```
#!/usr/bin/python2.7
#DIGITAL TWIN MOBILE ROBOT
#Modelo processamento de imagens para OpenCV - Canny Edge Detection
import rospy
from sensor_msgs.msg import Image
import cv2
import numpy as np #biblioteca para computacao cientifica
import time

#ROS image message -> OpenCV2 image converter
from cv_bridge import CvBridge, CvBridgeError #pacote do ROS com diferentes tipos
de formatacao de imagens

#for delay measure
delay=0
delta=0
bridge = CvBridge()

import sys
import signal
def signal_term_handler(signal, frame):
        rospy.logger('User KeyboardInterrupt')
        sys.exit(0)
signal.signal(signal.SIGINT, signal_term_handler)

#callback function funcao principal
def img_callback(ros_data):
        global delta
        print(int(round(time.time()*1000)) - delta)
        delta = int(round(time.time()*1000))

        try:
                image_np = bridge.imgmsg_to_cv2(ros_data, "mono8") #tipagem do
open cv rgb
                #image_np = bridge.imgmsg_to_cv2(ros_data, "bgr8") #tipagem do
open cv em escala de cinza
                flipVertical_image_np = cv2.flip(image_np, 1)
        except CvBridgeError as e:
                print(e)

        edges = cv2.Canny(image_np, 50, 150, apertureSize=3) #basicamente um novo
frame. opera em escala de cinza
        flipVertical_edges = cv2.flip(edges, 1)

        #cv2.imshow("output", image_np)
        #cv2.imshow("output", np.hstack([image_np, edges])) #concatenacao de
matrizes
        cv2.imshow("output", np.hstack([flipVertical_image_np,
flipVertical_edges]))
        cv2.waitKey(1) #sleep de 1ms


def main():
        rospy.init_node('csf_robotCS_cannyKinect')
        rospy.Subscriber("camera/rgb/image_raw", Image, img_callback, queue_size=1)
        #rospy.Subscriber("/kinect/rgb_image", Image, img_callback, queue_size=1)

        try:
                rospy.spin()
        except KeyboardInterrupt:
                print('Shutting down ...')
        cv2.destroyAllWindows()

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