1. GİRİŞ

Nesne algılama, görsel bir görüntü veya videoda belirli bir nesneyi keşfetmeye çalıştığı bilgisayarla görme görevlerinden biridir [1]. Nesne algılamanın amacı, bilgisayarlı görü uygulamalarının temel problemlerinden biri olan bilgisayarlı görü uygulamalarının ihtiyaç duyduğu en basit bilgi parçalarından birini sağlayan bir hesaplamalı model veya teknik geliştirmektir., Neler var? Ve nerede bulunur? [2]. Nesneleri algılama, bilgisayarla görme alanındaki en iyi bilinen zorluklardan biridir [3]. Son on yılda, nesne keşfi, bilgisayarlı görme alanında en aktif ve büyüyen araştırma alanı haline gelmiştir [4]. Bu nedenle nesne algılama, segmentasyon, nesne izleme, görüntü açıklama, olay algılama, etkinlik tanıma vb. gibi karmaşık veya üst düzey bilgisayarla görme görevlerini çözmenin temelini oluşturur [5]. Uygulama açısından, nesne keşfi iki bölüme ayrılabilir: Birincisi, "genel nesnenin keşfi", bu tür keşiflerde insan vizyonunu ve algısını simüle etmek için birleşik bir çerçeve içinde farklı türdeki nesneleri keşfetmeyi amaçlar, İkinci ise “algılama uygulamaları”, bu tür keşif teknikleri de yaya algılama, yüz algılama, metin algılama vb. gibi önceden tanımlanmış türleri keşfetmeyi amaçlar [6]. Nesne keşfi, robotik görme, otonom sürüş, insan-bilgisayar etkileşimi, akıllı video izleme ve tıbbi görüntü algılama dahil olmak üzere çok çeşitli uygulamaları desteklemektedir [7]. Nesneleri algılama teknikleri kapsamlıdır ve hızlı bir şekilde geliştirilmekte olan bilgisayar görüşü için önerilen birçok algoritma vardır. Nesne algılama için en uygun tekniği seçmek için test teknikleri zaman ve çaba gerektirir. Her teknolojinin avantajları ve dezavantajları olduğundan uygun teknolojinin seçimini etkileyen birçok faktör vardır [8].

2. AMAÇ VE HEDEFLER

Nesne algılama nedir?

Bir görüntü veya videodaki nesnelerin türünü ve yerini tanımlamamızı sağlayan bir bilgisayarlı görme tekniğidir. Bu teknik, belirli bir sahnedeki nesnelerin toplamını sayabilir ve bunları doğru bir şekilde bulabilir.

Nesne algılamanın önemi, uygulanan algoritmanın bir fotoğraf veya videodaki belirli bir nesnenin türünü ve konumunu, bunu yapmak için gereken hızı ve bu algoritmanın algılayabileceği nesneler miktarını belirleme yeteneğidir. Bunun önemi, pratikte nesne algılama uygulandığında görülmektedir.

Nesnel algılama hedefleri [9]:

1. 'AI' veya makine öğreniminin birçok sözde hedefinden biri, bir sahneyi bir insan kadar kesin olarak tanımlamaktır. Bu amaca doğru basamak taşlarından biri, sahnedeki farklı önemli nesnelerin tespit edildiği ve semantikte altta yatan semantiğin anlaşılmaya çalışıldığı nesne tespitidir.
2. Nesne algılamanın bir diğer amacı, belirli bağlamlardaki kusurların / anomalilerin tespitidir. Bunun harika bir uygulaması, mimari alanların otomatik olarak araştırılması ve bakımıdır.
3. Nesne algılama ayrıca, nesnelerin geçmiş video karelerinde algılandıktan sonra gelecekteki konumunun tahmin edilmesi, daha fazla analiz (tanıma ve etiketleme) için canlı videodaki yüzlerin otomatik olarak açıklanması gibi video dizileri aracılığıyla nesnelerin izlenmesinde de kullanım alanı bulur.
4. Nesne algılamanın temel amacı, durağan görüntü veya video verilerinden bir veya daha fazla etkili hedefi tanımlamak ve bulmaktır. Görüntü işleme, desen tanıma, yapay zekâ ve makine öğrenimi gibi çeşitli önemli teknikleri kapsamlı bir şekilde içerir.
5. Karayolu trafik kazalarının önlenmesi, fabrikalardaki tehlikeli malların uyarıları, askeri kısıtlı alan izleme ve gelişmiş insan-bilgisayar etkileşimi gibi alanlarda geniş uygulama olanaklarına sahiptir.
6. Gerçek dünyada çoklu hedef tespitinin uygulama senaryoları genellikle karmaşık ve değişken olduğundan, doğruluk ve bilgi işlem maliyetleri arasındaki ilişkiyi dengelemek zor bir iştir.

3. GEREKSİNİMLER ARAÇLARI

Nesne algılama programını uygulamak için iki tür cihaza ihtiyacımız var. Biri donanımla ilgili cihaz, ikincisi yazılımla ilgili donanımdır.

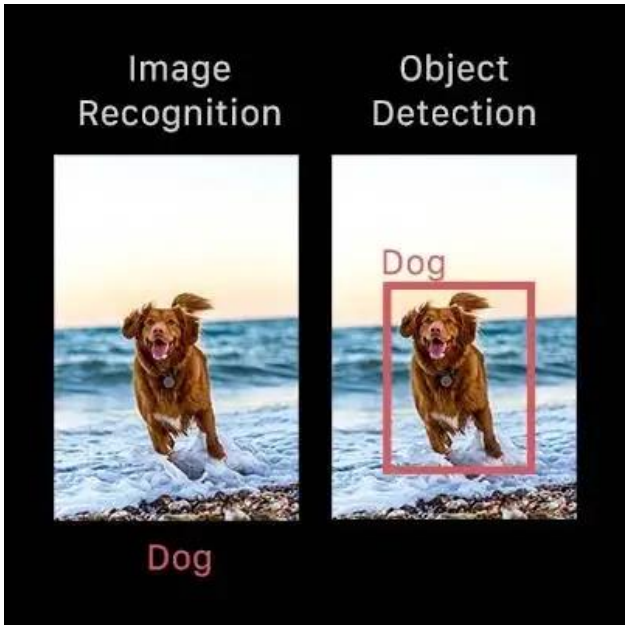
- ❖ Donanım Aparatı:
  - Windows sürüm 7 veya üzeri yüklü bir bilgisayar
  - İyi bir CPU veya işlemci
  - İnternet bağlantısı
- ❖ Yazılım Aparatı
  - Anaconda, Pycharm, visual studio, Matlab gibi bir yazılım geliştirme ortamı.
  - Python ve Opencv, Numpy Kütüphanesi

- CPU Nesne Tanıma Dosyaları
- Yerel cihazdaki bir görüntü
- Web kamerası

❖ **OpenCV**

Açık kaynaklı Görüntü İşleme temel olarak gerçek zamanlı Görüntü İşleme için kullanılan bir kütüphanedir. Python, C, C++, Java gibi çok çeşitli programlama dillerini destekler ve ayrıca Windows, Linux, MacOS gibi farklı platformları destekler. Bunu kullanarak, bir insanın nesnelerini, yüzlerini ve hatta el yazısını tanımlamak için görüntüleri ve videoları işleyebilir [7]. Numpy gibi çeşitli kütüphanelerle entegre olduğunda, Python analiz için OpenCV dizi yapısını işleyebilir. Görüntü desenini ve çeşitli özelliklerini tanımlamak için vektör uzayı kullanıyoruz ve bu özellikler üzerinde matematiksel işlemler gerçekleştiriyoruz. [7].

Nesne algılama genellikle görüntü tanıma ile karıştırılır. -> Görüntü tanıma bir görüntüye bir etiket atar. Bir arabanın resmi "köpek" etiketini alır. İki köpeğin resmi, hala "köpek" etiketini alıyor. -> Nesne algılama ise her köpeğin etrafına bir kutu çizer ve kutuyu "köpek" olarak etiketler. Model, her nesnenin nerede olduğunu ve hangi etiketin uygulanması gerektiğini tahmin eder. Bu şekilde, nesne algılama, görüntü hakkında tanımadan daha fazla bilgi sağlar.



Şekil 3.1: Nesne algılama ve görüntü işlemcisi arasındaki fark

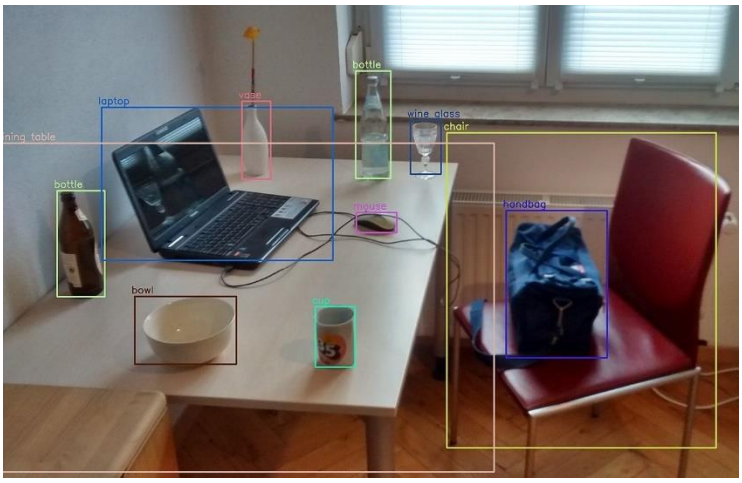
4. PYTHON KULLANARAK UYGULAMA

4.1. Yaklaşıma Nasıl Başlanır

OpenCV Kullanarak Nesne Algılama

Nesne algılama, bir görüntü veya videodaki nesneleri tanımlamamızı ve bulmamızı sağlayan bir bilgisayarlı görme tekniğidir. Bu tür bir tanımlama ve yerelleştirme ile, nesne algılama, bir sahnedeki nesneleri saymak ve kesin konumlarını belirlemek ve izlemek için kullanılabilir, ancak bunları doğru bir şekilde etiketler.

Derin öğrenme tabanlı nesne algılama modelleri genellikle iki bölümden oluşur. Bir kodlayıcı bir görüntüyü girdi olarak alır ve nesneleri bulmak ve etiketlemek için kullanılan istatistiksel özellikleri ayıklamayı öğrenen bir dizi blok ve katmandan geçirir. Kodlayıcıdan gelen çıktılar daha sonra her nesne için sınırlayıcı kutuları ve etiketleri tahmin eden bir kod çözücüye geçirilir.



Şekil 4.1: nesneler algılama

Bir dizi popüler nesne algılama modeli R-CNN ailesine aittir. Yıllar geçtikçe hem daha doğru hem de hesaplama açısından daha verimli hale geldiler. Tek atış dedektörü ailesine ait bir dizi model de vardır. MobileNet + SSD modelleri MobileNet tabanlı bir kodlayıcıya sahiptir ve YOLO modeli kendi evrişimli mimarisine sahiptir.

- Bu projede Python'da OpenCV kütüphanesi yardımıyla hareketsiz görüntüden objeleri tespit edeceğiz. OpenCV kütüphanesi görüntü işleme, nesne algılama için yaygın olarak bilinir ve birçok gerçek dünya uygulamasına sahiptir.
- Gerekli kütüphaneleri içe aktardıktan sonra, örnek görüntüyü okur, modeli eğitir, önceden tanımlanmış sınıflar (nesneler) için coco veri kümesini kullanır ve sonuç

olarak nesneyi, sınıfın konumunu, doğruluğunu ve dizinini (nesneyi tanımlamamıza yardımcı olur) algılardık.

- Bu projede başarılı bir şekilde insan, araba, kamyon ve trafik ışığı gibi nesneler görüntüden doğru bir şekilde algılanır. Ayrıca modelin ortalama doğruluğu %60'tan fazladır ve bu da yeterince adildir.

4.2. Kod Satırı Açıklaması

Bu bölümde nesne algılama yazılımı kodunu açıklayacağız, komutu kullanarak kolayca kurulabilen OpenCV kütüphanesini yüklemeliyiz*: **pip install opencv-python*** ve daha sonra pip komutunu kullanarak NumPy kütüphanesini kuracağız: ***pip install numpy***. Şimdi objectDetection.py gibi bir dosya oluşturun

Adım 1: Gerekli tüm kütüphaneleri içe aktarma. Başlamak için öncelikle **np** olarak adlandırılan **NumPy** kütüphanesini ve **cv2** olarak adlandırılan **OpenCV** kütüphanesini içe aktarmalıyız.

```
#importing libraries

import cv2
import numpy as np

print("Libraries imported successfully!")

Libraries imported successfully!
```

Adım 2: Programdaki görüntünün yolunu belirtin

```
#read the image
def ImgFile():
  img = cv2.imread('person.png')
```

Veya Programdaki video veya kameranın yolunu seçin

```
#read the Video
def Camera():
  cam = cv2.VideoCapture(0)
cam.set(3, 740)
cam.set(4, 580)
```

Adım 3: Modeli ve dondurulmuş model dosyasını eğitmek için yapılandırma dosyasına ihtiyacımız var. Bir nesne algılama modeli, birden çok nesne sınıfının varlığını ve konumunu algılamak için eğitilir. Örneğin, bir model, çeşitli meyve parçaları içeren görüntülerin yanı sıra temsil ettikleri meyve sınıfını belirten bir etiketle (ör. elma, muz veya çilek) ve her nesnenin görüntüde nerede göründüğünü belirten verilerle eğitilebilir.

Nesne algılama için ünlü algoritma:

SSD-MobileNetv2, SSD-MobileNetv3

```
configPath = 'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
weightPath = 'frozen_inference_graph.pb'
```

Adım 4: Nesne algılama için ünlü veri kümesi:
Prova edilen nesneleri tanımlar. Video resminde keşfedilebilir

COCO

```
#coco.names
person
bicycle
car
motorcycle
airplane
bus
train
truck
boat
traffic light
fire hydrant
street sign
stop sign
parking meter
bench
bird
cat
dog
horse
sheep
cow
elephant
bear
zebra
giraffe
hat
backpack
umbrella
shoe
eye glasses
handbag
tie
suitcase
frisbee
skis
snowboard
sports ball
kite
baseball bat
baseball glove
skateboard
surfboard
tennis racket
bottle
plate
wine glass
```

```
cup
fork
knife
spoon
bowl
banana
apple
sandwich
orange
broccoli
carrot
hot dog
pizza
donut
cake
chair
couch
potted plant
bed
mirror
dining table
window
desk
toilet
door
tv
laptop
mouse
remote
keyboard
cell phone
microwave
oven
toaster
sink
refrigerator
blender
book
clock
vase
scissors
teddy bear
hair drier
toothbrush
hair brush
```

Adım 5: dnn_dectionModel değeri ayarlama

```
net = cv2.dnn_DetectionModel(weightpath, configPath)
net.setInputSize(320 , 230)
net.setInputScale(1.0 / 127.5)
net.setInputMean((127.5, 127.5, 127.5))
net.setInputSwapRB(True)
```

Adım 6: resim yazıları kutusu çizim kodu

```
#green box
      if len(classIds) !=0:
         for classId, confidence, box in zip(classIds.flatten(),
confs.flatten(), bbox):
            cv2.rectangle(img, box, color=(0, 255, 0), thickness=2)
```

```
            cv2.putText(img, classNames[classId-1], (box[0] + 10,
box[1] + 20),
                        cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0),
thickness=2)
```

Adım 7: Çıktı

```
cv2.imshow('Output', img)
cv2.waitKey(1)
```

Adım 8: Görüntüler için nesne algılama kodu

```
###### Image için #######

def ImgFile():
   img = cv2.imread('person.png')

   classNames = []
   classFile = 'coco.names'

   with open(classFile, 'rt') as f:
      classNames = f.read().rstrip('\n').split('\n')

   configPath = 'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
   weightpath = 'frozen_inference_graph.pb'


   net = cv2.dnn_DetectionModel(weightpath, configPath)
   net.setInputSize(320 , 230)
   net.setInputScale(1.0 / 127.5)
   net.setInputMean((127.5, 127.5, 127.5))
   net.setInputSwapRB(True)

   classIds, confs, bbox = net.detect(img, confThreshold=0.5)
   print(classIds, bbox)

   for classId, confidence, box in zip(classIds.flatten(),
confs.flatten(), bbox):
      cv2.rectangle(img, box, color=(0, 255, 0), thickness=2)
      cv2.putText(img, classNames[classId-1], (box[0] + 10, box[1] +
20),
                  cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0),
thickness=2)


   cv2.imshow('Output', img)
   cv2.waitKey(0)
# ########################################
```

Adım 9: Kamera için nesne algılama kodu

```
####### Camera için #######
def Camera():
   cam = cv2.VideoCapture(0)

   cam.set(3, 740)
```

```
    cam.set(4, 580)

    classNames = []
    classFile = 'coco.names'

    with open(classFile, 'rt') as f:
        classNames = f.read().rstrip('\n').split('\n')

    configPath = 'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
    weightpath = 'frozen_inference_graph.pb'

    net = cv2.dnn_DetectionModel(weightpath, configPath)
    net.setInputSize(320 , 230)
    net.setInputScale(1.0 / 127.5)
    net.setInputMean((127.5, 127.5, 127.5))
    net.setInputSwapRB(True)

    while True:
        success, img = cam.read()
        classIds, confs, bbox = net.detect(img, confThreshold=0.5)
        print(classIds, bbox)

#green box
        if len(classIds) !=0:
            for classId, confidence, box in zip(classIds.flatten(),
confs.flatten(), bbox):
                cv2.rectangle(img, box, color=(0, 255, 0), thickness=2)
                cv2.putText(img, classNames[classId-1], (box[0] + 10,
box[1] + 20),
                            cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0),
thickness=2)


        cv2.imshow('Output', img)
        cv2.waitKey(1)
#####################################
```

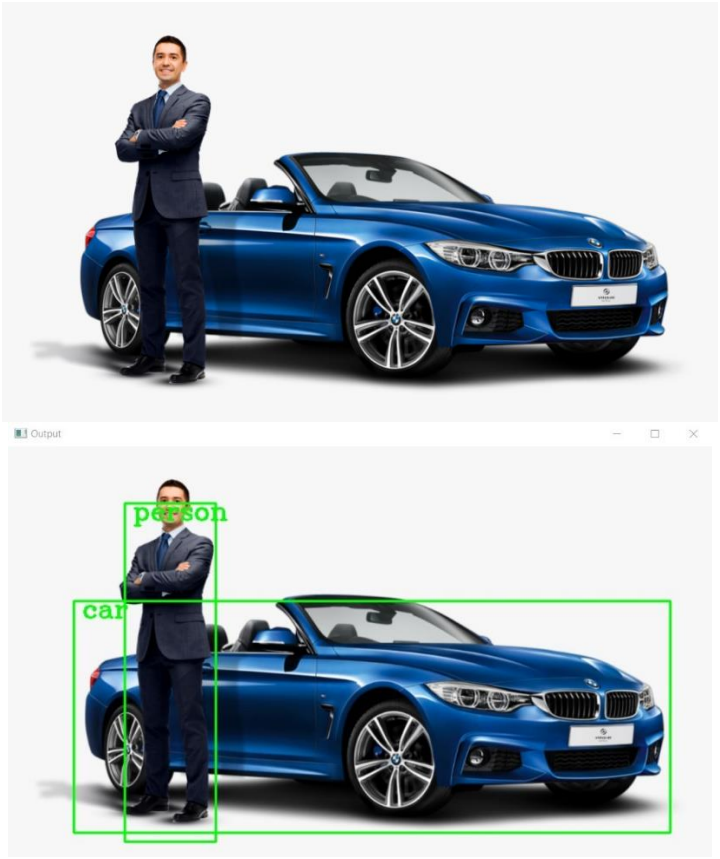
5. DENEYSEL ÇIKTILAR

Nesne tespiti için deneysel sonuçların çıktılarını iki yöntemle gösterdik.

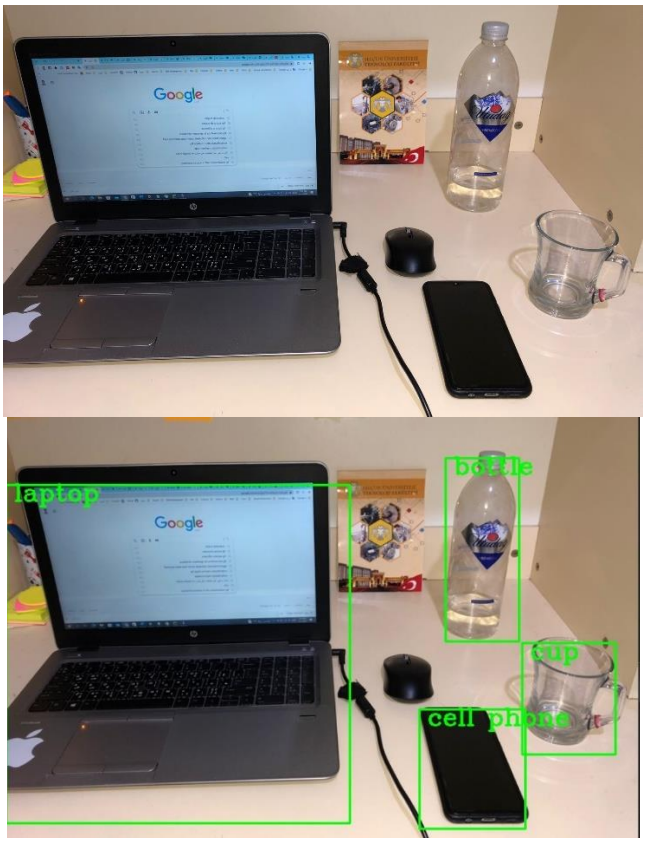
1- Fotoğraflardaki nesneleri keşfetme.

2- Gerçek zamanlı nesne algılama (kamera).

1. Fotoğraflardaki nesneleri keşfetme.

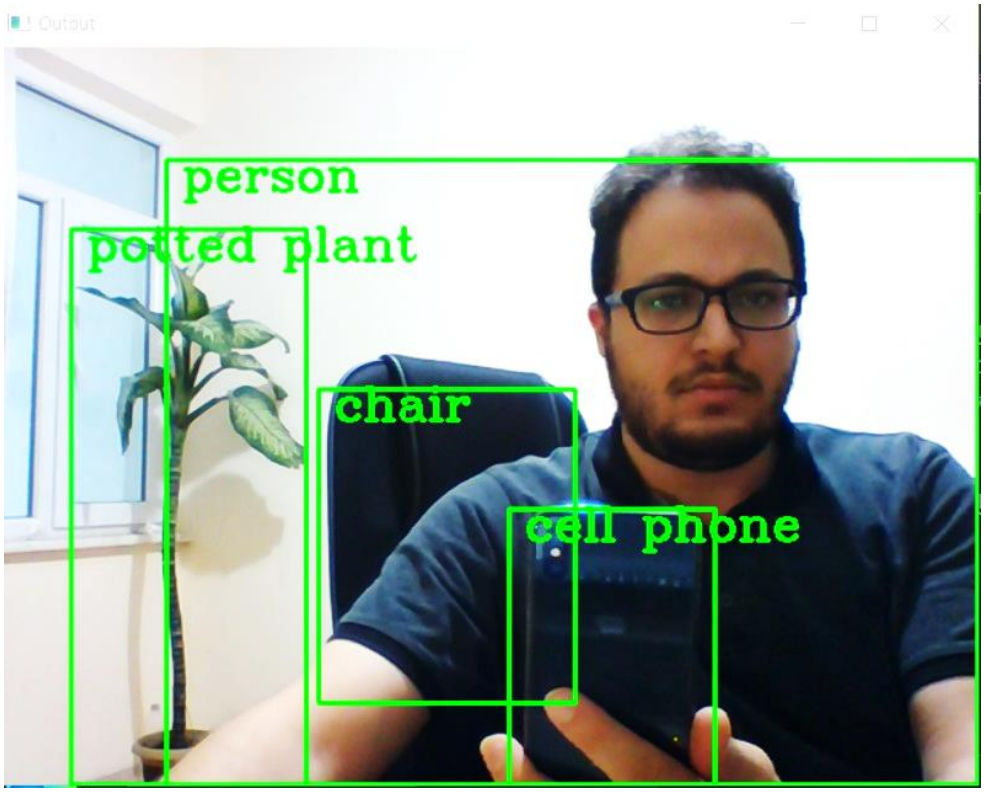


Şekil 5.1: Nesneleri görüntü olarak algıla



Şekil 5.2: Nesneleri görüntü olarak algıla

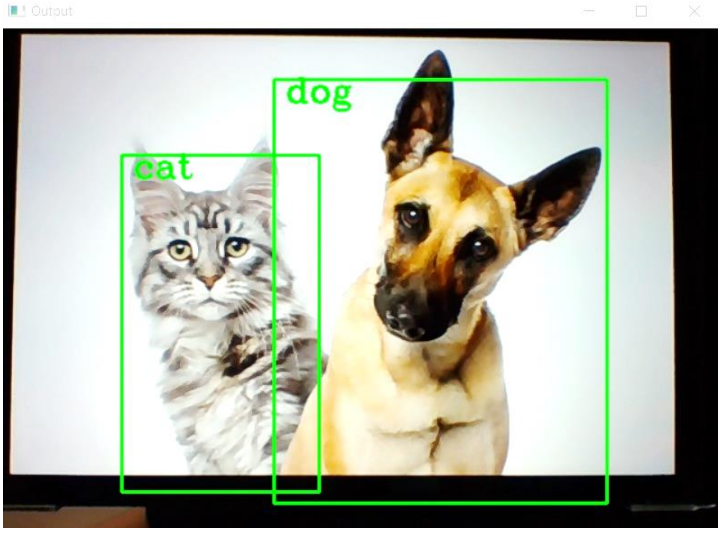
2- Gerçek zamanlı nesne algılama (kamera).



Şekil 5.3: Web kamerasındaki nesneleri açığa çıkarın

Şekil 5.4: nesneleri algılamak (araba)

Köpek ve kedi algılama görüntüsü için çıktı:

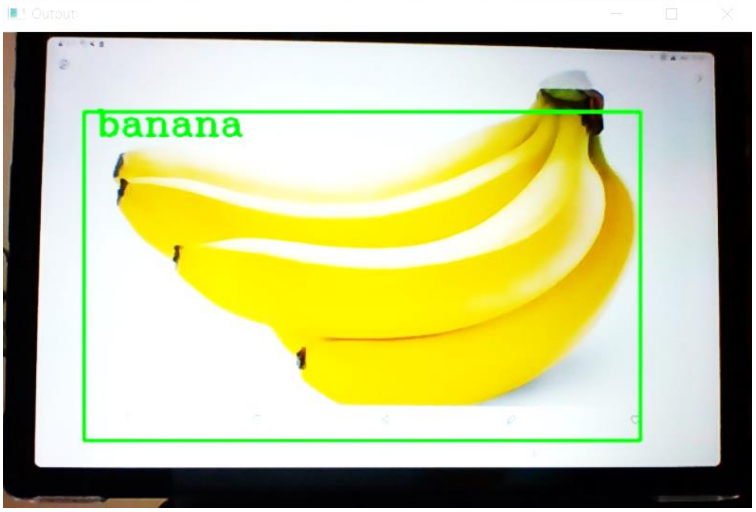


Şekil 5.5: Nesneleri algılama (köpek ve kedi)

Trafik Sinyali Algılama Görüntüsü için Çıkış:

Şekil 5.6: nesneleri algılama (trafik ışığı)

Muz algılama görüntüsü için çıktı:



Şekil 5.7: nesneleri algılama (muz)

6. SONUÇ

Derin öğrenmeye dayalı yöntemlere benzer şekilde, bu çalışmada kullanılan bu yöntem, önceden eğitilmiş masaların ve dosyaların varlığı nedeniyle birçok nesnenin keşfedilmesini sağlar, bu da GPU'nun nesneyi tanımak için gereken süreyi azaltmasını kolaylaştırır. Web kamerasını kullanırken videonun boyutunu da küçülttük, böylece nesneleri algılarken daha verimli olur. COCO dosyasında bulunan tespit edebileceğimiz birçok nesne var ve program iyi çalışıyor ve diğer yöntemlere kıyasla yüksek bir algılama doğruluğuna sahip. Bu deneme, bir nesnenin konumunu tahmin etmek için özellik ayıklama, eşleştirme ve RANSAC kullanan boyut, şekil veya renk özelliği tabanlı nesne bulma yöntemleri gibi basit nesne özelliklerini kullanan görüntü segmentasyonu ve blob analizini temel alır. Nesneler genellikle küçükse (görüntünün %5'inden az) algılanmaz. Nesneler genellikle birbirine yakın yerleştirilmişse algılanmaz. Bu program güvenlik kameralarında kullanılabilir ve belirli nesneleri algılaması ve istenen nesne algılandığında bir uyarı vermesi için eğitilerek geliştirilebilir. Özetlemek gerekirse, nesne algılama hayatımızı her zamankinden daha olumlu bir şekilde etkileyebilir.

7. KAYNAKÇA

[1] K. Li and L. Cao, "A review of object detection techniques," in *2020 5th International Conference on Electromechanical Control Technology and Transportation (ICECTT)*, 2020: IEEE, pp. 385-390.

[2] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *arXiv preprint arXiv:1905.05055,* 2019.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM,* vol. 60, no. 6, pp. 84-90, 2017.

[4] Diwakar and D. Raj, "Recent Object Detection Techniques: A Survey," *International Journal of Image, Graphics and Signal Processing,* vol. 14, no. 2, pp. 47-60, 2022, doi: 10.5815/ijigsp.2022.02.05.

[5] L. Liu *et al.*, "Deep learning for generic object detection: A survey," *International journal of computer vision,* vol. 128, no. 2, pp. 261-318, 2020.

[6] X. Zou, "A review of object detection techniques," in *2019 International Conference on Smart Grid and Electrical Automation (ICSGEA)*, 2019: IEEE, pp. 251-254.

[7] Y. Xiao *et al.*, "A review of object detection based on deep learning," *Multimedia Tools and Applications,* vol. 79, no. 33, pp. 23729-23791, 2020.

[8] M. Noman, V. Stankovic, and A. Tawfik, "Object detection techniques: Overview and performance comparison," in *2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 2019: IEEE, pp. 1-5.

[9] pawangfg. "Object Detection vs Object Recognition vs Image Segmentation." https://www.geeksforgeeks.org/object-detection-vs-object-recognition-vs-image-segmentation/ (accessed 14 January 2023.