

LAMPIRAN

Program Utama

```
import cv2
import numpy as np
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
from imutils.perspective import four_point_transform

import RPi.GPIO as GPIO
from time import sleep
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
Ena,In1,In2 = 22,17,27

#from imutils import contours
#import imutils

cameraResolution = (720, 480)

# initialize the camera and grab a reference to the raw camera capture
camera = PiCamera()
camera.resolution = cameraResolution
camera.framerate = 30
camera.brightness = 60
camera.rotation = 360
rawCapture = PiRGBArray(camera, size=cameraResolution)

# allow the camera to warmup
time.sleep(2)

def findTrafficSign():
    """
    This function find blobs with blue color on the image.
    After blobs were found it detects the largest square blob,
    that must be the sign.
    """

    # define range HSV for blue color of the traffic sign
    lower_blue = np.array([90,80,50])
    upper_blue = np.array([110,255,255])

while True:
    # The use_video_port parameter controls whether the camera's image or video port is used
    # to capture images. It defaults to False which means that the camera's image port is used.
    # This port is slow but produces better quality pictures.
    # If you need rapid capture up to the rate of video frames, set this to True.
    camera.capture(rawCapture, use_video_port=True, format='bgr')

    # At this point the image is available as stream.array
    frame = rawCapture.array

    frameArea = frame.shape[0]*frame.shape[1]
```

```

# convert color image to HSV color scheme
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

# define kernel for smoothing
kernel = np.ones((3,3),np.uint8)
# extract binary image with active blue regions
mask = cv2.inRange(hsv, lower_blue, upper_blue)
# morphological operations
mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)

# find contours in the mask
cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[-2]

# define string variable to hold detected sign description
detectedTrafficSign = None

# define variables to hold values during loop
largestArea = 0
largestRect = None

# only proceed if at least one contour was found
if len(cnts) > 0:
    for cnt in cnts:
        # Rotated Rectangle. Here, bounding rectangle is drawn with minimum area,
        # so it considers the rotation also. The function used is cv2.minAreaRect().
        # It returns a Box2D structure which contains following details -
        # ( center (x,y), (width, height), angle of rotation ).
        # But to draw this rectangle, we need 4 corners of the rectangle.
        # It is obtained by the function cv2.boxPoints()
        rect = cv2.minAreaRect(cnt)
        box = cv2.boxPoints(rect)
        box = np.int0(box)

        # count euclidian distance for each side of the rectangle
        sideOne = np.linalg.norm(box[0]-box[1])
        sideTwo = np.linalg.norm(box[0]-box[3])
        # count area of the rectangle
        area = sideOne*sideTwo
        # find the largest rectangle within all contours
        if area > largestArea:
            largestArea = area
            largestRect = box

    if largestArea > frameArea*0.02:
        # draw contour of the found rectangle on the original image
        cv2.drawContours(frame,[largestRect],0,(0,0,255),2)

        # cut and warp interesting area
        warped = four_point_transform(mask, [largestRect][0])

        # show an image if rectangle was found
        #cv2.imshow("Warped", cv2.bitwise_not(warped))

        # use function to detect the sign on the found rectangle
        detectedTrafficSign = identifyTrafficSign(warped)
        #print(detectedTrafficSign)

        # write the description of the sign on the original image

```

```

cv2.putText(frame, detectedTrafficSign, (100, 100), cv2.FONT_HERSHEY_SIMPLEX,
1.5, (0, 0, 255), 2)
    print(detectedTrafficSign)
    # show original image
    cv2.imshow("Original", frame)

if detectedTrafficSign == 'Move Straight':

    GPIO.setup(25,GPIO.OUT)
    pwm=GPIO.PWM(25, 360)
    pwm.start(0)
    pwm.ChangeDutyCycle(42) #CENTER
    sleep(1)
    #pwm.ChangeDutyCycle(55) #LEFT
    #sleep(1)
    pwm.stop(1)
    GPIO.setup(Ena,GPIO.OUT)
    GPIO.setup(In1,GPIO.OUT)
    GPIO.setup(In2,GPIO.OUT)
    pwm = GPIO.PWM(Ena,25)
    pwm.start(0)
    GPIO.output(In1,GPIO.HIGH)
    GPIO.output(In2,GPIO.LOW)
    pwm.ChangeDutyCycle(40)
    sleep (2.8)
    pwm.stop(1)

if detectedTrafficSign == 'Turn Back':

    #kiri
    GPIO.setup(25,GPIO.OUT)
    pwm=GPIO.PWM(25, 360)
    pwm.start(0)
    pwm.ChangeDutyCycle(54)
    sleep(1)
    #sleep(1)
    pwm.stop(1)
    GPIO.setup(Ena,GPIO.OUT)
    GPIO.setup(In1,GPIO.OUT)
    GPIO.setup(In2,GPIO.OUT)
    pwm = GPIO.PWM(Ena,25)
    pwm.start(0)
    GPIO.output(In1,GPIO.HIGH)
    GPIO.output(In2,GPIO.LOW)
    pwm.ChangeDutyCycle(40)
    sleep (4.3)
    pwm.stop(1)

    GPIO.setup(25,GPIO.OUT)
    pwm=GPIO.PWM(25, 360)
    pwm.start(0)
    pwm.ChangeDutyCycle(45) #CENTER
    sleep(1)
    #pwm.ChangeDutyCycle(55) #LEFT
    #sleep(1)
    pwm.stop(1)
    GPIO.setup(Ena,GPIO.OUT)
    GPIO.setup(In1,GPIO.OUT)
    GPIO.setup(In2,GPIO.OUT)
    pwm = GPIO.PWM(Ena,25)

```

```
pwm.start(0)
GPIO.output(In1,GPIO.HIGH)
GPIO.output(In2,GPIO.LOW)
pwm.ChangeDutyCycle(40)
sleep (2)
pwm.stop(1)
```

```
if detectedTrafficSign == 'Turn Left':
```

```
#kiri
GPIO.setup(25,GPIO.OUT)
pwm=GPIO.PWM(25, 360)
pwm.start(0)
pwm.ChangeDutyCycle(54)
sleep(1)
#sleep(1)
pwm.stop(1)
GPIO.setup(Ena,GPIO.OUT)
GPIO.setup(In1,GPIO.OUT)
GPIO.setup(In2,GPIO.OUT)
pwm = GPIO.PWM(Ena,25)
pwm.start(0)
GPIO.output(In1,GPIO.HIGH)
GPIO.output(In2,GPIO.LOW)
pwm.ChangeDutyCycle(40)
sleep (4.3)
pwm.stop(1)

GPIO.setup(25,GPIO.OUT)
pwm=GPIO.PWM(25, 360)
pwm.start(0)
pwm.ChangeDutyCycle(45) #CENTER
sleep(1)
#pwm.ChangeDutyCycle(55) #LEFT
#sleep(1)
pwm.stop(1)
GPIO.setup(Ena,GPIO.OUT)
GPIO.setup(In1,GPIO.OUT)
GPIO.setup(In2,GPIO.OUT)
pwm = GPIO.PWM(Ena,25)
pwm.start(0)
GPIO.output(In1,GPIO.HIGH)
GPIO.output(In2,GPIO.LOW)
pwm.ChangeDutyCycle(40)
sleep (2)
pwm.stop(1)
```

```
if detectedTrafficSign == 'Turn Right':
```

```
#kiri
GPIO.setup(25,GPIO.OUT)
pwm=GPIO.PWM(25, 360)
pwm.start(0)
pwm.ChangeDutyCycle(54)
sleep(1)
#sleep(1)
pwm.stop(1)
GPIO.setup(Ena,GPIO.OUT)
```

```

GPIO.setup(In1,GPIO.OUT)
GPIO.setup(In2,GPIO.OUT)
pwm = GPIO.PWM(Ena,25)
pwm.start(0)
GPIO.output(In1,GPIO.HIGH)
GPIO.output(In2,GPIO.LOW)
pwm.ChangeDutyCycle(40)
sleep (4.3)
pwm.stop(1)

GPIO.setup(25,GPIO.OUT)
pwm=GPIO.PWM(25, 360)
pwm.start(0)
pwm.ChangeDutyCycle(45) #CENTER
sleep(1)
#pwm.ChangeDutyCycle(55) #LEFT
#sleep(1)
pwm.stop(1)
GPIO.setup(Ena,GPIO.OUT)
GPIO.setup(In1,GPIO.OUT)
GPIO.setup(In2,GPIO.OUT)
pwm = GPIO.PWM(Ena,25)
pwm.start(0)
GPIO.output(In1,GPIO.HIGH)
GPIO.output(In2,GPIO.LOW)
pwm.ChangeDutyCycle(40)
sleep (2)
pwm.stop(1)

# clear the stream in preparation for the next frame
rawCapture.truncate(0)

# if the `q` key was pressed, break from the loop
if cv2.waitKey(1) & 0xFF is ord('q'):
    cv2.destroyAllWindows()
    print("Stop programm and close all windows")
    break

def identifyTrafficSign(image):
"""

In this function we select some ROI in which we expect to have the sign parts. If the ROI has more active pixels than threshold we mark it as 1, else 0
After path through all four regions, we compare the tuple of ones and zeros with keys in dictionary SIGNS_LOOKUP
"""

# define the dictionary of signs segments so we can identify
# each signs on the image
SIGNS_LOOKUP = {
    (1, 0, 0, 1): 'Turn Right', # turnRight
    (0, 0, 1, 1): 'Turn Left', # turnLeft
    (0, 1, 0, 1): 'Move Straight', # moveStraight
    (1, 0, 1, 1): 'Turn Back', # turnBack
}

THRESHOLD = 150

image = cv2.bitwise_not(image)
# (roiH, roiW) = roi.shape

```

```

#subHeight = thresh.shape[0]/10
#subWidth = thresh.shape[1]/10
(subHeight, subWidth) = np.divide(image.shape, 10)
subHeight = int(subHeight)
subWidth = int(subWidth)

# mark the ROIs borders on the image
#cv2.rectangle(image, (subWidth, 4*subHeight), (3*subWidth, 9*subHeight), (0,255,0),2) # left
block
#cv2.rectangle(image, (4*subWidth, 4*subHeight), (6*subWidth, 9*subHeight), (0,255,0),2) # center
block
#cv2.rectangle(image, (7*subWidth, 4*subHeight), (9*subWidth, 9*subHeight), (0,255,0),2) # right
block
#cv2.rectangle(image, (3*subWidth, 2*subHeight), (7*subWidth, 4*subHeight), (0,255,0),2) # top
block

# subtract 4 ROI of the sign thresh image
leftBlock = image[4*subHeight:9*subHeight, subWidth:3*subWidth]
centerBlock = image[4*subHeight:9*subHeight, 4*subWidth:6*subWidth]
rightBlock = image[4*subHeight:9*subHeight, 7*subWidth:9*subWidth]
topBlock = image[2*subHeight:4*subHeight, 3*subWidth:7*subWidth]

# we now track the fraction of each ROI
leftFraction = np.sum(leftBlock)/(leftBlock.shape[0]*leftBlock.shape[1])
centerFraction = np.sum(centerBlock)/(centerBlock.shape[0]*centerBlock.shape[1])
rightFraction = np.sum(rightBlock)/(rightBlock.shape[0]*rightBlock.shape[1])
topFraction = np.sum(topBlock)/(topBlock.shape[0]*topBlock.shape[1])

segments = (leftFraction, centerFraction, rightFraction, topFraction)
segments = tuple(1 if segment > THRESHOLD else 0 for segment in segments)

cv2.imshow("Warped", image)

if segments in SIGNS_LOOKUP:
    return SIGNS_LOOKUP[segments]
else:
    return None

def main():
    findTrafficSign()

    if __name__ == '__main__':
        main()

```

1.Tahapan cara Install Python versi terbaru di Raspberry

Hal petama yang dapat dilakukan adalah mengecek apakah Python default Anda ada pada Python 3.x atau pada Python 2.x, untuk memastikan hal ini gunakan command sebagai beriku pada bagian terminal :

```
$ python --version
```

Beberapa hal perlu dilakukan untuk mengubah Python 2.7 menjadi Python 3.6. Hal pertama yang perlu diperhatikan adalah memastikan bahwa beberapa package dependencies sudah di-install pada Raspberry Pi. Pastikan dengan menggunakan command sebagai berikut:

```
# apt-get update  
  
# apt-get install build-essential tk-dev libncurses5-dev libncursesw5-dev libreadline6-dev libdb5.3-dev  
libgdbm-dev libsqlite3-dev libssl-dev libbz2-dev libexpat1-dev liblzma-dev zlib1g-dev
```

Setelah melaksanakan beberapa program tersebut, Anda dapat langsung mengunduh source code Python 3.6 dari website resmi Python. Source code Python 3.6 yang tersedia berupa tar.xz sehingga anda perlu melakukan proses unpack. Sebelum mengunduh dengan wget, pastikan Anda sudah di direktori /tmp/ agar source code yang Anda unduh akan terhapus secara otomatis setelah reboot.

```
$ cd /tmp/  
  
$ wget https://www.python.org/ftp/python/3.6.4/Python-3.6.4.tar.xz  
  
$ tar xf Python-3.6.4.tar.xz
```

Setelah Anda berhasil mengekstrak seperti pada Gambar 2, Anda dapat melihat bahwa source code Python 3.6.4 sudah berada pada /tmp/Python-3.6.4. Untuk melanjutkan proses instalasi Python 3.6.4, akses direktori Python-3.6.4

```
$ cd Python-3.6.4
```

Setelah membuka direktori, ikuti proses ini untuk instalasi Python 3.6.4:

```
$ ./configure  
$ make  
# make altinstall
```

Setelah instalasi Python 3.6, Anda akan melihat bahwa Python default masih seperti Python sebelumnya,

untuk saya Python masih 2.7.13. Untuk mengubah default Python yang digunakan perlu digunakan update-alternatives. Pertama, pastikan Anda tahu posisi instalasi Python 3.6 dan bahwa Python 3.6 yang sudah diinstal dapat diakses. Lakukan kedua command ini untuk mengetahui posisi Python 3.6

```
$ ls /usr/bin/python*  
$ ls /usr/local/bin/python*
```

Setelah mengetahui posisi executable Python 3.6, lakukan update-alternatives untuk menambahkan Python 2.7.13 dan Python 3.6.4 sebagai alternatif Python. Gambar 3 menunjukkan hasil keluaran dari command untuk mengetahui posisi executable dari Python2.7, Python3.5 dan Python3.6. Tambahkan semua executable Python ini sebagai alternatif dari Python. Setelah mengetahui posisi yakni pada /usr/bin/ untuk Python2.7 dan Python3.5 dan pada /usr/local/bin/ pada Python3.6 yang baru diinstal. Lakukan proses penambahan alternatif /usr/bin/python dengan menggunakan command sebagai berikut:

```
# update-alternatives --install /usr/bin/python python /usr/bin/python2.7 1  
# update-alternatives --install /usr/bin/python python /usr/bin/python3.5 2  
# update-alternatives --install /usr/bin/python python /usr/local/bin/python3.6 3
```

Setelah menambahkan sesuai urutan di atas, Python3.6 menjadi mode automatis dari python.

Berikut ini adalah proses instalasi OpenCV yang akan digunakan untuk implementasi robot ini. Masukan perintah dibawah ini satu persatu hingga selesai.

```
sudo apt install cmake build-essential pkg-config git  
sudo apt install libjpeg-dev libtiff-dev libjasper-dev libpng-dev libwebp-dev libopenexr-dev  
sudo apt install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev libxvidcore-dev libx264-dev  
libdc1394-22-dev libgstreamer-plugins-base1.0-dev libgstreamer1.0-dev  
sudo apt install libgtk-3-dev libqtgui4 libqtwebkit4 libqt4-test python3-pyqt5  
sudo apt install libatlas-base-dev liblapacke-dev gfortran  
sudo apt install libhdf5-dev libhdf5-103  
sudo apt install python3-dev python3-pip python3-numpy
```

Untuk langkah selanjutnya kita harus memperbesar ‘swap file’ di memory card Raspi untuk memudahkan proses instalasi dari 100 menjadi 2048

```
sudo nano /etc/dphys-swapfile
```

```
CONF_SWAPSIZE=2048
```

untuk menyimpan settingan di atas tekan CTRL-X, pilih Y (yes) kemudian enter. Jangan lupa restart untuk memastikan swap file sudah berubah.

```
sudo systemctl restart dphys-swapfile
```

Setelah itu masukkan perintah berikut satu persatu.

```
git clone https://github.com/opencv/opencv.git
```

```
git clone https://github.com/opencv/opencv_contrib.git
```

```
mkdir ~/opencv/build
```

```
cd ~/opencv/build
```

```
cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=/usr/local \
-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules \
-D ENABLE_NEON=ON \
-D ENABLE_VFPV3=ON \
-D BUILD_TESTS=OFF \
-D INSTALL_PYTHON_EXAMPLES=OFF \
-D OPENCV_ENABLE_NONFREE=ON \
-D CMAKE_SHARED_LINKER_FLAGS=-latomic \
-D BUILD_EXAMPLES=OFF \
make -j$(nproc)
```

```
sudo make install
```

```
sudo ldconfig
```

Proses instalasi paket OpenCV sudah selesai. Anda dapat mengecek apakah instalasi berhasil dengan perintah sbb:

```
python
import cv2
cv2.__version__
```

biasanya pada project computer vision dengan OpenCV membutuhkan library tambahan yakni ‘opencv-contrib-python’.

```
pip install opencv-contrib-python
```

```
sudo nano /etc/dphys-swapfile
```

```
CONF_SWAPSIZE=100
```

```
sudo systemctl restart dphys-swapfile
```



PAPER NAME

TA-18.F1.0016.docx

WORD COUNT

4981 Words

CHARACTER COUNT

29853 Characters

PAGE COUNT

32 Pages

FILE SIZE

62.8KB

SUBMISSION DATE

Oct 13, 2022 2:17 PM GMT+7

REPORT DATE

Oct 13, 2022 2:18 PM GMT+7

● **10% Overall Similarity**

The combined total of all matches, including overlapping sources, for each database.

- 9% Internet database
- Crossref database
- 7% Submitted Works database
- 1% Publications database
- Crossref Posted Content database

[Summary](#)