# Görüntü İşlemeye Giriş

El Hareketleri İle Kontrol Edilebilen Klavye

# Proje Detayları

Projenin amacı, el hareketlerinin takibi sağlanarak klavye, mouse ve ses düzeyinin kontrol edilmesidir. Proje üç aşamadan oluşmaktadır:

1. **Klavyenin Oluşturulması**
2. **Mouse Hareketlerinin Algılanması**
3. **Ses Düzeyinin Kontrol Edilmesi**

# Projede Kullanılan Kütüphaneler

## OpenCV

- **OpenCV** (Open Source Computer Vision), bilgisayarlı görü, makine öğrenimi, görüntü işleme gibi uygulamalar için kullanılan açık kaynak kodlu bir kütüphanedir.
- OpenCV platform bağımsız bir kütüphanedir. C++, Python, Java vb. gibi çok çeşitli programlama dillerini destekler ve Windows, Linux, OS X, Android ve iOS gibi farklı platformlarda kullanılabilir.

## CVZone

- **CVZone,** görüntü işleme ve yapay zeka işlevlerini çalıştırmayı kolaylaştıran bir bilgisayarlı görü paketidir. Temel olarak OpenCV ve Mediapipe kütüphanelerini kullanır.
- **MediaPipe:** MediaPipe Google tarafından sunulan, cross-platform makine öğrenmesi çözümleri sunan açık kaynak bir pakettir. Yüz tanıma, iris algılama, nesne takibi gibi görevlerde kullanılmaktadır.

## AutoPy / NumPy

- **AutoPy,** Python için basit, platformlar arası bir GUI otomasyon kitaplığıdır. Klavye ve fareyi denetleme, ekranda renkleri ve bit eşlemleri bulma ve uyarıları görüntüleme işlevleri içerir.
- **NumPy,** dizilerle çalışmak için kullanılan br Python kütüphanesidir. Doğrusal cebir, fourier dönüşümü ve matrisler alanlarında çalışmak için işlevlere sahiptir.

## pynput

- **pynput** , giriş cihazlarının kontrol edilmesi ve izlenmesini sağlar. Desteklenen her giriş aygıtı türü için alt paketler içermektedir. Alt paketlerden herhangi birini kullanmak için ana paketten içe aktarım yapılmalıdır.

# Projede Kullanılan Kütüphaneler

## pycaw

- **pycaw,** Python dilinde yazılmış bir kütüphanedir ve Windows işletim sistemi üzerinde çalışan bir uygulamanın ses ayarlarını kontrol etmek için kullanılır. pycaw, Windows'un COM (Component Object Model) arabirimini kullanarak, Windows'taki ses ayarlarını yönetmek için gerekli olan API'leri kullanarak, ses ayarlarını kontrol etmek için kullanılabilir. pycaw, Windows'taki tüm ses cihazlarının listesini almanıza, varsayılan cihazları değiştirmenize, ses seviyelerini değiştirmenize ve kaynakları yönetmenize olanak tanır.

## comtypes

- **comtypes,** Python programlama dilinde Microsoft Component Object Model (COM) arabirimlerine erişmek için kullanılan bir kütüphanedir. COM, farklı programlama dilleri veya farklı bilgisayarlar arasında biletiği olan nesnelerin birbiriyle etkileşimini sağlayan bir Microsoft teknolojisidir.

## Ctypes

- **ctypes,** Python programlama dilinde kullanılan bir kütüphanedir ve C diliyle yazılmış kütüphanelere erişim sağlamak için kullanılır. Windows, Linux ve macOS gibi birçok işletim sistemi için destek sağlar.

# Aşama 1 : Klavyenin Oluşturulması

# Görüntünün Alınması

```
# Görüntünün alınması
cap=cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH,1150)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT,800)
cv2.namedWindow("Image", cv2.WINDOW_NORMAL)
cv2.resizeWindow("Image", 1150, 800)
```

# Ellerin Algılanması

```python
detector = HandDetector(detectionCon=0.8, maxHands=1)

while True:
    state, frame = cap.read()
    frame = cv2.flip(frame, 1)

    # Ellerin Algılanması
    hands, frame = detector.findHands(frame)
    lmList =[]
    if hands:
        hand1 = hands[0]
        lmList1 = hand1['lmList'] #Eldeki 21 noktanın listesi
        bbox1 = hand1['bbox'] # Çerçeve (x,y,w,h)
        centerPoint1 = hand1['center'] #Elin orta noktası (cx,cy)
        handType1 = hand1['type'] #Sağ el veya sol el
        finger1 = detector.fingersUp(hand1)

        lmList = lmList1
```

# Butonların Oluşturulması

```
#Keybord için değişkenler
finalText = []
keyboard = Controller()

#Klavye harflerinin liste haline getirlimesi
keys=[["0","1","2","3","4","5","6","7","8","9"],
      ["Q","W","E","R","T","Y","U","I","O","P"],
      ["A","S","D","F","G","H","J","K","L","EN"],
      ["Z","X","C","V","B","N","M",".","  ","Del"]]
```

```
# Klavye tuşlarının oluşturulması
class Button():
    def __init__(self,pos,text,size=[80,80]):
        self.pos = pos
        self.size= size
        self.text = text

buttonList = []

# Tuşların buttonList'e eklenmesi
for i in range(len(keys)):
    for j, key in enumerate(keys[i]):
        buttonList.append(Button([100*j+50, 100*i+30],key))
```

Klavyede bulunacak 40 butonun tek tek el ile oluşturulması oldukça uzun vakit almaktadır. Bu sebeple bir döngü yardımıyla butonların oluşturulması amaçlandı.

İlk olarak butonların isimleri bir liste olarak tutuldu. Daha sonra ‘Button’ adlı bir class oluşturularak parametreler belirlendi. Butonların tutulacağı boş bir liste oluşturuldu ve iç içe iki for döngüsü kullanılarak butonların bu listeye eklenmesi sağlandı.

# Butonların Çizidirilmesi

```python
# Butonların Çizilmesi
def drawButtons(frame,buttonList):

    for button in buttonList:
        x, y = button.pos
        w, h = button.size
        # Dıştaki mavi renkli karenin oluşturulması
        cv2.rectangle(frame, button.pos, (x + w, y + h), (158, 163, 90),cv2.FILLED)
        # Harflerin yazılması
        cv2.putText(frame, button.text, (x + 8, y + 50), cv2.FONT_HERSHEY_PLAIN, 3, (255, 255, 255),3)

    return frame
```

Butonların çizdirilmesi için bir fonksiyon oluşturuldu. Bu fonksiyonda bir for döngüsü kullanılarak butonların tutulduğu buttonList listesine erişildi ve gerekli pozisyon ve boyut bilgileri alınarak butonların görselleştirilmesi sağlandı.

# Butonların Üzerinde Gezinme

Butonlara erişilebilmesi için bir for döngüsü kullanılarak butonlara erişildi ve pozisyon ve boyut bilgileri alındı.
Klavye üzerinde gezinmeyi sağlamak için baş parmağın ucunun x ve y koordinatları alındı ve klavye tuşlarının üzerinde olup olmadığı kontrol edildi. Eğer baş parmağın ucunun koordinatları klavye tuşlarının üzerinde ise butonun renginin mor olması sağlandı.

Daha sonra CVZone kütüphanesine ait findDistance() fonksiyonu kullanılarak baş parmak ile işaret parmak arasındaki uzaklık elde edildi.

```
drawButtons(frame,buttonList)

# Burada butonlara tıklanması halinde oluşacak görsel değişiklikler tanımlandı
if lmList:
    for button in buttonList:
        x, y = button.pos
        w, h = button.size
        x1, y1 = lmList[4][0:2]

        # Eğer parmağımızın ucu butonların bulunduğu x ve y değerlri arasındaysa
        if x < lmList[4][0] < x + w and y < lmList[4][1] < y + h:
            # Mevcut durumda üstünde bulunulan butonun rengi koyu mor olur.
            cv2.rectangle(frame, button.pos, (x + w, y + h), (130, 72, 126), cv2.FILLED)
            cv2.putText(frame, button.text, (x + 10, y + 50), cv2.FONT_HERSHEY_PLAIN, 3, (255, 255, 255), 3)

            # Burada parmak hareketi tanımlandı.
            length, info, frame = detector.findDistance(lmList1[4][0:2], lmList1[8][0:2], frame)
```

# Butonların Tıklanabilir Hale Getirilmesi

Belirlenen bu iki parmak arsındaki uzaklık eğer 30'dan küçük ise klavye tuşlarının bej rengine dönmesi ve tıklama işleminin yapılması amaçlandı. Bu tıklama işleminin yapılabilmesi için pynput kütüphanesinden keyboard alt paketi kullanılarak buton üstündeki harflerin bilgisayar klavyesinden bastırılması sağlandı. Özel olarak delete ve Enter butonlarının x ve y değerleri belirlenerek özel işlevlerinin atanması sağlandı

```python
# Yukarıda belirlenen parmak hareketinin length'ine göre "tıklama" sonucu butonun rengi bej rengi olur.
if length<30:
    #Buton üstündeki harflerin klavye üstünden bastırılarak bilgisayar üstünde kullanılması
    if(x==950 and y==330):
        #Delete butonu için:
        keyboard.press(Key.backspace)
        cv2.rectangle(frame, button.pos, (x + w, y + h), (190, 188, 218), cv2.FILLED)
        cv2.putText(frame, button.text, (x + 10, y + 50), cv2.FONT_HERSHEY_PLAIN, 3, (255, 255, 255), 3)
        sleep(0.17)
    elif x == 950 and y == 230:
        # Enter butonu için:
        keyboard.press(Key.enter)
        cv2.rectangle(frame, button.pos, (x + w, y + h), (190, 188, 218), cv2.FILLED)
        cv2.putText(frame, button.text, (x + 10, y + 50), cv2.FONT_HERSHEY_PLAIN, 3, (255, 255, 255), 3)
        sleep(0.17)
    else:
        #Diğer butonlar için:
        keyboard.press(button.text)
        cv2.rectangle(frame, button.pos, (x + w, y + h), (190, 188, 218), cv2.FILLED)
        cv2.putText(frame, button.text, (x + 10, y + 50), cv2.FONT_HERSHEY_PLAIN, 3, (255, 255, 255), 3)
        sleep(0.15)
```

# Aşama 2 : Mouse Hareketlerinin Algılanması

# Değişkenler Ve Mouse Hareket Alanı

```
smoothening = 7
prev_locationX, prev_locationY = 0,0
crnt_locationX, crnt_locationY = 0,0
widthScreen, heightScreen = autopy.screen.size()
```

```
#MOUSE
# işaret ve orta parmakların ucunu alınması
if len(lmList) != 0:
    x1, y1 = lmList[8][0:2]
    x2, y2 = lmList[12][0:2]

    cv2.rectangle(frame, (50, 700), (1000,450),
                  (255, 0, 255), 2)
```

Mouse hareketlerinin belirlenmesinde kullanılacak değişkenler tanımlandı. Daha sonra kullanılması belirlenen parmakların lmlist içinden koordinatları alındı ve mouse'un hareket alanı belirlendi.

# Mouse İşlemleri

```
# Birinci parmak havadaysa ve ikincisi değilse sadece moving yani gezinme işlemi yapılır.
if (finger1[1] == 1 and finger1[2] == 0) and (50<=x1 and x1<=1000) and (450<=y1 and y1<=700):
    # 5. convert coordinates
    x3 = np.interp(x1, (50,1000), (0, (widthScreen)))
    y3 = np.interp(y1, (450, 650), (0, (heightScreen)))
    # 6. smoothen values
    # Çok fazla titreme olduğundan smoothing işlemi gerçekleştirilir.
    crnt_locationX = prev_locationX + (x3 - prev_locationX) / smoothening
    crnt_locationY = prev_locationY + (y3 - prev_locationY) / smoothening
    # 7. move mouse
    autopy.mouse.move(crnt_locationX, crnt_locationY)
    cv2.circle(frame, (x1, y1), 15, (255, 0, 255), cv2.FILLED)

    prev_locationX, prev_locationY = crnt_locationX, crnt_locationY
```

Tek parmağın havada olması durumunda imleç ile gezinme işlemi gerçekleştirildi. Mouse'un hareket işlemleri autopy.mouse.move() fonksiyonu kullanılarak sağlandı.

Numpy.interp() fonksiyonu, doğrusal enterpolant işlemlerinde kullanılmaktadır. Koordinatlar arasındaki farkları ölçeklendirmek ve bu değerleri cv2.rectangle ile oluşturulan mouse'un hareket aralığı çerçevesine sıkıştırmak için np.interp() fonksiyonu kullanılarak x1 ve y1 değişkenlerinin değerleri x3 ve y3 değişkenlerine dönüştürüldü.

# Mouse İşlemleri

```
# Her iki parmakta havadaysa tıklama işlemi yapılır.
if finger1[1] == 1 and finger1[2] == 1  and (50<=x2 and x2<=1000) and (450<=y2 and y2<=650):
    length, info, frame = detector.findDistance(lmList[8][0:2], lmList[12][0:2], frame)
    # İki parmak arası uzaklık kontrol edilir.
    if(length<=40):
       cv2.circle(frame,(info[4],info[5]),15,(0,255,0),cv2.FILLED)
    #İki parmak havada ve yan yana ise : tıklama
       autopy.mouse.click()
```

İki parmağın havada olması ve parmaklar arasındaki mesafenin 40'tan küçük olduğu durumlarda tıklama işlemi gerçekleştirildi. Mouse'un tıklama işlemi autopy.mouse.click() fonksiyonu kullanılarak sağlandı.

# Mouse İşlemleri

Mouse hareket halindeyken imleçte titremeler yaşandığı görülmüştür. Bu titremeyi yok etmek amacıyla smoothing işlemi gerçekleştirilmiştir. Smoothing işlemi için öncelikle X ve Y koordinat değerleri için önceki ve güncel konum değerleri atanmıştır. İmleç konumunun mevcut ve önceki konumu arasındaki farkı hesaplanmış ve belirlenen smoothening değeri kullanılarak hareketlerin daha pürüzsüz hale getirilmesi sağlanmıştır.

Autopy modülü kullanılarak, imlecin crnt_locationX ve crnt_locationY değişkenlerinde belirtilen konuma hareket etmesi sağlanmaktadır. Son işlem olarak prev_locationX ve prev_locationY değişkenleri, mevcut imleç konumunu saklamak için güncellenmiştir.

```
smoothening = 7
prev_locationX, prev_locationY = 0,0
crnt_locationX, crnt_locationY = 0,0

# 6. smoothen values
# Çok fazla titreme olduğundan smoothing işlemi gerçekleştirilir.
crnt_locationX = prev_locationX + (x3 - prev_locationX) / smoothening
crnt_locationY = prev_locationY + (y3 - prev_locationY) / smoothening
# 7. move mouse
autopy.mouse.move(crnt_locationX, crnt_locationY)
cv2.circle(frame, (x1, y1), 15, (255, 0, 255), cv2.FILLED)

prev_locationX, prev_locationY = crnt_locationX, crnt_locationY
```

# Aşama 3 : Ses Düzeyinin Kontrol Edilmesi

# Değişkenler Ve Ses Kontrol Alanı

```
#Volume için değişkenler
devices = AudioUtilities.GetSpeakers()
interface = devices.Activate(
    IAudioEndpointVolume._iid_, CLSCTX_ALL, None)
volume = cast(interface, POINTER(IAudioEndpointVolume))
volRange = volume.GetVolumeRange()

minVol = volRange[0]
maxVol = volRange[1]
vol = 0
volBar = 400
volPer = 0
```

```
#VOLUME:

x4, y4 = lmList[4][0:2]
x5, y5 = lmList[8][0:2]

cv2.rectangle(frame, (1030, 450), (1250, 700),
              (255, 0, 255), 2)
```

AudioUtilities modülünün GetSpeakers() yöntemi kullanılarak varsayılan ses aygıtını temsil eden bir cihaz nesnesi elde edildi.

IAudioEndpointVolume, aygıtın ses seviyesini okuyabilen ve ayarlayabilen bir arabirimdir. iid ifadesi, IAudioEndpointVolume arabiriminin kimliğini belirtir ve CLSCTX_ALL, arabirimin etkinleştirilmesi için kullanılacak bağlamı belirtir. Bir kez arabirim elde edildikten sonra, ses seviyesini kontrol etmek için kullanılabilen IAudioEndpointVolume arabirimi örneğine dönüştürülür.

MinVol ve maxVol değişkenleri ile ses düzeylrinin minimum ve maximum değerleri tutuldu.

Ses düzeyi için kullanılacak değişkenler tanımlandı. Kullanılması belirlenen parmakların koordinatları alınarak ses kontrol alanı oluşturuldu.

# Ses Düzeyi Kontrolü

```python
#İki parmağında havada olması ve belirlenen dikdörtgenin içinde bulunması kontrol edilir.
if (finger1[0] == 1 and finger1[1] == 1 and (1030<=x4 and x4<=1250) and (450<=y4 and y4<=700)):

    length, info, frame = detector.findDistance(lmList[4][0:2], lmList[8][0:2], frame)

    #50,150 parmaklar arasındaki uzaklığı temsil etmektedir.
    vol = np.interp(length, [50, 150], [minVol, maxVol])
    volBar = np.interp(length, [50, 150], [400, 50])
    volPer = np.interp(length, [50, 150], [0, 100])


    #Volum oranının 5'er artması sağlanır.
    smoothnessV = 5
    volPer = smoothnessV * round(volPer / smoothnessV)
    print(int(length), vol)


    volume.SetMasterVolumeLevelScalar(volPer / 100, None)

    #Volum bar'ın görselleştirilir.
    cv2.rectangle(frame, (1100, 50), (1130,400 ), (130, 72, 126), 3)
    cv2.rectangle(frame, (1100, int(volBar)), (1130, 400),(130, 72, 126), cv2.FILLED)
    cv2.putText(frame, f'{int(volPer)} %', (1060, 430), cv2.FONT_HERSHEY_COMPLEX, 1, (130, 72, 126), 3)
```

İki parmağın havada olması ve kontrol alanı içinde bulunması durumu kontrol edildi. Daha sonra bu iki parmak arasındaki mesafe bulundu.

Parmaklar arasındaki mesafe 50 ile 150 arasında iken:

- Bu mesafeler enterpolasyon işlemi kullanılarak minVol ve maxVol aralığı ile eşleştirildi.
- volBar değişkeni enterpolasyon işlemi ile y ekseninde 400 ile 50 aralığında hareket edebilecek şekilde ayarlandı.
- volPer değişkeninde 50 ile 150 arasındaki mesafe bilgisayardaki ses düzey aralığı olan 0-100 aralığı ile eşleştirildi.

smoothnessV değeri ile ses düzeyinin 5'er artması sağlandı ve akabinde titreme sorunu giderildi.

Son olarak ses düzeyinin kontrol edilmesi görselleştirildi.

# Teşekkürler....

Selin YİĞİT
19010011014

Nihal DEMİR
19010011016

Edanur ERGENÇ
19010011050