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 : ATIŞ – HEDEF ANALİZ SİSTEMİ

TARİH :28.02.2018

GİRİŞ

Atış – Hedef Analiz Sistemi poligonlarda hedef kağıdının yanına gitmeden görüntü işleme yöntemiyle isabet analizini görebildiğimiz bir tasarımdır. Bu projede bir kamera sayesinde hedef kağıdının sabit bir açısından alınan görüntü Raspberry Pi'mize iletilerek görüntümüz işlenip çıktı olarak bize hedefe yapılan atışın merkeze uzaklığını diğer görüntüleri maskeleyerek vermektedir.

Bu projemizde Raspberry Pi, kamera ve kameramızı sabitlemek için tasarlanan bir platform kullanılmıştır. Görüntü işleme için Python ve Opencv kullanılırken Raspberry Pi için işletim sistemi olarak Raspbian Stretch kullanılmıştır.

Gerekli Donanım Bileşenleri

1. 1 adet Rasperry Pi

2. 1 adet Kamera

3. 1 adet Özel Tasarım Platform

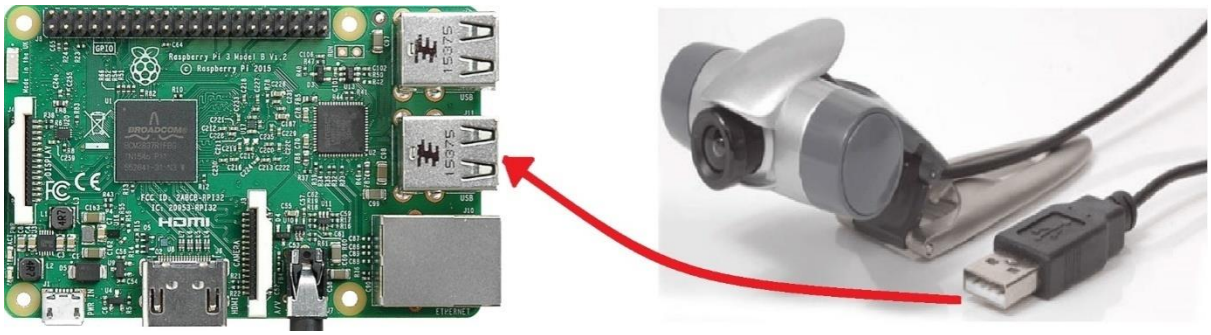
Gerekli Yazılım Bileşenleri

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. Platform:
   Kameramızı sabitlemek için kullandığımız bu bileşen istenilen konum ve kamera açısı için bizim tarafımızdan tasarlanmıştır.

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

```
$ sudo raspi - config
$ sudo reboot
$ sudo apt - get purge wolfram - engine
$ 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

```
#  Gerekli kütüphaneler dahil ediliyor

import cv2
import numpy as np from shutil
import copyfile
import time

# Kamera ile fotoğraf çekildi fotoğrafın kararmaması için program 1 sn bekletilir

cap = cv2.VideoCapture(0)
time.sleep(1)
_, img = cap.read()

# Çekilen görüntü kaydedilir

cv2.imwrite('/home/pi/Documents/goruntu_isleme/birlestirme/cekilen.jpg', img)

# Perspektif düzeltilir

rows, cols, ch = img.shape
pts1 = np.float32([[73, 131],[562, 122],[6, 479],[639, 479]])
pts2 = np.float32([[0, 0],[842, 0],[0, 595],[842, 595]])
matrix = cv2.getPerspectiveTransform(pts1, pts2)
M = cv2.warpPerspective(img, matrix, (842, 595))
frame = M

# Perspektifi düzeltilen görüntü kaydedilir

cv2.imwrite('/home/pi/Documents/goruntu_isleme/birlestirme/perspective.jpg', frame)

# Görüntü maskelemeden önce gerekli çevirim işlemleri yapılır

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
bgr = cv2.cvtColor(gray, cv2.COLOR_GRAY2BGR)
hsv = cv2.cvtColor(bgr, cv2.COLOR_BGR2HSV)

# Maskeleme yapılacak alan aralığın alt ve üst hsv değerleri kaydedilir

lower_black = np.array([0, 0, 0])
upper_black = np.array([90, 165, 100])

# Maskeleme yapılır

mask = cv2.inRange(hsv, lower_black, upper_black)

# Maskeleme kaydedilir

cv2.imwrite('/home/pi/Documents/goruntu_isleme/birlestirme/mask.jpg', mask)

# Maskelenen görüntülerin şekilleri bulunur

cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[-2]
```

```
center = None

# Şekillerin ağılık merkezi etrafına daire çizilir ve merkezden ne kadar uzakta olduğu hesa
planıp perspektif ile düzeltilen görünütünün üzerine yazdırılır

if len(cnts) > 1:
 c = max(cnts, key = cv2.contourArea)
((x, y), radius) = cv2.minEnclosingCircle(c)
M = cv2.moments(c)
center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
if radius > 2:
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, cent
er[1] + 15), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (0, 0, 255), 1)
a = center[0] - 421
b = center[1] - 298
c = (a * a + b * b) * * 0.5
cv2.putText(frame, "(" + str(a) + "," + str(b) + ")", (760, 575), cv2.FONT_HERSHEY_SIMPLEX,
 0.4, (0, 0, 255), 1)
cv2.putText(frame, "(" + str(c) + ")", (760, 589), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (0, 0, 25
5), 1)

# İlk çekilen fotoğraf, perspektif ile düzeltilen fotoğraf ve üzerine bilgiler yazdırılan f
otoğraf gösterilir

cv2.imshow('frame', frame)
cv2.imshow('mask', mask)
cv2.imshow('input', img)
cv2.imwrite('/home/pi/Documents/goruntu_isleme/birlestirme/frame.jpg', frame)

# Bir tuşa basılması beklenir

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Kaynak Kodu

Buradaki proje resimlerine ve kaynak koduna https://github.com/ozdemiryemre/RaspberryP-/tree/masteradresinden erişilebilir.

Nasıl Kullanılır

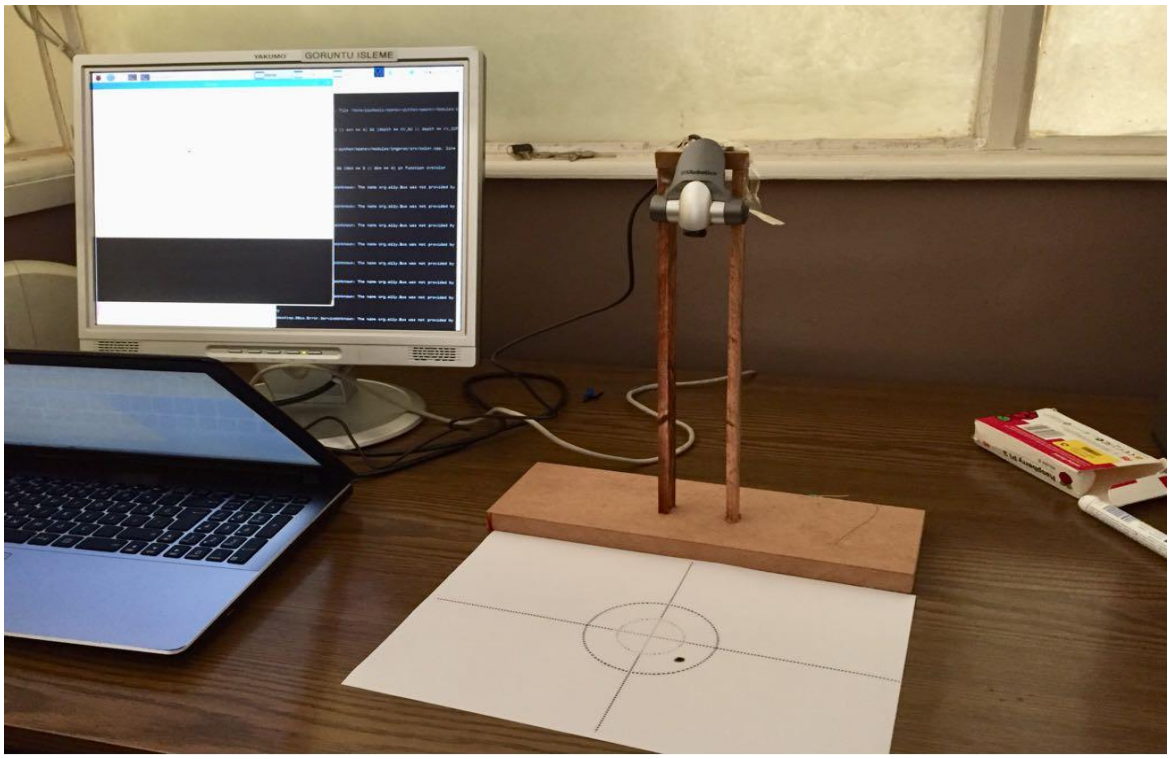
Kamera sistemin sabitlenmesi ve gerekli ayarların bir kez yapıldıktan sonraki kullanımlarda kolay kullanım amaçlanmıştır.

Konsola

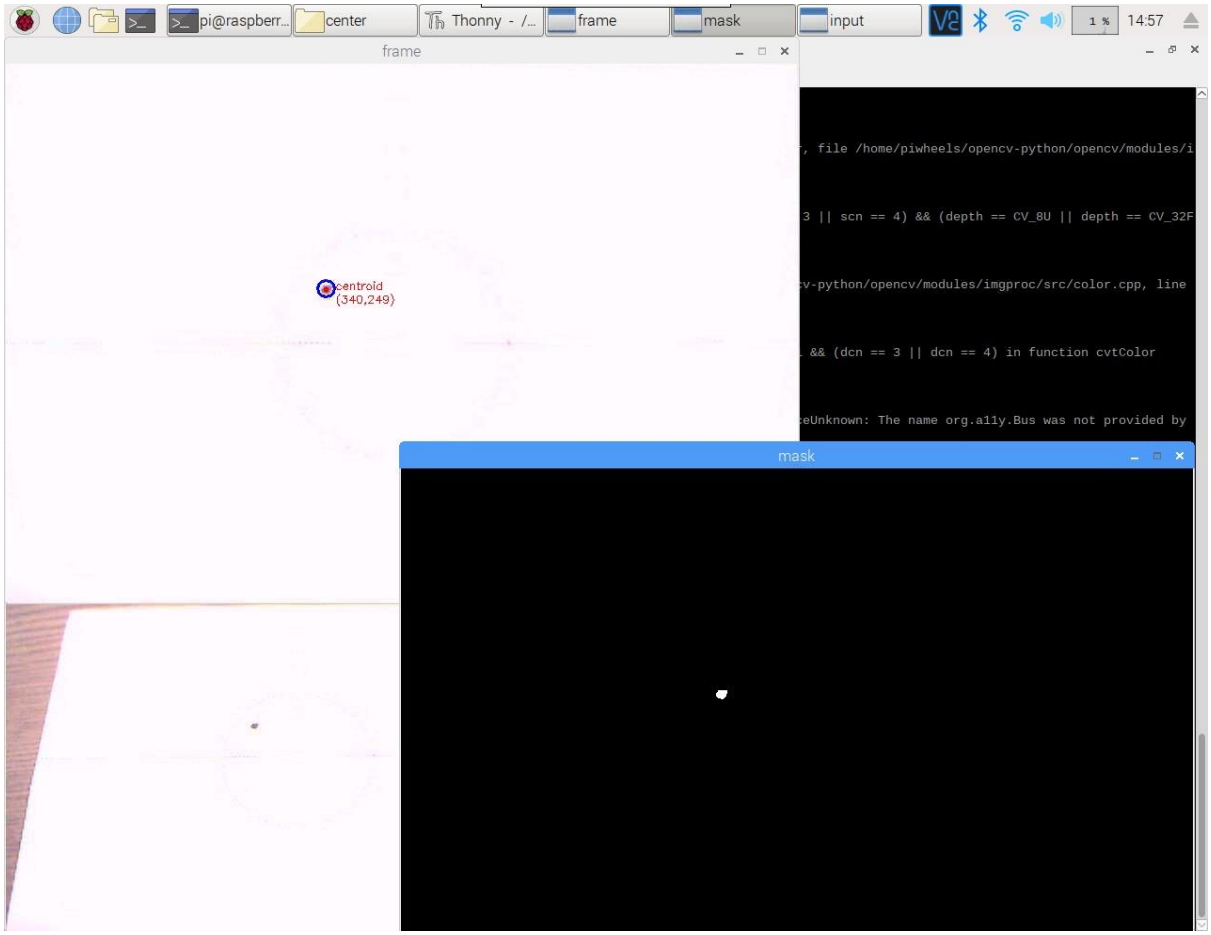
```
python hedef.py
```

komutunun girilmesinin ardından sistem otomatik fotoğrafı alıp gerekli işlemleri yapar ve çıktıyı ekrana gönderir.

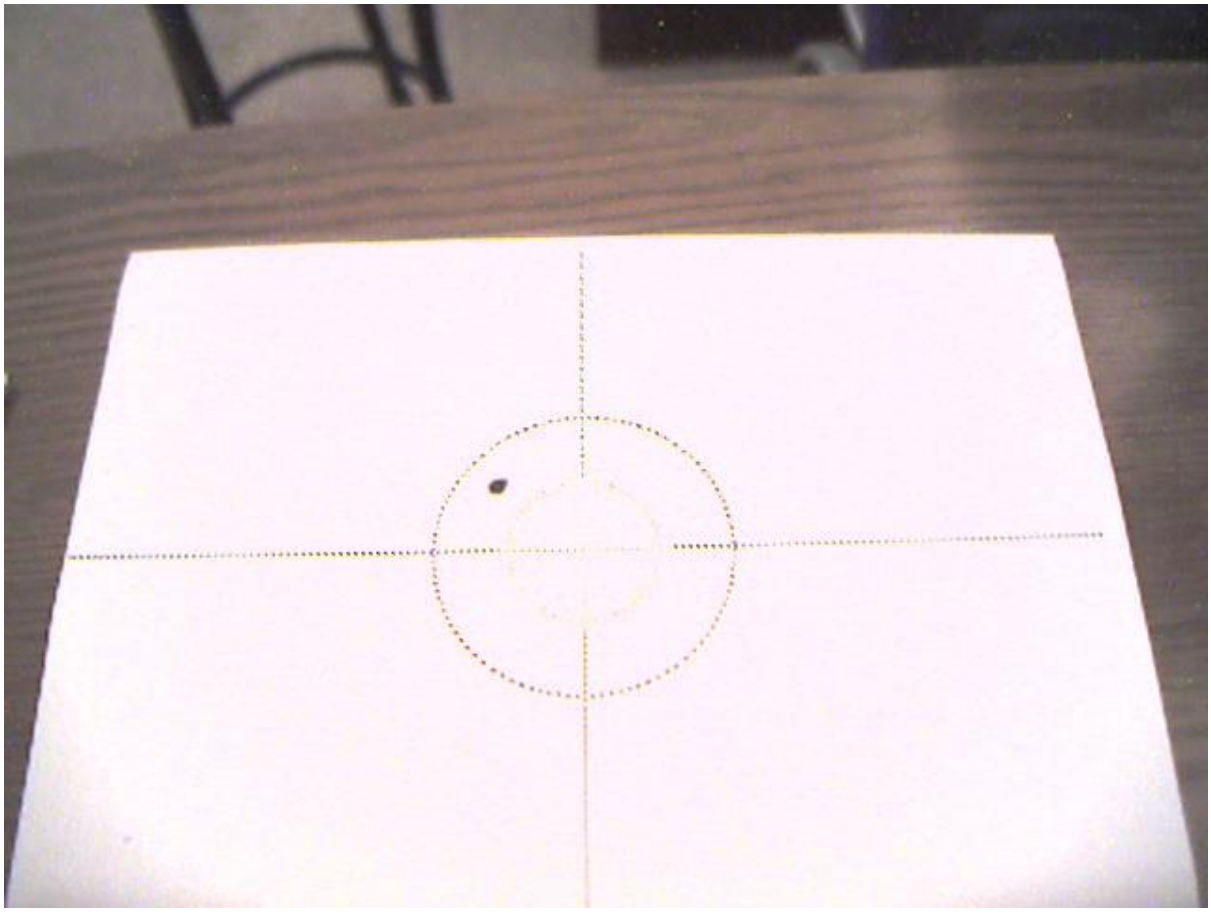
Proje Resimleri



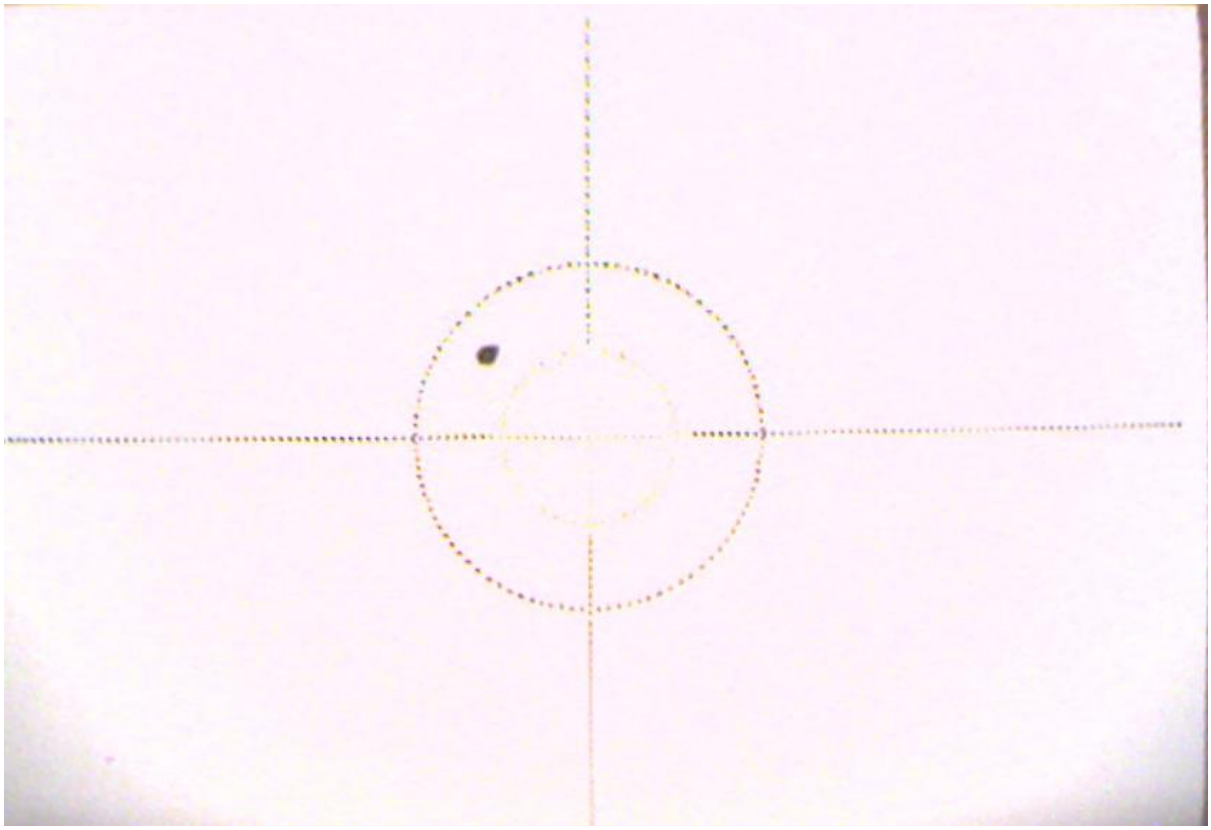
Ekran Görüntüleri



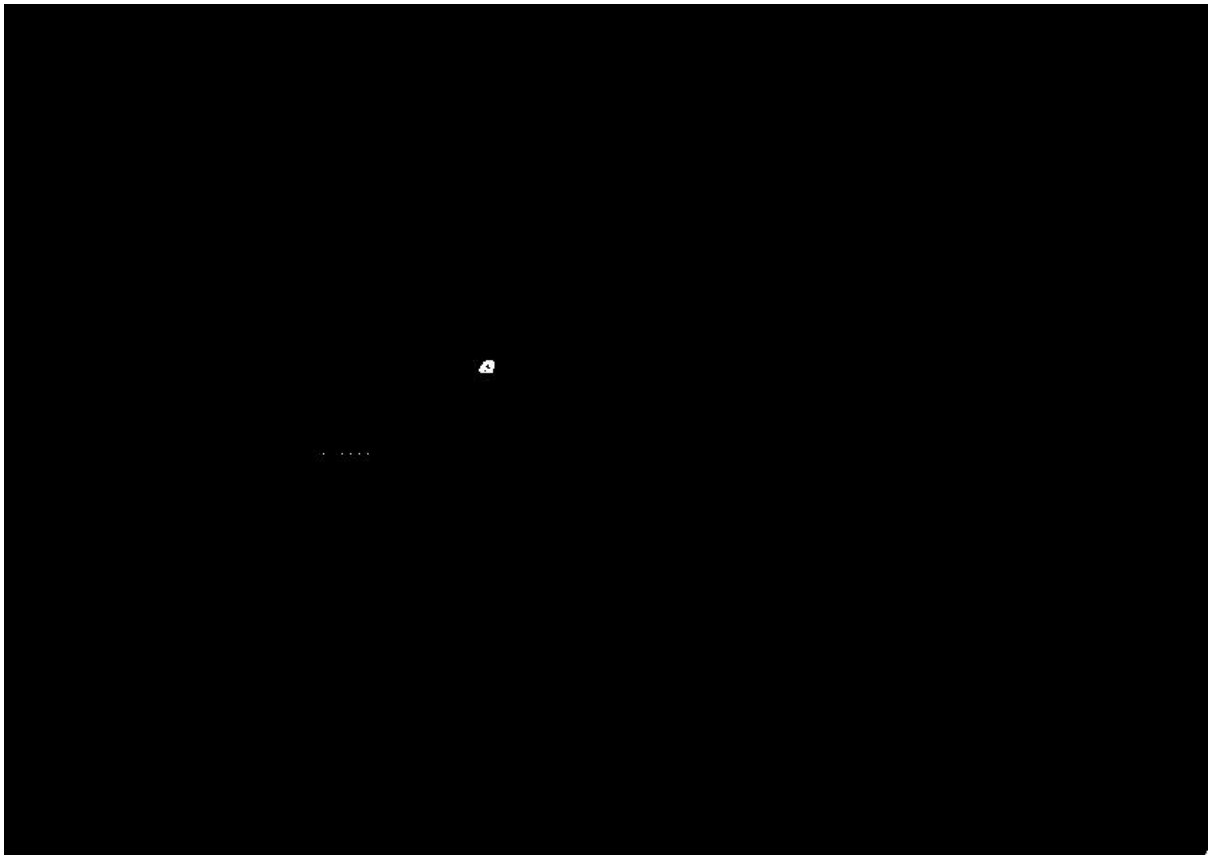
RESİM 1: Raspberry Pi'mizden programın çalışması sırasında alınan bir görüntü



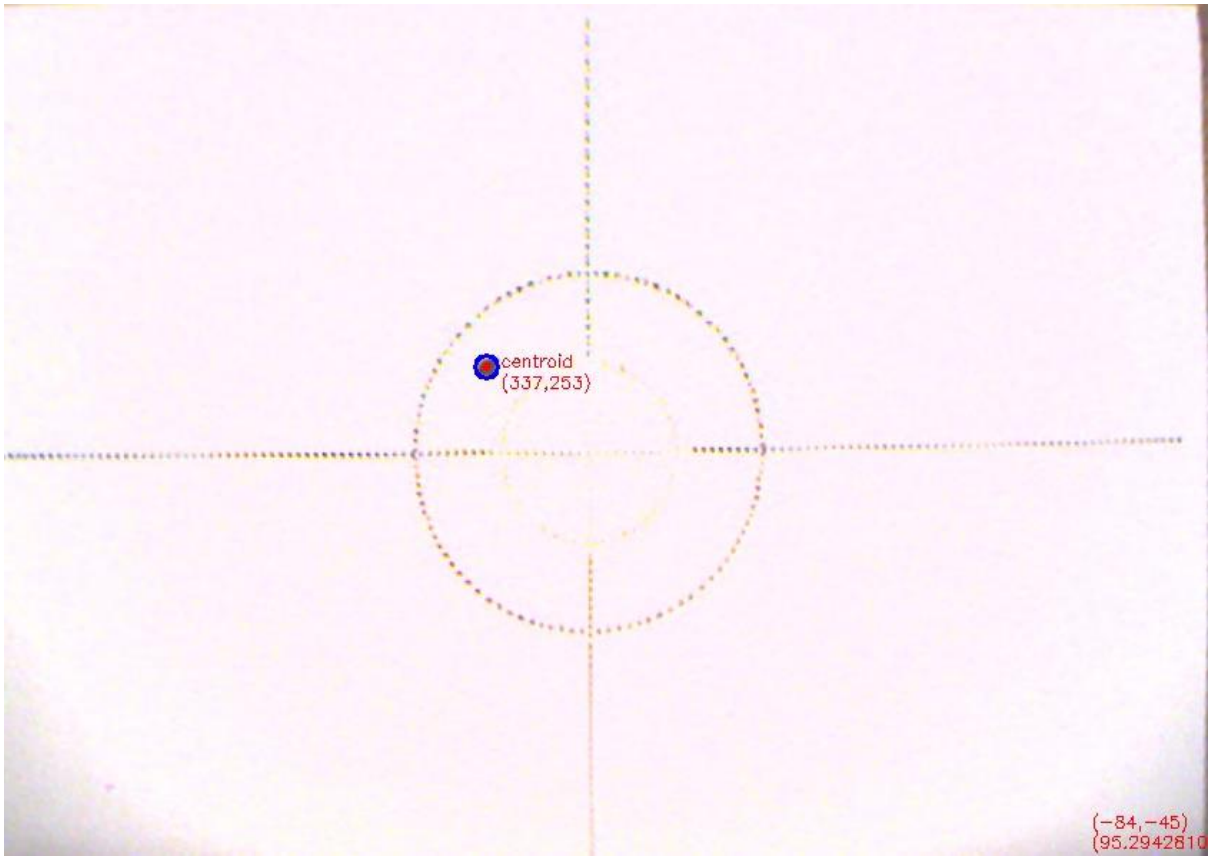
RESİM 2: Raspberry Pi’mize gelen kamera görüntüsü



RESİM 3: Kameradan alınmış perspektifi düzeltilmiş görüntü



RESİM 4: Perspektifi düzeltilmiş maskelenmiş görüntü



RESİM 5: Atışın merkeze olan uzaklığını çıktı olarak aldığımız görüntü

Versiyon : 2.0

Bu versiyonda projemizi atış sayısını artırarak analizi ve de puanlama sistemini getirerek geliştirdik. Bu versiyonumuz için;

Versiyon 2.0 Python Kodu:

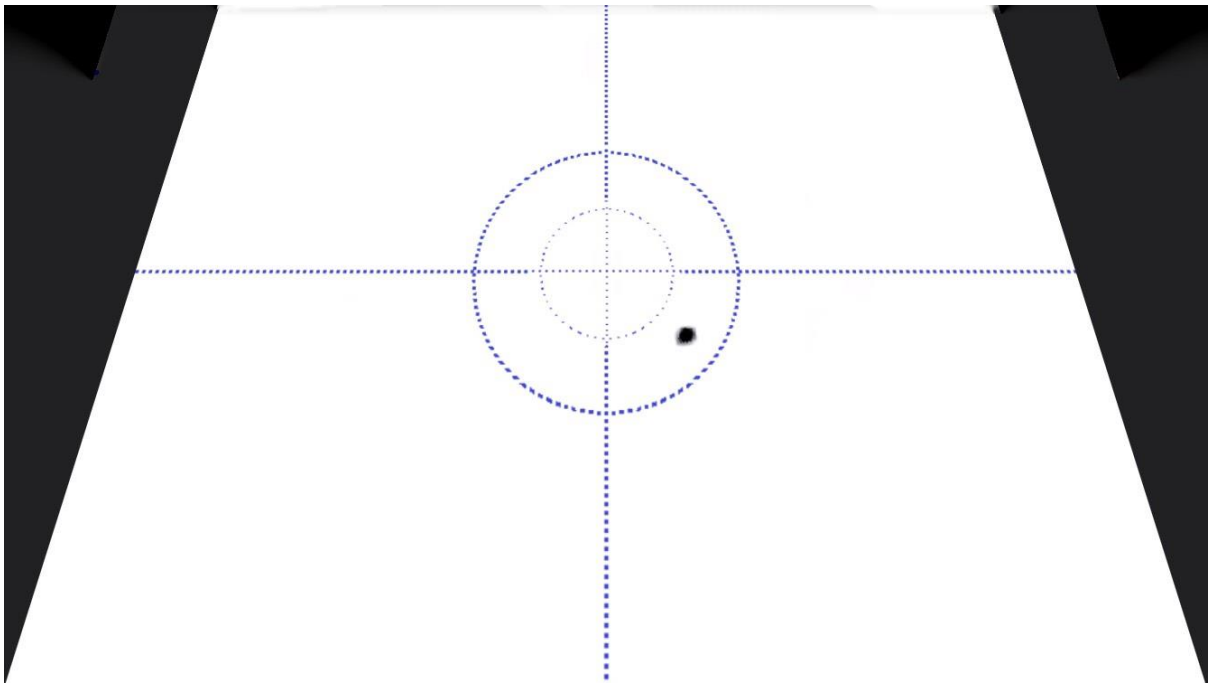
```
#15.04.2018

import cv2
import numpy as np
from shutil import copyfile
import time

lcnts = 0
lmermi = 0

frame = cv2.imread('frame.png', cv2.IMREAD_COLOR)

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
bgr = cv2.cvtColor(gray, cv2.COLOR_GRAY2BGR)
hsv = cv2.cvtColor(bgr, cv2.COLOR_BGR2HSV)

lower_black = np.array([0, 0, 0])
upper_black = np.array([50, 50, 50])

mask = cv2.inRange(hsv, lower_black, upper_black)

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)
lcnts = 1
lmermi = 1

frame2 = cv2.imread('frame2.png', cv2.IMREAD_COLOR)

gray2 = cv2.cvtColor(frame2, cv2.COLOR_BGR2GRAY)
bgr2 = cv2.cvtColor(gray2, cv2.COLOR_GRAY2BGR)
hsv2 = cv2.cvtColor(bgr2, cv2.COLOR_BGR2HSV)

 mask2 = cv2.inRange(hsv2, lower_black, upper_black)

cnts2 = cv2.findContours(mask2.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[-2]
center = None

lcnts2 = len(cnts2)

if (cnts2 != cnts):
print("emre")
if (lcnts2 == lcnts):
print("emre")
c = cnts[0]
c2 = cnts2[0]
lmermi = 2
elif(lcnts2 == 1):
c = max(cnts2, key = cv2.contourArea)
lmermi = 1
elif(lcnts2 == 2):
c = cnts2[0]
c2 = cnts2[1]
lmermi = 2
frame3 = cv2.imread('frame3.png', cv2.IMREAD_COLOR)

```

```
gray3 = cv2.cvtColor(frame3, cv2.COLOR_BGR2GRAY)
bgr3 = cv2.cvtColor(gray3, cv2.COLOR_GRAY2BGR)
hsv3 = cv2.cvtColor(bgr3, cv2.COLOR_BGR2HSV)

mask3 = cv2.inRange(hsv3, lower_black, upper_black)

cnts3 = cv2.findContours(mask3.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[-2]
center3 = None

lcnts3 = len(cnts3)

if (cnts3 != cnts2):
if (lcnts3 == lcnts2):
if (lcnts3 == 1):
c3 = max(cnts3, key = cv2.contourArea)
lmermi = lmermi + 1
if (lcnts3 == 2):
c3 = max(cnts3, key = cv2.contourArea)
lmermi = 3
elif(len(cnts3) == 1):
c = max(cnts3, key = cv2.contourArea)
lmermi = 1
elif(len(cnts3) == 2):
if (lmermi == 1):
c = cnts3[0]
c2 = cnts3[1]
          lmermi = 2
elif(lmermi == 2):
lmermi = 3
c3 = cnts3[0]

elif(len(cnts3) == 3):
c = cnts3[0]
c2 = cnts3[1]
c3 = cnts3[2]
lmermi = 3

if (lmermi == 0):
a = 842
b = 596
z = 956
       #
       a2 = 842
       b2 = 596
       z2 = 956
       #
       a3 = 842
       b3 = 596
       z3 = 956

   if (lmermi == 1):
       ((x, y), radius) = cv2.minEnclosingCircle(c)
       M = cv2.moments(c)
       center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
       x = center[0]
       y = center[1]
       a = x - 421
       b = y - 298
       z = (a * a + b * b) * * 0.5
       cv2.circle(frame3, (int(x), int(y)), int(radius + 23), (255, 0, 0), 2)
       cv2.circle(frame3, center, 3, (0, 0, 255), -1)
       cv2.putText(frame3, "centroid", (center[0] + 10, center[1]), cv2.FONT_HERSHEY_SIMPL
   EX, 0.4, (0, 0, 255), 1)
       cv2.putText(frame3, "(" + str(center[0]) + "," + str(center[1]) + ")", (center[0] +
    10, center[1] + 15), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (0, 0, 255), 1)
       #
```

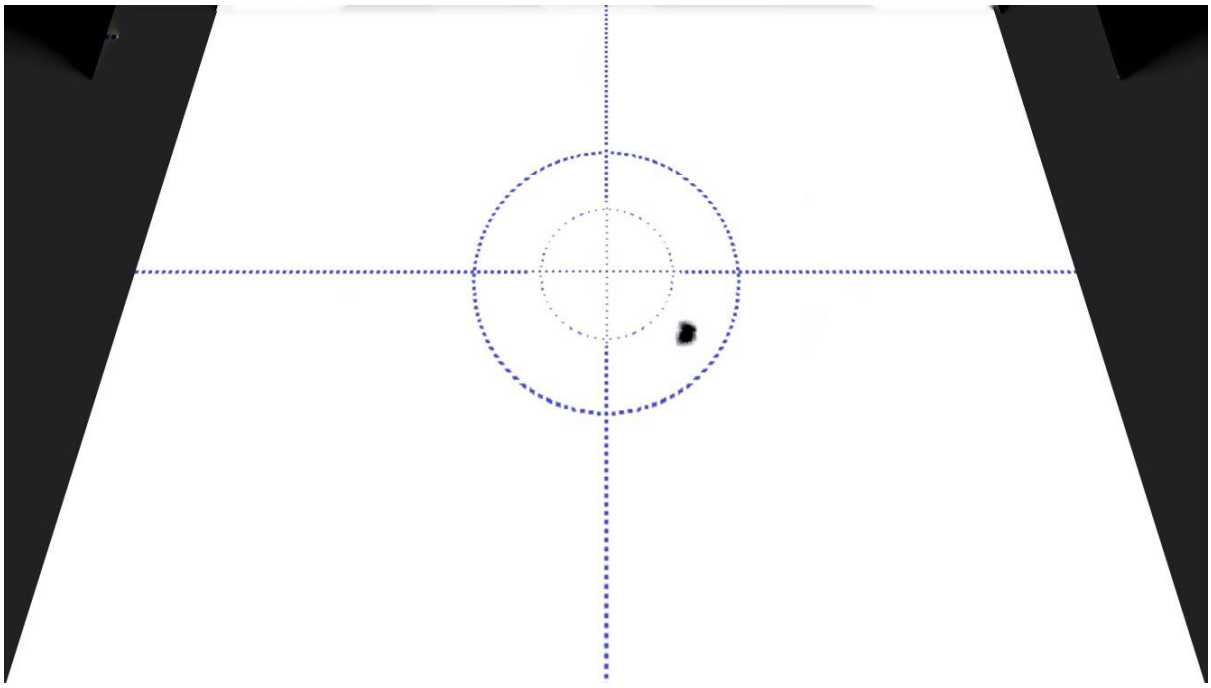
```
        a2 = 842
        b2 = 596
        z2 = 956
        #
        a3 = 842
        b3 = 596
        z3 = 956

    if (lmermi == 2):
        ((x, y), radius) = cv2.minEnclosingCircle(c)
        M = cv2.moments(c)
        center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
        x = center[0]
        y = center[1]
        a = x - 421
        b = y - 298
        z = (a * a + b * b) * * 0.5
        cv2.circle(frame3, (int(x), int(y)), int(radius + 23), (255, 0, 0), 2)
        cv2.circle(frame3, center, 3, (0, 0, 255), -1)
        cv2.putText(frame3, "centroid", (center[0] + 10, center[1]), cv2.FONT_HERSHEY_SIMPL
    EX, 0.4, (0, 0, 255), 1)
        cv2.putText(frame3, "(" + str(center[0]) + "," + str(center[1]) + ")", (center[0] +
     10, center[1] + 15), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (0, 0, 255), 1)
        #
        ((x, y), radius) = cv2.minEnclosingCircle(c2)
        M2 = cv2.moments(c2)
        center2 = (int(M2["m10"] / M["m00"]), int(M2["m01"] / M["m00"]))
        x2 = center[0]
        y2 = center[1]
        a2 = x2 - 421
        b2 = y2 - 298
        z2 = (a2 * a2 + b2 * b2) * * 0.5
        cv2.circle(frame3, (int(x2), int(y2)), int(radius + 23), (255, 0, 0), 2)
        cv2.circle(frame3, center2, 3, (0, 0, 255), -1)
        cv2.putText(frame3, "centroid", (center[0] + 10, center[1]), cv2.FONT_HERSHEY_SIMPL
    EX, 0.4, (0, 0, 255), 1)
        cv2.putText(frame3, "(" + str(center[0]) + "," + str(center[1]) + ")", (center[0] +
     10, center[1] + 15), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (0, 0, 255), 1)
        #
        a3 = 842
        b3 = 594
        z3 = 956

    if (lmermi == 3):
        ((x, y), radius) = cv2.minEnclosingCircle(c)
        M = cv2.moments(c)
        center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
        x = center[0]
        y = center[1]
        a = x - 421
        b = y - 298
        z = (a * a + b * b) * * 0.5
        cv2.circle(frame3, (int(x), int(y)), int(radius + 23), (255, 0, 0), 2)
        cv2.circle(frame3, center, 3, (0, 0, 255), -1)
        cv2.putText(frame3, "centroid", (center[0] + 10, center[1]), cv2.FONT_HERSHEY_SIMPL
    EX, 0.4, (0, 0, 255), 1)
        cv2.putText(frame3, "(" + str(center[0]) + "," + str(center[1]) + ")", (center[0] +
     10, center[1] + 15), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (0, 0, 255), 1)
        #
        ((x, y), radius) = cv2.minEnclosingCircle(c2)
        M2 = cv2.moments(c2)
        center2 = (int(M2["m10"] / M2["m00"]), int(M2["m01"] / M2["m00"]))
        x2 = center2[0]
        y2 = center2[1]
        a2 = x2 - 421
        b2 = y2 - 298
```

```
        z2 = (a2 * a2 + b2 * b2) * * 0.5
        cv2.circle(frame3, (int(x2), int(y2)), int(radius + 23), (255, 0, 0), 2)
        cv2.circle(frame3, center2, 3, (0, 0, 255), -1)
        cv2.putText(frame3, "centroid", (center2[0] + 10, center2[1]), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (0, 0, 255), 1)
        cv2.putText(frame3, "(" + str(center2[0]) + "," + str(center2[1]) + ")", (center2[0] + 10, center2[1] + 15), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (0, 0, 255), 1)
        #
        ((x, y), radius) = cv2.minEnclosingCircle(c3)
        M3 = cv2.moments(c3)
        center3 = (int(M3["m10"] / M3["m00"]), int(M3["m01"] / M3["m00"]))
        x3 = center3[0]
        y3 = center3[1]
        a3 = x3 - 421
        b3 = y3 - 298
        z3 = (a3 * a3 + b3 * b3) * * 0.5
        cv2.circle(frame3, (int(x3), int(y3)), int(radius + 23), (255, 0, 0), 2)
        cv2.circle(frame3, center3, 3, (0, 0, 255), -1)
        cv2.putText(frame3, "centroid", (center3[0] + 10, center3[1]), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (0, 0, 255), 1)
        cv2.putText(frame3, "(" + str(center3[0]) + "," + str(center3[1]) + ")", (center3[0] + 10, center3[1] + 15), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (0, 0, 255), 1)


    H = 100 - (z + z2 + z3) / 10

    cv2.putText(frame3, "(" + str(H) + ")", (760, 589), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (0, 0, 255), 1)

    cv2.imshow('frame3', frame3)

    print(z)

    print(z2)

    print(z3)


    cv2.waitKey(0)

    cv2.destroyAllWindows()
```

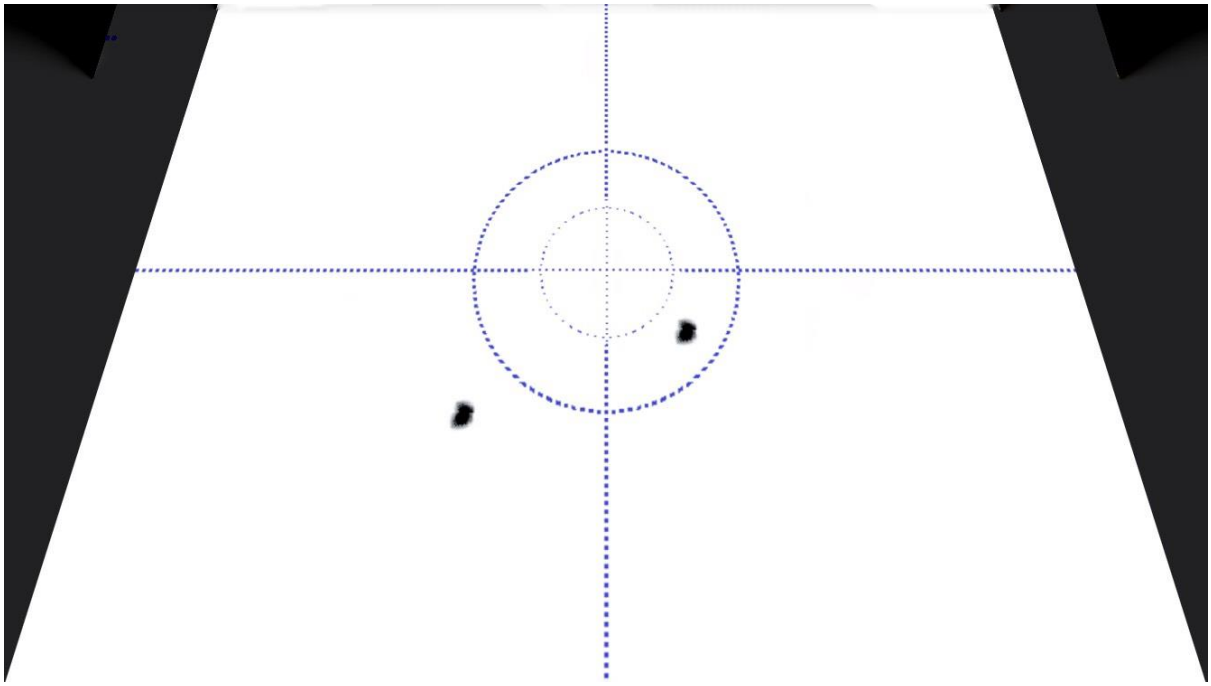
Versiyon 2.0 Ekran Resimleri



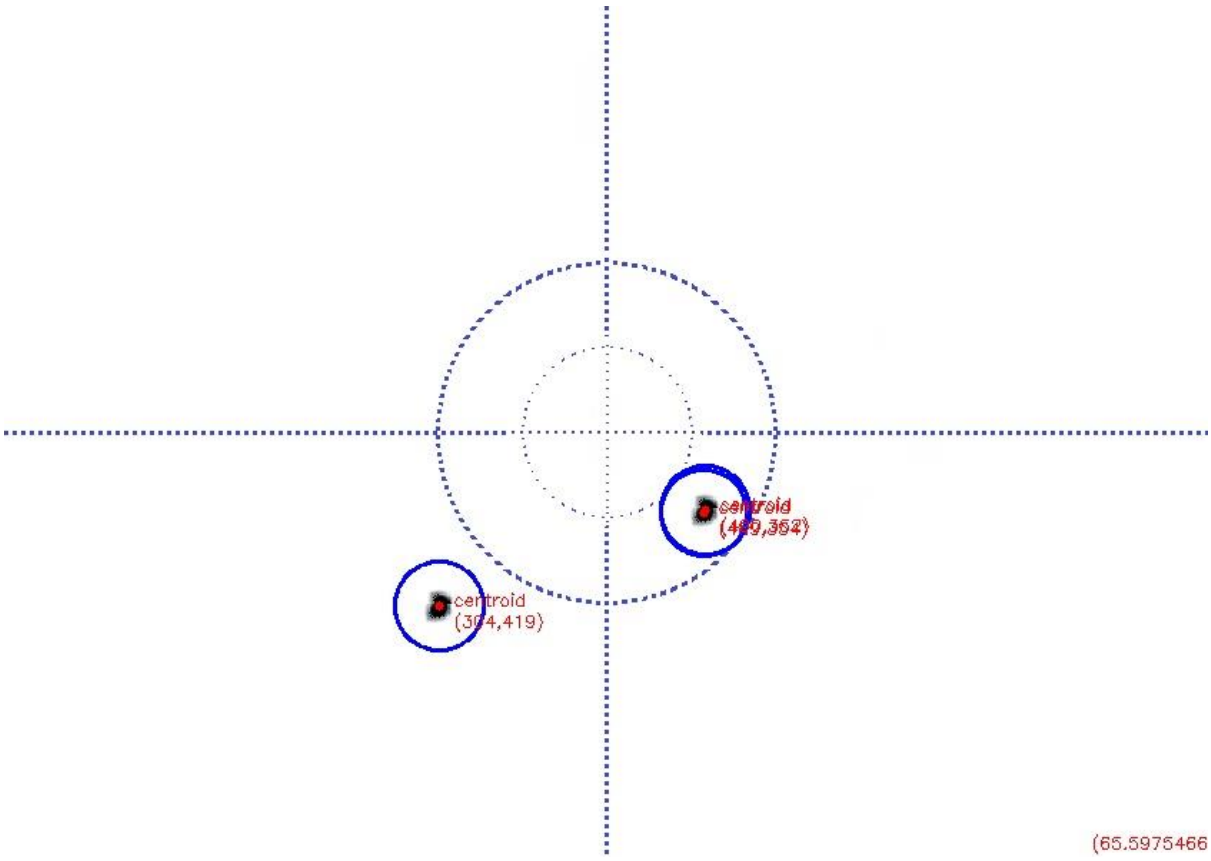
Versiyon 2.0 Ekran Görüntüsü 1 : Birinci Atış



Versiyon 2.0 Ekran Görüntüsü 2 : Birinci Atışın Yakınındaki İkinci Atış



Versiyon 2.0 Ekran Görüntüsü 3 : Üçüncü Atış



Versiyon 2.0 Ekran Görüntüsü 4 : Önceki 3 resime göre oluşturulan çıktı. Üç atışın merkeze göre konumu ve sağ alt köşede 100'lük sisteme göre hassas bir şekilde puanlar.

Öneriler

İlerleyen aşamalarda atışların aritmetik ortalama standart sapma gibi istatistiksel yöntemler tarafında değerlendirilip kaç atış yapıldığının silah sesinin algılanarak hesaplanıp ve algılanan silah sesi ile sistemin buton, tuş vb... bir aktif etme girişini ihtiyaç duymadan otomatik çalışabilir.