International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 10, October - 2013

Detection` of Embedded Human Tracking by DM3730

G. Harshavardhan Reddy ,
(M.Tech) Embedded systems,
Aurora's Technological And Research Institute,
Hyderabad.

Dr. P.S.Sarma,
professor. ECE,
Aurora's Technological And Research Institute,
Hyderabad.

***Abstract*—**Human detection and pose estimation are important tasks in Computer Vision, with applications ranging from content based image retrieval, surveillance, and vision based robotics. These two tasks are often inter-linked: detection is usually necessary for pose estimation which has been shown to increase its accuracy. They are also heavily involved in human motion analysis and segmenting humans from a complex environment. In the embedded world, computer vision applications have to fight with limited processing power and limited resources to achieve optimized algorithms and high performance. This paper presents work on implementing a human tracking system on both Intel based PC platform and embedded systems to optimize the algorithms for high performance. The algorithms are benchmarked on the Intel platform processor and BeagleBoardxM based on low-power Texas Instruments (TI) DM3730 ARM processor. Functions and library in OpenCV which developed by Intel Corporation was utilized for building the human tracking algorithms.

Keywords-Human tracking; opencv; histogram; beagleboard;

I. INTRODUCTION

Video surveillance systems are becoming common and their deployment is becoming more widespread as societies become more complex and population continues to grow. The usage of cameras, computers and even embedded system for surveillance system is growing too. This is the result of the need for monitoring a specific area of interest or an area of high security. This is especially true when the area that needs to be monitored requires constant monitoring and it is difficult to have human operators constantly monitoring the surveillance system. The deployment of video surveillance in a growing complex society has uncovered several problems. These problem are, but not limited to, recognizing human behavior and tracking a suspicious character in the surveillance system. Traditionally image processing algorithms are implemented on a desktop computer which has powerful processors and large amount of memory resource. In contrast, an embedded system has limited processing power and limited memory resources.The implementation of computer vision algorithms is computationally intensive and resource exhaustive. The performance can be improved by selecting the proper mobile processor by using different on-chip resources.

In this paper, we implemented a human tracking system and ported to an embedded platform. The system is able to track the human across the camera field of view by comparing the human color histogram in consecutive frame. The selected non embedded platform is the Intel Duo Core processor with 2.13GHz clock and 4 GB of memory. The embedded platform selected is the BeagleBoard xM which running at 1GHz DM3730 processor from TI and has 512 MB of LPDDR RAM POP memory. It has asymmetric dual-core architecture with an ARM Cortex-A8 and TMS320DMC64X+ Digital Signal Processor (DSP).

Recently, researcher has benchmarked and optimized their computer vision algorithms in FPGA platform and Digital Signal Processor platform. To increase the performance and use the available resources to speed up the computation in DM3730 BeagleBoard xM processor, OpenCV algorithms is used. OpenCV is an open source computer vision library developed by Intel Corporation.

The rest of the paper is organized as follows: Section 2 describes the system overview of the platform to be tested. Section 3 describes the architecture of the human tracking system.

Section 4 describes the system setup and implementation of the algorithm into the embedded platform. Section 5 describes the experiments conducted for this paper along with the experimental results. Finally, section 6 draws a conclusion for the paper.

II. SYSTEM OVERVIEW

A. Embedded Platform

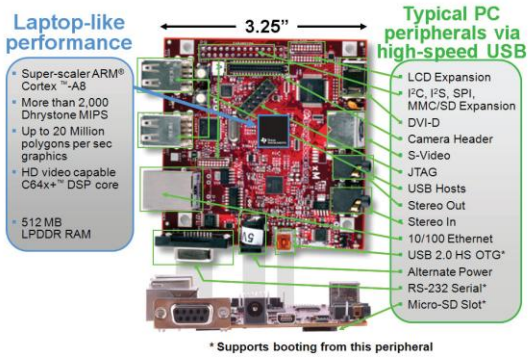
The BeagleBoard xM is a low-cost, low power, fan-less open source hardware single board computer produced by Texas Instrument in association with Digi-Key. BeagleBoard xM is the modified version of BeagleBoard which has faster CPU core (clocked at 1GHz compared to 720MHz) and more RAM (512 MB compare to 256 MB). The BeagleBoard xM was designed with open source development in mind and as a way of demonstrating the Texas Instruments DM3730 system-on-a-chip. The board uses up to 2 W of power and because of the low power consumption, no additional cooling and heat sinks are required.

The BeagleBoard xM is designed to address the Open Source Community. By eliminating all of the on-board peripherals and by providing standard expansion buses like high-speed USB 2.0, Ethernet port and HDMI port, developers and researchers can bring their own peripherals and expand the board ability what they want. It has been equipped with a minimum set of features to allow the user to experience the power of the processor.

1) MPU subsystem: Multiprocessor Unit (MPU) subsystem handle the transaction within the ARM core, NEON SIMD coprocessor, cache memory and interrupt controller (INTC). The MPU subsystem intergrated the ARM subchip with additional logic for floating point operations acceleration, emulation, interrupt handling.

2) IVA2.2 Subsystem: This subsystem available at DM37x families only. BeagleBoard xM includes the high-performance Texas Instruments image video and audio accelerator (IVA2.2) based on the TMS320DMC64X+ VLIW digital signal processor (DSP) core.

3) SGX Subsystem: The 2D/3D graphics accelerator(SGX) subsystem accelerates 2-dimensional (2D) and 3-dimensional (3D) graphics applications. The SGX subsystem is based on the POWERVR SGX core from Imagination Technologies. The SGX graphics accelerator can processes Pixel data, Vertex data, Video data and General-purpose processing efficiently and concurrently.



Figure(1). BeagleBoard xM

B. OpenCV

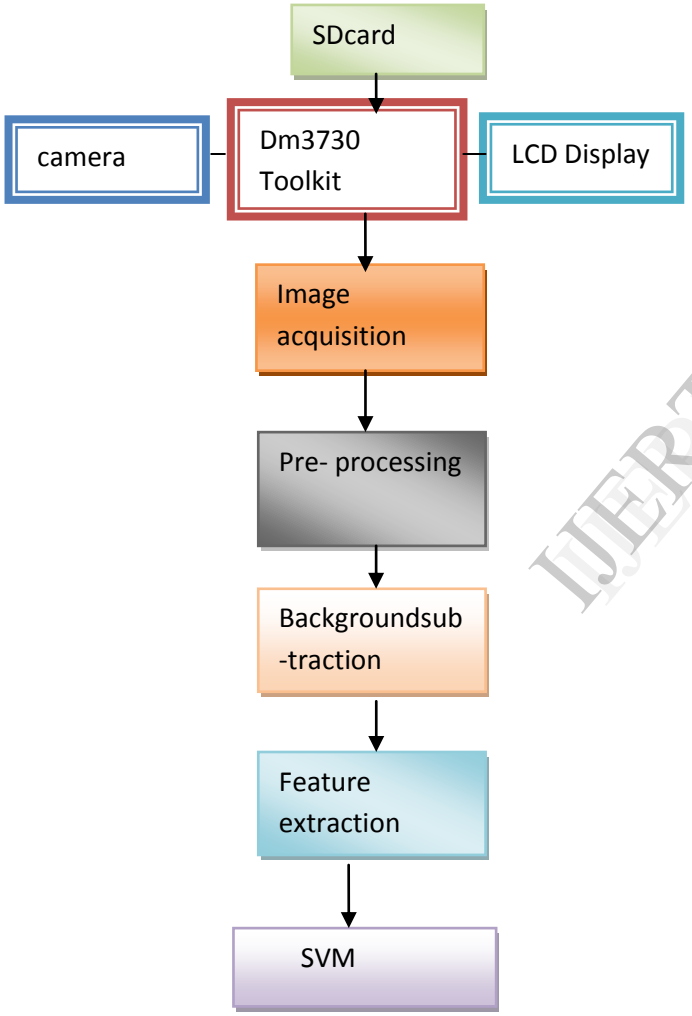
OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real time computer vision developed by Intel in year 1999 and now supported by Willow Garage. It is free for use under the open source BSD license. The library is cross-platform and is the key components to implement the human tracking algorithm in embedded platform. OpenCV library is written in C and C++.It was designed for computational efficiency and with a strong focus on real-time applications.

The main goals of OpenCV is to provide a simple –to-use computer vision infrastructure that helps people build fairly sophisticated vision applications quickly. Up to date, OpenCV library contains more than 500 functions that span many areas in vision, including general image processing functions, medical imaging, surveillance security system, camera calibration, robotic and etc.

III. ALGORITHM

This section describes the development of a visual surveillance system which has the ability to detect and track human in the camera field of view. The image capture from the camera needs to preprocess to reduce the noise. Background subtraction technique is

used to subtract out the background and leave the moving object for further analysis. Color histogram is calculates from the moving object at feature extraction stage. Last but not least, feature matching approach is used to establishing consistent labels across single camera FOV. Figure 2 shows the architecture of the human tracking system.



Figure(2):architecture of humantracking system

A. Preprocessing

The image acquisition module captured the image from the camera. Gaussian noise also called normal noise usually arise due to factors such as poor illumination or high temperature.The acquired images are smooth with a Gaussian filter.

B. Background subtraction

Background subtraction technique was traditionally applied to motion detection. It is a technology that uses the difference of the current image and the background image to detect the motion region .This method generally able to provide complete data or feature of the moving objects. For this paper, the operation starts with background image initialization by using the first frame as background image $I_{init}(x,y)$. For every current frame captured $I(x,y)$, subtract with the background image $I_{init}(x,y)$. If the pixel difference is greater than the zero, then determines that the pixels is a foreground moving object, otherwise, is the background pixels. Expression is as follow:

$$I_{0ut}(x, y)=\{1, i(x,y)-i_{init}(x,y)>0 \\ i_{back}, \text{other}$$

Where I_{out} is the binary image, I_{init} is the initial background and i_{back} is the background image.

As the complexity of the background, the binary image obtained contains the motion region, but usually not the complete human motion region. Morphological methods named closing operation used for further processing. The closing, denoted A_B, is the dilation of A by structuring element B, followed by a erosion of the result by B, thus is defined as

$$A.B=(A+B)-B$$

Firstly, dilate operation is taken at binary image to effectively fuses narrow breaks. Secondly, erode operation is taken to filter out most of the non-body motion regions while gives a complete human shape. After closing operation, small holes are eliminates and filling up the gaps of the contour, hence a accurate and complete human motion region is obtained.

C. Feature extraction

A color histogram is a popular form of representation of the object appearance one of the human feature in object tracking.It is also independent from scaling and rotation of the object and robust to the partial occlusion. This method makes a model of color histogram according to the appearance of the human shape extracted from the image. The color histogram can be calculated in a grey level space, in RGB or any other color space. In this paper, we proposed using Hue channel

of HSV color space to build the color histogram model. The HSV color space corresponds closely with the way humans describe and interpret color. Hue (H) channel represents the "color". Saturation (S) channel is the "amount" of color and Value (V) is the brightness of the color. HSV model has their information stored in hue, saturation and value respectively. Hence, by using Hue channel (color information) is sufficient for the human tracking.

The moving object is extracted out by using the binary image obtained from the background subtraction stage. The binary image will act like a mask to filter out non interested region and the histogram model will only build at the moving object region. If the pixel in the mask image is not zero, the pixel from the current frame will copy to a new image as the fore ground. Expression is as follow:

$$I_{fore}(x,y)=I_{current}(x,y) \quad \text{if } I_{mask}(x,y)\neq 0$$

Where I_{fore} is the foreground color image, $I_{current}$ is the current color image and I_{mask} is the binary image.

The foreground image I_{fore} is converted to HSV color space.1D hue channel histogram $h(r_k)$ is calculated as representation space for the appearance of the object. The histogram hue levels range from 0 until 255. Expression is as follow:

$$H(r_m)=n_m \quad m = 0,1,2,...255$$

Where r_m is the m_{th} hue level and n_m is the number of pixels in the hue image.

D.Support Vector Mechine(SVM)

The computation of a descriptor is done in the SVM according the following steps:

1) compute horizontal G_H and vertical G_V gradient of image by filtering image with [−1 0 1]

2) compute both norm and orientation of the gradient:

$N_G(X,Y)=\sqrt{G_H(X,Y)^2+G_V(X,Y)^2}$ (figure 3-b)

$O_G(X,Y)=a\ Tan\left(G_H(X,Y)/G_V(X,Y)\right)$ (figure 3-c)

3) split image into cells (figure 3-d),

4) compute one histogram for each cell (figure 3-e),

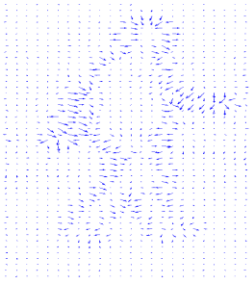
5) normalize all histograms within a block of cell.

The last step is a specificity of this descriptor, since the normalization can reduce the illumination variability.The final descriptor is obtained by grouping all normalized histograms into a single vector.

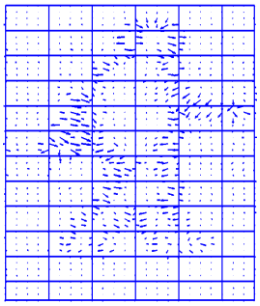
Image characterization using HoG in SVM as follows

Figure(a)original image

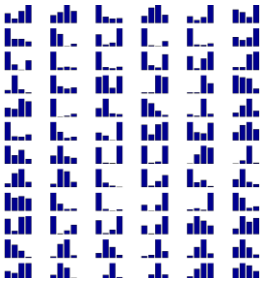
Figure(b)gradient norm



Figure(c)gradient orientation

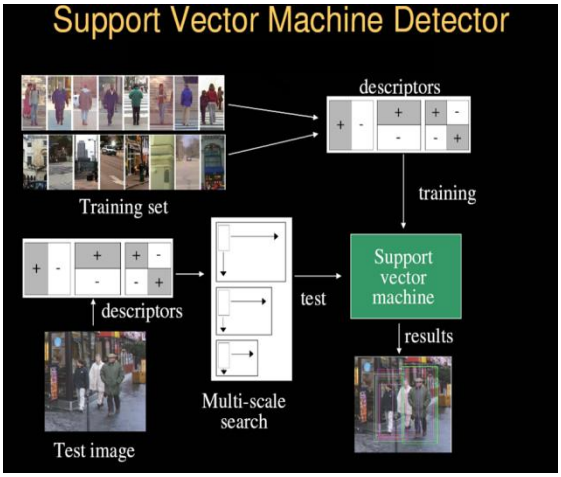


Figure(d)cell splitting



Figure(e)histogram computation

SVM measures the similarity between the histogram of a reference histogram and the histogram of the tracked image. After the matching, the similarity of moving object is determine by a its coefficients values. If its coefficients score that grater than the zero value i.e positive value indicate a good match, a unique color boundary box will be constantly labeled at the moving object.



Figure(f):working of SVM

IV. SYSTEM SETUP AND IMPLEMENTATION

A. Intel Platform

The human tracking system is implemented in Intel platform system with Ubuntu 10.10 operating system. GNU Compiler Collection (GCC) is a compiler system produced by the GNU Project supporting various programming languages. The opencv algorithm is compiled and processed using GCC and the execution time is recorded.

B. BeagleBoard xM (ARM platform)

The strip and optimized version of ARM Ubuntu distribution 12.04 OS is ported to the beagleboard. Beagleboard will boot the OS from SD card. Ubuntu operating system is chosen because it has a great community support and the kernel is much more stable after stress tested by Beagleboard running 24/7 under a heavy load. OpenCV library is build from the source code and installed so that the algorithm can be performed in beagleboard. The algorithm is compiled in the beagleboard instead of compiling at PC. The input image resolution chosen is 640x480 pixels and running at 30fps. The output of the system can be observed by connecting a monitor display having DVI-D and keyboard; mouse is connected for control use.

V. RESULT

Experiment result of the human tracking system running in Beagleboard that use the method we propose are shown in Fig

Figure. Result of human tracking algorithm

The execution time for the algorithms on Intel platform and BeagleBoard are calculated as shown in Table1.

Plat form	Time(ms)
Intelprocessor platform	21
Beagleboard xm arm plat form	400

The performance of the algorithms shows that the Intel based PC and the BeagleBoard has the average execution time of 21ms and 400ms respectively. The Intel platform has the lower execution time than the ARM platform.The reason is OpenCV libraries are highly optimized for Intel chipset architecture. The big different in execution time justified by the fact that Intel platform is non portable, high power consumption and has more resources, compared to the BeagleBoard which is highly portable, low power consumption and limited recourses.

VI. CONCLUSION

This paper implemented a human tracking system on the PC based and Embedded platform. From the results, it is suggesting that by exploiting all the available on chip hardware resources for example ARM NEON technology to accelerate the calculation of the floating point in image processing. Moreover, fully utilize the DSP core by off load the intensive calculate from the ARM processor can shorten the execution time.

REFERENCES

[1] *BeagleBoard-xM Rev C System Reference Manual*, http://beagleboard.org, Revision 1.0, April 4, 2010

[2] G. Bradski and A. Kaehler, *Learning OpenCV*, OReilly Publications,2008

[3] R.C. Gonzalez and R. E, Woods, *Digital Image Processing*, Pearson Education, 3rd Edition, 2008.

[
4] L. Zhang, Y. Liang, "Motion Human Detection Based on Background Subtraction", 2010 Scond International Workshio on Education Technology and Computer Science (ETCS), Vol. 1, pp. 284–287, March2010.

[5] J.U. Cho et al, "A Real-Time Color Feature Tracking System Using Color Histograms", International Conference on Control, Automationand Sytems 2007, pp. 1163–1167, October 2007.

[6] Embedded Linux Wiki, "BeagleBoardUbuntu", Embedded LinuxWiki[Online].Available: http://elinux.org/BeagleBoardUbuntu

[7] A. Broggi, M. Bertozzi, M. Felisa, P. Grisleri, S. Ghidoni, G. Vezzoni,C. Hilario G´omez, and M. Del Rose. Pedestrian Detection by means of Far-infrared Stereo Vision. Computer Vision and Image Under-standing, 106(2):194–204, 2007, doi:10.1016/j.cviu.2006.07.016.