Görüntü İşleme

DUYGU ANALİZİ

Projenin Amacı

•Projemizin amacı, yapay zeka teknolojisini kullanarak insanların duygu durumlarını analiz etmek ve anlamak için bir sistem geliştirmektir. Bu sistem, önceden belirlenmiş bir set duygusal kategorilere (mutlu, üzgün, kızgın vb.) dayalı olarak, insanların yüz ifadelerini analiz ederek duygusal durumlarını belirleyecektir.

•

•Bu projenin hedefi, birçok farklı alanda kullanılabilecek birçok fayda sağlamaktır. Örneğin, eğitim sektöründe öğrencilerin dersleri nasıl aldıklarını veya öğrenme ortamlarının onların duygu durumları üzerindeki etkisini belirlemek için kullanılabilir. Bu proje aynı zamanda, pazarlama sektöründe de kullanılabilir. Örneğin, müşterilerin ürünler hakkındaki görüşleri hakkında daha fazla bilgi edinmek için kullanılabilir.

Projenin Amacı

•Sonuç olarak bu proje, yapay zeka teknolojisinin kullanımını duygu durumlarının analizi için kullanarak, farklı sektörlerde birçok fayda sağlayabilecek bir projedir.

Kütüphaneler

- Open CV
- Numpy
- sys
- Time
- PyQt5

Open CV

Open Source Computer Vision Library

- OpenCV, görüntü işleme işlemleri, nesne algılama, yüz tanıma, yüz ifadesi analizi, el hareketleri tespiti, optik karakter tanıma (OCR) gibi birçok işlemi yapmak için kullanılabilir.
- OpenCV, birçok özellik ve fonksiyon sunar. Bunlar arasında resim işleme, video işleme, nesne algılama, kamera ve video giriş-çıkış işlemleri, hızlı işlem yapma ve kolay kullanım gibi özellikler yer alır.

Numpy

Numerical Python

- NumPy (Numerical Python), Python programlama dilinde bilimsel hesaplamalar için kullanılan bir kütüphanedir. NumPy, yüksek performanslı çok boyutlu dizi nesneleri ve bu diziler üzerinde çalışacak bir dizi fonksiyon ve araç sağlar.

- NumPy, bilimsel hesaplamalar ve veri işleme için bir dizi fonksiyon ve araç sunar. NumPy'nin bazı kullanım alanları: Matematiksel hesaplamalar, Dizi işlemleri, Lineeer cebir işlemleri.

0SYS

●SYS, Python programlama dilinin standart kütüphanelerinden biridir. Bu kütüphane, Python programlarının sistemle etkileşimde bulunmasını sağlar. SYS kütüphanesi, Python yorumlayıcısı ve çalışma zamanı çevresi hakkında bilgi sağlar, komut satırı argümanlarını işler, standart girdi, çıktı ve hata akışını yönetir, modül yükleme işlemlerini kontrol eder ve sistemle ilgili diğer işlemleri gerçekleştirir.

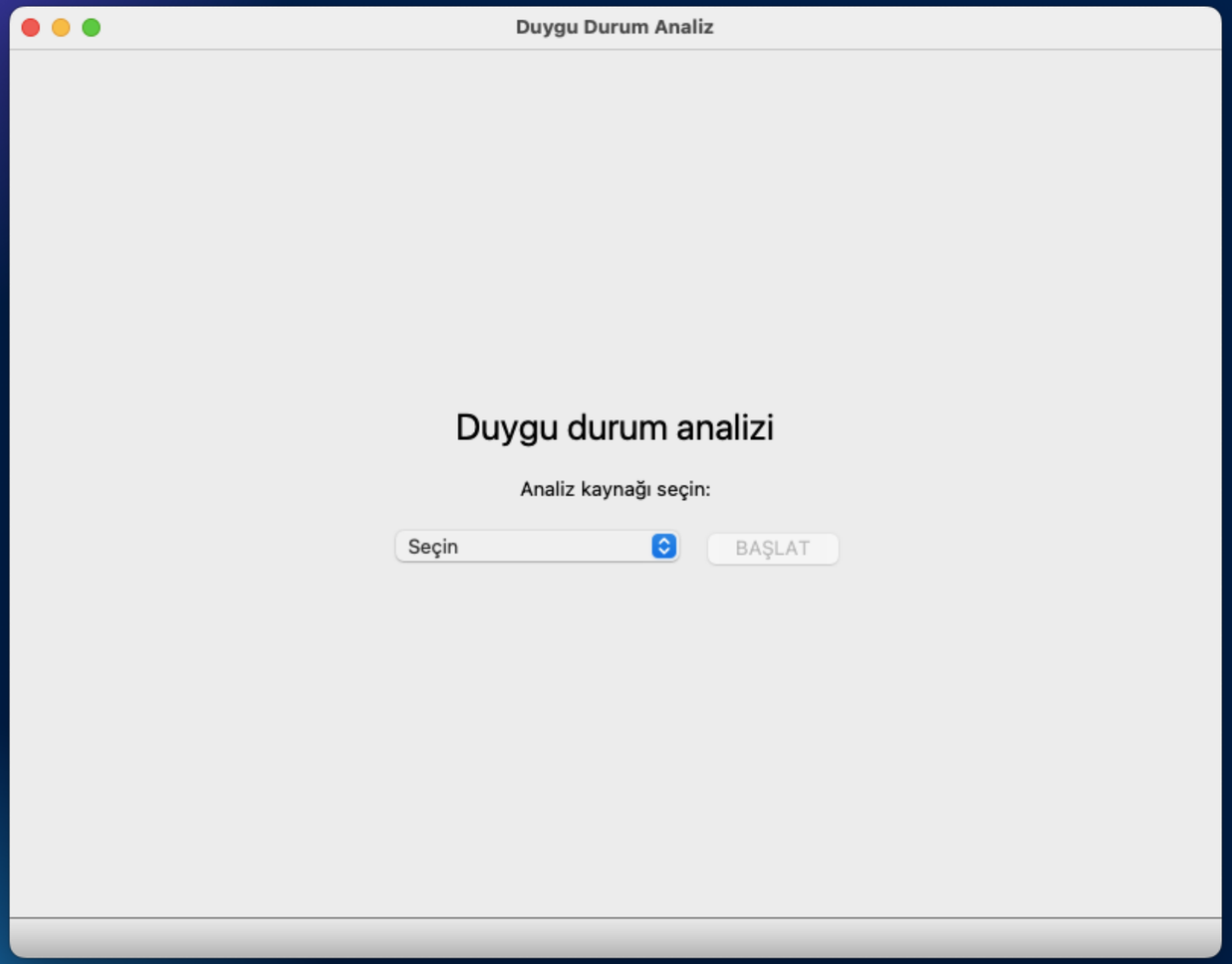
Time

●Time, Python programlama dilinin standart kütüphanelerinden biridir. Bu kütüphane, zamanla ilgili işlemler yapmamızı sağlar. Time kütüphanesi, zamanı okumak, ölçmek, işlemek ve kontrol etmek için bir dizi işlev içerir.

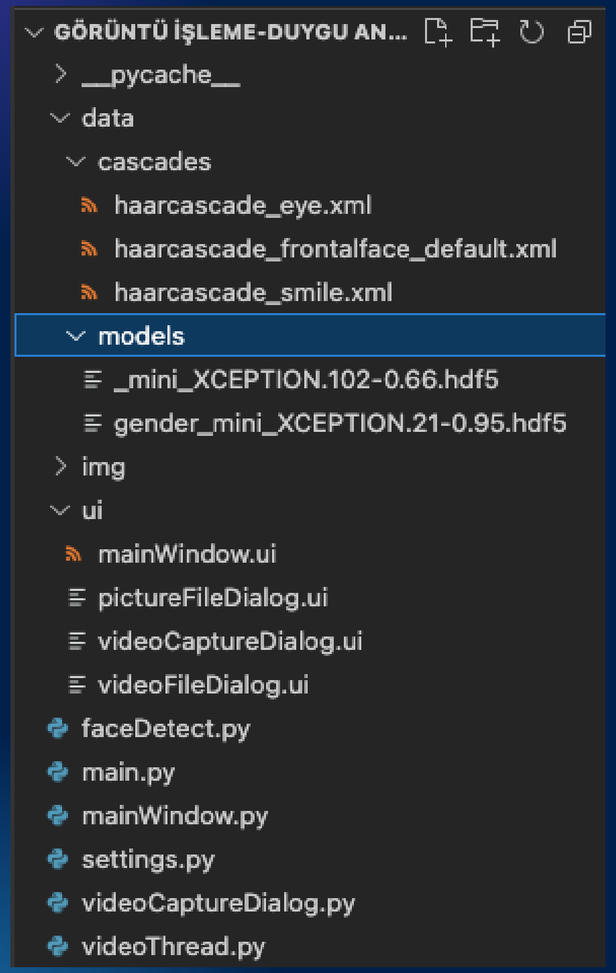
PyQt5

●PyQt5, Python programlama dili için bir GUI (Grafiksel Kullanıcı Arayüzü) kütüphanesidir. PyQt5, Qt5 GUI toolkit'inin Python için bir arabirimidir. Qt5, C++ diliyle yazılmış bir GUI toolkitidir ve PyQt5, Qt5 API'sinin Python için bir sarmalayıcısıdır.

●PyQt5 ayrıca, QtDesigner gibi bir GUI tasarım aracı da içerir. Bu araç, görsel bir arayüz tasarlamanıza ve ardından bu tasarımı PyQt5 koduna dönüştürmenize olanak tanır.



●Projemizde duygu durum analizi için kameradan alınan görüntü anlık kullanılmaktadır.



Projemizde yer alan dosyalardan kısaca bahsedecek olursak;

Main.py

Seçim yapmak için bir ana sayfa hazırladık.

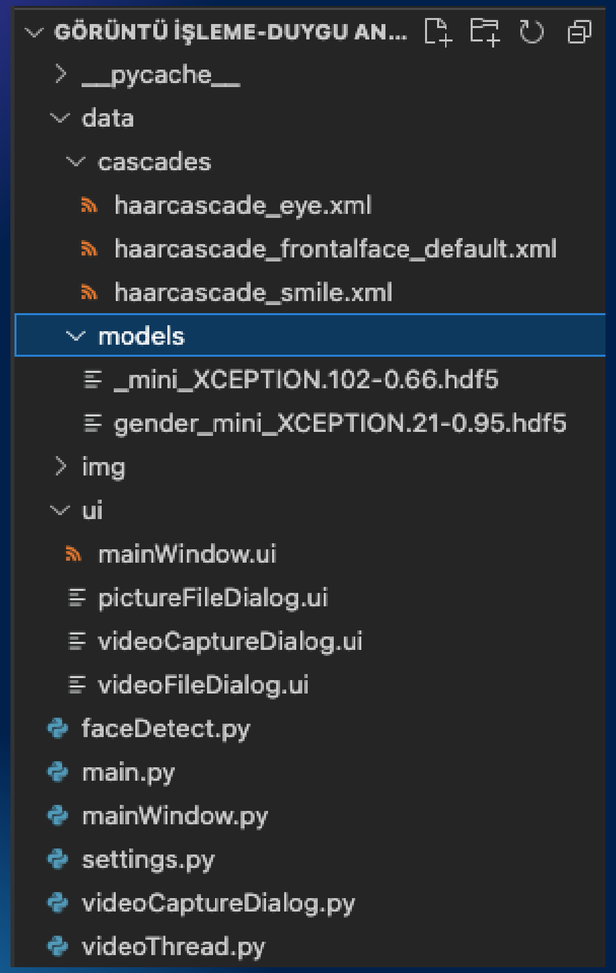
- **Data klasörü**

Modeller

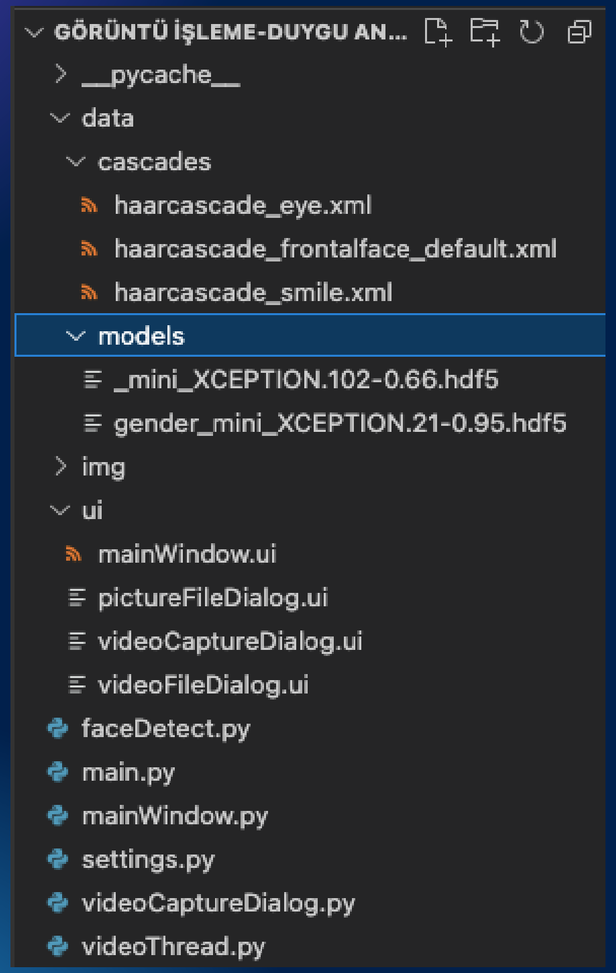
Cascades (OpenCV tarafından kullanılan nesne tespiti için eğitilmiş önceden tanımlanmış sınıflandırıcıları ifade eder.)

- **Settings.py**

Programın tamında geçerli olacak ayarları da bir .py dosyasında belirttik.



Genel olarak, .ui dosyaları, Qt Designer ile oluşturulan GUI tasarımlarını saklamak ve PyQt veya PySide gibi kütüphanelerle entegre edilerek Python ile GUI uygulamaları oluşturmak için kullanılır.



KAMERA İŞLEMİ İÇİN

- VideoCaptureDialog

ANA İŞLEMLER İÇİN

- FaceDetect,
- VideoThread,

KAMERA İŞLEMİ İÇİN

- **VideoCaptureDialog**

```python
def __init__(self):
    super(VideoCaptureDialog, self).__init__()

    self.Window = uic.loadUi('ui/videoCaptureDialog.ui', self)
    self.setWindowTitle('Video yakala')

    self.setFixedSize(self.size())

    self.videoThread = VideoThread()
    self.videoThread.changePixmap.connect(self.setImage)
    self.videoThread.setTerminationEnabled(True)

    self.Window.checkBoxGender.setVisible(True)
    self.Window.checkBoxFemale.setVisible(False)
    self.Window.checkBoxMale.setVisible(False)
    self.Window.checkBoxEmotion.setVisible(True)
    self.Window.checkBoxAngry.setVisible(False)
    self.Window.checkBoxDisgust.setVisible(False)
    self.Window.checkBoxScared.setVisible(False)
    self.Window.checkBoxHappy.setVisible(False)
    self.Window.checkBoxSad.setVisible(False)
    self.Window.checkBoxSurprised.setVisible(False)
    self.Window.checkBoxNeutral.setVisible(False)
    self.Window.pushButtonStart.setVisible(True)

    self.checkBoxGender.stateChanged.connect(self.onGenderChanged)
    self.checkBoxEmotion.stateChanged.connect(self.onEmotionChanged)
```

```python
@pyqtSlot()
def on_pushButtonStart_clicked(self):
    self.Window.labelVideoFrame.setText('Başlatılıyor, lütfen bekleyin...')

    option_genders = []
    if self.Window.checkBoxGender.isChecked():
        if self.Window.checkBoxFemale.isChecked():
            option_genders.append(0)
        if self.Window.checkBoxMale.isChecked():
            option_genders.append(1)

    option_emotions = []
    if self.Window.checkBoxEmotion.isChecked():
        if self.Window.checkBoxAngry.isChecked():
            option_emotions.append(0)
        if self.Window.checkBoxDisgust.isChecked():
            option_emotions.append(1)
        if self.Window.checkBoxScared.isChecked():
            option_emotions.append(2)
        if self.Window.checkBoxHappy.isChecked():
            option_emotions.append(3)
        if self.Window.checkBoxSad.isChecked():
            option_emotions.append(4)
        if self.Window.checkBoxSurprised.isChecked():
            option_emotions.append(5)
        if self.Window.checkBoxNeutral.isChecked():
            option_emotions.append(6)

    self.videoThread.options = {
        'genders': option_genders,
        'emotions': option_emotions,
    }
```

```python
@pyqtSlot()
def onGenderChanged(self):
    if self.Window.checkBoxGender.isChecked():
        self.Window.checkBoxFemale.setVisible(True)
        self.Window.checkBoxMale.setVisible(True)
    else:
        self.Window.checkBoxFemale.setVisible(False)
        self.Window.checkBoxMale.setVisible(False)

@pyqtSlot()
def onEmotionChanged(self):
    if self.Window.checkBoxEmotion.isChecked():
        self.Window.checkBoxAngry.setVisible(True)
        self.Window.checkBoxDisgust.setVisible(True)
        self.Window.checkBoxScared.setVisible(True)
        self.Window.checkBoxHappy.setVisible(True)
        self.Window.checkBoxSad.setVisible(True)
        self.Window.checkBoxSurprised.setVisible(True)
        self.Window.checkBoxNeutral.setVisible(True)
    else:
        self.Window.checkBoxAngry.setVisible(False)
        self.Window.checkBoxDisgust.setVisible(False)
        self.Window.checkBoxScared.setVisible(False)
        self.Window.checkBoxHappy.setVisible(False)
        self.Window.checkBoxSad.setVisible(False)
        self.Window.checkBoxSurprised.setVisible(False)
        self.Window.checkBoxNeutral.setVisible(False)
```

ANA İŞLEMLER İÇİN

- **FaceDetect**

```python
class FaceDetect:
    options = {
        'genders': [],
        'emotions': [],
    }

    def __init__(self):
        # Yüz tespiti için gerekli XML verisi
        self.face_cascade = cv2.CascadeClassifier(settings.DATA['cascades']['face'])

        # Göz tespiti için gerekli XML verisi
        self.eye_cascade = cv2.CascadeClassifier(settings.DATA['cascades']['eye'])

        # Gülümseme tespiti için gerekli XML verisi
        self.smile_cascade = cv2.CascadeClassifier(settings.DATA['cascades']['smile'])

        # Duygu durum modeli
        self.emotion_model = load_model(settings.DATA['models']['emotion'], compile=False)

        # Cinsiyet modeli
        self.gender_model = load_model(settings.DATA['models']['gender'], compile=False)
```

```python
def run(self, frame, flip=False, options=None, _eyes=False, _smiles=False):
    # Görüntüyü aynalıyoruz
    if options is None:
        options = self.options

    if flip is True:
        frame = cv2.flip(frame, 1)

    # Görüntüyü gri tona çeviriyoruz
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Oluşturulan gri görüntüden yüzleri tespit ediyoruz
    faces = self.face_cascade.detectMultiScale(gray, 1.3, 5)

    label = {}

    # Her bulunan yüz üzerinde işlem yapmak için bir döngü kullanıyoruz
    if len(faces) > 0:
        for (x, y, w, h) in faces:
            # Bulunan yüzün oldugu kareyi siyah beyaz yapıyoruz
            roi_gray = gray[y:y + h, x:x + w]

            # Bulunun yüzün olduğu kareyi renkli yapıyoruz
            roi_color = frame[y:y + h, x:x + w]

            # Bulunan yüz verisini ölçeklendirip numpy ve keras ile işliyoru
            roi_gray = cv2.resize(roi_gray, (64, 64))
            roi_gray = roi_gray.astype('float') / 255.0
            roi_gray = img_to_array(roi_gray)
            roi_gray = np.expand_dims(roi_gray, axis=0)
```

ANA İŞLEMLER İÇİN

- **FaceDetect**

```python
# Cinsiyeti analiz ediyoruz
gender_predict = self.gender_model.predict(roi_gray)[0]
# En yüksek skoru alan cinsiyet
gender_probability = np.max(gender_predict)
# Tüm cinsiyet verileri
genders = []
for (i, (gender, probability)) in enumerate(zip(settings.GENDERS, gender_predict)):
    genders.append({
        'title': gender,
        'percent': round(probability * 100, 2),
    })

# Duygu durumunu analiz ediyoruz
emotion_predict = self.emotion_model.predict(roi_gray)[0]
# En yüksek skoru alan duygu durumu
emotion_probability = np.max(emotion_predict)
# Tüm duygu durum verileri
emotions = []
for (i, (emotion, probability)) in enumerate(zip(settings.EMOTIONS, emotion_predict)):
    emotions.append({
        'title': emotion,
        'percent': round(probability * 100, 2),
    })
```

```python
# Elde edilen skorları bir nesneye atıyoruz
label = {
    'gender': {
        'title': settings.GENDERS[gender_predict.argmax()],
        'percent': round(gender_probability * 100, 2),
    },
    'emotion': {
        'title': settings.EMOTIONS[emotion_predict.argmax()],
        'percent': round(emotion_probability * 100, 2),
    },
    'genders': genders,
    'emotions': emotions,
}

# Bulunan yüze uygulayacağımız çerçeveyi oluşturuyoruz
if gender_predict.argmax() in options['genders'] or emotion_predict.argmax() in options['emotions']:
    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 1)

# Cinsiyet verisini görüntüye yazdırıyoruz
if gender_predict.argmax() in options['genders']:
    cv2.putText(frame, str(label['gender']['title']) + ' (%' + str(label['gender']['percent']) + ')',
                (x, y - 30), cv2.FONT_HERSHEY_SIMPLEX, .6, (0, 0, 255), 2)

# Duygu verisini görüntüye yazdırıyoruz
if emotion_predict.argmax() in options['emotions']:
    cv2.putText(frame, str(label['emotion']['title']) + ' (%' + str(label['emotion']['percent']) + ')',
                (x, y - 15), cv2.FONT_HERSHEY_SIMPLEX, .6, (0, 0, 255), 2)
```

ANA İŞLEMLER İÇİN

- **FaceDetect**

```
        # print(label)

        if _eyes:
            # Bulunan yüzdeki gözleri tespit ediyoruz
            eyes = self.eye_cascade.detectMultiScale(roi_gray)
            # Bulunan gözlerde işlem yapmak için bir döngü kullanıyoruz
            for (ex, ey, ew, eh) in eyes:
                # Bulunan gözleri çerçeve içine alıyoruz
                cv2.rectangle(roi_color, (ex, ey), (ex + ew, ey + eh), (0, 255, 0), 2)

        if _smiles:
            # Bulunan yüzde gülücük arıyoruz
            smiles = self.smile_cascade.detectMultiScale(roi_gray, 1.7, 22)
            # Bulunan gülücük verisininde işlem yapmak için bir döngü kullanıyoruz
            for (sx, sy, sw, sh) in smiles:
                # Bulunan gülücüği çerçeve içine alıyoruz
                cv2.rectangle(roi_color, (sx, sy), (sx + sw, sy + sh), (0, 0, 255), 2)

# Oluşturulan yeni görüntüyü çıktı olarak veriyoruz
return {
    'frame': frame,
    'label': label
}
```

ANA İŞLEMLER İÇİN

- VideoThread

```python
from faceDetect import FaceDetect


class VideoThread(QThread):
    changePixmap = pyqtSignal(QImage)
    capture = None
    path = None
    flip = True
    options = {
        'genders': [],
        'emotions': [],
    }

    def setPath(self, path):
        self.path = path

    def setOptions(self, options):
        self.options = options

    def getCapture(self):
        if self.capture is None:
            if self.path is None:
                source = 0
            else:
                source = self.path
                self.flip = False

            self.capture = cv2.VideoCapture(source)

        return self.capture
```

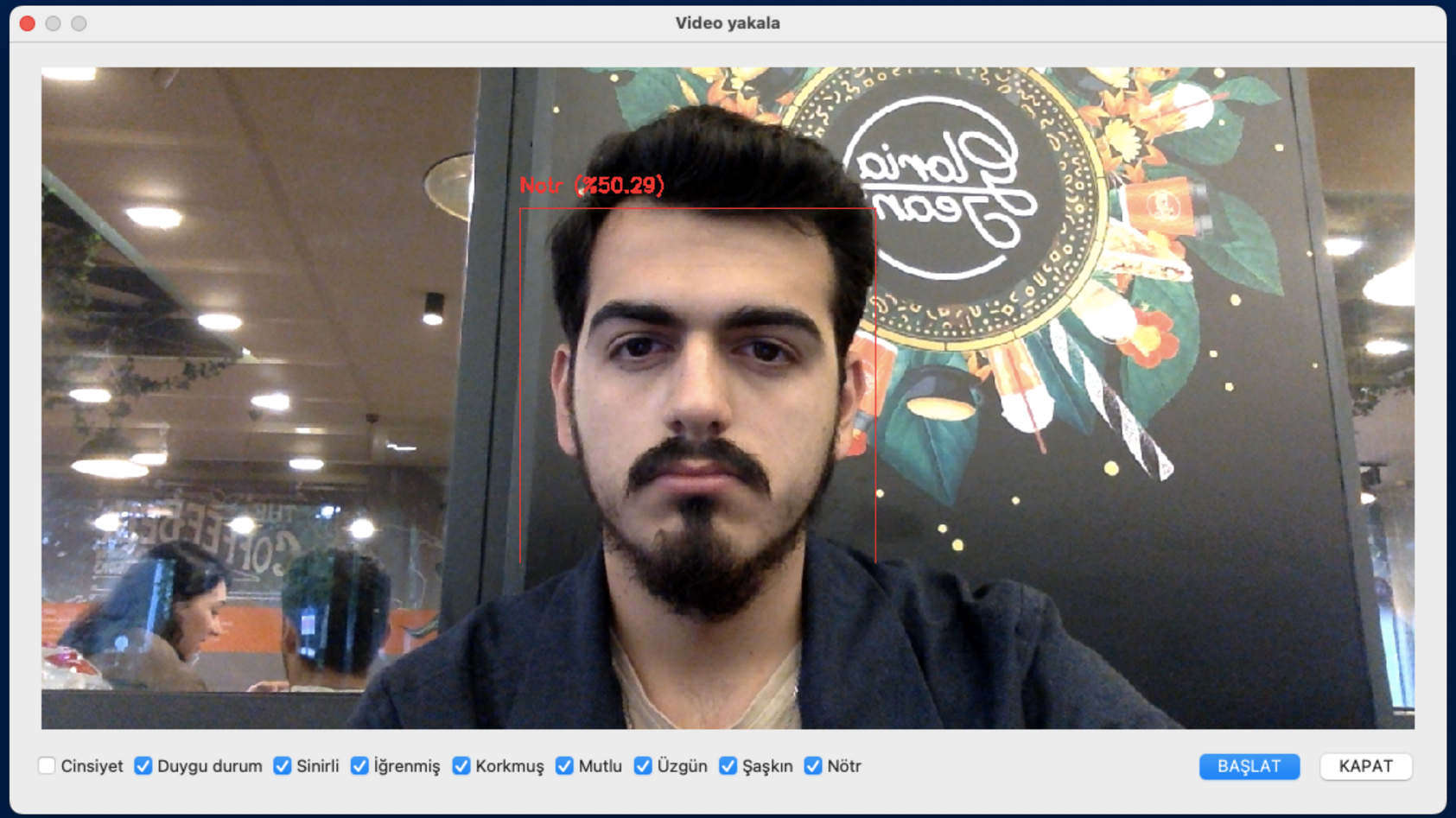
```python
def run(self):
    capture = self.getCapture()
    face_detect = FaceDetect()

    while True:
        # Her döngü yenilenmesinde bir kare alıyoruz
        ret, frame = capture.read()

        if ret:
            # Yüz tanıma sınıfımızı kullanarak görüntüyü alıyoruz
            detect = face_detect.run(frame=frame, flip=self.flip, options=self.options)['frame']

            # Görüntü renklerini düzenliyoruz
            frame = cv2.cvtColor(detect, cv2.COLOR_BGR2RGB)

            # Görüntüyü QT objesine aktarıyoruz
            h, w, ch = frame.shape
            bytesPerLine = ch * w
            convertToQtFormat = QImage(frame, w, h, bytesPerLine, QImage.Format_RGB888)
            p = convertToQtFormat.scaled(800, 600, Qt.KeepAspectRatioByExpanding)
            self.changePixmap.emit(p)
```

ANA İŞLEMLER

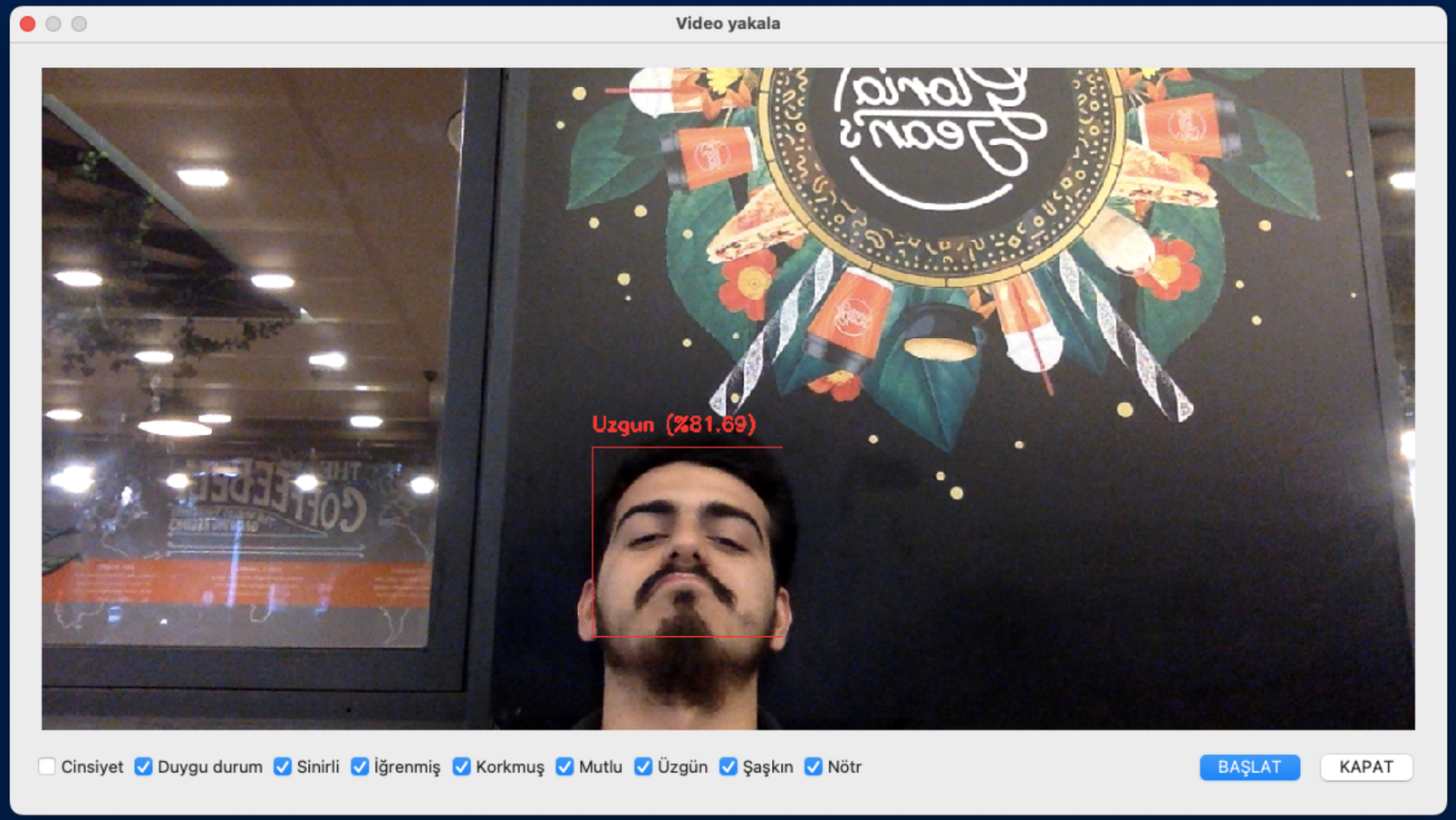
- **settings**

```python
# DataSet
DATA = {
    'cascades': {
        # Yüz tespiti için gerekli XML dosyasının yolu
        'face': os.path.join(BASE_DIRECTORY, 'data', 'cascades', 'haarcascade_frontalface_default.xml'),
        # Göz tespiti için gerekli XML verisi
        'eye': os.path.join(BASE_DIRECTORY, 'data', 'cascades', 'haarcascade_eye.xml'),
        # Gülümseme tespiti için gerekli XML verisi
        'smile': os.path.join(BASE_DIRECTORY, 'data', 'cascades', 'haarcascade_smile.xml'),
    },
    'models': {
        # Yüz ifadesi modeli
        'emotion': os.path.join(BASE_DIRECTORY, 'data', 'models', '_mini_XCEPTION.102-0.66.hdf5'),
        # Cinsiyet modeli
        'gender': os.path.join(BASE_DIRECTORY, 'data', 'models', 'gender_mini_XCEPTION.21-0.95.hdf5'),
    },
}

# Duygu durumları
EMOTIONS = ['Sinirli', 'Igrenmis', 'Korkmus', 'Mutlu', 'Uzgun', 'Saskin', 'Notr']

# Cinsiyetler
GENDERS = ['Kadin', 'Erkek']
```

Kamerada anlık alınan ifadelerin duygu tespit örnekleri

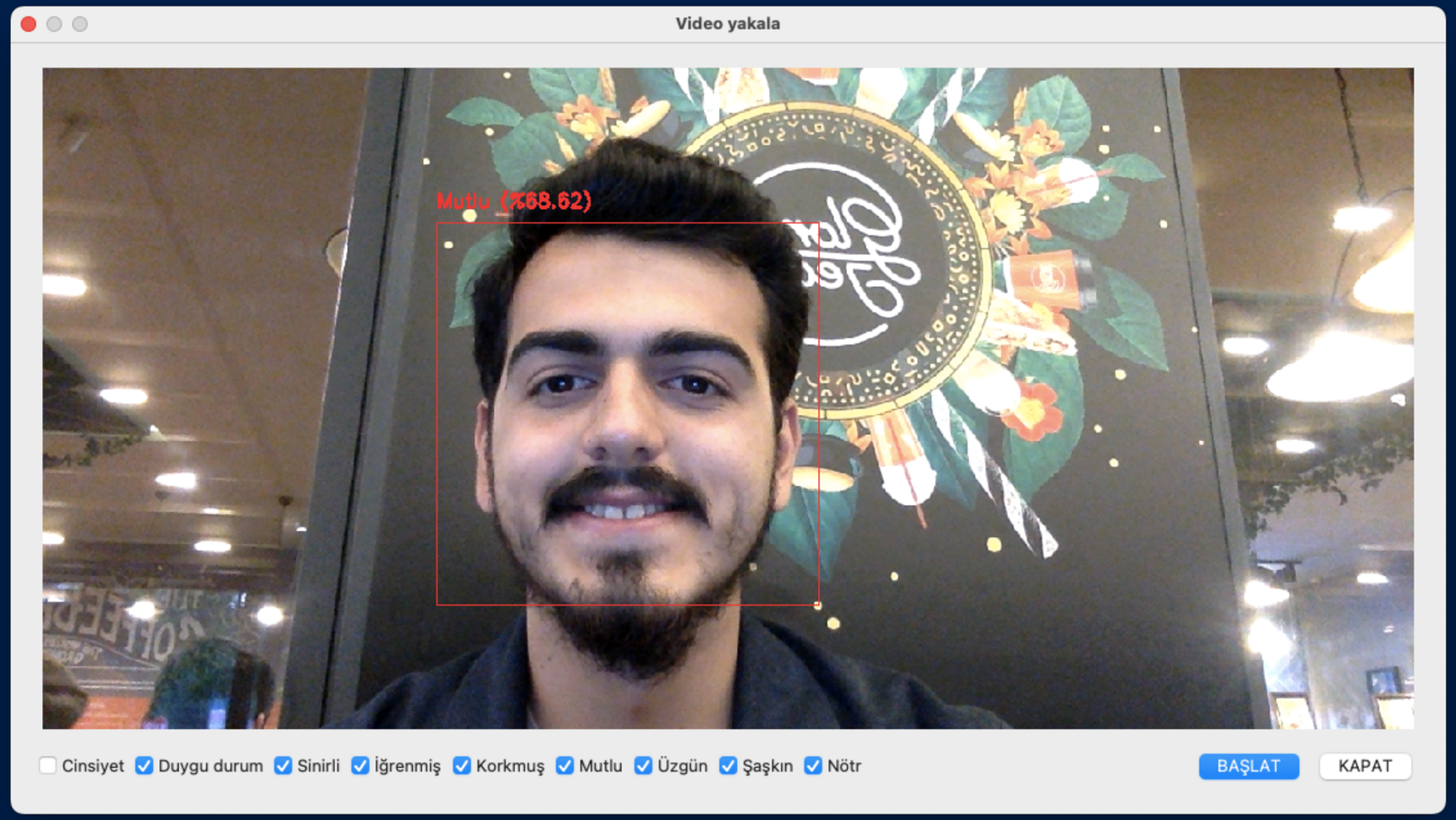


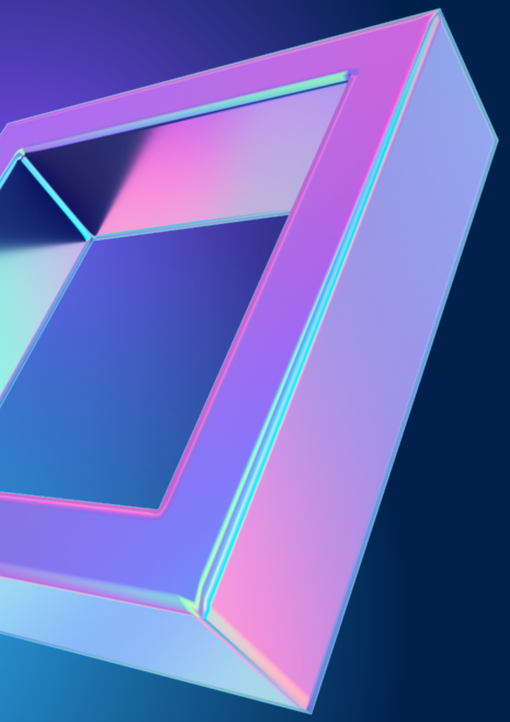
Kamerada anlık alınan ifadelerin duygu tespit örnekleri



Kamerada anlık alınan ifadelerin duygu tespit örnekleri

Kamerada anlık alınan ifadelerin duygu tespit örnekleri





Dinlediğiniz İçin Teşekkür Ederiz

MERVENUR ÇETİN & EBUBEKİR GÜLEN