LAMPIRAN

Program *color tracking*

```
# import the necessary packages
from __future__ import print_function
from imutils.video import VideoStream
import argparse
import imutils
import time
import cv2

# initialize the video stream and allow the Camera sensor to
     warmup
vs = VideoStream(0).start()
time.sleep(2.0)

# define the lower and upper boundaries of the object
# to be tracked in the HSV color space
colorLower = (10, 100, 100)
colorUpper = (30, 255, 255)

while True:
    # grab the next frame from the video stream, Invert 0o, resize
     the
    # frame, and convert it to the HSV color space
    frame = vs.read()
    frame = imutils.resize(frame, width=640)
    frame = imutils.rotate(frame, angle=0)
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    # construct a mask for the object color, then perform
    # a series of dilations and erosions to remove any small
    # blobs left in the mask
    mask = cv2.inRange(hsv, colorLower, colorUpper)
    mask = cv2.erode(mask, None, iterations=2)
    mask = cv2.dilate(mask, None, iterations=2)
    # find contours in the mask and initialize the current
    # (x, y) center of the object
    cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE)
    cnts = cnts[0] if imutils.is_cv2() else cnts[1]
    max_area = 0
    best_cnt = 1
    if len(cnts) > 0:
            for Pic, contour in enumerate(cnts):
                    area = cv2.contourArea(contour)
                    if (area > max_area) :
                            max_area = area
                            best_cnt = contour
                            x,y,w,h = cv2.boundingRect(best_cnt)
                            cv2.rectangle(frame, (x, y),
     (x+w,y+h),(0, 255, 0), 2)
```

```
    cv2.imshow('frame', frame)
    key = cv2.waitKey(1) & 0xFF
    if key == ord("q"):
                break

cv2.destroyAllWindows()
vs.stop()
```

Program *color tracking* dengan sistem *Pan-Tilt*

```
#PigPio module for servo instead of RPi.GPIO in Raspberry Pi avoids
      jittering.
import cv2
from imutils.video import VideoStream
import numpy as np
import imutils
import time
import RPi.GPIO as GPIO
import pigpio
import argparse
from time import sleep
from numpy import interp

#define Servos GPIOs
panServo = 22
tiltServo = 27

# initialize LED GPIO
redLed = 17
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(redLed, GPIO.OUT)

# initialize buzzer GPIO
buzzer = 18
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(buzzer, GPIO.OUT)

MIN_PW = 500
MID_PW = 1500
MAX_PW = 2500

panPos = 1500
tiltPos = 1000

servo = pigpio.Pi()
servo.set_servo_pulsewidth(panServo, panPos)
servo.set_servo_pulsewidth(tiltServo, tiltPos)

minMov = 10
maxMov = 60

# define the lower and upper boundaries of the object
# to be tracked in the HSV color space
colorLower = (10, 100, 100)
colorUpper = (30, 255, 255)

# initialize the video stream and allow the Camera sensor to warmup
vs = VideoStream(0).start()
time.sleep(2.0)
cod = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('detec1.avi', cod, 100.0, (640,480))

# Start with LED off
GPIO.output(redLed, GPIO.LOW)
ledOn = False
# Start with buzzer off
GPIO.output(buzzer, GPIO.LOW)
buzzerOn = False
```

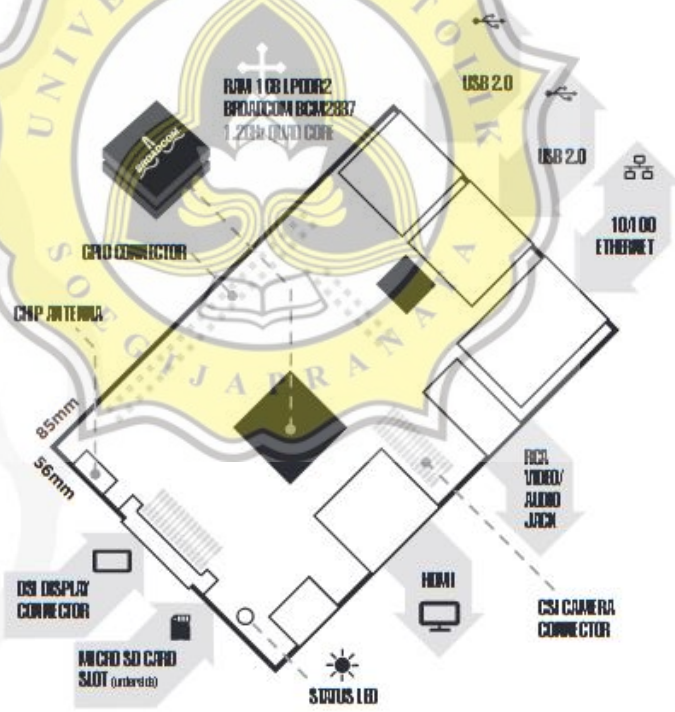
```
def movePanTilt(x, y, w, h):
    global panPos
    global tiltPos
    cv2.rectangle(frame, (x, y), (x+w, y+h),(0, 255, 0), 2)
    if int(x+(w/2)) > 360:
        panPos = int(panPos - interp(int(x+(w/2)), (360, 640), (minMov,
      maxMov)))
    elif int(x+(w/2)) < 280:
        panPos = int(panPos + interp(int(x+(w/2)), (280, 0), (minMov,
      maxMov)))
    if int(y+(h/2)) > 280:
        tiltPos = int(tiltPos + interp(int(y+(h/2)), (280, 480), (minMov,
      maxMov)))
    elif int(y+(h/2)) < 200:
        tiltPos = int(tiltPos - interp(int(y+(h/2)), (200, 0), (minMov,
      maxMov)))
    if not panPos < 2500 or panPos > 500:
        servo.set_servo_pulsewidth(panServo, panPos)
    if not tiltPos < 2500 or tiltPos > 500:
        servo.set_servo_pulsewidth(tiltServo, tiltPos)

while True:
    # grab the next frame from the video stream, Invert 0o, resize the
    # frame, and convert it to the HSV color space
    frame = vs.read()
    frame = imutils.resize(frame, width=640)
    frame = imutils.rotate(frame, angle=0)
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    # construct a mask for the object color, then perform
    # a series of dilations and erosions to remove any small
    # blobs left in the mask
    mask = cv2.inRange(hsv, colorLower, colorUpper)
    mask = cv2.erode(mask, None, iterations=2)
    mask = cv2.dilate(mask, None, iterations=2)
    # find contours in the mask and initialize the current
    # (x, y) center of the object
    cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE)
    cnts = cnts[0] if imutils.is_cv2() else cnts[1]
    max_area = 0
    best_cnt = 1
    if len(cnts) > 0:
            for Pic, contour in enumerate(cnts):
                    area = cv2.contourArea(contour)
                    if (area > max_area) :
                            max_area = area
                            best_cnt = contour
                            (x, y, w, h) = cv2.boundingRect(best_cnt)
                            movePanTilt(x, y, w, h)
                            # if the buzzer is not already on, turn the
                                    buzzer on
                            if not buzzerOn:
                                GPIO.output(buzzer, GPIO.HIGH)
                                buzzerOn = True
                            # if the led is not already on, turn the LED
                                     on
                            if not ledOn:
                                GPIO.output(redLed, GPIO.HIGH)
                                ledOn = True
    # if the ball is not detected, turn the buzzer off
    elif buzzerOn:
            GPIO.output(buzzer, GPIO.LOW)
```

```
            buzzerOn = False
    # if the ball is not detected, turn the LED off
    elif ledOn:
            GPIO.output(redLed, GPIO.LOW)
            ledOn = False

    cv2.imshow('frame', frame)
    out.write(frame)
    key = cv2.waitKey(1) & 0xFF
    if key == ord("q"):
                break

servo.set_servo_pulsewidth(22, 0)
servo.set_servo_pulsewidth(27, 0)
servo.stop()
GPIO.cleanup()
out.release()
cv2.destroyAllWindows()
```

Raspberry Pi

Raspberry Pi 3 Model B

Product Name	**Raspberry Pi 3**
Product Description	The Raspberry Pi 3 Model B is the third generation Raspberry Pi. This powerful credit-card sized single board computer can be used for many applications and supersedes the original Raspberry Pi Model B+ and Raspberry Pi 2 Model B. Whilst maintaining the popular board format the Raspberry Pi 3 Model B brings you a more powerful processer, 10x faster than the first generation Raspberry Pi. Additionally it adds wireless LAN & Bluetooth connectivity making it the ideal solution for powerful connected designs.
RS Part Number	896-8650



www.rs-components.com/raspberrypi

Raspberry Pi

Raspberry Pi 3 Model B

Specifications

Processor	Broadcom BCM2387 chipset. 1.2GHz Quad-Core ARM Cortex-A53 802.11 b/g/n Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE)
GPU	Dual Core VideoCore IV® Multimedia Co-Processor. Provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode. Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure
Memory	1GB LPDDR2
Operating System	Boots from Micro SD card, running a version of the Linux operating system or Windows 10 IoT
Dimensions	85 x 56 x 17mm
Power	Micro USB socket 5V 1, 2.5A

Connectors:

Ethernet	10/100 Base T Ethernet socket
Video Output	HDMI (rev 1.3 & 1.4 Composite RCA (PAL and NTSC)
Audio Output	Audio Output 3.5mm jack, HDMI USB 4 x USB 2.0 Connector
GPIO Connector	40 pin 2.54 mm (100 mil) expansion header: 2x20 strip Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines
Camera Connector	15-pin MIPI Camera Serial Interface (CSI-2)
Display Connector	Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane
Memory Card Slot	Push/pull Micro SDIO

Key Benefits

- Low cost
- Consistent board format
- 10x faster processing
- Added connectivity

Key Applications

- Low cost PC/tablet/laptop
- IoT applications
- Media centre
- Robotics
- Industrial/Home automation
- Server/cloud server
- Print server
- Security monitoring
- Web camera
- Gaming
- Wireless access point
- Environmental sensing/monitoring (e.g. weather station)

www.rs-components.com/raspberrypi

Raspberry Pi Camera v2

Part number: *RPI 8MP CAMERA BOARD*

- 8 megapixel camera capable of taking photographs of 3280 x 2464 pixels
- Capture video at 1080p30, 720p60 and 640x480p90 resolutions
- All software is supported within the latest version of Raspbian Operating System

The Camera v2 is the new official camera board released by the Raspberry Pi foundation.

The Raspberry Pi Camera Module v2 is a high quality 8 megapixel Sony IMX219 image sensor custom designed add-on board for Raspberry Pi, featuring a fixed focus lens. It's capable of 3280 x 2464 pixel static images, and also supports 1080p30, 720p60 and 640x480p60/90 video. It attaches to Pi by way of one of the small sockets on the board upper surface and uses the dedicated CSi interface, designed especially for interfacing to cameras.

- 8 megapixel native resolution sensor-capable of 3280 x 2464 pixel static images
- Supports 1080p30, 720p60 and 640x480p90 video
- Camera is supported in the latest version of Raspbian, Raspberry Pi's preferred operating system

The board itself is tiny, at around 25mm x 23mm x 9mm. It also weighs just over 3g, making it perfect for mobile or other applications where size and weight are important. It connects to Raspberry Pi by way of a short ribbon cable.
The high quality Sony IMX219 image sensor itself has a native resolution of 8 megapixel, and has a fixed focus lens on-board. In terms of still images, the camera is capable of 3280 x 2464 pixel static images, and also supports 1080p30, 720p60 and 640x480p90 video.

Applications

- CCTV security camera
- motion detection
- time lapse photography


POLITEKNIK NEGERI JAKARTA
SERTIFIKAT
PNJ SNTE
Nomor :197/PL3/DL/2019
Diberikan Kepada :
Christophorus Edward
SEBAGAI PEMAKALAH
SEMINAR NASIONAL TEKNIK ELEKTRO 2019
"Implementasi Riset Teknologi Inovatif pada Era Revolusi Industri 4.0"
Gedung Direktorat Politeknik Negeri Jakarta
Kamis, 21 November 2019
Ketua Jurusan Teknik Elektro
Ir. Sri Danaryani, M.T.
NIP. 19630503 199103 2 001
Ketua Pelaksana SNTE 2019
Dr. Isdawimah, S.T., M.T.
NIP. 19630505 198811 2 001

FAKULTAS TEKNIK
Jl. Pawiyatan Luhur IV/1 Bendan Duwur Semarang 50234
Telp : (024) 8441555 (hunting) Fax : (024) 8415429 – 8445265
Email : elektro@unika.ac.id

Unika SOEGIJAPRANATA

SURAT-TUGAS

Nomor : 00130/B.8.7/ST.FT/11/2019

Dekan Fakultas Teknik Universitas Katolik Soegijapranata Semarang, dengan ini memberikan tugas kepada staf mahasiswa Program Studi Teknik Elektro sebagai berikut:

Nama	:	Christophorus Edward Setiawan Ghanie
Status	:	Mahasiswa Program Studi Teknik Elektro Fakultas Teknik Universitas Katolik Soegijapranata Semarang
Tugas	:	Sebagai Pembicara Seminar Nasional Teknik Elektro 2019 dengan judul makalah "**Penerapan Sistem Pan-Tilt Camera untuk Deteksi Objek berdasarkan Warna menggunakan Raspberry Pi**"
Waktu	:	Kamis, 21 November 2019
Tempat	:	Gedung Rektorat (Gedung Q) Lantai 3, Politeknik Negeri Jakarta, Kampus Universitas Indonesia Depok.
Lain-lain	:	Harap dilaksanakan dengan sebaik-baiknya dengan penuh tanggung jawab dan setelah selesai harap memberikan laporan.

Semarang, 18 November 2019
Dekan,

Prof. Dr. Ir. Slamet Riyadi, MT
NPP. 058.1.1992.110

Telah melaksanakan tugas,

(_____)

PNJ SNTE

Ketua Panitia SNTE 2019
Dr. Isdawimah, S.T., M.T.
NIP.196305051988112001

PLAGIARISM CHECK.ORG

6.83% PLAGIARISM APPROXIMATELY

Report #10354064

BAB IPENDAHULUAN Latar BelakangTeknologi seperti kamera pada awalnya hanya digunakan untuk menangkap gambar saja. Namun seiring perkembangan teknolog , kamera dapat difungsikan sebagai sensor untuk mendeteksi apapun gambar yang ditangkap oleh kamera. Gambar yang dihasilkan kemudian diolah oleh perangkat komputer yang sudah terprogram untuk tujuan tersebut. Teknologi itu dinamakan Computer Vision. Teknologi ini banyak digunakan untuk tujuan tertentu. Dikarenakan hanya mengandalkan kamera yang sudah terkomputasi, teknologi ini dapat menggantikan fungsi dari banyak sensor seperti sensor warna,sensor jarak,sensor gerak dan sebagainyaADDIN [1]. Pada penelitian ini penulis akan menerapkan pendeteksi objek berdasarkan warna untuk sistem Pan-Tilt Camera otomatis yang terprogram dari komputermini Raspberry Pi. Program tersebut dibuat menggunakan bahasa pemrograman Python dan library OpenCV untuk menangkap objek yang sudah ditentukan warnanya berdasarkan warna RGB (Red,Green,Blue) yang kemudian dikonversi kembali menjadi HSV (Hue,Saturation,Value) untuk menentukan titik atas dan bawah warna tertentu yang akan dideteksi menggunakan library OpenCVADDIN [2]. Setelah warna pada objek tertangkap dengan baik maka kamera beserta Pan-Tiltnya akan terus melacak dan bergerak mengikuti objek yang sudah