T.C.
FIRAT ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

2021-2022 BAHAR DÖNEMİ

BİTİRME PROJESİ 1. ve 2. HAFTA RAPORU

KÜTÜPHANE KURULUMU VE TESTİ

185260012 – DİNÇER ŞİPKA
185260009 – EMİRHAN AKTAŞ
185260019 - SELİM CAN ERKAN

Bitirme projemizin en büyük amaçlarından olan görüntü işleme, makine öğrenmesi için kaynakların çoğunluğu, gerekli kütüphanelerin kolay elde edilmesi ve kurulması gibi faktörlerden dolayı projemizde Python programlama dilini kullanıyoruz.

Genel olarak Python kullanırken kodları test etme, kütüphane ekleme vb. işlemler de Komut Satırı yardımı ile yapılıyorken bu kullanımı kolaylaştıran ve ara yüzleştiren ayrıca farklı projeler de ayrı ayrı kütüphane kurma olanağı sağlayarak kütüphanelerin çakışma sorununun önüne geçen Anaconda Navigator ve kodlama için ise JupyterLab uygulamasını kullanıyoruz.

Projemizde temeli ve ana noktası olan Makine Öğrenmesi ile parmakların tespitidir. Tespit ne kadar iyi ve doğru bir şekilde sonuç verirse proje o kadar başarılı olacağı için Google tarafında geliştirilen MediaPipe kütüphanesini kullanacağız. Bu kütüphane başlangıçta YouTube'da video ve sesi gerçek zamanlı analiz etmek için geliştirilen ama zamanla farklı alanlara da entegre edilmiştir. Android, iOS gibi işletim sistemlerinde kullanıla bilen ayrıca Python, JavaScript ya da C++ programlama dilleri ile kodlanabilmektedir. Kütüphane Yüz Tespiti, El Tespiti, Göz Bebeği Tespiti, Obje Tespiti ya da Obje Takibi gibi konularda çözümler sunmaktadır.

Kütüphanenin Kurulumu ve Testi

Proje için Python 3.8 versiyonunu kullanıyoruz. Kameradan video çekme ya da video veya resmi dosya olarak okumak için open-cv ve elin tespiti için mediapipe kütüphanelerini kuruyoruz. Gerekli kütüphanelerin kurulumu için aşağıdaki kod satırlarını yönetici olarak başlattığımız Anaconda Navigatordaki terminale yazarak yüklüyoruz. Kodlar;

Open-cv için : pip install opencv-python

MediaPipe için : pip install mediapipe

Bu iki kütüphane ile birlikte google'ın mediapipe için düzenlediği dokümantasyondan faydalanarak bir resimdeki parmakları tespit etmeyi denedik.

Bu testte ilk olarak kullanacağımız kütüphaneleri projemize ekledik ayrıca mediapipe için de kullanacağımız çözüm olan hands'i bir değişkene atadık.

```
import cv2
import mediapipe
handsModule = mediapipe.solutions.hands
```

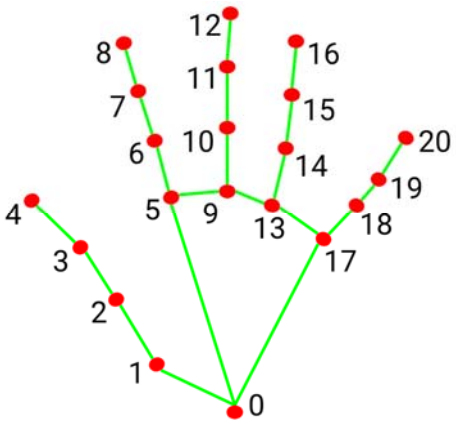
Ek 1

Daha sonra örnek amaçlı çektiğimiz bir el resimini open-cv kütüphanesi aracılığıla okuyup, bu resimin yükselik ve genişlik değerlerini birer değişkene atadık.

```
img = cv2.imread(r'C:\Users\ERKAN\Desktop\sol.jpg')
yukseklik, genislik, channels = img.shape
```

Ek 1.1

Dokümantasyonda bir elde tespit edilebilecek yerler indexlenerek aşağıdaki görseldeki gibi belirtilmiştir. Bu indexler aracılığıyla bir görseli işleyip istediğimiz indexin olup olmadığı ya da nerede olduğu, başka bir indexten yukarıda mı yoksa aşağıda mı olduğu gibi farklı amaçlarla kullanabilme olanağı sağlamaktadır.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Ek 1.2

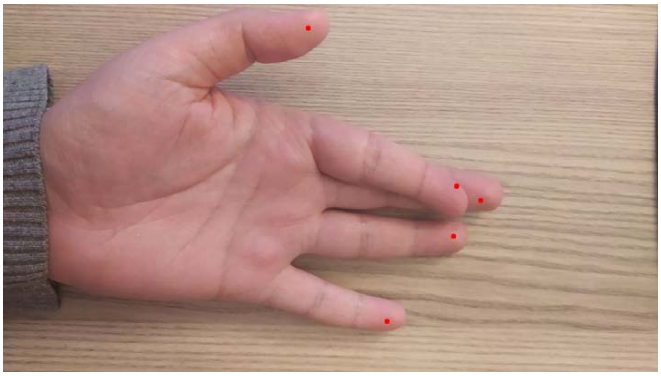
Kullandığımız çözüm içindeki Hands metodu ile belirli paremetreler girerek bir değişken tanımlıyoruz. Bu değişkendeki process metodu yardımı ile de BGR formatında okunmuş resimi RGB formatına dönüştürdükten sonra parametre olarak gönderiyor ve görseli model yardımıyla işlemiş oluyoruz. Bu işlemden sonra istediğimiz indexlerin sonuçta var olup olmadığını varsa koordinatlarını alıyoruz. Bu koordinatlar 0 ile 1 arasında bir değer olduğundan resimdeki koordinatlarını bulmak amacıyla yükselik ve genişlikle çarpıyor ve değerin float olmasına karşın int formata çeviriyoruz. Daha sonra bu koordinatlara birer circle çizerek görselleştiriyoruz.

```
yedek = img.copy()
with handsModule.Hands(static_image_mode=True, min_detection_confidence=0.7, max_num_hands=2) as hands:
    results = hands.process(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    if results.multi_hand_landmarks != None:
        for handLandmarks in results.multi_hand_landmarks:
            for parmak in indexler:
                tip = handLandmarks.landmark[parmak]
                tip_coor =((int)(tip.x*genislik),(int)(tip.y*yukseklik))

                cv2.circle(yedek, tip_coor, 5, (0,0,255), -1)

    cv2.imshow('Sonuc', yedek)
    cv2.waitKey(0);
    cv2.destroyAllWindows()
```

Ek 1.3

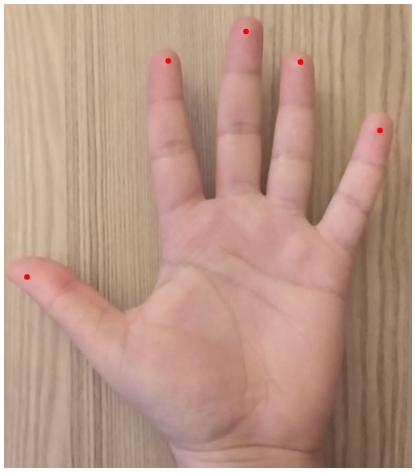
Bazı örnek resimleri ve sonuçları;

Şekil 1



Şekil 1.1

Şekil 1.2



Şekil 1.3

Burada da görüleceği gibi bazı resimlerde parmak ucu görünmese dahi model tespit edebilmiştir. Dokümantasyonda bunun ile alakalı olarak model öncelikli olarak avuç içini tespit ediyor daha sonra el boğumları için doğrudan bir koordinat tahmini yapıyor. Bunu yapabilmek ve modelin daha doğru bir sonuç vermesi için yaklaşık olarak 30.000 adet resim el ile 21 adet boğumu tanımlanıyor(Bu boğumlar yukarıda bahsettiğimiz indexler).