T.C.

MARMARA ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

MEKATRONİK MÜHENDİSLİĞİ BÖLÜMÜ

PROJE RAPORU

ÖĞRENCİLERİN

ADI SOYADI : Yunus Emre ÖZDEMİR

NUMARASI : 170215030

ADI SOYADI : Zeliha AKIN

NUMARASI : 170215033

DERSİN

ADI : : Gömülü Sistemler

ÖĞRETİM ÜYESİ :Yrd. Doç. Dr. Hüseyin YÜCE

PROJE ADI : HEDEF TAKİP SİSTEMİ

TARİH :22.05.2018

GİRİŞ

Hareket eden bir nesnenin, cismin takibi her zaman farklı amaçlar için kullanılan gerekli görülmüş bir ihtiyaçtır. Eski zamanlarda bu işi gözcü insanlar yaparken gelişen teknolojiyle yavaş yavaş bunu teknolojik ve daha güvenilir araçlara bırakmıştır. Projemiz gerek askeri gerek güvenlik gerek otomasyon gerekse eğlence alanında istenilen doğrultuda geliştirilmeye ve ilerlemeye açıktır. Askeriyede otonom kamikaze araçlarda kullanılabileceği gibi güvenlik sistemlerinde en basit mobese sisteminin oluşumunda kullanılabilir.

Hedef Takip Sistemimizde bu amaçlar doğrultusunda çalışmak üzere tasarlanmıştır.

Raspberry Pi kamera modülü iki eksenli bir servo standına monte edilecektir. Servo standına bulunan Raspberry Pi kamera modülü vasıtasıyla görüntü alınıp, alınan görüntü Raspberry Pi cihazına aktaracaktır. Daha sonrasında Raspberry Pi cihazında, Opencv kütüphanesinden yararlanarak elde edilen görüntü işlenecektir.

Gerekli Donanım Bileşenleri

1. 1 adet Rasperry Pi

2. 1 adet Kamera

3. 1 adet Kamera Standı

Gerekli Yazılım Bileşenleri

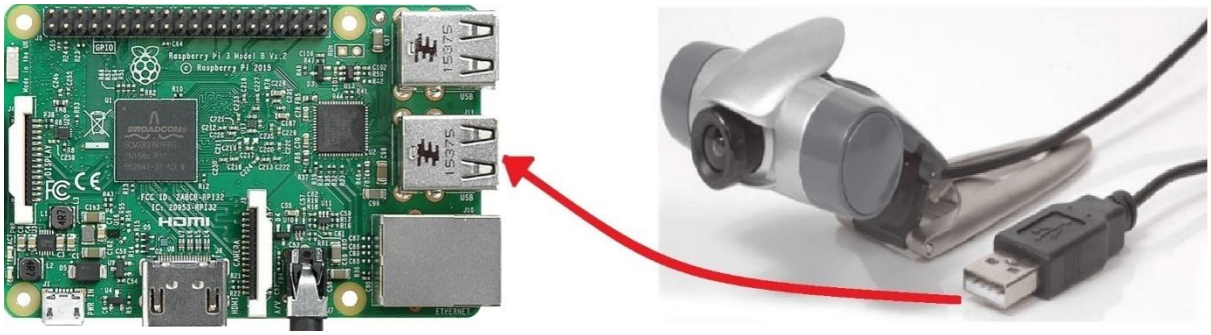
1. Raspbian Stretch (https://www.raspberrypi.org/downloads/raspbian/)
2. Opencv (https://opencv.org/)

Kullanılan Bileşenlerin Özellikleri

1. Raspberry Pi:
   Tek kartlı bir bilgisayar olan Raspberry Pi, Raspbian, Snappy Ubuntu Core, Windows 10 IoT Core gibi işletim sistemlerini destekler. Python programlama dili ile programlanabildiği gibi C, Perl dillerinde de programlama yapılabilmektedir. Raspberry Pi 2, Raspberry Pi 3 gibi modellerini bulunan Raspberry Pi'nin aynı zaman da Model A ve Model B olmak üzere çeşitleri bulunmaktadır.
   Biz projemize uygun olarak Raspberry Pi 3 Model B kullandık. Bu modelde 1.2 GHz 64-bit dört çekirdekli CPU, 1 GB RAM, 4 adet USB 2.0 port gibi yetkinlikleri vardır.
2. USRobotics Camera:
   Görüntü almak için kullandığımız bu webcam elimizde hali hazırda bulunmakta olup USB bağlantısı ve fotoğraf çekebilme özelliği ile bizim projemiz için uygun bir webcamdir. (https://support.usr.com/support/9640/9640-tu-ug/index.html)
3. Kamera Standı:

2 eksenli 2 servo motor bulunan bir standdır. Servolara gönderilen pwm sinyali ile kontrol edilmektedir.

Şematik Çizimi



Yapım Aşamaları

OpenCV Kurulumu

OpenCV (Open Source Computer Vision) açık kaynak kodlu görüntü işleme kütüphanesidir. Open source yani açık kaynak kodlu bir kütüphanedir ve BSD lisansı ile altında geliştirilmektedir. BSD lisansına sahip olması bu kütüphaneyi istediğiniz projede ücretsiz olarak kullanabileceğiniz anlamına gelmektedir. OpenCV platform bağımsız bir kütüphanedir, bu sayede Windows, Linux, FreeBSD, Android, Mac OS ve iOS platformlarında çalışabilmektedir. C++, C, Python, Java, Matlab, EmguCV kütüphanesi aracılığıyla da Visual Basic.Net, C# ve Visual C++ dilleri ile topluluklar tarafından geliştirilen farklı wrapperlar aracılığıyla Perl ve Ruby programlama dilleri ile kolaylıkla OpenCV uygulamaları geliştirilebilir.

OpenCV kütüphanesi içerisinde görüntü işlemeye (image processing) ve makine öğrenmesine (machine learning) yönelik 2500'den fazla algoritma bulunmaktadır. Bu algoritmalar ile yüz tanıma, nesneleri ayırt etme, insan hareketlerini tespit edebilme, nesne sınıflandırma, plaka tanıma, üç boyutlu görüntü üzerinde işlem yapabilme, görüntü karşılaştırma, optik karakter tanımlama OCR (Optical Character Recognition) gibi işlemler rahatlıkla yapılabilmektedir.

Adım 1: Dosya siteminin genişletilmesi

```
1. $ sudo raspi - config
2. $ sudo reboot
3. $ sudo apt - get purge wolfram - engine
4. $ sudo apt - get purge libreoffice *
```

```
$ sudo apt - get clean
$ sudo apt - get autoremove
```

2. Adım: Bağlı dosyaların yüklenmesi

```
$ sudo apt - get update && sudo apt - get upgrade
$ sudo apt - get install build - essential cmake pkg - config
$sudo apt - get install libjpeg - dev libtiff5 - dev libjasper - dev libpng12 - dev
$ sudo apt - get install libavcodec - dev libavformat - dev libswscale - dev libv4l - dev
$ sudo apt - get install libxvidcore - dev libx264 - dev
$ sudo apt - get install libgtk2.0 - dev libgtk - 3 - dev
$ sudo apt - get install libatlas - base - dev gfortran
$ sudo apt - get install python2.7 - dev python3 - dev
```

3. Adım: OpenCV'nin kaynak kodlarının indirilmesi

```
$ cd~
$ wget - O opencv.zip https: //github.com/Itseez/opencv/archive/3.3.0.zip
$ unzip opencv.zip
$ wget - O opencv_contrib.zip https: //github.com/Itseez/opencv_contrib/archive/3.3.0.zip
$ unzip opencv_contrib.zip
```

4. Adım: Paketleme işlemleri

```
$ wget https: //bootstrap.pypa.io/get-pip.py
$ sudo python get - pip.py
$ sudo python3 get - pip.py
$ sudo pip install virtualenv virtualenvwrapper
$ sudo rm - rf~/.cache/pip
#  virtualenv and virtualenvwrapper
export WORKON_HOME = $HOME / .virtualenvs source / usr / local / bin / virtualenvwrapper.s
$ echo - e "\n# virtualenv and virtualenvwrapper" >> ~/.profile
$ echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.profile
$ echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.profile
$ source~/.profile
$ pip install numpy
```

5. Adım: Sanal alanın oluşturulması

```
$ mkvirtualenv cv - p python3
```

6. Adım: Derleme ve yüklemenin tamamlanması

```
$ workon cv
$ cd~/opencv-3.3.0/
$ mkdir build
$ cd build
$ cmake - D CMAKE_BUILD_TYPE = RELEASE\
        - D CMAKE_INSTALL_PREFIX = /usr/local\
        - D INSTALL_PYTHON_EXAMPLES = ON\
        - D OPENCV_EXTRA_MODULES_PATH = ~/opencv_contrib-3.3.0/modules\
```

```
            - D BUILD_EXAMPLES = ON..
            CONF_SWAPSIZE = 1024
$ sudo / etc / init.d / dphys - swapfile stop
$ sudo / etc / init.d / dphys - swapfile start
$ make - j4
$ sudo make install
$ sudo ldconfig
$ ls - l / usr / local / lib / python3.5 / site - packages / total 1852 - rw - r--r--
1 root staff 1895932 Mar 20 21: 51 cv2.cpython - 34 m.so
$ cd / usr / local / lib / python3.5 / site - packages /
$ sudo mv cv2.cpython - 35 m - arm - linux - gnueabihf.so cv2.so
$ cd~/.virtualenvs/cv / lib / python3.5 / site - packages /
$ ln - s / usr / local / lib / python3.5 / site - packages / cv2.so cv2.so
```

Python Kodu

```
import cv2
import numpy as np

def nothing(x):
   pass

import RPi.GPIO as GPIO
import time

import os
os.system("sudo pigpiod")
import pigpio
pi = pigpio.pi()
p = 1750
r = 1000
y = 1000
pi.set_servo_pulsewidth(4, p)
time.sleep(1)
pi.set_servo_pulsewidth(17, y)
time.sleep(1)

cap = cv2.VideoCapture(0)

while True:
   _, frame = cap.read()
   blur = cv2.blur(frame, (7, 7))
   hsv = cv2.cvtColor(blur, cv2.COLOR_BGR2HSV)

   lower_renk = np.array([2, 170, 71])
   upper_renk = np.array([12, 255, 200])
   mask = cv2.inRange(hsv, lower_renk, upper_renk)

   result = cv2.bitwise_and(frame, frame, mask = mask)
   cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[-2]
   center = None

if len(cnts) > 0:
     c = max(cnts, key = cv2.contourArea)
     ((x, y), radius) = cv2.minEnclosingCircle(c)
     M = cv2.moments(c)
     if M["m00"] == 0:
       M["m00"] = 1
```

```
     center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
     if radius > 1:
       cv2.circle(frame, (int(x), int(y)), int(radius + 3), (255, 0, 0), 2)
      # cv2.circle(frame, center, 3, (0, 0, 255), -1)
      # cv2.putText(frame, "centroid", (center[0] + 10, center[1]), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (0, 0, 255), 1)
      # cv2.putText(frame, "(" + str(center[0]) + "," + str(center[1]) + ")", (center[0] + 10, center[1] + 15), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (0, 0, 255), 1)

      h = 320 - int(center[0])
      if h > 250:
         p = p + 150
      elif h > 175:
         p = p + 100
      elif h > 80:
         p = p + 30
      elif h > 40:
         p = p + 10
      elif h < -250:
         p = p - 150
      elif h < -175:
         p = p - 100
      elif h < -80:
         p = p - 30
      elif h < -40:
         p = p - 10

if p > 2480:
         p = 2480
if p < 520:
         p = 520
       pi.set_servo_pulsewidth(4, p)
      # time.sleep(.1)

              m = 240 - int(center[1])

       if m > 200:
         r = r + 80
       elif m > 145:
         r = r + 50
       elif m > 70:
         r = r + 15
       elif m > 60:
         r = r + 8
       elif m < -200:
         r = r - 80
       elif m < -145:
         r = r - 50
       elif m < -70:
         r = r - 15
       elif m < -60:
         r = r - 8

       if r > 1700:
         r = 1700
       if r < 700:
         r = 700

       pi.set_servo_pulsewidth(17, r)
       # time.sleep(.1)
       print(h, p)
       print(m, y)

       cv2.imshow('blur', blur)
       cv2.imshow('mask', mask)
       cv2.imshow("result", result)
```

```
        cv2.imshow('frame', frame)

        k = cv2.waitKey(5) & 0xFF
        if k == 27:
            break

        pi.stop()
        os.system("sudo pkill pigpiod")
        cv2.waitKey(0)
        cap.release()
        cv2.destroyAllWindows()
```

Kaynak Kodu

Buradaki proje resimlerine ve kaynak koduna https://github.com/zelihaakin/hedeftakip/tree/master adresinden erişilebilir.

Nasıl Kullanılır

Kamera sistemin sabitlenmesi ve gerekli ayarların bir kez yapıldıktan sonraki kullanımlarda kolay kullanım amaçlanmıştır.

Konsola

```
python hedef.py
```

komutunun girilmesinin ardından sistem hedefi yakaladığı anda takibe başlar.

Proje Resimleri