CO-548-A RIS Project

Title: Semantic Segmentation
Jacobs University Bremen, Research I

Spring Semester 2022

May, 20

Authors of the report: Kurmaleyeva Leya,
Orynbekov Arman

Natural Science Laboratory
RIS Module

Experiment conducted by : Kurmaleyeva Leya, Orynbekov Arman

Place of execution : Teaching Lab
Date of execution : May 20, 2022

Introduction

Visualizing environment is the most crucial task for robots. One of the possible advantages is recognizing objects and pathways. In this project we study development of the real-time semantic segmentation algorithm. As a result, we have implemented an algorithm which takes each individual frame from duckie - bot camera, converts it to cv2 type image, applies k-means clustering with a predetermined k, and combines the final images into a video. During the development and testing process it became clear that we had to work with some strict technical limitations. Nevertheless, the project has achieved its initial goals and performs reasonably well.

Related Work

Recently, many works have been investigating semantic segmentation by a driving robot. [1] Identifying objects and regions they cover can be achieved by introducing convolutional neural network (CNN) on mobile robots [3] , agricultural robots [4] with a high prediction accuracy. This multi-step visualization network assigns to each of neurons specific weight and gives an output. The main problems of Convolutional Neural Network is its slow performance, difficult classification under different conditions (lighting, angles) and difference from traditional human vision [5].

Another possible way to achieve semantic segmentation is using K-Means from Machine Learning Algorithm. It clusters the visualization into discrete regions and
Studies have shown that K-Means is specifically efficient for simple problems [6].Instead of applying Convolutional matrix, we enhance the quality of a picture and add weights.

Problem Formulation

The task of our Experiment is to investigate application of semantic segmentation on a robot. The robot, that we used is DB - 19. It was calibrated with regards to a camera and keyboard accordingly. Robot has to be rules by the keyboard and process the environment by a camera on a display of the system. Applying Semantic Segmentation divides visualized environment into discrete areas, as stated earlier. Segmented picture is published on a display as well. The amount of segmented discrete areas depends on the number of centers from algorithm. In our experiment, we stick to 3, 5 and 9 regions, highlighted by RGBA - 8 color palette.

Approach

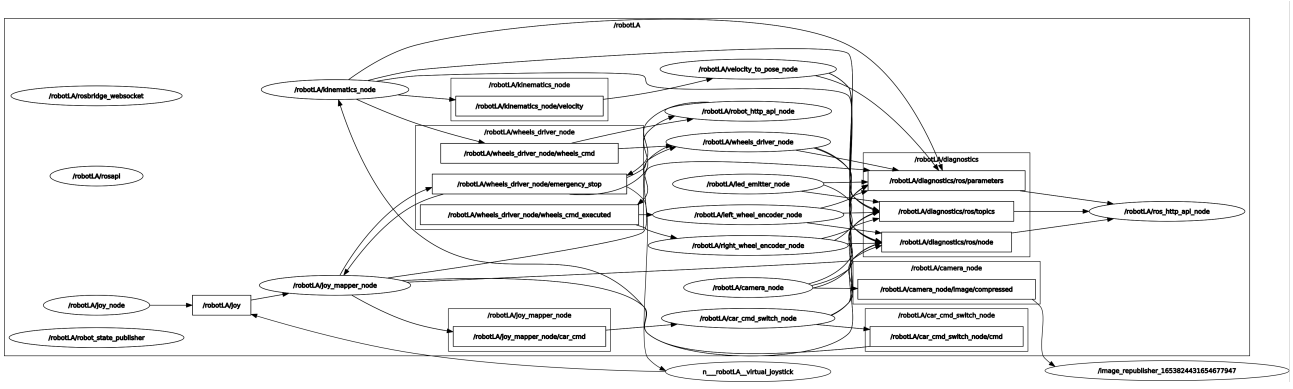
We started by adding libraries rospy, cv2, matplotlib.plt and np.
Rospy library enables connection between user and ROS system on Python. CV2 library is used in order to apply Computer Vision methods and plt library for plotting results.

Open — listener3.py — ~/catkin_ws/src/rosproject/scripts — Save

```python
#!/usr/bin/env python

#Import some important opencv libraries
import rospy
import cv2
import matplotlib as plt
import matplotlib.pyplot as plt
import numpy as np

#Import messages types
from cv_bridge import CvBridge, CvBridgeError
from sensor_msgs.msg import Image
from sensor_msgs.msg import CompressedImage

print("HI") # Intial check

bridge = CvBridge()

# Initialize the ROS Node named 'opencv_example', allow multiple nodes to be run with this name
rospy.init_node('opencv_example', anonymous=True)

# Print "Hello ROS!" to the Terminal and ROSLOG
rospy.loginfo("Hello ROS!") # Check

img_list = [] # Intialize an array where images are going to be stored into

# Function to create an image window
def show_image(img):
     cv2.imshow("Image Window", img)
     cv2.waitKey(3)

# Callback function converts images from img_smg format to cv2 format for later use in Python
def image_callback(img_msg):

    global img_list
    # log some info about the image topic
    rospy.loginfo(img_msg.header)

    try:
         cv_image = bridge.imgmsg_to_cv2(img_msg, "rgba8")
    except CvBridgeError as e:
         rospy.logerr("CvBridge Error: {0}".format(e))

    img_list.append(cv_image) # Add converted image to the array



# Callback function performs K-means clustering on the images from the beforementioned array
def timer_callback(event):
    global img_list
    for count in range(len(img_list)):

        cv2.imwrite(str(count) + 'ci1.jpeg', img_list[count])

        path = str(count) + 'ci1.jpeg'
```

Python — Tab Width: 8 — Ln 81, Col 35 — INS

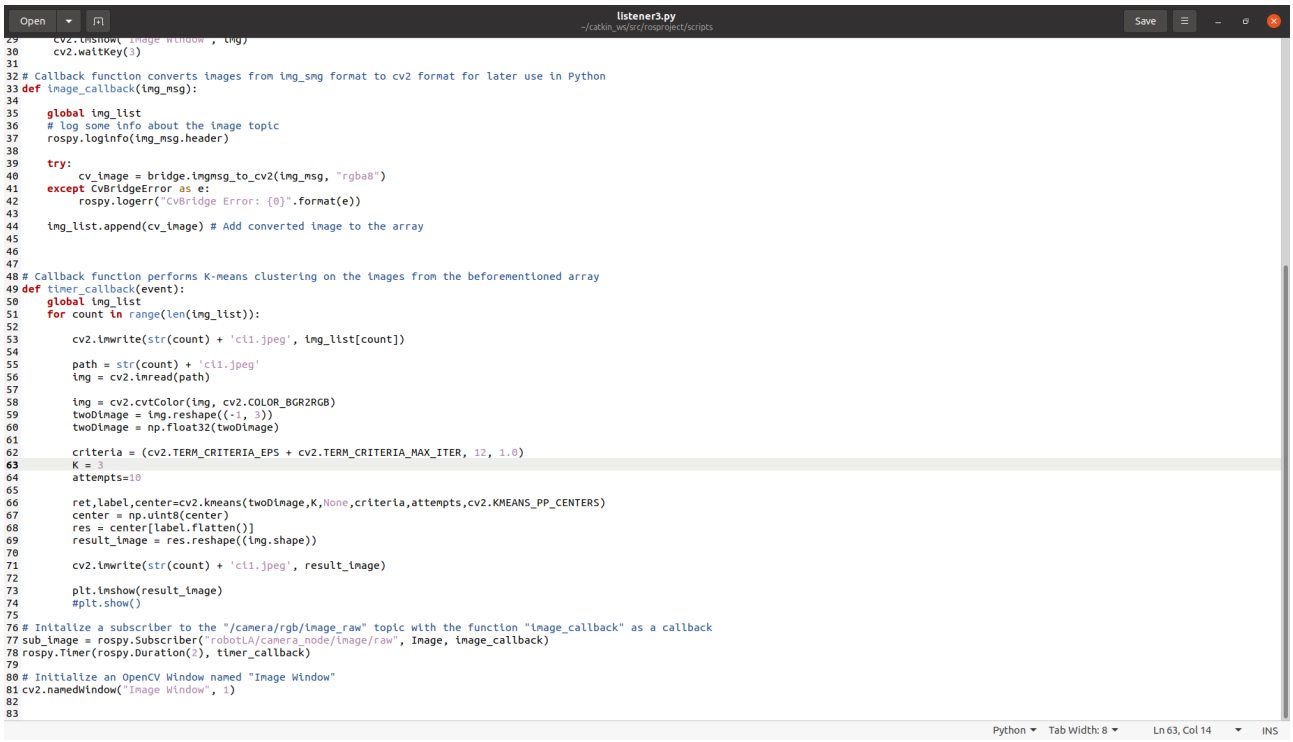
CV2 Library is especially efficient with conversion between our ROS system and it's topics and messages and python environment.



First of all, we initialize the node 'opencv_example', then we create an array, containing the ROS frames, which is expanding until we exit the code. Next, we display unedited frames, that robot shoot on the screen for comparison with further editing.

Secondly, we initialize callback function : image_callback( ), which accepts message with ROS images (frames). We print topic as an output with rospy.loginfo( ) function.
With an internal CV2 function bridge.imgmsg_to_cv2(img_msg, "rgba8") we get conversion from ROS message to CV2 image with rgba - 8 color palette.

In case of error, we throw the message and then add converted image to our array of images.
After converting frames from ROS msg system to Python images, we apply Semantic Segmentation with K - Means Clustering for each image in the array.
K - Means Clustering is an Unsupervised Machine Learning approach, which enables data segmentation. We have K as a number of centers, which allocates objects according to their similarities. So in our case, we will divide image into discrete regions covering objects on that exact image. The bigger is number of centers K, the closer is algorithm to real word. Hence, we can find so - called bottleneck, which compromises simple data segmentation for small values of K and efficiency, for high values of K. It randomly selects the regions according to the specific number K and then applies centroid calculations for higher accuracy.

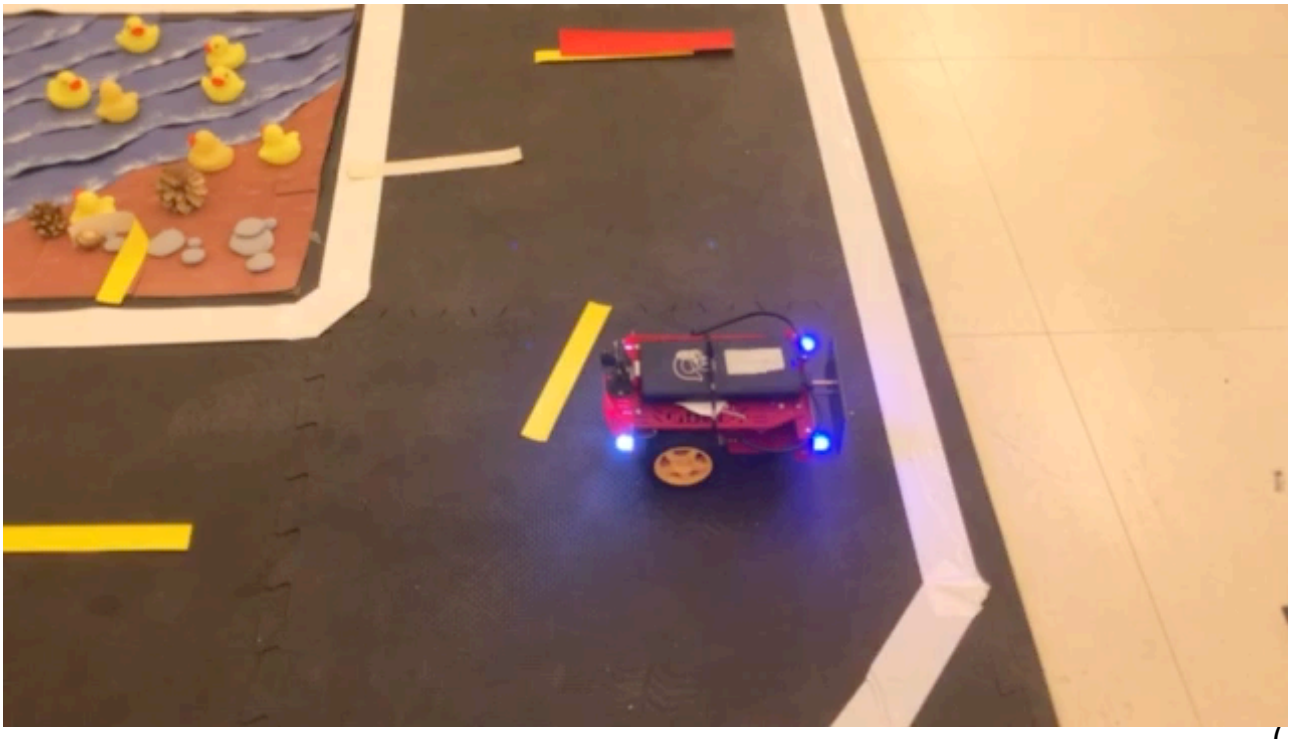


Thirdly, we convert image into RGB float vector . Next, we define criteria, according to which k- means clustering will be performed. In our case, number of iterations = 12 and the accuracy is 1.
Number K, can be altered to achieve different segmentations. Applying before mentioned criteria and arguments we achieve K-means clustering on an image vector and replace pixels with their center values.

As a next step, we initialize subscriber to our ROS raw image topic and call function above: image_callback( ).

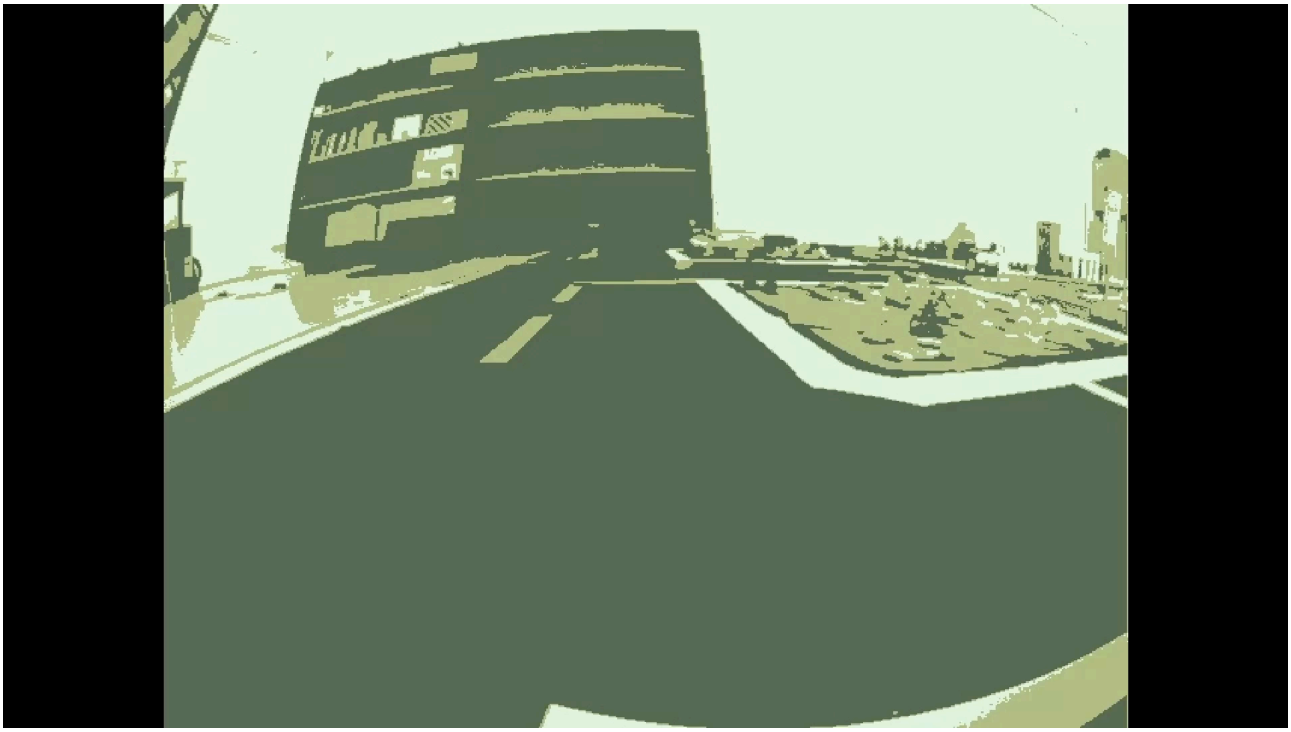
Perspective view from self - driving vehicle (robot), navigated from the keyboard:

Perspective view from

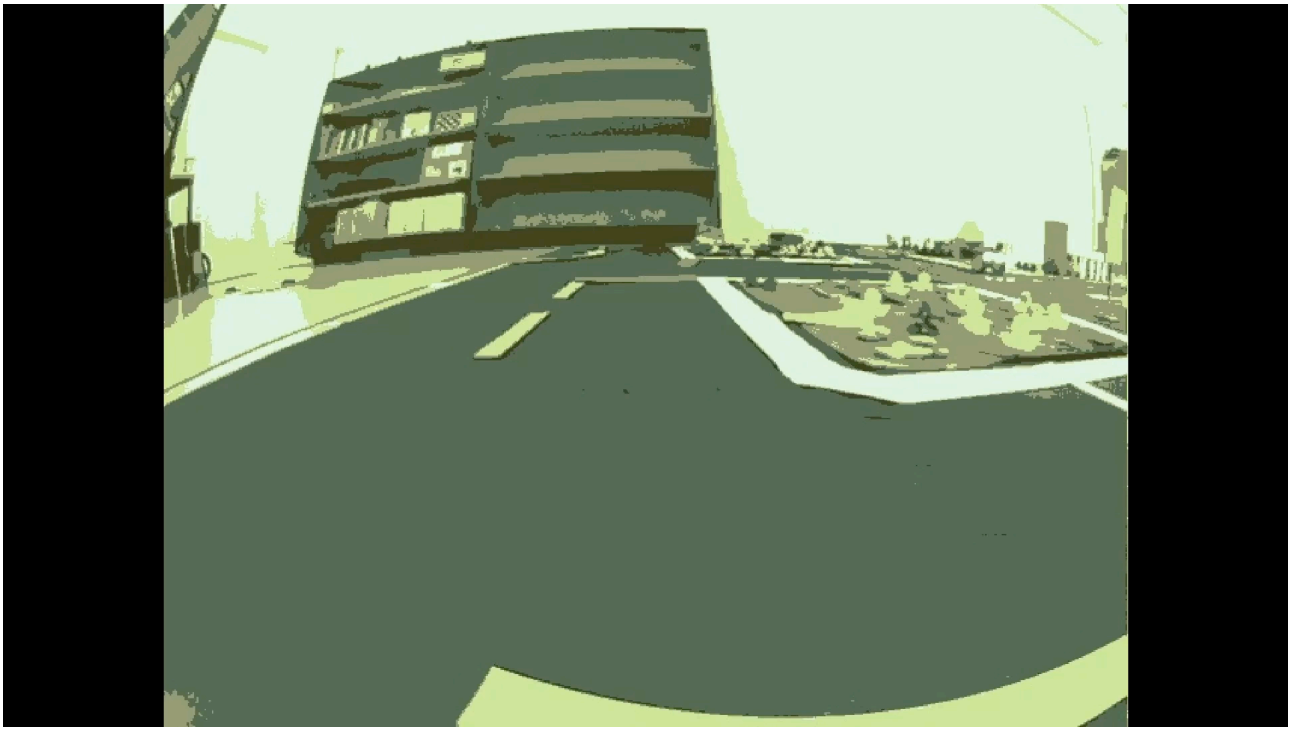


In this experiment we test K-Means Algorithm particularly for number of centers as: K = 3, 5, 9. We will attach the results for each of experiments.

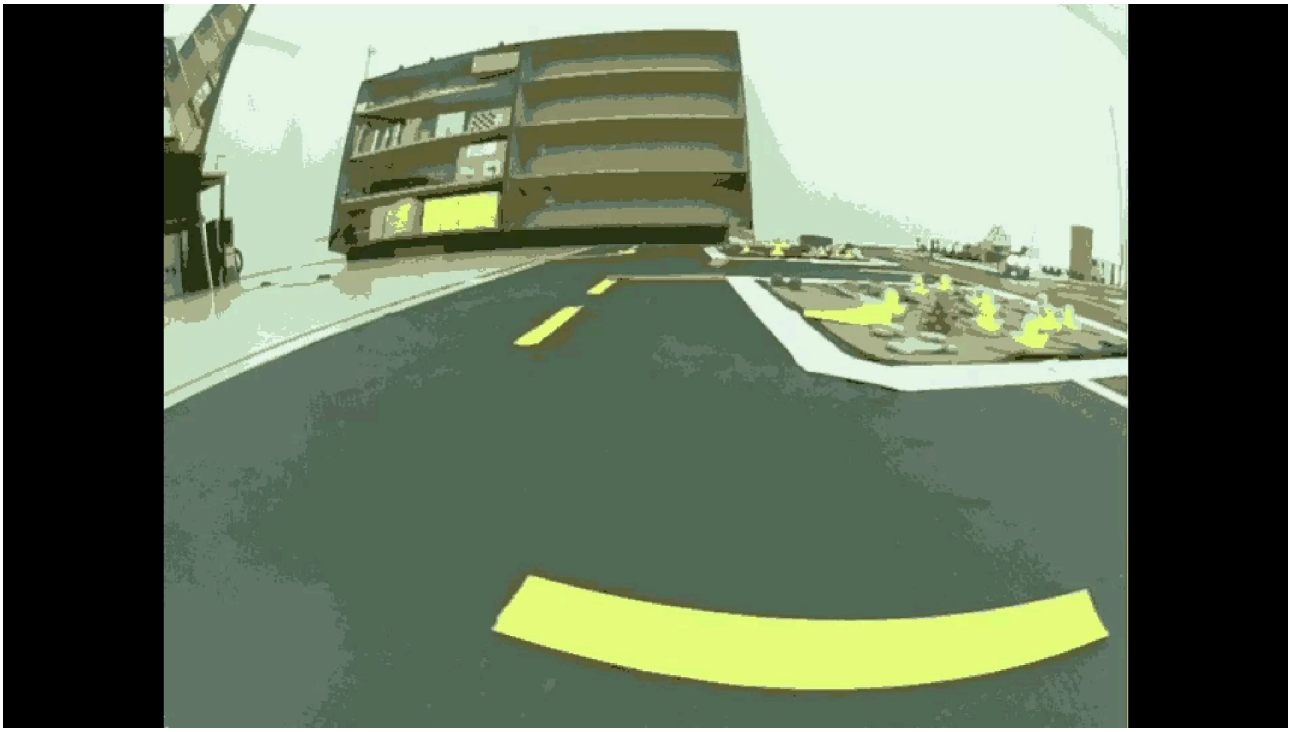
K = 3:



K = 5:



K = 9:



Conclusion:

In this research we investigated how to implement Semantic Segmentation Algorithm on a raw topic images from ROS. Our work was divided into 2 steps: process topic from self-driving duckiebot and get images as a python vector. Next, we applied K - Means clustering on the vector and altered the number of centers K to get 3 experiments. We have learnt how to implement cv_bridge, which enables Conversion between ROS and OpenCV images, we also learnt how to apply Machine Learning Algorithm on the same image.

References:

[1] https://ieeexplore.ieee.org/abstract/document/9023474
[2]https://ieeexplore.ieee.org/abstract/document/8206470
[3]https://www.mdpi.com/2227-7390/8/5/855
[4] https://ieeexplore.ieee.org/abstract/document/8460962
[5] https://iq.opengenus.org/disadvantages-of-cnn/
[6] https://www.sciencedirect.com/science/article/pii/S1877050915014143#:~:text=K%20%2Dmeans%20clustering%20algorithm%20is,the%20quality%20of%20the%20image.