

Project Title

**ROS Image Processing node development using webcam,
OpenCV 3 and C++**

Developed by: MSc. Shiva Agrawal

Place: Germany

Date: November 2018

Content

Project Title	1
Content	2
List of Figures	3
1. Introduction	4
1.1. Project description	4
1.2. Outline.....	4
2. ROS and related packages.....	5
2.1. ROS (Robot Operating System)	5
2.2. ROS package.....	5
2.3. vision_opencv	5
2.4. image_transport	5
3. Concept development and Implementation.....	6
4. Test and results.....	8
5. Conclusion and Future Scope.....	10
6. References	10

List of Figures

Figure 1 - Process Flow Chart-----	6
Figure 2 - CmakeLists.txt-----	7
Figure 3 - usb_cam -----	8
Figure 4 – cameraImageAquisitionAndProcessingNode -----	8
Figure 5 - Raw Image (left) and Processed image (right) -----	9

1. Introduction

1.1. Project description

Vision sensor (camera) is one of the prominent sensors for robots and self-driving cars. In order to interface the camera with ROS and then do the image processing, some steps are required. As camera is very famous among ROS community, many drivers are already written to use the camera (webcam or USB cam) and get the image directly as ROS sensor_msgs type.

Hence instead of re-inventing the wheel, I have used the **usb_cam** [1], which provides the webcam images at topic **usb_cam/image_raw**.

Next step after reading the camera image is to do the necessary image processing and then again publish the processed image data, so that next node of the processing chain can use it for further tasks. This part is the main aim of the project where a ROS node is developed to achieve this purpose.

For the project, ROS kinetic is installed on Ubuntu 16.04. OpenCV3 and C++ are used for image processing and webcam of my Laptop is used to acquire real time images.

The following ROS stacks and packages are also used

- **vision_opencv** [2]: The vision_opencv stack provides packaging of the popular OpenCV library for ROS. The two main packages provided by this stack are:
 - **cv_bridge** [3]: Bridge between ROS messages and OpenCV. It is used to convert ROS images to CV images and vice-versa.
 - **image_geometry**: Collection of methods for dealing with image and pixel geometry
- **Image_transport** [4]: It should always be used to subscribe to and publish images. It provides transparent support for transporting images in low-bandwidth compressed formats.

1.2. Outline

- Chapter 1: Project description and outline of the project report
- Chapter 2: ROS and related packages
- Chapter 3: Concept development and Implementation
- Chapter 4: Test and results
- Chapter 5: Conclusion and Future Scope
- Chapter 6: References

2. ROS and related packages

2.1. ROS (Robot Operating System)

ROS (Robot Operating System) provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. ROS is licensed under an open source, BSD license. The official website of the ROS www.ros.org is full with everything one needs to start with ROS [5].

Different versions of the ROS are available for the user purpose, but I have used ROS Kinetic.

2.2. ROS package

Software are organized in ROS using packages. Each package then further contains one or more nodes (a process that performs computation), dataset, configuration files, 3rd party software piece, robot model files, launch files, service and parameter files, etc. [6]

Common files and directories inside a package directory are:

- include/package_name: C++ include headers
- msg/: Folder containing Message (msg) types
- src/package_name/: Source files (C++ / Python)
- srv/: Folder containing Service (srv) types
- scripts/: executable scripts
- launch/: contain launch files to execute multiple nodes at a time
- urdf/: contains xacro and urdf files of robot model
- CMakeLists.txt: CMake build file
- package.xml: Package catkin/package.xml

NOTE: detail description of package development is out of scope of this document. Readers are requested to check ROS wiki.

2.3. vision_opencv

refer section 1.1

2.4. image_transport

refer section 1.1

3. Concept development and Implementation

In this project, ROS package **image_processing_with_opencv_and_ros** is created which contains the node **cameraImageAquisitionAndProcessingNode**.

This node subscribes the image from usb_cam, process it and then publish it as ROS topic.

The process flow is shown below in Fig 1.

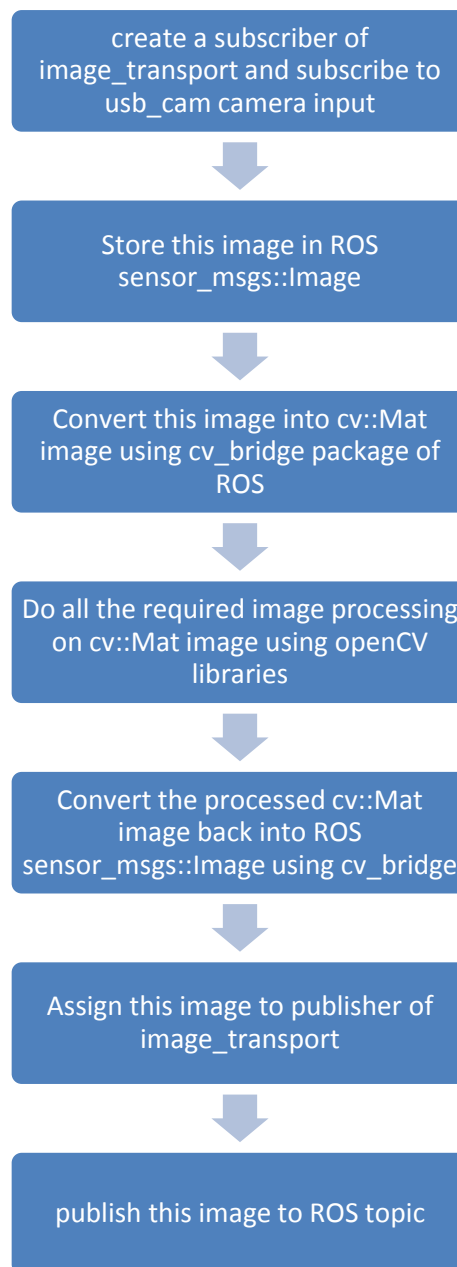


Figure 1 - Process Flow Chart

As in the above process explained, the subscriber of the `image_transport` reads image data from topic published by `usb_cam` node, it is necessary to first start the `usb_cam` node to publish real time camera images from webcam.

NOTE: As the main aim of the project is to learn and understand the process of image conversions between opencv and ROS, the image processing part is kept very minimum. Hence for this project, only the image brightness is increased and then image is published. Similarly, instead of brightness, other image processing can also be done easily with the same approach.

As opencv 3 is used for the project with ROS Kinetic, it is important to provide the path for include and libs of opencv in `CMakeLists.txt`. The Fig 2 below shows the code (as image) [7].

```

1  cmake_minimum_required(VERSION 2.8.3)
2  project(image_processing_with_opencv_and_ros)
3
4  find_package(catkin REQUIRED COMPONENTS
5      cv_bridge
6      image_transport
7      roscpp
8      sensor_msgs
9      std_msgs
10 )
11 find_package(OpenCV REQUIRED)
12
13 catkin_package(
14     # INCLUDE_DIRS include
15     # LIBRARIES image_processing_with_opencv_and_ros
16     # CATKIN_DEPENDS cv_bridge image_transport roscpp sensor_msgs std_msgs
17     # DEPENDS system_lib
18 )
19 include_directories(
20     # include
21     ${catkin_INCLUDE_DIRS}
22     ${OpenCV_INCLUDE_DIRS}
23 )
24
25 add_executable(cameraImageAquisitionAndProcessingNode src/cameraImageAquisitionAndProcessingNode.cpp)
26 target_link_libraries(cameraImageAquisitionAndProcessingNode ${catkin_LIBRARIES} ${OpenCV_LIBRARIES})

```

Figure 2 - `CMakeLists.txt`

Lines added to link with OpenCV are:

1. Line 11: `find_package(OpenCV REQUIRED)`
2. Line 22: `${OpenCV_INCLUDE_DIRS}`
3. Line 26: `${OpenCV_LIBRARIES}`
 - Add this for every new node

There are no changes required in `package.xml` file of the package.

Complete ROS package is available in **src** folder of the project repository and this report is available in the **doc** folder of the project repository.

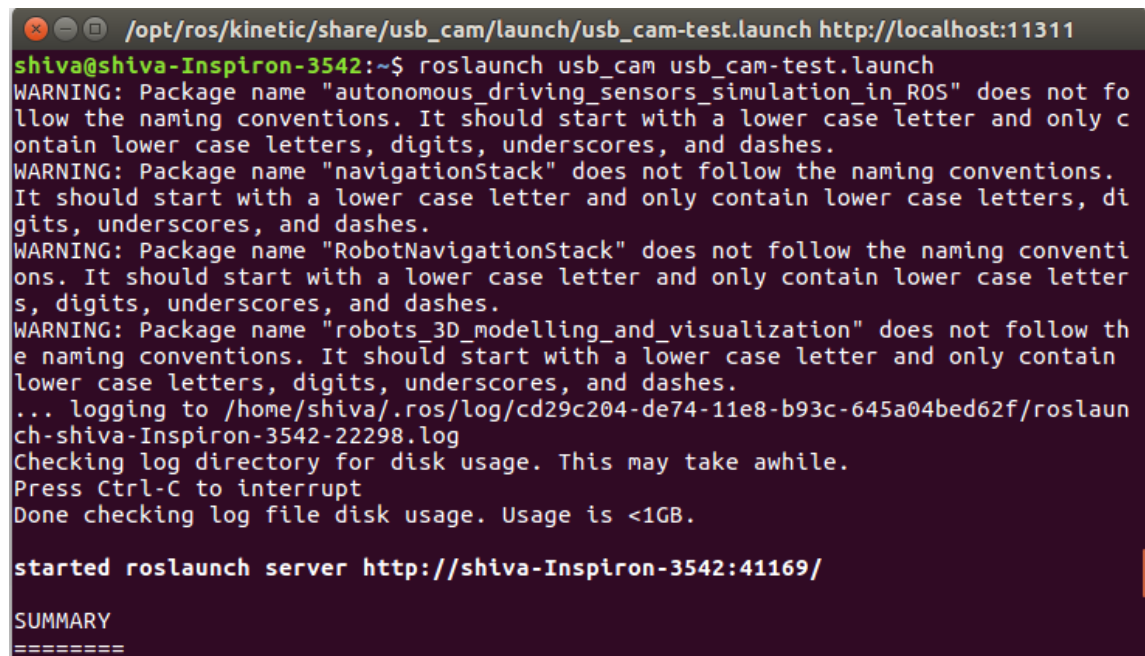
4. Test and results

1. Check the webcam of the laptop for its functionality.
2. Install usb_cam using terminal

```
$ sudo apt-get install ros-kinetic-usb_cam
```

3. Start the usb_cam

```
$ roslaunch usb_cam usb_cam-test.launch
```



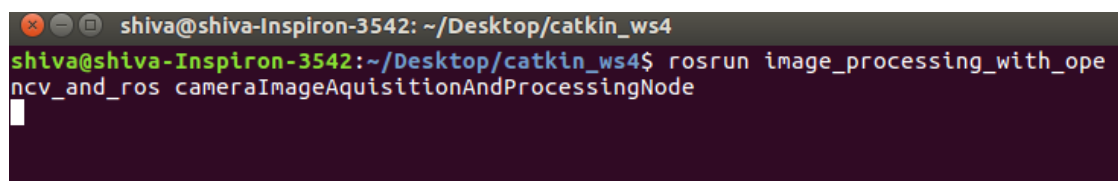
```
/opt/ros/kinetic/share/usb_cam/launch/usb_cam-test.launch http://localhost:11311
shiva@shiva-Inspiron-3542:~$ roslaunch usb_cam usb_cam-test.launch
WARNING: Package name "autonomous_driving_sensors_simulation_in_ROS" does not follow the naming conventions. It should start with a lower case letter and only contain lower case letters, digits, underscores, and dashes.
WARNING: Package name "navigationStack" does not follow the naming conventions. It should start with a lower case letter and only contain lower case letters, digits, underscores, and dashes.
WARNING: Package name "RobotNavigationStack" does not follow the naming conventions. It should start with a lower case letter and only contain lower case letters, digits, underscores, and dashes.
WARNING: Package name "robots_3D_modelling_and_visualization" does not follow the naming conventions. It should start with a lower case letter and only contain lower case letters, digits, underscores, and dashes.
... logging to /home/shiva/.ros/log/cd29c204-de74-11e8-b93c-645a04bed62f/roslaunch-shiva-Inspiron-3542-22298.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://shiva-Inspiron-3542:41169/

SUMMARY
=====
```

Figure 3- usb_cam

4. Start the node created in this project
 - a. **Package name:** image_processing_with_opencv_and_ros
 - b. **Node name:** cameraImageAquisitionAndProcessingNode



```
shiva@shiva-Inspiron-3542: ~/Desktop/catkin_ws4
shiva@shiva-Inspiron-3542:~/Desktop/catkin_ws4$ rosrn image_processing_with_opencv_and_ros cameraImageAquisitionAndProcessingNode
```

Figure 4 – cameraImageAquisitionAndProcessingNode

Fig 5 (left) shows the original image acquired by webcam and Fig 5 (right) shows the processed image (brightness increased) by the node developed in this project.



Figure 5 - Raw Image (left) and Processed image (right)

All the above screenshots are also available in the **test** folder of project repository.

5. Conclusion and Future Scope

The image processing node is developed, implemented and tested successfully. The same node is then planned to use in further projects where more than 1 camera will be installed on RC car with raspberry pi and images from each camera will be published separately and this node will be used to do image processing to implement self-driving car functions.

6. References

- [1] "usb_cam," Open sourceRobotics Foundation, [Online]. Available: http://wiki.ros.org/usb_cam.
- [2] "vision_opencv," Open Source Robotics Foundation, [Online]. Available: http://wiki.ros.org/vision_opencv.
- [3] "cv_bridge," Open Source Robotics Foundation, [Online]. Available: http://wiki.ros.org/cv_bridge.
- [4] "image_transport," Open source robotics foundation, [Online]. Available: http://wiki.ros.org/image_transport.
- [5] "ROS wiki," open source Robotics Foundation , [Online]. Available: <http://www.ros.org/>.
- [6] "packages," ROS open source, [Online]. Available: <http://wiki.ros.org/Packages>.
- [7] L. Joseph, Mastering ROS for Robotics Programming, Packt.