

PROJECT REPORT

PROJECT NAME:

Sesim Var!

CONTENTS:

Project Name	Page No: 1
Introduction	Page No: 2
Method	Page No: 3
Indication	Page No: 5
Results and Discussion	Page No: 17
Resources	Page No: 19
Additions	Page No: 20

1. INTRODUCTION:

The dictionary meaning of the disabled is someone who has obstructed, obstructed by a certain set of constraints, encountered with an obstacle. In more general terms, diseases or disabilities which come with birth or later can be defined as disabled person, social attitudes and preferences that are restricted in many areas of life as a result of disability. Communication is a process in which messages containing feelings, thoughts and needs are exchanged among individuals. The easiest way for people to express themselves in their communication with each other is to communicate by talking. Unfortunately, speaking and hearing is not the best way to express themselves for people with speaking and hearing impairments (Tonkatz, 2009, page 14)

There are currently 2.5 million people with language and speech difficulties in our country and 270 thousand people with speech impairments. When we look at the structuring of the disabled in our country, it is a pity that when there are too many schools of the visually impaired, schools of the mental retardation and schools of the hearing impairment, on the other hand, speech impediment schools are absent. Of course, it is good to have schools for the other disability groups and increasing their numbers, but there is also a need for schools for speech impairment. Speech impaireds identified with hearing impairments till nowadays, are now being recognized as different group. (<https://eodev.com/gorev/207172>. Access Date: 04.01.2017)

We know that it is not possible to transfer a thought directly from one brain to another. You may have witnessed in some science-fiction films, the scenes in which the transfer from one brain to another through electrodes, cables. However, in real life these stages do not seem possible to be realized, at least for now. (Konrot, 2005)

THE AIM OF THE PROJECT:

- Facilitating the lives of speech-impaired individuals.
- To help speech-impaired individuals to speak out by listening to the sounds they want to remove by studying the camera's instant lip moves and removing them from the computer
- To increase their quality of life.
- To make speech-impaired individuals more social in their life.
- To help increase the sense of independence of the speech impaired person.
- To help individuals with speech impairment develop self-confidence by increasing their belief in their skills and abilities.
- To help overcome the psychological tension experienced by the speech impaired person and the resulting social problems.
- To prevent the speech impaired individual from isolating himself / herself from society and to help him / her connect with the life.
- To give a new perspective to the works have done

Beacause of all these listed purposes, the project's slogan is "**Everyone should hear our voice!**" The project team tried to liberate the communication limits of people with disabilities by following these pathways and aimed at putting the words that the computer perceives in the mouths that the computer perceives as words. The biggest aim of the project is to facilitate the life of speech-impaired individuals and to offer a free and unhindered life to disabled people.

1. METHOD:

Previous studies about speech and language impairments were identified and scanned and information related to this topic was revealed. Later, a different perspective was considered from similar projects, roughly the design of the project was made, and the most suitable ones among these designs were preferred. In addition, keeping the project open for updates is always on one side of the minds of the project team. We have to say that we are always happy to be in the computer world by researching and writing code, we are pleased to have such projects and we are pleased to be able to bring out this project and bring a benefit to the world of disabilities due to our long experience.

In Turkish, 3 different words are used for people with disabilities, so there is a complex disorder. These; disabled, crippled and handicapped people. Although each one has a different meaning, they are used by the general public in the same sense. However, while the word crippled refers to the state of being a sick or missing limb / part in the body, namely, a physiocathal state and the loss of the body's organ, the word disability refers to the reduction of the disabled to be victims while doing basic plans for daily life.

In other words, the word 'disabled' is created for the people who can not benefit from the rights(public transport,education,public servives) that everyone can easily benefit from, not the disability.(Tonkaz,2009.p.14)

The definition of disabled used in research is for those who have lost their physical, mental, emotional and social skills at various levels and can not obey the norms of normal life from birth or later.Based on these data according to the results of the sample survey of 12:29% of Turkey's population is disabled. Due to the population of Turkey in 2002 with 67,844,903 people figure corresponds to 8,338,139 people (Calik, 2004).

What is Speech Disabled?

It is generally accepted that there is a kind of speaking problem in the individual if the speaking is deviating or deviating from the boundary adopted in any environment. In speaking behavior of an individual who has a speech problem, the following characteristics can be multiplied:

- Not speaking loudly enough to be heard.
- If speaking can not be understood easily.
- Disturbing outward appearance of voice or talking.
- A certain voice can not be said as expected.
- Difficulty in talking.
- The lack of accent, melody, rhythm, in speech.
- Syntax, syntax, and so on. deviations.
- Voice and speech that are not appropriate for age, gender, physical development.

As can be seen, it is clear that even with this summary definition, how complicated the language and speech problems are, it requires a separate expertise.

What are the symptoms?

Language and speech problems can be clustered according to their reasons as follows:

- Language and speech problems related to anatomical reasons. (cleft lips or palate, laryngectomy, hearing problems, etc.)
- Language and speech problems due to physiological reasons. (musculoskeletal dystrophy etc.)
- Language and speech problems related to neurological reasons. (CVA, Parkinson's Disease, Cerebral Palsy, etc.)
- Language and speech problems due to biochemical reasons. (Anoxia etc.)
- Language and speech problems related to psychological / psychiatric reasons. (According to some experts, speaking problems such as stuttering are handled in this cluster)
- Language and speech problems related to developmental disruptions. (Delayed language and speech, learning difficulty, reading difficulty, etc.)
- Language and speech problems related to adverse environmental factors. (Delay in language and speech development, etc.)
- Language and speech problems that can not be connected to any cause.
- Tongue and speech problems due to complex reasons. (Mental disability etc.)

Nasıl Tanı Konur?

- Konuşmanın anlaşılır şekilde olmaması,
- Konuşmanın duyulmasında yetersizlik olması,
- Sesin bozuk ve tırmalayıcı olması,
- Sesin çıkarılmasının, ritminin ve vurgularının bozuk olması, • Dil yönünden kelime dağarcığı ve gramer yetersizliklerinin olması,
- Konuşmanın bireyin yaşına ve fiziksel yapısına uygunsuzluğu.

Tedavi Yöntemleri Nelerdir?

Bunun için her konuşma bozukluğu hakkında ayrı ayrı bilgi edinip sorunun çeşidine göre eğitim ve sağaltım yöntemleri uygulamak gerekir. Bu nedenle konuşma bozukluklarından artikülasyon bozukluğu, gecikmiş konuşma, kekemelik ve diğer konuşma engelleri (yabancı dil ve bölgesel konuşma ayrılıkları, damak veya dudak yarıklığı, beyin engeli, afazi, dizartri ve disleksi) ile ilgili temel bilgilerin ve sağaltım etkinliklerin ele alınması uygun görülmektedir.

Konuşma Bozukluğu Nedir?

Konuşma ilindeki sesler, nefesin ses bontlarını titreştirerek ya da titreştirmeden gırtlaktan geçtikten sonra ağız ve burun boşluğunda şekillenmiş halidir. Konuşma seslerini çıkarma işlemine artikülasyon denir.(<http://kucukcekmecehastahane.com/dil-ve-konusmaproblemleri>. Erişim Tarihi: 13.11.2016)

3. BULGULAR

A. PROJE İÇERİĞİ VE SÜREÇ:

Projede kullanılan yazılım teknolojilerinin platform bağımsızlığına sahip olması sayesinde bilgisayar, telefon ve web gibi ortamlarda kullanılabilir ya da bu yazılımı çalıştıracak başka platformlar geliştirilebilir.

Bu projeyi yaparken, sivil savunmada kullanılan insan, araba ve benzeri obje algılama ve görüntü işleme yöntemlerinden yararlanılmıştır.

Projenin geliştirilmelere açık olması, başka insanların güncelleyerek daha iyi bir proje hazırlamasına olanak tanır. Mesela konuşma engelli bir birey kendi isteği üzerine proje değiştirebilir ve daha rahat kullanıma sahip bir uygulama yaratabilir bunu da yayınlayarak herkesin ulaşabilmesini ve başkalarının da proje üzerine yeni eklentiler eklemesine olanak sağlar.

Projenin bu şekilde daha hızlı gelişebileceğini düşünülmektedir.

Projenin yeterli hızda ve performansta olduğu düşünülmektedir fakat başka bir kullanıcı projenin geliştirilebilir olma özelliğini kullanarak bazı algılama ayarlarını değiştirebilir ve daha performanslı bir uygulama ortaya çıkarabilir.

Kullanılan yazılım açık kaynaklı olduğundan kodları kullanılabilir ve kendinize ait yeni bir proje hazırlayabilirsiniz hatta ticari amaçla da kullanabilirsiniz. Projenin en göze çarpan bölümü de bu şekildedir.

B. PROJEDE KULLANILAN TEKNOLOJİLER:

Projede kullanılan yazılım teknolojileri platform bağımsızlığına sahip, bu sayede projeyi istediğiniz bir işletim sisteminde çalıştırabilirsiniz. (Örneğin: MacOS, iOS, Android, Linux, Windows)

1. Programlama Dili:

Projede kullanılan yazılım dili, günümüzde en popüler diller arasında olan, kullanımı kolay, performanslı ve birçok firmanın (Örneğin: Google, Nasa) tercihi olan **Python(Versiyon: 3.5.2)** programlama dilidir.



(Fotoğraf Kaynak: <https://www.python.org/community/logos/>. Erişim Tarihi: 29.12.2016)

Neden Python ?

Python günümüzde oldukça popüler okunması ve kullanması kolay açık kaynak programlama dilidir.

Okuması, yazması oldukça basittir.

Python ile ilgili internette herkese açık ve ücretsiz sayısız kaynak bulunmaktadır.

Dolayısıyla öğrenilmesi kolaydır.

Python ile yapılmış bir sürü projeye internetten rahatça ulaşabilirsiniz.

Python programlama dilinin ne kadar sade ve okunabilir bir dil olduğunu göstermek için, günümüzde popüler olan C++ programlama dili ile küçük bir karşılaştırma yapalım:

C++ programlama dili ile yazılmış basit bir kod:

```
#include <iostream> using
namespace std;
int main()
{
    count << "Merhaba Dünya!" << endl;
    return 0;
}
```

Yukarıdaki C++ ile yazılmış programla aynı görevi gören Python kodu:

```
print("Merhaba Dünya!")
```

Yukarıdaki kodları çalıştırdığımızda ekranımızda “Merhaba Dünya!” yazacaktır.

Proje raporunun ilerleyen bölümlerinde detaylıca değineceğimiz ve görüntüyü işlerken kullanacağımız OpenCV kütüphanesi ile uyumlu ve kullanımı kolay olması Python dilini seçmemizi sağlayan büyük bir etkidir.

Projede kullanılan, kodlar ve açıklamaları raporda, ek 12 “Proje kodları” bölümünde bulunmaktadır.

2. OpenCV ile Görüntü İşleme:

Projemizin görüntü işleme bölümünde, günümüzde oldukça popüler ve kullanımı basit olan OpenCV kütüphanesi kullanılmıştır.

OpenCv INTEL tarafından 1999’da C ve C++ dilleri kullanılarak geliştirilmeye başlamıştır. Günümüzde ise Google, Itseez, Nvidia gibi şirket ve toplulukların desteği ile gelişim süreci devam etmektedir.

Açık kaynak kodlu bir görüntü işleme kütüphanesidir. Bu sayede başka kişiler tarafından geliştirilebilir yapıdadır.

OpenCV adı Open Source(Açık Kaynak Kodu) cümlesinin ilk harfi ve Computer Vision (Bilgisayar Görüşü) kelimelerinin baş harflerinden oluşur.

OpenCV ile yapılabileceklerin sınırı yoktur mesela yüzleri, insan vücutlarını, hareketli veya hareketsiz nesneleri algılayabilir ve bunlarla işlemler yapılabilmektedir. Mesela bir insanın boyunu ölçebilir, hız limitini aşan araçların plakalarını öğrenebilir ve daha birçok şey yapılabilmektedir.

Projemizde OpenCV, konuşma engelli bireyin ağız şeklini anlık olarak algılayarak çıkarmak istedikleri sesi bulmamızı ve daha sonra bu sesi bilgisayardan çıkarmamızı sağlamıştır.

Ağızının Yapısını Algılamak:

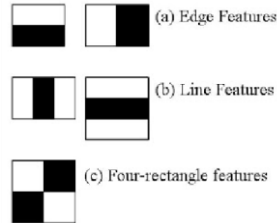
Konuşma engelli kişinin ağız yapısını algılamak için önce kişinin yüzünü algılamak gerekmektedir. Yüz algılamak için çeşitli algoritmalar bulunmaktadır. Bunlardan en popüler olan ve doğru bir şekilde yüz bulma olasılığının en fazla olduğu “Haar Cascade” yöntemini seçilmiştir.

Haar Cascade Yöntemi:

Bu yöntem günümüzde oldukça popülerdir çünkü bir yüzü doğru şekilde algılama olasılığı çok yüksektir ve hata payı oldukça azdır. OpenCV içerisinde hazır bir fonksiyonu bulunduğu için bu yöntemin OpenCV ile kullanımı kolaydır.

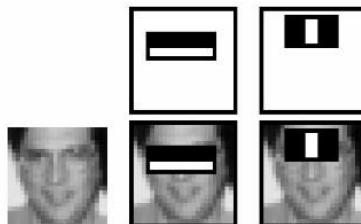
En basit haliyle bu yöntemi şu şekilde anlatabiliriz:

Program aşağıdaki fotoğraflardaki çerçeveleri, içerisinde nesne araması yapılacak fotoğrafın içerisine birçok yere yerleştiriyor. Daha sonra çerçevenin, beyaz bölgesinde kalan toplam piksel değerleri; siyah bölgesinde kalan toplam piksel değerlerinden çıkartılıyor. Eğitilmiş sınıflandırıcı içerisindeki (Eğitilmiş sınıflandırıcı konusuna raporun ileriki bölümlerinde değinilmiştir.) oranlarıyla karşılaştırarak orada nesne olup olmadığı algılanabilmektedir. Günümüzde bu işlemin daha hızlı ve doğru sonuçlar verebilmesi için “Haar Cascade” yöntemi üzerine belirli kütüphaneler ve algoritmalar da uygulanabilir.



(Fotoğraf kaynak: http://docs.opencv.org/3.1.0/haar_features.jpg. Erişim Tarihi: 25.12.2016)

Bu çerçevelerin, örnek bir yüz fotoğrafı üzerine yerleştirilmiş hali:



(Fotoğraf kaynak: <http://docs.opencv.org/3.1.0/haar.png>. Erişim Tarihi: 25.12.2016)

Eğitilmiş Sınıflandırıcı Nedir ?

OpenCV algoritmasının nesneleri bulabilmesi için daha önceden içerisinde algılanması istenen nesnenin bulunduğu (Pozitif) ve bulunmadığı (Negatif) binlerce fotoğrafı incelemesi ve bu fotoğrafların “Haar Cascade” yöntemi sonucunda ki piksel değerlerini bilmesi gerekmektedir. Bu duruma algoritma eğitimi denilmektedir.

Algoritmayı eğitmek için ise şöyle bir yöntem izlenir:

Binlerce, içerisinde algılanması istenen nesnenin bulunduğu (Pozitif) ve bulunmadığı (Negatif) fotoğraflar ve nesnenin bulunduğu fotoğraflardaki nesnenin fotoğraf içerisinde bulunduğu konumları, OpenCV fonksiyonlarıyla incelenir ve belirli bir eğitilmiş sınıflandırıcı dosyalarında veriler tutulur. Bu süreçte, nesne algılama algoritmanın en doğru sonuçları verebilmesi için çok fazla fotoğrafla eğitilmesi gerekmektedir. Bazen bu eğitim süresi kullandığınız bilgisayara bağlı olarak haftalarca ya da aylarca sürebilmektedir.

Eğitilmiş sınıflandırıcı dosyaları xml dosya formatında tutulmaktadır.

Projede algılanmak istenen ağız çoğu nesneye göre algılanması oldukça zor olan bir nesnedir ve ağız bulmak için olan algoritma eğitim süresi de dolayısıyla oldukça uzundur.

OpenCV içerisindeki algoritmamızı eğitilmiş sınıflandırıcı dosyalarımızla çalıştırdığımızda bize, nesnenin başlama noktasını x-y koordinatı şeklinde değerler ile birlikte nesnenin genişliğinin ve yüksekliğinin değerlerini dönmektedir. Bizde bu değerleri kullanarak fotoğraftaki yüzleri algılayabiliyoruz.

Bu işlemde kod bu şekildedir:

```
# Yüzümüzü bulacak OpenCV algoritmamızla çalıştıracağımız eğitilmiş dosyalarımızı  
belirliyoruz: y u z _ c a s c a d e = c v 2 . C a s c a d e C l a s s i f i e r ( ' d a t a / h a a r c a s c  
a d e s / haarcascade_frontalface_default.xml')  
  
# Görüntüyü alacağımız kameramızı tanımlıyoruz:
```

```

cap = cv2.VideoCapture(0)

# Tanımladığımız kameradan bir fotoğraf çekiyoruz ve alıyoruz:
ret, img = cap.read()

# Eğitilmiş sınıflandırıcı dosyamızla OpenCV nesne bulma algoritmamızı çalıştırıyoruz:
# Eğer yüzler bulunursa yüzlerin konumlarını fonksiyonumuz bize veriyor:
yuzler = yuz_cascade.detectMultiScale(img, 1.3, 5)

# Gelen yüzler içerisinde geziyoruz:      for
(x,y,w,h) in yuzler[0:kac_yuz_algilama]:

    # Yüzün bulunduğu alanı opencv kütüphanemiz ile bir dörtgen içine alıyoruz:
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)

    # Yüzün konumunu kaydediyoruz:
    # İleride göz ve ağızı bulmak için bu verileri kullanacağız:
    yuz = img[y:y+h, x:x+w]

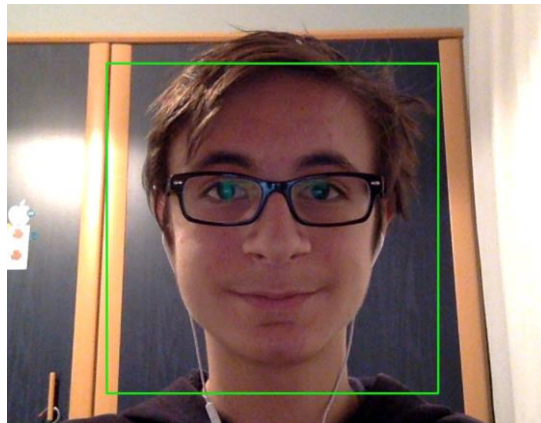
# Yüzümüzün dörtgen içine alınmış fotoğrafını yeni bir pencerede kullanıcıya gösteriyoruz:
cv2.imshow('Sesim Var!',img)

#Kamerayı kapatıyoruz: cap.release()

# Fotoğrafi kullanıcıya gösterirken kullandığımız uygulama penceresini kapatıyoruz:
cv2.destroyAllWindows()

```

Bu kodu çalıştırdığımızda ise aşağıdaki fotoğrafta gördüğümüz gibi yüz başarılı bir şekilde algılanmış ve yeşil dörtgen içine alınmış durumdadır.



Şimdi ise bu yüz içerisinde kişinin ağızını bulmamız gerekmektedir. Bunun için yine yüz algılama da kullandığımız “Haar Cascade” yöntemi kullanılacaktır.

Ağzın daha doğru bir şekilde algılanması için sadece yüz içinde ve gözlerin altındaki bölgeye bakılmasının yeterli olabileceği düşündüğümüzden gözlerin de bulunması gerektiğine karar verilmiştir:

Gözlerin algılanması için kod şu şekildedir:

```
# OpenCV algoritmamızla çalıştıracağımız eğitilmiş sınıflandırıcı dosyalarımızı belirliyoruz:
goz_cascade = cv2.CascadeClassifier('data/haarcascades/haarcascade_eye.xml')

# Eğitilmiş sınıflandırıcı dosyamızla OpenCV nesne bulma algoritmamızı çalıştırıyoruz.
# Ve bulunan göz koordinatlarını kaydediyoruz:
gozler = goz_cascade.detectMultiScale(yuz)
# Ağızı bulurken kullanmamız gereken bir değişken oluşturuyoruz:
göz_lerin_alti_y = 0
# Bulunan gözler içerisinde geziyoruz:
for (ex,ey,ew,eh) in gozler[0:2]:

    # Gözün bulunduğu alanı opencv kütüphanemiz ile bir dörtgen içine alıyoruz:
    cv2.rectangle(yuz,(ex,ey),(ex+ew,ey+eh),(0,255,255),2)
    # İleride kullanmak için aşağıda olan gözün altının y koordinatını kaydediyoruz:
    if göz_lerin_alti_y > ey+eh:
        göz_lerin_alti_y = ey+eh
```

Şimdi ise ağzın algılanması için kod şu şekildedir:

```
# OpenCV algoritmamızla çalıştıracağımız eğitilmiş dosyalarımızı belirliyoruz:
agiz_cascade = cv2.CascadeClassifier('data/haarcascades/mouth2.xml')

# Ağız sadece gözlerin altında olduğu için gözlerin üstünü aramamız performansı düşürebilir.
# Bu yüzden ağızı sadece gözlerin altında aramalıyız:
goz_alti_yuz = img[y+göz_lerin_alti_y:y+h, x:x+w]

# Eğitilmiş sınıflandırıcı dosyamızla OpenCV nesne bulma algoritmamızı çalıştırıyoruz.
# Ve bulunan ağız koordinatlarını kaydediyoruz:
agizlar = agiz_cascade.detectMultiScale(goz_alti_yuz)
```

```

# Bulunan ağızlar içerisinde geziyoruz:
for (mx,my,mw,mh) in agizlar[0:1]:

    # Eğer gözler algılanmadıysa ve ağız gözün üstündeyse:
    if len(gozler) < 1 or göz_lerin_alti_y > my:

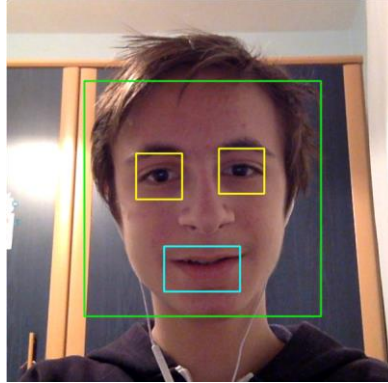
        # Ağız hatalı bulunmuştur:
        # Eğer varsa diğer bulunan ağıza geçilir.
        continue

    # Ağızı dörtgen içine alıyoruz:
    cv2.rectangle(goz_alti_yuz,(mx,my),(mx+mw,my+mh),(255,255,0),2)

    # İleride kullanmak için ağız genişliğini ve yüksekliğini kaydediyoruz:
    # agiz = [Genişlik, Yükseklik]
    agiz = [int((mx+mw)-mx),int((my+mh)-my)]

```

Kod çalıştırıldığında ise aşağıdaki fotoğrafta görüleceği üzere istenilen sonucun başarıyla alındığı görülmektedir:



Ağız boyutları doğru bir şekilde alındığına göre, bu şekle bakarak sinir ağları ile bu ağız şekline uygun olan daha sonra seslendireceğimiz harfin alınması gerekmektedir. “Sinir Ağları Ve Karar Ağaçları” başlığı altında bu konuya değinilecektir.

3. Sinir Ağları Ve Karar Ağaçları:

Projemizde sinir ağları ve karar ağaçları ağız yapısından harf tahmini yapmak için kullanılmıştır.

Sinir ağları ve karar ağaçları “sklearn” kütüphanesi kullanılarak hazırlanmıştır. Sklearn kütüphanesi oldukça popüler bir yapay zeka kütüphanesidir.

Bu projede sinir ağları ve karar ağaçları daha önceki eğitim verisinden aldığı deneyimler ile ağız yapısına en uygun harfi seçmek için kullanılır bu sayede daha önce hiç görülmeyen bir ağız yapısı algılandığında hangi harfi ifade ettiği tahmin edilebilir.

Sinir ağırları sınıflandırma işleminde oldukça iyi işlemektedir burada derin öğrenme yöntemleri sinir ağırlarımızı güçlendirir.

Karar ağaçları ise az veride oldukça iyi performans vermektedir uygulamamızın kullanıcı odaklı çalışma bölümünde karar ağaçları daha verimli çalışır.

Sinir ağırları ve karar ağaçları program kullanım aşamasında iken kendini yeni verilerle tekrardan eğitebilmektedir bu sayede hatalarını daha iyi düzelte bilir.

Yazdığımız sinir ağırları(siniraglari.py) ve karar ağaçları(harfogren.py) kodları raporumuzun **EK 12** bölümünde, tüm satırların açıklamaları ve tercih etme nedenlerimiz ile bulunmaktadır.

4. Database:

Projede kullanılan SQLite, günümüzde oldukça popüler, kullandığımız programlama dili Python ile kullanımı oldukça kolay olan, açık kaynak kodlu, C ve C++ dilleri ile geliştirilmiş, MacOS, Linux ve Windows platformlarında çalışabilen SQL veritabanı motorudur.



(Fotoğraf kaynak: <http://logonoid.com/images/sqlite-logo.png>. Erişim Tarihi: 25.12.2016)

Database projede, kullanıcıların ağız verilerini ve bu ağız verilerine göre harfleri tablo şeklinde tutmaktadır. Bu şekilde program kapatıldığında ve yeniden açıldığında kullanıcı, ağız ve harf verileri silinmemiş olur.

Projede ağız bilgileri (Ağız yüksekliği ve genişliği) alındıktan sonra SQLite database sorgusu yapıyor ve ağız bilgilerine uygun harf, programa iletiliyor.

Kullanıcı ağız ve harf bilgilerine Database'mizden rahat bir şekilde ulaşabilmek için bir Python programı yazılmasına karar verdik. Bu şekilde veri alışverişi ve database bağlantısı daha rahat bir şekilde yapılmaktadır. Yazdığımız bu programın adına "data_islemler" dedik. Bu programın kodları "data_islemler.py" dosyası içerisinde tutulmaktadır.

"data_islemler" programını yazma amaçlarımızdan bir diğeri ise SQLite database kullanan diğer programlara da dahil edilebilmesini sağlamak. Bu şekilde başka projelerde yeniden yazılmasına gerek kalmayacak ve bu konuda programlar hazırlayan başka kişilerde programlarına, yaptığımız bu programı dahil edebilecekler.

Python ile yazılan “data_islemler” programının kodları ve açıklamaları şu şekildedir:

```
# SQLite kütüphanemizi dahil ediyoruz: import
sqlite3

# Database bağlantısı kurmak: def
set_sql_connect(database_name):
return sqlite3.connect(database_name)

# Database Cursor ayarlamak: def
set_sql_cursor(database_connect):
return database_connect.cursor()

# Database bağlantısı kurup Cursor ayarlamak: def
set_connect_and_cursor():
    vt = set_sql_connect('data/database/database.sqlite')
db = set_sql_cursor(vt)    return vt, db

# Database bağlantısını kapatmak için:
def close_connect(vt):    if vt:
    vt.commit()
vt.close

# Database'de yaratılmamışsa yeni tablo yaratmak için:
def tablo_yarat(table_name, columns):    vt, db =
set_connect_and_cursor()
    db.execute("CREATE TABLE IF NOT EXISTS {0} ({1})".format(table_name, columns))
close_connect(vt)

# Database'den veri almak için: def
veri_al(sql_komut):    vt, db =
set_connect_and_cursor()
db.execute(sql_komut)    gelen_veri
= db.fetchall()    close_connect(vt)
```

```

return gelen_veri

# Database'e veri eklemek için: def
data_ekle(table, eklenecek_sutun, eklenecek):
vt, db = set_connect_and_cursor()
db.execute("INSERT INTO '{0}'({1}) VALUES {2}".format(table, eklenecek_sutun,
eklenecek))
# eklenecek_sutun örnek: 'sütun1','sütun2'
# eklenecek örnek: data1, data2
close_connect(vt)

# Database'deki verileri güncellemek
için: def data_guncelle(table, nerden,
nasil): vt, db =
set_connect_and_cursor()
db.execute("UPDATE {0} SET {2} WHERE {1}".format(table, nerden, nasil))
close_connect(vt)

```

Projenin ana kodlarına “data_islemler” kodlarını aşağıdaki kod satırı ile dahil ediyoruz:

```
import data_islem
```

Şimdi ise “data_islemler” içerisinde bir fonksiyonu ana programdan nasıl çalıştırılacağına göz atarsak:

Örneğin database.sqlite dosyasına bağlanalım:

```
data_islemler.set_sql_connect('database.sqlite')
```

Şimdi ise “data_islemler” kullanarak ağız şekline göre kullanıcının ifade etmek istediği harfi Database’den çekecek olan fonksiyona bakalım:

```

# “harf_al” adında iki argüman alan bir fonksiyon oluşturuyoruz:
def harf_al(kullanici_adi, agiz_data):

    # Kullanıcı adı “kullanici_adi” değişkeni ile string tipinde fonksiyonumuza gelir.
    # “agiz_data”, [Genişlik, Yükseklik] şeklinde listenin içinde fonksiyonumuza gelir.

    # “data_islemler” dosyamızdan “veri_al” fonksiyonunu kullanarak kullanıcı adı ve ağız
yapısına uygun harfi alıyoruz:
    # Harf char tipinde “gelen_veri” değişkenine atanıyor:

```

```
gelen_veri = data_islem.veri_al("SELECT harf FROM agiz_harf WHERE kullanıcı='{0}'  
AND agiz_genislik={1} AND agiz_yukseklik={2}".format(kullanici_adi, agiz_data[0],  
agiz_data[1]))  
  
# Fonksiyonumuz "gelen_veri" değişkenini döndürür: return gelen_veri
```

Eğer kullanıcının ağız bilgilerinin Database’de bir harf karşılığı yoksa boş liste dönecektir.

Kullanıcının ağız bilgilerini, ifade etmek istediği harf ile Database’e kaydetmek için ise şöyle bir fonksiyon yazılmıştır:

```
# "data_kayit" adında iki argüman alan bir fonksiyon  
oluşturuyoruz: def data_kayit(kullanici_adi, agiz_data):  
print('Anlayamadım :(\\nBu söylediğiniz hangi harf nedir?') #  
Girilecek harfi tutacağımız değişkeni oluşturuyoruz: harf = "  
while 1:  
    # Kullanıcıdan bir harf alınır:  
    harf = input('Söylediğiniz harf: ')  
# Veri girildi mi kontrol edilir: if  
harf != "":  
    # Veriler database'de küçük harf şekline tutuluyor.  
    # Bunun için girilen harf her zaman küçük harf olmalı.  
# Bu yüzden girilen harfin küçük harf karşılığı tutuluyor.  
harf = harf[0].lower()  
    break  
else:  
    print('Hatalı Giriş!')  
# Veri Database'mize kaydoluyor: data_islem.data_ekle('agiz_harf', 'kullanici, harf,  
agiz_genislik, agiz_yukseklik', (kullanici_adi, harf, agiz_data[0], agiz_data[1]))
```

Ağız bilgilerinin karşılık geldiği harf bilindiğine göre şimdi de yazıyı nasıl sese çevirdiğimize bakalım.

5. Seslendirme:

Projedeki seslendirmeler, MacOS işletim sisteminin Terminal (Sürüm 2.7.1) uygulaması ile birlikte gelen say komutunun yeni process olarak çalıştırılmasıyla yapılmıştır. (Multi Process konusuna proje raporunun ileriki bölümlerinde detaylı bir şekilde değinilecektir.) Diğer işletim sistemlerinde de farklı uygulamalar ile benzer şekilde işlem yapılabilir.

Proje, Apple MacBook Pro (Retina, 13-inch, Early 2015) cihazında ve Apple’ın MacOS Sierra (Sürüm 10.12.1) işletim sisteminde hazırlanmıştır.

Seslendirme İçin Multi Process:

Projenin seslendirme bölümünde bir sorunla karşılaşmıştır. Kişinin ifade ettiği sözcükler uzun olunca seslendirme, kullanıcının ağız hareketlerinden ayrılıyor ve ağız ile seslerde senkronize sorunları oluyor.

Kullanıcının ağızının hareketiyle bilgisayardan çıkan seslerin eşzamanlı olabilmesi için ise şöyle bir çözüm düşünülmüştür:

Algılanan veriler, yeni bir process ile arka planda seslendirme programına gidiyor bu şekilde ana program ağız algılayıp harfleri bulurken yeni bir process arka planda ana programı etkilemeden, ana programdan gelen yazıları seslendiriyor. Bu şekilde bilgisayardan sesler, kullanıcının ağız hareketleriyle eşzamanlı olarak çıkıyor. Bu da çıkan seslerin, kullanıcının kendi sesiymiş gibi hissetmesini sağlıyor.

Seslendirme Process'i için yazılan kod ve açıklamaları:

```
# İhtiyacımız olan kütüphaneleri dahil ediyoruz:
import os
import platform
from multiprocessing import Process

def seslendir(text):
    # İşletim sistemini kontrol ediyoruz:
    if(platform.system() == 'Darwin'):
        # Process'e gönderilecek argümanı hazırlıyoruz:
        arg = 'say {0}'.format(text)
        # Process'imizi oluşturuyoruz ve seslendirmeyi başlatıyoruz:
        Process(target=os.system, args=(arg,)).start()
    else:
        # İşletim sistemi uygun değilse hata mesajı gösteriyoruz:
        print('Seslendirme şimdilik sadece Darwin kabuğunda çalışmaktadır.')
    return
```

Projenin ekran görüntülerini raporun ekler kısmında bulunmaktadır.

Projede kullanılan bütün kodları kendim yazmış bulunmaktayım. Ayrıca TÜBİTAK proje sunumu bittikten sonra kendime ait internet sayfamda bu projenin nasıl yapıldığını, nasıl çalıştığını ve nasıl geliştirilebileceğini anlatan bir yazı paylaşmayı düşünüyorum. Bu şekilde başka insanları; bu konuda bilinçlendirebilir, projemizden yararlanmasını ve projemizi geliştirmelerini sağlayabiliriz. Bu proje geliştirilmelere açıktır ve herkes tarafından kullanılabilir.

İnternet sayfama yandaki adresten ulaşabilirsiniz: <http://www.ardamavi.com>

C. PROJE YAPIM SÜRECİ VE YAŞANILANLAR:

21 Eylül - 5 Ekim: Projenin düşünülmesi ve fikirlerin birleşmesi.

5 Ekim - 15 Ekim: Projenin algoritmalarının hazırlanması ve test edilmesi.

15 Ekim - 20 Ekim: Yüz ve göz tanıma algoritmalarının yazılması. 20 Ekim-

2 Kasım: Ağız algılama ve sese çevirme sisteminin oluşturulması.

2 Kasım- 15 Kasım: Proje test edilmesi ve sorunların giderilmesi.

25 Kasım- 25 Aralık: Projenin kodların temizlenmesi ve kodların açıklamalarının eklenmesi.

14 Aralık – 5 Ocak: Projenin ve raporun tamamlanması

Proje Yapım Süreci İş Dağılımı:

- Arda Mavi:

- Proje algoritmalarının hazırlanması.
- Proje programlama.
- Görüntü İşleme.
- Eğitilmiş sınıflandırıcı dosyalarının hazırlanması.
- Database ile data işlemleri.
- Proje raporu yazım sürecinin bir kısmı.

- Zümra Uğur:

- Yazılan programın, konuşma ve işitme engellilere uyarlanması.
- Proje raporu yazım süreci.
- Literatür taraması.
- Kaynak süreci.

4. SONUÇLAR VE TARTIŞMA:

- Bu çalışma ile konuşma engellileri yaşadığı problemlerin en az seviyeye indirilmesi ve onları da sosyal hayata daha kolay adapte edebilmek için yapılan çalışmalara alternatif, ucuz, basit ve kullanışlı yeni bir çalışma üretilmiştir.
- Herhangi bir platformda çalışabilen bu projemiz daha da geliştirilebilir durumdadır.
- Engelliler için yapılan diğer çalışmalarla beraber kullanılabilir.

- Konuşma engelli bireyin ifade etmek istediği sözcükleri, bireyin ağızını algılayarak sese çeviren programımız, konuşma engelli bireyin ağızıyla senkronize bir şekilde çalışarak bireye konuşuyormuş hissini verir.
- Konuşma engelli bireylerin kullandıkları sayılı yöntemlerin popülerlerinden olan işaret ile iletişim yönteminin çevresi tarafından bilinmemesi sonucunda konuşma engellilerin duygusal anlamda rahatsız olmalarını neden olacağından, onları yaşadığı psikolojik gerginlik ve bunun sonucu oluşan sosyal sorunları aşmaya yardımcı olması için üretilmiş bir proje hazırlamış olduğumuza inanıyoruz.
- Üniversiteler, sivil toplum kuruluşları, dernekler, vakıflar, iş dünyası işbirliği sağlayıp konuşma engellilere yönelik kampanyalar ve çalışmalar yapıp bu tür gelişme ve etkinliklerden haberdar olmaları sağlanabilir ve hayatlarına az da olsa kolaylaştıran ve basitleştiren bir fayda olduğu düşündürülebilir.
- Toplumun tüm katmanlarında engellilik bilinci aşılanmalı, engelli olmanın anlık bir şey olduğu söylenmeli; televizyon, radyo, gazete, dergi, el ilanları seminerler, konferanslar vb. yollar kullanılmalı ve bu şekilde daha çok kesime ulaşılmalıdır.
- Tüm üniversiteler arası koordinatörlükler geliştirilip paylaşım platformu oluşturulabilir. Her koordinatör kendi üniversitesinde konuşma engelliler ile ilgili her durumu ve aktiviteyi paylaşabilmelidir. Bu şekilde yardımlaşma ve işbirliği sistemi kurulabilir.
- Konuşma engelli bireylerin toplumla bütünleşmesini, özgüvenlerini geliştirmelerini, sorumluluk duygularını geliştirebilmelerini ve bağımsız bireyler olarak yaşamlarını daha rahat sürdürebilmelerini sağlayacak projeler desteklenmeli ve proje sahiplerine yeterli maddi kaynak sağlanmalıdır.
- Engelli bireylerin toplumumuzda var olduğu gerçeği unutturulmamalı ve sürekli bilinçlendirilme ile ilgili toplantılar yapılmalıdır. Burada en büyük görev engelli dernek ve vakıflarına düşmektedir.
- Engelli istihdamı kamuda ve özel sektörde arttırılmalıdır.
- Engellilere yönelik ayrımcı davranışların görülmesinde en büyük etken bu konuda halkın bilinçsizliği ve yanlış hareketleridir. Nasıl davranılacağını bilmeyenler, engelli bireylere daha fazla zarar vermektedirler. Acıma ve pişmanlık duygusuyla yapılan yardımlar engelli bireylerin dış dünyadan daha fazla kopmalarına yol açabilmektedir. Bu amaçla halk, engelliler konusunda daha fazla bilinçlendirilmelidir. Engelli bireyler ve halk sıklıkla bir araya gelmedirler. Ancak, bu şekilde bir bütünleşme sağlanabilecektir.
- 2002 yılında yapılan engellilere ilişkin en kapsamlı yapılan araştırma “özürlülük araştırması” tekrar yapılmalı ve güncellenmelidir. Dünya ve ülke nüfusu arttıkça toplumdaki engelli oranları da değişmiştir. Dolayısıyla böyle bir araştırmanın tekrarlanması da zorunluluktur.
- Yeterli olanaklar ve fırsat eşitliği sağlanırsa topluma katkıda bulunan, başarılı ve sosyal bireyler konuşma engellilerden yetiştirilebilir.

- İmkani olmayan konuşma engelliler için meslek edindirme kurslarının sayısı artırılmalıdır.
- Konuşma engellilerin görev alabileceği kurum ve iş kolları tespit edilip netleştirilmeli ve kullanabileceği cihazlar söylenmelidir. Bu şekilde engelli insanı mutlu etmek için bir vasıta yapılmış olur. Birey mutlu olursa toplum mutlu olur.
- Unutmayınız !!! Biz de engelli olabiliriz. Her an başımıza gelebilir.
- Bu çalışmanın sonuçları daha sonra yapılacak çalışmalara ışık tutabilirse ve konuşma engelli bir bireyin hayatına olumlu yönde değişiklik yapabilirse çok mutlu olacağız.

5. Kaynakça:

Çalık, S. (2004). Özürllülüğün Ölçülmesinde Metodolojik Yaklaşımlar ve 2002 Türkiye Özürllüler Araştırması. Öz-veri Dergisi, 12, 153-375.

<https://eodev.com/gorev/207172>. Erişim Tarihi:04.01.2017

<http://kucukcekmecehastahabilyasyon.com/dil-ve-konusma-problemleri>. Erişim Tarihi: 13.11.2016

Konrot, A. (2005). Sözel Dil ve Konuşma Sorunları. Eskişehir.

Tonkaz, V. (2009). Görme Engelli İnsanlar İçin Sensör Kontrollü Seyahat Yardım Aracı Tasarımı Ve Simülasyonu. Isparta: Isparta Üniversitesi Fen Bilimleri Enstitüsü. Yüksek Lisans Tezi

Görsel Kaynaklar:

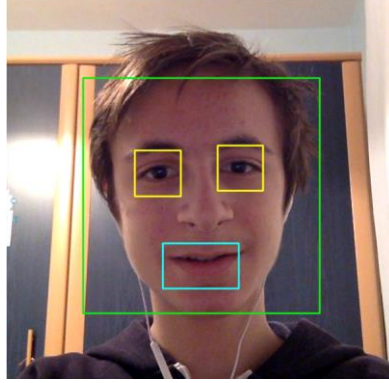
- <https://www.python.org/community/logos/>. Erişim Tarihi:29.12.2016
- http://docs.opencv.org/3.1.0/haar_features.jpg. Erişim Tarihi: 25.12.2016
- <http://docs.opencv.org/3.1.0/haar.png>. Erişim Tarihi: 25.12.2016

6. Ekler:

EK 1: Projede yararlanılan kaynaklar:

- Python eğitim belgeleri: <https://docs.python.org/3/>
- OpenCV eğitim belgeleri: <http://opencv.org/documentation.html>
- SQLite eğitim belgeleri: <https://www.sqlite.org/docs.html>

EK 2: Proje yapım aşamasındayken ki çalışma ortamımız ve proje ekran görüntüsü:



EK 3: Projede kullandığımız programlama dili: Python 3.5.2

EK 4: Projede kullandığımız kütüphaneler:

- OpenCV 3.2.0-dev
- sklearn
- os
- platform
- sqlite3
- multiprocessing
- Kendi yazmış olduğum: data_islem

EK 5: Projemizi hazırlarken kullandığımız bilgisayarımız ve özellikleri:

İşletim sistemi: MacOS Sierra (Version: 10.12.1) Retina
ekranlı 13 inç MacBook Pro 2015 model.
128 GB SSD, 2.7 GHz işlemci.
8 GB 1866 MHz LPDDR3 bellek.

EK 6: Projemizin kodlarını yazarken kullandığımız editörler:

- Atom(1.12.7)
- Vim

EK 7: Proje raporunu yazıldığı editörler:

- Pages(6.0)

- Microsoft Word

EK 8: Proje içindeki Database çalışmasını test ettiğim program:

- SQLiteBrowser(3.8.0)

EK 9: İlgilendiğim yazılım teknolojileri:

Python, C++, OpenCV, Sql, JavaScript, Qt, iOS(Swift), Java (SE), C#, Arduino, Unity Game Engine, C, Shell Scripting

EK 10: Üzerinde Çalıştığım Alanlar:

- Masaüstü uygulama yazılımları
- Mobil yazılımlar (iOS, iPhone, iPad, akıllı saatler, akıllı televizyonlar)
- Yapay zeka ve yapay sinir ağları
- Masaüstü ve mobil oyun geliştirme
- Web tasarımı (HTML,CSS)
- Apple ürünleri için yazılımlar
- 3D ve 2D oyun hazırlama
- Sistem ve sistem yazılımları
- Nesnelerin interneti
- Robotik
- Görüntü işleme
- Bilgisayar öğrenmesi
- Microchip (Pic, Arduino vb.)

EK 11: Deneyimlerim:

2016 3 aylık yaz stajı yaptığım şirket: Nart Bilişim Hizmetleri (TechNarts) Technarts
İnternet Sayfası: technarts.com

Yaptığım projelerden bazıları:

- C++ İle Yapay Zeka Satranç Oyunu Ve Oyunun Web’de Çalıştırmak
- Yapay Zeka Tic-Tac-Toe Oyunu
- Twitter Benzeri Sosyal Ağ
- 3D Labirent Oyunu
- ...

Diğer projelerim için internet sayfamı ziyaret edebilirsiniz. İnternet sayfam: ardamavi.com

“Sıfırdan, Az Enerji Harcayan Bilgisayar” projemin Sponsoru: Onur Mühendislik A.Ş. Onur Mühendislik İnternet Sayfası: onur.net

EK12: Proje Kodları:

Not: OpenCV ve eğitilmiş sınıflandırıcı dosyalarını çok uzun olmasından dolayı buraya ekleyemiyoruz.

- Ana Program Kodları:

```
# Arda Mavi import
cv2 import numpy
as np
import os import
sys import
platform
from multiprocessing import Process
import data_islem as di import
harfogren as ho

# Debugging için ayarlar: harfogren_var = bool(int(sys.argv[1])) # Datalar
# girilene kadar kapalı kalmalı.
# Programa gelen argüman string olarak alınır tam sayıya çevrilir ve bool yapılır.
# Programa gelen ilk argümanlar:
# Eğitim süreci ise = 0
# Yapay zeka ile çalışma anında = 1

tum_yuzleri_algilama = False
seslendirme_yap = True
gozluklu = True

# İşletim sistemine göre ekranı temizleyen fonksiyonumuz:
def clear_screen(): # clear_screen Screen:
if(platform.system() == 'Linux' or 'Darwin'):
    os.system('clear')
else:
    os.system('cls')
return def
seslendir(text):
```

```

if(platform.system() == 'Darwin'):
    arg = 'say {0}'.format(text)
    Process(target=os.system, args=(arg,)).start()
else:
    print('Seslendirme şimdilik sadece Darwin kabuğunda çalışmaktadır.')
return

# harfогren aktif ve database boş değil ise
True: def hoT_dbT(kullanici_adi):
    if kullanici_adi != "":
        kullanici_adi = 'WHERE kullanici="{0}"'.format(kullanici_adi)
        return di.veri_al('SELECT * FROM agiz_harf {0}'.format(kullanici_adi)) != [] and harfогren_var

def data_kayit(kullanici_adi, agiz_data, clf):
    clear_screen()
    print('Anlayamadım :(\nBu söylediğiniz hangi harf nedir?\nNot: Eğer bir harf belirtmek istemiyorsanız \'x\' giriniz.')
    # Girilecek harfi tutacağımız değişkeni oluşturuyoruz:
    harf = ""
    while 1:
        # Kullanıcıdan bir harf alınır:
        harf = input('Söylediğiniz harf: ')
    # Veri girildi mi kontrol edilir:
    if harf != "":
        # Veriler database'de küçük harf şekline tutuluyor.
        # Bunun için girilen harf her zaman küçük harf olmalı.
        # Bu yüzden girilen harfin küçük harf karşılığı tutuluyor.
        harf = harf[0].lower()
        break
    else:
        print('Hatalı Giriş!')
        # Veri Database'mize kaydoluyor:
        di.data_ekle('agiz_harf', 'kullanici', harf, agiz_genislik, agiz_yukseklk, (kullanici_adi, harf, agiz_data[0], agiz_data[1]))
        if harfогren_var:
            clf = ho.db_egitim(kullanici_adi)
            return clf

def harf_al(kullanici_adi, agiz_data, clf):
    # agiz_data = [Genişlik, Yükseklik]
    if hoT_dbT(kullanici_adi):
        gelen_veri = ho.getHarf(clf, agiz_data[0], agiz_data[1])
        clear_screen()
    else:
        if kullanici_adi != "":

```



```

        kullanıcı_adi = 'kullanici="{0}" AND'.format(kullanıcı_adi)
        # "data_islemler" dosyamızdan "veri_al" fonksiyonunu kullanarak kullanıcı adı ve ağız
        yapısına uygun harfi alıyoruz:
        gelen_veri = di.veri_al("SELECT harf FROM agiz_harf WHERE agiz_genislik={1}
        AND agiz_yukseklik={2}".format(kullanıcı_adi, agiz_data[0], agiz_data[1])) #
        Fonksiyonumuz "gelen_veri" değişkenini döndürür: return gelen_veri

def goruntu_isleme(kullanıcı_adi, clf):
    # OpenCV algoritmamızla çalıştıracağımız eğitilmiş dosyalarımızı belirliyoruz:
    y u z _ c a s c a d e = c v 2 . C a s
c a d e C l a s s i f i e r ( ' d a t a / h a a r c a s c a d e s /
haarcascade_frontalface_default.xml')

    # Debugging için:
    if gozluklu:
        goz_cascade = cv2.CascadeClassifier('data/haarcascades/
haarcascade_eye_tree_eyeglasses.xml') else:
        goz_cascade = cv2.CascadeClassifier('data/haarcascades/haarcascade_eye.xml')

    agiz_cascade = cv2.CascadeClassifier('data/haarcascades/mouth2.xml')

    # Görüntüyü alacağımız kameramızı
    tanımlıyoruz: cap = cv2.VideoCapture(0) while
    1:
        # İleride kullanmak için ağız değişkeni oluşturuyoruz:
        agiz = [-1,-1]
        # Tanımladığımız kameradan bir fotoğtaf çekiyoruz ve alıyoruz:
        ret, img = cap.read()

        # Yüz tanıma sistemimizin daha hızlı çalışması için fotoğrafı gri renk uzayına çeviriyoruz:
        # RGB(3 katmanlı pixel matrixi) sisteminden gri renk uzayına(tek katmanlı pixel
        matrixine) çeviriyoruz:
        # Nesne algılama sistemimizi, gri renk uzayındaki(tek katmanlı pixel matrixindeki)
        fotoğrafımızda yapacağız. grayImg = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        # Eğitilmiş dosyamızla OpenCV nesne bulma algoritmamızı çalıştırıyoruz:
        # Eğer yüzler bulunursa yüzlerin konumlarını fonksiyonumuz bize veriyor:
        yuzler = yuz_cascade.detectMultiScale(grayImg) kac_yuz_algilama = 1
        # Kaç yüz algılanmak isteniyor
        belirliyoruz: if tum_yuzleri_algilama:
        kac_yuz_algilama = len(yuzler)
        # Gelen yüzler içerisinde geziyoruz:

```



```

for (x,y,w,h) in yuzler[0:kac_yuz_algilama]:
    # Yüzün bulunduğu alanı opencv kütüphanemiz ile bir dörtgen içine
    aliyoruz: cv2.rectangle(img,(x,y),(x+w,y+h),(0,230,0),2) # Yüzün
    konumunu kaydediyoruz:
    # İleride göz ve ağızı bulmak için bu verileri
    kullanacağız:
    yuz = img[y:y+h, x:x+w]
    grayYuz = grayImg[y:y+h, x:x+w]

    # Eğitilmiş dosyamızla OpenCV nesne bulma algoritmamızı
    çalıştırıyoruz. # Ve bulunan göz koordinatlarını kaydediyoruz:
    gozler = goz_cascade.detectMultiScale(grayYuz)
    # Ağızı bulurken kullanmamız gereken bir değişken oluşturuyoruz:
    göz_lerin_alti_y = 0
    # Bulunan gözler içerisinde geziyoruz:
    for (ex,ey,ew,eh) in gozler[0:2]:
        # Gözün bulunduğu alanı opencv kütüphanemiz ile bir dörtgen içine alıyoruz:
        cv2.rectangle(yuz,(ex,ey),(ex+ew,ey+eh),(0,255,255),2)
        # İleride kullanmak için aşağıda olan gözün altının y koordinatını kaydediyoruz:
        if göz_lerin_alti_y < ey+eh: göz_lerin_alti_y = ey+eh

        # Göz bulamadıysa tekrar ara: if
        len(gozler) < 1 and göz_lerin_alti_y > 0:
        continue

        # Ağız sadece gözlerin altında olduğu için gözlerin üstünü aramamız performansı
        düşürebilir..
        # Bu yüzden ağızı sadece gözlerin altında aramalıyız:
        goz_alti = img[y+göz_lerin_alti_y+1:y+h, x:x+w]
        grayGoz_alti = grayImg[y+göz_lerin_alti_y+1:y+h, x:x+w]

        # Ağız bölgesi çizilir: cv2.rectangle(img,(x+1,y+göz_lerin_alti_y+1),(x+w-
        1,y+h-1),(255,255,255),1)

        # Eğitilmiş dosyamızla OpenCV nesne bulma algoritmamızı çalıştırıyoruz.
        # Ve bulunan ağız koordinatlarını kaydediyoruz:
        agizlar = agiz_cascade.detectMultiScale(grayGoz_alti)
        # Bulunan ağızlar içerisinde geziyoruz: for
        (mx,my,mw,mh) in agizlar[0:1]:
            # Ağızı dörtgen içine alıyoruz:
            cv2.rectangle(goz_alti,(mx,my),(mx+mw,my+mh),(255,255,0),2)
            # İleride kullanmak için ağız genişliğini ve yüksekliğini kaydediyoruz:
            # agiz = [Genişlik, Yükseklik]
            agiz = [int((mx+mw)-mx),int((my+mh)-my)]
            # Seslendirme yapılacak ise ve ağız bulunmuşsa:

```



```

        if seslendirme_yap and agiz != [-1,-1]:
while 1:
            cv2.destroyAllWindows()
harfler = harf_al(kullanici_adi, agiz, clf)
if harfler != []:
    # Fotoğrafın son halini gösteriyoruz:
cv2.imshow('Sesim Var!',img)
    # Eğer kişinin ağzı kapalıysa 'x' değeri döneceğinden seslendirme
yapılmaz: if harfler[0][0] != 'x': seslendir(harfler[0][0])
break else:
    cv2.putText(img,'Bilinmeyen Kelime !', (150,200),
cv2.FONT_HERSHEY_COMPLEX, 3, (0, 0, 255))
    cv2.imshow('Sesim Var!',img)
    clf = data_kayit(kullanici_adi, agiz, clf)

    else:
        cv2.imshow('Sesim Var!',img)

        if cv2.waitKey(1) == 27: # Decimal 27 = Esc
break
    # Kamerayı kapatıyoruz:
cap.release()

    # Fotoğrafi kullanıcıya gösterirken kullandığımız uygulama penceresini kapatıyoruz:
cv2.destroyAllWindows() return

def giris_yap(): print('Not: Üyeliğiniz yoksa yeni
üyelik oluşturulur!')
    print('Not: Kullanıcı odaklı olmasını istemiyorsanız boş bırakınız!')
kullanici_adi = input('Kullanıcı Adı: ') clear_screen() return
kullanici_adi

def main(): clear_screen() print('Sesim Var!\nArda Mavi -ardamavi.com\n\nÇıkış:
Esc\nÇıkmaya Zorla: kntrl+c\n') # Eğer DataBase veya tablo yok ise yeni process ile
yaratılır.
    Process(target=di.tablo_yarat, args=('agiz_harf',kullanici, harf, agiz_genislik,
agiz_yukseklk)).start()
    kullanici_adi = giris_yap() clf =
" if hoT_dbT(kullanici_adi):
clf = ho.db_egitim(kullanici_adi)

```

```

    goruntu_isleme(kullanici_adi, clf)
clear_screen()
    print('Sesim Var! - Program Sonlandırıldı !\nArda Mavi -ardamavi.com\n')
return

if __name__ == "__main__":
    main()

```

• siniraglari.py Kodları:

```

# Arda Mavi import os
import numpy as np
import data_islem as db
from sklearn.neural_network import MLPClassifier

def veriler(kullanici):
    # Sinir ağlarının eğitim sürecinde kullanılacak dataların çekilmesi:
    if kullanici != "":
        kullanici = 'WHERE kullanici="{0}"'.format(kullanici)
        db_veriler = db.veri_al('SELECT agiz_genislik, agiz_yukseklik FROM agiz_harf
{0}').format(kullanici))
        db_etiketler = db.veri_al('SELECT harf FROM agiz_harf {0}').format(kullanici))

        # Çıktı değerlerinin(etiketlerin) sinir ağlarının anlayacağı şekilde yapılandırması:
        # Databaseden gelen [(x,), (x,)] şeklinde veriler, ['x', 'x'] şekline
        çevrilir    db_etiketler = np.ravel(db_etiketler)    return db_veriler,
db_etiketler

def db_egitim(kullanici):
    X, y = veriler(kullanici)

    # Sinir Ağlarının oluşturulması:
    # MLPClassifier -> multi-layer perceptron (MLP)
    # hidden_layer_sizes verilere göre değişiklik gösterebilir.
    clf = MLPClassifier(solver='lbfgs', hidden_layer_sizes=(5,3))

    # TODO: Sinir ağları yapılandırılacak

    # Sinir ağlarının eğitimi:
    clf = clf.fit(X, y)    return
    clf

```

```
def getHarf(clf,w,h):
    # Sinir ağlarını kullanarak ağız yapısından harf tahmini:
    harf = clf.predict([[w,h]])
    print('Sinir Ağları Harf Tahmini: ', harf)
    return harf
```

• harfogren.py Kodları:

```
# Arda Mavi import os
import data_islem as db
from sklearn import
tree

def db_egitim(kullanici):
    if kullanici != "":
        kullanici = 'WHERE kullanici="{0}"'.format(kullanici)
        db_veriler = db.veri_al('SELECT agiz_genislik, agiz_yukseklik FROM agiz_harf
{0}'.format(kullanici))
        db_etiketler = db.veri_al('SELECT harf FROM agiz_harf {0}'.format(kullanici))

        # Alınan verime bağlı olarak sınıflandırma yöntemi değiştirilebilir.

        # Farklı sınıflandırma yöntemleri ve verilere göre alınan performansları grafikler şeklinde
        # projemiz içerisinde 'Ekler' klasöründe 'classification.png' içerisinde gösterilmiştir.

        # Sınıflandırma yöntemleri arasından, elimdeki verileri(eğitim verilerini) kullanarak
        deneğim,
        # sınıflandırma işlemlerinden en iyi performansı aldığım yöntemi seçtim.
        # Fakat yöntem seçimi sonuçlardan alınan verime göre değiştirilebilir.

        # Projemizde Scikit(sklearn) kütüphanemizle kullandığımız sınıflandırıcı:
        DecisionTreeClassifier
        # DecisionTreeClassifier = Karar Ağacı Sınıflandırıcısı

        clf = tree.DecisionTreeClassifier()
        clf = clf.fit(db_veriler, db_etiketler)
        return clf

def getHarf(clf,w,h):
    harf = clf.predict([[w,h]])
    print('Karar Ağacı Harf Tahmini: ', harf)
    return harf
```

• data_islem.py Kodları:


```

# Arda Mavi - ardamavi.com
import sqlite3

# Database bağlantısı kurmak: def
set_sql_connect(database_name):
return sqlite3.connect(database_name)

# Database Cursor ayarlamak: def
set_sql_cursor(database_connect):
return database_connect.cursor()

# Database bağlantısı kurup Cursor ayarlamak:
def set_connect_and_cursor():
    vt =
set_sql_connect('data/database/database.sqlite')    db
= set_sql_cursor(vt)    return vt, db

# Database bağlantısını kapatmak
için: def close_connect(vt):    if vt:
    vt.commit()
vt.close

# Database'de yaratılmamışsa yeni tablo yaratmak için:
def tablo_yarat(table_name, columns):    vt, db =
set_connect_and_cursor()
    db.execute("CREATE TABLE IF NOT EXISTS {0} ({1})".format(table_name, columns))
close_connect(vt)

# Database'den veri almak için: def
veri_al(sql_komut):    vt, db =
set_connect_and_cursor()
db.execute(sql_komut)    gelen_veri
= db.fetchall()    close_connect(vt)
return gelen_veri

# Database'e veri eklemek için: def
data_ekle(table, eklenecek_sutun, eklenecek):
vt, db = set_connect_and_cursor()
    db.execute("INSERT INTO '{0}'({1}) VALUES {2}".format(table, eklenecek_sutun,
eklenecek))

```

```
# eklenecek_sutun örnek: 'sütun1','sütun2'
# eklenecek örnek: data1, data2
close_connect(vt)

# Database'deki verileri güncellemek için:
def data_guncelle(table, nerden, nasil):
    vt, db = set_connect_and_cursor()
    db.execute("UPDATE {0} SET {2} WHERE {1}".format(table, nerden, nasil))
    close_connect(vt)
```