Stereo Vision using CMOS cameras in real time on low cost embedded boards using assembly code written in sub cores known as PRUs (programmable real time unit)

❏ Kartik Nighania

Processor: AM335x 1GHz ARM® Cortex-A8 in BeagleBone Black

❖ Project Name - **Beagle_Eyes**

❖ Stereo Vision-

- It is the extraction of 3D information from digital images.
- By comparing information about a scene from two cameras, 3D information can be extracted by examination of the relative positions of objects in the two images. This is similar to the biological process stereopsis that our eyes do to grasp the world around in 3D and hence differentiate between near and far objects that too in real time and make decisions.

❖ Use of stereo vision in robotics and its shortcomings in real time behaviour-

The amount of interest in the field of implementation of robotic systems for tasks like-

- ➢ Indoor automation
- ➢ Driverless transportation and
- ➢ Unknown environment exploration

have increased exponentially among the community of researchers and hobbyist.

This project aims at -

Autonomous Obstacle Avoidance and Maneuvering on a Stereo Vision-Guided MAV using On-Board Real Time Processing

Stereo vision has an edge over other techniques due to its ability to provide three dimensional information about how the environment looks like and decide how obstacles can be avoided for safe navigation through the environment.

The shortcomings are -

- ❏ The currently available stereo cameras are very much **expensive and requires special drivers and software** (that are not open source) to interface with processing platforms making it unable to use for hobbyist.

- **Real time processing is difficult due to multiple image processing algorithms,** implemented on the images obtained simultaneously from the two cameras.
- **The core processor is so used up as it is unable to perform other tasks**, even then in most cases real time determinism is not obtained.

Using BeagleBone and PRU subsystem-

BeagleBone Black gets the edge over other boards by its amazing PRU's-

- The two PRU's consists of dual 32-bit RISC cores working at 200 MHz each.
- Giving **no pipelining or throughput and great determinism by executing single cycle instructions at 5 nanoseconds. Making them perfect for real time applications.**
- **These PRU's can thus give stereo depth images in real time.**
- The 1 GHz main **chip am335x** running linux can then **use these depth images for decision making** to move MAV or other terrestrial robots.

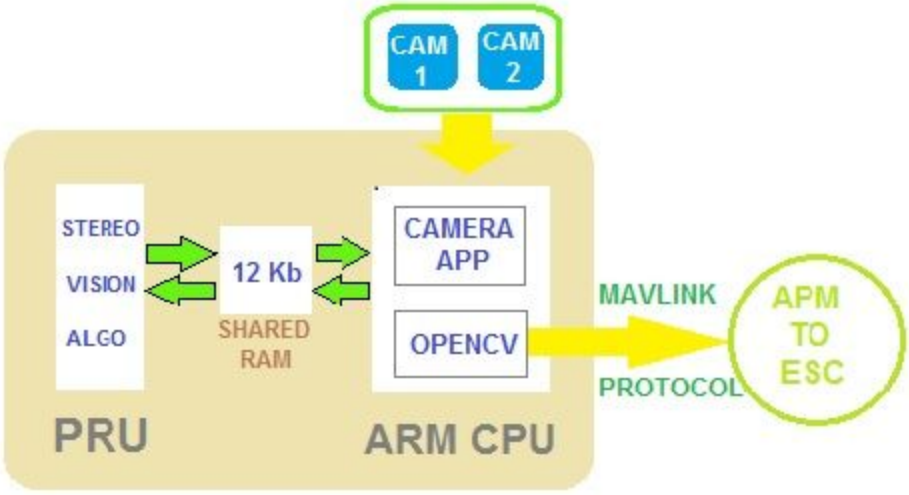
Thus above mentioned shortcomings are solved by -

- **Using PRU's for meeting real time constraints**.
- **Offloading stereo vision task to PRU's and giving main processor to breath and perform other tasks.**
- **Using cheap cameras and making customisable code, that can interface with most of the cameras selected as per users choice**.

❖ How the implementation will benefit people-

Unlike we humans it is not easy for a robot to perceive the world around. Obstacle avoidance using distance, light, proximity sensors have their own limitations and constraints. Giving the robot stereo vision opens the path to a wide range of tasks that robots can now perform and provides a platform for **autonomous behavior**. The **real time capability gives it the edge** which cannot be implemented in most of the other boards in market. Students, engineers and **hobbyist all around the world will now be able to implement and afford real-time stereo vision to their bots**. For the once with aerial vehicles having these open source hardware and software, **indoor automation for beginners will no longer be a difficult and challenging task** anymore.

❖ PROJECT IMPLEMENTATION-



❖ **The project is divided into 5 sections**-

1) The **camera app**
2) The stereo vision **PRU code**
3) **Opencv** triangulation & contour detection and output using ***roscopter*** to MAV

❖ **HARDWARE**-

4) Taking two cheap USB cameras and using a hub, input to BBB. The alignment of cameras has to be made perfect which will be done by **3d printed frames** (can also be done by accurately making wooden cut frames).
5) To provide **designs of 3d printable/ laser cut files** as well of stereo vision camera frames. (Sample already made-link below)

❑ **The interface of camera is done by the USB hub and not by PRU on BeagleBone due to following reasons-**

1) The speed of **USB 2.0 is 60 MegaBytes/sec very sufficient to get the video in real-time**. No need of extra PRU interfacing.
2) Most of the webcams have USB output. **Cutting them for PRU interface will make them unusable** for other projects which is not affordable for many.
3) By having USB support other **apps will find it easy to use camera for other tasks in linux**.

4) Every camera is different in terms of its clock frequency, data lines, enable signals etc. **Changing the camera will make the PRU code completely unusable** as code is designed strictly according to timing diagram of one particular camera.
As soon as the camera app is started it **initiates the PRU code and starts capturing images from the two cameras**. The stereo camera will provide simultaneously taken left and right image pairs as an input to the processing unit.

V4l2 - **is a powerful image capture app for linux and is closely integrated with kernel and is known to be very stable.**

Reasons to choose this-

- **Highly customisable** input format according to user needs which gives varied resolution image formats and other changeable attributes of a picture like contrast, brightness, RGB max-min values etc.
- **Supports many USB devices**, providing user to choose the **camera of his choice**. Because not everyone is able to afford expensive cameras.
- Great support by community where **customisable example apps already created**.
  Example- **graber.c (will be modified and used in this project)**
  and capture.c for image and video acquisition.
- Test showed from logitech C920 that **1920x1060 size image in H264 format at 30fps was transferred at 24 Mbits/sec [1] with beagleBone freq. of 1GHz**. Therefore can easily handle low resolution images in real time.

The input images will be of - **160x120 pixel size**
where image format will be **PPM (portable pixmap format) [2]**

which is **absolute raw uncompressed data of pixels**.
This is best suited due to **packed RGB data arrangement for easy manipulation using assembly code of PRU** .
Where for mode- **V4L2_PIX_FMT_RGB24 [3]**
Each pixel = 24 bits= 3 bytes
Size of image = 160 x 120 x 3= 56.25 Kb/image
112.5 Kb total size of stereo images

Images will be divided into 10 portions each and given to shared ram for PRU to process .

The PRU code sits and waits for a command. As the stereo correspondence algorithm is implemented in the PRU. The dense stereo algorithm **outputs disparity depth map**.

This is again retrieved in parts by using the shared memory and **dumped into a CAIRO [4] buffer** in portions.

The complete depth image is then created by the camera app and saved in a PPM format.

★PRU stereo vision code (true power of BeagleEyes)-

The **epipolar lines are completely parallel and horizontal** by carefully making the camera module.

Therefore **pixel shift search is brought down to a single-dimension** (rather than 2D), giving a

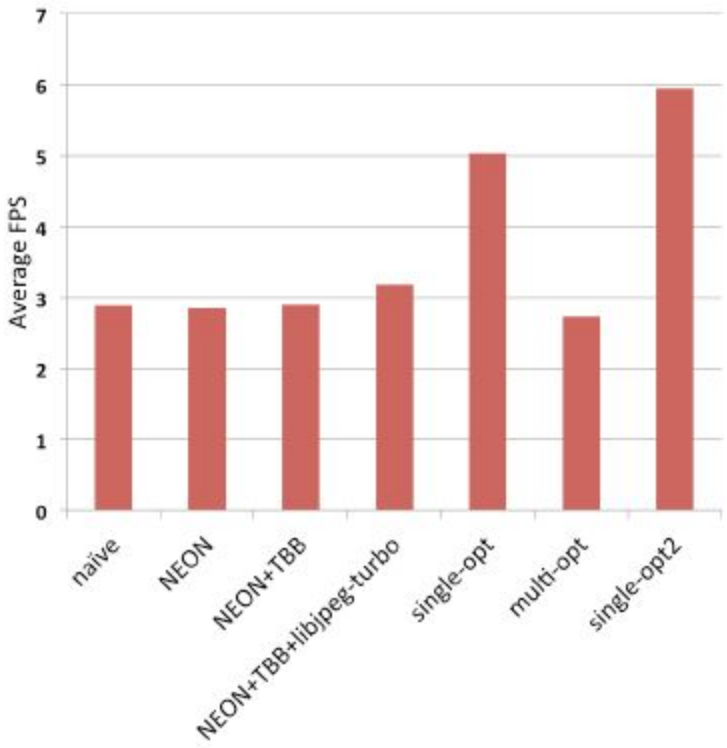
simple and realistic approach that is possible to implement in the PRU assembly language to get depth image by searching for pixel shift in single line to line comparison of both images.

The above algorithm implementation using intensity shift can be found here**[5].**

This will give real time power that is not possible using image processing libraries in linux, reason is-

No matter whichever library we use goes through a chain of abstraction layers and even has complex stereo algo implemented like the famous block-matching algo etc giving excessive cpu load and taking a lot of time to process. A practical implementation of this done before proves this as shown in **[6]**.

Even for Non-stereo vision output, by just giving a single video input, detect corner function in OpenCV and after implementing various optimisation method the result is only 6 FPS.



But in this project

Dual video input + depth image making + contour detection + roscopter protocol if done on single 1GHz onboard chip will surely lag real-time expectations.[7].

Therefore **PRU selection for stereo vision code is a MUST, IF we want to achieve real time processing** where input method as discussed above, by careful examination, USB protocol suits best.

Also **reason it wasn't implemented before is**-

- ❏ People are new to PRU and don't want to spend lots of time learning and implementing stereo vision code.
- ❏ People use their own image processing software and moves to other expensive higher clock and RAM allocated boards for better results.

OpenCV provides basic as well as advanced functions used in computer vision as an open source package. This library is configured with C++ and is used in this project. It does two things. -

1) **Correction algo implementation- using chessboard function in OpenCV**
   which creates **rotation/translation matrix and also helps in removing radial distortion created by the lens.** As the camera module is accurately made, the epipolar lines are almost parallel and horizontal of the two cameras. So the correction is not much needed but the code will still run once to tell the user if the module made by him is correctly aligned or not.

2) **Triangulation and contour detection-**
   The pixel intensity of the depth image generated will be feeded into the triangulation formula which will **then decide the distance of objects from the camera**. **OpenCV function Contour detection on the image will then decide the movement of MAV IN 4 DIRECTIONS. Also the work till here is completely based on creating stereo vision depth and detection, it can be used in all kind of robots be it aerial or terrestrial.**

Finally using-

roscopter, [8] a compact ROS package that allows serial communication with devices supporting the MAVLink protocol
(a standardised protocol used for communication to and from autopilots) will allows us to **take control over the autopilot and give commands to move MAV and avoid collision.**

For MAV i am already having the PixHawk Fire Cape [9] made by Erle Brain. That is the present implementation of Ardupilot on BeagleBone Black.**[10]**

My related codes-

- I have previously made an **autonomous pathfinding and obstacle avoidance** terrestrial robot using **overhead camera** and openCV library. By making some changes the openCV code needed for this project is ready to go.

- https://github.com/kartik-nighania/multiple-Object-Tracking-and-shape-detection
- I have **experience of making 3d-printing files using CATIA**. Below is a **link for 3D view of stereo frame**. Contains images and .stl file made for Logitech c270 camera. Many more supported camera frame making is added in timeline too.
- https://github.com/kartik-nighania/BeagleEyes---Stereo-Vision-BBB/blob/master/3D_printFile_Logitech%20C270.stl

❑ REFERENCES-

[1] videos capture at 18 MBits/sec
http://tinypic.com/r/smderr/9
[2] PPM format info
https://www.cs.swarthmore.edu/~soni/cs35/f13/Labs/extras/01/ppm_info.html
[3] format used of v4l
https://linuxtv.org/downloads/v4l-dvb-apis/packed-rgb.html
[4] Cairo Buffer used.
https://en.wikipedia.org/wiki/Cairo_%28graphics%29
[5] SIMPLE way to create DEPTH IMAGE
http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT11/node5.html
[6] unable to achieve Real Time behaviour
http://whatnicklife.blogspot.in/2010/05/beagle-has-2-eyes-opencv-stereo-on.html
[7] Arm core chip not fit for image processing. Page 8. Page 18-27
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-117.pdf
[8] ROSCOPTER package
https://erlerobotics.gitbooks.io/erlerobot/content/en/mavlink/ros/roscopter.html
[9] pixhawk fire cape
http://erlerobotics.com/blog/product/pixhawk-fire-cape/
[10] My mounted pixhawk ready to fly :)
http://tinypic.com/r/2emezj7/9

About me-

- Name - Kartik Nighania
- Website- www.KartikNighania.com
- College- National Institute of Technology, SURAT (SVNIT)
- Branch- Electronics & Communication Engineering
- College site- www.svnit.ac.in
- Email-id - kkstrack@gmail.com
- Contact no- 91-9624666836, 91-9925420155
- Nationality- Indian
- Language-- English and Hindi
- Time Zone- Indian Standard Time

- Softwares used- MATLAB, Netbeans, mySQL, Atmel Studios, Arduino IDE, Android studios, Processing, Fritzing, SimuLink, OpenCV, visual studios, Eclipse, QT, Proteus, CATIA, gitHub