International Journal of Engineering & Technology, 8 (3) (2019) 261-269

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Comprehensive study and investigation of ROS for computer vision applications using Raspberry Pi

Jignesh Patoliya[1*] and Hiren Mewada[2]

[1,2]*Department of Electronics and Communication Engineering Chandubhai S. Patel Institute of Technology (CSPIT), Charotar University of Science and Technology (CHARUSAT)- Changa*
[*]*Email: jigneshpatoliya@charusat.ac.in*

Abstract

The machine in the form of the robots has a very large community which makes impressive progress in recent trends. Progressive examples of these types of robots are land based mobile robots, quadcopters, humanoid, etc. Motion tracking and object recognition is the base process in major robotic applications. For better flexibility and integration of robots with video processing applications, the ROS framework is largely used. The major issue with ROS is its latency and integrity. This paper investigates the integration of the ROS framework with OpenCV libraries on the Raspberry PI processor for video processing applications. In the proposed experiment setup, a camera node interfaced with the raspberry PI captures images and publishes it in ROS message form on a specific topic. The subscriber node converts ROS message into an image using cvbridge. Converted image is processed again using OpenCV library on the raspberry Pi board. The extracted information can be used to actuate peripheral devices interfaced with the raspberry Pi. An investigation of the raspberry Pi based implementation reveals that ROS introduces 0.63% overhead and optimum implementation on raspberry Pi can avoid the high configured computer and raspbeery Pi can process the video at 13 frames per second at most.

Keywords: *Object tracking, Robot operating system, raspberry Pi*

1. Introduction

Robotics is one of the emerging fields of control and automation. Human-Robot Interaction (HRI) is an essential requirement of robotics where the sample robot can behave like a human and think smartly in all versatile area. If the sample robot can work in all areas with a minimal error then HRI can replace human to perform hazardous jobs. Many challenges are involved in designing such a robot. To develop robots, many frameworks have been proposed by the researchers resulting in hassle-free prototype model development [1]. The video processing application demands high computational power and hence can be considered as high bandwidth data processing application. Various software is available to process the videos including MATLAB, OpenCV, R-tool, etc. However, the integrity integration of the software with ROS and latency on an embedded platform with ROS is were the big challenges for high bandwidth data processing applications. This paper presents a ROS system development using ARM-based processor which is the current state of art processor to develop various embedded applications. With this, following objectives are established to investigate ROS for computer vision applications (a) implementation of the ROS framework on ARM Processor (b) Integration of OpenCV libraries with ROS framework (c) Investigation of execution time for real-time video tracking applications (d) To compare the overhead of embedded Linux platform on an ARM processor with Linux platform of standard desktop PC with Intel processor. With these objectives, the paper is organized as follows: ROS framework, OpenCV and raspberry Pi based ARM processor are introduced in section II. Various computer vision applications developed using the ROS framework for embedded platforms are presented in section III. Section IV presents the system development where ROS is installed on raspberry Pi board and video tracking applications are used to investigate the ROS and OpenCV overhead.

2. Basic components used in the system

Robots can be considered as born of new-age technology where the majority of human works will be carried out by these robots. Previously uses of robots were limited to automated manufacturing units only. However, advancement in the technologies like software formation, fast and small size smart hardware developments has changed the role of robots from industries to social, medical, tourism and all majority of areas to reduce the human efforts with a better quality of work. The major challenges involved in robotics applications are a wide variety of framework, high computation work to be done in a shorter time. And writing the program for the robotics applications is a time-consuming process and requires great training at the hardware side. Thus To facilitate fast developments with the consideration of these aforementioned

Copyright © 2019 Author. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

challenges, few frameworks were proposed by the researchers. Few of them are as follows:
A block-based programming software CoBlox was used and its comparative study for robotics programming environment was pretested in [2]. Where authors reported a subjective analysis of the ease of software utilization and development time. Other proposed alternates are Mobile Robot Programming Toolkit (MRPT) [3], Microsoft Robotics Developer Studio(MRDS) [4], Gazebo, USARSim [5], SARGE [6], ROS, UberSim, etc. Gazebo allows the 3D design of an outdoor and indoor environment for robotics applications. USARSim is an open source tool providing the control of robots using TCP Support packages. SARGE provides the implementation of Robotics applications in-game engine. In comparison with discussed simulators, robotic operating system (ROS) is the most flexible and open source tool supporting a wide range of robotics applications. The support provided by the ROS like the addition of new packages over the development time, updating with new libraries of intended tasks in robotics and support of OpenCV tool for computer vision applications makes this tool strengthen over the other available Robotics simulators. Therefore, this paper proposes the analysis ROS in Computer vision based robotic applications. With this consideration, the basic tools required along with ROS are discussed in the following sections.

2.1. Robot Operating System (ROS)

ROS is an open source Linux based framework designed to fulfill and implement the various kinds of functionalities on a robot. One of the advantages of ROS is that it supports major widely used programming languages like C, C++, Python, and Java, etc. The Hierarchical structure of ROS deployed on Hardware is shown in Fig. 1. Detail features of the ROS can be obtained from [7].

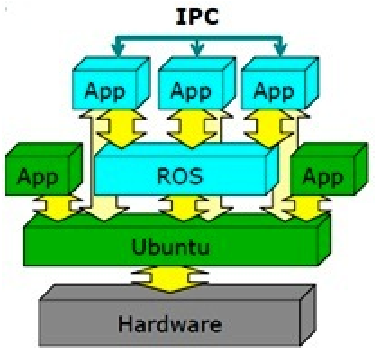


Figure 1: Hierarchical structure of ROS deployed on Hardware

Any general purpose processor, Embedded Processor, Embedded development boards such as raspberry Pi, Beaglebone Black, Jetsun can be considered as hardware/ heart of robot in Fig.1. ROS performs node based peer to peer communication where each node needs to published and subscribes the messages to be communicated with other nodes. Architecture widely adopted to design a robot using Publisher / Subscriber mechanism is shown in Fig.2. This utility is used for the multifunctional robot in which even if one of the functional nodes of the robot crashes, it doesn't stop the entire robot functionality. Here Camera node acts as a publisher and image processing node is subscriber node. The camera node register topic in ROS core to publish an image to other nodes. The published image is converted into the message. On another side subscriber node continuously query the topic and subscribe the message from the topic on which message is published. After subscribing the message, the node will take actions accordingly. Applications developed using ROS are discussed in section II (c).

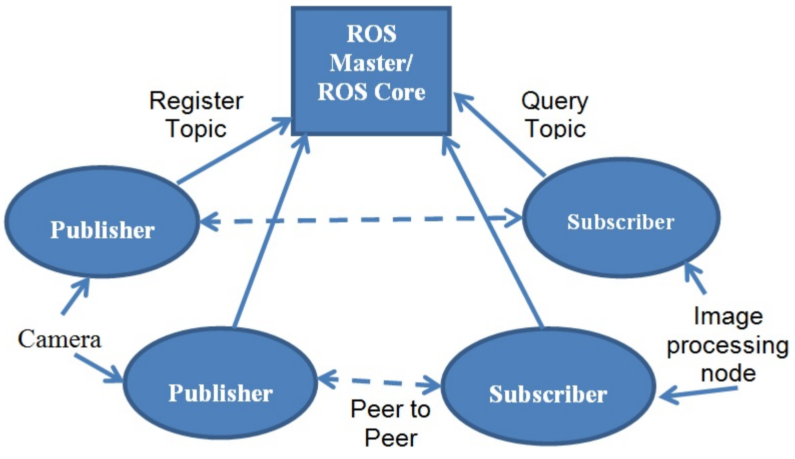


Figure 2: ROS Architecture

2.1.1. ROS Compatible Image and Video processing software

Both OpenCV using C++, OpenCV with python and MATLAB are the most popular software for image and video processing application amongst the researchers. MATLAB is versatile and it contains very powerful matrix laboratory. All mathematical stuff like partial differential equations and all large linear system can be easily solved and implemented in MATLAB. The large and strong support of signal processing

Parameter	MATLAB	OpenCV
Speed	3-4 frames analyzed per second Slower	30 frames analyzed per second Much Faster
Resources needed	Huge memory required for processing (giga bytes)	Less memory required for processing (Mega bytes)
Cost	Commercial	Open source
Portability	Run on windows, Mac & Linux	Run on windows, Mac & Linux
Ease of use	Relatively easy and pretty high-level scripting language No worry about libraries, declaring variables, memory management or other lower-level programming issues	Comparatively Complex
Memory Management	Memory allocation in smart way (No issue of memory leakage)	Issue of memory leakage
Integrated Development Environment (IDE)	Own Integrated development environment	There is no specific IDE

toolbox and image and video processing toolbox with the exceptional implementation of optimization makes it most suitable for the researchers. However, OpenCV with python /C++ is now a day offers the best alternative due to its open-source advantage in comparison with MATLAB. The second constraint of MATLAB is its execution time. Often, MATLAB needs the support of the C programming language for faster execution. Though OpenCV has limited support of Machine learning algorithms, due to scientific support provided by Python for OpenCV, it becomes the strongest competitor over the MATLAB. The further comparative analysis of MATLAB and OpenCV is presented in table 1. It can be summarized that OpenCV with C++ shall be selected for the product development but for the prototype model implementation and testing of the algorithm, MATLAB and OpenCV with python are suitable. The real-time embedded applications need faster execution and selection of embedded hardware plays an important role in such applications. The following subsection discusses the role of hardware in ROS for high computational computer vision applications.

2.2. Hardware Support for ROS

The standalone robot requires an embedded platform to perform the desired task in real time. All robots contain the embedded system incorporated with sensors, motors, and various communication peripheral components. The reason for the embedded platforms in addition to real-time processing is its low cost, size and weight and faster execution of code in comparison with general purpose PC. Vast development in VLSI technology has provided huge varieties of an embedded system including x86 architectures (i.e. mini-ITX, PC/104), ARM-A processor (i.e. raspberry Pi, Beaglebone, Sumstix, etc.), ARM-M processor like embed, STM32 discovery, and AVR controller (i.e. Teensy 2.0, Arduino, etc.). The following Fig 3 presents the connection of ROS for the embedded platform. Arduino and raspberry Pi are the most attractive embedded system for Robotic applications. ROS can run on raspberry Pi or ROS can be integrated with Arduino microcontroller using the serial connection. However, due to limited resources available on board of Arduino, it is used for controlling applications more in comparison with the processing application. Whereas, the raspberry Pi allows execution of ROS onboard and hence acts as a single board computer. Due to the support of multiple thread execution on raspberry Pi, it is most suitable for the ROS based computer vision applications. Hence, this paper provides the analysis of ROS environment on raspberry Pi embedded system with the consideration of tracking application.

3. Literature Reviews

Many computer vision based robotic applications have been developed by the researcher. Robotics applications may deal with machinery and structure of the robot; it may deal with robot intelligence and control application or observation of the robots. Major computer vision applications contain robot navigation and localization. Mishra and Javed [9] proposed a service robot platform using ROS for self-localization and mapping in an indoor environment. They adopted adaptive Monte Carlo algorithm for tracking and localization in the visible band using RGB-D camera.

Pieter Simoens et al [10] discussed technical challenges in the internet of robotics things (IoRT), an integration of the Internet of Things and robotics. They classified the robotic systems with nine attributes like perception, manipulation, motion, decisional autonomy, cognitive, configurability, adaptability, dependability, and Interaction. They used the ROS framework to connect the mobile robots to IoT service. Rebecca A. Greenberg [11] investigated the feasibility of human tracking using ROS and MATLAB. The human segmentation was achieved by calculating the histogram's depth and region properties of RGB-D frames. Later Kinect sensor based skeleton tracking algorithm was used to track human. Alex Goldhoorn et al [12] presented a simulation of two tracking approaches: reinforcement learning algorithm and particle filter with the consideration of multiple (i.e. five) robots. Each robot explored the close location of the human. These algorithms were tested using two robots and analyzed the searching time with an approximation of human distance from the robot. Fang et al [13] proposed robust and effective a multi-feature tracking strategy when the target is moving speedily. They integrated the color histogram similarity features and histogram of the gradient (HOG) features with the motion model in tracking. The presented approach was robust to the occlusion, color intervention and loss of target. However, in the development of all these applications, researchers have used general

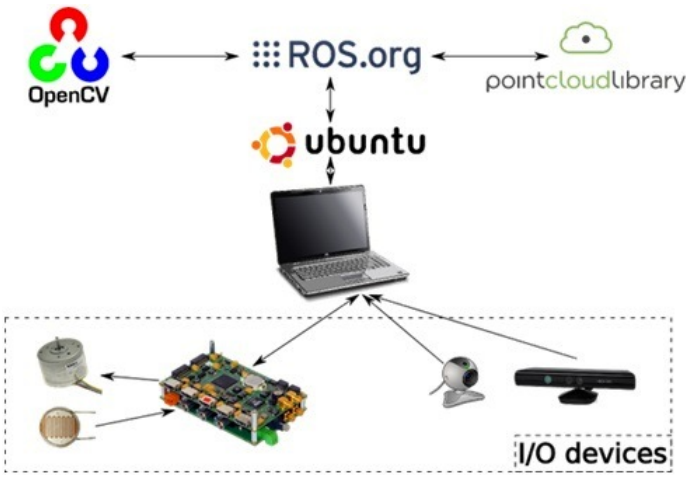


Figure 3: ROS support and hardware interface [8]

purpose PC for ROS installation and they controlled the hardware actuators using this PC.
Bisi et al developed a robot control system using electromagnetic waves [14]. They used a Teensy microcontroller and raspberry Pi to control the robots and demonstrated navigation of the robots. The results are evaluated by checking collision avoidance. Santosh Krishna et al [15] were tried to develop a robot for an industrial environment where the raspberry Pi is connected with various sensors and robot controlling devices (i.e. motors). The ROS generates the control signals and these signals were communicated to the PI using Wi-Fi. Finally, data are stored in the cloud for further processing. Ponnu and George implemented simultaneous localization and mapping (SLAM) robot where raspberry Pi-2 is the heart of the robot [16]. H. Peel et al have presented an approach of the autonomous navigation system where they used Ubuntu operated raspberry Pi to access ROS on board [17]. In summary, very few robotic applications are developed using the embedded platform without the support of PC. This is the main challenge as application need to be customized based on available limited resources. Therefore, the experiment set up using raspberry Pi based embedded system is presented to investigate the ROS integrated computer vision application in the following section.

4. ROS Targeted Tracking Applications

Paper aims to implement ROS environment in an embedded system and to investigate this system for the real time video processing applications. Few robot based tracking applications are tracking of people and face recognition and object tracking in the indoor environment. This paper implements the said applications on the embedded device having low resources. As discussed in section II, ARM processor based raspberry Pi is used in embedded system for the implementation. The major constraint of low resources makes the implementation of all these applications challenging. This section initially introduces these applications followed by experiment setup and result discussion.

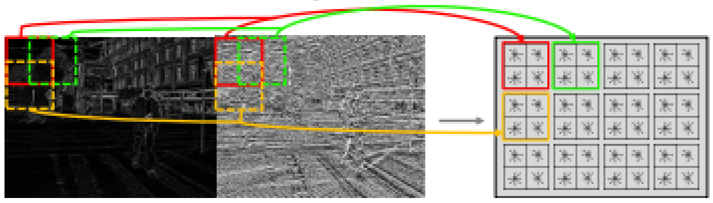
4.1. Person tracking

Locating and tracking a person in digital image/video is a key process for robotic applications. Variation in cloths, poses and background makes the application more challenging. The real time limitations make it more attentive for the implementation of hardware. Local gradients and its orientation can characterize the human in a better way. Histogram of Gradient (HoG) computes the occurrence of gradients in the image. HoG involves two processes: the computation of Gradient i.e. changes in the intensity and the computation of histogram i.e. direction of change in color.
In the proposed experiment, local gradient is calculated using 16 x 16 windows with 50% overlap. The next histogram is obtained using 9-bin of the orientation. In each block, all values are normalized to avoid contrast variation. To implement on device with low resources, parallel computation in Gradient and histogram formation is employed as expressed in Fig. 4. The pseudo-codes for these two processes are as follows:

4.2. Face detection

Face detection is one of the most emerged computer vision application. Face detection played a vital role in applications where security is a major aspect. However, in robotic applications, it is used for human-machine interaction. An example includes service robots, where face identification is required to command the robot from a known identity only. A robot can be used to serve people based on their identity. Therefore, this paper implements and studies the face recognition algorithm for robotic application. Several algorithms using feature extraction has been proposed in the literature. In the proposed experiment, a popular method of Haar wavelet based face recognition method has been used. This algorithm works by training the cascade classifier with images having faces (i.e. positive images) and non-faces (negative images). These training images are convolved with Haar wavelet filter to extract various edges, lines, and rectangle features. The simple convolution operation yields too many feature sets causing large time for recognition. Therefore, the convolution operation is performed on face region only and further feature reduction has been achieved using the Adaboost classifier which is referred to as the weak classifier. These features are grouped at the final stage. The weighted sum of the weak classifier is used to classify the face region. The Haar features calculation is fast and calculation time is constant. The overall flow of the cascade classifier using Haar features is shown in Fig 5 below:

(a) Input image



(b) Gradient, corresponding orientation and histogram in 8x 8 window

Figure 4: HOG Calculation (adopted from [23]).

Table 1: pseudo code HoG Process

```
Input :Frame I, size [M,N]
Output: Magnitude M and orientation O, size [M,N]
%Parallel computation of gradinet
For x=0 to M and y = 0 to N
$Gx = I(x,y)–I(x-1,y)$
$Gy = I(x,y)–I(x,y-1)$
$M(x,y) = sqrt(Gx^2 + Gy^2)$
$O(x,y) = arctan(Gy,Gx)$
Input: Magnitude M and orientation O, size [M,N]
Output: Fb[Hb][Wb][S]
for y=0 to Hb and x=0 to Wb do
```

4.3. Ball detection and tracking

Following an object having a particular color by the robot is another interesting application to demonstrate the use of robots for a defined path. The simplest approach of color detection has been used to track the ball. Initially, RGB color mapped frame is converted to HSV model. Then the ball is detected by taking a difference between two consecutive frames. To avoid false detection, the morphological process followed by erosion and dilation operations is used. Later, the contour of the ball is calculated with its centre coordinates. This center coordinates help to draw sufficient window covering the ball. The entire process has been interfaced with ROS to make it compatible with the robotic application. This process has been explained in section IV.

5. System Development of Targeted ROS tracking application

5.1. Integration of ROS and OpenCV

Integration of ROS and OpenCV is required when the robot taking the input data in the form of Images. The important feature of ROS is it can publish and subscribe the data in form of message only. So when the input data is in form of the images then it must be converted in to message form. Before publishing the data we must apply some image processing algorithm on the image data which is in the form of a message. For image processing operation on data requires integration of OpenCV and ROS. Fig. 6 shows the integration of ROS and OpenCV. As shown in Fig. 4 CVBridge is an important element for integrating OpenCV and ROS. It takes ROS image message as input and through CVBridge it is converted into OpenCVlpll image which is processed image. The process of passing the captured frame from the camera to ROS using message formation is presented in Fig. 7 below.
The communication between the ROS and sensor/actuators requires registration of the message to the pre-defined ROS topic. Subscriber node is the main processing node and it waits for events by subscribing for messages on a ROS topic. When the Subscriber node receives a message, it converts that ROS message back into the OpenCV frame. It then applies various image processing algorithms on top of the converted OpenCV frame to obtain a resulting image. The resulting image is then published further to another node (for example, to store the

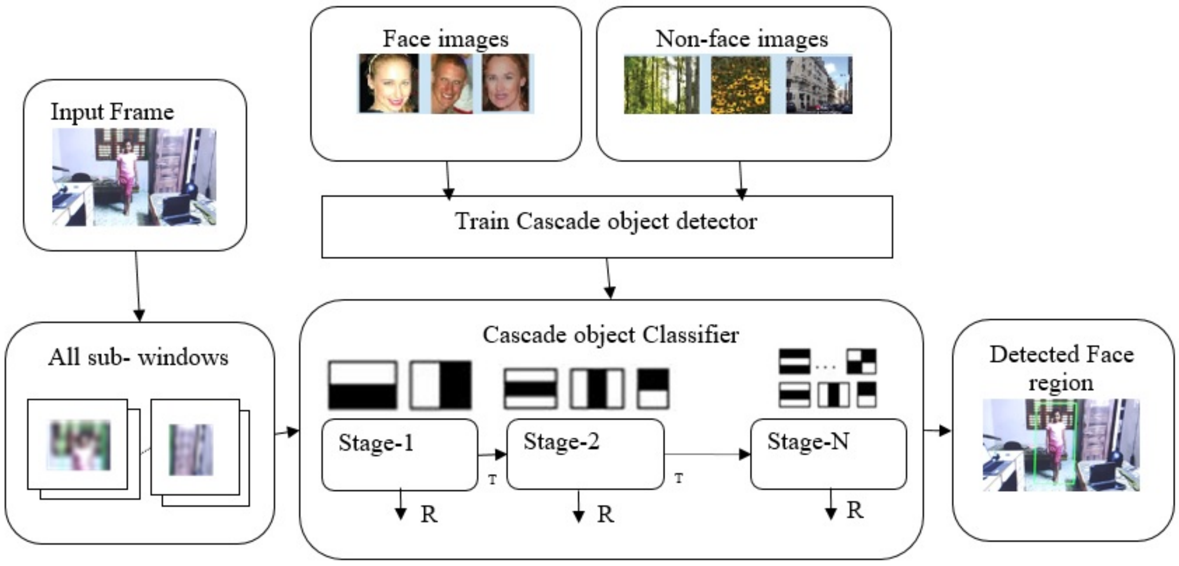


Figure 5: Haar Feature based face detection process (T indicates True sub-window, R indicates rejected sub-window)

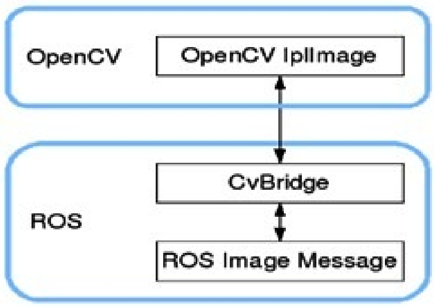


Figure 6: Integration of OpenCV & ROS [18]

results in a database). Additionally, based on the resulting image, the code further decides the type of action to be performed to control the connected actuator (for example, actions to control camera movements to continuously track a person or object). The message flow for the publisher node and subscriber node is expressed in Fig 7 and 8 respectively.

5.2. System development using raspberry Pi and ROS framework

Block diagram of the proposed system is presented in Fig 9. It contains three major blocks: Sensors, Publisher/Subscriber node, and Actuators. For robotics applications, there are four major sensors covered like Camera – Eye Sensor, Compass Sensor –for direction purpose, Ultrasonic sensor for direction purpose and IR sensor – for range purpose. This is not limited to only four sensors. More sensors can be connected as per the need of the application. raspberry Pi or any IoT module (i.e. ESP-32) play a major role in Publisher / Subscriber node along with the cloud services. The third section introduces the actuator part of the system whereas an actuator we can use DC motor, Stepper motor or Servo motor. Actuators are used to control the robot movement. The overall experimental setup can be expressed as follows:

- Camera Sensors captures the image/video and given to the raspberry Pi for further processing.
- raspberry Pi module works as a processor unit and a publisher node. ROS is deployed on the raspberry Pi.
- Specifically for the image data, it will convert into the message by the ROS. Because ROS can publish or subscribe the data in form of message only.
- Once the raspberry Pi gets the data from the sensor, it will publish the same on to the cloud.
- On the other side, IoT module working in the subscribing node will subscribe to the data in the form of a message from the cloud.
- According to the information given in the message, the IoT module will take the actions and run the respective actuator.

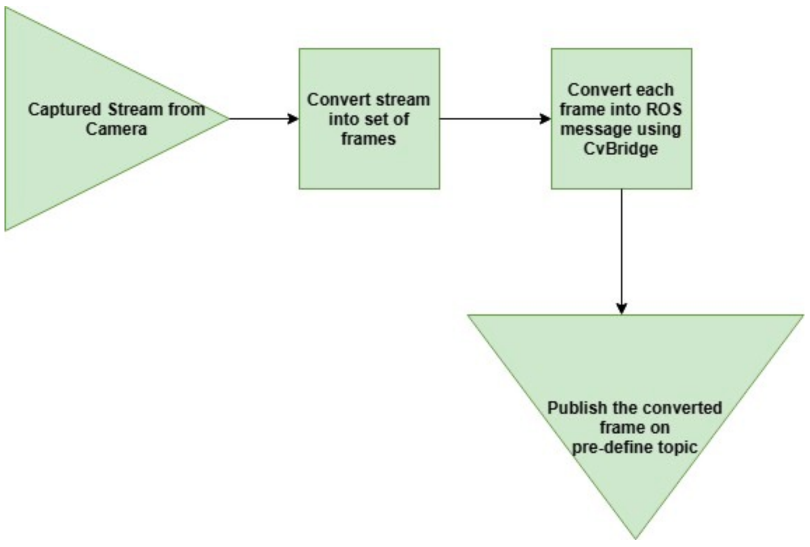

Captured Stream from Camera
Convert stream into set of frames
Convert each frame into ROS message using CvBridge
Publish the converted frame on pre-define topic

Figure 7: Publisher Node

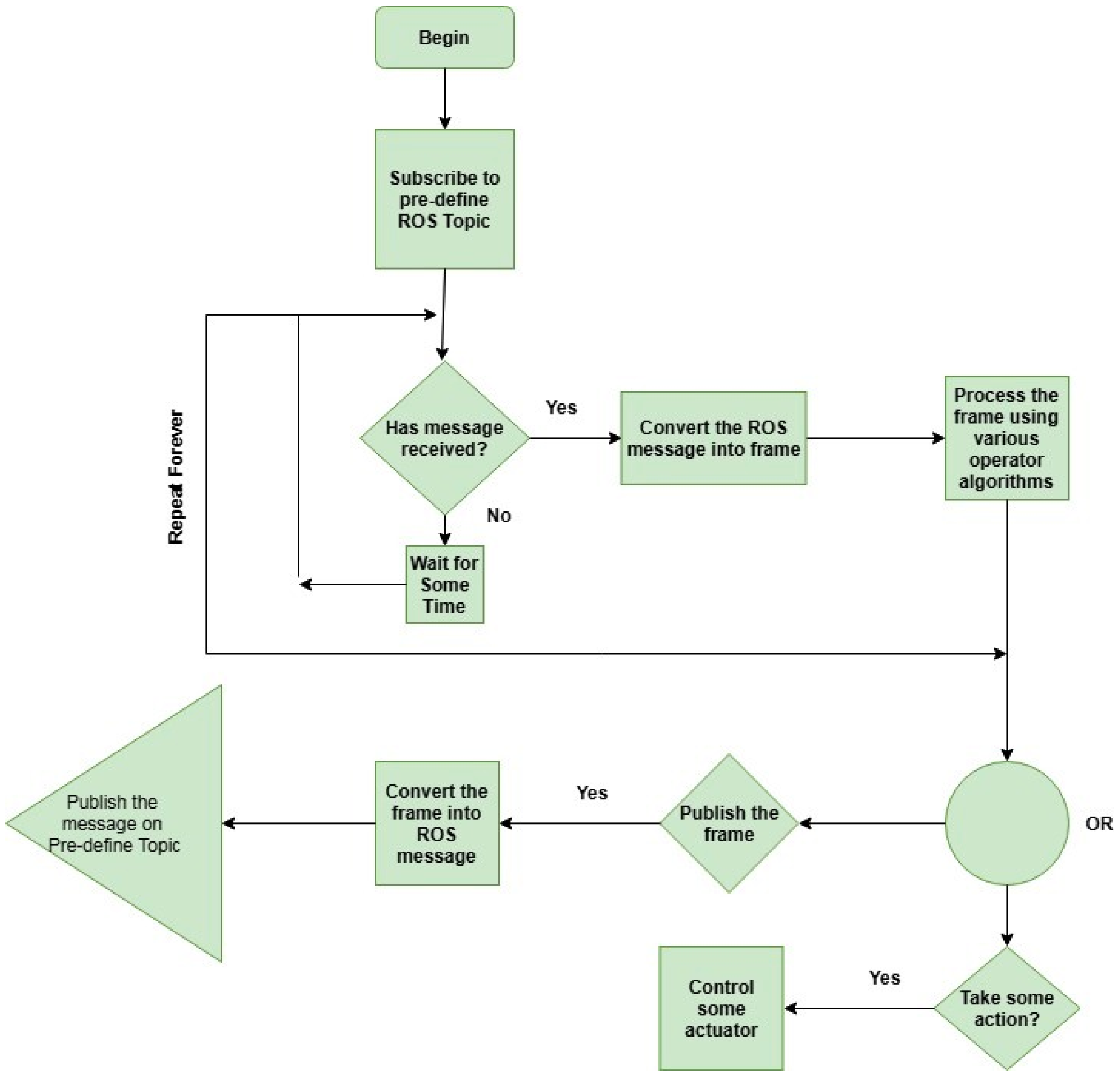

Begin
Subscribe to pre-define ROS Topic
Repeat Forever
Has message received?
Yes
No
Convert the ROS message into frame
Process the frame using various operator algorithms
Wait for Some Time
OR
Publish the frame
Yes
Convert the frame into ROS message
Publish the message on Pre-define Topic
Take some action?
Yes
Control some actuator

Figure 8: Subscriber Node

6. Test Results & Discussion:

The aim of this paper is to investigate the ROS environment on an embedded platform without using the general PC. The actual experimental setup is presented in Fig 10.

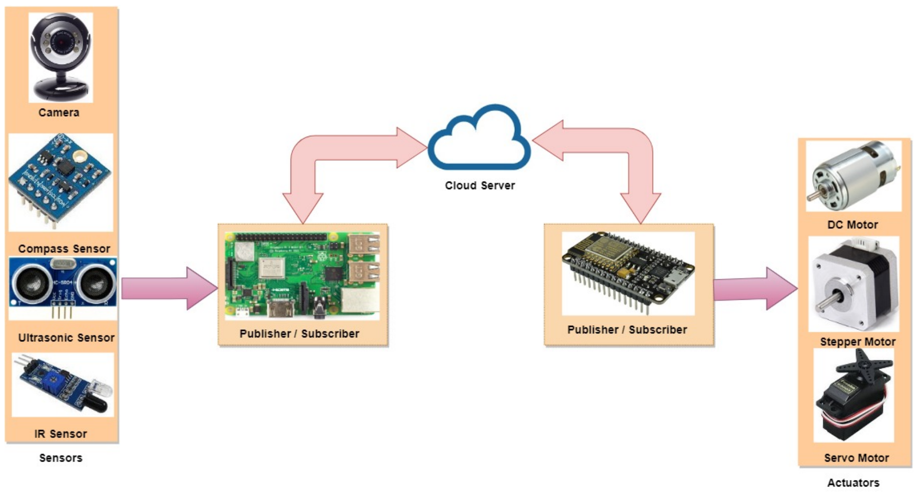


Figure 9: Proposed Block Diagram

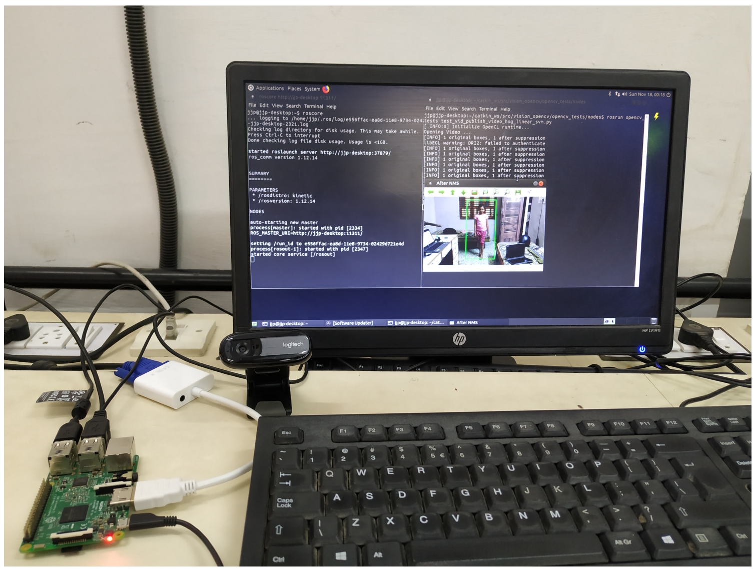


Figure 10: Test setup of ROS integrated embedded system for video tracking application

ROS is implemented on raspberry Pi OS (a Linux flavor OS) and it is deployed to raspberry Pi board. The PI camera is attached to the PI board captures the video. Three different computer vision applications of tracking including (a) Person tracking using Histogram of Gradient and SVM based classification [19](b) Face detection using HAAR wavelet [20] (c) Ball detection and tracking using threshold-based approach [21] are used in the experimental study. The obtain codes are modified to integrate with ROS and to fit in the raspberry Pi environment. In all applications, all video has a resolution of 1920 x 1080 (HD resolution). As real-time video processing has been targeted for the setup, the execution time has been considered as the comparative parameter for a desktop PC has an i3 processor and 8 GB RAM and raspberry Pi based embedded system. To investigate the overhead due to ROS and OpenCV, four experimental setups are tested as follows: (i) OpenCV based simulation of algorithm on raspberry Pi OS (RPi OS) (i.e. ROS is not used) (ii))OpenCV based simulation of algorithm on raspberry Pi OS with ROS (iii)) OpenCV based simulation of algorithm on Desktop PC (i.e. ROS is not used) (ii)) OpenCV based simulation of algorithm on Desktop PC with ROS. All algorithms were executed for 30 frames per second. The execution times for the selected tracking algorithm with described setups are presented in table 2. All experiments are repeated number of times by tuning the set of parameters and average execution time is considered to establish a comparison between two systems. Table 2 summarised that ROS introduces very little overhead of 0.63% to the OpenCV process either on Desktop PC or on an embedded system. However, there is a vast execution time difference for the Desktop PC and raspberry Pi based Embedded System due to the differences in onboard resources like RAM memory. In addition, care must be taken that it does not occupy the entire onboard memory of raspberry Pi. Further improvement in the speed of algorithm can be obtained by modifying the algorithms. As an example, if the face recognition algorithm listed in table 2 is implemented using the feature sets of Eigenface and local binary pattern then we can achieve the speed of 13 frames per second [22].

7. Conclusion

This paper proposed the ROS framework implementation on an embedded platform for robot-based computer vision applications. The initial study was presented on the availability of robot compatible software and it has been observed that ROS is the most suitable choice for the embedded platform due to its flexibility and compatibility for computer vision applications. Then OpenCV with python has been considered as a better alternative for ROS based application in comparison with other available tools due to its faster execution time. The

Table 2: Execution time comparison for various video tracking applications using ROS and OpenCV on i3 processor and raspberry Pi Processor

Application	**raspberry Pi**		**i3 Processor**	
	RPi OS	**ROS + RPi OS**	**Ubuntu**	**ROS + Ubuntu**
(a) Face tracking using HAAR transform				
1st trial	60.7085325717926	64.0305240154266	5.3431134223938	5.6197159290314
2nd trail	61.0927023887634	64.6293199062347	5.1572413444519	5.4171621779947
3rd trial	61.0623686313621	64.4575998783111	5.3396003246307	5.5988240242004
4th trail	61.2242202758789	64.5443081855773	5.1935508251190	5.4645090103149
5th trail	60.4388854503631	6.76640009880066	5.1839239597320	5.4422800540924
6th trial	61.1919252872467	64.4635298252105	5.2050473690032	5.4689500331878
(b) Person detection using histogram of Gradient and Linear SVM classifier				
1st trial	41.2345154285430	41.2672190666187	2.6499180793762	2.6563589572906
2nd trail	40.8422560691833	40.8765630722045	2.6125936508179	2.6187310218811
3rd trial	41.0494439601898	41.0839579109377	2.6852788925170	2.6916329860687
4th trail	41.4808056354522	41.5146629810333	2.6824131011963	2.6889839172363
5th trail	41.4150753021240	41.4488361907959	2.6451706886291	2.6516020298004
6th trial	41.4077975749969	41.4432990550994	2.6421489715576	2.6485469341278
(c) Ball tracking using thresholding approach				
1st trial	28.5023989677000	28.5369129184479	1.2002069950100	1.2065610885616
2nd trail	28.6445100307000	28.6782709193719	1.1789331436200	1.1850705146832
3rd trial	28.2672221661000	28.3010795116811	1.1837930679300	1.1903638839700
4th trail	28.3583700657000	28.3938715458025	1.1823208332100	1.1887187957802
5th trail	28.4376459353200	28.4714068239919	1.1638250351000	1.17025637627126
6th trial	28.4813457620000	28.5151066506719	1.20625090599000	1.21268224716126

real-time process also depends on the selection of an embedded platform. Section II presented a study on available embedded platforms and raspberry Pi is suggested as a standalone embedded system to develop the entire applications. To validate the selection of suggested software and hardware, an experimental setup is developed and a comparative analysis between the Intel i3 processor and raspberry Pi embedded system is presented. The experiment setup reveals that Raspberry can process the video at max 13 frames per second and ROS framework introduces 0.63% overhead in the system. However, to avoid the heavy Desktop PC during the execution of robotics applications in real time, an algorithm needs optimization in computation cost or it requires a more powerful standalone PC configuration.

References

[1] Kramer, J., Scheutz, M, "Development environments for autonomous mobile robots: A survey", *Autonomous Robot*, Vol.22, No.2, (2007), pp.101-132.

[2] Weintrop, D., Afzal, A., Salac, J., Francis, P., Li, B., Shepherd, D. C., Franklin, D. , "Evaluating coblox: A comparative study of robotics programming environments for adult novices", *In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, (2018), pp.366-377.

[3] Mobile Robot Programming Toolkit available at *https://www.mrpt.org/*

[4] Microsoft Robotics Developer Studio 4 available at *www.microsoft.com/en-in/download/details.aspx?id=29081*

[5] S. Carpin, M. Lewis, J. Wang, S. Balakirsky and C. Scrapper, "USARSim: a robot simulator for research and education", *Proceedings 2007 IEEE International Conference on Robotics and Automation*,Roma, 2007, pp.1400-1405.

[6] SARGE, Available at*http://sarge.sourceforge.net/page11/page11.html*

[7] Robot operating system available at *http://www.ros.org/about-ros/*

[8] ROS conceptual flow available at ref*http://gtms1318.wordpress.com/*

[9] Mishra, R., Javed, A. , "ROS based service robot platform", *In 2018 4th International Conference on Control, Automation and Robotics (ICCAR)* , Vol.22, No.2, (2018), pp.55-59.

[10] Simoens, Pieter, Mauro Dragone, and Alessandro Saffiotti , "The Internet of Robotic Things: A review of concept, added value and applications", *International Journal of Advanced Robotic Systems*, Vol.15, No.1, (2018), 1729881418759424.

[11] Greenberg, Rebecca, "A. Investigating the feasibility of conducting human tracking and following in an indoor environment using a Microsoft Kinect and the Robot Operating System.", *Naval Postgraduate School Monterey United States*,(2017).

[12] Goldhoorn, A., Garrell, A., Alquézar, R.,Sanfeliu, A. , "Searching and tracking people with cooperative mobile ro-bots. Autonomous Robotsy", *Autonomous Robot*, Vol.42, No.4, (2007), pp.739-759.

[13] Fang, F., Qian, K., Zhou, B., Ma, X. , "Research and Implementation of Person Tracking Method Based on Multi-feature Fusion", *In International Conference on Intelligent Robotics and Applications*, (2017), pp.141-153, Springer, Cham.

[14] Bisi, S., De Luca, L., Shrestha, B., Yang, Z. and Gandhi, V. , "Development of an EMG-Controlled Mobile Robot", *Robotics,*, Vol.7, No.3, (2018), pp.36-42.

[15] Krishna, BV Santhosh, J. Oviya, S. Gowri, and M. Varshini, "Cloud robotics in industry using raspberry Pi", *In 2016 Second International Conference on Science Technology Engineering and Management (ICONSTEM)*, (2016), pp.543-547.

[16] GauthamPonnu and Jacob George, "Real-time ROSberryPi SLAM Robot .", *Thesis of Master of Engineering, School of Electrical and Computer Engineering of Cornell University*,(2016).

[17] Peel, H., Luo, S., Cohn, A. and Fuentes, "R. An improved robot for bridge inspection ", *In Proceedings of the 34th ISARC*, 2017, pp.663-670.

[18] Lentin Joseph , "Mastering of ROS for robotics programming ", *Packt publishing*, 2015, chapter 8.

[19] Adrian Rosebrock,"Pedestrian Detection OpenCV code" *https://www.pyimagesearch.com/2015/11/09/pedestrian-detection-OpenCV/* 2015.

[20] James Bowman "Face detection code" available at ref$http : //docs.ros.org/jade/api/OpenCV_tests/html/rosfacedetect_8py_source.html$

[21] "Ball tracking code" available at ref*http://www.robot-home.it/blog/en/software/ball-tracker-con-filtro-di-kalman/*

[22] "Face recognition using Local binary Pattern" available at ref*https://www.pyimagesearch.com/2018/06/25/raspberry-pi-face-recognition/*

[23] Campmany, V., Silva, S., Espinosa, A., Moure, J. C., Vázquez, D., and López, A. M. "Development environments for autonomous mobile robots: A survey", *GPU-based pedestrian detection for autonomous driving*, Vol.80, (2016), pp.2377-2381.