running on it. Video capturing is performed by using a webcam that is connected to one of the USB ports. The webcam is capable of capturing 30 frames per second and the captured video frames are displayed on the touch screen that is connected to the GPIO pins. There are 40 GPIO pins available on the Raspberry Pi B+ model and out of which 26 pins are used to interface the touch screen to the Raspberry Pi board. The touchscreen used is a 4 wire resistive type, 3.2 inches TFT touch screen with the resolution of 320 x 240. The operating system is loaded on the SD card that is placed on the SD card slot of the board.

When the system is initialized to perform video processing, the camera is turned on and it captures the frames at the rate of 30 fps. Video processing algorithm is applied on the captured frames and in real time. In order to change the operation being performed the on video frames, the events are given though the touch screen. As the values of the coordinates of the touch screen are changed, video frames captured in real time are rotated in clock wise and anti-clock wise and also resized by performing zoom in and zoom out.

## 4. System Architecture

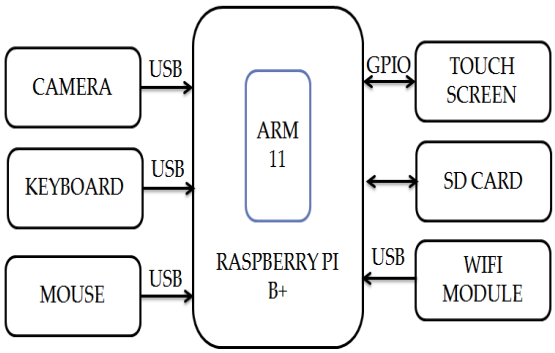


**Figure 1:** Block diagram of the proposed model

SD card containing the operating system is placed in the SD card socket. The keyboard and mouse are used for programming purpose which is connected to the hardware platform through USB. Webcam is connected via USB port. Touch screen display is mounted directly on the GPIO pins. The remaining pins are left as it is and hence no connections are made. Upon doing these initial connections the hardware is powered through the Nano USB port. The operating system takes some amount of time to boot and check whether all the interfaces are working fine. The system can be connected to the internet through the Wi-Fi adaptor via USB port. There is also an Ethernet port through which the system can be connected to the internet but having a wireless system is more advantages compared to wire.

**Figure 2:** Functional block diagram of the proposed model

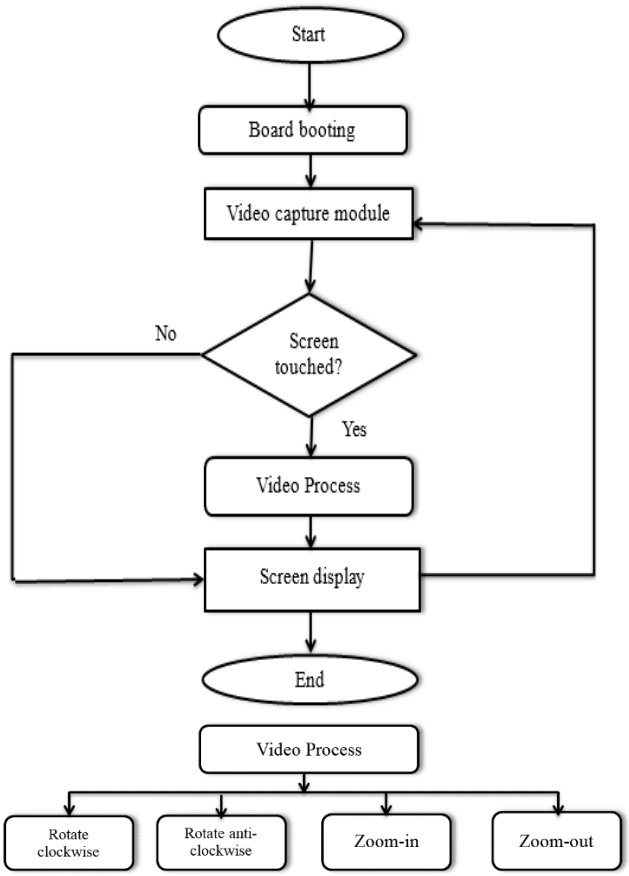


**Figure 3:** Overview of the model

The initial steps involve the powering up the Raspberry Pi. The OS takes some amount of time for booting in order to check the interfaces are working fine. After the boot is done, the video processing source code is run. The camera module starts capturing the frames and sends these frames to the processor and will be displayed on the touch screen display. If the screen is not touched then no further processing is done by the processor, it will display the captured frames on the screen. If the user touches the screen then the hardware performs the processing with the help of the graphic library and adds the changes to the captured video frames and displays them on the screen. Processing of the video frames includes rotating the image in clockwise and anti-clockwise and resizing. The resizing function includes the resizing of the window in which the frames appear or resizing the

frames directly and keeping the window size constant. There are many ways to rotate and resize videos and prominent among them are affine transformation technique and interpolation technique. The processing of the video frames is performed upon the events that are given to the processor through the touch screen. The processing and displaying of the video frames in real time involve some amount of delay. The delay is in some milliseconds; however this can be reduced to some fair amount of time by doing some fine tuning.

**Figure 4:** Lab setup of the proposed model

## 5. Results

When the source code is run on Raspberry PI B+, the webcam starts capturing the video frames and send these frames to the processor. If there are no touch events sensed by the processor then the received frames are directly displayed on to the touch screen display. When the user touches the screen, an event is sensed by the touch screen controller and this change is sent to the processor. The touch screen display is having X and Y coordinates which are used to know the area touched on the screen. The source code is written such a way that the value of the coordinate touched is stored and used for the next touch operation. If the value of the coordinate increases or decrease then there will be change in the processing of the video frames based on the X and the Y co-ordinates.

**Figure 5:** The original frame

**Figure 6:** Rotated frame in clockwise direction when x=346 and y=1623 co-ordinates are touched

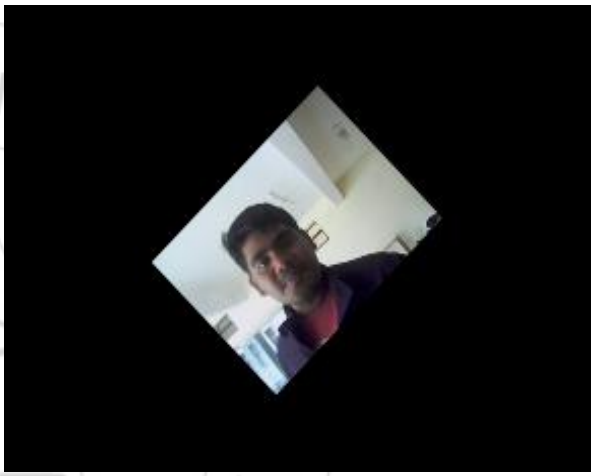

**Figure 7:** ROTATED FRAME IN ANTI-CLOCKWISE DIRECTION WHEN X=346 AND Y=861 CO-ORDINATES ARE TOUCHED

**Figure 8:** RESIZED (zoom-in) frame when x=1237 and y=1504 co-ordinates are touched

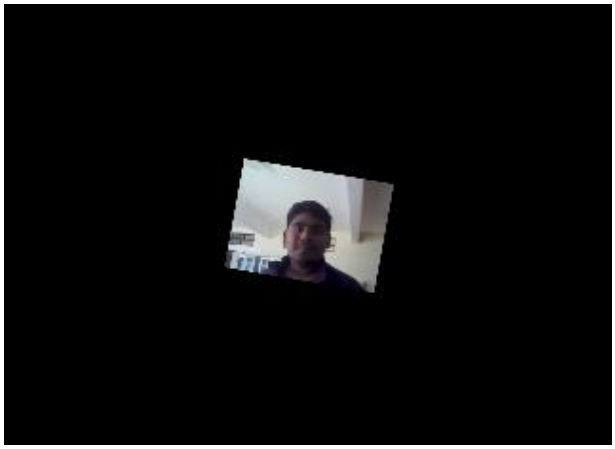

**Figure 9:** RESIZED (zoom-out) frame when x=389 and y=423 co-ordinates are touched

## 6. Future Work

The proposed paper is basically for single touch and this can be further improved to multi touches to perform various kinds of processing on the video frames. Also the coding can be improved to perform the processing on a particular area on the video frames on the screen such as drawing a circle on screen which can perform the operation of zoom in that perticular region, that is circled. The application can also be implemented on the capacitive touch screen which will improve the sensitivity of the touch. The application can be built on Android platform so that a mobile app can be developed for video processing particularly. This proposed applications can be used for various other kinds of projects in the field of survillence system, medical application, military applications,space application i.e in satellites and industrial automaton.

## 7. Conclusion

In this paper I have proposed a novel video processing application using a touch screen for mobile phones and other pesonal assistane deices which are capable of capturing videos. The source code includes the code for the touch events from the touch screen which are processed by the touch screen controller and sent to Raspberrry Pi processor and the video processing code. Because of lack of the resources such as the processing power and memory requirement, the processing of video frames was limited only to perform the rotation and resizing the frames in real time. Some amount of delay is observed in the processing of the video frames and this is reducing by optimizing further the video processing function and fine tuning the code.

## References

[1] The *Raspberry Pi* User Guide, co-authored by Eben Upton with *Gareth Halfacree*
[2] Learning OpenCVby Gary Bradski and Adrian Kaehler.
[3] An Embedded Digital Image Processing System using Blackfin 548 DSP by S. Mahesh Kumar, Anitha Julian.
[4] A Review of Image and Video Processing by Byeong-hokang
[5] Low cost smart security camera with night vision capability using RaspberryPi and OpenCVbyAbaya, W.F.; Basa, J.; Sy, M.; Abad, A.C.; Dadios, E.P.
[6] Porting the Linux Kernel to a New ARM PlatformWookey and Tak-Shing
[7] A brief introduction to OpenCV by Ivan Culjak, David Abram, TomislavPribanic, HrvojeDzapo, Mario Cifrek
[8] Hardware/Software Co-design of Embedded Image Processing System Using SystemCModeling Platform by Wang hongand Zheng Hong

## Author Profile

**Prasannakumar Malingaray Vaggi** was born in Ranebennur, India in 1991. He completed Bachelor of Engineering in Electronics and Communication Engineering in 2013 and M.Tech in VLSI Design and Embedded Systems in 2015, from Visvervaray Technological University Belgaum. Some of the projects undertaken include Visible Light Communication,Wireless sensor network for agricultural applications.