Robot Operating System (ROS)

Lab 5: OpenCV in ROS

Haitham El-Hussieny, PhD

November 28, 2022

Department of Mechatronics and Robotics Engineering
Egypt-Japan University of Science and Technology (E-JUST)
Alexandria, Egypt.

Outline

1. OpenCV Overview.
2. Installing OpenCV.
3. Bridging OpenCV and ROS.
4. Laser Range Finder.

OpenCV Overview.

OpenCV Overview.

The most common image processing library is OpencV.

- Open Source Computer Vision Library.
- Free for both academic and commercial use
- C++/Python/Java
- Windows, MacOS, Linux, iOS, Android
- Strong focus on real-time (written in C++ and optimized)

OpenCV Overview: Functionalities

Image Segmentation

Image segmentation is the process of partitioning a digital image into more meaningful and easier to analyze multiple image segments.

(a) Image (b) Semantic Segmentation (c) Instance Segmentation (d) Panoptic Segmentation

OpenCV Overview: Functionalities

Image Thresholding

In digital image processing, thresholding is the simplest method of segmenting images. From a grayscale image, thresholding can be used to create binary images.

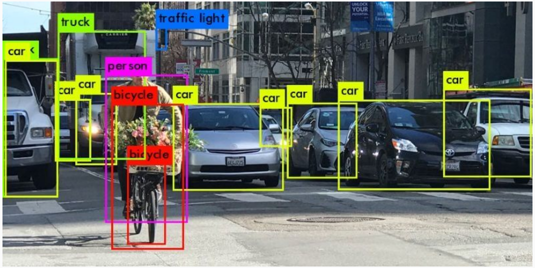
Original image.

The binary image resulting from a thresholding of the original image.

OpenCV Overview: Functionalities

Object Detection and Recognition.

Detection of instances of semantic objects of a certain class (humans, buildings, or cars) in digital images.



OpenCV Overview: Functionalities

Edge Detection.

Edge detection includes a variety of mathematical methods that aim at identifying edges, curves in a digital image at which the image brightness changes sharply or, more formally, has discontinuities.

OpenCV Overview: Functionalities

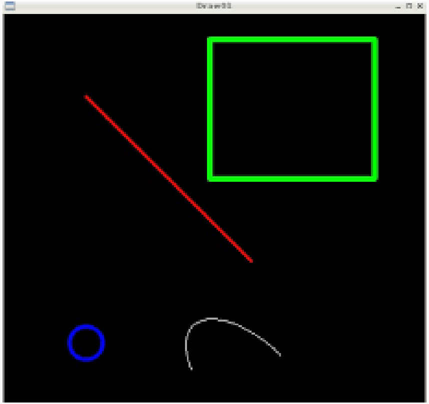
Video/Image Input Output.

OpenCV loads and saves images and videos from the disk or from a USB cams.

OpenCV Overview: Functionalities

Shapes Drawing.

OpenCV can be used to draw basic shapes on images or videos.



Installing OpenCV.

Installing OpenCV.

Before ROS Noetic.

```
$ sudo apt-get update
$ sudo apt-get install ros-melodic-opencv
$ sudo apt-get install ros-melodic-opencv3
```

To verify: see if no errors appeared.

```
$ python
>>> import cv2
>>> exit()
```

Install usb-cam:

```
$ sudo apt-get install ros-melodic-usb-cam
```

Install image-view:

```
$ sudo apt-get install ros-melodic-image-view
```

INSTALLING OPENCV.

On ROS Noetic.

```
$ sudo apt-get update
$ sudo apt-get install ros-noetic-vision-opencv
```

In case you face a problem with **imshow**,

```
$ sudo apt-get install libopencv-*
$ pip install opencv-contrib-python
```

Install usb-cam:

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

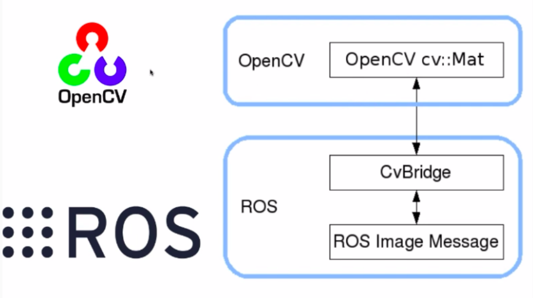
Install image-view:

```
$ sudo apt-get install ros-noetic-image-view
```

Bridging OpenCV and ROS.

Bridging OpenCV and ROS.

- The challenge is the image type supported by ROS is different from the one that is supported by OpenCV.
- CvBridge allows to convert from ROS to OpenCV format and vice versa.



A ROS subscriber to Show Webcam Images.

1. Create a new ROS package:

```
$ cd catkin_ws/src/
$ catkin_create_pkg my_image_processing std_msgs rospy roscpp sensor_msgs
image_transport cv_bridge
```

2. catkin_make in your workspace:

```
~ /catkin_ws$ catkin_make
```

3. create a new python script in the **script** folder:

```
~ /catkin_ws/src/my_image_processing/scripts/$ gedit image_processing.py
```

A ROS subscriber to Show Webcam Images.

4. Write the image subscriber code:

Import important libraries and create a bridge object

```
import rospy
import cv2

from std_msgs.msg import String
from sensor_msgs.msg import Image
from cv_bridge import CvBridge, CvBridgeError

import sys
bridge = CvBridge() #Create a bridge
```

A ROS subscriber to Show Webcam Images.

4. Write the image subscriber code:

main function

```
def main(args):
    rospy.init_node('image_converter', anonymous=True)
    image_sub = rospy.Subscriber("/usb_cam/image_raw", Image, image_callback)

    try:
        rospy.spin()
    except KeyboardInterrupt:
        print("Shutting down")
        cv2.destroyAllWindows()
if __name__ == '__main__':
    main(sys.argv)
```

A ROS subscriber to Show Webcam Images.

4. Write the image subscriber code:

recieving image callback

```
def image_callback(ros_image):
    print ('got an image')
    global bridge

    try :
        cv_image = bridge.imgmsg_to_cv2(ros_image, 'bgr8')
    except CvBridgeError as e:
        print (e)
    #from now you can work with opencv commands:
    cv2.imshow("Image_window", cv_image)
    cv2.waitKey(3)
```

A ROS subscriber to Show Webcam Images.

5. Make sure the libraries are define im CMakeLists.txt:

```
find_package(catkin REQUIRED COMPONENTS
  roscpp
  rospy
  std_msgs
  image_transport
  cv_bridge
  sensor_msgs
)
```

6. change the python file to executable:

```
$ chmod a+x scripts/image_processing.py
```

A ROS subscriber to Show Webcam Images.

7. catkin_make your ROS workspace.

```
$ catkin_make
```

8. Run the image_processing.py node:

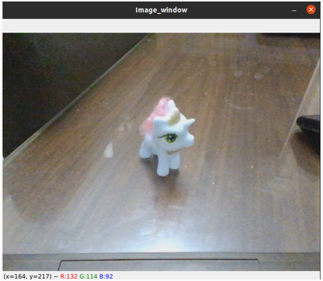
```
$ roscore
$ rosrun my_image_processing image_processing.py
```

9. Run the usb_cam node:

```
$ rosrun usb_cam usb_cam_node _pixel_format:=yuyv
```

10. Verify the image topic is published by **rostopic list**.

```
$ rostopic list
```



Adding a Publisher to Show Image with Edges.

```
def main(args):
    rospy.init_node('image_converter', anonymous=True)

    image_sub = rospy.Subscriber("/usb_cam/image_raw", Image, image_callback)
    global image_pub
    image_pub = rospy.Publisher("/canny_image",Image)
```

```
#from now you can work with opencv commands:
cv2.imshow("Image_window", cv_image)
cv2.waitKey(3)
edge_image = cv2.Canny(cv_image,100,200)
ros_image = bridge.cv2_to_imgmsg(edge_image)
image_pub.publish(ros_image)
```

Laser Range Finder.

Laser Range Finder.

What is Laser Range Finder?

- Cameras can not measure the distance (depth).
- Laser range finder is based on measuring the **time of flight** between the sent and received laser beam to estimate the distance.
- The laser range finder could be used in:
  - Obstacle avoidance.
  - Building maps (SLAM).
  - Robot navigation.

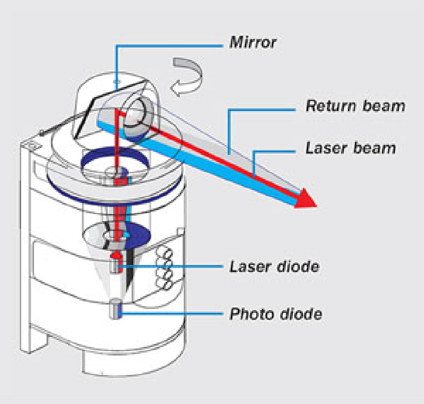
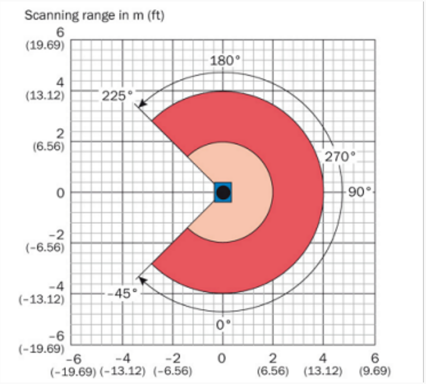


Illustration of LIDAR sensor demonstrating the time of flight principle. *(Courtesy of SICK, Inc.)*

Laser Range Finder.

Characteristics of laser range finder:

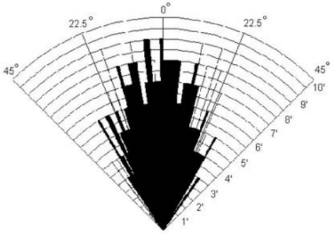
- Minimum and maximum angle.
- Angle increment.
- Time increment.
- Scan time.
- Minimum and maximum range.
- List of ranges.



Laser Range Finder.

Characteristics of laser range finder:

- Minimum and maximum angle.
- Angle increment.
- Time increment.
- Scan time.
- Minimum and maximum range.
- List of ranges.

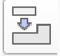


ranges = [1.2, 1.3, 2.0, ...

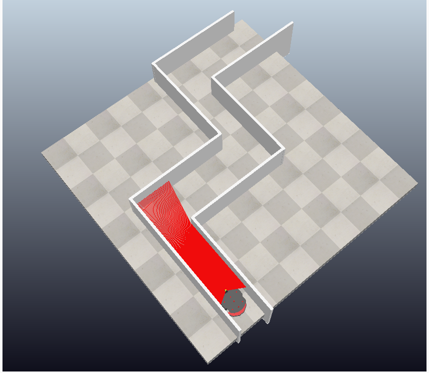
4.2, 2.4, 2.6, ...]

Hokuyo Laser Scanner in CoppeliaSim

- Add a Pioneer P3DX robot in the CoppeliaSim simulation environment.
- Add a Hokuyo URG 04LX (Fast ROS) laser scanner.
- Attach the laser scanner to the robot surface by selecting the laser scanner + CTRL + any red point on the robot surface then click the assemble icon



- Clear any code attached to the Pioneer robot.

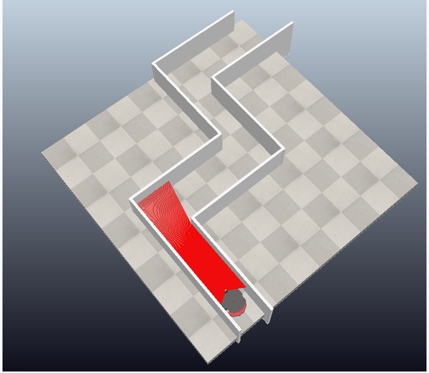


Hokuyo URG 04LX UG01_Fast ROS.ttm

pioneer p3dx.ttm

Hokuyo Laser Scanner in CoppeliaSim

- Try listing the topics and see the `/hokuyo` topic.
- Run **rviz** to visualize the laser scans.



End of Lecture

haitham.elhussieny@ejust.edu.eg