**Image Processing Techniques**

**5.1 Introduction**

Image Processing is the breakdown of images or video frames in digital format to abstract useful data from them. In the case of this research, images are processed to abstract features for hand shape recognition.

These are image processing techniques such as Canny edge detection, background subtraction using Gaussian Mixture Models,face detection, adaptive skin detection, CAMShift tracking , hierarchical Chamfer matching, connected components analysis. Each of these techniques is discussed in a separate subcategory below.

**5.1.1 Canny Edge Detection**

Edge detection is the method of ﬁnding the edges within an image. An edge is deﬁned as a point in an image with a disjointedness in brightness, or, in simple relations, a sharp change in brightness [4]. In edge detection is simpliﬁes an image representation to that of only its structural vision-based information. Canny developed the Canny Edge detection technique [13] in 1986 and it is one of the most common and strong edge detection techniques [56]. The Canny algorithm attempts to satisfy the following three conditions:

1. A low error rate: The detection of edges should be as correct as possible. The edges found in a picture should not be falsely ignored because error of these edges could aﬀect a system’s performance.

2. Good localization: The detachment between detected edge pixels and the actual edge pixels must be minimalized.

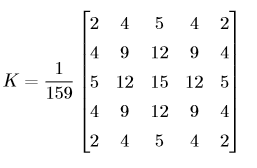
3. Minimal response: Multiple replies to an edge should be avoided by limiting detection to only a single response per edge.

The Canny edge detection algorithm contains four steps [37]. These are: the image smoothing via a Gaussian ﬁlter; calculation of the gradients in the image to highlight potential edges; applying non-maximum suppression to reach thin edges; and double thresholding to conquer edge lines. These steps are described in the below subsections

**5.1.1.1 the Image Smoothing Using a Gaussian Filter**

The primary step of Canny edge detection involves qualifying any extra noise in an image. Images normally hold some amount of noise. These sources of noise can simply, but incorrectly, be detected as edges–sharp changes in intensity–within the image.

As such, the image is smoothed using a Gaussian ﬁlter [67]. This includes convolving a Gaussian kernel K with the image I. Below is an instance of a Gaussian kernel of size 5×5 using a standard deviation of σ = 1.4 which can be used, but higher kernels can be used as well.



**3.1.1.2 Computation of the Image Gradients**

Once the image has been smoothed and excess noise has been ﬁltered out from it, the next step is to determine the intensity gradients of the image. These gradients are computed because they give an indication of the strength of edges in the image. At each pixel in the smoothed image, the gradients are determined using the Sobel operator as follows.

The gradients are approximated using a pair of three by three convolution masks, Sx and Sy. Sx represent the edges in the x-direction while Sy represent the edges in the y-direction. These 3x3 convolution masks are given as:

Convolving the two masks with the real image outcomes in two gradient images Gx and Gy. The Equation 3.2 below is then calculated at each pixel (i,j) to ﬁnd the gradient strong point at that pixel using the law of Pythagoras:

(5.2)

A simpler ration can also be used to approximate the gradient at pixel (i,j) in the form of the Manhattan distance measure given by:

(5.3)

The direction of the edge θ is also calculated at each pixel (i,j). The precise direction of the edge is determined using the following equation:

(5.4)

The considered direction θ of the edge is then rounded oﬀ to the adjoining 45-degree angle representative the directions of the horizontal and vertical neighbours and those of the two diagonal neighbours. As such, it is rounded of to one of four likely angles: 0 -degree, 45-degree, 90-degree or 135-degree.

**3.1.1.3 Applying a Non-maximum Suppression**

When the direction and magnitude of the edges have been resolved, a non-maximum suppression is applied to narrow edges by removal non-maximum pixels in each edge. This outcome in accurate narrow edges in the image, as required.

This is reached by investigative the gradient values of the neighbours on either side of each pixel (i,j) in the direction vertical to the direction of the pixel. If the gradient value of the pixel is larger than that of both neighbours, it is remarkable as being an edge pixel. If it is not, it is rejected i.e. set to zeor.

For instance, if the gradient direction for a pixel θ(i,j) is 0, meaning that it is North-South associated, it is related to the two neighbours on either side in the East-West direction. If its gradient value is larger than that of these neighbours, it is noticeable as an edge pixel. If not, it is set to 0.

**3.1.1.4 Double Thresholding**

Once the non-maximum suppression has been applied, a double threshold is applied to remove false edges which can cause features such as edge lines. The double threshold contains of an upper and lower threshold. The below steps are following to complete the edge detection process:

1. The upper threshold is applied to classify all ‘strong’ edges. A pixel is measured a ‘strong’ or conﬁrmed edge pixel if the gradient value of that edge exceeds the upper threshold.

2. The lower threshold is applied to classify all ‘weak’ edges. A pixel is measured a ‘weak’ or rejected edge pixel if the pixel gradient is below the lower threshold. Such edges are rejected.

3. All pixels that have a gradient value between the upper and lower threshold are measured as edge pixels if they are associated to a strong edge pixel in a 3 × 3 neighbourhood area.

4. If pixels with a gradient value between the upper and lower threshold are not associated to a strong edge pixel in a three by three(3x3) neighbourhood but are associated to at least one other pixel that has a gradient value between the upper and lower threshold in the same neighbourhood area, the previous step is recurring with a five by five(5x5) neighbourhood. If no strong edges are found in this extended area, the edge is rejected.

Canny suggested a double threshold ratio (upper:lower) of between (2:1) and (3:1) [13]. Figure 3.1 shows an example of an image to which the Canny edge detection algorithm has been used.

Figure 3.1: Canny edge detection: (a) The original image. (b) Application of the Canny edge detection algorithm [13].

**3.1.2 Face Detection**

The Viola-Jones [65] framework is a common framework for object detection. The framework has been used to face detection and it has demonstrated to be highly accurate and computationally eﬃcient[66,67,68].

The Viola-Jones object recognition framework classiﬁes objects in images with simple fundamental features called Haar-like wavelets. In addition, uses a novel data structure called an Intergral Image to signiﬁcantly get faster the detection of these features. In last, a modiﬁed Adaboost classiﬁer method is applied to arrange a series of weak classiﬁers trained to detect various Haar-like features into a denial cascade. This system results in a strong and highly eﬃcient object detector.

The following subcategories define each of these steps, namely: the nature and computation of haar-like features; the use of an integral image to get faster computation of haar-like features; the use of Adaboost to select suitable features for face detection; and the use of a ﬁnal refusal cascade as a face detector.

**3.1.2.1 Haar-Like Wavelet Feature Detection**

The object detection method of the Viola-Jones algorithm makes use of structures that are based on the value of Haar wavelets called Haar-like wavelet features. Haar-like wavelets contain of a set of discontinuous rectangles of the same shape and size that are either “light” or “dark” and are either vertically or horizontally together. Figure 3.2 shows two, three and four rectangle features.

Figure 3.2: Three types of Haar-like wavelet features used by the Viola-Jones face detector[66]

Each type of feature is passed over a goal image at various balances and positions. At each measure and position, the sum of the pixels corresponding to the dark region are subtracted from the sum of the pixels corresponding to the light region. If the outcome of this calculation exceeds a threshold value, this speciﬁc feature is determined to be present at this location and scale.

Two-rectangle features are considered by computing the summation of all the pixels in the dark region and take away these from the sum of all pixels in the light region and applying an acceptance threshold to the outcome. Three-rectangle features are added by applying an acceptance threshold to the diﬀerence between the joint sum of the pixels in the two light rectangles and the dark rectangle. Four-rectangle features are considered by applying an acceptance threshold to the diﬀerence between the combined sum of the pixels in the crosswise pairs of rectangles.

**3.1.2.2 The Use of An Integral Image to Compute Haar-Like Features**

Calculating the values of various features at every measure and position in an image is a very computationally expensive operation. Viola and Jones future an intermediate representation of an image called an Integral Image which enables the fast calculation of the sums of various features at any measure and position in the image.

Figure 3.3: Calculation of the Integral Image: The value of the Integral Image at (x,y) is the sum of all pixels to the top-left of the pixel, in the shaded region [66].

Specified an image I, the integral image representation G at any position (x,y) is the sum of the pixels to the top-left of (x,y), as shown in Figure 3.3, given by:

(3.5)

An alternative deﬁnition of the Integral Image is given in terms of the cumulative row sum S(x,y) at (x,y) as the following pair of recurrence relations which can be used to compute the image in a single pass:

G(x,y) = G(x − 1,y) + S(x,y) (3.6a)

S(x,y) = S(x,y − 1) + I(x,y) (3.6b)

where

S(x,−1) = 0 (3.6c)

and

G(−1,y) = 0

Using the Integral Image, it is possible to compute any Haar-like feature using only a few lookups in the image by easily computing the sum of any rectangle in the original image, as required. Referring to Figure 3.4, it is possible to compute the sum of the pixels inside the rectangle labeled D by subtracting the Integral Image value at point 4 from the sum of the Integral Image values at points 2 and 3, and adding back the Integral Image value at point 1 to counteract the excess caused by the intersection of rectangles (A + B) represented by point 2 and rectangles (A + C) represented by point 3.

The ability to compute the sum of pixels in any rectangle implies the ability to compute any Haar-like feature at any scale or location.

Figure 3.4: An example of the computation of the integral image [66].

**3.1.2.3 The Use of AdaBoost to Select Haar-Like Features**

AdaBoost is a learning algorithm which improves the classiﬁcation performance of weak classiﬁers. A modiﬁed version of the algorithm is used by the Viola-Jones face detection system to choose an optimal subset of the potentially large number of features and train a classiﬁer based on these features [65].

Even though each feature can be computed at a high speed, the computation of the set of features can be very slow since there are a large number of rectangular features associated with each image sub-window. Only those features are selected which best distinguish between positive and negative examples, thus limiting the number of features that are required to achieve a strong classiﬁer.

**3.1.2.4 A Rejection Cascade of Weak Feature Classiﬁers**

A rejection cascade of classiﬁers is constructed in such a manner as to achieve a high accuracy while signiﬁcantly lowering the computational cost for negative examples. The principle behind this idea is that simpler, and thus faster, boosted classiﬁers can be created to reject most of the negative sub-windows while still being able to detect almost all of the positive instances.

Figure 3.5: The typical structure of a rejection cascade[66].

The rejection cascade has the structure of a degenerate decision tree and it is depicted in Figure 3.5. With reference to Figure 3.5, when the ﬁrst classiﬁer obtains a positive result, it triggers the evaluation of the second classiﬁer, and a positive result from the second classiﬁer triggers the third classiﬁer. As long as every classiﬁer returns a positive result, this process continues on to the ﬁnal classiﬁer, after which a face is determined to have been detected in the sub-window in question.

On the other hand, if the result is negative at any classiﬁer, the sub-window is immediately rejected. This signiﬁcantly reduces the computational overhead of the algorithm for sub-windows in which no face exists.

**3.1.2.5 Evaluation of the Face Detection System**

The Viola-Jones face detection system was evaluated on the MIT+CMU frontal face dataset [55]. Some examples of the dataset with face detection performed on them are shown in Figure 3.6. The evaluation aimed to measure the speed as well as the accuracy of the technique. The system was shown to achieve a real-time detection speed of 15 frames per second (fps) on images with a resolution of 384 × 288 pixels when operating on a 700 MHz Intel Pentium III computer. The system achieved an accuracy of 93.9% with only 167 false detections.

Figure 3.6: Example of the testing data from the MIT+CMU dataset [55].

**3.1.3 Adaptive Skin Detection**

Skin detection is an image processing technique which segments skin pixels from non-skin pixels. It eliminates all non-skin pixels in an image and highlights only the skin pixels in the image. Applications of skin detection include human-computer interaction, human detection, hand tracking, face detection and face recognition [17, 27, 36]. Skin detection in this research assists in initializing and maintaining the hand tracking algorithm to track the hands of the user. The adaptive skin detection algorithm used was initially proposed by Achmed [2] and used in the feature extraction procedure of Li [36].

The procedure works as follows the face is detected; the skin colour distribution of the user is extracted from the face; it is back projected onto the original image to obtain a skin probability distribution; ﬁnally, the skin probability distribution is thresholded to obtain a binary skin map of the original image Each step of this procedure is explained in further detail in the following subsections.

**3.1.3.1 Face Detection**

The Viola-Jones face detection algorithm is used to determine the position of the face. A 10 × 10 pixel area at the centre of the detected facial frame is extracted and used as a representative skin colour distribution in the form of a histogram. Achmed showed that this region represents the skin colour very well as it is usually void of non-skin obstructions such as shadows, hair, eyes and spectacles [2]. The 10 × 10 pixel area of the nose is converted from the default Red, Green and Blue (RGB) colour space to the Hue, Saturation and Value (HSV) colour space. A histogram of the Hue and Saturation channels of the region is computed and taken as the representative skin colour distribution of the user.

**3.1.3.2 Histogram Back Projection and Thresholding**

The skin colour histogram is back-projected onto the original input frame resulting in a skin probability distribution of the input frame. The back-projection is achieved as follows. Given C represents the colour of a pixel in the image, and F is the probability that the pixel is skin, P(C|F) is the probability of drawing that colour when the pixel is actually skin. Then P(F|C) is the probability that the pixel is skin given its colour. This yields the following equation:

P(F|C) =

P(F) /P(C)

P(C|F)

The resulting back-projected skin probability image is converted into a binary image in which skin pixels have a value of 255 (white) and non-skin pixels have a value of 0 (black). This is achieved by thresholding the image using a threshold value of 60. This static threshold value was determined as being optimum by Brown [12] An example of a back-projected image is illustrated in Figure 3.7.

As seen in the ﬁgure, this technique eﬀectively segments skin pixels from non-skin pixels. There are, however, factors such as background noise or colours in the background which are similar to that of skin colour which can cause noise in the image. To this eﬀect, background subtraction in the form of Gaussian Mixture Models, described in the next section, are used to mitigate such sources of noise.

Figure 3.7: a) Original image and b) Skin-detected image

**3.1.4 Background Subtraction Using Gaussian Mixture Models**

Background subtraction is the segmentation of objects/regions in an image or a sequence of video frames that are of interest to an application, referred to as the foreground, from those that are not of interest, referred to as the background [57]. In the current case, the foreground consists of the hand of the user, while all other objects in the frame constitute the background.

Gaussian Mixture Models (GMMs) are a probabilistic method that can be used for eﬀective background subtraction. They can be used to highlight moving pixels in a frame with a history indicator over a set number of frames such that the brightness of a pixel indicates the recency of its motion, and regions with no motion over a number of frames appear as completely black.

Given an image sequence I, the history of a pixel at (i,j) at a speciﬁc time t can be represented as follows:

{I1,...,It} = {I(i,j,x) : 1 ≤ x ≤ t}

Each pixel can be modeled as a mixture of k Gaussian distributions. Letting Wx,t represent the weight estimate of the x-th Gaussian, the probability of a pixel possessing the value It at time t can be expressed using the equation below:

P(It) =

k X x=1

Wx,t × η(It,µx,t,Σx,t) (3.9)

where η(It,µx,t,Σx,t) is the normal distribution of the x-th Gaussian component with a mean of µx,t and expressed as:

η(It,µx,t,Σx,t) =

1

(2π)

n 2 | Σx,t |

1 2

e

−1 2 (It−µx,t)T Σ−1 x,t(It−µx,t) (3.10)

where Σk,t = σ2k,tI is the covariance of the k-th Gaussian component given I is the identity matrix.

A ﬁtness value Wx,t σx,t is used as a reference when ordering the number of distributions k and the ﬁrst M distributions are used for modeling the background scene, where the estimate of M is given by:

M = argminm(

m X x

Wx,t > Th) (3.11)

where Th is the threshold that represents the minimum portion of the background model.

Given an updated background, foreground detection is then achieved by labeling all pixels which are determined to be more than a standard deviation of 2.5 away from any of the M distributions as foreground pixels. If there is a match between the test value and the x-th Gaussian component Wx,t, it is updated as shown below:

Wx,t = Wx,t−1 (3.12a)

µx,t = (1 − ρ)µx,t−1 + ρIt (3.12b) σ2x,t = (1 − ρ)σ2x,t−1 + ρ(It − µx,t)T(It − µx,t) (3.12c) ρ = αη(It | µk,Σk)

where 1 α is deﬁned as the time constant which determines change. If there is no match between the Gaussian component and the test value, then it is updated as follows:

Wx,t = (1 − α)Wx,t−1 (3.13a)

µx,t = µx,t−1 (3.13b) σ2x,t = σ2x,t−1 (3.13c)

If the test value does not match any of the Gaussian components, a new Gaussian component with a high variance, low weight parameter, and the test value as its mean replaces the Gaussian component with the lowest probability. An example of GMMs applied to highlight the moving foreground of an image is illustrated in Figure 3.8.

Figure 3.8: The application of Gaussian Mixture Models (GMMs) to achieve background subtraction: a) Original image and b) Background-subtracted image.

**3.1.5 Hand Detection Using Hierarchical Chamfer Matching**

The hierarchical chamfer matching technique is explained in this section [8]. It is a matching algorithm used to detect a template object in an image. In the case of this research, it is used to detect the location and size of the signer’s hand and initialize the hand tracking algorithm. A template silhouette of the hand is used to ﬁnd a match in the input image.

Chamfer matching involves three stages: computation of an edge image on the image in which the search is carried out; computation of a Chamfer distance transform on the image in which the search is carried out; and edge matching of the template edge image with the search image distance transform. Subsections 3.1.5.1 and 3.1.5.2 describe the computation of the Chamfer distance transform and the edge matching process, respectively.

A hierarchical approach can be used to signiﬁcantly speed up the edge matching process. This is described in Subsection 3.1.5.3.

**3.1.5.1 Computation of the Chamfer Distance Transform**

A distance transform is an algorithm which converts an edge image into a distance image. Each non-edge pixel of the hand template silhouette image is given an intensity value ranging from 0 to 255. The intensity value is a measurement of the distance of the pixel to the closest edge pixel.

Various distance masks can be used to eﬀectively calculate the distance image. Li showed that a 3 × 3 mask with a (3,4) distance transform produced excellent matching results.

The process of computing a distance transform involves two passes which are made over an image by propagating the computed distance values across the image like a wave. First a “forward” pass from left to right and from top to bottom is carried out, followed by a “backward” pass from right to left and from bottom to top. For an image V of size W × H pixels, a computation of the forward pass is given by:

(3.14a)

The backward pass is given by:

Vi,j = minimum(Vi,j,Vi,j+1 + 3,Vi+1,j−1 + 4,Vi+1,j + 3,Vi+1,j+1 + 4) (3.15a)

∀ i = {H − 1,...,1} and j = {W − 1,...,1} (3.15b)

The computation of the distance transform from an edge image provides a basis for template-based shape matching, even in conditions where the foreground image is unclear/noisy.

**3.1.5.2 Chamfer Distance for Template Matching**

Chamfer distance matching is the process of determining the position in the search image distance transform of greatest similarity to the template silhouette image. In the case of this research the template is a hand silhouette image which is created by combining skin and motion cues.

Template matching is achieved by passing the template silhouette over the search image distance transform column-wise and row-wise. At each position of the template over the search image, the sum of all distances corresponding to pixels in the search image that overlap with edges in the template is computed.

The summed value is known as the distance measure and the region with the smallest sum value is considered the closest matching position.

Figure 3.9: Flowchart of the Hierarchical Chamfer Matching Algorithm

**3.1.5.3 Hierarchical Template Matching**

Chamfer matching does a good job of detecting a target object if the size of the object in the search image is exactly the same as that of the target image. If the target object changes size in the search image, such as if the hand moves closer to or further away from the camera, or as is observed with variations in users, matching needs to be done at several diﬀerent scales. Scanning the image at various scales can be very computationally expensive.

Hierarchical Chamfer matching oﬀers a solution to this problem. It provides a coarseto-ﬁne resolution search using a pyramid of images at various resolutions to boost the chamfer matching process. This pyramid of images, also known as a resolution hierarchy, consists of multiple duplicates of the original search image at various resolutions.

Figure 3.9 depicts a ﬂowchart of the Hierarchical Chamfer matching algorithm. Chamfer distance matching is initially executed on the lowest resolution image to obtain an approximation for the general region of the target object in the search image. The process is repeated on a higher-resolution image down the next level of the hierarchy, limiting the search in the new image only to the area determined in the previous level. This process is repeated until the search is performed on the original image to locate the target object.

The main advantage of using this approach is the reduction in computational cost, as the number of scans is strategically reduced.

**3.1.6 Connected Component Analysis**

Connected Component Analysis (CCA) is an algorithm for the detection and extraction of the contours of objects in an image [19]. The technique, also known as Connected Components labeling, passes over an image at a pixel-by-pixel level to search for all connected pixel regions. It can be performed on binary images, as well as grayscale images. Regions are said to be connected when adjacent pixels share the same set of intensity values V . In the case of a binary image, V = {255}. The 4-connectivity and 8-connectivity labeling operators are shown in Figure 3.10.

Figure 3.10: An example of the 8-connectivity labeling operator [16].

In order to compute the connected components of a binary image using the 8-connectivity operator, each pixel p that has an intensity value V = 255 is scanned and labeled. If V = 255 for the current p, the 8 neighbours of p which have been encountered before in the scan are examined and p is labeled using the following criteria:

1. If all the neighbours of p are of the intensity value 0, then assign a label q.

2. If all of the neighbours of p possess the value 255, then assign a label p.

3. If more than one of the neighbours have the intensity value of 255, assign the label of one of the neighbours to p and keep track of the equivalences.

Once the scan has been completed, a secondary pass is carried out to replace each label resulting from the ﬁrst pass with its equivalent class label. Pixels labeled p are considered as foreground. Connected foreground blobs are then labeled as separate foreground objects, each with a unique index.