TRAKYA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

LİSANS BİTİRME PROJESİ DÖNEM RAPORU
PROJE II

PROJE ADI:
DİLSİZ ÇEVİRMENİ

HAZIRLAYANLAR:
1201602020 Ozan Can SARI
1201602039 Buse UYSAL
1201602649 Melisa VASİJA
1201602617 Anal KARAALI

DANIŞMAN:
Dr. Öğr. Üyesi Rembiye KANDEMİR

Edirne, Haziran 2024

İÇİNDEKİLER

KISALTMA LİSTESİ

TİD : Türk İşaret Dili

IDA : İşaret Dili Algılama

M. Ö. : Milattan Önce

ABD : Amerika Birleşik Devletleri

WPF : Windows Presentation Foundation

NYP : Nesne Yönelimli Programlama

DDİ : Doğal Dil İşleme

AI : Artificial İntelligence

ML : Machine Learning

IDE : Integrated Development Environment

YZ : Yapay Zeka

Viz : Visualization

DNN : Deep Neural Networks

GUI : Grafiksel Kullanıcı Arayüzü

VR : Virtual reality

VS Code : Visual Studio Code

OpenCV : Open Source Computer Vision Library

FLANN : Fast Library for Approximate Nearest Neighbors

Calib3D : Camera Calibration and 3D Reconstruction

Videostab : Video Stabilization

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

ÖNSÖZ

Bu proje işitme engelli bireyler ile daha kolay bir şekilde iletişim kurabilmek, engelli bireyin işaret dili bilmeyen birisine ihtiyaçlarını rahatlıkla anlatabilmesi ve işaret dili öğrenilmesinde kolaylık sağlanması adına hazırlanmıştır.

Bu projenin planlanmasında, araştırılmasında ve yürütülmesinde bizlere ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübesiyle bizlere yol gösteren sayın hocamız Dr. Öğr. Üyesi Rembiye KANDEMİR'e sonsuz teşekkürlerimizi sunarız. Kendisine sadece projemize sağladığı yönlendirme ve destek için değil, aynı zamanda bize öğrenciler olarak ilham verdiği için içtenlikle teşekkür ederiz.

Projede birlikte emek harcayan dört arkadaş olarak, her bir arkadaşımıza içten saygılarımızı iletiyoruz. Gelecekte, bilgisayar mühendisi olarak edindiğimiz bilgi ve becerileri her zaman toplum için faydalı projelerde kullanacağımıza söz veriyoruz.

ÖZET

Günlük yaşantımızın karmaşıklığında farkına varılmayan birçok zorlukla karşılaşıyoruz. Bu zorluklardan biri, işitme engelli bireylerin iletişim zorluğuyla baş etmeleridir. Projemiz, bu iletişim zorluklarını en aza indirmeyi amaçlayarak geliştirilmiştir.

Proje, esasen işitme engelli bireylerin işaret dili bilmeyen kişilerle daha kolay iletişim kurmalarını hedeflemektedir. Kullanıcılar, kamera karşısına geçerek işaret diliyle iletişim kuracak ve bu işaretleri Türkçe'ye çevirerek diğer kişilerin algılamasını sağlayacaktır. Bu sayede, işitme engelli bireyler, işaret dili bilmeyen kişilerle daha etkili bir şekilde iletişim kurma imkânına sahip olacaklardır.

Projemiz, teknolojinin gücünü kullanarak engelli bireylerin toplumsal katılımını artırmayı ve iletişimde karşılaştıkları zorlukları en aza indirmeyi amaçlamaktadır. Bu süreçte, kullanıcıların günlük yaşamlarında daha etkin ve bağımsız olmalarını sağlamak adına projemizin önemli bir adım olduğuna inanıyoruz.

ABSTRACK

In the complexity of our daily lives, we encounter many challenges that often go unnoticed. One of these challenges is dealing with communication difficulties faced by individuals with hearing impairments. Our project has been developed with the aim of minimizing these communication challenges.

The project essentially aims to facilitate communication for individuals with hearing impairments with those who do not know sign language. Users will stand in front of a camera, communicate using sign language, and the system will translate these signs into Turkish to allow others to perceive the message. This way, individuals with hearing impairments will have a more effective means of communicating with those who do not know sign language.

By harnessing the power of technology, our project seeks to increase the social participation of disabled individuals and minimize the communication challenges they face. Throughout this process, we believe that our project is a significant step in enabling users to be more effective and independent in their daily lives.

BÖLÜM 1

GİRİŞ

İnsan kulağı belirli sesleri duyma yeteneğine sahiptir. Fakat bazı biyolojik nedenlerden dolayı duyma yetisi azalabilir veya yok olabilir. Bu duruma *"işitme kaybı"* denir. Bu durumu yaşayan işitme engelli bireyler işaret dili sayesinde iletişim kurarlar.

Türk İşaret Dili (TİD) dünyadaki en eski işaret dillerinden biri olarak kabul edilir (Zeshan, 2003). TİD Türkiye'de ve Kuzey Kıbrıs Türk Cumhuriyeti'nde kullanılmaktadır. Her işaret dili gibi TİD de Türkçe'nin gramer yapısından farklı olarak kendine has bir gramer yapısı mevcuttur. Türkçe'de eş seslilik mevcutken, TİD'de eş işaretlilik mevcuttur.

1.1 Projenin Amacı ve Hedefi

İşitme engelli bireyler ile daha kolay iletişim kurulabilmesi için işaret dili ve görüntü işleme tekniğinden yararlanılarak yeni bir sistem tasarlanması amaçlanmıştır. Bu proje sayesinde engelli bireylerin ihtiyaç ve arzularını, yaşadıkları iletişim problemini en aza indirerek bildirmeleri hedeflenmektedir.

Unutulmamalıdır ki her insan bir engelli adayıdır. Bu sebeple TİD'in öğrenilmesinde kolaylık sağlanması ve TİD bilen kişi sayısının artması da hedeflenmektedir.

Şekil 1.1 İşaret Dili ile ilgili görsel

1.2 Türk İşaret Dili Hakkında

Türk İşaret Dili, kısaca TİD, Türkiye ve Kuzey Kıbrıs Türk Cumhuriyeti'nde kullanılan, esas olarak parmak, el, kol, ağız hareketlerine ve mimiklere dayanan görsel bir dildir. Türk İşaret Dili sadece işitme engellilerin kendi aralarında kullandıkları bir dil değil, aynı zamanda işitme ve konuşma engeli olmayan bireylerle de iletişim kurmalarını sağlayan bir araçtır.

Türk İşaret Dilinin kendine özgü bir dil bilgisi ve söz dizimi vardır. Türk dilinden bu yönüyle farklılık göstermektedir.Bugün bildiğimiz kadarıyla işaret dilinin ilk izlerine Yusuf Has Hacib'in yazdığı Katadgu Bilig ve Kaşgarlı Mahmud'un yazdığı Divanü Lugati't Türk eserlerinde rastlanmıştır. Ancak bu eserlerde bahsi geçen işaretleşmenin bugünkü Türk İşaret Dilinin temellerini oluşturduğu düşünülmemektedir.

Bugün konuşulan TİD'nin, niteliği ve özellikleri açısından Osmanlı İmparatorluğu'nda kullanılan işaret diline dayandığı varsayılmaktadır. Osmanlı İşaret Dili de saray ve saray dışı olarak ikiye ayrılmaktaydı. İşitme engelliler sarayda resmi görevlerde aktif olarak yer almakta ve padişaha yakın hizmet etmekteydiler.

Evliya Çelebi'nin Seyahatname'sinde Osmanlı'da kullanılan bu işaret dilinden bahsedilmektedir. Saray dışındaki işaret diliyle ilgili ise pek fazla bilgi bulunmamaktadır. Ancak işaret diline yönelik ilk girişimin, 19.yüzyılın sonunda Ferdinand Grati tarafından kurulan Dilsiz Mektebi olduğu bilinmektedir. Bu mektep sonrasında II. Abdülhamit tarafından 1902 yılında kurulan Yıldız Sağırlar Okulu'na öncülük etmiştir.

Günümüz Türk İşaret Dilinin temellerinin de burada atıldığı tahmin edilmektedir. Bundan sonraki dönemde, Türkiye'de işaret dilinin gelişimine yönelik yeterli adımlar atılamamış, işitme engellilerin konuşmaya yönelik sözlü eğitime teşvik edilmeleri amaçlanmış ve bu kapsamda 1953 yılında, sağır dilsiz okullarında işaret dili yasaklanmıştır.

Ne yazık ki bu uygulama 2000'li yıllara kadar etkili olmuş, dünyadaki pek çok ülkenin ulusal işaret dilleri gelişirken, Türkiye'de çok geri kalmıştır. Günümüzde Türk Dil Kurumu'nun ve Milli Eğitim Bakanlığı'nın TİD (şekil 1.2) üzerine çeşitli araştırma ve çalışmaları, eğitim kitleri ve basılı birer sözlükleri bulunmaktadır.

Şekil 1.2 TİD

1.3 Gereksinim Analizi

Projenin ihtiyaç duyduğu ana modüller analiz edilmiş (software requirements specification), proje amaçları ve hedefleri detaylandırılmıştır.

1.3.1 Zaman Fizibilitesi

Projenin zaman fizibilitesi hazırlanmıştır. Planlanan gantt şeması Çizelge 1.1'de gösterilmektedir.

Çizelge 1.1 Projenin Güz Dönemi Zaman Fizibilitesi

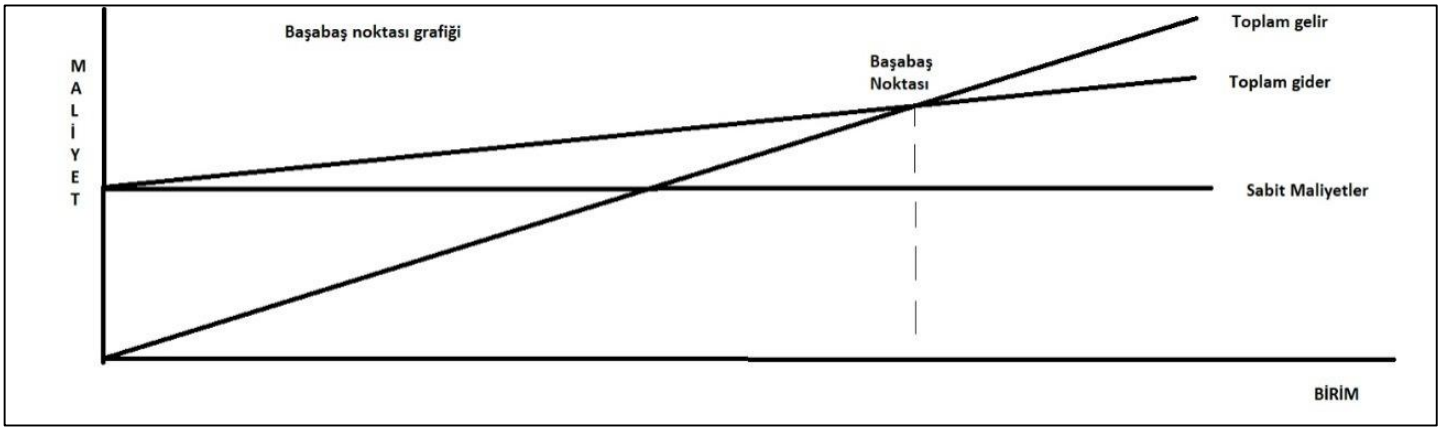
1.3.2 Ekonomik Fizibilite

Bir proje için ekonomik fizibilite analizi, projenin finansal olarak uygulanabilir olup olmadığını değerlendiren kritik bir süreçtir. Bu analiz, projenin başlangıç maliyetlerini, işletme giderlerini ve olası gelir kaynaklarını detaylı bir şekilde incelemeyi içerir. Ayrıca, projenin potansiyel getirileri ve bu getirilerin maliyetlere oranı üzerine yapılan analizlerle projenin finansal sürdürülebilirliği değerlendirilir.

Ekonomik fizibilite analizi, projenin yatırım getirisi, geri dönüş süresi ve net bugünkü değeri gibi finansal göstergeler üzerinde odaklanarak projenin uzun vadeli karlılığını öngörme konusunda önemli bilgiler sağlar. Bu şekilde, projenin başarıyla tamamlanması ve hedeflenen sonuçların elde edilmesi için gerekli finansal stratejilerin belirlenmesine katkıda bulunur.

Projemizin gerçekleştirilmesi için gerekli olan donanım, yazılım lisansları ve diğer maliyetler detaylı bir şekilde hesaplanmıştır. Donanım maliyetleri arasında kullanılacak bilgisayarlar, sensörler ve iletişim ekipmanları bulunmaktadır. Yazılım lisansları ise projenin geliştirilmesi ve uygulanması için gereken programlama dilleri ve ilgili yazılım araçları için tahsis edilmiştir.

Proje için gerekli maliyetler hesaplanmıştır. Bu doğrultuda ekonomik fizibilite hazırlanmıştır. Tüm bu maliyetler, projenin bütçe yönetimi stratejileri doğrultusunda etkin bir şekilde yönetilerek projenin başarılı bir şekilde tamamlanmasına katkı sağlamaktadır. Planlanan gantt şeması Çizelge 1.2'de gösterilmektedir.



Çizelge 1.2 Projenin Ekonomik Fizibilitesi

1.3.3 Teknik Fizibilite

Proje Tanımı:

Proje Adı: Dilsiz Çevirmen
Proje Amacı: İşitme engelli bireylerin ve işitme düzenekleri kullanıcılarının konuşmalarını anlamalarına yardımcı olmak için bir yazılım geliştirmek.

Teknik Gereksinimler:
İşaret Dili Tanıma: İşaret dili anlama ve çözümleme yetenekleri.
Çeviri Motoru: Çeşitli diller arasında anlam çıkarabilen bir çeviri motoru.
Kullanıcı Arayüzü: Kullanıcıların kolayca etkileşimde bulunabileceği bir arayüz.

Yazılım Geliştirme Süreci:
Agile veya Scrum gibi esnek yazılım geliştirme metodolojileri kullanılabilir.
İteratif geliştirme ile kullanıcı geri bildirimleri dikkate alınabilir.

Teknolojik Altyapı:
Bulut Bilişim: Hesaplama gücü ve depolama sağlamak için bulut tabanlı altyapılar kullanılabilir.
Yüksek Performanslı Sunucular: Hızlı ve etkili işlemler için güçlü sunucular.

Veri Güvenliği ve Gizlilik:
Kullanıcı verilerinin güvenliği için endüstri standardı şifreleme yöntemleri kullanılmalıdır.
GDPR ve diğer gizlilik düzenlemelerine uygunluk sağlanmalıdır.

Uyum ve Entegrasyon:
Mobil uygulama ve web tarayıcıları üzerinden erişilebilirlik sağlamak için platform bağımsız bir tasarım.
İşitme cihazları ve diğer teknolojik araçlarla uyumluluk.

Test ve Doğrulama:
Kapsamlı test süreçleriyle yazılımın doğruluğu ve güvenilirliği sağlanmalıdır.
Beta testleri ve kullanıcı geri bildirimleri ile sürekli iyileştirmeler.

Güncelleme ve Bakım:
Yazılımın güncel ve güvenli kalması için periyodik bakım ve güncelleme yapılmalıdır.

1.3.4 Gantt Şeması

Gantt şeması, proje yönetiminde yaygın olarak kullanılan bir araç olup, projedeki görevlerin zaman çizelgesini ve ilerlemesini görselleştirmek için kullanılmaktadır. Bu araç, projenin genel yapısını ve ilerlemesini proje yöneticileri, ekip üyeleri ve paydaşlar gibi ilgili taraflara net bir şekilde gösterme imkanı sağlamaktadır.

Gantt şemasının ilk olarak kullanımı, 1910 yılında Amerikalı mühendis ve yönetim danışmanı olan Henry Gantt tarafından önerilmiştir. Gantt, şemasını endüstriyel üretim süreçlerini yönetmek için geliştirmiş ve başlangıçta işçilerin iş programlarını görselleştirmek için kullanmıştır.

Gantt şeması, her görevin başlangıç ve bitiş tarihlerini belirleyerek ve görevler arasındaki bağımlılıkları netleştirerek projenin daha iyi planlanmasına yardımcı olur. Bu sayede, proje sürecinin yönetimi kolaylaşır ve kaynakların daha etkin bir şekilde kullanılması sağlanır.

Projelerin zamanında tamamlanması için Gantt şeması büyük önem taşır. Şema, projenin ilerlemesini takip etmek, gecikmeleri belirlemek ve iletişimi kolaylaştırmak için önemli bir araçtır. Ayrıca, ekip üyelerine belirli görevleri ve zaman çizelgesini açıkça gösterir, böylece herkesin projenin hedeflerine katkı sağlaması kolaylaşır.

İlerleme Gantt şemalarında, görevlerin tamamlanma derecesine göre gölgelendirilirler: %60 tamamlanan bir görev, soldan başlayarak %60 gölgelendirilir. İlerleme Gantt şeması oluşturulduğunda bir zaman işareti boyunca dikey bir çizgi çizilir ve bu çizgi daha sonra gölgeli görevlerle karşılaştırılabilir.

Eğer her şey planlandığı gibi ilerliyorsa, çizginin solunda kalan tüm görev kısımları gölgelendirilirken, çizginin sağındaki tüm görev kısımları gölgelenmez. Bu, projenin ve görevlerinin zamanında ilerleyip ilerlemediğini görsel olarak gösterir. Bağlantılı Gantt şemaları, görevler arasındaki bağımlılıkları gösteren çizgiler içerir.

Ancak, basit olmayan durumlarda, bağlantılı Gantt şemaları hızlı bir şekilde karışık hale gelir. Kritik yol ağ diyagramları, görevler arasındaki ilişkileri görsel olarak iletmek için üstündür. Bununla birlikte, eğitim almadan Gantt şemalarının kolayca yorumlanabiliyor olması nedeniyle, ağ diyagramlarına göre Gantt şemaları tercih edilir.

Gantt şema yazılımları genellikle görev bağımlılıklarını bağlamak için mekanizmalar sağlar, ancak bu veri görsel olarak temsil edilebilir veya olmayabilir. Gantt şemaları ve ağ diyagramları genellikle aynı projede kullanılır, her ikisi de aynı veriye dayanarak bir yazılım uygulaması tarafından oluşturulur

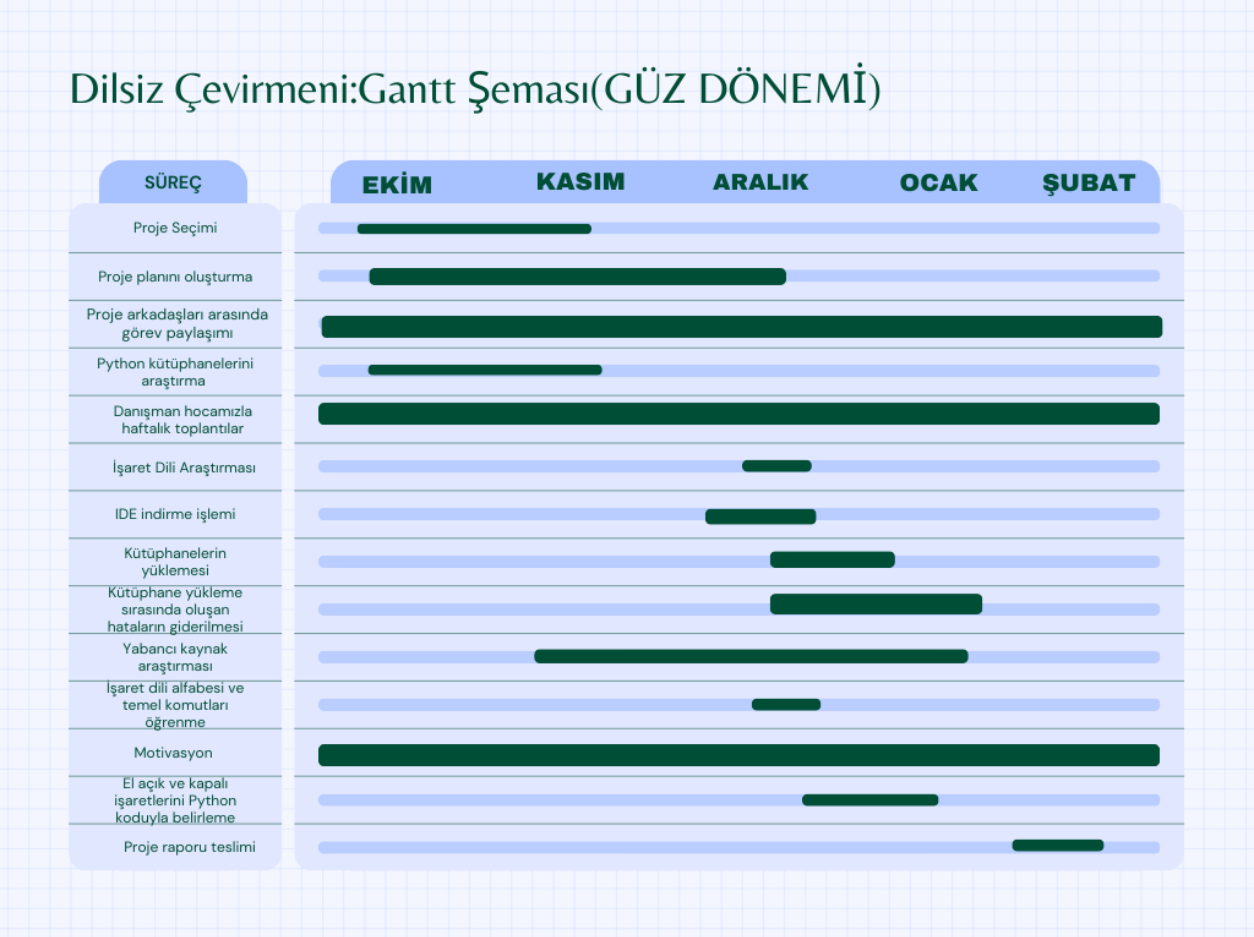
Biz roje kapsamında, güz dönemi ve bahar dönemi olmak üzere iki ayrı Gantt şeması oluşturduk. Her iki şemada da aylar belirli bir şekilde organize edilmiştir; güz dönemi Ekim, Kasım, Aralık, Ocak, Şubat, bahar dönemi ise Mart, Nisan, Mayıs ve Haziran aylarını kapsamaktadır.

Bu şekilde, projenin iki farklı dönemi için ayrı ayrı zaman çizelgeleri belirlenmiş ve her bir dönemin görevleri net bir şekilde planlanmıştır. Bu yaklaşım, projenin zaman yönetimi ve ilerlemesini daha etkin bir şekilde takip etmemize yardımcı olmuştur.

Projenin güz dönemi boyunca yürütülen süreçlerin zamanlaması Gantt şemasıyla belirlenmiştir. İlk olarak, proje seçimi aşamasında, uygun bir projenin belirlenmesi için zaman ayrılmıştır. Ardından, proje planının oluşturulması için gerekli adımlar atılmış ve planlama süreci tamamlanmıştır. Daha sonra, projenin gerçekleştirilmesi için gereken görevler proje arkadaşları arasında paylaşılmış ve görev dağılımı yapılmıştır.

Python kütüphaneleri araştırma aşaması, projenin gereksinimlerini karşılayacak uygun araçların belirlenmesi için gerçekleştirilmiştir. Danışman hocamızla yapılan düzenli toplantılar, proje ilerlemesini değerlendirmek ve yönlendirmeler almak için düzenlenmiştir. Ayrıca, işaret dil araştırması ve IDE indirilmesi gibi önemli görevlerin tamamlanması için belirlenen süreçler takip edilmiştir. Bu süreçler, Ekim, Kasım, Aralık, Ocak ve Şubat aylarını kapsamaktadır.

Her bir görevin başlangıç ve bitiş tarihleri Gantt şemasıyla belirlenmiş ve projenin güz dönemi boyunca başarıyla tamamlanmıştır. Projemiz güz dönemi için geliştirdiğimiz Gantt şeması, yürütülen süreçlerin zamanlamasını net bir şekilde göstermektedir ve aşağıda yer almaktadır. Geliştirilen süreçlerin zamanlamasını daha ayrıntılı olarak görmek için, Gantt şemasına (Çizelge 1.3) inceleyebilirsiniz.

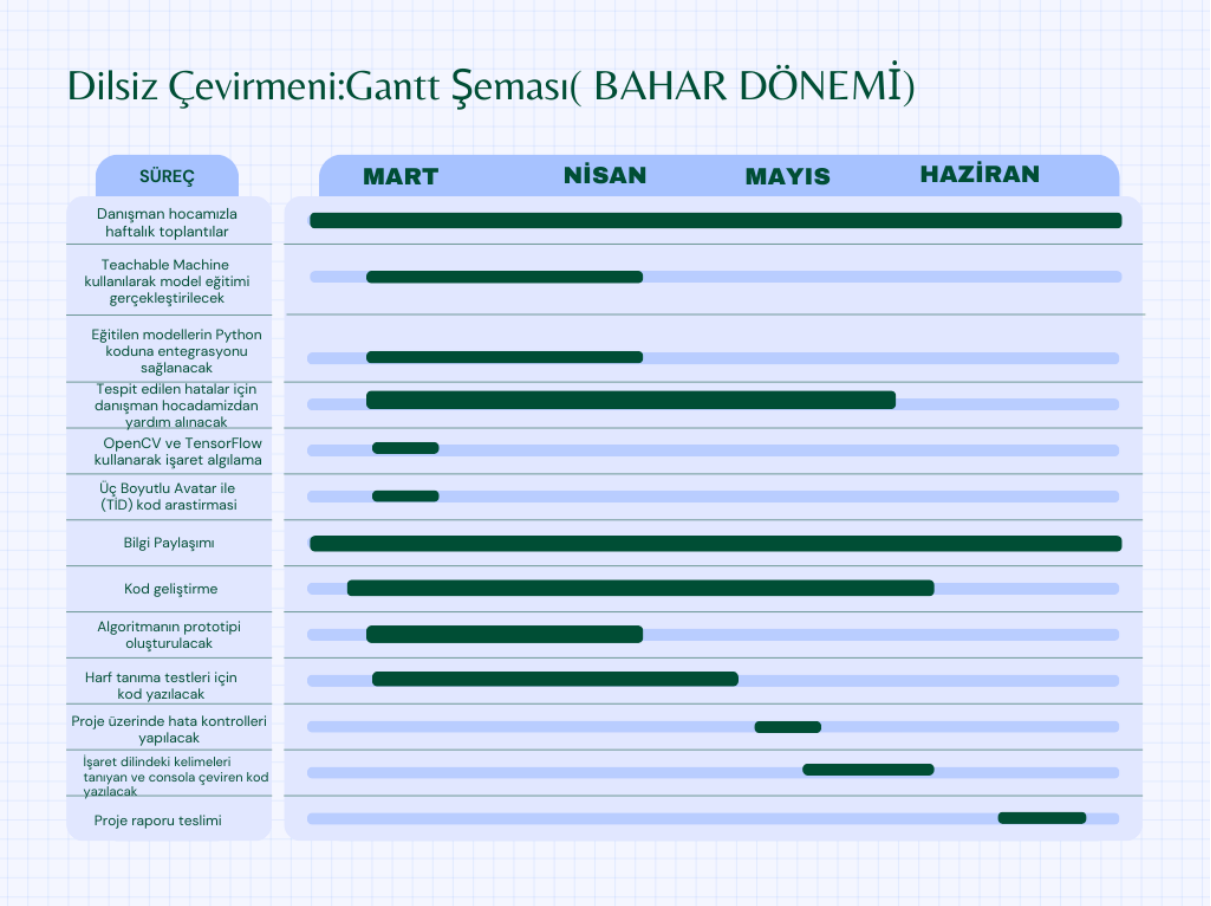


Çizelge 1.3 Güz Dönemi Gantt Şeması

Biz projemizin yönetiminde, hem güz hem bahar dönemlerini kapsayacak şekilde iki ayrı Gantt şeması oluşturduk. Bahar döneminde, projenin ilk aşamalarını planladık ve temel görevleri belirledik.

Bu dönemde, projenin analiz ve tasarım aşamalarını tamamlamayı hedefledik ve bunun için gereken görevleri belirledik. Bahar döneminde projemizin ilerleyişini sağlamak adına çeşitli adımlar attık. Danışman hocamızla düzenli haftalık toplantılar düzenleyerek proje ilerlemesini değerlendirdik ve yönlendirmeler aldık.

Ayrıca, teachable model kullanarak model eğitimi gerçekleştirdik ve eğitilen modelin Python koduna entegrasyonunu sağladık. Bu entegrasyon sürecinde, kod geliştirme çalışmaları da yaptık, modelin performansını artırmak için çeşitli iyileştirmeler ve güncellemeler gerçekleştirdik. Daha fazla bilgi için, proje süreçlerinin zamanlamasını gösteren Bahar Dönemi Gantt Şemasına (Çizelge 1.4) bakınız.



Çizelge 1.4 Bahar Dönemi Gantt Şeması

1.4 İşaret Dilinin Tarihsel Gelişimi

İşaret diline dair en eski yazılı kayıt M.Ö. 5. yüzyıla dayanmaktadır. 19. yüzyıla kadar işaret dili parmak alfabesi ile sınırlıydı. İlk parmak alfabesini geliştiren kişinin Pedro Ponce'de León (1520-1584) olduğu söylenir.

Türk İşaret Dili'nin (TİD) kökeni kesin olarak bilinmese de varsayımsal olarak kökü Osmanlı saraylarında bulunan sağırlara dayandırılmaktadır. Eğitim konusunda ise ilk olarak II. Abdülhamit döneminde yapılmıştır. Bu dönemde sağırlar için çeşitli yerlerde sağır okulları açılmıştır.

Türkiye Cumhuriyeti'nin ilk yıllarında açılan okullar oralist metotla çalışmaktaydı, fakat 2000'li yıllarda bu metodun terk edilmesiyle, Türkiye'de bulunan okullar işaret dilinde eğitime geçti.

İşaret dillerinin genellikle bulunduğu bölgelerdeki konuşma dilleriyle arasında hiçbir dilbilimsel bağ bulunmaz. İşaret dili ve konuşma dili arasındaki ilişki karmaşık ve konuşma dilinden çok bulunduğu ülkeye göre değişim gösteren bir şeydir.

Zaman zaman, sağır insanların yaygınlığının yeterli olduğu yerlerde, yerel topluluğun tamamı tarafından bir işaret dili kullanılmaya başlanır ve bu *"köy işaret dili"* ya da *"ortak işaret topluluğu"* oluşmasına yol açar. En çok bilinen örnekler: Martha's Vineyard İşaret Dili (ABD), Mardin İşaret Dili (Türkiye), Orta Toroslar İşaret Dili (Türkiye).

İşaret dili çevirisi bazen konuşma içeren televizyon programları için sağlanmaktadır. Paddy Ladd 1980'lerde Britanya televizyonunda sağırlar için bir program başlattı ve kendisi işaret dilini televizyona taşıyan ve sağır çocukların işaret dilinde eğitim görmesini sağlayan kişi olarak bilinir. [1]

BÖLÜM 2

SİSTEM BİLEŞENLERİ

2.1 Projede Kullanılacak Teknolojiler

İşaret dili çevirmeni projesinde kullanılacak teknolojiler aşağıdaki gibi sıralanabilir:

Donanım ve İşletim Sistemi

Proje, Windows 10 işletim sistemi üzerinde çalışan bir dizüstü bilgisayarda geliştirilmiştir. Projenin üzerinde çalışacağı donanım sistemi, bilgisayarın işlemci gücü, RAM miktarı ve ekran çözünürlüğü gibi faktörler göz önünde bulundurularak seçilmiştir.

Programlama Dili

Projenin ana geliştirme dili olarak Python kullanılmıştır. Python, güçlü ve esnek bir dil olması nedeniyle işaret dili çevirmeni gibi karmaşık bir proje için uygun bir seçimdir. Python, aşağıdaki özelliklere sahiptir:

Projede, OpenCV görüntü işleme kütüphanesi kullanılmıştır. OpenCV, görüntü işleme için kullanılan açık kaynaklı bir kütüphanedir.

Projede, TensorFlow makine öğrenimi kütüphanesi kullanılmıştır. TensorFlow, makine öğrenimi için kullanılan açık kaynaklı bir kütüphanedir.

Projede, MySQL veritabanı kullanılmıştır. MySQL, açık kaynaklı bir ilişkisel veritabanı yönetim sistemidir.

2.2 Metotlar ve Yöntemler

Proje tasarlanırken birçok metot ve yöntemden yararlanılmıştır.

2.2.1 Donanım Sistemi ve İşletim Sistemi

Proje, genel olarak kişisel bilgisayarları hedeflerken, farklı işletim sistemlerini de destekleme potansiyeline sahiptir. Öncelikli olarak Windows işletim sistemi düşünülmüş olsa da, projenin ölçeği ve kullanıcı kitlesi göz önüne alındığında, diğer popüler işletim sistemleri olan macOS ve Linux'a da destek verilebilir.

2.2.2 Kullanılan Programlama Dili (Python)

Python programlama dilini tercih etmemizin temel sebepleri şunlardır: geniş kütüphane ve modül desteği, esnek sözdizimi ve dinamik tip sistemi. Bu özellikler, işitme engelli bireylerle etkili iletişim kurabilmek adına projemizin temel ihtiyaçlarına uygun çözümler sunar. Python'un doğal dil işleme (NLP) yetenekleri, kullanıcı arayüzü geliştirme ve veri analizi gibi alanlarda geniş bir destek sağlar. Özellikle, işitme engelli bireylerin dilinden işaret diline geçişte doğal dil işleme algoritmaları, projemizin temelini oluşturan unsurlardan biridir.

Python'un açık kaynak olması, geniş bir geliştirici topluluğuna sahip olması ve sürekli güncellenmesi, projemizin sürdürülebilirliğini ve gelişimini destekler. Ayrıca, Python'un platform bağımsız olması, projenizin farklı işletim sistemlerinde sorunsuz bir şekilde çalışabilmesine olanak tanır ve kullanıcılar için geniş bir erişim sağlar.Python'un hızlı prototipleme imkanı, projenizin geliştirilme sürecini hızlandırır ve işlevselliğin daha erken aşamalarda test edilmesine imkan tanır. Bu da, projenin başarıya ulaşma sürecini hızlandırır ve kullanıcı geri bildirimlerini daha etkili şekilde değerlendirme fırsatı verir.

Yapay zeka ve makine öğrenimi alanında Python'un öncü bir dil olması, projenizin ileri seviye yapay zeka entegrasyonlarını desteklemesi açısından önemli bir avantaj sağlar. TensorFlow, PyTorch gibi kapsamlı kütüphaneler sayesinde, projenizin yapay zeka bileşenlerini entegre etmek ve işitme engelli bireyler için daha zengin, etkileşimli deneyimler sunmak mümkün hale gelir.Sonuç olarak, Python'un geniş kütüphane desteği, esnekliği, topluluk desteği ve yapay zeka yetenekleri, işitme engelli bireyler için geliştirilen projelerde ideal bir seçenek olmasını sağlar. Bu özellikler, projenizin etkili iletişim çözümleri sunmasına ve kullanıcı ihtiyaçlarına uygun çözümler üretmesine olanak tanır. Şekil 2.1'de Python logosu bulunmaktadır. [2]

Şekil 2.1 Python Programlama Dilinin Logosu

2.2.3 Tümleşik Geliştirme Ortamı (Visual Studio Code)

Visual Studio Code (**VS Code**), projemizin geliştirme sürecinde temel tümleşik geliştirme ortamı olarak seçilmiştir. VS Code, hafif yapısı ve genişletilebilir özellikleriyle öne çıkar. Python programlama dili için özel olarak tasarlanmış eklentileri ve kütüphaneleri destekleyerek, geliştirme sürecinde verimliliği artırır. Özellikle, hızlı kod düzenleme ve hata ayıklama imkanlarıyla, projenin karmaşıklığına uygun çözümler sunar. Ayrıca, entegre terminal desteği sayesinde geliştirme ve test süreçlerini tek bir arayüzde yönetebilme avantajı sağlar.

Projemizde Visual Studio Code'un sunduğu çoklu dil desteği ve kod otomatik tamamlama özellikleri büyük önem taşımaktadır. Python dilinin yanı sıra HTML, CSS ve JavaScript gibi web geliştirme dillerini destekleyerek, işitme engelli bireyler için kullanıcı dostu web arayüzlerinin oluşturulmasını sağlar. VS Code'un sağladığı zengin eklenti ekosistemi, veri analizi, görselleştirme ve yapay zeka uygulamaları gibi farklı alanlarda projenin ihtiyaçlarına uygun çözümler sunabilmesini sağlar. Bu özellikler, projenin geliştirme sürecini çeşitli boyutlarda optimize eder ve işitme engelli bireylerin dijital içeriklere erişimini kolaylaştırır.

Son olarak, Visual Studio Code'un açık kaynaklı olması ve büyük bir geliştirici topluluğuna sahip olması, projenin sürekli iyileştirilmesi ve güncellenmesi için önemli bir avantaj sağlar. Topluluk desteği sayesinde, geliştiricilerin projenin her aşamasında karşılaştıkları zorlukları aşmalarına ve yenilikçi çözümler üretmelerine olanak tanır. Bu özellikler, projenin uzun vadeli başarısı ve sürdürülebilirliği için önemli bir temel oluşturur. Şekil 2.2'de VS Code logosu bulunmaktadır.

Şekil 2.2 Visual Studio Code Logosu

2.2.4 Geliştirme Sürecini Kolaylaştıracak Kütüphaneler

2.2.4.1 OpenCV:

OpenCV (**Open Source Computer Vision Library**), işitme engelli bireyler için yapay zeka destekli işaret dili çeviri sistemi geliştirmekte kullandığımız güçlü bir açık kaynak görüntü işleme kütüphanesidir. Intel tarafından 1999 yılında başlatılan bu proje, zaman içinde sürekli olarak geliştirilmiş ve geniş bir kullanıcı kitlesine ulaşmıştır. Python başta olmak üzere C++ gibi popüler programlama dillerini desteklemesi, geliştiricilere büyük bir esneklik sağlamaktadır.

Projemizde OpenCV'yi, işaret dili tanıma, el hareketlerini algılama ve anlama gibi temel ve gelişmiş görüntü işleme görevlerini gerçekleştirmek için entegre ettik. Özellikle, işitme engelli bireyler için geliştirdiğimiz uygulamada, kullanıcıların işaret dilini gerçek zamanlı olarak anlamalarına yardımcı olacak algoritmaları OpenCV aracılığıyla uygulamaya koyduk. Bu kütüphane, elde edilen verileri analiz etmek ve işaret dilini metne veya seslere çevirmek için kritik bir rol oynamaktadır.

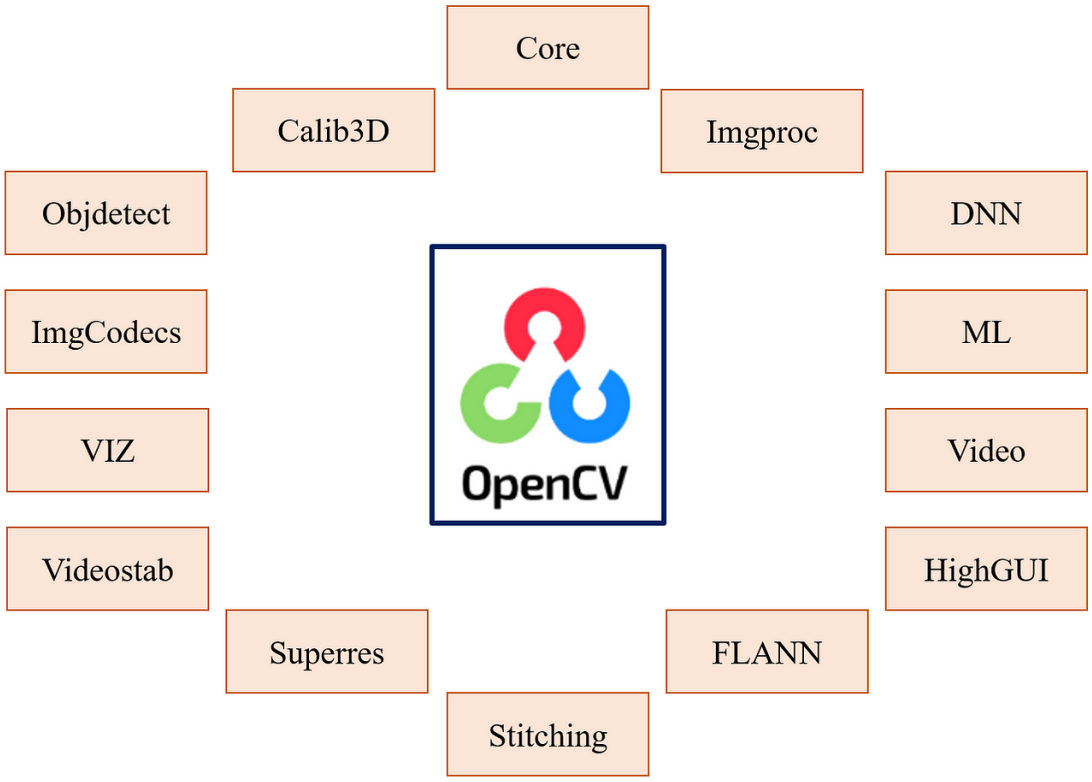
OpenCV'nin genişletilebilirlik, hızlı işleme yetenekleri, platform bağımsızlık ve ücretsiz olması gibi avantajları, projemizin başarılı bir şekilde hayata geçirilmesine ve işitme engelli bireylerin dijital içeriklere erişimini kolaylaştırmaya büyük katkı sağlamaktadır. Ayrıca, OpenCV'nin kapsamlı belgeleri ve aktif topluluğu, geliştirme sürecinde karşılaşılan zorlukların üstesinden gelmemize ve çözümler üretmemize yardımcı olmuştur. Şekil 2.3'te OpenCv logosu bulunmaktadır. [3]

Şekil 2.3 OpenCV Kütüphane Logosu

Fotoğrafta, ***(Sekil 1.1.OpenCV Modüllerinin Geniş Kapsamı: Görüntü İşleme, Nesne Algılama ve Daha Fazlası)*** OpenCV'nin çeşitli modüllerini temsil eden logonun detayları bulunmaktadır. Logonun kenarlarında yer alan yazılar, OpenCV kütüphanesinin farklı yeteneklerini ve modüllerini ifade etmektedir.

Bu modüller arasında Superres, Stitching, FLANN, VideoStab, Viz, ImgCodecs, ObjDetect, Core, ImgProc, DNN gibi görüntü işleme ve yapay zeka alanlarında kullanılan özellikler bulunmaktadır. Ayrıca, ML (Machine Learning) ve Video gibi geniş işlevsel gruplar da OpenCV'nin sunduğu çeşitli kullanım senaryolarını yansıtmaktadır.

Proje sürecinde belirtilen modüllerden bazıları doğrudan kullanılmamış olsa da, OpenCV'nin sağladığı geniş işlevsellik ve kütüphane desteği projemizin gelişimine katkı sağlamıştır (Şekil 2.4).



Şekil 2.4 OpenCV Modüllerinin Geniş Kapsamı: Görüntü İşleme, Nesne Algılama ve Daha Fazlası

Bazı temel OpenCV modüllerinin açıklamaları bulunmaktadır:

flann: Hızlı Yakınsak En Yakın Komşu (Nearest Neighbor) Arama Kütüphanesi olarak bilinir. Bu modül, büyük veri setleri üzerinde etkili arama işlemleri gerçekleştirmek için kullanılır.

highgui: Basit ve kullanıcı dostu grafik kullanıcı arayüzü (GUI) işlevleri sunar. Görüntü işleme uygulamalarında kullanıcı etkileşimini yönetmek için temel araçlar sağlar.

video: Video işleme işlevlerini içeren bu modül, video akışları üzerinde işlem yapmayı ve video dosyalarını okumayı, yazmayı sağlar.

dnn (Deep Neural Networks): Derin Sinir Ağı modülü, derin öğrenme model ve algoritmalarını uygulamak için kullanılır. Özellikle nesne tanıma ve sınıflandırma gibi görevler için kullanılır.

imgproc(Image Processing): Görüntü işleme işlevlerinin çoğunu içeren temel modüldür. Görüntü düzenleme, filtreleme, morfolojik işlemler gibi işlevleri içerir.

calib3d(Camera Calibration and 3D Reconstruction): Kamera kalibrasyonu, stereo görüntü eşleme ve 3D modelleme işlemlerini gerçekleştirmek için kullanılır.

objdetect(Object Detection): Nesne tespiti ve sınıflandırması için kullanılan bu modül, önceden eğitilmiş modelleri ve özellikleri kullanarak nesneleri algılamayı sağlar.

imgcodecs(Image Codecs): Görüntü dosyalarını okuma ve yazma işlemleri için gerekli olan çeşitli formatları destekler.

viz(Visualization): 3 boyutlu veri görselleştirmesi ve manipülasyonu için kullanılır. Özellikle stereo görüntüler ve 3D modeller üzerinde çalışırken kullanışlıdır.

videostab(Video Stabilization): Video stabilizasyon işlevleri sağlar, titreme veya sallantıları azaltarak daha pürüzsüz video akışları elde etmeyi sağlar.

2.2.4.2 Tensorflow:

TensorFlow, açık kaynaklı bir makine öğrenimi ve derin öğrenme kütüphanesidir ve Google tarafından geliştirilmektedir. Derin öğrenme modelleri oluşturmak, eğitmek ve dağıtmak için kullanılan güçlü bir araçtır. TensorFlow, özellikle büyük veri setleri üzerinde karmaşık öğrenme süreçlerini gerçekleştirmek için tasarlanmıştır. Kütüphane, makine öğrenimi projeleri için geniş bir yelpazede kullanılabilecek modüler araçlar ve kaynakları içerir.

TensorFlow'un başlıca özelliklerinden biri, kullanıcılarına derin öğrenme modellerini hızlı ve etkili bir şekilde oluşturma yeteneğidir. Hem araştırmacılar hem de endüstri profesyonelleri tarafından yaygın olarak kullanılan bu kütüphane, görüntü tanıma, doğal dil işleme, ses analizi gibi çeşitli uygulamalarda başarılı sonuçlar elde etmek için uygun algoritmalar sunar.

TensorFlow, aynı zamanda TensorFlow Lite gibi platformlar arası entegrasyon araçlarıyla mobil ve yerleşik sistemlerde kullanımı da destekler. Bu, özellikle taşınabilir cihazlarda veya gömülü sistemlerde uygulama geliştirme süreçlerini kolaylaştırır. TensorBoard gibi araçlar, model eğitimi ve performansını görselleştirmek için kullanıcı dostu bir arabirim sunar.

İşaret dili tanıma projeleri için TensorFlow, önceden eğitilmiş derin öğrenme modellerini kullanma, transfer öğrenme uygulama ve özelleştirilmiş modeller oluşturma konusunda geliştiricilere esneklik sağlar. Şekil 2.5'te TensorFlow logosu bulunmaktadır. [4]

Şekil 2.5 TensorFlow Kütüphanesinin Logosu

2.2.4.3 Keras

Keras, yüksek seviyeli bir derin öğrenme API'sidir ve özellikle TensorFlow ile sıkı entegrasyona sahiptir. Derin öğrenme modeli oluşturma, eğitim ve değerlendirme süreçlerini kullanıcı dostu bir arayüzle sağlar. Özellikle işaret dili tanıma gibi çeşitli uygulamalar için idealdir. Keras'ın temel özellikleri, modüler ve esnek tasarımıyla dikkat çeker.

Keras, önceden eğitilmiş derin öğrenme modelleri sunarak kullanıcıların transfer öğrenme yöntemini etkin bir şekilde kullanmalarını sağlar. Popüler modellerin önceden eğitilmiş versiyonlarının bulunması, yeni projelerde hızlı başlangıç yapmayı mümkün kılar. Ayrıca, çeşitli aktivasyon fonksiyonlarını destekleyen Keras, sigmoid, ReLU, tanh gibi yaygın olarak kullanılan fonksiyonlarla modelin öğrenme sürecini optimize etme imkanı sunar.

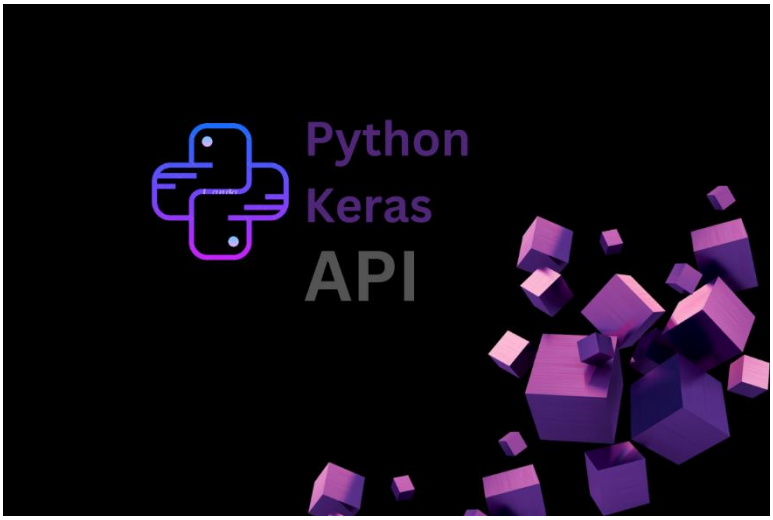
GPU desteği sayesinde Keras, büyük veri setleri üzerinde GPU tabanlı hesaplamalarla eğitim süreçlerini hızlandırır. Bu özellik, derin öğrenme modellerinin büyük ve karmaşık veri setleri üzerinde etkili bir şekilde çalışabilmesini sağlar. Keras'ın sağladığı bu avantajlar, projelerinizdeki derin öğrenme uygulamalarının performansını artırırken, geliştirme süreçlerini de daha verimli hale getirir.

Projemizde Keras kütüphanesini kullanarak, derin öğrenme modellerini oluşturduk, eğittik ve değerlendirdik. Keras'ın sunduğu bu kapsamlı özellikler sayesinde projelerimizin başarıyla tamamlanmasına ve teknik zorlukların üstesinden gelmemize önemli katkı sağladı. Şekil 2.6'da Keras görseli mevcuttur.

Keras'ın temel özellikleri şunlardır:

Modüler ve Esnek Tasarım: Keras, modüler bir yapıya sahiptir ve kullanıcıların kolayca özel derin öğrenme modelleri oluşturmalarını sağlar. Katmanlar, aktivasyon fonksiyonları ve optimizasyon algoritmaları gibi bileşenlerin kolayca birleştirilebilmesi, projelerin gereksinimlerine uygun modellerin tasarlanmasını kolaylaştırır.

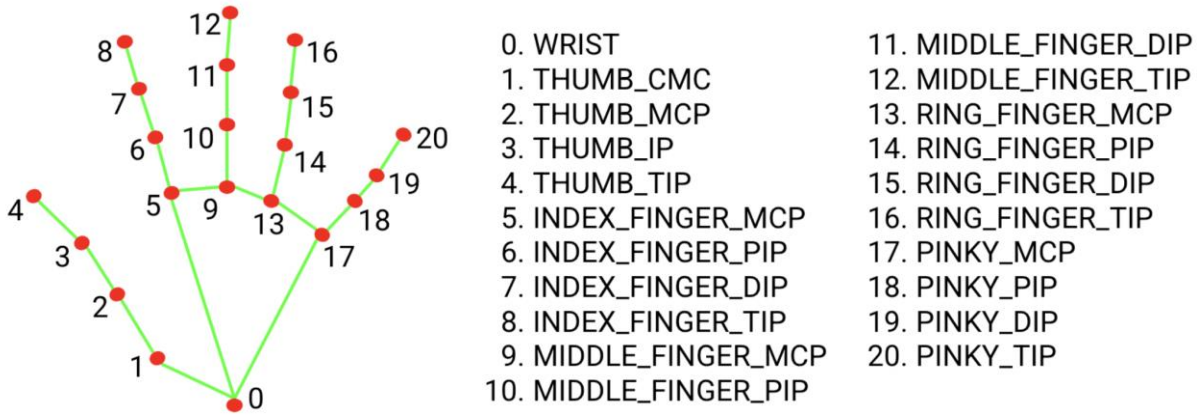
Önceden Eğitilmiş Modeller: Keras, popüler derin öğrenme modellerinin önceden eğitilmiş versiyonlarını içerir. Bu, kullanıcıların transfer öğrenme yöntemini kullanarak kendi projelerine hızlı bir başlangıç yapmalarını sağlar. [5]



Şekil 2.6 Keras Kütüphanesi

2.2.4.4 Mediapipe

MediaPipe, Google tarafından geliştirilen bir açık kaynaklı işaret işleme kütüphanesidir. Bu kütüphane, çeşitli görsel işleme görevlerini gerçekleştirmek ve uygulamalara entegre etmek için kullanılır. MediaPipe, özellikle el, yüz, vücut ve nesne izleme gibi görsel algılama görevlerini kolaylaştırmak için tasarlanmıştır. El hareketleri Şekil 2.7de, vücut hareketleri Şekil 2.8'de gösterilmektedir. İşitme engelli bireylerle iletişimi geliştirmek ve işaret dili projeleri gibi uygulamalarda kullanılabilecek bir dizi özelliğe sahiptir. MediaPipe'in bazı temel özellikleri şunlardır:



Şekil 2.7 Mediapipe'ın algıladığı elde koordinatlarını döndürdüğü noktalar

Hands (Eller) Modülü: MediaPipe, ellerin izlenmesi ve analizi için özel bir "Hands" modülü içerir. Bu modül, el hareketlerini algılamak ve izlemek için kullanılır. İşitme engelli bireylerle etkileşimli bir sistemde, işaret dili hareketlerini tanımak için el izleme bu modül aracılığıyla gerçekleştirilebilir.

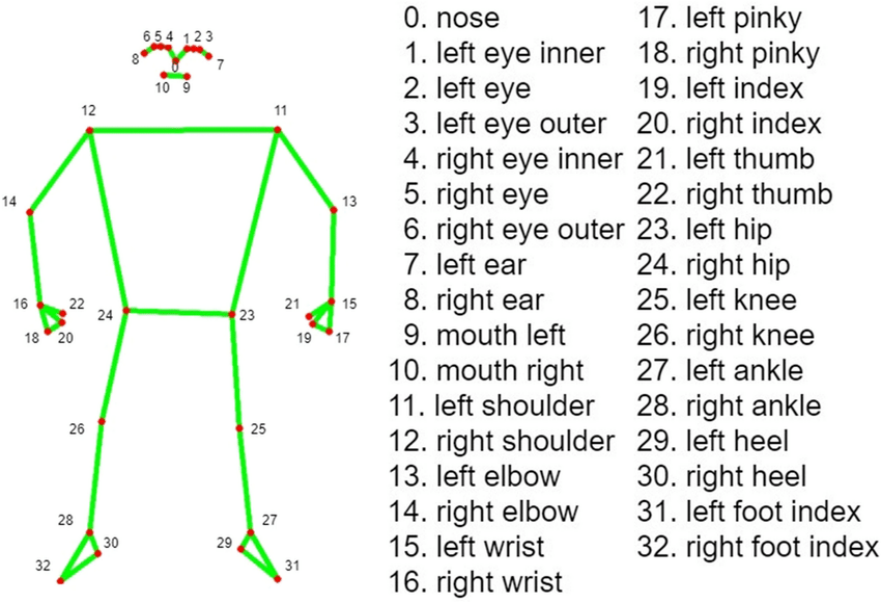
"Hands landmark" noktaları, elin belirli bölgelerindeki önemli noktaları temsil eder. Her bir landmark, eldeki belirli bir bölgeyi ifade eder. Bu noktalar, elin bileşenlerini, pozisyonunu ve hareketlerini anlamak için kullanılır. Bu noktalar genellikle 3D uzayda belirli koordinatlarla temsil edilir.

Örneğin, "wrist" (bilek) landmark'i elin bilek bölgesini temsil eder ve genellikle "0" numarası ile ifade edilir. "thumb" (baş parmak) bölgesindeki landmark'lar "1" ila "4" numaraları arasında yer alır, "index" (işaret parmağı) bölgesindeki landmark'lar "5" ila "8" numaraları arasında yer alır.

Bu "landmark" noktaları, elin pozisyonunu ve hareketlerini izlemek, analiz etmek veya tanımak için kullanılır. Özellikle işaret dili tanıma projeleri gibi uygulamalarda, bu noktalar, belirli el hareketlerini veya işaretleri tanımlamak için önemli bir rol oynar.

Holistic Modülü: "Holistic" modülü, vücut, el ve yüz izleme işlemlerini birleştiren geniş kapsamlı bir modüldür. İşitme engelli bireylerle iletişimde, yüz ifadelerini, el hareketlerini ve vücut pozisyonlarını takip ederek çeşitli işaret dili ifadelerini anlamak için kullanılabilir.

Birden Çok Platform Desteği: MediaPipe, birden çok platformda (Windows, macOS, Linux, Android, iOS) kullanılabilir. Bu özellik, projenizi farklı cihazlarda uygulamak veya taşımak istediğinizde esneklik sağlar.



Şekil 2.8 Mediapipe'ın algıladığı vücutta koordinatlarını döndürdüğü noktalar

Yüz ve El İzleme: MediaPipe, yüz ve el izleme işlemlerini kolaylaştıran özel modüllere sahiptir. Bu modüller, yüz ifadelerini ve el hareketlerini doğrulamak ve analiz etmek için kullanılabilir. İşitme engelli bireylerle etkileşimli bir sistemde, yüz ifadeleri ve el işaretleri önemli bir rol oynar.

"Face landmark" noktaları, yüz üzerinde belirli bölgeleri temsil eden referans noktalarıdır ve genellikle görsel işleme uygulamalarında yüz izleme veya analizi için kullanılır. Her bir nokta, yüzdeki belirli bir anatomi bölgesini temsil eder ve bu sayede yüzün farklı bölgelerindeki detayları belirleme olanağı sağlar. Örneğin, bu noktalardan biri, burunun ucu, diğerleri sol göz, sağ göz, ağızın sol köşesi, ağızın sağ köşesi ve çene gibi yüzdeki belirli özelliklere karşılık gelir.

Yüzdeki farklı bölgeleri temsil eden bu noktalar, çeşitli analizler ve uygulamalar için temel oluşturur. Örneğin, ağız köşeleri olan 9. ve 10. noktalar arasındaki mesafe, bir kişinin gülümsüyor olup olmadığını belirlemek için kullanılabilir. Aynı şekilde, gözlerin konumu ve burunun ucu gibi diğer noktalar, yüz ifadelerini tanımlamak ve anlamak için kullanılabilir. Bu "face landmark" noktaları, bilgisayarlı görü ve yapay zeka uygulamalarında yüz tabanlı etkileşimleri izlemek, yüz ifadelerini analiz etmek ve yüz animasyonları oluşturmak gibi birçok uygulama için hayati öneme sahiptir.

Yüz izleme teknolojisi, genellikle kamera görüntülerini işleyerek bu "face landmark" noktalarını belirler. Bu, yüz analitiği, duygu analizi ve yüz tabanlı etkileşimlerin geliştirilmesi gibi birçok alanda kullanılabilir. Yüz izleme, bilgisayarlı görü ve yapay zeka alanlarında önemli bir araştırma ve uygulama konusu olmuştur, çünkü yüz ifadelerini anlama, tanıma ve yorumlama yeteneği birçok uygulama için önemlidir. [6]

2.2.4.5 PyTorch

PyTorch, araştırma odaklı derin öğrenme projeleri için tasarlanmış açık kaynaklı bir makine öğrenimi kütüphanesidir. Esnek ve hızlı hesaplama yetenekleri sunan PyTorch, özellikle TensorFlow gibi diğer büyük derin öğrenme kütüphaneleriyle rekabet eder. Grafiksel işlemler yerine dinamik hesaplama grafiği kullanan bir yapısı vardır, bu da kullanıcıların modelin işleyişini anlamasını ve değiştirmesini kolaylaştırır. PyTorch logosu Şekil 2.9'da gösterilmektedir.

PyTorch'ün temel özellikleri şunlardır:

Dinamik Hesaplama Grafiği: PyTorch, dinamik hesaplama grafiği kullanarak işlem yapar. Bu, her bir iterasyonda grafiği yeniden oluşturarak esneklik ve kontrol sağlar. Kullanıcılar, her bir adımda grafiği değiştirme ve modelin işleyişini anlama konusunda büyük bir esneklik kazanır.

Python Entegrasyonu: PyTorch, Python diline doğrudan entegre olmuştur ve Python dilinin sunduğu tüm avantajları kullanır. Bu, kullanıcıların modeli oluşturma, eğitme ve değerlendirme süreçlerinde Python'un geniş ekosisteminden yararlanmasını sağlar.

GPU Hesaplama Desteği: PyTorch, GPU üzerinde hesaplama yapabilme yeteneği sunar. Bu özellik, büyük veri setleri üzerinde derin öğrenme modellerini eğitmek için gereken hesaplama gücünü artırır ve işlemleri hızlandırır.

Hızlı Prototip Geliştirme: Araştırmacılar ve mühendisler için ideal olan PyTorch, hızlı prototip geliştirme imkanı sunar. Modelleri hızlıca oluşturabilir, değiştirebilir ve test edebilirsiniz. Bu, yenilikçi fikirlerin ve önerilerin hızla hayata geçirilmesini sağlar.

Geniş Kullanıcı Topluluğu ve Ekosistem: PyTorch, geniş bir kullanıcı topluluğuna sahiptir ve sürekli olarak geliştirilmektedir. Bu da belgelendirme, öğreticiler, eğitim kaynakları ve sorun giderme için zengin bir ekosistem sunar.

PyTorch'ü projemizde kullanarak, derin öğrenme modelleri oluşturduk, eğittik ve değerlendirdik. Dinamik hesaplama grafiği yapısı sayesinde modelimizin karmaşıklığını yönetebildik ve geliştirdiğimiz çözümleri hızla prototip düzeyinde test edebildik. GPU desteği ile büyük veri setleri üzerinde etkin bir şekilde çalışarak projelerimizin performansını artırdık.

Şekil 2.9 PyTorch Kütüphanesi

2.2.4.6 Numpy

Numpy, Python programlama dilinde kullanılan güçlü bir sayısal hesaplama kütüphanesidir. Temel olarak çok boyutlu dizilerle çalışmayı sağlayan Numpy, bilimsel ve matematiksel hesaplamalar için optimize edilmiş bir dizi işlev ve araç sunar. İşitme engelli bireylerle etkileşimli sistemler veya işaret dili projeleri gibi uygulamalarda, Numpy kütüphanesi, veri manipülasyonu, analizi ve işlemenin yanı sıra sayısal hesaplamalar için güçlü bir araç olarak öne çıkar. Numpy logosu şekil 2.10'da gösterilmektedir.

Numpy kütüphanesinin temel özellikleri şunlardır:

Çok Boyutlu Diziler (Arrays): Numpy, homojen veri tipleriyle çalışan çok boyutlu dizileri destekler. Bu, büyük veri setlerinin hızlı ve etkili bir şekilde işlenmesine olanak tanır.

Hızlı Matematiksel İşlemler: Numpy, vektör ve matris işlemleri gibi matematiksel operasyonları hızlı bir şekilde gerçekleştirebilen özel uygulamalara sahiptir. Bu, işitme engelli bireylerle etkileşimli sistemlerde matematiksel operasyonların hızlı ve doğru bir şekilde yapılmasını sağlar.

Rastgele Sayı Üretimi: Numpy, rastgele sayı üretme işlevleri sunar. Bu, işaret dili projelerinde kullanılacak örnek veri setlerini oluşturmak veya test senaryolarını çeşitlendirmek için faydalıdır. [7]

Şekil 2.10 Numpy Kütüphanesi

2.2.4.7 OS

Python'un standart kütüphanelerinden biri olan os, işletim sistemi işlevlerine erişim sağlamak için kullanılır. Bu kütüphane, Python uygulamalarının taşınabilirliğini artırır ve çeşitli işletim sistemi işlevlerine programatik olarak erişmeyi sağlar. os modülü, dosya ve dizin yönetimi, işletim sistemi üzerinde komut çalıştırma, çevre değişkenleri ve kullanıcı hesap bilgileri gibi birçok işlevi içerir. Bu sayede Python programları, farklı işletim sistemlerinde tutarlı bir şekilde çalışabilir ve sistem kaynaklarını etkin bir şekilde yönetebilir. OS ile ilgili görsel Şekil2.11'dedir.

Dosya ve Dizin Yönetimi: os modülü, dosya ve dizin işlemleri için bir dizi işlev sunar. Dosya oluşturma, silme, yeniden adlandırma gibi temel işlemleri yapabilirsiniz. Ayrıca, dosya ve dizinlerin varlığını kontrol etme, mevcut dizini değiştirme gibi işlevler de bulunur.

İşletim Sistemi Bağımsızlığı: os modülü, Python programlarının işletim sisteminden bağımsız olarak çalışmasını sağlar. Bu, programlarınızın farklı platformlarda (Windows, Linux, macOS vb.) aynı şekilde davranmasını ve aynı işlevselliği sunmasını sağlar.

Çevre Değişkenleri ve Kullanıcı Bilgileri: os modülü, işletim sistemi çevresinde tanımlanmış çevre değişkenlerine (environment variables) ve kullanıcı bilgilerine erişim sağlar. Örneğin, geçici dizin, kullanıcı ana dizini gibi bilgilere programatik olarak erişebilirsiniz.

İşletim Sistemi Fonksiyonlarına Erişim: os modülü, işletim sistemi işlevlerine erişim sağlayarak, sistem komutlarını çalıştırabilir ve işletim sistemi üzerinde çeşitli görevleri yerine getirebilir. Bu, sistem kaynaklarını yönetmek ve sistem düzeyinde görevleri otomatikleştirmek için güçlü bir araç sağlar.

Dosya Yolu İşlemleri: os modülü, dosya yolları üzerinde işlemler yapmayı sağlar. Dosya yolunu birleştirme, genişletme, mutlak ve göreceli yol işlemleri gibi işlevler bu kütüphane içinde yer alır.

İşlem Yönetimi: os modülü, işletim sistemi üzerinde çalışan işlemlerle ilgili işlevler de içerir. Örneğin, işlemi sonlandırma, işlem kimliği (PID) alma gibi işlemleri yapabilirsiniz.

Platform Bağımsızlığı: Python'un platform bağımsız doğası sayesinde os modülü, farklı işletim sistemlerinde tutarlılık sağlar. Bu, Python programlarının farklı ortamlarda kolayca dağıtılmasını ve çalıştırılmasını mümkün kılar.

Şekil 2.11 OS Kütüphanesi

2.2.4.8 Tkinter

Tkinter, Python'un standart kütüphanelerinden biridir ve Grafiksel Kullanıcı Arayüzü (GUI) uygulamaları geliştirmek için kullanılır. Tkinter, Tk GUI toolkit'inin Python sürümüdür ve birçok işletim sisteminde (Windows, macOS, Linux) doğrudan çalışabilir. Tkinter ile ilgili görsel Şekil 2.12'dedir.

Özellikleri ve Kullanım Alanları

Tkinter, basit ve hızlı GUI uygulamaları geliştirmek için idealdir. Birçok temel widget (pencere, düğme, etiket, metin kutusu vb.) ve layout yöneticileri (pack, grid, place) sunar. Bu widget'lar, kullanıcı arayüzünün farklı bölümlerini ve öğelerini oluşturmak için kullanılır.

Tkinter, Python diline derinlemesine entegre edilmiştir ve dilin doğal yapılarına uygun olarak çalışır. Bu, Python programcıları için öğrenmesi ve kullanması kolaydır.

Widget'lar ve Özelleştirme

Tkinter, çeşitli widget'lar sunar ve bu widget'ları özelleştirmek için geniş seçenekler sunar. Örneğin, düğmelerin renklerini, etiketlerin metin stilini veya metin kutularının boyutunu ve biçimini kolayca değiştirebilirsiniz.

Olaylar ve İşlevsellik

Kullanıcı etkileşimlerini yönetmek için Tkinter, olay tabanlı programlama modelini destekler. Kullanıcı bir düğmeye tıkladığında veya bir metin kutusuna yazı yazdığında gerçekleşen olaylar üzerine işlevler (callback fonksiyonlar) atanabilir.

Cross-platform Desteği

Tkinter, Python ile birlikte gelir ve dolayısıyla çoğu Python dağıtımında (Anaconda, Python.org) doğrudan kullanılabilir. Bu, geliştiricilerin farklı platformlarda (Windows, macOS, Linux) aynı kod tabanını kullanarak uygulamalar geliştirmesine olanak tanır.

Genişletilebilirlik

Tkinter, temel olarak bir GUI araç seti sunar, ancak ihtiyaçlarınıza göre genişletilebilir. Özel widget'lar oluşturabilir veya Tkinter'i diğer Python kütüphaneleriyle (örneğin, matplotlib ile grafikler entegre etmek) entegre edebilirsiniz.

Şekil 2.12 Tkinter Kütüphanesi

2.2.4.9 Math

Bizim projemizde math kütüphanesi, özellikle görüntü boyutlandırma ve konum hesaplamaları için kullanılmıştır. Kod, elin algılanan konumuna bağlı olarak kesilmiş bir görüntünün boyutunu yeniden ayarlamak için math.ceil() fonksiyonunu kullanır. Bu fonksiyon, görüntü boyutlarını yukarı doğru yuvarlamak için kullanılır, böylece elde edilen kesilmiş görüntüyü belirli bir boyuta yeniden boyutlandırırken doğru boyutları sağlar.

Örneğin, elin boyutuna bağlı olarak imgCrop adlı görüntüyü keser ve ardından bu görüntüyü math.ceil() kullanarak yeniden boyutlandırır. Bu işlem, elin boyutlarına göre görüntünün genişlik ve yükseklik değerlerini doğru bir şekilde hesaplamak için önemlidir, böylece işlenmiş görüntü uygun bir şekilde yeniden boyutlandırılabilir. Math kütüphanesi ile ilgili görsel Şekil 2.13'tedir.

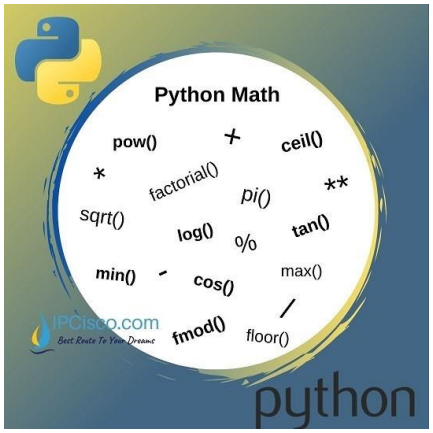
Math kütüphanesi, math, matematiksel işlemleri gerçekleştirmek için geliştirilmiş bir kütüphanedir. Bu kütüphane, çeşitli matematiksel fonksiyonlar, sabitler ve sayısal işlemler için fonksiyonlar içerir. İşte math kütüphanesinin sağladığı bazı temel işlevler:

Temel Matematik Fonksiyonları: math kütüphanesi, temel matematiksel işlemleri gerçekleştirmek için bir dizi fonksiyon içerir. Örneğin, math.sqrt() (karekök alma), math.pow() (üs alma), math.log() (doğal logaritma), math.exp() (e^x fonksiyonu), math.sin(), math.cos(), math.tan() (sinüs, kosinüs, tanjant fonksiyonları) gibi işlemler bulunur.

Sayılar ve Sabitler: math kütüphanesi, pi sayısı (math.pi), Euler sabiti (math.e), sonsuzluk (math.inf), negatif sonsuzluk (math.inf), NaN (sayısal olarak tanımsız) gibi sabitleri içerir. Bu sabitler, matematiksel hesaplamalar için standart değerler sağlar.

Trigonometrik Fonksiyonlar: math kütüphanesi, trigonometrik işlemleri gerçekleştirmek için standart trigonometrik fonksiyonları içerir. Örneğin, sinüs (math.sin()), kosinüs (math.cos()), tanjant (math.tan()), ark sinüs (math.asin()), ark kosinüs (math.acos()), ark tanjant (math.atan()) gibi fonksiyonlar, geometrik veya fiziksel hesaplamalarda kullanılabilir.

Sayısal İşlemler: math kütüphanesi, sayısal işlemler için çeşitli fonksiyonlar sunar. Örneğin, math.ceil() (yukarı yuvarlama), math.floor() (aşağı yuvarlama), math.trunc() (tamsayıya yuvarlama), math.fabs() (mutlak değer alma), math.factorial() (faktöriyel hesaplama) gibi işlemler, matematiksel hesaplamalarda kullanılan temel işlemlerdir.



Şekil 2.13 Math Kütüphanesi

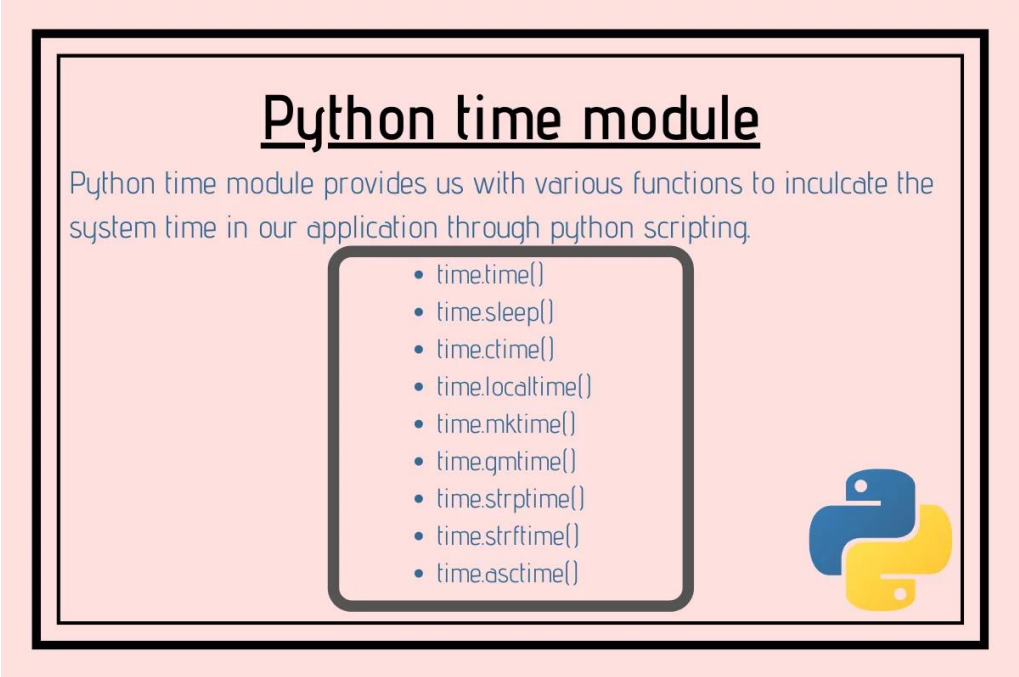
2.2.4.10 Time

Time modülü, Python programlamada zamanla ilgili işlemleri gerçekleştirmek için kullanılan bir standart kütüphanedir. Bu modül, zamanı ölçmek, zaman damgaları oluşturmak, zaman aralıkları hesaplamak ve zamanla ilgili diğer çeşitli işlemleri yapmak için işlevler sağlar. Time kütüphanesinin kapsamlı analizi Şekil 2.6'de sunulmuştur

Bu modülün en temel işlevlerinden biri, zamanı ölçmek ve işlem sürelerini hesaplamaktır. time.time() fonksiyonu, sistem saati üzerinden geçen zamanı saniye cinsinden döndürür. Bu özellik, kodunuzun belirli bölümlerinin ne kadar sürede çalıştığını ölçmek veya performans analizi yapmak için kullanılır.

Time.sleep(saniye) fonksiyonu ise belirli bir süre boyunca programınızın çalışmasını duraklatır. Bu işlev, zaman gecikmeleri eklemek veya belirli aralıklarla işlemleri gerçekleştirmek için kullanışlıdır. Örneğin, bir döngü içinde belirli aralıklarla veri okumak veya işlemek için time.sleep() fonksiyonu kullanılabilir.

Ayrıca, time modülü, zaman damgaları oluşturmak ve tarih-zaman bilgilerini işlemek için de kullanılır. time.localtime() ve time.strftime(format, time_struct) gibi fonksiyonlar, tarih ve saat bilgilerini istenen formatta formatlamak için kullanılır. Bu tür işlevler, dosya oluşturma tarihleri, günlük zaman damgaları veya zamanı temsil eden diğer bilgileri işlemek için kullanışlıdır. Time kütüphane fonksiyonları Şekil 2.14'te gösterimektedir.



Şekil 2.14 Time kütüphanesine ait bazı fonksiyonlar

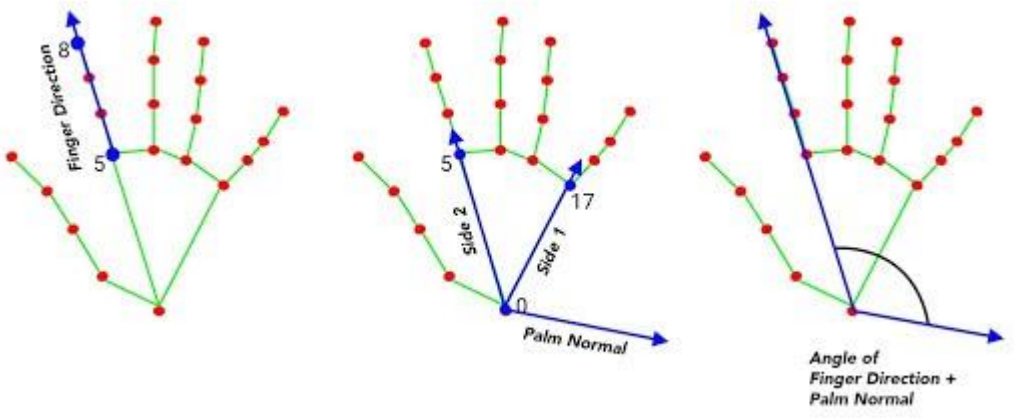
2.2.4.11 HandDetector

HandDetector, el algılama ve takip işlemleri için özel olarak tasarlanmış bir Python kütüphanesidir. Bu kütüphane, cvzone kütüphanesinin bir parçası olarak sunulmaktadır ve cv2 (OpenCV) kütüphanesiyle entegre çalışır. HandDetector, bilgisayarla görme (computer vision) tekniklerini kullanarak video akışı veya görüntü verisi üzerinde el tespiti yapar ve bu elin sınırlayıcı kutularını (bbox) belirler.

Kullanıcıların ellerinin pozisyonunu ve belirli el hareketlerini belirlemek için kullanılır. Proje içinde HandDetector kullanımı, detector = HandDetector(maxHands=1) şeklinde bir nesne oluşturarak başlar. maxHands parametresi, tespit edilecek maksimum el sayısını belirler (örneğin, burada 1 olarak ayarlanmıştır). findHands() metodunu kullanarak her video karesinde elleri tespit eder ve bu işlem sonucunda elin sınırlayıcı kutularını ve diğer özelliklerini sağlar.

HandDetector kütüphanesi, interaktif uygulamalar, sanal gerçeklik (VR), hareket yakalama sistemleri ve benzeri alanlarda kullanıcı etkileşimini sağlamak amacıyla tercih edilir. Sunduğu yetenekler sayesinde, kullanıcıların doğal ve etkileşimli bir şekilde ortamla etkileşim kurmalarını sağlayabilir.

Bu kütüphane, cvzone kütüphanesi içinde yer aldığından, cv2 (OpenCV) ile birlikte kullanılarak görüntü işleme ve el algılama projelerinde güçlü bir araç olarak hizmet verir. Şekil 2.15, projemizde kullanılan HandDetector kütüphanesinin el algılama ve takip işlemlerinde nasıl uygulandığını görsel olarak detaylandırmaktadır.



Şekil 2.15 El Algılama ve Takip için HandDetector Gösterimi

2.2.4.12 Speech Recognition

Speech Recognition, Python programlama dilinde ses tanıma işlemleri gerçekleştirmek için kullanılan bir kütüphanedir. Bu kütüphane, kullanıcıların mikrofonlarından gelen sesleri algılayabilir, bu sesleri farklı ses tanıma motorları aracılığıyla metin formatına dönüştürebilir ve bu dönüşüm sonuçlarını işleyebilir.

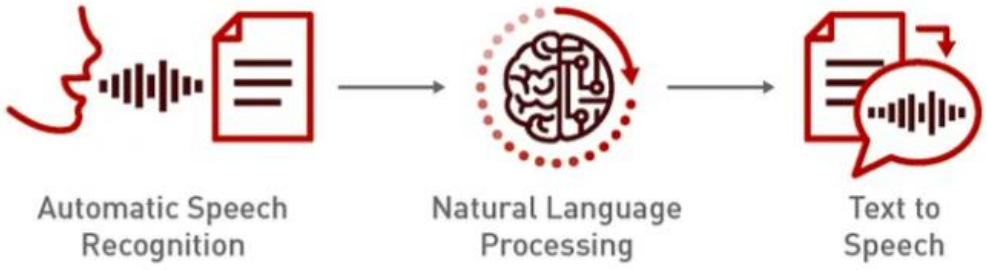
Kütüphane, genellikle aşağıdaki gibi çeşitli senaryolarda kullanılır:

Sesli Komut İşleme: Kullanıcıların sesli komutlarla etkileşime girebileceği uygulamalar geliştirme.

Metin Yönetimi: Sesle girilen verileri otomatik olarak metin formatına dönüştürme ve bu metinleri işleme.

Konuşma Tabanlı Arayüzler: Bilgisayar sistemlerinin veya cihazların sesle kontrol edilebilmesi için kullanım.

Speech Recognition kütüphanesi, çeşitli tanınma motorlarını destekler. Bunlar arasında Google Speech Recognition, IBM Speech to Text, CMU Sphinx gibi popüler motorlar bulunur. Kütüphane, bu motorlar aracılığıyla yüksek doğrulukta ses tanıma işlemleri gerçekleştirir ve kullanıcıların doğal dilde etkileşime girebilmelerini sağlar. Sekil 2.16'da ses tanıma teknolojileri ve metin işleme süreçleri gösterilmektedir.



Şekil 2.16 Ses Tanıma ve Metin İşleme Teknolojileri

2.2.4.13 Threading

Python'un standart kütüphanelerinden biri olan threading, çoklu iş parçacığı (thread) desteği sağlayan bir modüldür. İş parçacıkları, aynı anda çalışabilen bağımsız işlemlerdir ve Python'da paralel programlama için kullanılırlar. threading modülü, birden fazla işlemin eşzamanlı olarak çalışmasını sağlamak amacıyla kullanılır.

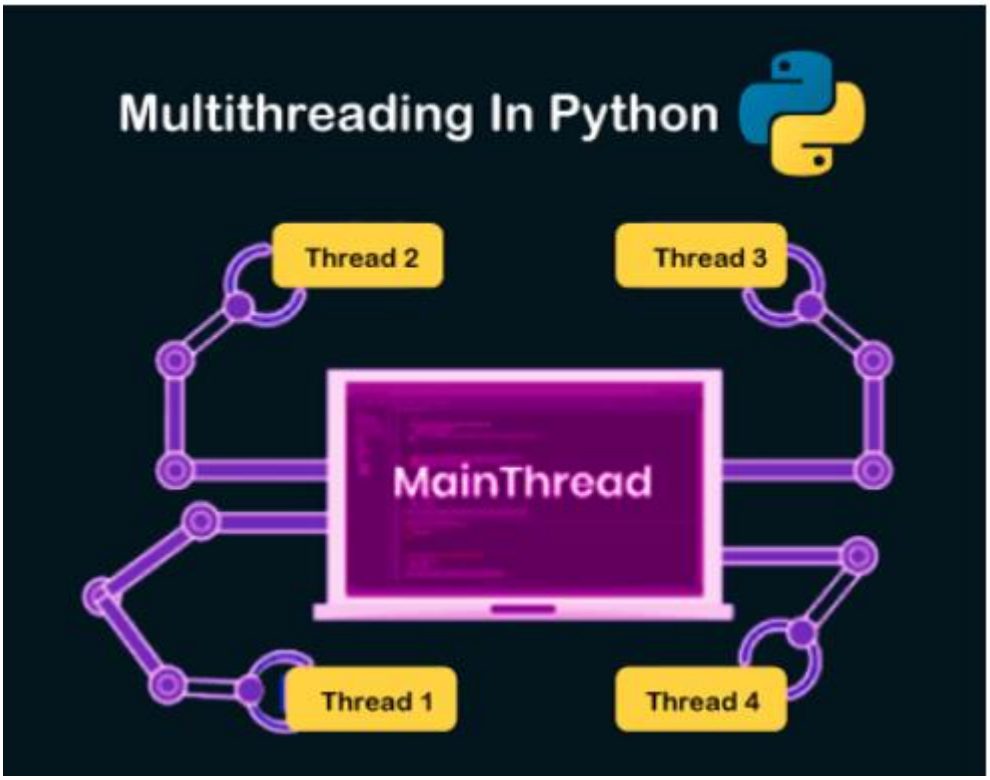
Temel Özellikler ve Kullanım Alanları:

Çoklu Görev Yönetimi: İş parçacıkları sayesinde, programlarınızı aynı anda birden fazla işi yapabilecek şekilde tasarlayabilirsiniz. Örneğin, ağ işlemleri, dosya okuma/yazma gibi işlemleri eşzamanlı olarak gerçekleştirebilirsiniz.

Performans İyileştirmesi: Uygun şekilde kullanıldığında, threading modülü programınızın performansını artırabilir. Özellikle uzun süren işlemleri paralel olarak çalıştırarak, toplam işlem süresini azaltabilirsiniz.

Senkronizasyon ve Veri Paylaşımı: İş parçacıkları arasında veri paylaşımı ve senkronizasyon mekanizmaları sağlar. Bu sayede, iş parçacıkları arasında güvenli bir şekilde veri iletebilir veya senkronize edebilirsiniz.

GUI Uygulamaları ve Arayüz Tepkisi: GUI (Grafiksel Kullanıcı Arayüzü) uygulamalarında, threading modülü kullanılarak arayüzü donmadan işlem yapabilen uygulamalar geliştirilebilir. Örneğin, uzun süren bir işlemi arka planda iş parçacıkları kullanarak çalıştırabilir ve arayüzün donmasını önleyebilirsiniz. Sekil 2.17'de Python'da multithreading kavramı görsel olarak temsil edilmiştir.



Şekil 2.17 Multithreading Kavramı

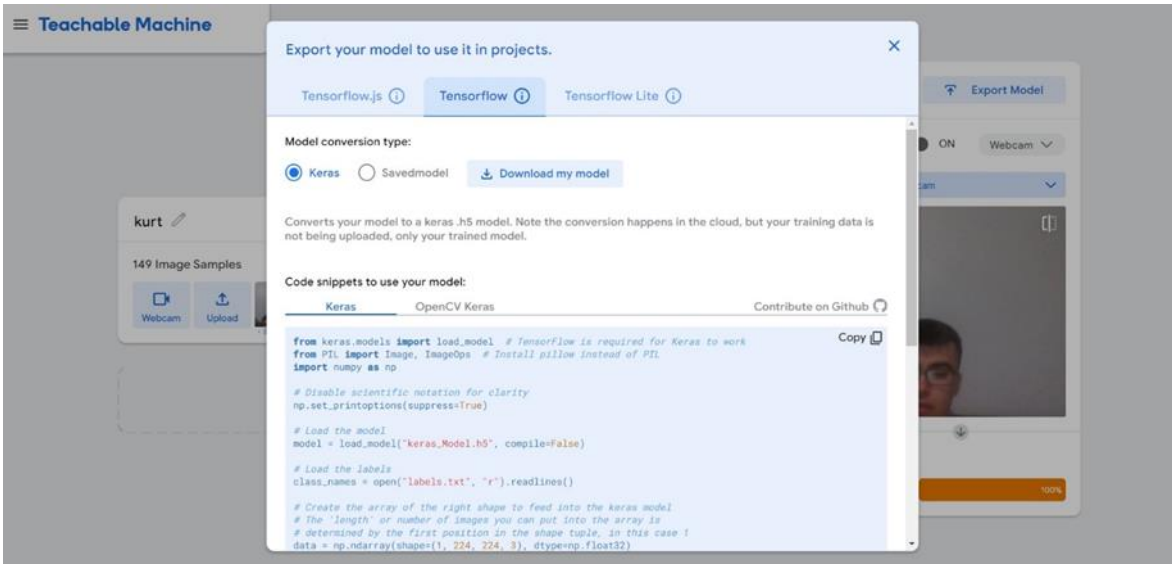
BÖLÜM 3

MANTIKSAL TASARIM

Veri setlerindeki fotoğrafların net, düzenli, ayırt edilebilir vb. olması makine öğrenmesi için oldukça önemlidir. Modelin doğruluk oranının yüksek olması için birçok yöntem denenmiştir.

3.1 Oluşturulan Veri Seti

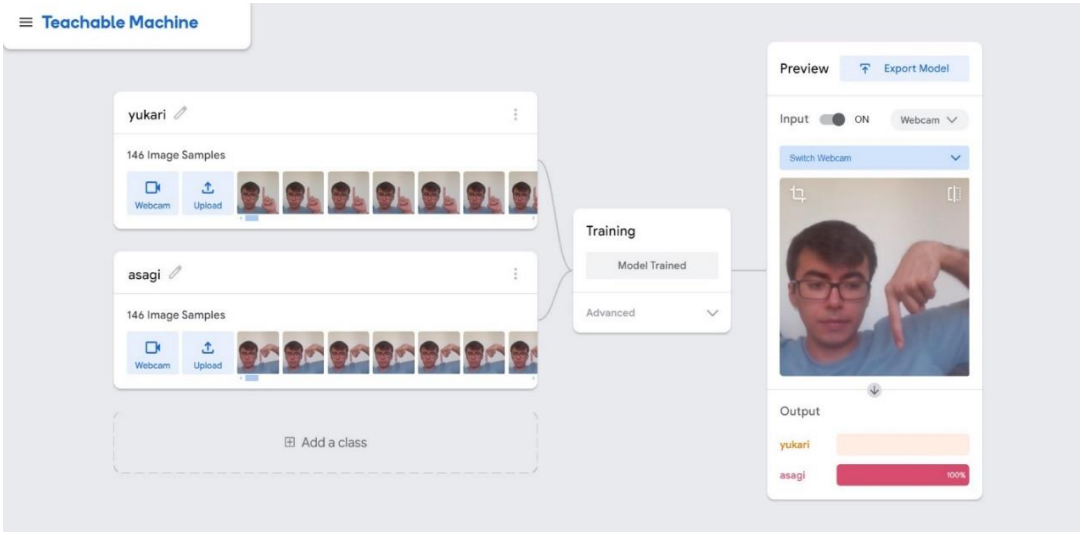
Veri setini oluşturmak için ilk adımda, el işaretlerinin fotoğraflarını çekmek amacıyla bir Python scripti yazdık (datacollection.py). Bu script ile çeşitli el işaretlerinin fotoğraflarını topladık. Topladığımız bu fotoğrafları, Teachable Machine web sitesindeki image project kısmına yükledik. Teachable Machine aracılığıyla fotoğrafları eğiterek, modelin el işaretlerini tanıyabilmesi için gerekli eğitim sürecini tamamladık. Eğitim sonrasında, modelin çıktılarını içeren Keras h5 ve etiketleri içeren txt dosyalarını aldık. Bu platform Şekil 3.1'de gösterilmektedir. Bu dosyaları, el işaretlerini tanıma ve sınıflandırma işlemlerinde kullanmak üzere projemizin koduna entegre ettik. Bu şekilde, veri setimizi oluşturup, modelimizi eğiterek projemizde kullanıma hazır hale getirdik. 110 kelime mevcuttur.



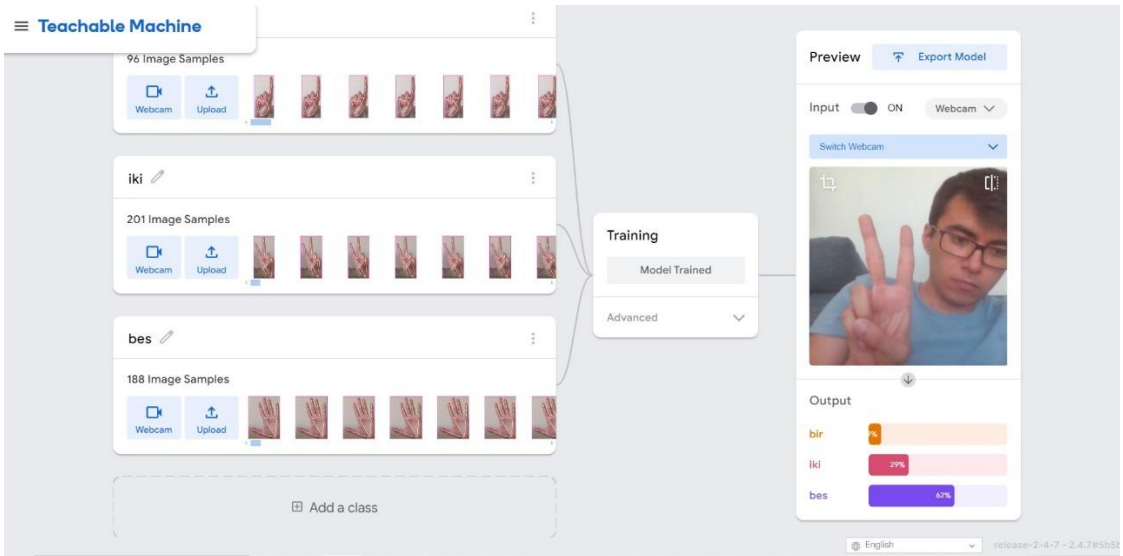
Şekil 3.1 Teachable Machine kullanılarak veri seti modelinin eğitilmesi

Oluşturulan veri seti yaklaşık 2500 fotoğraftan oluşuyor. Veri setindeki bir kelime veya harf 100-200 arası fotoğraftan oluşuyor. Bazı kelimeler Şekil 3.2 Şekil 3.3 ve Şekil 3.4'te gösterilmektedir.
Veri setinin içerdiği kelimeler:

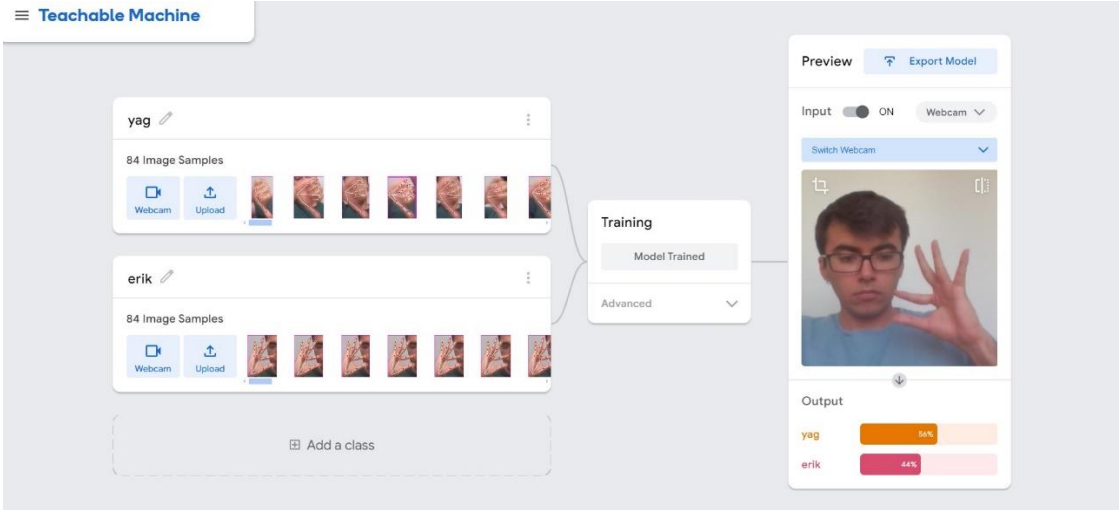
Eşya: anahtar, ayna, bardak, çakmak, ışık, makas, merdiven, musluk, tava, telefon
Harfler: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, R, S, T, U, V, Y, Z
Hayvanlar: akrep, arslan, at, balık, boğa, deve, eşek, hindi, horoz, kartal, kurt
Meslekler: asker, avukat, bakkal, doktor, hakim, hemşire, marangoz, müdür
Renkler: beyaz, gri, kahverengi, kırmızı, lacivert, mavi, pembe, sarı, siyah, turuncu
Sayılar: bir, iki, üç, dört, beş, altı, yedi, sekiz, dokuz
Taşıtlar: araba, dolmuş, gemi, helikopter, motosiklet, tır
Yiyecekler: armut, ayran, ayva, biber, elma, erik, helva, kayısı, kiraz, mısır, patates, patlıcan, şeftali, şeker, su, tuz, yağ
Yönler: arka, aşağı, batı, doğu, ön, sağ, sol, yukarı



Şekil 3.2 "aşağı" kelimesinin gösterimi



Şekil 3.3 "iki" kelimesinin gösterimi



Şekil 3.4 "erik" kelimesinin gösterimi

3.2 Veri Setinin Avantajları

Oluşturulan veri setinde el hareketleri kırpılarak işlendiği için kişiye özel değildir. Bu sayede birçok kişide doğru sonuç vermektedir. Veri seti her bir kelime için yaklaşık 200 fotoğraftan oluştuğu için farklı konumlardan da algılaması sağlanmaktadır.

3.3 Veri Setinin Dezavantjları

Ortam şartları değişince (ışık, mesafe gibi) ne yazık ki yanılmalar söz konusu olmaktadır. Gösterilen hareket ile sistemin verdiği cevap örtüşmemektedir. Doğru cevap alabilmek için çok deneme yapmak gerekmektedir. Bunun için kamera yönü değiştirmek, mesafeyi azaltmak gibi çözümler denenmelidir.

BÖLÜM 4

GELİŞTİRME

4.1 Arayüz

Bir arayüz tasarımı yapılmıştır. Bilgilendirme için sidebar tasarlandı. Kameraları açan butonlar mevcuttur. Kod şekil 4.1'de verilmiştir.

arayuz.py > ...

```python
import os
from subprocess import call
from customtkinter import *
from PIL import Image, ImageTk
import tkinter as tk

# Mevcut betiğin dizinini al
def get_script_directory():
    return os.path.dirname(os.path.abspath(__file__))

# Çalışma dizinini değiştir
def change_working_directory(script_dir):
    os.chdir(script_dir)

# Yeni bir pencere aç ve verilen metni göster
def open_new_window(text):
    new_window = CTk()
    new_window.geometry("980x600")
    new_window.resizable(0, 0)

    text_frame = tk.Frame(master=new_window, bg="black")
    text_frame.pack(pady=(30, 0), padx=30, fill="both", expand=True)

    text_widget = tk.Text(master=text_frame, font=("Arial", 15), wrap="word")
    text_widget.pack(pady=10, padx=10, fill="both", expand=True)

    text_widget.insert("1.0", text)
    text_widget.configure(state='disabled')
    text_widget.see("1.0")

    new_window.mainloop()

# Metin dosyasını okuma
def open_text_file(file_name):
    with open(os.path.join("Text", file_name), "r", encoding="utf-8") as file:
        return file.read()

```

Şekil 4.1.a Arayüz Kodu

```
# Belirli bir test dosyasını çalıştır
def run_test(test_file):
    call(["python", test_file])

# Kategori seçim penceresini aç
def open_category_window():
    category_window = CTk()
    category_window.geometry("300x500")
    category_window.resizable(0, 0)

    CTkLabel(master=category_window, text="Kategori Seçiniz", font=("Arial Bold", 20)).pack(pady=(20, 30))

    categories = ["Esya", "Fiil", "Hayvan", "Meslek", "Renk", "Sayi", "Tasit", "Yiyecek", "Yon"]
    for category in categories:
        CTkButton(master=category_window, text=category, font=("Arial", 12),
                  command=lambda c=category: run_test(f"Test/test_{c.lower()}.py")).pack(pady=5)

    category_window.mainloop()

# Cümleler penceresini aç
def open_sentences_window():
    # test_cumle.py dosyasını çalıştır
    run_test("Test/test_cumle.py")

    # subprocess kullanarak ses.py dosyasını aç
    call(["python", "ses.py"])


# Betik dizinini al ve çalışma dizinini değiştir
script_dir = get_script_directory()
change_working_directory(script_dir)

# Ana uygulama penceresi oluştur
app = CTk()
app.geometry("1200x650")  # Pencere boyutu 1200x650 olarak ayarlandı
app.resizable(0, 0)


# Karanlık mod ayarı
set_appearance_mode("dark")

# Yan panel oluştur ve butonlar ekle
sidebar_frame = CTkFrame(app, width=200)  # Sidebar genişliğini artırdık
sidebar_frame.pack(side="left", fill="y")

# Logo ekle
logo_image_path = "Foto/tu_logo.png"
logo_image = Image.open(logo_image_path).resize((180, 180))  # Resim boyutunu ayarladık
logo_image = ImageTk.PhotoImage(logo_image)

sidebar_label = tk.Label(sidebar_frame, image=logo_image)
sidebar_label.image = logo_image  # Referans tutmak için
sidebar_label.pack(pady=20)
sidebar_label.configure(bg="#282b2b")


buttons_info = {
    "İşaret Dili": "text1.txt",
    "Türk İşaret Dili": "text2.txt",
    "Proje Hakk□nda": "text3.txt",
    "Geliştirme Araçlar□": "text4.txt",
    "Proje Aşamalar□": "text5.txt",
    "Kaynak ve Referanslar": "text6.txt",
    "Proje Ekibi": "text7.txt",
    "Proje Dan□şman□": "text8.txt"
}

for btn_text, file_name in buttons_info.items():
    CTkButton(sidebar_frame, text=btn_text, font=("Arial", 12),
              command=lambda fn=file_name: open_new_window(open_text_file(fn))).pack(pady=10)

main_frame = CTkFrame(app)

# Ana başlık etiketi
CTkLabel(master=app, text="Dilsiz Çevirmeni",
         font=("Arial Bold", 20), justify="left").pack(anchor="w", pady=(10, 0), padx=(20, 0))

```

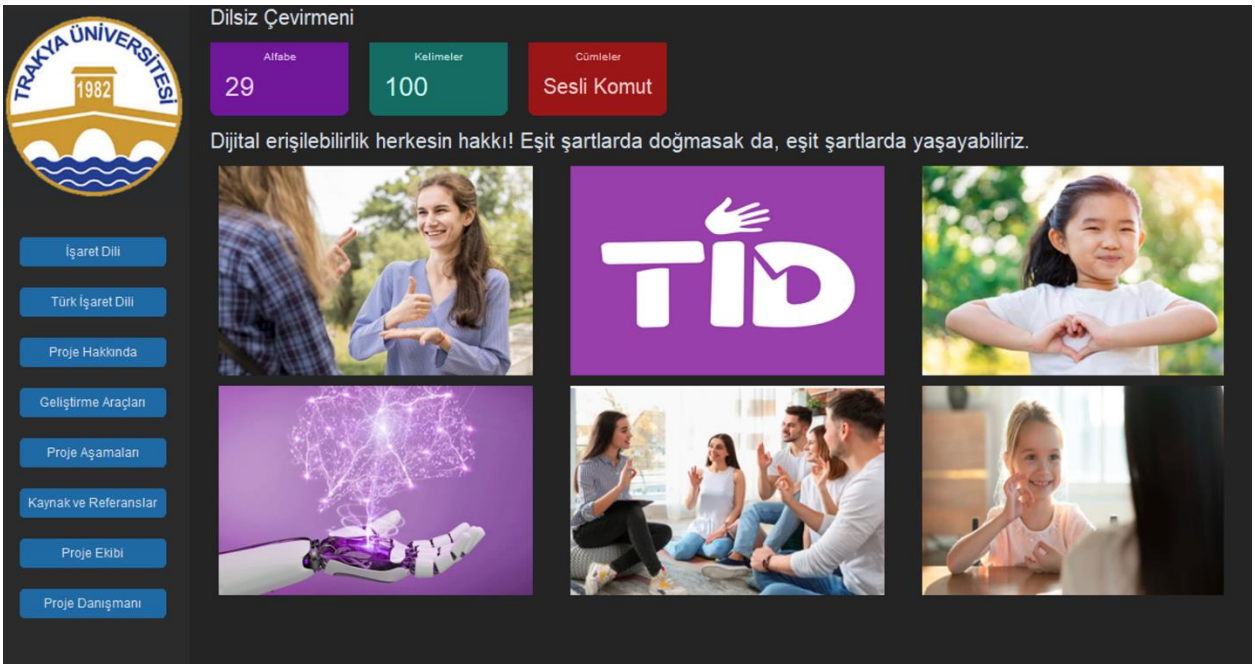
Şekil 4.1.b Arayüz Kodu

```
# İstatistik çerçeveleri
stats_frame = CTkFrame(master=app, fg_color="transparent")
stats_frame.pack(padx=(20, 0), pady=(10, 0), anchor="nw")

quizzes_taken_frame = CTkFrame(master=stats_frame, fg_color="#70179A", width=132, height=70, corner_radius=8)
quizzes_taken_frame.pack_propagate(0)
quizzes_taken_frame.pack(anchor="w", side="left", padx=(0, 20))

alpha_button = CTkButton(master=quizzes_taken_frame, text="Alfabe", font=("Arial Bold", 10), text_color="#F3D9FF",
                         command=lambda: run_test("Test/test_alfabe.py"), fg_color="#70179A")
alpha_button.pack(anchor="nw")
CTkLabel(master=quizzes_taken_frame, text="29", justify="left", font=("Arial Bold", 25), text_color="#F3D9FF").pack(anchor="nw", padx=(14, 0))

correct_qs_frame = CTkFrame(master=stats_frame, fg_color="#146C63", width=132, height=70, corner_radius=8)
correct_qs_frame.pack_propagate(0)
correct_qs_frame.pack(anchor="w", side="left", padx=(0, 20))

kelimeler_button = CTkButton(master=correct_qs_frame, text="Kelimeler", font=("Arial Bold", 10), text_color="#D5FFFB", command=open_category_window, fg_color="#146C63")
kelimeler_button.pack(anchor="nw")
CTkLabel(master=correct_qs_frame, text="100", justify="left", font=("Arial Bold", 25), text_color="#D5FFFB").pack(anchor="nw", padx=(14, 0))

highest_score_frame = CTkFrame(master=stats_frame, fg_color="#9A1717", width=132, height=70, corner_radius=8)
highest_score_frame.pack_propagate(0)
highest_score_frame.pack(anchor="w", side="left", padx=(0, 20))

cümleler_button = CTkButton(master=highest_score_frame, text="Cümleler", font=("Arial Bold", 10), text_color="#FFCFCF", command=open_sentences_window, fg_color="#9A1717")
cümleler_button.pack(anchor="nw")
CTkLabel(master=highest_score_frame, text="Sesli Komut", justify="left", font=("Arial Bold", 20), text_color="#FFCFCF").pack(anchor="nw", padx=(14, 0))

# Başlık etiketi
CTkLabel(master=app, text="Dijital erişilebilirlik herkesin hakkı! Eşit şartlarda doğmasak da, eşit şartlarda yaşayabiliriz.", font=("Arial Bold", 20),
         justify="left").pack(anchor="nw", side="top", padx=(20, 0), pady=(10, 0))

# Quiz çerçeveleri
quizzes_frame = CTkFrame(master=app, fg_color="transparent")
quizzes_frame.pack(pady=(10, 0), padx=(20, 0), anchor="nw")
```

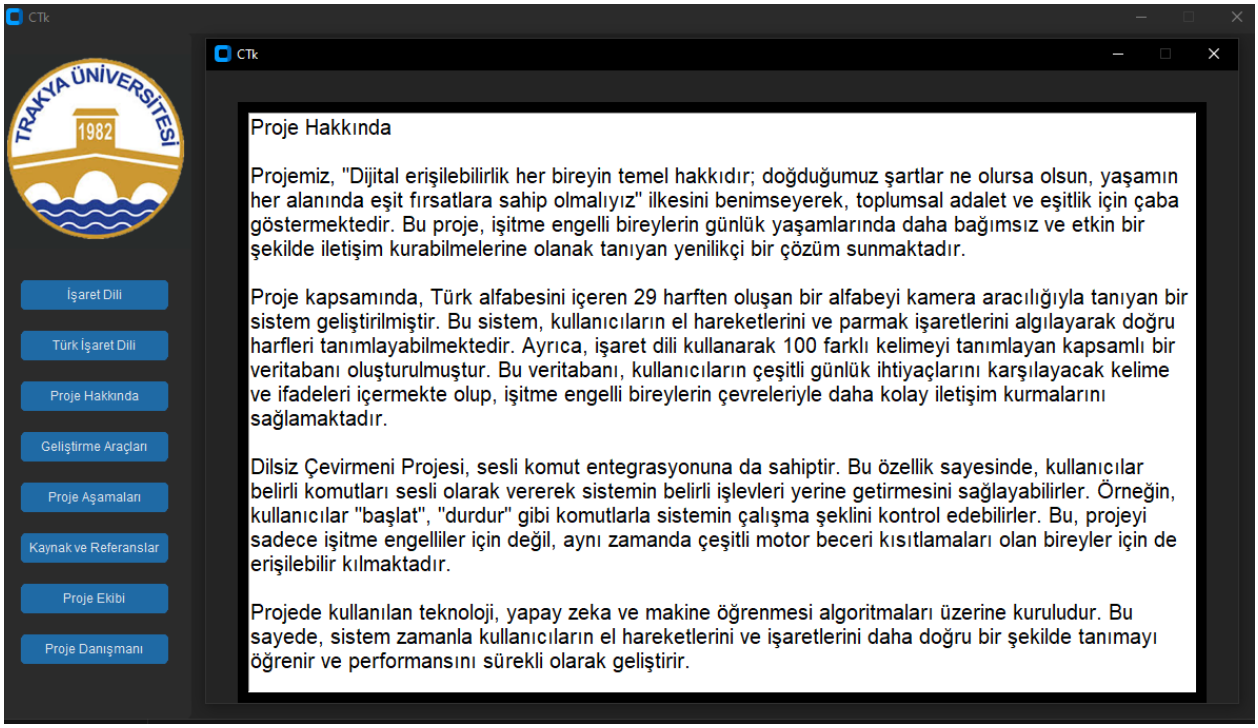
```
# Görsellerin listesi (resimlerin tıklanabilirliğini kaldırdık)
images = ["Foto/foto1.png", "Foto/foto2.png", "Foto/foto3.png", "Foto/foto4.jpg", "Foto/foto5.png", "Foto/foto6.png"]

# Görselleri 2x3 formatında düzenle
for i in range(0, len(images), 3):
    row_frame = CTkFrame(master=quizzes_frame, fg_color="transparent")
    row_frame.pack(anchor="nw", pady=(0, 10))

    for image_name in images[i:i+3]:
        img_data = Image.open(image_name).resize((300, 200))  # Resized to fit better in 2x3 layout
        img = CTkImage(light_image=img_data, dark_image=img_data, size=(300, 200))
        label = CTkLabel(master=row_frame, text="", image=img, corner_radius=8)
        label.pack(side="left", padx=(0, 20))

# Ana uygulama döngüsü
app.mainloop()
```

Şekil 4.1.c Arayüz Kodu

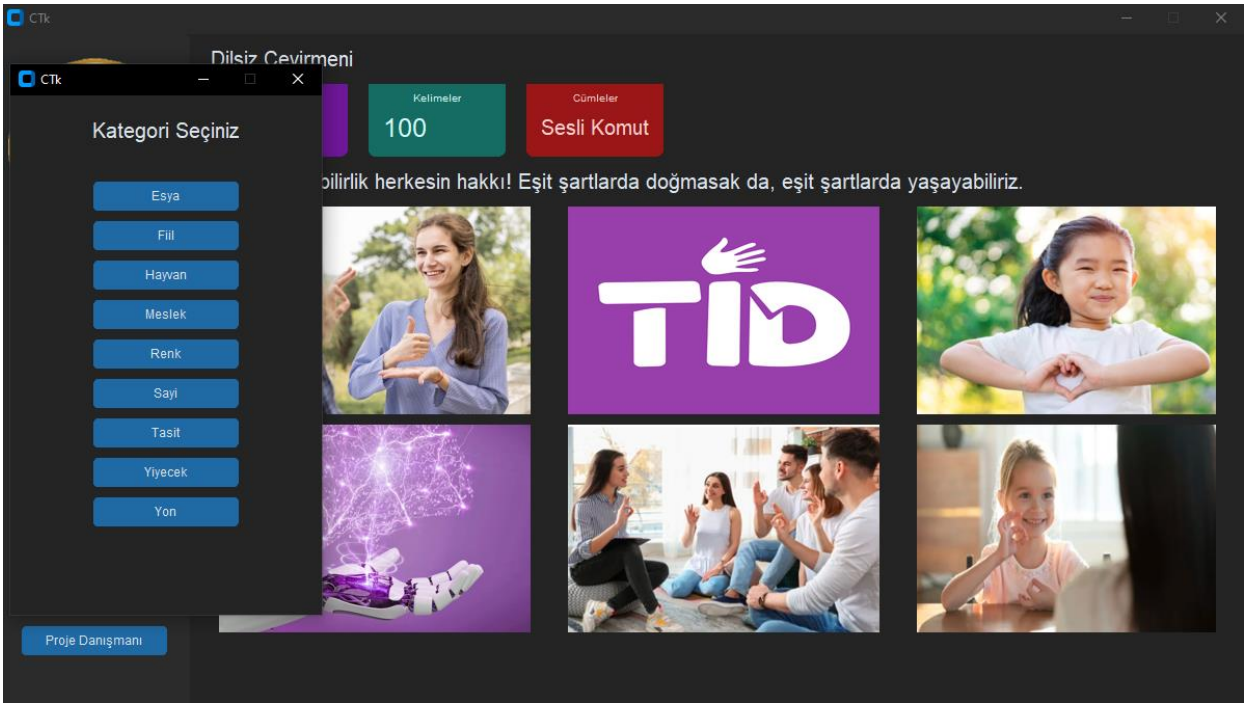
Sistem çalıştığında arayüz gösterimi Şekil 4.2'de verilmiştir. Arayüzde bilgilendirme amaçlı Sidebar mevcuttur. Şekil 4.3'te gösterilmektedir. Kategorilere ayrılmış butonlar sayesinde kamera aktif hale gelmektedir. Bu butonlar: alfabe, kelimeler ve cümledir. Kelimelere tıklandığı zaman kategori sayfası açılmaktadır. Şekil 4.4'te gösterilmiştir. Üniversite logosu ve çeşitli görseller ile tasarım tamamlanmıştır.



Şekil 4.2 Sistem Arayüzü



Şekil 4.3 Proje hakkında butonuna basılması

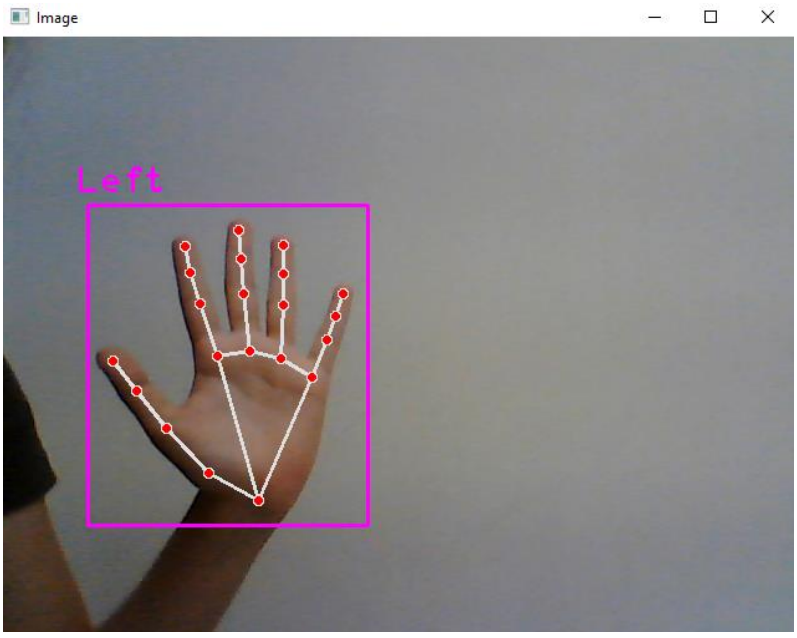


Şekil 4.4 Kategori Ekranı

4.2 DataCollection

DataCollection kısmında dataset oluşturulmuştur. Kamera açılmış ve kırpılmış fotoğraf çekilmiştir. Fotoğraf "Data" dosyasında ilgili dosyanın içine jpg uzantısıyla kaydedilmektedir. Bu kısım tek eli algılamaktadır.

Ekran çıktısı Şekil 4.5'te verilmiştir Kod Şekil 4.6'da verilmiştir.



Şekil 4.5 DataCollection Ekran Çıktısı

datacollection.py > ...

```python
#Dataset oluşturma

import cv2
from cvzone.HandTrackingModule import HandDetector
import numpy as np
import math
import time

cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)
offset = 20
imgSize = 300
counter = 0

folder = "Data/Gitmek"  #Oluşturulan resmin dosya yolu

#el kırpma
while True:
    success, img = cap.read()
    hands, img = detector.findHands(img)
    if hands:
        hand = hands[0]
        x, y, w, h = hand['bbox']

        imgWhite = np.ones((imgSize, imgSize, 3), np.uint8)*255

        imgCrop = img[y-offset:y + h + offset, x-offset:x + w + offset]
        imgCropShape = imgCrop.shape

        aspectRatio = h / w

        if aspectRatio > 1:
            k = imgSize / h
            wCal = math.ceil(k * w)
            imgResize = cv2.resize(imgCrop, (wCal, imgSize))
            imgResizeShape = imgResize.shape
            wGap = math.ceil((imgSize-wCal)/2)
            imgWhite[:, wGap: wCal + wGap] = imgResize

        else:
            k = imgSize / w
            hCal = math.ceil(k * h)
            imgResize = cv2.resize(imgCrop, (imgSize, hCal))
            imgResizeShape = imgResize.shape
            hGap = math.ceil((imgSize - hCal) / 2)
            imgWhite[hGap: hCal + hGap, :] = imgResize

        cv2.imshow('ImageCrop', imgCrop)
        cv2.imshow('ImageWhite', imgWhite)

    cv2.imshow('Image', img)
    key = cv2.waitKey(1)

    #resim kaydetme
    if key == ord("s"):
        counter += 1
        cv2.imwrite(f'{folder}/Image_{time.time()}.jpg', imgWhite)
        print(counter)

    #çıkış
    if key == ord("q"):
        break
```

Şekil 4.6 DataCollection Kodu

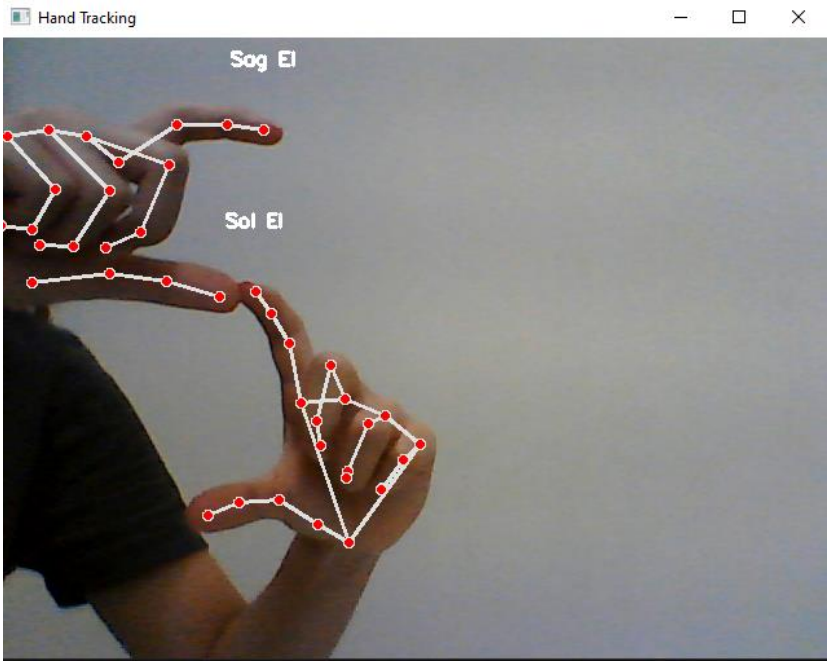
4.3 DataCollection2

DataCollection2 kısmında ise iki el algılama mevcuttur. Bu şekilde veri seti oluşturur. Kod Şekil 4.7'de verilmiştir. Ekran görüntüsü ekil 2.8'de verilmiştir.

datacollection2.py > ...

```python
import cv2
import mediapipe as mp
import os

# Mediapipe kütüphanesini yükler
mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils

# Elleri algılamak için bir Hands modeli oluşturur
hands = mp_hands.Hands(static_image_mode=False, max_num_hands=2, min_detection_confidence=0.5)

# Kamera açılır
cap = cv2.VideoCapture(0)

# Fotoğrafların kaydedileceği klasörü belirler
output_folder = "captured_photos"
os.makedirs(output_folder, exist_ok=True)

# Fotoğraf sayacı
photo_counter = 0

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        continue

    # Görüntüyü RGB'ye dönüştürür
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # Görüntüyü el işareti modeline verir
    results = hands.process(frame_rgb)

    # Eğer el algılandıysa
    if results.multi_hand_landmarks:
        # Her el için işaretçilerin konumlarını çizer
        for hand_landmarks in results.multi_hand_landmarks:
            mp_drawing.draw_landmarks(frame, hand_landmarks, mp_hands.HAND_CONNECTIONS)

            # Elin üstünde metin ekler
            if hand_landmarks.landmark[mp_hands.HandLandmark.INDEX_FINGER_TIP].x < hand_landmarks.landmark[
                mp_hands.HandLandmark.MIDDLE_FINGER_TIP].x:
                hand_side = "Sol El"
            else:
                hand_side = "Sag El"

            text_size = cv2.getTextSize(hand_side, cv2.FONT_HERSHEY_SIMPLEX, 0.5, 2)[0]
            cv2.putText(frame, hand_side, (int(hand_landmarks.landmark[mp_hands.HandLandmark.INDEX_FINGER_TIP].x * frame.shape[1] - text_size[0] / 2),
                                           int(hand_landmarks.landmark[mp_hands.HandLandmark.INDEX_FINGER_TIP].y * frame.shape[0] - 50)),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)

        # Bir tuşa basıldığında fotoğrafı kaydeder
        if cv2.waitKey(1) & 0xFF == ord('c'):
            photo_counter += 1
            photo_name = f"photo_{photo_counter}.jpg"
            photo_path = os.path.join(output_folder, photo_name)
            cv2.imwrite(photo_path, frame)
            print(f"Photo captured: {photo_name}")

    # Görüntüyü gösterir
    cv2.imshow('Hand Tracking', frame)

    # 'q' tuşuna basıldığında çıkış yapar
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Kaynakları serbest bırakır ve pencereleri kapatır
cap.release()
cv2.destroyAllWindows()
```

Şekil 4.7 DataCollection2 Kodu



Şekil 4.8 DataCollection2 Ekran Çıktısı

4.4 Ses

Ses kısmında sesi algılayarak yazıya çevirir ve aynı zamanda kamera açar. Bu sayede cümleyi çevirmektedir. Ekran görüntüsü Şekil 4.9'da verilmiştir Kod Şekil 4.10'da verilmiştir.

Şekil 4.9 Ses Ekran Çıktısı

ses.py > ...

```python
import speech_recognition as sr
import cv2
import customtkinter as ctk
import mediapipe as mp
from PIL import Image, ImageTk, ImageDraw, ImageFont
import threading
import numpy as np

# Konuşmayı metne dönüştürmek için tanıyıcıyı başlat
r = sr.Recognizer()

# Global değişkenler
cap = None
hands = None
user_text = ""
mp_hands = None

def speech_to_text_turkish():
    """Türkçe metin dönüştürme işlemi yapar."""
    global user_text
    try:
        # Giriş için mikrofonu kullan
        with sr.Microphone() as source:
            output_label.configure(text="Sesinizi kaydediyorum...")  # Kaydedildiğini kullanıcıya bildir (Türkçe)
            root.update()

            # Ortam gürültüsünü düzeltmek için ayarla
            r.adjust_for_ambient_noise(source, duration=0.2)

            # Kullanıcı girişini dinle
            audio = r.listen(source)

            # Türkçe modeliyle Google Konuşma Tanıma kullanarak konuşmayı tanı
            text = r.recognize_google(audio, language='tr-TR')
            output_label.configure(text="Dinledim: " + text)  # Tanınan metni ekranda göster (Türkçe)
            root.update()
            user_text = text.lower()

    except sr.RequestError as e:
        output_label.configure(text="İstek işlenemedi; {0}".format(e))  # Hata mesajını göster (Türkçe)
        root.update()
        user_text = None
    except sr.UnknownValueError:
        output_label.configure(text="Bilinmeyen hata oluştu")  # Genel hata mesajını göster (Türkçe)
        root.update()
        user_text = None

def on_key_press(event):
    """Tuşa basma olayını işlemek için fonksiyon"""
    if event.char == 'q':
        root.quit()

def reset_camera_and_speech():
    global cap, hands, user_text
    cap.release()
    user_text = ""
    output_label.configure(text="")
    initialize_camera()
    threading.Thread(target=main_loop).start()
    threading.Thread(target=speech_to_text_turkish).start()

def initialize_camera():
    global cap, hands, mp_hands
    cap = cv2.VideoCapture(0)
    mp_hands = mp.solutions.hands
    hands = mp_hands.Hands()
    return cap.isOpened()

def main_loop():
    global user_text
    while cap.isOpened():
        # Kameradan görüntü al
        ret, frame = cap.read()
        if not ret:
            break
```

Şekil 4.10.a Ses Kodu

```
        # Görüntüyü ters çevir (aynalama)
        frame = cv2.flip(frame, 1)

        # Renk uzayını BGR'den RGB'ye çevir
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        # El algılama işlemi yap
        result = hands.process(rgb_frame)

        # Tespit edilen elleri çiz
        if result.multi_hand_landmarks:
            for hand_landmarks in result.multi_hand_landmarks:
                mp_drawing.draw_landmarks(rgb_frame, hand_landmarks, mp_hands.HAND_CONNECTIONS)

        # Tanınan metni görüntüye yazdır
        if user_text:
            # Pillow ile yazı yazma
            pil_image = Image.fromarray(rgb_frame)
            draw = ImageDraw.Draw(pil_image)
            # Burada Türkçe karakterleri destekleyen bir font dosyası seçmelisiniz
            font = ImageFont.truetype("arial.ttf", 32)
            draw.text((10, 10), user_text, font=font, fill=(0, 255, 0))

            # Pillow görüntüsünü tekrar OpenCV formatına çevir
            rgb_frame = np.array(pil_image)

        # OpenCV görüntüsünü tkinter'da göster
        img = Image.fromarray(rgb_frame)
        imgtk = ImageTk.PhotoImage(image=img)
        camera_label.imgtk = imgtk
        camera_label.configure(image=imgtk)

        root.update_idletasks()
        root.update()

    cap.release()
    cv2.destroyAllWindows()

# CustomTkinter GUI penceresini oluştur
root = ctk.CTk()
root.title("Konuşma Tanıma ve El Algılama")

# Çıktı için etiket oluştur
output_label = ctk.CTkLabel(root, text="")
output_label.pack(pady=20)

# Kamera görüntüsü için bir etiket oluştur
camera_label = ctk.CTkLabel(root)
camera_label.pack(pady=20)

# Reset butonunu oluştur
reset_button = ctk.CTkButton(root, text="Reset", command=reset_camera_and_speech)
reset_button.pack(pady=20)

# Tuşa basma olayını bağla
root.bind("<KeyPress>", on_key_press)

# MediaPipe ve OpenCV kullanarak el algılama işlemini başlat
mp_drawing = mp.solutions.drawing_utils

if initialize_camera():
    threading.Thread(target=main_loop).start()
    threading.Thread(target=speech_to_text_turkish).start()
else:
    output_label.configure(text="Kamera açılamadı!")

root.mainloop()
```

Şekil 4.10.b Ses Kodu

BÖLÜM 5

SONUÇ

İşitme engelli bireylerin yaşadıkları zorluklar bilinmekle birlikte daha iyi anlaşılmıştır. İşaret dili sadece engelli bireyler tarafında değil tüm toplum tarafından bilinmesi gereken bir dildir. Çünkü iletişim tek taraflı değildir.

Bu projeye, yaşanılan bu iletişim sorununu en aza indirmek amacıyla başlanılmıştır. Proje ekibi tarafından veri seti oluşturulmuştur.

Projede Python dili kullanıldı. MediaPipe(), OpenCV() ve Speech Recognition() kullanılan başlıca kütüphanelerdir. Görüntü işleme yöntemleri sayesinde proje tamamlandı.

Böylece projemiz bu konuda çalışacak olanlara yol gösterici örnek bir proje olmuştur.

KAYNAKLAR

[1] https://tr.wikipedia.org/wiki/%C4%B0%C5%9Faret_dili

[2] https://www.python.org/

[3] https://tr.wikipedia.org/wiki/OpenCV

[4] https://en.wikipedia.org/wiki/TensorFlow

[5] https://pypi.org/project/keras/

[6] https://developers.google.com/mediapipe

[7] https://pypi.org/project/numpy/

ÖZGEÇMİŞ

Ad Soyad : Ozan Can SARI

Doğum Yeri :Merkez/Diyarbakır

Doğum Yılı :22.01.2001

Lise :2016-2020

Staj Yaptığı Yerler: NTT DATA Business Solutions Turkey

E-posta : 7ozzy35@gmail.com

Ad Soyad : Buse UYSAL

Doğum Yeri : Ezine/Çanakkale

Doğum Yılı : 20.09.2001

Lise : 2016-2020

Staj Yaptığı Yerler: Egemen Yazılım ve Otomasyon San. ve Tic. Ltd. Şti.

E-posta : buseuysal3158@gmail.com

Ad Soyad : Melisa VASİJA

Doğum Yeri : Prizren/Kosova

Doğum Yılı : 21.09.2001

Lise : 2017-2020

Staj Yaptığı Yerler: Best Vision Balkan

E-posta : vasijamelisa@gmail.com

Ad Soyad : Anal KARAALI

Doğum Yeri :Merkez/Tekirdağ

Doğum Yılı :28.05.2001

Lise :2015-2019

E-posta : anil.karaali59@gmail.com