. END

The COUNTER acts as the control element of the implemented device. It controls all elements by sending control signals and has a database containing information about the indexes of the pattern pixels, as shown in Figure 8.

Figure 7. Block diagram of the structure implemented for function \( \alpha \)
Figure 8. Indexed pixels of the pattern.

Figure 9. Technical scheme of the implemented structure.
Example results

Figure 10. Example result with many similar elements in the input image
7. Conclusion
Although the elaborated hardware implementation of the Hough technique is a first version, it is very promising. There were many problems during realisation of the project, not mentioned in this paper (e.g. the problem of interconnects control logic implementation). Further work in this area is expected to bring more efficient implementations of the Hough technique. This may enable one to build a versatile real-time Hough Transform computer vision.

References


Hardware Implementation of the Hough Technique for...

Technical Report - The Hough Transform versus the UpWrite.

Über die Bestimmung von Funktionen durch ihre Integralwerte langs gewisser Mannigfaltigkeiten

Ż. Application of the Circle Hough Transform with a Clustering Technique to Segmentation of Digital Images

Ż. Fingerprint and Iris Identification

Method Based on the Hough Transform

15

Ż. Irregular Colour Pattern

Recognition Using the Hough Transform

15

Ż. Irregular Pattern Recognition

Using the Hough Transform

9

Ż. Pattern Recognition in the Case of Small Differences.

Ż. Ż. Hardware Implementation of the Hough Technique for Irregular Pattern Recognition.

System-On-a-Programmable-Chip Development Board User Guide

Ż. Design of a hardware implementation of the Hough Transform

Projekt sprzetowej realizacji transformaty Hougha

Reviewed by Professor Włodzimierz Kwiatkowski

Received November 2002