Object Detection Algorithm for Real Time Embedded Systems

1M. S. Chelva, 2V. S. Halse

1Research Scholar, SRTM University, Nanded, Maharashtra, India
2Karnataka States Women’s University Vijaypur, Karnataka, India

Email: 1mschelva@gmail.com, 2drsvhalse@rediffmail.com

Abstract—Vision has become a part of every smart embedded system. The recent trends in automobile and automation systems have camera as an integral part of the design and development. Object recognition is one of the unique abilities that human beings possess. Humans are able to identify the object because of the power brain despite of the variations in the appearance of the object because of its color, position or the camera view. The recognition process, either generative or discriminative, is then carried out by matching the test image against the stored object representations or models. The development in technology has promoted implementation of video processing on an embedded platform to make systems portable and friendly. In this paper we propose a new algorithm to detect a still object in real time system with soft constrains. The algorithm is implemented on an embedded processor.

Key Words— Object detection, Real Time, JPEG, DCT, Embedded processor

I. INTRODUCTION

Image processing is conversion of an image into digital form and performing mathematical operations on it, through which we get an enhanced image or can extract some useful information from it. It is a type of processing signal in which input is image and output may be image or characteristics of that image. Image Processing includes two dimensional signals which we treat as images while applying already set signal processing methods to them. Image processing includes the following steps,

• Importing the image
• Image pre-processing
• Analizing the image
• Output stage in which result can be altered image or report of analyzed image.

Now a days there is a wide variety of applications in Image Processing which includes,

1. Intelligent Transportation Systems
2. Remote Sensing
3. Moving object detection and tracking
4. Defense surveillance
5. Biomedical Imaging techniques
6. Automatic Visual Inspection System

Among these all applications, Object detection is very challenging tasks since they have to deal with various problems such as different views of an object, various light and surface conditions as well as noise caused by image sensors. Object detection involves detecting objects from a particular class in an image. Though some algorithms SIFT [20], SURF [1] are available, but these are complex and require powerful hardware in order to operate in real time.

The efficiency of object detection and recognition is highly dependent on factors like luminance and position of the object. The presence of object in an image frame requires efficient detection of the object and hence a major issue to understand whether object is present or not or else wrong object may get detected. The object detection efficiency can be increased by providing some prior knowledge of the object to the algorithm. This may include the features of the object like corners, straight lines, etc, but it is a time consuming process. The object recognition issues may involve range of viewing angle of camera, segmentation cues, etc. Also, if the number of prior known object is large or even if the objects in an image frame are many in numbers, it takes long time to search and detect the object required. The interpretation tree technique [21] is used to find the overlapping parts of the objects. A second approach called as invariants, also uses the features of an object to identify it but here these features are global indices and hence do not change with the physical dimension of the object. One such approach is Scale Invariant Feature Transform (SIFT) [20]. Using the image based approach for object detection; multiple frames are used to confirm the detection of object. In image based object recognition, the detection is efficient if the object view from all the directions in the 2D space is available in the form of look-up table in the memory [22]. This will just be a comparison of the present image captured by the camera to the images stored in memory. But, an object can be looked around with infinite viewing angle and hence the reference images for a single object may be large in number. Moreover, as the images are bigger in size, it is not feasible to store the image-look up tables in the memory.
II. LITERATURE REVIEW

The SURF-Based System Designs Accelerated by FPGA Logic” [1] gives an overview of two embedded systems for object detection and pose estimation using point features. The feature detection step of the “Speeded-up Robust Features (SURF)” algorithm is increased by a special IP core. The first system is used to perform object detection and is implemented on Virtex-5 FPGA. The second system consists of an ARM-based microcontroller and intelligent FPGA-based cameras which support the main system [1].

In [2] the authors propose an algorithm that can be implemented on an embedded platform for embedded application such as a robot capable of tracking an object in 3-D environment. It is a real time operating system (RTOS) which runs the Digital Image Processing Algorithms. These algorithms are used to extract the information from the images. The camera connected on USB bus captures images on the ARM9 core running RTOS. Depending upon the information extracted, the further process is carried out. The camera is a CMOS USB-camera module having a resolution of 0.3MP. Video4Linux API’s provided by kernel are used to capture the image, and after decoding, the object location is detected using image processing algorithms. The embedded Linux kernel supports multitasking feature and ensures that the task is performed within the real time constraints. The OS makes system flexible such as interfacing new devices, handling the system and storage of data.

Processors that meet hard real time constraints are used in embedded environments, with low power and performance trade-offs. Object detection applications are associated with real-time performance constraints that originate from the embedded system [2]. Embedded systems using ARM32-bit microcontroller has the feature of image or video processing by using the variety of features and classification algorithm have been proposed for object detection.

Shape based object detection is the popular method to identify the objects in image frame. In an image with less number of objects, this approach works fine but when the image is too much crowded by objects, then it is difficult to detect the shape as there are many occlusions and shades [15]. Another simple approach is colour segmentation. The efficiency of such algorithm changes with the environment of application, mainly intensity of light affects to a great extent but on the contrary it has an advantage of least computational complexity [16]. A fixed template matching approach is useful when the object shape do not vary in the application. This is dependent on the orientation of camera with respect to the object position. Template matching is carried out in two ways. First is to match the complete image frames, also called as image subtraction technique [17]. This approach is limited by the conditions like light intensity where system is to be implemented and the orientation of camera with respect to object position. Second way to perform template matching is to find correlation between the current image frame captured from the camera and the standard template stored in the memory [17]. A deformable template matching approach is used when the shape of object changes due to rigid or non-rigid deformations. A good approach is developed by Zhong et al. to detect objects using prototype based deformable template models [18]. Here, deformities are introduced in the prototype and deformed template is obtained and used as reference for detection. A classifier based approach is presented by the Liu et al. [19]. The image frame is divided into multiple small blocks of uniform size. If the centre pixel belongs to the image background then it is called as background block or else it is an object block. These background blocks are then used to train the algorithm to detect the background and remove it. But due to classifier error the edges of the object are not detected perfectly and hence some error is introduced. A novel method of motion based object detection was developed by Viola and Snow [11]. The authors used various motion filters to efficiently detect the pedestrians. One of the techniques is to consider the inter-frame difference followed by thresholding to detect the temporal changes either at pixel level or block level. The algorithm detects the foreground only if it is in motion otherwise no object would be detected.

The methods for object detection can be classified into two major categories, generative [5, 6, 7, 8, 9] and discriminative [10, 11, 12, 13, 14]. Prior is the probability model for the pose variability of the objects together with an appearance model for the image appearance on a given pose, together with a model for background, i.e. non-object images. The model parameters can be estimated from training data and the decisions are based on ratios of posterior probabilities. The later builds a classifier that can discriminate between images (or sub-images) containing the object and those not containing the object. The parameters of the classifier are selected to minimize mistakes on the training data, often with a regularization bias to avoid over fitting.

A widely used technique for image compression is the discrete cosine transform (DCT) and is selected as the standard for JPEG making it more popular in the image processing field. DCT transforms the data into frequency domain and inherently performs compression.

(a) Input Image
The Fig.1 shows the comparison of Discrete Fourier Transform (DFT) and Discrete Cosine Transform (DCT) of an image. Fig.1a is the sample image and Fig.1b and Fig.1c shows the amplitude spectrum under DFT and DCT respectively. The histogram of DCT is more concentrated and compressed as compared to that of the DFT. This is the main reason that DCT is preferred in image compression.

DCT is an orthogonal transform, which has a fixed set of basis function. DCT has many advantages such as the ability to pack energy in the lower frequencies for image data; ability to reduce the blocking artifact effect and this effect results from the boundaries between sub-images become visible [3].

Transform coding constitutes an integral component of contemporary image/video processing applications [3]. Transform coding relies on the premise that pixels in an image exhibit a certain level of correlation with their neighboring pixels [4]. Similarly in a video transmission system, adjacent pixels in consecutive frames show very high correlation [4]. Consequently, these correlations can be exploited to predict the value of a pixel from its respective neighbors. A transformation is, therefore, defined to map this spatial (correlated) data into transformed (uncorrelated) coefficients. Clearly, the transformation should utilize the fact that the information content of an individual pixel is relatively small i.e., to a large extent visual contribution of a pixel can be predicted using its neighbors.

The discrete cosine transform has a property to concentrate high energy pixels in one corner. These high energy pixels contain almost complete information of the image. The Fig.2a shows the input image matrix and Fig.2b shows the matrix after applying DCT to input matrix. It can be clearly seen that the high energy pixels are concentrated in one corner in Fig.2b. Hence while decoding; the JPEG image is decoded only to lower resolution by considering only the higher energy part matrix. The processing time of the image, which includes preprocessing and object detection, is now reduced to small size image which is decompressed and used for further processing.

![Figure 1: Comparison in DCT and DFT of an Image](image1)

**Figure 1:** Comparison in DCT and DFT of an Image (a) Input Image (b) DFT of the image (c) DCT of the image

![Figure 2: Input Image and its Bit Map](image2)

**Fig 2 (a) Input Image and its Bit Map**

![Figure 2: Compression after applying DCT](image3)

**Fig 2. (b) Compression after applying DCT**

## III. METHODOLOGY

The methodology of spatial correlation is used to detect the object in the image frame. A mask image of smaller size compared to the image frame is defined and stored in the memory to use it as reference. For each frame, the correlation of the object mask and image frame is evaluated by traversing the mask over complete image frame. A center of the mask is defined as shown in the Fig.3. The correlation is calculated for every location on the image frame as shown in formula below. CORR(x, y) represents the output matrix, IMG is the image matrix and MSK is the mask of the object to be detected. For a certain (x, y) coordinate if the CORR(x, y) is less than a threshold value then the object detected.
\[
\text{CORR}(x,y) = \sum_{s=\text{a}}^{a} \sum_{t=\text{b}}^{b} (IMG(x + s, y + t) - MSK(s,t))
\]

**Figure 3: Spatial Correlation**

A. System Description

The camera based system that is used for implementation is ARM9 as its main processing unit [20]. The objective is to develop a standalone system to detect an object of some defined shape and further actuate the actuators as required. The image is captured by the camera on receiving command from the host unit. The camera sends the image data compressed in JPEG format. The image buffers while capturing image frame and processing are created in the SDRAM. If the location of object is detected, then a command is sent to the actuators through the GPIO port of the processor. Actuators are interfaced using the driver circuitry to handle high current.

B. Algorithm

The algorithm we developed and implemented is for a specific application. We assume that the object to be detected would be more or less rectangular and its color is black. The object is still, kept at a fixed location. The camera used is 0.3MP CMOS USB camera which provides the output frame in JPEG format.

The JPEG uses discrete cosine transform (DCT) to compress and encode the image. To get the image in raw pixel format inverse DCT is applied while decoding the image. The JPEG image compression technique consists of 5 functional stages.

1. A color space conversion from RGB to YIQ
2. Subsampling the chrominance component in the pixel data
3. Representation of the data in frequency domain using discrete cosine transform
4. Quantization of the data
5. Entropy encoding of the quantized data to ease the storage

C. Flow Chart

The following flowchart shows the algorithm used to detect the object.

**Figure 4: Flowchart**

d) Proposed Algorithm

The pseudo-code of the implementation is as follows:

i. Initialize processor

ii. Configure camera to give output frame in JPEG format

iii. Store object mask in SDRAM

iv. while(1)

malloc(input buffer)

input buffer = clickimage(camera device ID)

new image = decodeJPEG(input buffer, Image Resolution)

raw image = gray scale image of new image

r = rows of raw image
c = pixels in one row

for i = 1 to r

for j = 1 to c

get correlation between image and mask with mask center as (i, j)

if (correlation > threshold)

object present = true

break;

else

object present = false
end if

end j loop

if (object present == true)

object location = (i, j)

break;

end if

end iloop

free(input buffer)

free(raw image)

free(new image)

end while
IV IMPLEMENTATION

The algorithm is implemented on EP9302 SOC9 (ARM9 processor) which is a high-performance, low-power, RISC based single-chip computer with a maximum operating clock rate of 200MHz [23, 24]. It is a low power, simple, elegant and fully static design processor that is suitable for cost- and power-sensitive applications. The camera used here is the CMOS camera with the resolution of 640x480. The camera is interfaced to the system via USB interface. The embedded OS on board makes it easy to use the USB interface for the camera.

The Fig.4 (a) shows the input image captured using a CMOS camera of 0.3MP. The image is stored in an onboard SDRAM. The JPEG image decoded by considering the higher energy region of its resolution is shown in Fig.4 (b). The mask shown in Fig.5 is used to detect the object in the decoded image frame. Once the correlation at a certain location in image frame is found to be greater than a threshold value, it is considered as the object and hence the location is tracked properly. The detected object location is seen the output image Fig.4(c). Similar results of the algorithm with different input images are show in Fig.5 and Fig.6.

![Image 4](image4.png)  
**Figure 4:** (a) Input Image (b) Decoded Image (c) Detected Object Location

![Image 5](image5.png)  
**Figure 5:** Mask for co-relation

![Image 6](image6.png)  
**Figure 6:** (a) Input Image (b) Decoded Image (c) Detected Object Location

![Image 7](image7.png)  
**Figure 7:** (a) Input Image (b) Decoded Image (c) Detected Object Location

V. CONCLUSION

This paper presents the study of various techniques used to detect object. The acceptance of a particular algorithm is dependent on the environment conditions in the area of application. As the application here is implemented in an environment where light variations are very minimal and the orientation of object with respect to camera is almost constant, a correlation algorithm is used to find the location of the object. The algorithm described in this work is implemented on a standalone embedded system and uses the inherent property of DCT while decoding the image which improves efficiency of algorithm with respect to time as well as memory required to store the image and also executes in real time. It processes the frames at 18 fps on processor and hence is suitable for real time embedded application. Decoding the input image to lower resolution image frame results in requirement of less onboard memory and also increases the processing time.

REFERENCES


[16] Sebastien Lef evre and Emmanuel Bouton and Thierry Brouard and Nicole Vincent,” A new way to use hidden markov models for object tracking in video sequences “2003


[23] LTD, ARM. "ARM920T technical reference manual." ARM Ltd.