International Journal of Research Studies in Science, Engineering and Technology
Volume 1, Issue 5, August 2014, PP 8-15
ISSN 2349-4751 (Print) & ISSN 2349-476X (Online)

Multipiple Face Detection and Tracking using Adaboost and Camshift Algorithm

Mr. Ashwith Kumar S K[1], Dr. Suresh D S[2],Mr.Sanjeev Kubakaddi[3]

[1] PG Scholar, Department of ECE,
Channabasaveshwara Institute of Technology, Gubbi, India
ashwith16@gmail.com
[2] Professor and Head, Department of ECE,
Channabasaveshwara Institute of Technology, Gubbi, India
director@citumkur.org
[3] CEO, itie Knowledge Solutions Bangalore, India

Abstract: *Recent heightened security concerns have made the need for robust face detection and tracking more imperative, especially in the areas of security and surveillance. In addition, face detection and tracking are also crucial in applications such as teleconferencing, face and facial gesture recognition, telecommunications, robotics as well as human-computer interactions (HCI). But face detection and tracking is a very challenging and interesting problem in real time. This proposed work describes an application for automatic face detection and tracking on video streams through surveillance cameras in public or commercial places. Some situations may required to track the people those who are moving from one place to another in restricted or high alert areas. So here prototype system is designed to work with cameras for the face detection and tracking, based on the platforms CCS and OpenCV. The system adopted with Adaboost algorithm and cam-shift algorithm. This system can be used for security purpose to record the visitor face as well as to detect and track the face within the camera range. A algorithm is developed using OpenCV that can detect people's face and track them by taking input from the camera.*

Keywords: *TMS320C6748, Code Composer Studio V5, OpenCV, Haar like features, Cam-shift algorithm*

1. Introduction

The goal of this project is to provide an easier user friendly system to detect and to track human face. With the aid of a regular camera, a machine is able to detect and track a person's face.

Face detection is a process to analyze the input image and to determine the position and the orientation of face. Face detection is the base for face tracking and face recognition. The ability to reliably detect and track people and their face in real-world images is interesting for a variety of applications, such as video surveillance, human computer interaction, robotics, movie industry etc. At the same time, humans are one of the most challenging categories of object detection. A large variability in their local and global appearance is caused by various types and styles of cosmetics applied especially to their face. Face tracking is different from face detection in that face tracking uses temporal correlation to locate human faces in a video sequence, instead of detecting them in each frame independently. With temporal information, we can narrow down the search range significantly and thus make real-time tracking possible. The common face detection methods are: knowledge-based approach, Statistics-based approach and integration approach with different features or methods. The knowledge-based approach [1] [2] can achieve face detection for complex background images to some extent and also obtain high detection speed, but it needs more integration features to further enhance the adaptability. Statistics-based approach [3] [4] detects face by judging all possible areas of images by classifier, which is to look the face region as a class of models, and use a large number of “Face” and “non-face” training samples to construct the classifier..

The Adaboost algorithm [3] [4] appears in recent years; it trains the key category features to the weak classifiers, and cascades them into a strong classifier for detection of face. The method has real-time detection speed and high detection accuracy, but needs long training time.

This paper describes a system that can detect a human face in real time using haar-like features where the detection algorithm is based on Adaboost algorithm and track using cam-shift algorithm.

2. Related Work

Normally face detection is used in biometrics, as a part of a facial recognition system. Previous days the detection and tracking of face is achieved by algorithms without using hardware like development boards or separate circuits etc. But now a days some recent digital cameras also use face detection for autofocus.

Actually to detect a face, camera can be integrated into a device to monitor and detect any face that walks by. Similarly this paper shows prototype or partial implementation of this type of work. In this proposed methodology the camera is connected to the DSP board TMS320C6748 (LCDK), which is programmed using Code Composer Studio V5. The Adaboost algorithm is used here to detect the faces.

3. Description of Tools

In this section the tools and methodology to implement and evaluate face detection are detailed.

3.1. OpenCV

OpenCV (*Open* Source Computer Vision Library) is a library of programming functions mainly aimed at real time computer vision, developed by Intel and now supported by Willow Garage [5]. It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on real-time image processing. If the library finds Intel's Integrated Performance Primitives on the system [6], it will use these proprietary optimized routines to accelerate it. The library was originally written in C and this C interface makes OpenCV portable to some specific platforms such as digital signal processors. Wrappers for languages such as C#, Python, Ruby and Java (using JavaCV) have been developed to encourage adoption by a wider audience [3]. However, since version 2.0, OpenCV includes both its traditional C interface as well as a new C++ interface. This new interface seeks to reduce the number of lines of code necessary to code up vision functionality as well as reduce common programming errors such as memory leaks (through automatic data allocation and de-allocation) that can arise when using OpenCV in C[9].

3.2. Code Composer Studio

Code Composer Studio (CCS) is the integrated development environment (IDE) provided by Texas Instrument. It is based on the Eclipse framework and therefore requires a Java Runtime Environment (JRE).

3.2.1 System Requirements

To use Code Composer Studio, your operating platform must meet the following minimum requirements:

- Windows 7
- 2GB RAM
- 80GB HDD
- Intel processor

3.3. C6748 DSP Processor

Digital Signal Processors (DSP) are specific processors used for signal processing. Digital signal processing is used in many areas such as sound, video, computer vision, speech analysis and synthesis, etc. Each of these areas need digital signal processing and can therefore use these specific processors. DSPs are found in cellular phones, disk drives, radios, printers, MP3 players, HDTV, digital cameras and so on. DSPs take real world signal like voice, audio, video, temperature, pressure, position, etc. that have been digitized using an analog-to-digital converter and then manipulate them using mathematical operations.

DSPs are usually optimized to process the signal very quickly and many instructions are built such that they require a minimum amount of CPU clock cycles.

The C6748 is based on Texas Instruments very long instruction word (VLIW) architecture. This processor is well suited for numerically intensive algorithms. The internal program memory is structured so that a total of eight instructions can be fetched every cycle. The C6748 has a clock rate of 375 MHz and is capable of fetching eight 32-bit instructions every 1=375 MHz or 2.67 ns. The processor includes both floating-point and fixed-point architectures in one core.

The C6748 includes 326 kB of internal memory (32 kB of L1P program RAM/cache, 32 kB of L1D data RAM/cache and 256 kB of L2 RAM/cache), eight functional units composed of six ALUs and two multiplier units, an external memory interface addressing 256 MB of 16-bit mDDR SDRAM, and 64 32-bit general purpose registers. In addition, the OMAPL138 features 128 kB of on-chip RAM shared by its C6748 and ARM9 processor cores[7].

3.3.1 TMS320C6748 Fixed- and Floating-Point DSP

Features

- 375- and 456-MHz C674x Fixed- and Floating Point VLIW DSP
- C674x Instruction Set Features
- C674x Two-Level Cache Memory Architecture
- 128KB of RAM Shared Memory
- LCD Controller
- Two Serial Peripheral Interfaces (SPIs) Each with Multiple Chip Selects
- Two Multimedia Card (MMC)/Secure Digital (SD)Card Interfaces with Secure Data I/O (SDIO) Interfaces
- Two Master and Slave Inter-Integrated Circuits[8]

Applications

- Currency Inspection
- Machine Vision (Low-End)
- Biometric Identification

Fig 1. *TMS320C6748 LCDK kit*

4. Face Detection and Tracking

4.1. Face detection

The main function of this step is to determine whether human faces appear in a given image, and where these faces are located at. The expected outputs of this step are patches containing each face in the input image. In order to make further face detection system more robust and easy to design, face alignment are performed to justify the scales and orientations of these patches. Besides serving as the pre-processing for face recognition, face detection could be used for region-of-interest detection, retargeting, video and image classification, etc.

4.1.1 Feature Extraction:

After the face detection step, human-face patches are extracted from images. Directly using these patches for face recognition have some disadvantages, first, each patch usually contains over 1000 pixels, which are too large to build a robust recognition system Second, face patches may be taken from different camera alignments, with different face expressions, illuminations, and may suffer from occlusion and clutter. To overcome these drawbacks, feature extractions are performed to do information packing, dimension reduction, salience extraction, and noise cleaning. Here feature extraction takes place using haar cascade classifiers .

4.1.2 Face Detection using Haar Cascades

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, haar features shown in below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle.

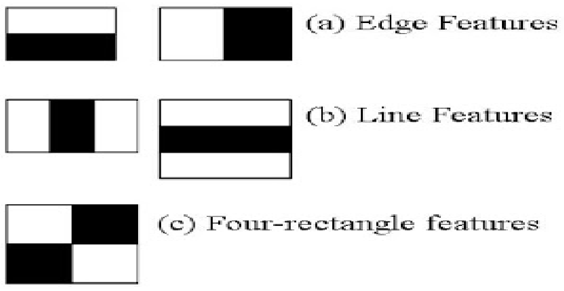

Fig 2. *Haar like features*

Now all possible sizes and locations of each kernel is used to calculate plenty of features. For each feature calculation, we need to find sum of pixels under white and black rectangles. To solve this, they introduced the integral images. It simplifies calculation of sum of pixels, how large may be the number of pixels, to an operation involving just four pixels. But among all these features we calculated, most of them are irrelevant. For example, consider the image below. Top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applying on cheeks or any other place is irrelevant. So we can select the best features out of 160000+ features using Adaboost.

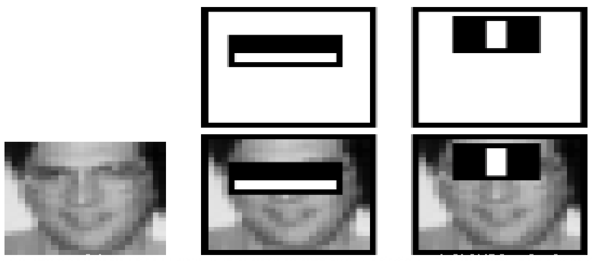

Fig 3. *Feature extraction using haar*

For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. But obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that best classifies the face and non-face images. In an image, most of the image region is non-face region. So it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot. Don't process it again. Instead focus on region where there can be a face. This way, we can find more time to check a possible face region.

4.1.3 Adaboost algorithm

In 1995, Freund and Schapire first introduced the AdaBoost algorithm [1]. It was then widely used in pattern recognition.

The AdaBoost Algorithm

a. **Input**: Give sample set

$$s = (x_i - y_i) \ldots\ldots (x_n - y_n)$$

$x_i \in X$, $y_i \in Y = \{-1,+1\}$ Number of iterations T

b. **Initialize**: $W_{i,j} = \frac{1}{N}$, i = 1, 2,……… N

c. **For** t = 1,2,………T

Train weak classifier using distribution W

ii) Calculate the weight (w) training error for each hypothesis

$$h_n \varepsilon_n = \sum_{i=1}^{N} W_{i,j} \mid h_i - y_i \mid$$

iii) Set

$$a_t = \frac{1}{2} \log \left(\frac{1 - \in_t}{\in_t} \right)$$

iv) Update the weights:

$$W_{t=1,i} = 1 + \frac{W_{i,t}}{Z_t} \times \begin{cases} e^{-at} \\ e^{at} \end{cases}$$

$$= \frac{W_{t,i} \exp(-a_t y_i h_t(x_i))}{Z_t}$$

Z_t is normalization constant

d. **Output**: the final hypothesis, also the stronger classifier.

$$H(x) = sign\left(\sum_{t=1}^{T} a_t h_t(x) \right)$$

For this they introduced the concept of Cascade of Classifiers. Instead of applying all the 6000 features on a window, group the features into different stages of classifiers and apply one-by-one. (Normally first few stages will contain very less number of features). If a window fails the first stage, discard it. We don't consider remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region or else non face region.

4.2. Face tracking:

In dynamic scenes, tracking is used to follow a face through the sequence. In order to incorporate the face changes over time, in terms of changes in scale, position and to localize the search for the face, it is essential to exploit the temporal correspondence between frames. Tracking [10] exploits the temporal content of image sequences.

Face tracking can be divided into two categories 1) head tracking and 2) facial feature tracking. Feature tracking methods track contours and points or follow eyes and mouth, and require independent trackers for each feature. Head tracking methods use the information from the entire head and can be region-based, color-based or shape-based [11]. Color-based approaches are not robust to lighting changes and approaches that use information from the entire head are, in general, unable to handle occlusion.

Tracking involves prediction and update for which filters like Kalman filter and Condensation filter have been used. Tracking approaches can also be model-based, for example, using statistical models, or exemplar-based (although only specific features of the face, e.g., lips have been tracked). A combination of feature and head tracking methods,together with filtering, have tried to eliminate the problems of the individual approaches.

4.2.1 Face tracking:

CAMshift is called Continuously Adaptive Mean Shift based on the mean shift algorithm[9]. CAMshift uses the Hue channel to trace objects since by using the Hue channel based on HSV color model, objects with different colors can be recognized. Based on the color information, CAMshift tracks objects faster and consumes relatively little CPU resources. Lower computing resource requirement enable CAMshift to become a one of real-time face tracking algorithms.

The CAMSHIFT algorithm can be summarize with these steps:

1. Choose the initial region of interest, which contains the object we want to track.
2. Make a color histogram of that region as the object model.
3. Make a probability distribution of the frame using the color histogram. As a remark, in the implementation, they use the histogram back projection method.
4. Based on the probability distribution image, find the center mass of the search window using mean-shift method.
5. Center the search window to the point taken from step and iterate step 4 until convergence.
6. Process the next frame with the search window position from the step 5.

The histogram is the tracked object's color probability map. Because in the project we focus on the face tracking, the hue channel of the face shown in the Fig 5 is used to illustrate what is the histogram. In the first step, the whole area of tracked object is scanned and the map which record how many pixels have a certain hue value in the tracked object area is built. And then, CAMshift finds out the peak the number of pixel in a certain hue value and normalizes the map into skin color probability map or histogram shown in the Fig 4. The horizontal bar of the Fig 4 is hue value and the vertical bar is the skin probability. According to the histogram, the popular color on the face is revealed.

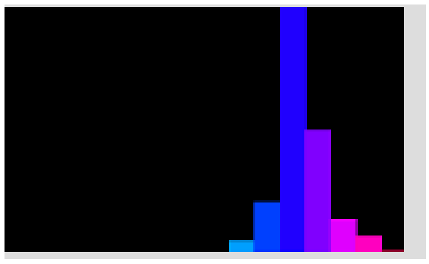

Fig 4. *Histogram*

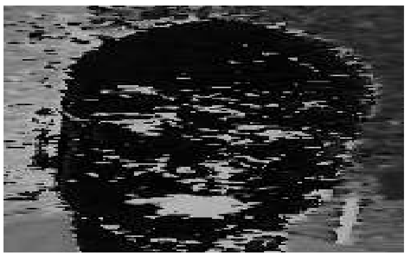

Fig 5 *H channel of the face*

Based on histogram, the skin probability of the next frame can be calculated in this way. In the first, the next frame is converted into H channel which represents the color information of the tracked object. And then looking up the histogram, we will get the skin probability for each pixel shown in the Fig 6 according to its hue value. The skin probability of the whole picture is obtained for the next step to find out the center of the face.

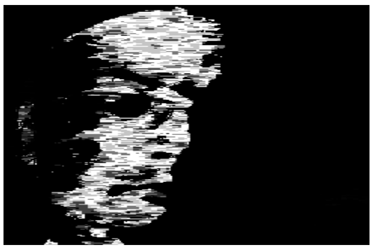

Fig 6 *The skin probability*

Based on the skin probability of the next frame, the peak probability as the center of the face is estimated by the zero moment and first moment [2]. The zero moment (M00) and first moment (M01 and M10) are calculated by the equations shown below.

$$M_{00} = \sum_{x,y} I(x,y)$$

$$M_{10} = \sum_{x,y} xI(x,y)$$

$$M_{01} = \sum_{x,y} yI(x,y)$$

In this way, the center of face can be estimated by the following equations.

$$x_c = \frac{M_{10}}{M_{00}}$$

$$y_c = \frac{M_{01}}{M_{00}}$$

Finally, the convergence of the center of the face (xc ,yc) is checked. If the center of the face is very close to the old center, the convergence is reached; otherwise, all the equations should be calculated again based on the new center.

5. Experimental Setup

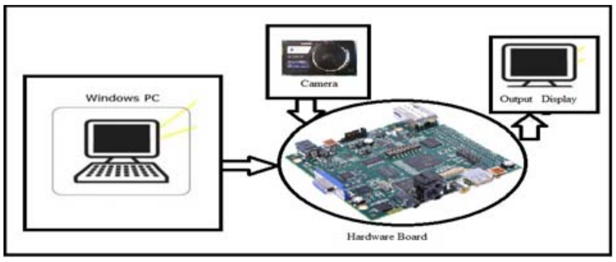

Fig 7. *Experimental setup*

Above figure 7 shows the working set up of the required model. Here developed algorithm is dumped to the DSP kit using the emulator XDS100v2 and Code Composer Studio. Camera is interfaced with DSP kit which is used to capture the video. The processing takes place in the DSP processor and the face detection output displayed in the monitor

To implement face detection, the tools required are:

5.1. Software Required

Code Composer Studio V5,C6000 SYS/BIOS by Texas Instrumentation.

5.2. Hardware Required

PC preferably running windows 7, Texas Instrumentation DSP LCDK kit TMS320C6748, JTAG Emulator XDS100v2 and a camera.

6. Results

The input video is captured by camera which connected to the hardware with the help of connector. The applied face detection algorithm, generate rectangle blue class which keeps track of the face coordinates. This results are shown in below figures 8.

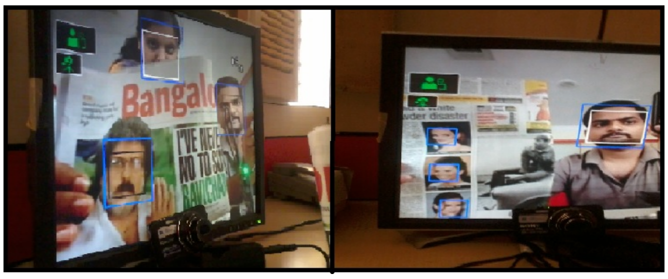

Fig 8. *Result of Face Detection and tracking in video*

7. Conclusion

Prototype system for automatic face detection and tracking is successfully implemented and tested. The test results show that the detection method used in the project can accurately detect and track human face in real time. This project shows the intersection of Image processing and embedded systems, by using OpenCV and TMS320C6748 (LCDK) DSP kit real time implementation is possible. Future Work: Along with face detection, eye, mouth and other object detection may also be implemented

References

[1] A. Faizi (2008), “Robust Face Detection using Template Matching Algorithm,”, University of Toronto, Canada.

[2] P. Feng (2004), “Face Recognition based on Elastic Template,”, Beijing University of Technology, China.

[3] Z. Zhang (2008), “Implementation and Research of Embedded Face Detection using Adaboost”, Shanghai JiaoTong University, China.

[4] K.J. Wang, SH.L. Duan & W.X. Feng (2008), “A Survey of Face Recognition using Single Training Sample”, Pattern Recognition and Artificial Intelligence, China, Vol. 21, Pp. 635–642..

[5] CH. Y. Lu, CH.SH. Zhang & F. Wen (1999), “Regional Feature based Fast Human Face Detection”, J. Tsinghua Univ. (Sci. and Tech.), China, Vol. 39, Pp. 101–105.

[6] Open Source Computer Vision Library Reference Manual –intel

[7] “Digital Signal Processing Application on the Texas Instrument C6748 Processor” by Julien Osmalskyj and Jean-Jacques Embrechts(2014)

[8] http://www.ti.com.cn/cn/lit/ds/symlink/tms320c6748.pdf

[9] Gary Bradski & Adrian Kaehler O□Reilly (2008), “Learning OpenCV”, O’REILLY Media

[10] J. Yang and A. Waibel, “A Real-Time Face Tracker,” Proceedings of WACV’96, pp. 142-147

[11] D. DeCarlo and D. Metaxas, “Deformable Model-Based Face Shape and Motion Estimation,” IEEE Proc. of ICFG, 1996