Available at: www.researchpublications.org

Implementation of Background subtraction on TMS3206713 DSP processor

Mr. Sandip V Patil and Dr. Mrs. V. V. Gohokar
SSGMCE, Shegaon, Maharashtra-443101 (India)

***Abstract*—The paper presents implementation of motion detection algorithm for object tracking on TMS320C6713 DSP processor. Background subtraction which is the most popular motion detection algorithm is simulated by the differentiation of moving objects from a maintained updated background model. An object detection mask is generated which will clearly identify objects present in the image. We have used a Code Composer Studio to achieve this. We have perform our task on TMS320C6713 processor in C code, the C code is written using CCS editor, compile by compile and generated an .out file, then .out file is dump in processor memory and executed, after that result is analysis**

***Index Terms*—Integrated Development Environment (IDE), , Real Time Debugging (RTD).**

I. INTRODUCTION

In recent past, extreme research is going on video surveillance systems due to increase in terrorist activity and general social problems. This has been a inspiration for the development of a well-built and accurate automatic processing system, an essential tool for safety and security in both public and private sectors. The need for advanced video surveillance systems has motivated progress in many important areas of technology including traffic control, understanding of human activity, home safety, monitoring of endangered group, observation of people and vehicles within a busy environment, with so many application.

Now there are various motion detection and object tracking algorithm are present. Our motto here is to implement these algorithms on standalone hardware so in our first step we choose TMS3206713 processor to implement basic motion detection algorithm and verify our result. Here we choose background subtraction method for implementation to analyze our result we have taken background image and actual image. We have used a Code Composer Studio for our development.

Code Composer Studio (CCS) provides integrated development environment (IDE), which has the software tools for building and debugging programs for the TMS3206713 processor kit, i.e. the DSP board. CCS includes tools for code generation, such as a C Compiler, an assembler, and a linker. It has graphical capabilities and supports real time debugging. It provides an easy to use software tools to build and debug programs [5].

The C complier complies C source program which present in file with extension .c and produce an assembly source file with extension of .asm. The assembler assembles an .asm source file to produce a machine language object file with extension .obj. Then the linker combines the object files and objects libraries and produces an executable file with an extension .out. This executable file represents a linked common object file format (COFF). This executable file can be loaded and run directly on TMS3206713 Processor. The software development is done in three phases for the overall DSP system design process [3].

1) Coding and building: - writing code using the editor, creating a 'project', compiling and linking.
2) Debugging: - syntax checking, probe points, break points
3) Analysis:- statistics, benchmarking, real-time debugging.

Before starting actual working on CCS the appropriate DSP processor selection is important from list of processor present in CCS development environment. For selection go to Code Composer Studio Setup, select processor 6713 DSK board, Add, Save & Quit. This will start CCS with TMS320C7613 DSP Processor build environment [4].

Rest of paper is organizing as Section II, Explain Background subtraction method. Section III, contains our stepwise explanation of Foreground Detection (FD) for TMS3206713. Section VI, presents the experimental results obtained by using our FD

method. Section V, contains our concluding comments.

II. Background Subtraction Method

Background detection is an potential method of motion detection, flow chart is shown in fig. 1, It has a fixed or maintained background calculated and updated based on various algorithm used. Using Background image and current image (taken from real time camera or video stream) foreground is detected by subtracting background image from present image, Then using foreground image object detection mask is calculated based on proper threshold, foreground pixel is compared with pixel of current image if difference is more than threshold it will be taken as one otherwise zero [1], here threshold is decided base on experimental result. Also there are various algorithms present to calculate threshold at run time. Our next section explains implementation of Motion detection using background subtraction method with fix background and fix threshold on DSP320C6713 processor.

III. Foreground Detection Step Using TMS3206713

A new project is created in CCS environment as shown in fig 2. Convert our Background image and Actual image in gray scale using MATLAB then adjust its dimension, to fit it in character arrays of some desire dimension, here we have taken it as 128*128 [2].

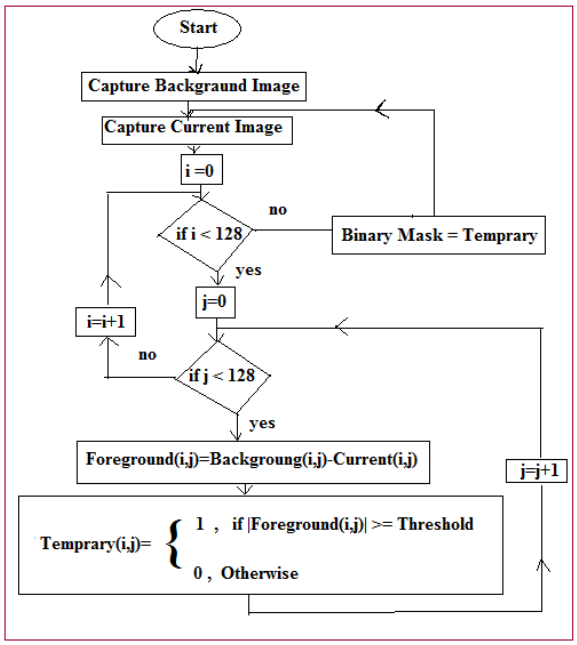

Fig.1. Flow chart of Background subtraction

Store these arrays as list of pixel values in text file with .h extension. We have name these file as 'Background.h' and 'Accual.h' and in these files assign these values to 'char variable_name[128][128] ={pixel value};' respectively, here chose a proper variable name. create new text file and save it with .c extension and written our C code in it for subtracting background image variable from actual variable and the difference is store in one more char array of 128 by 128 define as global variable, here it is define with name 'out_img' it should be of same size of our images store in .h file then we calculate the difference of background image and actual image pixel by pixel and store it in respective pixel of 'out_img' and compared it with some threshold value in C code and if it is less than threshold value then assign it to zero else assign it to 255 which is nothing but we are generating a binary object mask. We include our created .h files in C code so that the global variable should get created for background and actual image, which is used in C code for difference calculation and our mask variable also define in C code globally.

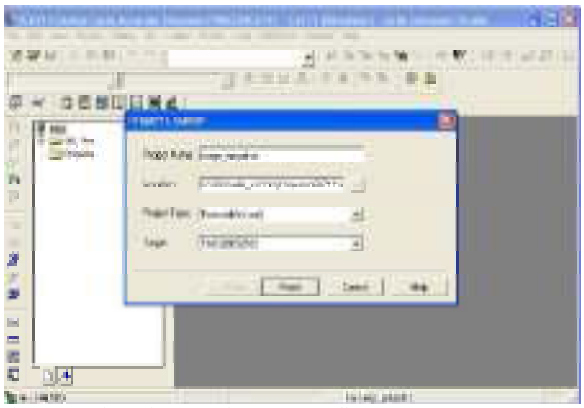

Fig.2. Creation of new project CCS studio.

Fig.3. Browser window for adding support file.

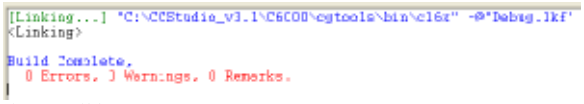

Fig.4. Build status.

Now we include some library header file for our C code these are 'stdio.h' and 'math.h' and add printf statement with some message as a last statement of our main function, which will inform us execution is completed at the time of execution, here the C code work is done. Add the C file in project, remember file should be save in our project folder. Add required support file. select the file which we need to add as show in fig 3. Here we add two library files with .lib extension, rts6700.lib and img62x.lib, these files are support files. To specify a memory allocation at the link time, we need to add command linker file 'sinewave.cmd', it contains memory rage of region and it's type. Now We are ready to build the project, build it, it will get compile and link, if any error is there then rectify the same. You will get build completion status as shown in fig 4. After successful build of our project the executable .out file will be created which is present in our project folder inside the 'Debug' folder. Flash our executable file with .out extension in TMS3206713 DSP processor, execute program [4]. After successful execution of our program, we will get our message in message window. Now it's time to analysis our result to view the output and input images for analysis, use the graph option from CCS environment as shown in fig 5. Then enter the image parameter as shown in fig 6.

To see our object detection mask which is defined as 128 by 128 arrays with name 'out_img' has parameter as shown in fig 6. Now our object detection mask will be displayed. After that to view our input images i.e. actual image and background image, which is needed to compare result, do same procedure for that which is done to display our object detection mask with respective changes.

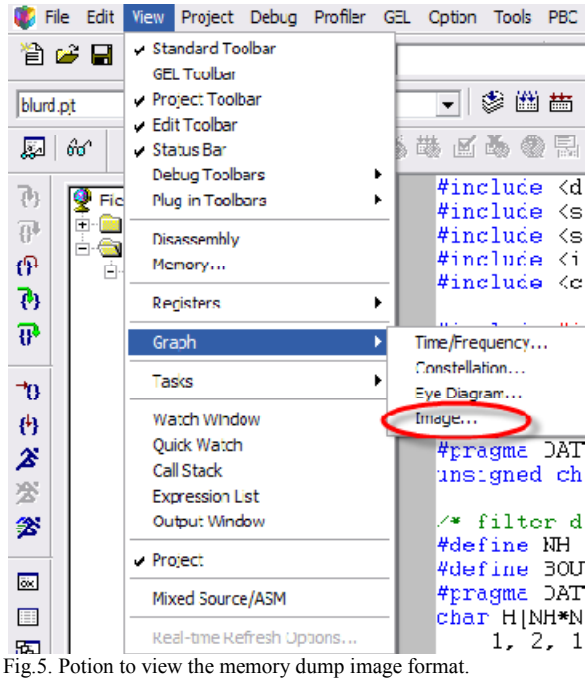

Fig.5. Potion to view the memory dump image format.

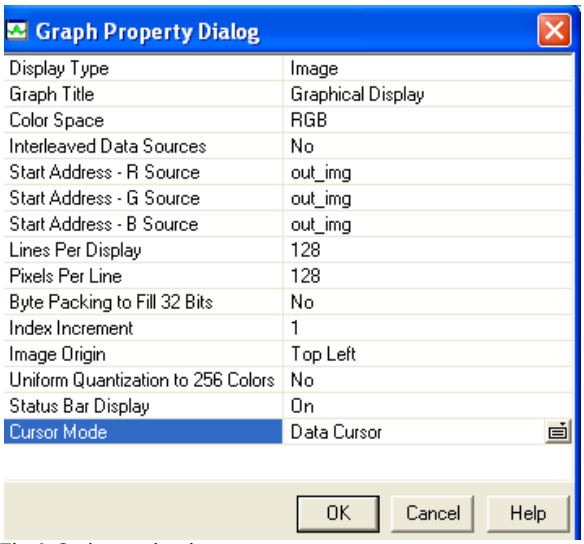

Fig.6. Option to view image.

IV. Experimental Results

The objective of this section is to demonstrate the experimental results, which we have got using our implementation. Here we have pictures, which demonstrate our result, these are explain as, fig 7, shows our actual image frame, fig 8, shows our background image frame, and fig 9, gives the binary object mask detected using DSP processor TMS3206713. We have analyzed this result; here we found the foreign objects are detected by our binary detection mask.

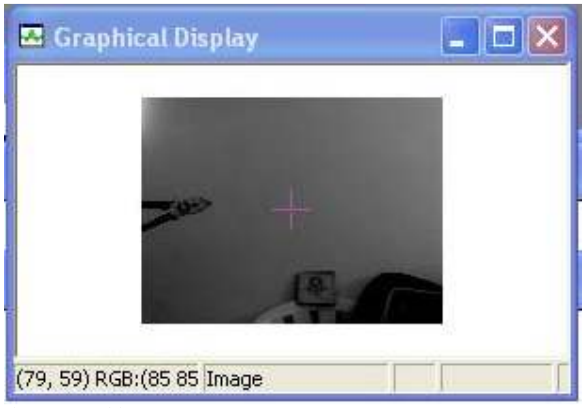

Fig.7. Actual image.

V. Conclusion

This paper has presented a method to implement image processing algorithm on TMS3206713 DSP processor in step by step manner, to explain this we have used actual image and background image and generated a binary mask for objects, which are present in actual image. This means we have implemented a motion detection using background subtraction method on TMS3206713 DSP processor and demonstrated our result, which was found satisfactory.

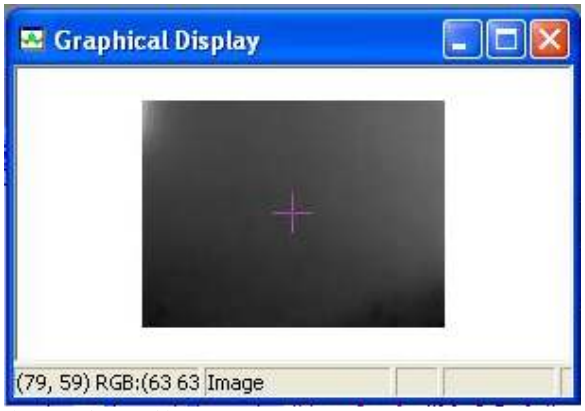

Fig.8. Background image.

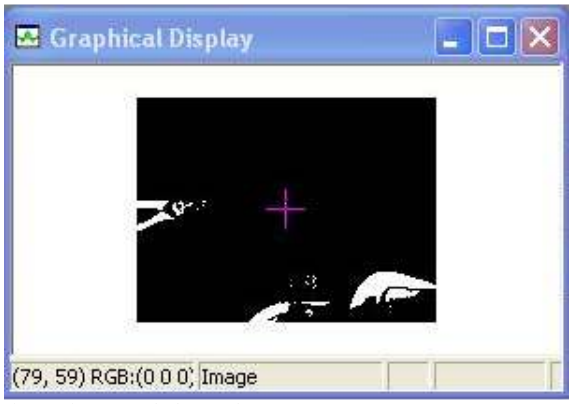

Fig.9. Binary mask detected.

References

[1] M. Oral and U. Deniz, "Center of mass model: A novel approach tobackground modeling for segmentation of moving objects," *Image Vis.Comput.*, vol. 25, pp. 1365–1376, Aug. 2007.

[2] S. Jayaraman, S. Esakkirajan, T. Veerakumar, "Digital Image Processing," Tata McGraw-Hill Education, 2011.

[3] User's Guide, "Code Composer Studio IDE Getting Started Guide", Texas Instruments, May 2005

[4] User's Guide, "TMS320C6713 DSP Starter Kit", SPECTRUM DIGITAL INC, 2003.

[5] http://www.analog.com/

[6] http://www.ti.com/