#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

####################################################################

# 프로그램명 : hough\_drive\_a2.py

# 작 성 자 : 자이트론

# 생 성 일 : 2020년 08월 12일

# 본 프로그램은 상업 라이센스에 의해 제공되므로 무단 배포 및 상업적 이용을 금합니다.

####################################################################

import rospy, rospkg

import numpy as np

import cv2, random, math

from cv\_bridge import CvBridge

from xycar\_motor.msg import xycar\_motor

from sensor\_msgs.msg import Image

import sys

import os

import signal

def signal\_handler(sig, frame):

os.system('killall -9 python rosout')

sys.exit(0)

signal.signal(signal.SIGINT, signal\_handler)

image = np.empty(shape=[0])

bridge = CvBridge()

pub = None

Width = 640

Height = 480

Offset = 340

Gap = 40

def img\_callback(data):

global image

image = bridge.imgmsg\_to\_cv2(data, "bgr8")

# publish xycar\_motor msg

def drive(Angle, Speed):

global pub

msg = xycar\_motor()

msg.angle = Angle

msg.speed = Speed

pub.publish(msg)

# draw lines

def draw\_lines(img, lines):

global Offset

for line in lines:

x1, y1, x2, y2 = line[0]

color = (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))

img = cv2.line(img, (x1, y1+Offset), (x2, y2+Offset), color, 2)

return img

# draw rectangle

def draw\_rectangle(img, lpos, rpos, offset=0):

center = (lpos + rpos) / 2

cv2.rectangle(img, (lpos - 5, 15 + offset),

(lpos + 5, 25 + offset),

(0, 255, 0), 2)

cv2.rectangle(img, (rpos - 5, 15 + offset),

(rpos + 5, 25 + offset),

(0, 255, 0), 2)

cv2.rectangle(img, (center-5, 15 + offset),

(center+5, 25 + offset),

(0, 255, 0), 2)

cv2.rectangle(img, (315, 15 + offset),

(325, 25 + offset),

(0, 0, 255), 2)

return img

# left lines, right lines

def divide\_left\_right(lines):

global Width

low\_slope\_threshold = 0

high\_slope\_threshold = 10

# calculate slope & filtering with threshold

slopes = []

new\_lines = []

for line in lines:

x1, y1, x2, y2 = line[0]

if x2 - x1 == 0:

slope = 0

else:

slope = float(y2-y1) / float(x2-x1)

if abs(slope) > low\_slope\_threshold and abs(slope) < high\_slope\_threshold:

slopes.append(slope)

new\_lines.append(line[0])

# divide lines left to right

left\_lines = []

right\_lines = []

for j in range(len(slopes)):

Line = new\_lines[j]

slope = slopes[j]

x1, y1, x2, y2 = Line

if (slope < 0) and (x2 < Width/2 - 90):

left\_lines.append([Line.tolist()])

elif (slope > 0) and (x1 > Width/2 + 90):

right\_lines.append([Line.tolist()])

return left\_lines, right\_lines

# get average m, b of lines

def get\_line\_params(lines):

# sum of x, y, m

x\_sum = 0.0

y\_sum = 0.0

m\_sum = 0.0

size = len(lines)

if size == 0:

return 0, 0

for line in lines:

x1, y1, x2, y2 = line[0]

x\_sum += x1 + x2

y\_sum += y1 + y2

m\_sum += float(y2 - y1) / float(x2 - x1)

x\_avg = x\_sum / (size \* 2)

y\_avg = y\_sum / (size \* 2)

m = m\_sum / size

b = y\_avg - m \* x\_avg

return m, b

# get lpos, rpos

def get\_line\_pos(img, lines, left=False, right=False):

global Width, Height

global Offset, Gap

m, b = get\_line\_params(lines)

if m == 0 and b == 0:

if left:

pos = 0

if right:

pos = Width

else:

y = Gap / 2

pos = (y - b) / m

b += Offset

x1 = (Height - b) / float(m)

x2 = ((Height/2) - b) / float(m)

cv2.line(img, (int(x1), Height), (int(x2), (Height/2)), (255, 0,0), 3)

return img, int(pos)

# show image and return lpos, rpos

def process\_image(frame):

global Width

global Offset, Gap

# gray

gray = cv2.cvtColor(frame,cv2.COLOR\_BGR2GRAY)

# blur

kernel\_size = 5

blur\_gray = cv2.GaussianBlur(gray,(kernel\_size, kernel\_size), 0)

# canny edge

low\_threshold = 60

high\_threshold = 70

edge\_img = cv2.Canny(np.uint8(blur\_gray), low\_threshold, high\_threshold)

# HoughLinesP

roi = edge\_img[Offset : Offset+Gap, 0 : Width]

all\_lines = cv2.HoughLinesP(roi,1,math.pi/180,30,30,10)

# divide left, right lines

if all\_lines is None:

return 0, 640

left\_lines, right\_lines = divide\_left\_right(all\_lines)

# get center of lines

frame, lpos = get\_line\_pos(frame, left\_lines, left=True)

frame, rpos = get\_line\_pos(frame, right\_lines, right=True)

# draw lines

frame = draw\_lines(frame, left\_lines)

frame = draw\_lines(frame, right\_lines)

frame = cv2.line(frame, (230, 235), (410, 235), (255,255,255), 2)

# draw rectangle

frame = draw\_rectangle(frame, lpos, rpos, offset=Offset)

#roi2 = cv2.cvtColor(roi, cv2.COLOR\_GRAY2BGR)

#roi2 = draw\_rectangle(roi2, lpos, rpos)

# show image

cv2.imshow('calibration', frame)

return lpos, rpos

def start():

global pub

global image

global cap

global Width, Height

rospy.init\_node('auto\_drive')

pub = rospy.Publisher('xycar\_motor', xycar\_motor, queue\_size=1)

image\_sub = rospy.Subscriber("/usb\_cam/image\_raw", Image, img\_callback)

print "---------- Xycar A2 v1.0 ----------"

rospy.sleep(2)

while True:

while not image.size == (640\*480\*3):

continue

lpos, rpos = process\_image(image)

center = (lpos + rpos) / 2

angle = -(Width/2 - center)

drive(angle, 4)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

rospy.spin()

if \_\_name\_\_ == '\_\_main\_\_':

start()