1. İndirilecek Paketler:
   1. opencv-python : Python’da görüntü işleme yapmak için kullanılır.
   2. opencv-contrib-python: Opencv’de bulunan trackerları kullanmak için kullanılır.
   3. numpy: Python’da 3 ve 2 boyutlu dizilerde işlem yapmak için kullanılır.
   4. scipy.spatial.distance : Pythonda mesafe ölçmek için kullanılır.

İndirirken terminalde aşağıdaki komut satırları çalıştırılır.

Opencv-python : pip install opencv-python

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

numpy: pip install numpy

scipy.spatial.distance: pip install scipy.spatial.distance

1. Dahile edilecek Paketler:

İndirilen veya indirilmesine gerek olunmayan paketler aşağıdaki gibi Python dosyasında dahil edilir.

import cv2  
import numpy as np  
from collections import deque  
import math  
import pickle  
import os.path as os  
import scipy.spatial.distance as dist  
from time import time

1. Mouse Event’ini yakalamak:
   1. def click\_event(event, mouseX, mouseY, flags, params) ile fonksiyon oluşturulur.
   2. Mouse’nin tıklama eventinin sol tuşuna tıklanması için cv2.EVENT\_LBUTTONDOWN, sağ tuşuna tıklanması için cv2.EVENT\_RBUTTONDOWN olup olmadığı sorgulanır.
   3. Yapılacak işlemler bu fonksiyon içine de yazılabilir ya da kontrol kullanılarak True veya False yapılarak kontrol edilir.
   4. cv2.setMouseCallback("windowName", click\_event) ile Mouse hareketlerinin kullanılmak istediği ekran için çalıştırılır.
2. Kamera Kalibrasyonu: Kamera kalibrasyonu, görüntü bozukluğunu düzeltmek için gerekli bazı verilere ulaşmak için yapılır. Bu projede kamera matrisi sayesinde kamera fov değerini bulmak için kullanılmıştır.  
   Kamera Kalibrasyonu Yapımı:
   1. Satır ve Sütun sayısını bildiğimiz bir satranç tahtası görseli çıkartılır.
   2. Satranç tahtası görseli ile kamera karşısına geçilir.
   3. findChessboardCorners() metodu ile giriş olarak vereceğimiz görüntü üzerindeki köşeler hesaplanır.
      1. Kalibrasyon işlemi esnasında drawChessboardCorners() metodunu kullanarak tespit edilen noktaları görüntü üzerinde işaretleyebiliriz
   4. Eğer belirttiğimiz satır ve sütün sayısınca köşe doğru olarak tespit edilmişse cornerSubPix() metodu ile köşelerin veya radyal sırt noktalarının alt pikselinin, doğru konumunu bulmak için kendi içerisinde yineler.
   5. Elde ettiğimiz bu noktaları ve değerleri kalibrasyon verisi olarak saklayabiliriz.
   6. Bu kalibrasyon işlemini doğruluğunu arttırmak için satranç tahtası görüntüsünü farklı açılardan ve yerlerden olacak şekilde kendi belirleyeceğimiz bir iterasyon adedi boyunca tekrarlarız.
   7. calibrateCamera() fonksiyonu sayesinde kamera kalibrasyonu değerlerini alabiliriz
3. Tracker’ların kullanımı:
4. Kullanılmak istenilen tracker belirlenir ve trackerName kısmına kullanılmak istenen trackerın adı yazılır.
5. Eğer opencv-contrib-python paketi indirilmezse tracker çalışmaz.
6. Trackbar oluşturmak: Trackbarları program çalışırken belli ayrıntıları ayarlamak veya fonksiyondaki parametleri seçmek için kullanılır.
7. Trackbar oluştururken:
   1. cv2.namedWindow("windowName") ile oluşturulacak ekranın ismi veriliyor.
   2. cv2.resizeWindow("windowName ", width, height) oluşturulacak ekranın boyutları belirleniyor.
   3. cv2.createTrackbar("trackbarName", " windowName ", default value, max value, callback) ile trackbarlar belirleniyor. Belirlenecek trackbarın ismi, ekranın ismi, varsayılan değer, maksimum alabileceği değer ve geri dönüş değeri verilir.
   4. Geri dönüş değeri istenmiyorsa koda “def empty(a): pass” şekilde fonksiyon eklenir ve callback yerine empty yazılır.
8. Trackbarları kullanırken:
9. deger = cv2.getTrackbarPos("trackbarName ", " windowName ") ile trackbarlardan gelen değer değişkene atılır ve istenilen yerde kullanılır.
10. Renk ile Tespit:
    1. blurred= cv2.GaussianBlur(img, (15,15), 0) komutu ile resimdeki renk ayrımlarını daha kolay algılamak ve detayları azaltmak için resme blur atılıyor
    2. blurredHsv=cv2.cvtColor(blurred, cv2.COLOR\_BGR2HSV) renkleri ayrımını daha iyi yapabilmek için HSV formatına çevriliyor (Hue Saturation Value)
    3. mask=cv2.inRange(blurredHsv,lowerColor,upperColor) verilen renk aralığında resimden bir maske yapıyor
    4. mask=cv2.erode(mask,None,iterations=4)

mask=cv2.dilate(mask,None,iterations=4) Maskedeki resim hatasından dolayı oluşan noktaları silmek için erode işlemini yapıyor ve eski haline getirmek için dilate işlemini yapıyor

* 1. (contours,\_)=cv2.findContours(mask, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE) maskeden konturları buluyor
  2. if len(contours) > 0 and not track:

for cs in contours:

rect=cv2.boundingRect(cs)

x, y, w, h = int(bbox[0]), int(bbox[1]), int(bbox[2]), int(bbox[3])

cv2.rectangle(img, (x, y), ((x + w), (y + h)), (255, 0, 255), 3, 1)

oluşan kontur varsa konturların etrafına bir kutu çiziyor ve bu kutuyu resme yazılıyor

1. Cascade ile Tespit:
2. Gereken Program: Cascade Traner Gui

İndirme linki: https://amin-ahmadi.com/cascade-trainer-gui/

1. Dataset Oluşturmak: Cascade oluştururken önce bir klasör açılır bu klasörün içine n ve p isimli 2 klasör açılır. Bu klasörler negatif ve pozitif fotoğrafları koymak içindir. Negatif fotoğraflar tespit edilmek istenen nesne dışındaki fotoğraflar, pozitif fotoğraflar ise tespit edilmek istenen nesnenin görüntüleridir. Daha sonra nesne için pozitif fotoğraf ve negatif fotoğraflar toplanarak gerekli klasörlere yerleştirilir.
2. Cascade Traner Gui ile Cascade Oluşturmak:
   1. Cascade oluşturduktan sonra klasör içinde classifier dosyası oluşturulur ve onun içinde bulunan cascade xml dosyası oluşturulan cascadedir.
   2. Train Sayfasından içinde n ve p klasörlerinin bulunduğu klasör seçilir. Positive Image Usage seçeneği seçilir ve n klasöründe bulunan fotoğraf sayısı Negative Input Count kısmına girilir.
   3. Common Sayfasında Number of Stages sayısı seçilir önerilen 15tir. Öğrenim basamaklarıdır. Sayı arttıkça öğrenmenin alacağı zaman da artar.
   4. Cascade Sayfasından Sample Width ve Sample Height fotoğraflara göre ayarlanır.
   5. Hepsi ayarlandıktan sonra Start’a tıklanarak cascade oluşturmaya başlanır.
3. Oluşturulan Cascadenin kullanımı:
4. cascade.xml dosyası proje klasörüne dahil edilir
5. cascade = cv2.CascadeClassifier("cascade.xml") ile cascade dosyası verilir.
6. nesne = cascade.detectMultiScale(frame, scaleVal, neighbor) ile tespit edilen nesne nesne değişkenine atılar.
7. scaleVal ve neighbor değişkenleri elle ile de girilebilir trackbarlardan da alınabilir.
8. Yolo ile Tespit:
9. Eğitimin yapılacağı ortam: GoogleColab
10. Görüntüleri labellamak için kullanılan araç: LabelImg

İndirme Linki: <https://tzutalin.github.io/labelImg/>

1. Darkneti indirme linki: https://github.com/AlexeyAB/darknet.git
2. Datasetini optimize etmek için pre-trained .weight dosyaları:

yolov4.cfg, yolov4-custom.cfg için <https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.conv.137>

yolov4-tiny.cfg, yolov4-tiny-3l.cfg, yolov4-tiny-custom.cfg için <https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v4_pre/yolov4-tiny.conv.29>

yolov3.cfg, yolov3-spp.cfg için <https://pjreddie.com/media/files/darknet53.conv.74>

yolov3-tiny-prn.cfg , yolov3-tiny.cfg için <https://drive.google.com/file/d/18v36esoXCh-PsOKwyP2GWrpYDptDY8Zf/view?usp=sharing>

1. Gerekli cfg dosyaları darkneti indirdiğiniz zaman darknet/cfg içerisinde bulabilirsiniz
2. Labelled\_data.data, train.txt ve text.txt dosyalarını oluşturmak için çalıştırılacak creating-files-data-and-name.py ve creating-train-and-test-txt-files.py dosyalarını indirmek için link : <https://github.com/jakkcoder/training_yolo_custom_object_detection_files>
   1. Labelled\_data.data kendiniz oluşturmak isterseniz. Bir txt dosyasına

classes = Oluşturulacak Sınıf Sayısı

train = train.txt Dosyasının Yeri

valid = test.txt Dosyasının Yeri

names = Nesne İsimlerinin .names Uzantılı Dosyasının Yeri

backup = Eğitilirken Oluşturulacak Yedek Dosyaların Yeri

Yazılır ve dosya farklı kaydet ile .data şeklinde kaydedilir

Örnek bir .data dosyası

classes = 2

train = data/train.txt

valid = data/test.txt

names = data/obj.names

backup = backup/

* 1. train ve test .txt dosyalarını kendiniz oluşturmak isterseniz. Bir txt dosyasına

resim dosyasının bulunacağı yer/resim dosyasının adı

şeklinde resimler tek tek yazılır. train ve test .txt dosyaları için ayrı ayrı bu işlemi yaparsınız. Train ve Test dosyalarında kullanılan resimlerin farklı olması gerekmektedir. Train ve Test dosyalarında kullanılacak resim oranı %80 e %20 oranı önerilmektedir.

Örnek bir train.txt dosyası

data/obj/img1.jpg

data/obj/img2.jpg

data/obj/img3.jpg

1. Dataseti Oluşturma:
2. Tespit edilmesi istenen nesne için fotoğraf toplanır
3. Toplanan nesneler yolo formatı ayarlanarak label yapılır. Label yaparken programda fotoğraf açılır ve fotoğrafta nesnenin tamamını kaplayacak şekilde olan en küçük dikdörtgen çizilir. Bu çizilen alanda bulunan nesnenin adı yazılır ve kaydedilir. Program yolo için fotoğrafın isminde bir txt dosyası oluşturur.
4. Nesnelere verilen isimler txt dosyasına kaydedilir. Ayrıca bu dosya.names uzantısı ile de kaydedilir.
5. Google Drive’da bir klasör oluşturulur. Oluşturulan klasörün içine datasetini yüklemek için custom\_data klasörü açılır ve labelladığımız fotoğrafları, .txt dosyalarını, .names ve .data uzantılı dosyaları koyulur.
   1. Daha hızlı yüklemek isterseniz dosyaları bir klasörde toplayıp sıkıştırıp (.zipleyip) drive yükleyebilirsiniz. Google colabda drive hesabınızı bağladıktan sonra !unzip '/content/drive/MyDrive/”zipDosyasınınKonumu”' -d '/content/drive/MyDrive/”ÇıkarılacakYer” '  
      komutunu vererek zip dosyasını çıkartabilirsiniz
6. Datasetini Eğitmek:
7. Google Colab’da yeni bir defter açılır ve drive hesabına bağlanılır. Drive hesabına bağlanmak için aşağıdaki komut çalıştırılır.

from google.colab import **drive** drive.mount("/content/drive")

1. Komut çalıştıktan sonra drive hesabına bağlanmak için verilen linke tıklanarak hesap seçilir ve şifre alınır. Alınan şifre istenen yere yapıştırılır ve drive’a bağlanılır.
2. Darknet’i indirmek için drive’daki klasörün içine darknet isimli klasör oluşturulur.
3. GoogleColab üzerinden !git clone "https://github.com/AlexeyAB/darknet.git" "/content/drive/MyDrive/klasöradı/darknet" komutu çalıştırılarak darknet klasörüne darknet indirilir.
4. İndirilen creating-files-data-and-name.py ve creating-train-and-test-txt-files.py dosyalarının içindeki full\_path\_to\_images "custom\_data" olarak değiştirilir. Değişiklik yapılan dosyalar kaydedilerek custom\_datanin içine kaydedilir.
5. %cd "/content/drive/MyDrive/klasöradı komutu çalıştırılarak genel klasöre geri dönülür.
6. Dosya konumlarını tutmak için label\_data.data dosyası oluşturulur bu dosya ise !python custom\_data/creating-files-data-and-name.py komutu çalıştırılarak oluşturulur.
7. Text ve train dosyalarını oluşturmak için ise !python custom\_data/ creating-train-and-test-txt-files.py
8. Drivedaki darknet klasöründe bulunan Makefile dosyasında değişiklik yapılır. Drivedaki Makefile dosyası silinir ve değişiklik yapılmış dosya yerine yüklenir.

Yapılacak değişiklikler:

GPU=1

CUDNN=1

OPENCV=1

1. GoogleColab üzerineden drive’daki darknet klasörüne girmek için %cd "/content/drive/MyDrive/klasöradı/darknet" komutu çalıştırılır.
2. Darknet !make komutu çalıştırılarak compile edilir.
3. %cd "/content/drive/MyDrive/klasöradı komutu çalıştırılarak genel klasöre geri dönülür.
4. Darknet klasörünün içindeki cfg klasöründeki yolov3.cfg(ben yolo3 kullandığım için başka versiyonlar kullanılacaksa o versiyonlar üzerinde işlem yapılmalı) dosyası indirilir ve eğitim için ayarlanır. Eğitmek için Testing’in altında bulunan batch ve subdivision comment satırı haline getirilmeli ve Trainingin altında bulunanlar uncomment yapılmalı. max\_batches değeri number of class\* 2000’e eşitlenmeli ve steps değerleri de max\_batches değerlerinin %80,%90 ı olarak yazılmalı. (benim 3 sınıfım var, max\_batches = 6000 steps=4800,5200 olarak ayarladım.).
5. Son olarak dosyada yolo kelimesi aratılmalı ve [yolo] başlığı altında bulunan classes sayısı sınıf sayısı ile değiştirilir. Her [yolo] başlığının hemen üzerindeki filters kısmı (sınıf sayısı + 5) \* 3 hesaplanarak o yazılmalı.
6. Dosya bu şekilde kaydedilir ve yolov3\_custom.cfg ismiyle kaydedilerek cfg klasörüne yüklenir.
7. Driveda bulunan ana klasöre custom\_weight klasörü açılır indirilen pre-trained weight dosyası bu klasörün içine eklenir. Pre-trained dosya datasetini optimize etmek için kullanılır.
8. Son olarak drivedaki ana klasöre sonuçları kaydetmek için backup klasörü oluşturulur.
9. İşlemlerin hepsi tamamlandıktan sonra GoogleColab’ta not defteri ayarlarına tıklanarak GPU ile başlanılır ve kodlar en baştan çalıştırılır (indirmeler ve githubtan yapılan clonelar hariç).
10. Tüm çalıştırmalar bittikten sonra aşağıdaki komut çalıştırılarak eğitim başlanır. !darknet/darknet detector train custom\_data/labelled\_data.data darknet/cfg/yolov3\_custom.cfg custom\_weight/darknet53.conv.74 -dont\_show
    1. Her 100 iterasyonda backup klasöründe yolo-obj\_last.weights dosyası güncellenir.
    2. Her 1000 iterasyonda backup klasöründe yolo-obj\_xxxx.weights dosyası oluşur.

Bu dosyaları alıp kullanabilirsiniz veya

bu dosyaları yukarıdaki komuttaki custom\_weight/darknet53.conv.74

yerine backup/ yolo-obj\_last.weights

yazarak kaldığınız yerden train etmeye devam edebilirsiniz.

* 1. Train ederken avg loss değeri 0.xxx altında bir değere düşmüş ve değer çok değişmiyorsa traini durdurmanız gerekmektedir.

Region Avg IOU: 0.798363, Class: 0.893232, Obj: 0.700808, No Obj: 0.004567, Avg Recall: 1.000000, count: 8 Region Avg IOU: 0.800677, Class: 0.892181, Obj: 0.701590, No Obj: 0.004574, Avg Recall: 1.000000, count: 8

**9002**: 0.211667, **0.60730 avg**, 0.001000 rate, 3.868000 seconds, 576128 images Loaded: 0.000000 seconds

* 1. Train sırasında her iterasyon sonunda buna benzer değerler alırsınız 9002: iterasyon sayısı   
     0.60730 avg ise ortalama hata sayısıdır (avg loss).

1. Train işlemi bitti veya istenilen değere gelinince durdurduğunuz zaman oluşan .weight dosyalarını test etmek için !darknet/darknet detector map customData/customData.data darknet/cfg/yolov4Custom.cfg backup/yolov4Custom\_1000.weights Kullanarak karşılaştırma yapılanabilir.
2. Kullanımı:
3. Drive’daki backup klasörünün içinden yolov3\_custom\_6000.weight (yolov3\_custom.cfg kullandığım ve max\_batches’ım 6000 olduğu için bu isimde) dosyası ve darknet’e yüklediğimiz yolov3\_custom.cfg dosyaları indirilir.
4. OpenCV’ye tanıtılması için aşağıdaki komut satırı kullanılır

net = cv.dnn.readNetFromDarknet('yolov3\_custom.cfg', 'yolov3\_custom\_6000.weights')

1. classes = []

with open(classFile, "r") as f:

classes = f.read().splitlines()

komutu ile .names isimli dosyadan isimler alınır.

1. blob = cv2.dnn.blobFromImage(img, 1/255, (416, 416), (0, 0, 0), swapRB=True, crop=False) komutu ile resim yolonun tanıması için blob a çevrilir (416,416 değeri yerine 32 nin katı olan değerlerle değiştirilerek hızı ve başarı oranı değiştirilebilir)
2. net.setInput(blob) komutu ile blob yolo ya verilir
3. output\_layers\_names = net.getUnconnectedOutLayersNames()

layerOutputs = net.forward(output\_layers\_names) Komutları ile blobdan tahminlerini yapar

1. for output in layerOutputs:

for detection in output:

scores = detection[5:]

class\_id = np.argmax(scores)

confidence = scores[class\_id]

if confidence > 0.5:

center\_x = int(detection[0] \* width)

center\_y = int(detection[1] \* height)

w = int(detection[2] \* width)

h = int(detection[3] \* height)

yolox = int(center\_x - w / 2)

yoloy = int(center\_y - h / 2)

boxes.append([yolox, yoloy, w, h])

confidences.append((float(confidence)))

class\_ids.append(class\_id)

bulunan tahminlerde tek tek gezerek en büyük güven değerli class adını ve bulur. O classın güven değeri 0.5 (%50) üzeri ise kutu oluşturur ve bu değerleri bir listeye kaydeder

1. indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4) ile NMS (Non Maximum Suppression) yaparak nesnelerdeki fazla kutuları tek bir kutuya indirger
2. if len(indexes) > 0:

for i in indexes.flatten():

yolox, yoloy, w, h = boxes[i]

label = str(classes[class\_ids[i]])

confidence = str(round(confidences[i], 2))

cv2.rectangle(img, (yolox, yoloy), (yolox + w, yoloy + h), (0,255,0), 2)

cv2.putText(img, label + " " + confidence, (yolox, yoloy + 20), font, 2, (255, 255, 255), 2)

ve en son kutular resme çizilir ve class adları yazılır