2020

INFO 802
Master Advanced Mechatronics
Luc Marechal

UNIVERSITÉ SAVOIE MONT BLANC

POLYTECH ANNECY-CHAMBÉRY

ROS

Course 5
Vision - OpenCV

Course 4 : Vision - OpencV

Overview

- Interface camera in ROS
- Acquire images from the camera.
- Subscribe to an image camera topic
- Perform simple image processing with OpenCV

Packages Related to USB Camera

- **ibuvc-camera**: interface package for operating cameras with the UVC standard.
- **uvc-camera**: convenient package with relatively detailed camera settings. (ideal for stereo camera configuration).
- **usb-cam**: very simple camera driver for Bosch.
- **freenect-camera**, **openni-camera**, **openni2-camera**: for depth cameras like Kinect or Xtion. These sensors include a color camera, (also called RGB-D cameras). These packages are required to use their color images.
- **camera1394**: driver for cameras using IEEE 1394 standard FireWire.
- **prosilica-camera**: used in AVT's prosilica camera, (widely used for research purposes).
- **camera-calibration**: camera calibration package that applies OpenCV's calibration feature. Many camera-related packages require this package.

Interfacing USB webcams in ROS

Find the video devices present on the system

```
> ls /dev | grep video
```

Test camera with Cheese.
if not installed:

```
> sudo apt-get install cheese
```

usb_cam driver package installation

```
> sudo apt-get install ros-noetic-usb-cam
```

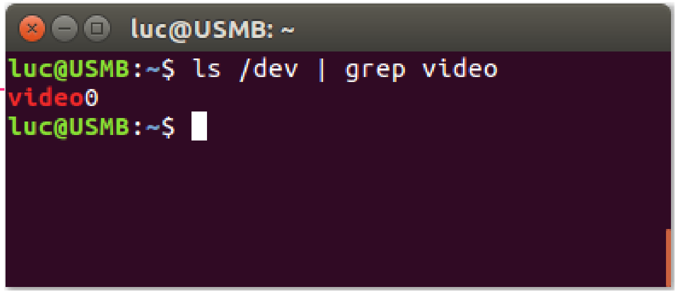
uvc_camera driver package installation

```
> sudo apt-get install ros-noetic-uvc-camera
```

Image Related Package Installation

```
> sudo apt-get install ros-noetic-image-*
> sudo apt-get install ros-noetic-rqt-image-view
```

Found a device numbered 0



```
luc@USMB: ~
luc@USMB:~$ ls /dev | grep video
video0
luc@USMB:~$ 
```

If no video found, click on Virtualbox menu Devices > Webcams > Integrated Webcams

Known issue: https://doc.ubuntu-fr.org/v4l1

Interfacing USB webcams in ROS

Running uvc_camera node

```
> roscore
> rosrun uvc_camera uvc_camera_node
```

Running usb_cam node

```
> roscore
> rosrun usb_cam usb_cam_node
```

Verify Topic Message

```
> rostopic list
```

images are published in multiple ways, compressed and uncompressed (useful to send images to other ROS nodes and occupying little space)

```
/camera_info
/usb_cam/image_raw
/usb_cam/image_raw/compressed
/usb_cam/image_raw/compressed/parameter_descriptions
/usb_cam/image_raw/compressed/parameter_updates
/usb_cam/image_raw/compressedDepth
/usb_cam/image_raw/compressedDepth/parameter_descripti
/usb_cam/image_raw/compressedDepth/parameter_updates
/usb_cam/image_raw/theora
/usb_cam/image_raw/theora/parameter_descriptions
/usb_cam/image_raw/theora/parameter_updates
/rosout
/rosout_agg
```

```
/camera_info
/image_raw
/image_raw/compressed
/image_raw/compressed/parameter_descriptions
/image_raw/compressed/parameter_updates
/image_raw/compressedDepth
/image_raw/compressedDepth/parameter_descriptions
/image_raw/compressedDepth/parameter_updates
/image_raw/theora
/image_raw/theora/parameter_descriptions
/image_raw/theora/parameter_updates
/rosout
/rosout_agg
```

Interfacing USB webcams in ROS

The usb_cam-test.launch file can launch the USB cam driver with the necessary parameters

```
> roslaunch usb_cam usb_cam-test.launch
```

```
<launch>
  <node name="usb_cam" pkg="usb_cam" type="usb_cam_node"
output="screen" >
    <param name="video_device" value="/dev/video0" />
    <param name="image_width" value="640" />
    <param name="image_height" value="480" />
    <param name="pixel_format" value="yuyv" />
    <param name="camera_frame_id" value="usb_cam" />
    <param name="io_method" value="mmap"/>
  </node>
  <node name="image_view" pkg="image_view" type="image_view"
respawn="false" output="screen">
    <remap from="image" to="/usb_cam/image_raw"/>
    <param name="autosize" value="true" />
  </node>
</launch>
```

you will get this warning message about the camera calibration, '[WARN] [1423194481.257752159]: Camera calibration file /home/xxx/.ros/camera_info/camera. yaml not found.'

because the calibration file is missing. You can ignore this for now.

Interfacing USB webcams in ROS

Visualize the image in another window with image view node

```
> rosrun image_view image_view image:=/usb_cam/image_raw
```

Republish the image in an other format on an other topic

```
> rosrun image_transport republish compressed in:=/usb_cam/image_raw
[output format] out:=/usb_cam/image_raw/republished
```

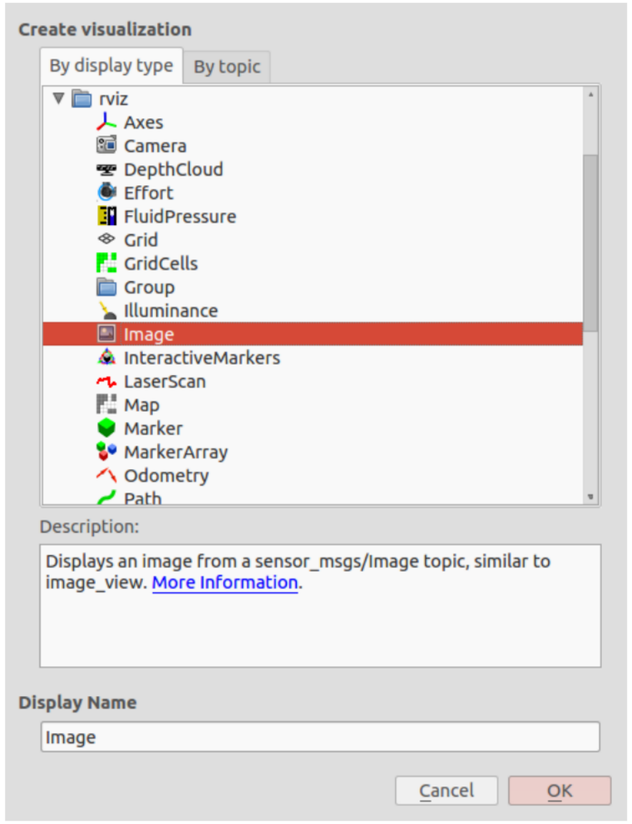
Visualize with rqt_image_view Node

```
> rosrun rqt_image_view rqt_image_view image:=/usb_cam/image_raw
```

Interfacing USB webcams in ROS

Visualize with rviz

```
> rqt_image_view image:=/usb_cam/image_raw
```



Create visualization

By display type | By topic

- rviz
  - Axes
  - Camera
  - DepthCloud
  - Effort
  - FluidPressure
  - Grid
  - GridCells
  - Group
  - Illuminance
  - Image
  - InteractiveMarkers
  - LaserScan
  - Map
  - Marker
  - MarkerArray
  - Odometry
  - Path

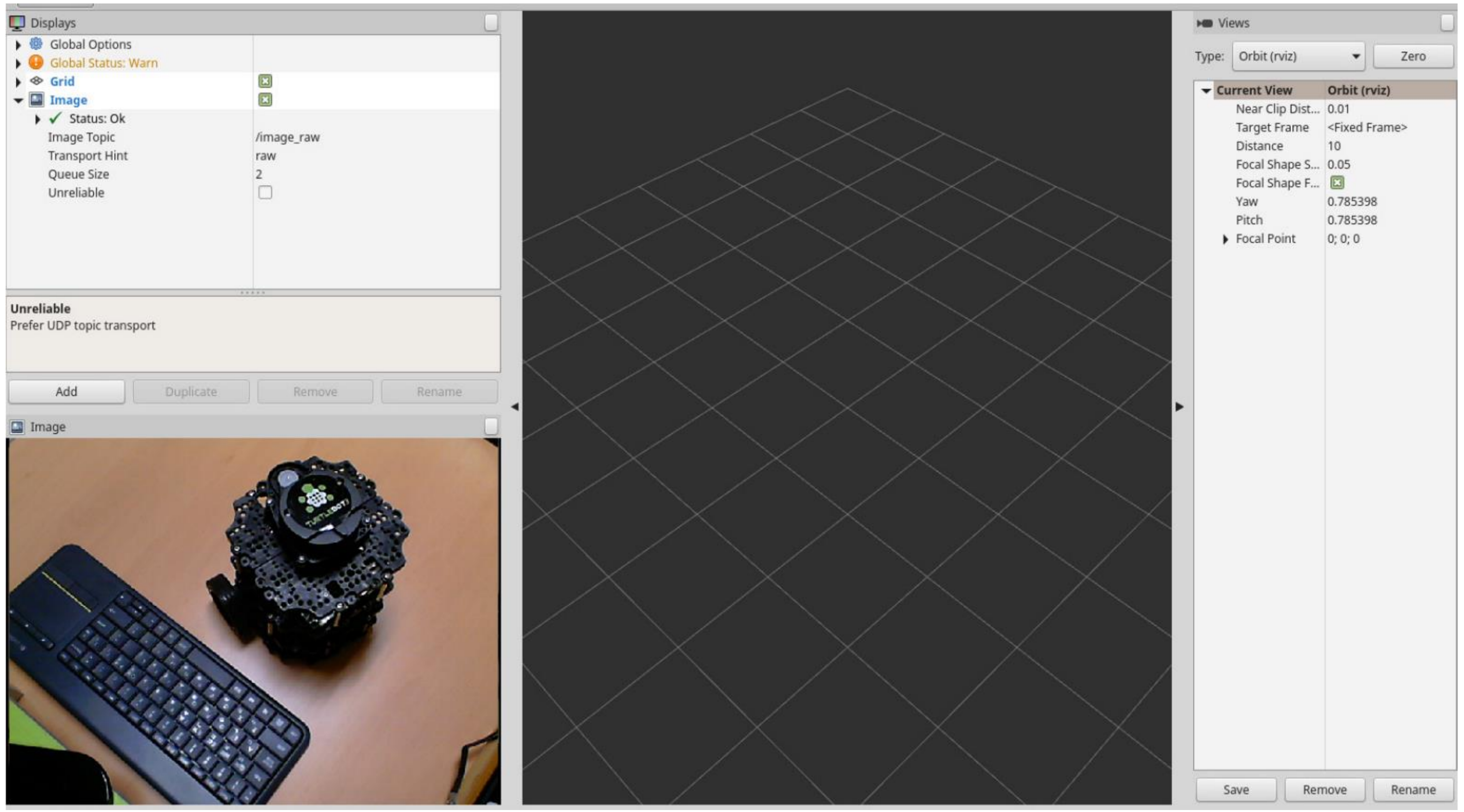
Description:

Displays an image from a sensor_msgs/Image topic, similar to image_view. More Information.

Display Name

Image

Cancel | OK



Displays

Global Options	
Global Status: Warn	
Grid	✓
Image	✓
Status: Ok	
Image Topic	/image_raw
Transport Hint	raw
Queue Size	2
Unreliable	☐

Unreliable
Prefer UDP topic transport

Add | Duplicate | Remove | Rename

Image

Views

Type: Orbit (rviz) | Zero

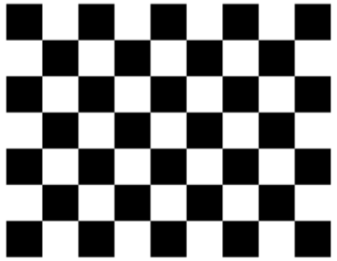
Current View	Orbit (rviz)
Near Clip Dist...	0.01
Target Frame	<Fixed Frame>
Distance	10
Focal Shape S...	0.05
Focal Shape F...	✓
Yaw	0.785398
Pitch	0.785398
Focal Point	0; 0; 0

Save | Remove | Rename

Change the value of [Image Topic] in the [Image] option to ‘/usb_cam/image_raw’

ROS camera calibration

You need a large checkerboard with known dimensions. Calibration uses the interior vertex points of the checkerboard



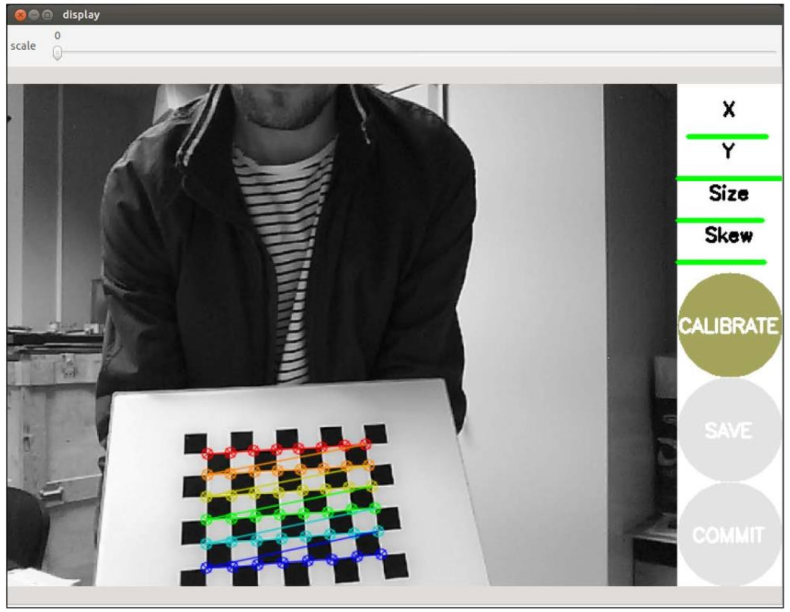
http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration?action=AttachFile&do=view&target=check-108.pdf

Getting the dependencies and compiling the driver

```
> rosdep install camera_calibration
```

Running the Calibration Node

```
> rosrun camera_calibration cameracheck.py --size 8x6
monocular:=/forearm image:=image_rect
```



More Info
http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration

OpenCV

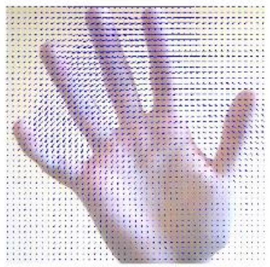
- OpenCV is a library of programming functions mainly aimed at real-time computer vision.
- Stands for the Open Source Computer Vision Library
- Developed by Intel in 1999.
- Supported by Willow Garage.
- Cross Platform
- Free for use under ‘’Open-Source BSD License’’.

OpenCV Algorithm Modules Overview

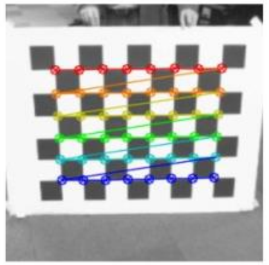
Image Processing

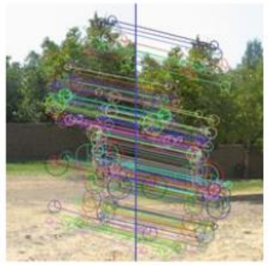
Transforms

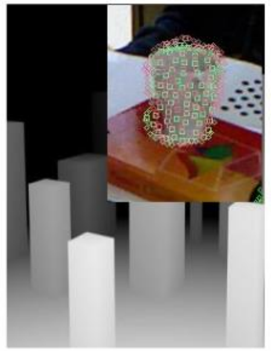
Fitting

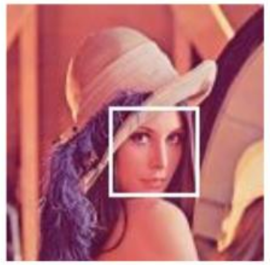

Optical Flow
Tracking

Segmentation


Calibration


Features
VSLAM


Depth, Pose
Normals, Planes,
3D Features


Object recognition
Machine learning

Computational
Photography

Content source
G. Bradsky
Willow garage

OpenCV Image Processing

Colors

- **BGR**: The default color. Normal 3 channel color
- **HSV**: Hue is color, Saturation is amount, Value is lightness. 3 channels
- **GRAYSCALE**: Gray values, Single channel

OpenCV requires that images to be in BGR or Grayscale in order to be shown

OpenCV Image Processing

Colors

In OpenCv for HSV:

Hue range is [0 , 179]

Saturation range is [0 , 255]

Value range is [0 , 255]

Find HSV values to track

```
> green = np.uint8([[[0,255,0 ]]])
> hsv_green = cv.cvtColor(green,cv.COLOR_BGR2HSV)
> print(hsv_green)
[[[ 60 255 255]]]
```

take [H-10, 100,100] and [H+10, 255, 255] as lower bound and upper bound respectively

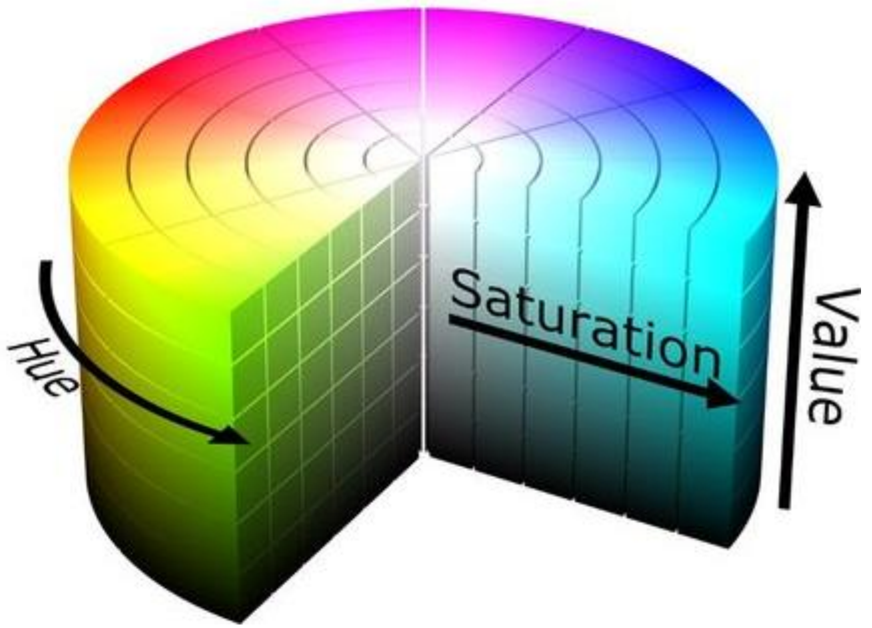


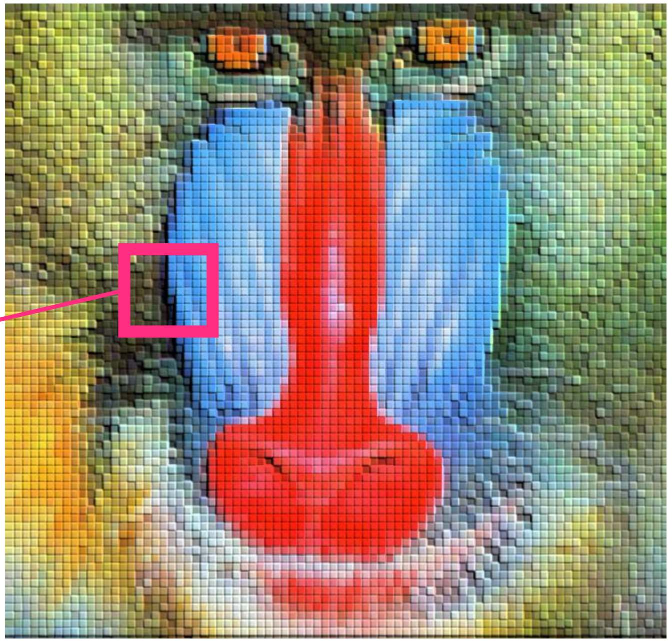
Image Processing

What is an Image?

- 2D array of pixels
- Binary image (bitmap)
  - Pixels are bits
- Grayscale image
  - Pixels are scalars
  - Typically 8 bits (0..255)
- Color images
  - Pixels are vectors
  - Order can vary: RGB, BGR
  - Sometimes includes Alpha

but the camera sees this

```
186 203 127 127 139
185 206 136 136 150
255  30 101 133 170
256 200 139 138  40
257 179 122 179  30
129 165 123 167  36
```



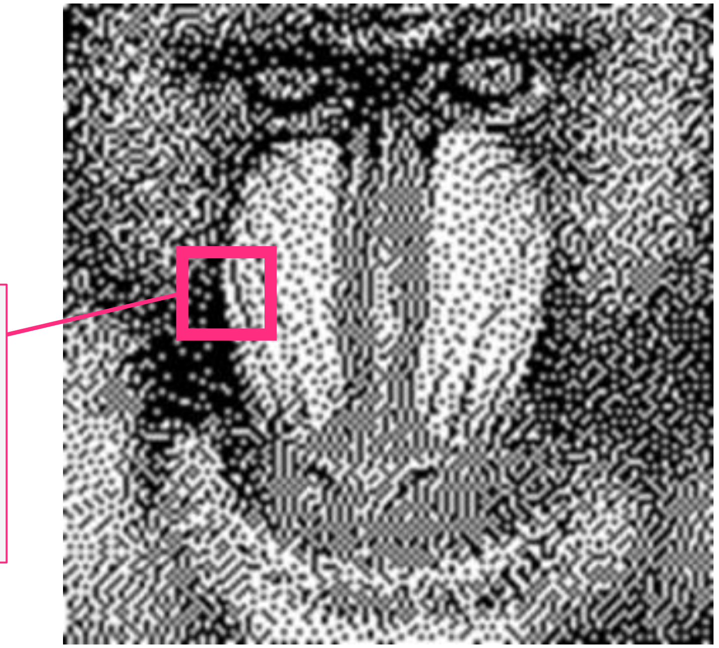
Content source
Rahul Sukthankar - The Robotics Institute, Carnegie Mellon

Image Processing

What is an Image?

- 2D array of pixels
- Binary image (bitmap)
  - Pixels are bits (black=0, white=1)
- Grayscale image
  - Pixels are scalars
  - Typically 8 bits (0..255)
- Color images
  - Pixels are vectors
  - Order can vary: RGB, BGR
  - Sometimes includes Alpha

```
1 1 0 0 1 1 0 1
1 1 1 0 1 0 1 1
1 1 1 1 0 0 1 1
1 1 1 1 0 0 0 1
1 1 1 1 1 0 0 0
```



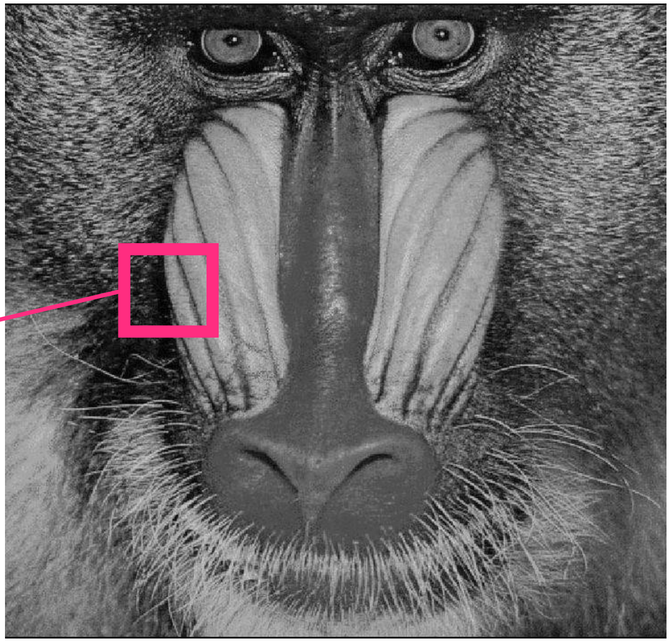
Content source
Rahul Sukthankar - The Robotics Institute, Carnegie Mellon

Image Processing

What is an Image?

- 2D array of pixels
- Binary image (bitmap)
  - Pixels are bits
- Grayscale image
  - Pixels are scalars
  - Typically 8 bits (0..255)
- Color images
  - Pixels are vectors
  - Order can vary: RGB, BGR
  - Sometimes includes Alpha

186	203	127	127	139
185	206	136	136	150
255	30	101	133	170
256	200	139	138	40
257	179	122	179	30
129	165	123	167	36

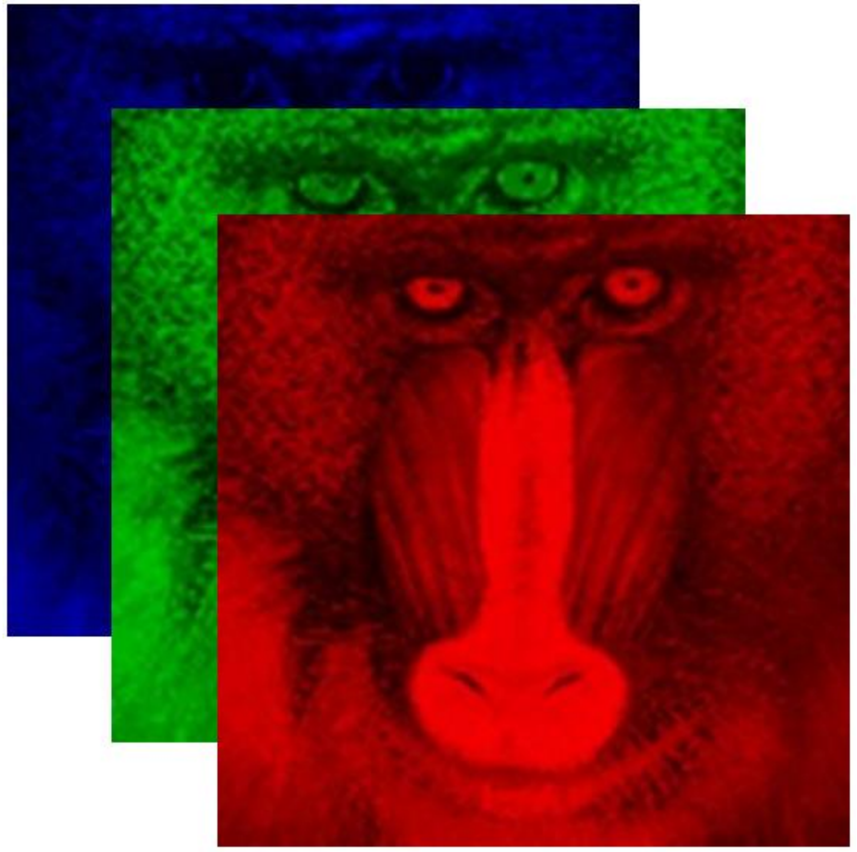


Content source
Rahul Sukthankar - The Robotics Institute, Carnegie Mellon

Image Processing

What is an Image?

- 2D array of pixels
- Binary image (bitmap)
  - Pixels are bits
- Grayscale image
  - Pixels are scalars
  - Typically 8 bits (0..255)
- Color images
  - Pixels are vectors
  - Order can vary: RGB, BGR !!!
  - Sometimes includes Alpha

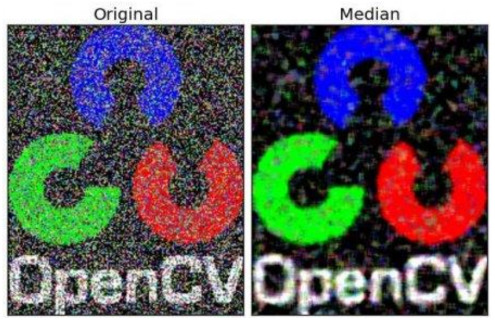


Content source
Rahul Sukthankar - The Robotics Institute, Carnegie Mellon

OpenCV Image Processing

Basic Computer Vision Techniques

- Filtering
  - Remove noise from image
- Segmentation
  - Partition image into groups of pixels
  - Similarity can be decided based on intensity, color, pattern
- Feature detection
  - Extract interesting parts from the image
  - Edge detection, corner detection
  - For example in navigation systems it may prove useful to extract only floor lines from an image



Content source
Roi Yehoshua
- Bar Ilan University

OpenCV in ROS

Step-by-step procedures:

1. Subscribe the images from the camera driver from the topic /usb_cam/image_raw
   (or different topic depending on the driver used)

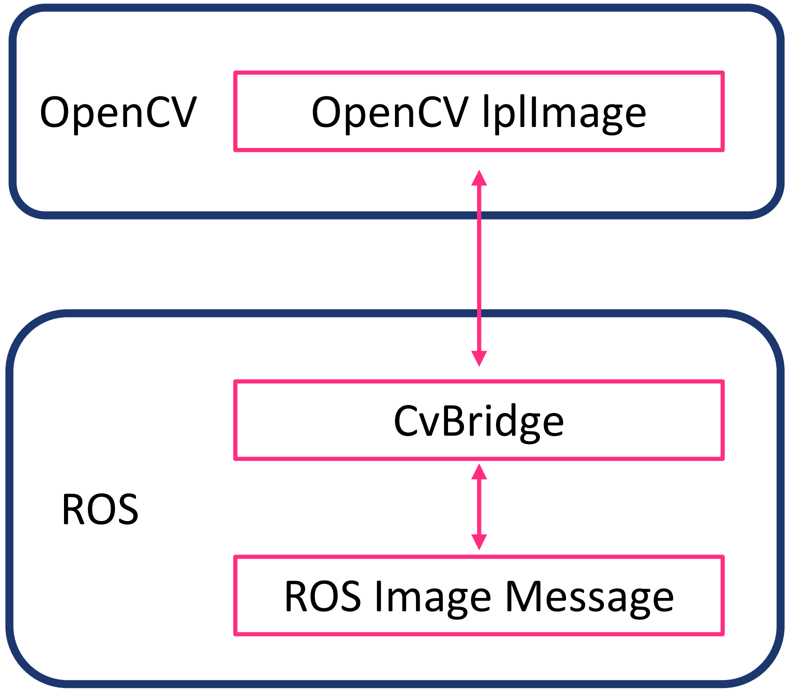
2. Convert the ROS images to the OpenCV image type using CvBridge

3. Process the OpenCV image using its APIs and find the edges on the image

4. Convert the OpenCV image type of the edge detection to the ROS image messages and publish into the topic /edge_detector/processed_image

OpenCV in ROS

- OpenCV uses image native format from taken from the Intel Image Processing Library (IplImage)
- ROS passes images in its own *sensor_msgs/Image* message
- *cv_bridge* is a ROS package that provides functions to convert between ROS *sensor_msgs/Image* messages and the objects used by OpenCV



OpenCV in ROS

Installation

```
> sudo apt-get install libopencv-dev
```

Alternatively

```
> sudo apt-get install ros-kinetic-opencv3
```

Creating a Simple Image Subscriber Node (Python)

Writing the Node

Create package

```
> cd ~/catkin_ws/src/
> catkin_create_pkg vision_tutorials rospy roscpp
cv_bridge
```

Edit script

```
> cd ~/catkin_ws/src/vision_tutorials
> mkdir scripts
> gedit simple_camera_view_node.py
```

Make script executable

```
> cd ~/catkin_ws/src/script
> chmod +x simple_camera_view_node.py
```

Make package and source environment

```
> cd ~/catkin_ws
> catkin_make
> source ./devel/setup.bash
```

```python
#! /usr/bin/python3

import rospy
from sensor_msgs.msg import Image
from cv_bridge import CvBridge
import cv2

bridge = CvBridge()

def image_callback(msg):
    cv2_img = bridge.imgmsg_to_cv2(msg, "bgr8")
    cv2.imshow("Usb camera video", cv2_img)
    cv2.waitKey(3)

def simple_camera_view_node():
    rospy.init_node('image_subscriber')
    image_topic = "/usb_cam/image_raw"

    rospy.Subscriber(image_topic, Image, image_callback)
    rospy.spin()

if __name__ == '__main__':
    simple_camera_view_node()
```

https://github.com/LucMarechal/ROS_Lectures/blame/3e8e45a6e527217b7a2f9c787f6523a5ae1d22b6/vision_tutorials/simple_camera_view_node.py

Creating a Simple Image Subscriber Node (Python)

Examining the Node

```python
#! /usr/bin/python3

import rospy
from sensor_msgs.msg import Image
from cv_bridge import CvBridge
import cv2

bridge = CvBridge()

def image_callback(msg):
    cv2_img = bridge.imgmsg_to_cv2(msg, "bgr8")
    cv2.imshow("Usb camera video", cv2_img)
    cv2.waitKey(3)

def simple_camera_view_node():
    rospy.init_node('image_subscriber')
    image_topic = "/usb_cam/image_raw"

    rospy.Subscriber(image_topic, Image, image_callback)
    rospy.spin()

if __name__ == '__main__':
    simple_camera_view_node()
```

- ROS Image message → `from sensor_msgs.msg import Image`
- ROS Image message -> OpenCV Image converter → `from cv_bridge import CvBridge`
- Instantiates a *cvBridge* object → `bridge = CvBridge()`
- Converts ROS Image message to OpenCV2 → `cv2_img = bridge.imgmsg_to_cv2(msg, "bgr8")`
- Creates a window and display the image → `cv2.imshow("Usb camera video", cv2_img)`
- Introduces a delay of n milliseconds while rendering images to windows → `cv2.waitKey(3)`
- Defines the image topic → `image_topic = "/usb_cam/image_raw"`
- Setup *subscriber* and define its *callback** → `rospy.Subscriber(image_topic, Image, image_callback)`
- Spin until ctrl + C → `rospy.spin()`

***Recall**: *callback* = function passed as an argument to an other function

Creating a Simple Image Subscriber Node (Python)

Running the Node

Launch the webcam driver

```
> roslaunch usb_cam usb_cam-test.launch
```

Run the `simple_camera_view_node` node

```
> rosrun vision_tutorials simple_camera_view_node
```

Creating a Simple Image Processing Node (Python)

Writing the Node

Edit script

```
> cd ~/catkin_ws/src/ vision_tutorials/scripts
> gedit simple_image_processing_node.py
```

Make script executable

```
> cd ~/catkin_ws/src/script
> chmod x+ simple_image_processing_node.py
```

Make package and source environment

```
> cd ~/catkin_ws
> catkin_make
> source ./devel/setup.bash
```

This node detects a specific color in the image and remove the others

https://github.com/LucMarechal/ROS_Lectures/blob/765e41aacd5c6428909f0e17e0e41b728617a580/vision_tutorials/simple_image_processing.py

Creating a Simple Image Processing Node (Python)

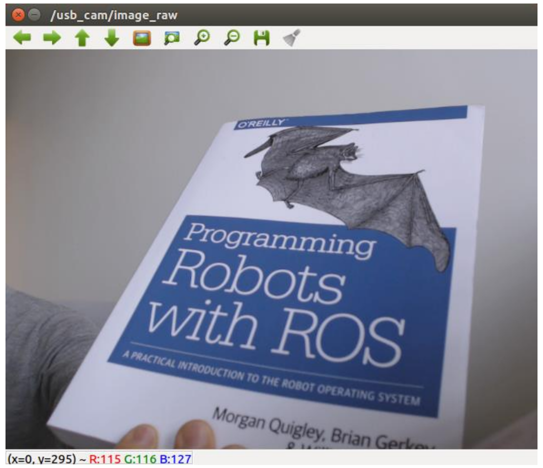
Writing the Node

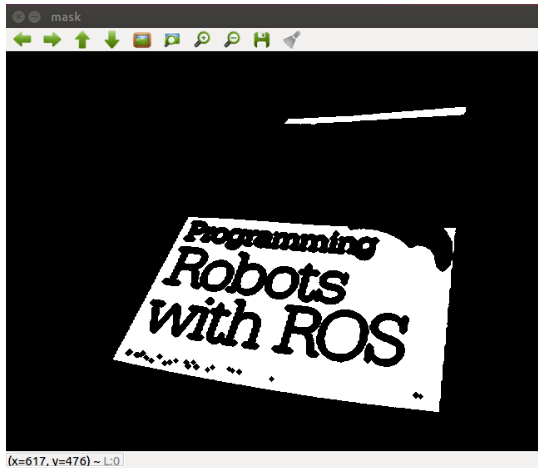
```python
#! /usr/bin/python3

import rospy
from sensor_msgs.msg import Image
from cv_bridge import CvBridge
import cv2

bridge = CvBridge()

def image_callback(msg):
    cv2_img = bridge.imgmsg_to_cv2(msg, "bgr8")
    frame = np.array(cv2_img, dtype=np.uint8)
    color_image = color_detection(frame)
    cv2.waitKey(3)

def color_detection(frame):
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    blue_lower = np.array([90,50,50],np.uint8)
    blue_upper = np.array([120,255,255],np.uint8)
    mask = cv2.inRange(hsv, blue_lower, blue_upper)
    cv2.imshow("image mask",mask)
```

```python
    res = cv2.bitwise_and(frame,frame, mask= mask)
    cv2.imshow("color",res)

    return res


def simple_image_processing():
    rospy.init_node('image_subscriber')
    image_topic = "/usb_cam/image_raw"

    rospy.Subscriber(image_topic, Image, image_callback)
    rospy.spin()



if __name__ == '__main__':
    simple_image_processing()
```

Creating a Simple Image Processing Node (Python)

Examining the Node





Color detection

code snippet

```
def image_callback(msg):
    cv2_img = bridge.imgmsg_to_cv2(msg, "bgr8")
    frame = np.array(cv2_img, dtype=np.uint8)
    color_image = color_detection(frame)
    cv2.waitKey(3)


def color_detection(frame):
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    blue_lower = np.array([90,50,50],np.uint8)
    blue_upper = np.array([120,255,255],np.uint8)
    mask = cv2.inRange(hsv, blue_lower, blue_upper)
    cv2.imshow("image mask",mask)

    res = cv2.bitwise_and(frame,frame, mask= mask)
    cv2.imshow("color",res)

    return res
```

Converts images from BGR to HSV

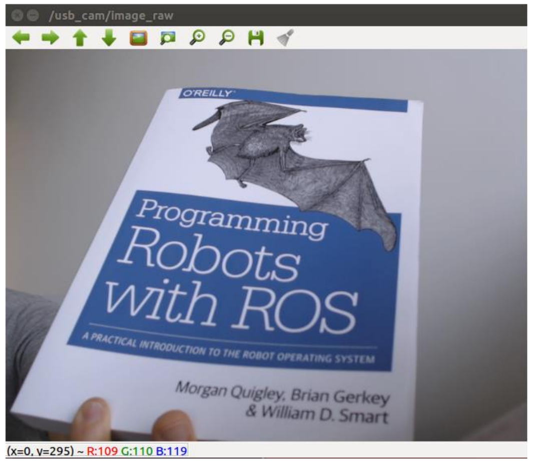
Defining the range of Blue color

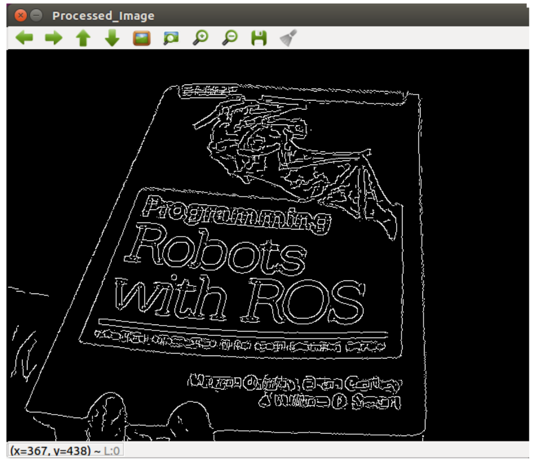
Finding the range blue colour in the image

The bitwise and of the frame and mask is done so that only the blue coloured objects are highlighted and stored in res

Creating a Simple Image Processing Node (Python)

Examining the Node





Edges detection

Converts images from BGR to GREYscale

Computes edges using Canny edges filter

code snippet

```
def image_callback(msg):
    cv2_img = bridge.imgmsg_to_cv2(msg, "bgr8")
    frame = np.array(cv2_img, dtype=np.uint8)
    color_image = edges_detection(frame)
    cv2.waitKey(3)


def edges_detection(frame):
    grey = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    grey = cv2.blur(grey, (7,7))
    edges = cv2,Canny(grey, 15.0, 30.0)

    return edges
```

Assignement

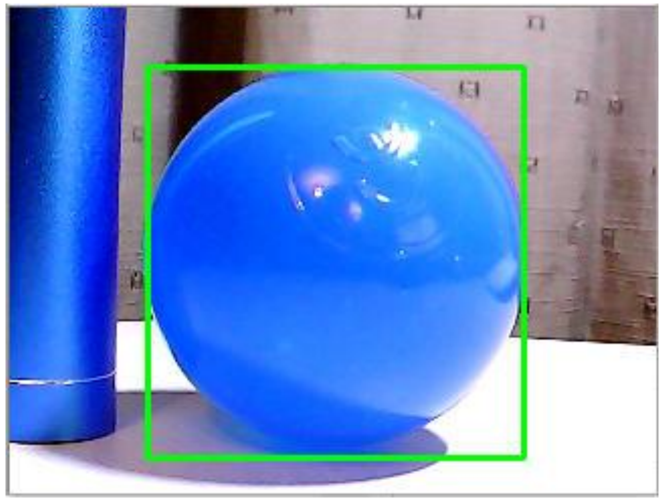
Object Detection and Tracking

Ball tracking

https://www.pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/

http://www.booppey.com/booppey/opencv-ball-detection-color-based/

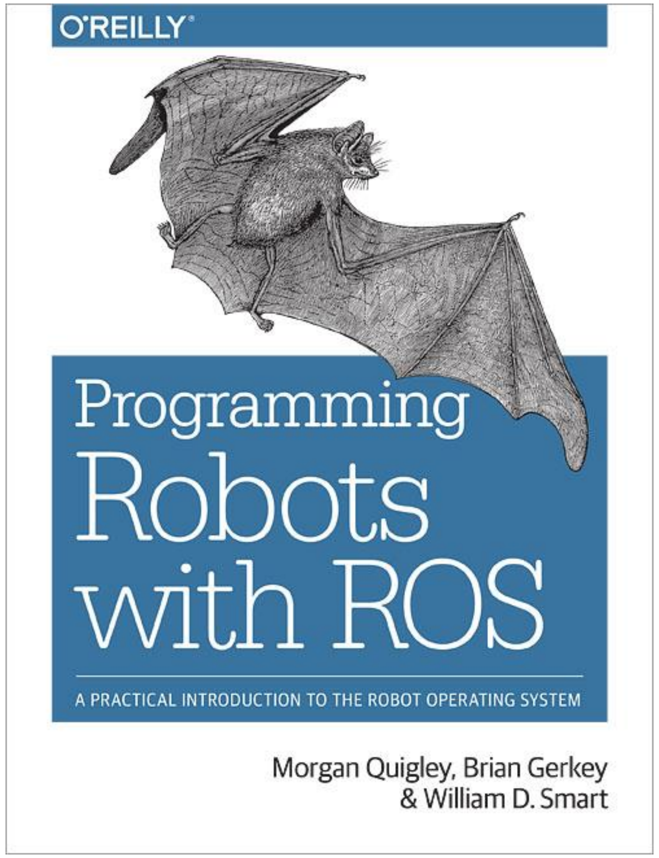
https://docs.opencv.org/master/d2/d96/tutorial_py_table_of_contents_imgproc.html



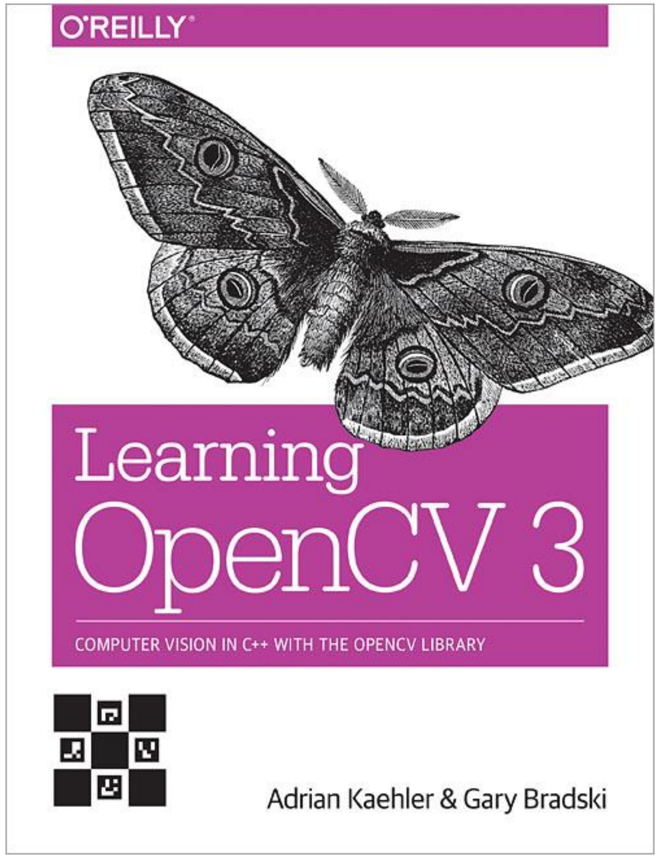
Further References

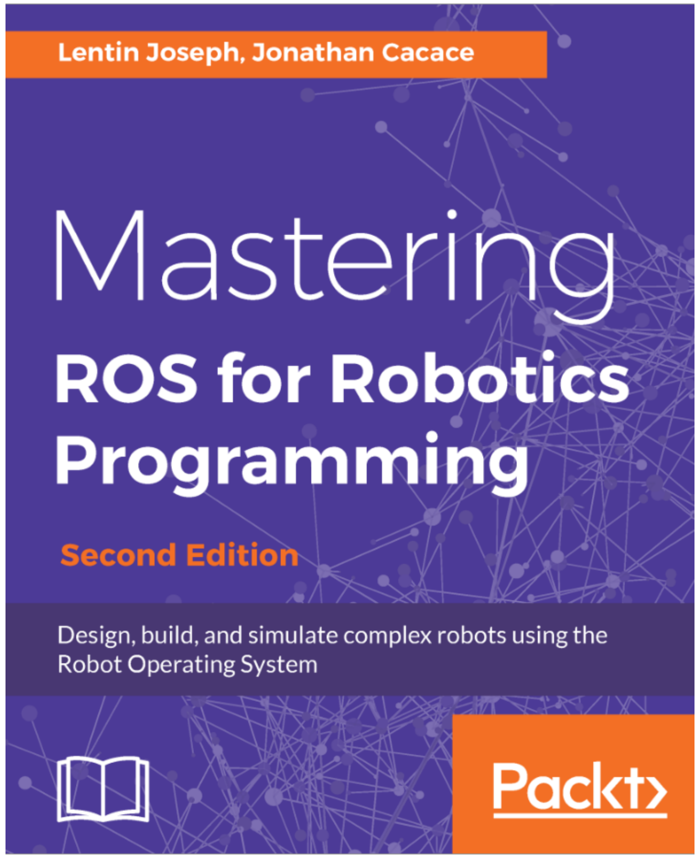
- OpenCV Image processing
  - https://docs.opencv.org/master/d2/d96/tutorial_py_table_of_contents_imgproc.html
- ROS Cheat Sheet
  - https://www.clearpathrobotics.com/ros-robot-operating-system-cheat-sheet/
  - https://kapeli.com/cheat_sheets/ROS.docset/

Relevant books


O'REILLY
Programming Robots with ROS
A PRACTICAL INTRODUCTION TO THE ROBOT OPERATING SYSTEM
Morgan Quigley, Brian Gerkey & William D. Smart

Chapter 12


O'REILLY
Learning OpenCV 3
COMPUTER VISION IN C++ WITH THE OPENCV LIBRARY
Adrian Kaehler & Gary Bradski


Lentin Joseph, Jonathan Cacace
Mastering ROS for Robotics Programming
Second Edition
Design, build, and simulate complex robots using the Robot Operating System
Packt

Chapter 10

Contact Information

University Savoie Mont Blanc

Polytech' Annecy Chambery
Chemin de Bellevue
74940 Annecy
France

https://www.polytech.univ-savoie.fr

Lecturer

Luc Marechal (luc.marechal@univ-smb.fr)
SYMME Lab (Systems and Materials for Mechatronics