PROJECT REPORT ON

OBJECT TRACKING

USING BEAGLEBONE BLACK (rev-C)

by

Akash Sinha	**(1106831006)**
Nirmit Seth	**(1106831042)**
Ratnodai Singh	**(1106831061)**
Vaibhav Chaudhary	**(1006831087)**

Department of Electronics & Communication Engineering

Meerut Institute of Engineering & Technology

Meerut, U.P. (India)

May, 2015

PROJECT REPORT ON

OBJECT TRACKING

USING BEAGLEBONE BLACK(rev-C)

by

Akash Sinha **(1106831006)**

Nirmit Seth **(1106831042)**

Ratnodai Singh **(1106831061)**

Vaibhav Chaudhary **(1006831087)**

Submitted to the Department of Electronics & Communication Engineering

in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Electronics & Communication Engineering

Meerut Institute of Engineering & Technology, Meerut

Uttar Pradesh Technical University, Lucknow

May, 2015

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning except where due acknowledgment has been made in the text.

Signature :

Name : Akash Sinha

Roll No. : 1006831006

Date :

Signature :

Name : Nirmit Seth

Roll No. : 1006831042

Date :

Signature :

Name : Ratnodai Singh

Roll No. : 1006831061

Date :

Signature :

Name : Vaibhav Chaudhary

Roll No. : 1006831087

Date :

CERTIFICATE

This is to certify that the Project Report entitled **"Object Tracking Using BeagleBone Black(rev-C)"** which is submitted by **Akash Sinha (1106831006), Nirmit Seth (1106831042), Ratnodai Singh (1006831042), Vaibhav Chaudhary (1106831087)** in partial fulfilment of the requirement for the award of degree B.Tech in **Department of Electronics & Communication Engineering** of Uttar Pradesh Technical University, is record of the candidate own work carried out by him under my/our supervision. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.

H P Singh
Asst. Prof.
Dept. of ECE
Date:

ACKNOWLEDGMENT

It gives us a great sense of pleasure to present the report of the B.Tech Project undertaken during B.Tech Final Year. We owe special debt of gratitude of **Professor H P Singh,** from the Department of Electronics & Communication Engineering, Meerut Institute of Engineering & Technology, Meerut for their constant support & guidance throughout the course of our work. Their sincerity, thoroughness and perseverance had been a constant source of inspiration for us. It is only their cognizant efforts that our endeavours have seen light of the day.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Signature :

Name : Akash Sinha

Roll No. : 1006831006

Date :

Signature :

Name : Nirmit Seth

Roll No. : 1006831042

Date :

Signature :

Name : Ratnodai Singh

Roll No. : 1006831061

Date :

Signature :

Name : Vaibhav Chaudhary

Roll No. : 1006831087

Date :

ABSTRACT

Object Tracking is one of the key tasks in the field of computer vision. This is a useful operation in automated security solutions, traffic monitoring system, etc. Various methods of object tracking are available. But two main methods used of tracking are widely used in real time application

MeanShift algorithm is designed for static distribution. The method tracks targets by finding the most similar distribution pattern in a frame sequences with its sample pattern by iterative searching. It is simple in implementation. But it fails to track the object when it moves away from Camera.

Continuous adaptive mean-shift (CAMShift) was used to overcome this problem. CAMShift is designed for dynamically changing distributions. These occur when objects in video sequences are being tracked and the object moves so that the size and location of the probability distribution changes in time. The CAMShift algorithm adjusts the search Window size in the course of its operation. Initial window size can be set at any reasonable value.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

mV	mili volts
V	Volts
K	Kilo ohm
A	Ampere
=	equal to
ms	milli second

LIST OF ABBREVIATIONS

ARM	Advanced RISC Machines
USB	Universal Serial Bus
HDMI	High-Definition Multimedia Interface
GPIO	General Purpose Input Output
FPGA	Field Programmable Gate Array
GCC	GNU Compiler Collection
eMMC	Embedded Multimedia Card
DDR3	Double Data Rate-3
ASIC	Application Specific Integrated Circuit
SoC	System on Chip
IC	Integrated Circuit

CHAPTER 1

INTRODUCTION

The project Object Tracking Using BeagleBone is an Embedded Linux based project, the system uses an ARM development board – BeagleBone Black (rev-C) that runs Linux on it. Object tracking and image processing is done using OpenCV (Open Source Computer Vision) library for C/C++. The function of this system is to track a user selected object from a webcam live feed.

This system is very useful for industries in the field of Augmented Reality, Human-Computer Interaction and Image Processing.

The aim of this project is to design a system that can track a user selected object from the camera live feed and drive the rover accordingly. Here object is tracked using CamShift algorithm method in OpenCV library. The rover follows the object as it moves in the camera frame. Driving directions and the GPIOs value will be displayed on the output screen.

Feature selection plays an important role in object tracking. The most commonly used features are color, edges, and various color spaces other than RGB are used for tracking purposes such as HSV. Because RGB colorspace does not correspond to the colour differences perceived by humans In this project we use CAMshift algorithm for tracking of moving object but we don't use any feature information of object to be tracked. This causes, any object can be tracked even though its feature information is not known to user. In conventional Camshift, Hue saturation value is adjusted to detect the object in each consecutive video frames. But here we are generating mask image. This will mask the other motion contours in frame, only required object is visible in binary foreground image. This binary foreground image is considered as reference image.

CHAPTER 2

TECHNICAL BACKGROUND

2.1 Project Model

Following is the block diagram of the system used to track object:

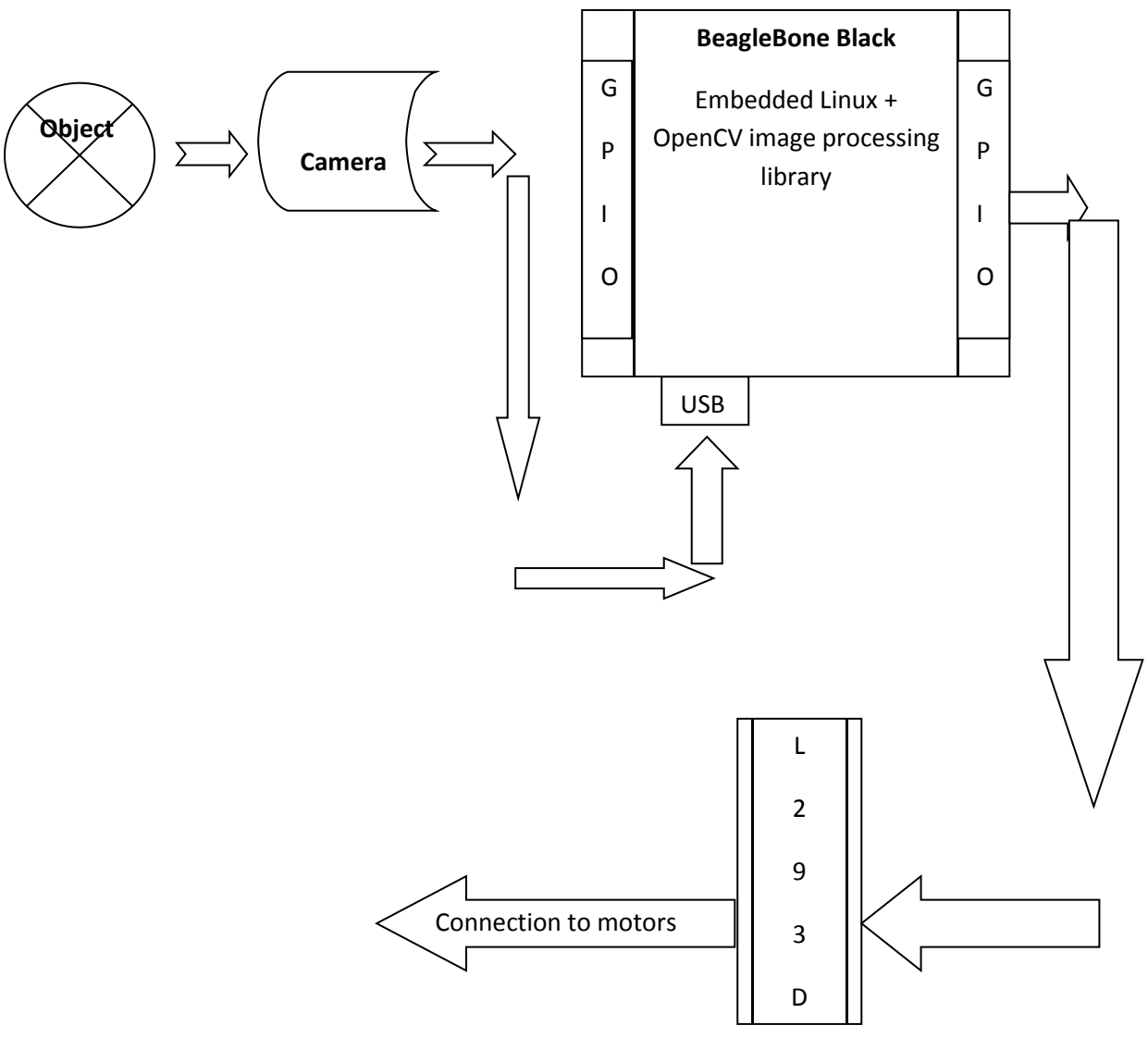


Fig 2.1: Block diagram

2.2 Components

Various types of components used in our project are as follows:

1. ARM development board : BeagleBone Black (rev-C)
2. Motor Driver IC : L293D
3. DC geared motor
4. WebCam

Now we discuss each and every component in brief.

2.2.1 ARM development board : BeagleBone Black (rev-C)

BeagleBone Black is a low-cost, community-supported development platform for developers and hobbyists. Boot Linux in under 10 seconds and get started on development in less than 5 minutes with just a single USB cable.

Fig 2.2: BeagleBone Black (rev-C)

Processor: AM335x 1GHz ARM® Cortex-A8

- 512MB DDR3 RAM
- 4GB 8-bit eMMC on-board flash storage
- 3D graphics accelerator
- NEON floating-point accelerator
- 2x PRU 32-bit microcontrollers

Connectivity

- USB client for power & communications
- USB host
- Ethernet
- HDMI
- 2x 46 pin headers

Software Compatibility

- Debian
- Android
- Ubuntu
- Cloud9 IDE on Node.js w/ BoneScript library
- plus much more

For specifications of BeagleBone Black please refer Appendix A.

2.2.2 Motor Driver IC : L293D

L293D is a dual H-bridge motor driver integrated circuit (IC). Motor drivers act as current amplifiers since they take a low-current control signal and provide a higher-current signal. This higher current signal is used to drive the motors.

L293D contains two inbuilt H-bridge driver circuits. In its common mode of operation, two DC motors can be driven simultaneously, both in forward and reverse direction. The motor operations of two motors can be controlled by input logic at pins 2 & 7 and 10 & 15. Input logic 00 or 11 will stop the corresponding motor. Logic 01 and 10 will rotate it in clockwise and anticlockwise directions, respectively.

Enable pins 1 and 9 (corresponding to the two motors) must be high for motors to start operating. When an enable input is high, the associated driver gets enabled. As a result, the outputs become active and work in phase with their inputs. Similarly, when the enable input is low, that driver is disabled, and their outputs are off and in the high-impedance state.

For specifications of L293D please refer Appendix B.

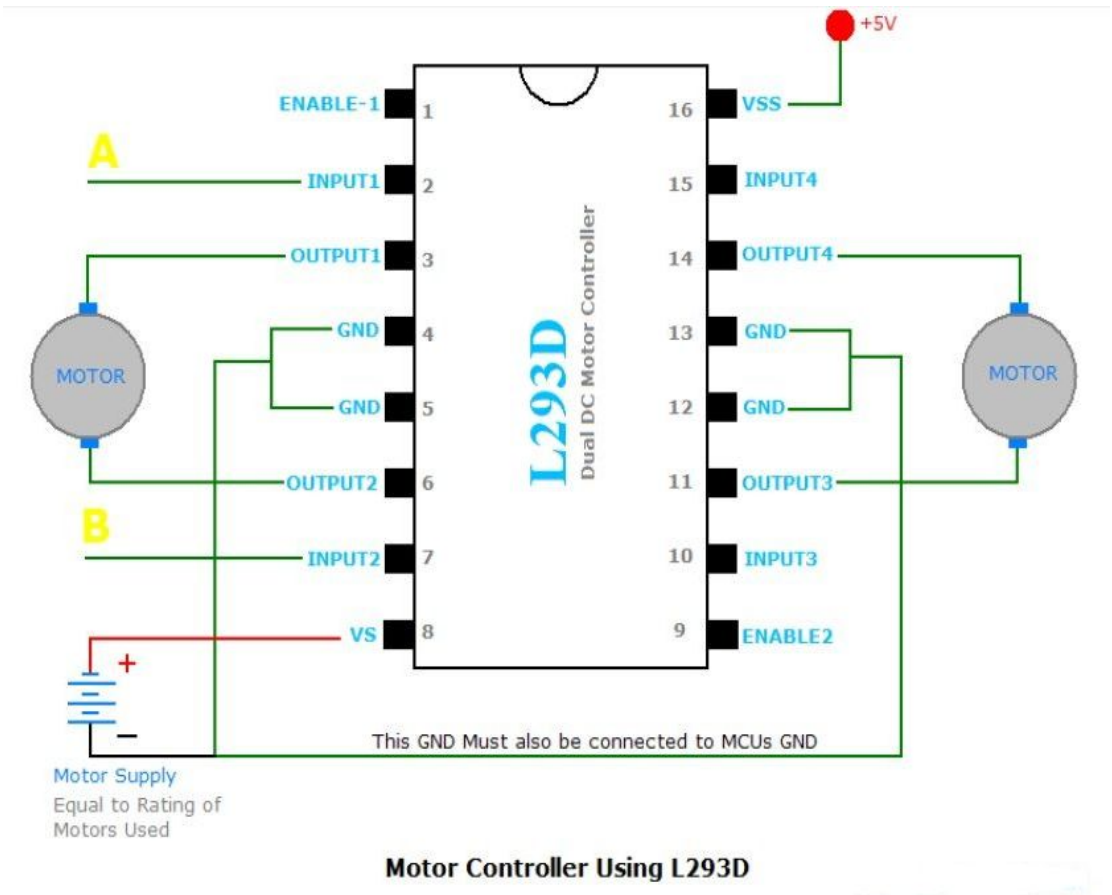
2.2.3 DC Geared Motor

A geared DC Motor has a gear assembly attached to the motor. The speed of motor is counted in terms of rotations of the shaft per minute and is termed as RPM .The gear assembly helps in increasing the torque and reducing the speed. Using the correct combination of gears in a gear motor, its speed can be reduced to any desirable figure. This concept where gears reduce the speed of the vehicle but increase its torque is known as gear reduction.

CHAPTER 3

INTERFACING

3.1 Interfacing Motor Driver IC with DC motor



Motor Controller Using L293D

Fig 3.1: Complete Interfacing Diagram

CHAPTER 4

PROJECT DEVELOPMENT

4.1 Porting Linux Kernel to BeagleBone Black

ARM development boards are the ideal platform for accelerating the development and reducing the risk of new SoC designs. The combination of ASIC and FPGA technology in ARM boards delivers an optimal solution in terms of speed, accuracy, flexibility and cost. The Embedded modules, based on ARM, can become very complex machines since these are meant to support varied tasks such as memory management, process management and peripheral interfaces. For seamless integration of these functional modules an OS has to be ported on these ARM based CPUs. Traditionally this OS porting is often the specialized work of third party vendors having expertise in this domain. For each new CPU architecture, the OS has to be customized, compiled and burnt into the core .With the coming of age of Linux as an embedded OS all this has changed quite significantly. Being in Open Source domain, Linux kernel can be freely downloaded and compiled for any system architecture and this includes ARM based systems also. This enables the developers to port the OS themselves.

Pre-built OS images are available like Angstrom, Ubuntu, Debian of various version, however to build your own OS for some reason, than you need to get the source code, patches for ARM7 and build using GCC cross compiler configuration to have it available to build whatever modules you would like.

For complete porting procedure please refer Appendix C.

4.2 Installing OpenCV on BeagleBone Black

OpenCV (open source computer vision) is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Ubuntu Linux. OpenCV was designed for computational efficiency and with a strong focus on real-time applications.

OpenCV is the most popular and advanced code library for Computer Vision related applications today, spanning from many very basic tasks (capture and pre-processing of image data) to high-level algorithms (feature extraction, motion tracking, machine learning). It is free software and provides a rich API in C, C++, Java and Python. Other wrappers are available. The library itself is platform-independent and often used for real-time image processing and computer vision.

Step 1 :

Copy the following script to gedit and save as opencv.sh :

```
1 version="$(wget -q -O - http://sourceforge.net/projects/opencvlibrary/files/opencv-unix | egrep -m1 -o '\"[0-9](\.[0-9]+)+' | cut -c2-)"
2 echo "Installing OpenCV" $version
3 mkdir OpenCV
4 cd OpenCV
5 echo "Removing any pre-installed ffmpeg and x264"
6 sudo apt-get -qq remove ffmpeg x264 libx264-dev
7 echo "Installing Dependenices"
8 sudo apt-get -qq install libopencv-dev build-essential checkinstall cmake pkg-config yasm libjpeg-dev libjasper-dev libavcodec-dev libavformat-dev libswscale-dev libdc1394-22-dev libxine-dev libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libv4l-dev python-dev python-numpy libtbb-dev libqt4-dev libgtk2.0-dev libfaac-dev libmp3lame-dev libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev libvorbis-dev libxvidcore-dev x264 v4l-utils ffmpeg cmake qt5-default checkinstall
9 echo "Downloading OpenCV" $version
```

```
wget -O OpenCV-$version.zip
http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/$version/opencv-
"$version".zip/download
echo "Installing OpenCV" $version
unzip OpenCV-$version.zip
cd opencv-$version
mkdir build
cd build
cmake -D CMAKE_BUILD_TYPE=RELEASE -D
CMAKE_INSTALL_PREFIX=/usr/local -D WITH_TBB=ON -D
BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=ON -D
INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D
BUILD_EXAMPLES=ON -D WITH_QT=ON -D WITH_OPENGL=ON ..
make -j2
sudo checkinstall
sudo sh -c 'echo "/usr/local/lib" > /etc/ld.so.conf.d/opencv.conf'
sudo ldconfig
echo "OpenCV" $version "ready to be used"
```

Step 2 :

Open terminal :

```
$ chmod +x opencv.sh
$ ./opencv.sh
```

Step 3 :

For compiling in C++ :

```
1 $ g++ -ggdb `pkg-config --cflags opencv` -o `basename opencvtest.cpp .cpp`
opencvtest.cpp `pkg-config --libs opencv`
2 $ ./opencvtest
```

4.3 System Cost

It is the total cost of the system which has been designed. It involves the summation of all the cost of all the individual components.

Table 4.1: System Cost

Item	Cost
BeagleBone Black (rev-C)	**5500**
Logitech	**799**
L293D	**115**
Robot chassis	**225**
DC motor (2)	**400**
Robot Tyres (2)	**200**
Total	**7239**

4.4 Requirements

The project has two requirements- software and hardware:

4.4.1 Software Requirements

1. Linux on host system
2. Linaro cross-compiler tool chain
3. U-BOOT bootloader
4. Linux Kernel
5. OpenCV image processing library

4.4.2 **Hardware requirements**:

1. Host System
2. ARM development board : BeagleBone Black (rev-C)
3. WebCam
4. DC motor
5. L293D board
6. Mini USB cable
7. Connectors

CHAPTER 5
ALGORITHMS

Meanshift and Camshift

Goal

In this chapter,

- We will learn about Meanshift and Camshift algorithms to find and track objects in videos.

5.1 Meanshift

The intuition behind the meanshift is simple. Consider you have a set of points. (It can be a pixel distribution like histogram backprojection). You are given a small window (may be a circle) and you have to move that window to the area of maximum pixel density (or maximum number of points). It is illustrated in the simple image given below:

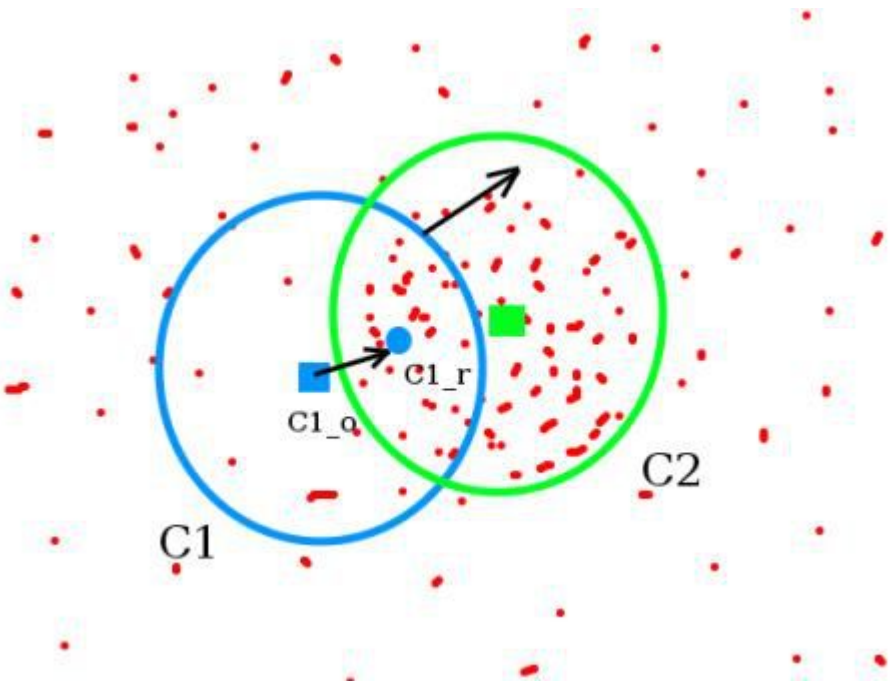


Fig 5.1: MeanShift mechanism

The initial window is shown in blue circle with the name “C1”. Its original center is marked in blue rectangle, named “C1_o”. But if you find the centroid of the points inside that window, you will get the point “C1_r” (marked in small blue circle) which is the real centroid of window. Surely they don’t match. So move your window such that circle of the new window matches with previous centroid. Again find the new centroid. Most probably, it won’t match. So move it again, and continue the iterations such that center of window and its centroid falls on the same location (or with a small desired error). So finally what you obtain is a window with maximum pixel distribution. It is marked with green circle, named “C2”. As you can see in image, it has maximum number of points.

So we normally pass the histogram backprojected image and initial target location. When the object moves, obviously the movement is reflected in histogram backprojected image. As a result, meanshift algorithm moves our window to the new location with maximum density.

Meanshift in OpenCV

To use meanshift in OpenCV, first we need to setup the target, find its histogram so that we can backproject the target on each frame for calculation of meanshift. We also need to provide initial location of window. For histogram, only Hue is considered here. Also, to avoid false values due to low light, low light values are discarded using **cv2.inRange()** function.

5.2 CAMshift

In meanshift window always has the same size when the object is far away or it is very close to camera. This is not good. We need to adapt the window size with size and rotation of the target. Once again, the solution came from “OpenCV Labs” and it is called CAMshift (Continuously Adaptive Meanshift) published by Gary Bradsky in his paper “Computer Vision Face Tracking for Use in a Perceptual User Interface” in 1988.

It applies meanshift first. Once meanshift converges, it updates the size of the window as, $s = 2 \times \sqrt{\frac{M_{00}}{256}}$. It also calculates the orientation of best fitting ellipse to it.

Again it applies the meanshift with new scaled search window and previous window location. The process is continued until required accuracy is met.

Camshift in OpenCV

It is almost same as meanshift, but it returns a rotated rectangle (that is our result) and box parameters (used to be passed as search window in next iteration).

CHAPTER 6
PROGRAM CODING

```
#include "opencv2/video/tracking.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"

#include <iostream>
#include <ctype.h>
#include <stdio.h>

using namespace cv;
using namespace std;

Mat image;

int flag=1;
bool selectObject = false;
int trackObject = 0;
Point origin,center;
Rect selection;
int vmin = 10, vmax = 256, smin = 30;
void drive(Point);

static void init()
{
        system("sudo echo 48 > /sys/class/gpio/export");
        system("sudo echo 49 > /sys/class/gpio/export");
        system("sudo echo 51 > /sys/class/gpio/export");
        system("sudo echo 60 > /sys/class/gpio/export");
        system("sudo echo out > /sys/class/gpio/gpio48/direction");
```

```
        system("sudo echo out > /sys/class/gpio/gpio49/direction");
        system("sudo echo out > /sys/class/gpio/gpio51/direction");
        system("sudo echo out > /sys/class/gpio/gpio60/direction");
}

static void intro()
{
system("clear");
printf("-----------------------------\n");
printf("Final Year Project (2014-15)\n");
printf("Object Tracking With BeagleBoneBlack\n");
printf("#linux  #OpenCV       #C,C++          #OpenSource\n");
printf("-----------------------------\n");
printf("Team Members : \n");
printf("-----------------------------\n");
printf("Akash Sinha       (1106831006)\n");
printf("Nirmit Seth       (1106831042)\n");
printf("Ratnodai Singh    (1106831061)\n");
printf("Vaibhav Chaudhary (1106831087)\n");
printf("-----------------------------\n");
}

static void onMouse( int event, int x, int y, int, void* )
{
   if( selectObject )
   {
      selection.x = MIN(x, origin.x);
      selection.y = MIN(y, origin.y);
      selection.width = std::abs(x - origin.x);
      selection.height = std::abs(y - origin.y);

      selection &= Rect(0, 0, image.cols, image.rows);
   }
```

```
    switch( event )
    {
        case CV_EVENT_LBUTTONDOWN:
            origin = Point(x,y);
            selection = Rect(x,y,0,0);
            selectObject = true;
            break;
        case CV_EVENT_LBUTTONUP:
            selectObject = false;
            if( selection.width > 0 && selection.height > 0 )
                trackObject = -1;
            break;
    }
}

static void help()
{
    cout << "\nObject Tracking \n"
    "You select a colored object and the algorithm tracks it.\n"
    "This takes the input from the webcam\n"
    "Usage: \n"
    "$ ./main [camera number]\n";

    cout << "\n\nKeyboard input options: \n"
    "\tESC - quit the program\n"
    "\ts - stop the tracking\n"
    "\tp - pause video\n"
    "\nTo start tracking an object, select the rectangular region around it with the
mouse\n\n";
}

int main( int argc, const char** argv )
```

```
{
   intro();
   help();
   init();
   VideoCapture cap;
   Rect trackWindow;
   int hsize = 16;
   float hranges[] = {0,180};
   const float* phranges = hranges;

   int camNum = 0;
   if(argc == 2)
      camNum = atoi(argv[1]);

   cap.open(camNum);
         printf("%f\t%f\n",cap.get(3),cap.get(4));

   if( !cap.isOpened() )
   {
      help();
      cout << "***Could not initialize capturing...***\n";
      cout << "Current parameter's value: " << camNum << endl;
      return -1;
   }

   namedWindow( "CamShift Object Tracker", 0 );
   setMouseCallback( "CamShift Object Tracker", onMouse, 0 );

   Mat frame, hsv, hue, mask, hist, histimg = Mat::zeros(200, 320, CV_8UC3),
backproj;
   bool paused = false;

   for(;;)
```

```
{
    if( !paused )
    {
        cap >> frame;
        if( frame.empty() )
            break;
    }

    frame.copyTo(image);

    if( !paused )
    {
        cvtColor(image, hsv, CV_BGR2HSV);

        if( trackObject )
        {
            int _vmin = vmin, _vmax = vmax;

            inRange(hsv, Scalar(0, smin, MIN(_vmin,_vmax)),
                    Scalar(180, 256, MAX(_vmin, _vmax)), mask);
            int ch[] = {0, 0};
            hue.create(hsv.size(), hsv.depth());
            mixChannels(&hsv, 1, &hue, 1, ch, 1);

            if( trackObject < 0 )
            {
                Mat roi(hue, selection), maskroi(mask, selection);
                calcHist(&roi, 1, 0, maskroi, hist, 1, &hsize, &phranges);
                normalize(hist, hist, 0, 255, CV_MINMAX);

                trackWindow = selection;
                trackObject = 1;
```

```
            }

            calcBackProject(&hue, 1, 0, hist, backproj, &phranges);
            backproj &= mask;
            RotatedRect trackBox = CamShift(backproj, trackWindow,
                                    TermCriteria( CV_TERMCRIT_EPS |
CV_TERMCRIT_ITER, 10, 1 ));

                    //printf("%f\t%f\n",trackBox.size.width,trackBox.size.height);
            center=trackBox.center;
            ellipse( image, trackBox, Scalar(0,0,255), 3, CV_AA );
                    drive(center);
        }
    }
    else if( trackObject < 0 )
        paused = false;

    if( selectObject && selection.width > 0 && selection.height > 0 )
    {
        Mat roi(image, selection);
        bitwise_not(roi, roi);
    }

    imshow( "CamShift Object Tracker", image );
    char c = (char)waitKey(10);
        //printf("%d\t%d\n",center.x,center.y);

        if( c == 27 )
        break;
    switch(c)
    {
        case 's':
```

```
            trackObject = 0;
            histimg = Scalar::all(0);
            break;

        case 'p':
            paused = !paused;
            break;

        default:
            ;
      }
   }

   return 0;
}

static void gpio()
{
        system("echo -n gpio48 : && cat /sys/class/gpio/gpio48/value");
        system("echo -n gpio49 : && cat /sys/class/gpio/gpio49/value");
        system("echo -n gpio51 : && cat /sys/class/gpio/gpio51/value");
        system("echo -n gpio60 : && cat /sys/class/gpio/gpio60/value");
}

void drive(Point ctrl)
{

        if(ctrl.x<150)
        {
                if(flag!=1)
                {
                system("clear");
                intro();
```

```
            printf("turn left\n");
            system("echo 1 > /sys/class/gpio/gpio48/value");
            system("echo 1 > /sys/class/gpio/gpio49/value");
            system("echo 0 > /sys/class/gpio/gpio51/value");
            system("echo 0 > /sys/class/gpio/gpio60/value");
            flag=1;
            gpio();
            }
    }
    else if(ctrl.x>490)
    {
            if(flag!=2)
            {
            system("clear");
            intro();
            printf("turn right\n");
            system("echo 0 > /sys/class/gpio/gpio48/value");
            system("echo 0 > /sys/class/gpio/gpio49/value");
            system("echo 1 > /sys/class/gpio/gpio51/value");
            system("echo 1 > /sys/class/gpio/gpio60/value");
            flag=2;
            gpio();
            }
    }

    else if(ctrl.y<125)
    {
            if(flag!=3)
            {
            system("clear");
            intro();
            printf("drive back\n");
            system("echo 0 > /sys/class/gpio/gpio48/value");
```

```
            system("echo 0 > /sys/class/gpio/gpio49/value");
            system("echo 1 > /sys/class/gpio/gpio51/value");
            system("echo 1 > /sys/class/gpio/gpio60/value");
            flag=3;
            gpio();
            }
    }

    else if(ctrl.y>355)
    {
            if(flag!=4)
            {
            system("clear");
            intro();
            printf("drive froward\n");
            system("echo 0 > /sys/class/gpio/gpio48/value");
            system("echo 0 > /sys/class/gpio/gpio49/value");
            system("echo 1 > /sys/class/gpio/gpio51/value");
            system("echo 1 > /sys/class/gpio/gpio60/value");
            flag=4;
            gpio();
            }
    }
    else
    {
            if(flag!=5)
            {
            system("clear");
            intro();
            printf("ruk ja thand rakh\n");
            system("echo 1 > /sys/class/gpio/gpio48/value");
            system("echo 0 > /sys/class/gpio/gpio49/value");
            system("echo 1 > /sys/class/gpio/gpio51/value");
```

```
                system("echo 0 > /sys/class/gpio/gpio60/value");
                flag=5;
                gpio();
                }
        }
}
```

CHAPTER 7
CONCLUSION, LIMITATIONS & SCOPE OF IMPROVEMENT

6.1 Conclusion

We proposed the efficient approach of CAMshift algorithm to track an object in video frame. In this approach, feature information of object to be tracked is not required. Hence one can track any type of object in video even though its feature information is not known to user. We finally evaluated our algorithm`s performance in practice, and showed how our approach can stand tracking even though feature information of object is not known to user. We have successfully implemented Use of Mask image to find initial location of object and continuous updated binary foreground image in CAMshift algorithm. This approach increases the efficiency of CAMshift algorithm to track object, even though its feature information is not known

By the proposed system object in camera live feed can be tracked and followed by the rover. Object's approximate centroid position relative to the video frame is also obtained.

In this way this system provides a great deal of usage in the field of image processing for tracking objects.

6.2 Limitations

The system proposed is confined to low resolution images as the BeagleBone is not much powerful at image processing. Erroneous tracking is observed in low and intense light conditions. Thus, an optimum bright environment is required for best results.

6.3 Further Improvements

In the future work, our algorithm will modify to track all motion contours in each frame by preparing link list of motion contours and tracked object. By doing other

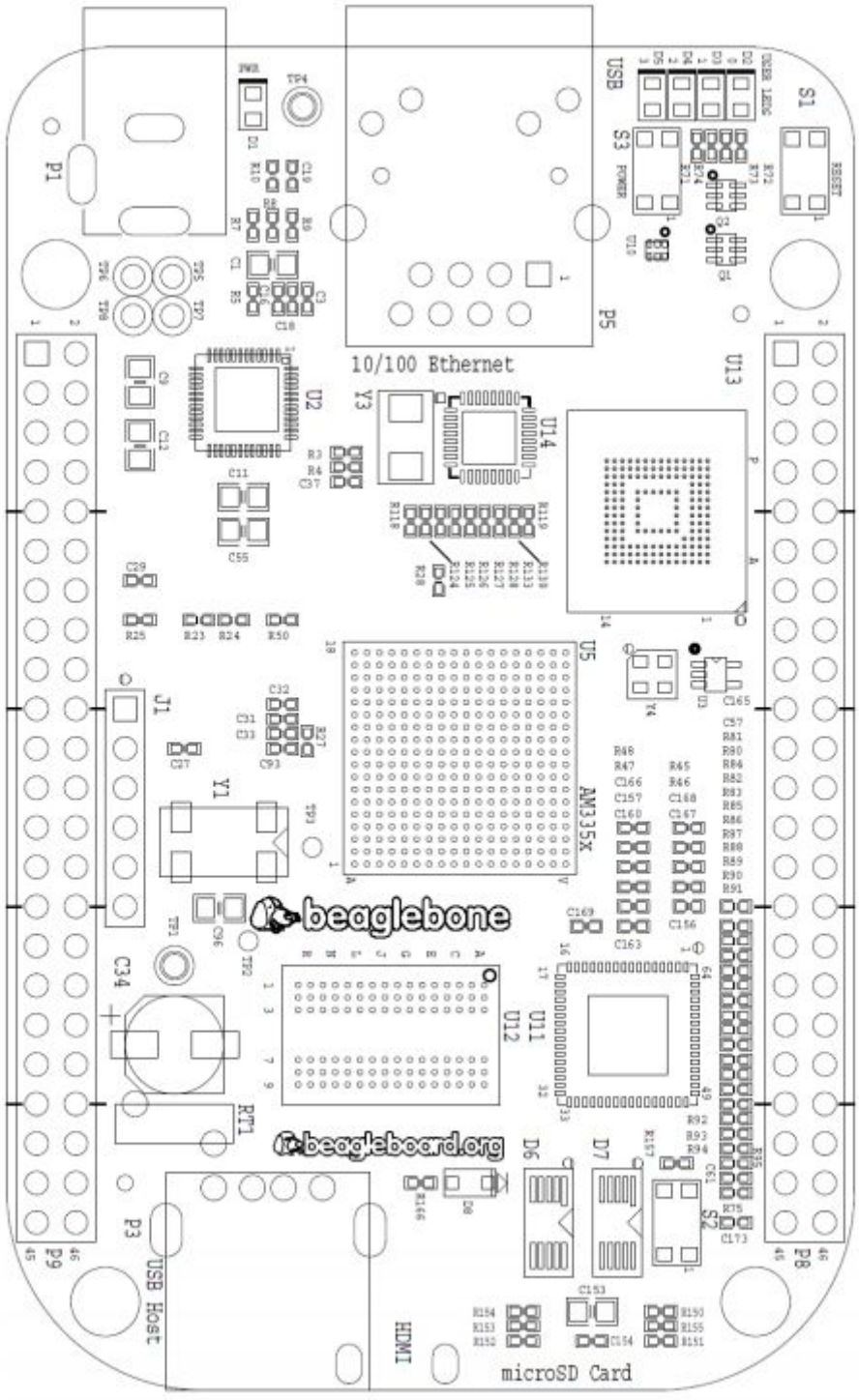
few changes, our algorithm is capable to track motion contours which are travel from one frame to another frame. And also it should identify newly introduce motion contours and also able to track them in consecutive video frames.

APPENDIX A: BeagleBone Black

Specifications:

	Feature	
Processor	Sitara AM3358BZCZ100 1GHz, 2000 MIPS	
Graphics Engine	SGX530 3D, 20M Polygons/S	
SDRAM Memory	512MB DDR3L 800MHZ	
Onboard Flash	4GB, 8bit Embedded MMC	
PMIC	TPS65217C PMIC regulator and one additional LDO.	
Debug Support	Optional Onboard 20-pin CTI JTAG, Serial Header	
Power Source	miniUSB USB or DC Jack	5VDC External Via Expansion Header
PCB	3.4" x 2.1"	6 layers
Indicators	1-Power, 2-Ethernet, 4-User Controllable LEDs	
HS USB 2.0 Client Port	Access to USB0, Client mode via miniUSB	
HS USB 2.0 Host Port	Access to USB1, Type A Socket, 500mA LS/FS/HS	
Serial Port	UART0 access via 6 pin 3.3V TTL Header. Header is populated	
Ethernet	10/100, RJ45	
SD/MMC Connector	microSD , 3.3V	
User Input	Reset Button Boot Button Power Button	
Video Out	16b HDMI, 1280x1024 (MAX) 1024x768,1280x720,1440x900 ,1920x1080@24Hz w/EDID Support	
Audio	Via HDMI Interface, Stereo	
Expansion Connectors	Power 5V, 3.3V , VDD_ADC(1.8V) 3.3V I/O on all signals McASP0, SPI1, I2C, GPIO(69 max), LCD, GPMC, MMC1, MMC2, 7 AIN(1.8V MAX), 4 Timers, 4 Serial Ports, CAN0, EHRPWM(0,2),XDMA Interrupt, Power button, Expansion Board ID (Up to 4 can be stacked)	
Weight	1.4 oz (39.68 grams)	
Power	Refer to Section 6.1.7	

Table A.1: BeagleBone Black(rev-C) Specifications



Component Side Silkscreen

Fig A.1: BeagleBone Black(rev-C)

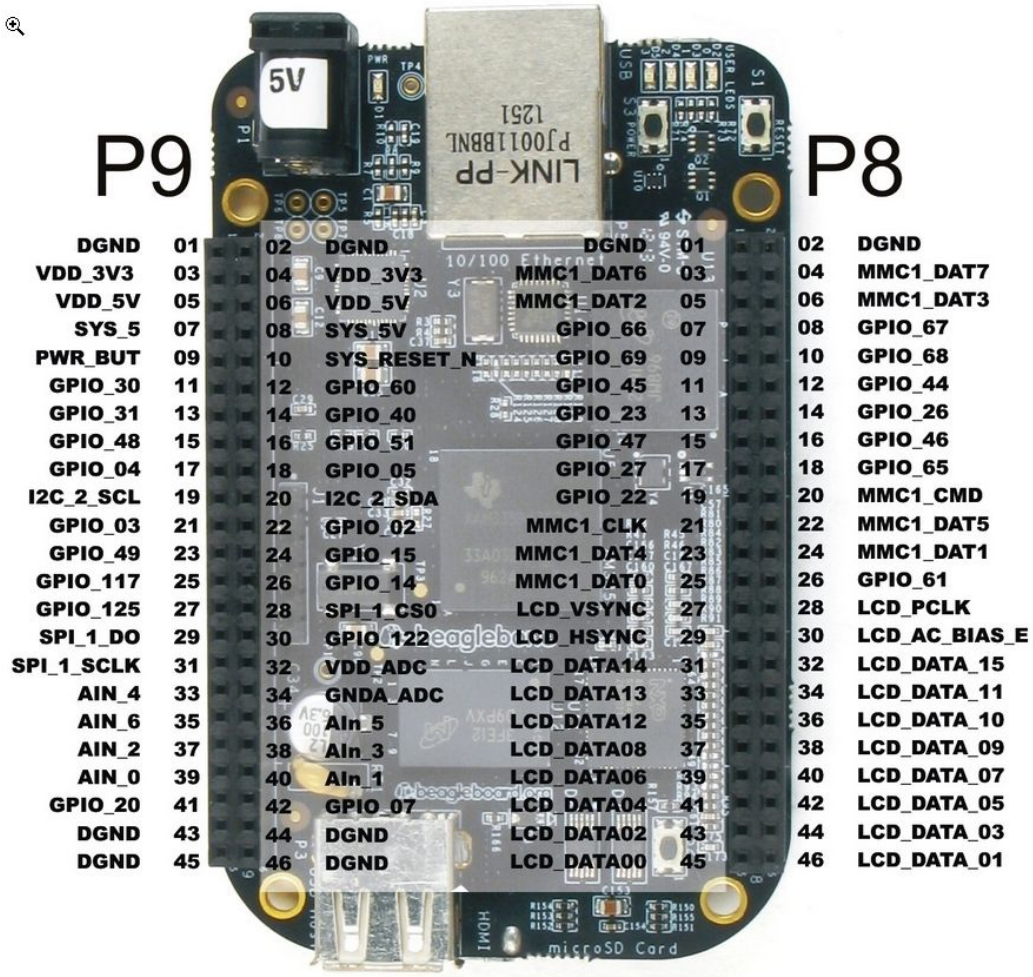


Fig A.2: GPIO pin configuration

APPENDIX B: L293D (motor driver IC)

Pin diagram:

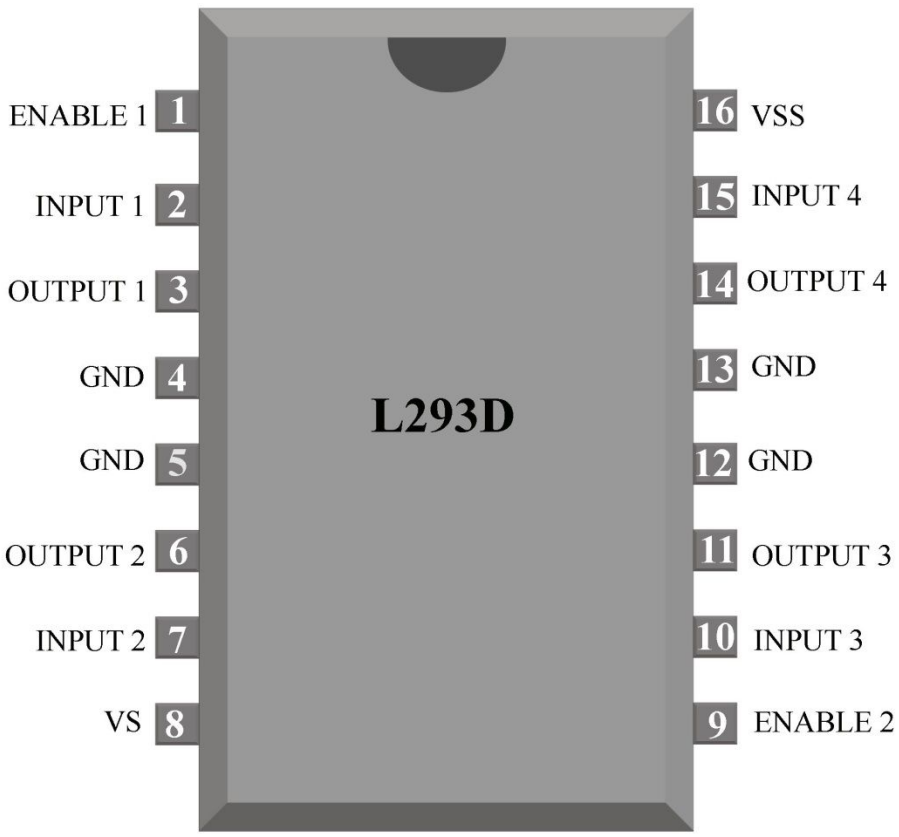


Fig B.1: L293D pin diagram

APPENDIX C: PORTING LINUX TO BEAGLEBONE BLACK

Step 0: Get build environment ready.

Step 0.0: install Debian/Ubuntu on a machine or VM

Environment and update/patch it up I prefer

64-bit Linux

Step 0.1:"sudo apt-get update"

Step 1: Get communication to Beagle Bone Black into a "known working state"

Step 1.0: beagle bone black has prebuilt angstrom distribution

Step 1.1: otherwise download latest image for angstrom distribution and write using disk-32 image writer.https://launchpad.net/win32-image-writer/ + download

Step 1.2: now open terminal and type

cd /dev

ls

Step 1.3: now connect beagle bone black by using usb cable.

Step 1.4: again type ls in terminal and carefully check which parameter is newly added.

In my board it is ttyACM0.

Step 1.5: sudo minicom –s

Go to serial port setup and change the first line as /dev/ttyACM0 save as default and exit.

Step 1.6: press enter if beagle bone log in is not displayed.

Step 1.7: log in as root. Your beagle bone black is working.

Step 2: Setup cross-compilation build environment Adapted from:

http://eewiki.net/display/linuxonarm/BeagleBone+Black#BeagleBoneBlack-BasicRequirements

Step 2.0: install 32-bit versions of key components sudo apt-get update && sudo apt-get upgrade && sudo apt-get install libc6:i386 listed++6:i386 libncurses 5:i386 zlib1g:i386

Step 2.1: download / extract cross-compiler

wget-chttps://launchpad.net/linaro-toolchain-binaries/trunk/2013.07/+download/gcc-linaro-arm-linuxgnueabihf-4.8-2013.07-1_linux.tar.xz

tar xJf gcc-linaro-arm-linux-gnueabihf-4.8-2013.07-1_linux.tar.x

export CC=`pwd`/gcc-linaro-arm-linux-gnueabihf-4.8-2013.07-1_linux/bin/arm-linux-gnueabihf

Step 2.2: Test set up

${CC} gcc –version

arm-linux-gnueabihf-gcc (crosstool-NG linaro-1.13.1-4.8-2013.07-1 - Linaro GCC 2013.07) 4.8.2 20130624(prerelease)

Copyright (C) 2013 Free Software Foundation, Inc.

This is free software; see the source for copying conditions. There is NO

Warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Step 3: Build U-Boot.

Step 3.0: Download U-Boot

git clone git://git.denx.de/u-boot.git

cd u-boot/

git checkout v2013.07 -b tmp

Step 3.1: Configure and Build U-Boot

wget https://raw.github.com/eewiki/u-boot-patches/master/v2013.07/0001-am335x_evm-uEnv.txt-bootzn-fixes.patch

patch -p1 < 0001-am335x_evm-uEnv.txt-bootz-n-fixes.patch

make ARCH=arm CROSS_COMPILE=${CC} distclean

make ARCH=arm CROSS_COMPILE=${CC} am335x_evm_config

make ARCH=arm CROSS_COMPILE=${CC}

Step 3.2: during distclean if it generates any file than clean its Again.

Step 4: Upgrade device tree compiler.

Step 4.0: Download and upgrade dtc

cd...
wget –c https://raw.github.com/RobertCNelson/tools/master/pkgs/dtc.sh
chmod +x dtc.sh
./dtc.sh

Step 5: Build Kernel and modules.
Step 5.0: Download kernel
git clone git://github.com/RobertCNelson/linux-dev.git
cd linux-dev/
git checkout origin/am33x-v3.8 -b tmp
Step 5.1: Build kernel
./build_kernel.sh
cd...

Step 6: Get rootfs setup on flash card.
Step 6.0: Download, verify and extract rootfs
Debian:
wget -c https://rcn-ee.net/deb/minfs/wheezy /debian-7 .1-minimal-armhf-2013-08-25.tar.xz
md5sum debian-7.1-minimal-armhf-2013-08-25.tar.xz
tar xJf debian-7.1-minimal-armhf-2013-08-25.tar.xz
Ubuntu:
wget -c https://rcn-ee.net/deb/minfs/raring/ubuntu-13.04-minimal-armhf-2013-08-25.tar.xz
md5sum ubuntu-13.04-minimal-armhf-2013-08-25.tar.xz
tar xJf ubuntu-13.04-minimal-armhf-2013-08-25.tar.xz
Step 6.1: figure out what device your MMC appears as in Build machine
ls /dev
Insert an MMC or a USB<->MMC adapter
ls /dev
Look for changes which have appeared. In my case, with a USB flash adapter, MMC cards appears as /dev/sdb
Step 6.2: setup a temp export for ease of use, wipe disk and setup partition table

```
export DISK=/dev/sdb
sudo dd if=/dev/zero of=${DISK} BS=1M Count=16
sudo fdisk /dev/sdb
P (print)
D (delete any existing partitions)
W (writes and exit)
sudo partprobe
sudo fdisk /dev/sdb
P (print partition table)
N (new partition)
P (select primary)
1 (partition number)
<Enter> (accept start at 2048)
+100M
T (change partition type)
1 (partition 1)
B (change to FAT32)
P (print partition table)
A (toggle boot flag)
1 (partition 1)
P (print partition table)
N (new partition)
2 (partition #2)
<Enter> (accept default start at XXXXX)
<Enter> (accept default, last sector onthe flash card)
P (print)
W (writes and exit)
sudo partprobe if it fails than you can simply plug in your usb again.
Step 6.3: format partitions
sudo mkfs.vfat -F 32 ${DISK} 1 -n boot
sudo mkfs.ext4 ${DISK} 2 -L rootfs
Step 6.4: mount partitions
sudo mount ${DISK} 1 /media/boot/
```

sudo mount ${DISK} 2 /media/rootfs/

step 6.5: install U-Boot on flash card

sudo cp -v ./u-boot/MLO /media/boot/

sudo cp -v ./u-boot/u-boot.img /media/boot/

step 6.6: save uEnv.txt into the build directory VI uEnv.txt (for edit new file)

I (for insert mode) and copy paste following by pressing ctrl+shift+v

Step 6.7: copy the uEnv.txt file into /media/boot

sudo cp -v ./uEnv.txt /media/boot/

Step 6.8: install rootfs

Look in linux-dev/deploy

Ls... Check version ie: 3.8.13-bone35.2

Export environment variable for kernel version

export kernel_version=3.8.13-bone35.2

Copy rootfs

sudo tar xfvp ./ubuntu-13.04-minimal-armhf-2013-08-25/armhf-rootfs-ubuntu-raring.tar -C /media/rootfs

Step 7.0: copy kernel files, device tree files and modules

sudo cp -v ./linux-dev/deploy/${kernel_version} .zImage /media/boot/zImage

sudo mkdir -p /media/boot/dtbs/

sudo tar xfov ./linux-dev/deploy/${kernel_version} -dtbs.tar.gz -C /media/boot/dtbs/

sudo tar xfv ./linux-dev/deploy/${kernel_version}-firmware.tar.gz -C /media/rootfs/lib/firmware

sudo tar xfv ./linux-dev/deploy/${kernel_version}-modules.tar.gz -C /media/rootfs/

Step 8: Edit configuration files on flash card.

Step 8.0: edit /etc/fstab in the MMC

sudo nano /media/rootfs/etc/fstab

Step 8.1: edit /etc/network/interfaces

sudo nano /media/rootfs/etc/network/interfaces

Step 8.2: edit /etc/udev/rules.d/70-persistent-net.rules

sudo nano /media/rootfs/ /etc/udev/rules.d/70-persistent-net.rules

Step 9: Configure a Serial console for debugging access.

Step 9.0: debian: edit /etc/inittab in the MMC

sudo nano /media/rootfs/etc/inittab

Add:

T0:23: respawn: /sbin/getty -L ttyO0 115200 vt102

Ubuntu: create serial.conf

sudo nano /media/rootfs/etc/init/serial.conf

Add:

Start on stopped RC RUNLEVEL= [2345]

Stop on runlevel [! 2345]

Respawn

Exec /sbin/Getty 115200 ttyO0

Step 10: Sync MMC and remove sudo sync.

Then plug in SD card into the beagle bone black and connect using usb cable. This time you cannot get serial console using ttyACM0, you required FTDI cable, or RS-232 to TTL 3.3v converter (refer my another paper how to get a serial console in beagle bone black). You can login as user: ubuntu and password: temppwd.

REFERENCES

[1] Open CV online documentation at "www.opencv.org"

[2] "Learning OpenCV" (E-book), By Adrian Kaehler and Gary Rost Bradski.

[3] Robert Love, "Linux Kernel Development*"*.

[4] Some Websites:

- www.alldatasheets.com
- www.datasheetcatalog.com
- www.electronicscircuits.com
- www.beaglebone.org
- www.free-electrons.com

[5] All Datasheets, "*Datasheet Design*", http:/www.alldatasheets.com

[6] Sawhney A. K., "*Electrical and Electronic Measurements and Instrumentation*" Dhanpat Rai & Co.(P) Ltd., 17th Edition.

[7] Robert L. Boylstead, Louis Nashelsky, "*Electronic Devices and Circuit Theory*", 9th Edition.

[8] Electronics for You, "*Types of Sensors & Specifications*", http:/www.electronics4u.com

[9] www.technowave.co.in

[10] www.scielectronics.com

[11] www.parallax.com