

T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MEKATRONİK MÜHENDİSLİĞİ ANABİLİM DALI



KAMERA VE GÖRÜNTÜ İŞLEME KULLANILARAK BORU VE
PROFİL ÇAPININ ÖLÇÜLMESİ

YÜKSEK LİSANS TEZİ

NURETTİN ERSİN BAĞRIYANIK

DENİZLİ , TEMMUZ-2024

T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MEKATRONİK MÜHENDİSLİĞİ ANABİLİM DALI



KAMERA VE GÖRÜNTÜ İŞLEME KULLANILARAK BORU VE
PROFİL ÇAPININ ÖLÇÜLMESİ

YÜKSEK LİSANS TEZİ

NURETTİN ERSİN BAĞRIYANIK

DENİZLİ, TEMMUZ-2024

Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğinin; bu çalışmanın doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğini beyan ederim.

NURETTİN ERSİN BAĞRIYANIK



ÖZET

KAMERA VE GÖRÜNTÜ İŞLEME KULLANILARAK BORU VE PROFİL ÇAPININ ÖLÇÜLMESİ

YÜKSEK LİSANS TEZİ

NURETTİN ERSİN BAĞRIYANIK

PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

MEKATRONİK MÜHENDİSLİĞİ ANABİLİM DALI

(TEZ DANIŞMANI: Dr. Öğr. Üyesi Erdem DİLMEN)

DENİZLİ, TEMMUZ-2024

Bu tez Python dili ile kodlanmış olup, kamera ile anlık olarak görüntü işleme kullanılarak boru ve profil dış çap ölçümü yapan bir algoritmadır. Boru ve profil üretimi yapan demir çelik sanayisinde, sürekli olarak çap ölçümü yapılması gerekmektedir. Ölçüm yapılmasının sebebi belirli yerlere farklı ölçülerde değişen çaplarda boru ve profil üretilmesidir. Üretim esnasında şekil verilen slablarda makaraların konumlarına bağlı olarak bazen çaplarda oynama olmaktadır. Ölçüm manuel olarak kumpas ile yapıldığından dolayı bozuk çapta boru ve profil üretilmektedir. Üretimler ise satılamamakta ve fire olarak ayrılmaktadır. Bundan dolayı zaman kaybı yaşanmakta ve ciddi ölçüde zarar edilmektedir. Yapılan algoritma, görüntü işleme yöntemi kullanılarak OpenCV kütüphanesi aracılığı ile yapılmıştır. Algoritmanın amacı sürekli olarak kameradan çap ölçüm işlemini gerçekleştirilmesi ve bu sayede ilk seferde doğru bir üretim almaktır. Sistem otomatik olarak çap ölçümü gerçekleştirdiğinden dolayı bu yaklaşım ile birlikte manuel olarak ölçüm işlemini ortadan kaldırılmıştır. Ekran üzerinde boru ve profiller için ayrı ayrı seçenekler sunulmuştur, bu sayede hem boru hem de profiller için ölçüm yapılabilme imkanı sağlanmıştır. Tolerans değeri ile girilen aralık değerleri haricinde, kırmızı renk ile fark kadar, büyük, küçük şeklinde uyarı vermektedir. Girilen değerlere göre önceden ölçülmüş olan boru ya da profile göre kalibrasyon yapılması gerekmektedir. Ölçülen boru ya da profilin kamera ile mesafesini ve açısını ayarlamak gerekmektedir, bu sayede kalibrasyon yapılmış olmaktadır. Gri renk skalasına göre kontur ve eşikleme yapmaktadır. Bu sayede gri renk çerçeve içine alınarak ölçüm doğruluğu artırılmış olmaktadır. Yazılımda, ekipman olarak harici kamera kullanılmıştır ve bu sayede istenilen pozisyon aralığında görüntü alması kolaylaştırılmıştır. Kamera mesafesi ölçüm aleti ile ölçülen çapa göre değişmektedir.

ANAHTAR KELİMELER: BORU VE PROFİL ÜRETİMİNDE ÇAP ÖLÇÜMÜ, EŞ ZAMANLI ÖLÇÜM, GRİ RENK SKALASI, ÇAP TÖLERANS DIŞI HATA MESAJI, KAMERA İLE GÖRÜNTÜ İŞLEME

ABSTRACT

MEASUREMENT OF PIPE AND PROFILE DIAMETER USING CAMERA AND IMAGE PROCESSING

MSC THESIS

NURETTİN ERSİN BAĞRIYANIK

PAMUKKALE UNIVERSITY INSTITUTE OF SCIENCE

MECHATRONICS ENGINEERING

(SUPERVISOR: Asst. Prof. Dr. Erdem DİLMEN)

DENİZLİ, TEMMUZ-2024

This thesis is coded in Python and features an algorithm that performs real-time image processing to measure the outer diameter of pipes and profiles using a camera. In the iron and steel industry that produces pipes and profiles, continuous diameter measurement is required. This measurement is necessary because different diameters are produced in various locations. During production, due to the positions of the rollers on the slabs, diameter variations may occur. Measurements are currently done manually using calipers, which can result in the production of faulty pipes and profiles. These products cannot be sold and are categorized as waste, leading to time loss and significant financial damage. The developed algorithm utilizes image processing methods via the OpenCV library. Its purpose is to continuously measure diameters from the camera, ensuring correct production on the first attempt. Since the system automatically performs diameter measurements, this approach eliminates the need for manual measurement. Separate options for pipes and profiles are presented on the screen, allowing measurement for both. Outside the tolerance values entered, warnings are provided in red, indicating whether the measurement is too large or too small. Calibration must be done according to pre-measured pipes or profiles based on the entered values. The distance and angle of the measured pipe or profile from the camera need to be adjusted to achieve calibration. Contours and thresholding are done according to the grayscale, increasing measurement accuracy by enclosing the gray color within a frame. The software utilizes an external camera as equipment, facilitating image capture within the desired positional range. The camera distance varies according to the diameter measured by the measuring device.

KEYWORDS: DIAMETER MEASUREMENT IN PIPE AND PROFILE PRODUCTION, SIMULTANEOUS MEASUREMENT, GRAY SCALE, ERROR MESSAGE FOR OUT-OF-TOLERANCE DIAMETER, IMAGE PROCESSING WITH CAMERA.

İÇİNDEKİLER

Sayfa

ÖZET.....	i
ABSTRACT	ii
İÇİNDEKİLER	iii
ŞEKİL LİSTESİ.....	iv
TABLO LİSTESİ	vi
SEMBOL LİSTESİ	vii
ÖNSÖZ.....	viii
1. GİRİŞ ve AMAÇ	1
2. LİTERATÜR TARAMASI	4
2.1 Python Programlama Dili ve Uygulama Alanları	4
2.2 Görüntü İşlemenin Temelleri ve Kullanım Alanları	5
2.3 Görüntü İşleme ile Yapılan Uygulamalar.....	6
2.4 Lazer Kullanılarak Yapılan Görüntü İşleme Uygulamaları	17
2.4.1 Lazer Işığını Takip Eden Araç Prototip Tasarımı.....	17
2.4.2 Görüntü İşleme Yöntemi ile Optik İşaret Tanıma ve Yönlendirme Sistemi	17
2.5 Kamera ile Görüntü İşleme Kullanılarak Mesafe Ölçümü.....	18
2.5.1 Mono Kameralar ile Mesafe Ölçümü	18
2.5.2 Stereo Kameralar ile Mesafe Ölçümü.....	19
2.6 Arduino Python Haberleşmesi ile Yapılan Projeler	20
3. KAMERA VE GÖRÜNTÜ İŞLEME KULLANILARAK BORU VE PROFİL ÇAPININ ÖLÇÜLMESİ PROJESİNDEKİ KULLANILAN KODLAR	22
3.1 Ön Çalışma Arduino Kodları	22
3.2 Ön Çalışma Python Yazılımı Kodları	24
4. ÖN ÇALIŞMADAKİ EKRAN KODLARI	29
5. BORU PROFİL ÇAP ÖLÇÜM PROJESİNDEKİ PYTHON KODLARI.....	31
5.1 Kullanılan Kütüphaneler ve Ne için Kullanıldıkları	31
5.1.1 OpenCV (cv2) kütüphanesi	31
5.1.2 NumPy (np) kütüphanesi	32
5.1.3 Tkinter (tk) kütüphanesi.....	32
5.1.4 PIL (Pillow) kütüphanesi	32
6. PROJE EKRAN ÇIKTILARI.....	33
7. TARTIŞMA	44
8. SONUÇLAR VE ÖNERİLER.....	45
9. KAYNAKLAR.....	48
EKLER.....	Hata! Yer işareti tanımlanmamış.
EK A 52	
EK B 64	

ŞEKİL LİSTESİ

Sayfa

Şekil 2.1: Python kodunun exe koduna dönüşümü.....	4
Şekil 2.2: Nesne için CNN'nin katmanlı mimarisini göstermektedir.....	7
Şekil 2.3: Nesne tespiti için kullanılan havuzlama katmanları (Pathak ve diğ. 2018).	8
Şekil 2.4: Colab ortamında bir nesneyi tanıma kodları.	9
Şekil 2.5: Kod çıktısı sonuçları	10
Şekil 2.6: Solda bulunan orijinal görüntü üzerinde kopyalama-taşıma işlemi yapılması sonucunda sağda bulunan manipüle edilmiş görüntü elde edilmiştir (Mushtaq ve Mir 2014).	12
Şekil 2.7: Kusur tespiti program çıktısı (Jaykumaran 2023).	12
Şekil 2.8: Daire tespiti denklemi.	13
Şekil 2.9: Program kodları.....	14
Şekil 2.10: Gri tonlama kodu örneği.	14
Şekil 2.11: Giriş resmi.	15
Şekil 2.12: Çıkış resmi	15
Şekil 2.13: GUI Laser Guided System Control Panel arayüzüne ait örnek görsel.	17
Şekil 2.14: Mono bir kameraya ait görüş alanının mesafe ile değişimi.....	18
Şekil 2.15: Üçgenleme metodu.....	19
Şekil 2.16: İnternet kontrolü ile tasarlanan lamba devresi Aygör (2019).	21
Şekil 3.1: Arduino HC-SR04 ultrasonik sensör bağlantısı.	22
Şekil 3.2: Arduino kodunun yazım şekli.	22
Şekil 3.3: Arduino haberleşme kodu.....	23
Şekil 3.4: Arduino UNO void Loop () döngüsündeki kodlar.....	24
Şekil 3.5: Projede Python yazılımında kullanılan kütüphaneler.	24
Şekil 3.6: Projede kullanılan Arduino ile seri haberleşme ve video başlatma kodları.....	25
Şekil 3.7: Tkinter penceresi oluşturulması için kullanılmış kodlar.	25
Şekil 3.8: Projede Tkinter arayüz penceresi oluşturma.	25
Şekil 3.9: Projede mesafe bilgisi için etiketlendirme.	26
Şekil 3.10: Ekranda mavi çizgi çizilmesi.	26
Şekil 3.11: Projedeki ana kodlar.	27
Şekil 4.1: Terminaldeki ekran ölçüm çıktıları.	29
Şekil 4.2: Gerçek ölçüm.	29
Şekil 4.3: GUI penceresi Tkinter görüntüsü.	30
Şekil 4.4: Ultrasonik mesafe sensörü.	30
Şekil 5.1: Yazılımda kullanılan kütüphane kodları.....	31
Şekil 6.1: Profil uzunluğu kumpas ölçüsü.	33
Şekil 6.2: profil genişliği kumpas ölçüsü.	34
Şekil 6.3: Profil ölçüm sonucu.	35
Şekil 6.4: Ölçüm sonucu 2.....	36
Şekil 6.5: Kare profil genişlik kumpas ölçümü.	37
Şekil 6.6: Kare profil uzunluk kumpas ölçümü.	38
Şekil 6.7: Kare profil ölçüm sonucu.	39
Şekil 6.8: Kumpas boru ölçüm sonucu.	40
Şekil 6.9: Boru ölçüm sonucu ilk deneme.	41
Şekil 6.10: Boru ölçüm sonucu ikinci deneme.	42
Şekil 6.11: Kamera ile cisim arasındaki mesafe.	43
Şekil A.1: Kod dizimi.	51

Şekil A.2: diameterMeasurementApp sınıfı.	52
Şekil A.3: hedef çap ve tolerans için girişler.	54
Şekil A.4: Çap ölçüm uygulaması GUI ekranı ve girişleri.	56
Şekil A.5: Profil seçeneği seçildiği zaman.	59
Şekil A.6: Boru seçeneği seçildiği zaman.	59
Şekil A.7: Radyo düğmeleri, gri ton aralıkları ve sonuçlar etiketleri kodları.	58
Şekil A.8: ölçüm esnasında çerçeve içine alınmış profil.	59
Şekil A.9: boru-profil seçim deaktif kodları.	60
Şekil A.10: update image kütüphanesi.	61
Şekil A.11: gri tonlama ve eşikleme.	61
Şekil A.12: konturları bulma.	62
Şekil B.1: Boru denemesi 1 kumpas ölçümü.	63
Şekil B.2: Boru denemesi 1 kamera ölçümü.	63
Şekil B.3: Kamera ve boru konumlanması.	64
Şekil B.4: Kamera ve boru arasındaki mesafe.	64
Şekil B.5: Boru denemesi 2 kumpas ölçümü.	65
Şekil B.6: Boru denemesi 2 kamera ölçümü.	65
Şekil B.7: Boru denemesi 3 kumpas ölçümü.	66
Şekil B.8: Boru denemesi 3 kamera ölçümü.	66
Şekil B.9: Boru denemesi 4 kumpas ölçümü.	69
Şekil B.10: Boru denemesi 4 kamera ölçümü.	69
Şekil B.11: Profil denemesi 1 kısa kenar kumpas ölçümü.	70
Şekil B.12: Profil denemesi 1 uzun kenar kumpas ölçümü.	68
Şekil B.13: Profil denemesi 1 kamera uzaklığı.	69
Şekil B.14: Profil denemesi 1 kamera ölçümü.	69
Şekil B.15: Profil denemesi 2 uzun kenar kumpas ölçümü.	70
Şekil B.16: Profil denemesi 2 kısa kenar kumpas ölçümü.	70
Şekil B.17: Profil denemesi 2 kamera ölçümü.	71

TABLO LİSTESİ

Sayfa

Tablo 8. 1: Fabrika ortamında kamera ölçümünün gerçek zaman ölçümü ile karşılaştırma tablosu.....	46
--	----



SEMBOL LİSTESİ

OpenCV	:	Open Source Computer Vision
İKA	:	İnsansız Kara Aracı
2D	:	2 Boyutlu
Hz	:	Hertz
Numpy	:	Numerical Python
PIL	:	Python Imaging Library
exe	:	executable
vb	:	ve benzeri
CNN	:	Concolutional Neural Network
EDS	:	Elektronik Denetim Sistemi
HSV	:	Hue, Saturation, and Value
RGB	:	Red, Green, Blue
ve diğ.	:	ve diğer
YOLO	:	You Only Look Once
AI	:	Artificial Intelligence
R-CNN	:	Regions with Convolutional Neural Networks
ei	:	Eklemeli İmalat
mp	:	Megapiksel
CMYK	:	Cyan, Magenta, Yellow, Key (black)
LORA	:	Long Range
GET	:	Hypertext Transfer Protocol Get Request
PHP	:	Hypertext Preprocessor
MySQL	:	My Structured Query Language
Vcc	:	Voltage at The Common Collector
Gnd	:	Ground
MacOS	:	Mac Operation system
cm	:	Santimetre
GUI	:	Graphical User Interface

ÖNSÖZ

Yüksek lisans programıma başladığım günden itibaren bana yol gösterip yanımda olan, ders sürecimde , yüksek lisans tezimde gerekli olan teknik altyapıyı sağlamamda yardımcı olan tez izleme komitesi jüri üyelerine; bugüne kadar bana hem maddi hem manevi anlamda her türlü yardımı ve desteği hiç esirgemedi gösteren, bugün ki mühendis kişiliğimi kazanmamda, güzel ahlaklı, sevgi dolu ve azim ile hiçbir şeyden korkmadan başarmamı sağlayan, şahsıma büyük emekleri olan başta annem Eylem BAĞRIYANIK ve babam Murat BAĞRIYANIK olmak üzere tüm aileme, projede bana bu fırsatı tanıyan danışman hocam Doktor Öğretim Üyesi Erdem DİLMEN ‘e, ülkeyi bize emanet eden başöğretmenimiz Ulu Önder MUSTAFA KEMAL ATATÜRK’ e ve emeği geçen tüm ecdadımıza, bana büyük veya küçük yardımı dokunan herkese teşekkür ederim.

**Sevgili Annem Eylem BAĞRIYANIK ve Babam Murat
BAĞRIYANIK ‘a...**

1. GİRİŞ ve AMAÇ

Günümüze kadar ve halen hızla gelişen teknoloji, görüntü işleme ve yapay zeka alanlarında önemli ilerlemelere yol açmıştır. Bu ilerlemeler, çeşitli endüstriyel ve bilimsel uygulamalarda kullanılabilecek yenilikçi çözümler sunmaktadır. Özellikle çap ölçümü ve şekil tanıma, birçok sektörde otomasyon, kalite kontrol ve veri analizi gibi önemli süreçlerde temel rol oynamaktadır.

Çap ölçümü, boru ve profillerin boyutlarının hassas bir şekilde belirlenmesini sağlayan bir tekniktir. Bu ölçüm yöntemi, üretim hatlarında boru ve profillerin mühendislik projelerinde, kalite kontrolünde, yapı elemanlarının doğruluğunu sağlamak için sıklıkla kullanılmaktadır. Boru ve profil ölçümünün doğruluğu ve güvenilirliği, genel sistem performansını ve ürün kalitesini doğrudan etkilemektedir. Dolayısıyla, bu alanda geliştirilecek yenilikçi ölçüm teknikleri, endüstride önemli iyileştirmelere yol açabilir.

Günümüzde Python ile yazılan yazılımlar, mekanik aparatlar ile sınırlandırılarak ölçüm işlemi yapabilmektedir. Aynı zamanda, İnsansız Kara Aracı (İKA)'lar da bu işlemleri Python sayesinde lazer tarama sensörü kullanılarak ortamı 2D olarak modellemiş mesafe ve haritalamalarını yapmıştır. Nesneleri sensörler ile algılamış ve 10 Hz tarama hızında saniyede maksimum 8000 örnekleme hızına çıktığına tanık olunmuştur. Bu da nesneleri algılama aralığını maksimum 12 metre olarak ölçülendirilmiştir (Topal ve Yiğit, 2021).

Bazı sistemlerde ise derinlik kameraları kullanılmaktadır. Bu kameralar, görüntüleme açısına giren nesneler ile kamera arasındaki mesafeyi ölçerek piksel - derinlik bilgisi üreten makinelerdir. Intel D435i gibi kameralar eş zamanlı olarak derinlik bilgisi üretebilen kameralara örnektir. Günümüzde de bu derinlik kameraları robotik ve üretim izleme gibi alanlarda kullanılmaktadır.

Bu tez çalışmasının amacı, Python programlama dili ve OpenCV kütüphanesi kullanılarak boru ve profil çap ölçümünde yenilikçi bir yaklaşım geliştirmektir.

Arayüz olarak Tkinter kullanılarak geliştirilen sistem, OpenCV (görüntü işleme), NumPy (sayısal işlemler), Tkinter (GUI oluşturma) ve PIL (görüntü işleme)

kütüphanelerinden yararlanmaktadır. Program, bilinen bir cismin piksel ve çap değerine dayalı olarak ölçek faktörünü hesaplayıp milimetre başına piksel oranını verir.

Ölçek faktörü, bir görüntüdeki nesnenin fiziksel boyutlarını hesaplamak için kullanılır. Bilinen bir çap ve bu çapın piksel cinsinden değeri kullanılarak ölçek faktörü hesaplanır. Bu ölçek faktörü, görüntüdeki ölçümlerden gerçek dünya boyutlarını hesaplamak için kullanılır. Formül şu şekildedir.

$$\frac{\text{Bilinen çap (mm)}}{\text{Bilinen çap (piksel)}} \quad (1.1)$$

Bilinen Çap (mm): Bu, gerçek dünya boyutlarında bilinen bir nesnenin fiziksel çapıdır. Örneğin, 50 mm.

Bilinen Çap (piksel): Bu, aynı nesnenin bir görüntüdeki piksel cinsinden çapıdır. Örneğin, 100 piksel.

Bu değerler ile ölçek faktörü bulunmaktadır. Ölçek faktörü, görüntüdeki herhangi bir mesafeyi gerçek dünya birimlerine dönüştürmek için kullanılır. Örneğin, bir nesnenin çapı görüntüde 200 piksel olarak ölçülmüşse, bu nesnenin gerçek dünya boyutundaki çapı şu şekilde hesaplanır:

$$\text{Gerçek dünya çapı (mm)} = \text{Çap (piksel)} \times \text{Ölçekfaktörü} \left(\frac{\text{mm}}{\text{piksel}} \right) \quad (1.2)$$

$$\text{Gerçek dünya çapı (mm)} = 200 (\text{piksel}) \times 0.5 \left(\frac{\text{mm}}{\text{piksel}} \right) = 100$$

Tkinter arayüzünde boru ve profil için ayrı seçenekler mevcut olup, bu seçenekler doğrultusunda daire ya da dikdörtgen/kare çerçeve içine alınarak hesaplama yapılmaktadır. Kamera uzaklığının kalibrasyonu, çapı bilinen bir cisim alınarak gerçekleştirilir ve kamera uzaklığı ayarlanır. Farklı geometrik şekiller için kalibrasyon uzaklıkları, cismin çapı ve boyutuna göre değişkenlik gösterebilmektedir. Kamera ayarlandıktan sonra arayüzde oluşturulmak istenen boru veya profil çapı girilir. Tolerans değer aralıkları ve gri renk tonu filtreleme ayarları belirlenir. Gün ışığına bağlı olarak 36-139 low-up ayarlarının ideal olduğu görülmüştür. Algoritma başlatılarak kamera ile çap ölçüm işlemi gerçekleştirilir.

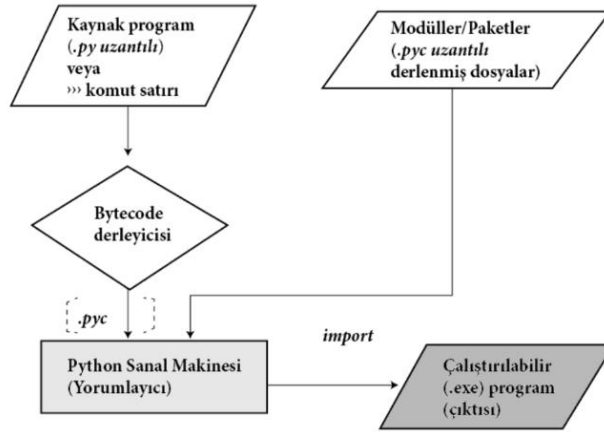
Bu tez çalışması, yüksek doğrulukta ve verimlilikte çap ölçümü sağlayacak yenilikçi bir çözüm sunmayı hedeflemektedir. Bu kapsamda, Python programlama dili ve ilgili kütüphaneler kullanılarak geliştirilen algoritmanın, endüstriyel uygulamalarda kullanım potansiyeli değerlendirilmiştir. Bu çalışma, literatüre ve endüstriye önemli katkılar sağlamayı amaçlamaktadır.



2. LİTERATÜR TARAMASI

2.1 Python Programlama Dili ve Uygulama Alanları

90'lı yılların başında geliştirilmeye açık olan Python dili, Hollandalı Guido van Rossum tarafından geliştirilmeye başlanmıştır. Guido van Rossum bu dili geliştirirken ismini, İngiliz bir komedi grubu olan The Mounth Python'dan almıştır (Python Kılavuzu 2013). Pek çok alanda kullanım imkanı sağlamıştır. Bunlardan birkaçı, robotik programlama, oyun programlama, yapay zeka programlama, veri analizi gibi alanlardır. Bu uygulamalar geliştirilebilirdir (Sevli 2021). Python iki aşamalı bir şekilde yorumlayıcı olarak çalışır. Python bytecode çözümleyicisi, bytecode çalıştıran sanal çözümleyici kullanarak kod revizyonu gerçekleştirebilir. Bu işlem Şekil 2.1'de gösterildiği gibi açıklanmıştır (Çobanoğlu, 2021).



Şekil 2.1: Python kodunun exe koduna dönüşümü.

Girintilere dayalı bir söz dizimi olan Python bu sayede diğer yazılım dillerine oranla daha akılda kalan bir yazılım dilidir. Modüler yapısı sayesinde birçok platformda kullanılabilir olmasını sağlamaktadır. Python yazılım dili, sistemlerin programlanmasından ağ üzerinden gerçekleştirilen çalışmalara, web tabanlı uygulamalardan veri tabanları içeren uygulamalara kadar geniş bir yelpazede yazılım geliştirme imkanı sağlamaktadır. Python yazılım dili, ABC yazılım diline alternatif olarak ortaya çıkmıştır.

Python, anlaşılması ve veri işlemesi daha kolay bir dil olduğundan dolayı kullanımda daha fazla tercih edilen bir programlama dilidir. Python açık kaynak kod kütüphanesi ücretsiz olarak kullanılabilir. Veri incelemeleri ve yapay zeka tasarımları gibi alanlarda kullanılması bakımından oldukça tercih edilen bir yazılımdır. Makine öğrenmesi, derin öğrenmesi gibi konularda araştırmacılara büyük bir şekilde kolaylık sağlamaktadır.

Robotik uygulamalarda en çok tercih edilen programlama dillerinden biri Python'dır. Açık kaynak kodlu olması, bunun başlıca sebeplerindendir. Oyunlarda ise genel olarak 2D oyun tasarımında ön plana çıkmaktadır. Python dilinde geliştirilen oyunlarda fonksiyonlar eklenebilir ve alışılmışın dışında fantastik oyunlar geliştirilebilir. Python'un belirli ağ protokolleri bunun yanında bilindik olan Youtube, Google, Spotify gibi uygulamalarda kullanılmıştır. Python, makine öğrenmede sıklıkla kullanılan bir programdır. Bunun sebebi, açık kaynak kodu bolluğunun yanında makine öğrenmesinin büyük veriler dizini ile çalışmasıdır (İlgın, 2023).

2.2 Görüntü İşlemenin Temelleri ve Kullanım Alanları

Görüntü işleme ilk olarak, Londra ve New York arasındaki görsellerin bir deniz altı aracılığı ile gönderilmesiyle gazetelerde kullanılmıştır. 1920'li yıllarda oluşturulan Bartlane hattında, Atlantik'ten bir resmin transfer süreci bir hafta alıyor iken, bu süreç 3 saat gibi bir süreye indirgenmiştir (Duman, 2019).

Değerlendirme ve kod yürütme işlemleri yapıldıktan sonra, başka bir sistemde okunabilir bir şekle dönüştürülmesi ya da bir elektronik sistemden başka bir elektronik sisteme aktarılması işlemine görüntü işleme denir (Gonzalez ve Woods, 2005). Farklı içeriklerden oluşan görüntüler, belirli bir hedefe yönelmiş halde olan görüntü kaynağından alınmış farklı farklı içeriklerden oluşur. Görüntülerin bilgisayar ile işlenmesiyle görüntü içeriği ayrıntılı bir şekilde ele alınabilir. Bu şekilde görüntü işleme gerçekleştirilmiş olur (Jahne, 2005). Günümüzde bilgisayarların yaygın olarak kullanılmasıyla birlikte görüntü işlemeden yaygın olarak istifade edilmeye başlanılmıştır. Görüntü işleme, çevre ile bir etkileşim olmadan belirli araçlar yardımı ile nesnelerin özellikleri hakkında bilgi edinmemizi sağlamaktadır. Bundan dolayı nesneler ile ilgili bilgi çıkarımı yapılabilmektedir (Perihanoğlu, 2015).

Görüntü işleme, bilgisayarda belirli algoritmalar kullanılarak ve geliştirilerek dijital olan görüntülerin değiştirilmesiyle ilgilenen uygulama ve çalışma alanlarını kapsar. Görüntü işlemenin amacı ise dijital olan görüntüleri daha kaliteli hale getirmektir. Yararlı bilgiler ile çıkarımlar yapmak, daha ileri analiz ve yorumlamaya uygun bir biçime getirmektir.

2.3 Görüntü İşleme ile Yapılan Uygulamalar

Yeni ortaya çıkan birçok teknoloji uygulamalarında yüz algılama algoritmaları, eğitilmiş olan derin sinir ağları ile modellerin yüz hatlarını bulmaya çalışır, bireylerin yüz hatları ile birlikte ortaya çıkan çıkarımlar yapabilmektedir. Bu işlemler için en başta gelen kütüphaneler arasında CNN ve ImageNet yer almaktadır. CNN kütüphanesi evrimsel (convolutional) sinir ağları olarak da bilinmektedir. Bu ağ, yapay sinir ağı tabanlıdır ve özelleştirilmiş bir derin öğrenme tasarımını barındırır. Günümüzde bazı teknolojik aletlerde, telefon, tablet, bilgisayar kameralarının odaklanmasında, güvenlik için kullanılan radarlarda, EDS'lerde kullanılmaktadır. OpenCV içerisinde bulunan ve nesnelerin tanındığı Haar-Cascade algoritması da görüntü işleme uygulamalarında kullanılmaktadır. Haar-Cascade algoritması ile yüz tespiti yapılabilir, görüntü içerisindeki yüzler tespit edilebilir. Buna ek olarak, matplotlib ve seaborn kütüphaneleri, iki boyutlu verileri görselleştirmek için oldukça sık kullanılmaktadır.

Sinir ağlarının kullanılabilmesi için oldukça fazla veriye ihtiyaç duyulmaktadır. Kişilerin yüzlerinin tanıtılması oldukça büyük bir sinir ağının eğitilmesinden geçmektedir. Bu eğitim genellikle CNN kullanılarak yapılmaktadır. Bu işlemde, yüzlerin baskın olan özellikleri belirlenerek bir sınıflandırma gerçekleştirilir. Bu nitelikler eğitim ile verildiği için çabuk bir şekilde yüzün hangi sınıfa ait olduğu belirlenmektedir. Kişilerin yüzleri veri tabanlarına eklendiklerinden dolayı kişilerin yüzleri ayrı bir sınıf olmuştur, girdiler ise geriye kalanlardır ve yüzlerin sınıflarını belirler (İlgın, 2020).

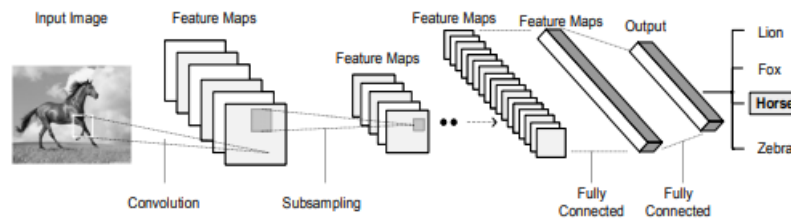
Video nesne algılamaları hareketli bir cisimin koordinatlarını kaydederek video aracılığı ile nesneyi çerçevelemektir. Nesneleri algılamak için sinir ağları eğitilebilir ya da eğitilmiş olan modeller kullanılabilir. Nesne tanımlama modelleri

etiketlenmiş olan bilgiler ile yapılabilir. Bir nesnenin hareket halinde olması nesnenin her bir karede farklı konumlarda olması anlamına gelmektedir (Sharma ve diğ., 2021).

Görüntülerde kullanılan HSV renklerin tanınması için kullanılan bir renk segmentasyonudur. HSV, RGB'den türetilmiş olan bir renk modelidir. HSV renginin elde edilmesi RGB aracılığı ile sağlanmaktadır. HSV, tonlardaki doygunluk anlamına gelmektedir. HSV'nin bir bileşeni olan Hue, renk dalga boyunun bir ölçüsüdür, 0 ile 255 arasında değişmektedir. Yine HSV'nin bir bileşeni olan saturation (doygunluk) ise son renge bağlı olarak renkteki parlaklığı artırmaya yönelik bir işleve sahiptir. Value (değer), bir rengin parlaklığını ölçmektedir. Örneğin, rengin boyutu %100 ise çok parlak bir şekilde görünecektir. %0 olur ise renk karanlık olarak görünür (Pulungan ve diğ., 2021).

Nesne tespiti, bir nesnenin ait olduğu sınıfın belirlenmesi ve etrafındaki sınırlayıcı kutunun çıktısını alarak nesnenin konumunun tahmin edilmesidir. Görüntüden alınan sınıfa göre tek sınıflı nesne tespiti yapılırken, görüntüdeki tüm nesnelerin sınıflarının tespit edilmesi çok sınıflı nesne algılama olarak adlandırılır. Kısmi veya tam tıkanma gibi çeşitli zorluklar ve değişen aydınlatma koşulları, pozlar, ölçekler gibi faktörler nesne tespiti sırasında dikkate alınmalıdır.

CNN'ler nesne tespitlerinde yaygın olarak kullanılmaktadırlar. CNN ileri beslemeli bir sinir ağıdır. Bir fonksiyonun diğeri ile nasıl olarak kesiştiğini gösteren bir entegrasyondur, herhangi iki fonksiyonun harmanlanmasıdır (Pathak ve diğ., 2018).



Şekil 2.2: Nesne için CNN'nin katmanlı mimarisini göstermektedir.

Mekansal karmaşıklığı azaltmak için görüntü, aktivasyon kodları ile evrilebilir, aynı zamanda soyutlanmış özellik haritaları havuz katmanları ile işlenebilir ve buna göre özellik haritaları oluşturulabilir. Ağın karmaşıklığını gidermek için CNN'e ait havuz katmanları aşağıdaki şekildeki gibi gösterilebilir.

Pooling layer	Description
Max pooling	It is widely used pooling in CNNs. It takes maximum value from the selected image patch and place in the matrix storing the maximum values from other image patches.
Average pooling	This pooling averages the neighborhood pixels.
Deformation pooling [40]	Deformable pooling has ability to extract deformable properties, geometric constraints of the objects.
Spatial pyramid pooling [53]	This pooling performs down-sampling of the image and produces feature vector with a fixed length. This feature vector can be used for object detection without making any deformations on the original image. This pooling is robust to object deformations.
Scale dependent pooling [54]	This pooling handles scale variation in object detection and helps to improve the accuracy of detection.

Şekil 2.3: Nesne tespiti için kullanılan havuzlama katmanları (Pathak ve diğ. 2018).

Nesne tespiti, gözetim (surveillance), savunma ve güvenlik alanlarında kullanılabilir. Görsel veriler ile coğrafi veriler harmanlanır ve mevcutta bulunan durumlar hakkında bilgi verir. Bu süreç genellikle insan, araç ve şüpheli olduğu görülen cisimleri takip ve tespit etmek için kullanılmaktadır. Hayatımızın her alanında işimizi kolaylaştırmak için tasarlanan görüntü işleme yazılımları otonom sürüş, sahipsiz bagaj uygulaması, hastalıklı hayvan ve bitki tespitlerinde vb. gibi birçok alanda kullanılmaktadır (Pathak ve diğ., 2018). Nesne algılama, video ve görüntülerdeki nesne vb. şeyleri tanımlamak için kullanılan bir bilgisayar görme tekniğidir. Nesne algılama spesifik olarak bir cisimi sınırlandırarak, belirli olan bir sahnede bulmamıza yardımcı olmaktadır. Nesne algılama, diğer bilgisayar görme tekniklerine bağlıdır. Görüntü veya videolardaki sahneleri anlamaya ve analiz etmeye yarar. Görüntü segmentasyonu, bir sahnedeki öğeleri pikseller ile işleyerek yapılırken, görüntü tanıma ise bir sahnedeki tanımlanmış bir nesne için etiket üretir.

Nesne algılama belirli olan bir resim veya videodaki nesnenin konumu ve türünü, bunu algılamak için gerekli olan hızı sağlanmasında yardımcı olmaktadır. Yaşantımızda bu tür uygulamalar konfor ve rahatlık bakımından önem arz etmektedir. Bunlardan bazıları şu şekilde sıralanabilir. Kendi kendine giden arabalar (otonom sürüş). Bu tür bir otonom sürüşün güvenli hale gelebilmesi nesnelerin tanınmasından geçmektedir. Otonom sürüşlerde nesnelerin tanınması, konumu ve aracın belirli sınırlar doğrultusunda gitmesi önemlidir. Bunu da nesne algılama amaçlı kullanılan ImageAI kütüphanesi ile yapabilmektedir. Tarım uygulamalarında da nesne algılama yöntemi kullanılabilir. Nesne algılama, tarımda insan gözü ile tespit edilemeyecek olan tehditleri tespit edilmesinde görev almaktadır. Tıp alanında da oldukça yaygın olarak kullanılmaktadır. Yüz tanıma, sivilce tespitlerinde kullanılabilir. Nesne algılama, deri hastalıklarının tespit edilmesinde kullanılmaktadır. Bunun önemi, anında tespit ve iyileştirme yöntemlerini gecikmeden uygulayabilmektir.

Nesne algılama yöntemleri genel olarak iki alt yöntem içerirler. İlki, görüntüyü kod yardımı ile girdi olarak alır. İkincisi ise bu görüntüyü tanımlama ve konumlandırma ile kullanır. Nesne algılama kodlayıcının çıktısını alır, her bir öge için sınırlayıcılar çizerek etiketler belirtir.

Yapay zeka tabanlı olan ImageAI kütüphanesi, derin öğrenme amaçlı kolay kullanıma sahip olmasından ötürü tercih edilmektedir. ImageAI, nesne video ve görüntülerdeki nesneleri algılama ve tespit için kullanılabilir. ImageAI, YOLO ve RetinaNet gibi şablonları destekler ve bu sayede yaklaşık olarak 80 farklı şekli algılayabilir (Asaad ve diğ., 2023).

```
from imageai.Detection import ObjectDetection

import os

execution_path = os.getcwd()

detector = ObjectDetection()

detector.setModelTypeAsRetinaNet()

detector.setModelPath(os.path.join(execution_path ,
"resnet50_coco_best_v2.0.1.h5"))

detector.loadModel()

detections = detector.detectObjectsFromImage(input_image=os.path.join(execution_path,
"Dog&Cat.jpg"),
output_image_path=os.path.join(execution_path , "imagenew.jpg"))

for eachObject in detections:

    print(eachObject["name"] , " : " , eachObject["percentage_probability"] )
```

Şekil 2.4: Colab ortamında bir nesneyi tanıma kodları.

Yukarıdaki kodun ilk satırlarında ImageAI, bilgisayarla görme AI uygulama kütüphanesinden ObjectDetction sınıfı içe aktarılmıştır. İkinci satırda os sınıfı Python'a aktarılmıştır. 4. satırda Python dosyasının yolunu tanımlamak için yürütme yolu değişkeni tanımlanmıştır. Sonrasında, Colab platformunda bulunan RetinaNet modeli ve gösterilecek görüntü eklenmiştir. 6. satırda ObjectDetction sınıfını temsil eden bir dedektör değişkeni tanımlanmıştır. Bir sonraki satırda yüklenen formun yolu

tanımlanmıştır. 8. satırda form programa yüklenmiştir. Daha sonra sınıfın en önemli fonksiyonu olan DetectObjects FromImage' a gelinmiştir. Bu fonksiyon sayesinde, içindeki nesneler tespit edilebilmektedir (Asaad ve diğ., 2023).



Şekil 2.5: Kod çıktısı sonuçları

Görüntü işlemeye bir örnek de bal peteği üzerindeki arı larvasının konum ve özellik tespitini yapabilen bir uygulamadır (Güngörmüş, 2020). Bu sayede bal arılarından bal haricinde birçok şekilde yararlanılabilmektedir. Besin değerleri bakımından en değerli olan arı sütüdür. Larva işlemi ise insanlar tarafından yapılan üretimde çok fazla zaman alan bir işlemdir. Bu işlemde zamanı azaltmak için insan faktörünün ve işçiliğinin ortadan kaldırılması gerekmektedir. Bunun için bir deney düzeneği kurulmuştur. Düzenekte 60 adet petek ile fotoğraflar çekilmiştir. 40 adet fotoğraf eğitim için, 20 adet fotoğraf ise test için olacak şekilde ayrılmıştır. faster R-CNN modeli kullanılmıştır. Larvaların piksel ile büyüklüğüne bakılarak arı sütü için ideal olup olmadığı tespit edilmeye çalışılmıştır. Sistemde OpenCV kütüphanesi kullanılmıştır.

Başka bir görüntü işleme uygulamasında titanyum implant malzemelerdeki korozyonu inceleyen bir görüntü işleme ele alınmıştır. Taramalı elektron mikroskobu yardımı ile sayısal görüntü işleme tekniği kullanılarak yapılan bir uygulamadır (Uğurlu, 2022).

Görüntü işleme ile yapılan diğer bir uygulama da imalat sektöründe kullanılmıştır (Duman, 2019). El ile üretimi yapılan parçalarda kalite açısından yeterince iyi olmaması nedeni ile görüntü işleme kullanılarak yüksek maliyetli uygulamaların az hata ve yüksek performans ile üretilmesi hedeflenmiştir. Eklemeli imalat ile üretilmiş parçaların yeterince kaliteli olmaması göze çarpmış ve yüksek maliyetli uygulamalarda bu tür bir imalattan kaçınılmıştır. Bunun sebebi ergime oluşumunda, toz sermayede meydana gelen hataların tespitinde zorluk çıkarmasıdır. Bu zorluğun önüne görüntü işleme uygulamaları ile geçilmiştir.

Görüntü işleme ile yapılan başka bir uygulama ise kalite kontrol amacıyla yapılmıştır (Tarı, 2020). Enjektör üretiminde, kalite kontrolünü sağlamak amacıyla sistem entegre edilmiştir. Uygulamanın amacı hız artırımıdır. Küçük, insan gözüyle görülemeyen kusurların, saydam olan enjektör malzemeden yapılmış olması da göz önüne alınır ise bu hata oranlarının ciddi şekilde azaltılması sağlanmıştır. Bu süreçte üretimin tamamen makine bazında gerçekleştirilmesi amaçlanmıştır. Çalışmada aydınlatma sistemi kullanılmış olup bir platform üzerine entegre edilmiştir. 12.2 MP endüstriyel amaçlı olarak kullanılan kameralardan yararlanılmıştır. Şablon eşleştirme, haar-cascade sınıflandırıcısı ve renk filtreleme yöntemi kullanarak uygulamaya entegre edilmiştir. Kullanılan sistem, amacına uygun şekilde çalışmış olup %85'in üzerinde bir başarı elde edilmiştir.

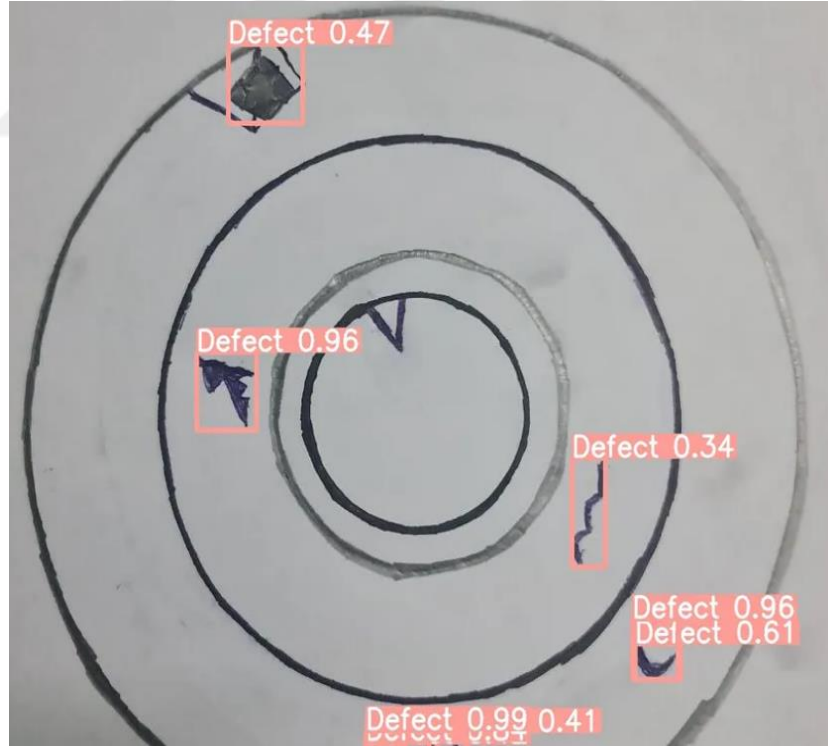
Görüntü işleme kendi içinde teknikleri bakımından ayrılmaktadır. Bunlardan bir tanesi sayısal görüntü işlemedir. Sayısal görüntü işleme uygulamaları da sıkça kullanılmaktadır. Bu uygulamalardan bir tanesi iki boyutlu görüntü üzerinden yapılan üç boyutlu manipülasyon analizidir. Bu teknikte en yaygın olarak görülen kopyala-taşı tekniğidir. Bu uygulamanın kolay olması nedeniyle görüntüdeki bazı bölgelerin kopyalanıp taşınması sağlanmıştır. Kopyalanmış öge ile diğer resim aralarında fark olmadan işlem gerçekleştirilmiştir. Kopyalanan bölge ile diğer bölge aynı olduğundan dolayı ise dinamik aralığı ve resmin geri kalanı resim ile uyumlu kalır.

Şekil 2.6'da beraber çekilen iki kadın görüntüsü başka bir görüntü üzerine alınmış ve yapıştırılmıştır. Kadınların olmadığı manipüle edilmiş bir sistem elde edilmiştir (Atılğan, 2017).



Şekil 2.6: Solda bulunan orijinal görüntü üzerinde kopyalama-taşıma işlemi yapılması sonucunda sağda bulunan manipüle edilmiş görüntü elde edilmiştir (Mushtaq ve Mir 2014).

Başka bir görüntü işleme uygulaması ise OpenCV ve YOLO kullanılarak eş merkezli dairelerde kenar kusuru tespiti için kullanılmıştır. Bilgisayar görme uygulamalarında nesnelerin algılanması ve analizi olması gereken temel bir görevdir. Uygulamanın amacı kalite kontrol uygulamasını iyileştirmek ve güçlendirmektir (Jaykumaran, 2023).

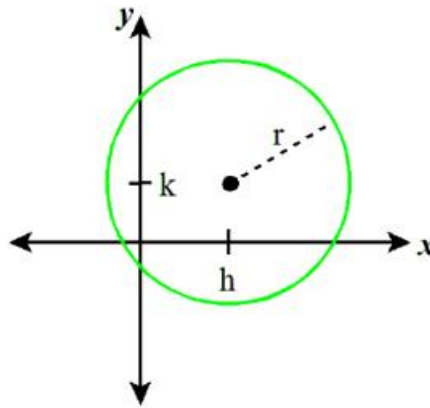


Şekil 2.7: Kusur tespiti program çıktısı (Jaykumaran 2023).

Programda daire tespitleri OpenCV kütüphaneleri aracılığı ile yapılmıştır. Şekil 2.7’de canny kodu ile kenar tespiti yapılmıştır, kenar tespiti kenarların belirlenmesinde

ve konturları bulmak için kullanılmıştır. Bulunan konturların daireleri temsil ettikleri varsayılarak tespit edilen konturlara elipsler yerleştirilmektedir. Son olarak daireler filtrelendir ve buna benzer olan dairelerin ortalama çapları hesaplanır. YOLO ile de videolardaki nesneleri yüksek doğrulukla algılamak mümkündür (Jaykumaran, 2023).

OpenCV kullanılarak daire tespit eden uygulamalarda belirli formüller kullanılmaktadır. Daire tespitinin iris ve beyaz kan hücresi gibi çeşitli segmentasyonlarda kullanım olanağı vardır. Bir daire aşağıdaki (2.1) ile tanımlanabilir hale gelir.



Şekil 2.8: Daire tespiti denklemi.

$$(x - h)^2 + (y - k)^2 = r^2 \quad (2.1)$$

Denklem (2.1)'de h , k ve r olarak 3 parametre bulunması gerekmektedir, bu matrisleri bulmak için biriktirici matris denilen 3 boyutlu bir matris kullanılmaktadır, a (merkezin x koordinatı) 1'den satırlara kadar, b (merkezin y koordinatı) 1'den sütunlara kadar olan bölgedir, ve r 1'den $\text{maxRadius} = a$ kadar değişebilir.

$$\sqrt{\text{rows}^2 + \text{cols}^2} \quad (2.2)$$

Denklem (2.2)'de akümülatör matrisinin başlatılması için $\text{rows} * \text{cols} * \text{maxRadius}$ boyutlarında bir matris, sıfırlar ile başlatılmıştır. Görüntünün ön işlenmesinde ise gri tonlama ve kenar dedektörü kullanılmıştır, daireler koyu hale

getirilmiş ve bu sayede kenarların görünmesi sağlanmıştır. Sonrasında r sabitlenmiş ve a ve b'nin değerini değiştirerek çift iç içe döngü oluşturulmuştur.

```
for a in range(rows):
    for b in range(cols):
        r = math.sqrt((xi - a)**2 + (yi - b)**2)
        accum_matrix[a][b][r] += 1
```

Şekil 2.9: Program kodları.

Python programlama dilinde gri renk tonlaması renk uzaylarının filtrelemesi ile oluşturulmaktadır. Bir görüntüyü CMYK, HSV, RGB, CMYK gibi renk uzaylarından gri tonlamaya dönüştürme işlemine denir. Gri tonlamada görüntüler tek boyutludur. Ek olarak model karışıklığını azaltmaya yardımcı olur, birçok algoritma sadece gri tonlama görüntüler üzerinden çalışılabilir şekilde özelleştirilmiştir. Kenar algılama işlemi de yalnızca gri tonlamalı görüntüler üzerinden çalışmaktadır. Renkli olan bir görüntü farklı şekillerde gri tonlamaya dönüştürülebilir, bunlardan birincisi, `cv2.cvtColor()` kodu ile yapılmaktadır ilk olarak orijinal görüntü `imread()` kodu ile okunur ardından `cv2.CvtColor()` ile gri tonlamaya dönüştürülebilir.

```
# import opencv
import cv2

# Load the input image
image = cv2.imread('C:\\Documents\\full_path\\tomatoes.jpg')
cv2.imshow('Original', image)
cv2.waitKey(0)

# Use the cvtColor() function to grayscale the image
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

cv2.imshow('Grayscale', gray_image)
cv2.waitKey(0)

# Window shown waits for any key pressing event
cv2.destroyAllWindows()
```

Şekil 2.10: Gri tonlama kodu örneği.



Şekil 2.11: Giriş resmi.



Şekil 2.12: Çıkış resmi.

Python programlama dili ile video oluşturulabilir. Bu işlem için ``cv.VideoCapture()`` kodu video yakalamak için kullanılan bir koddur. ``Cv2.imshow()`` kodu ile videodaki kareleri göstermek için kullanılır. Bu kodlar ile birlikte kamera açılabilir ve görüntü alma imkanı sağlanabilmektedir.

Görüntü işleme teknikleri, endüstriyel ve akademik araştırmalarda nesnelerin boyutlarının belirlenmesi için yaygın olarak kullanılmaktadır. Örneğin, "Real-time detection of object dimensions using computer vision techniques" adlı (Lu ve Sun, 2012) çalışmasında, endüstriyel boru sistemlerinde boruların çapının ölçülmesi için gerçek zamanlı görüntü işleme teknikleri kullanmıştır. Kamera görüntüleri, boruların çapını belirlemek amacıyla işlenmiş ve çeşitli görüntü işleme algoritmaları uygulanmıştır. Çalışmanın bulguları, bilgisayarla görmenin endüstriyel uygulamalarda doğruluğunu ve etkinliğini göstermektedir (Dwyer, 2009).

Bilgisayarla görme sistemleri, endüstriyel ölçümler ve kalite kontrol için güçlü bir araçtır. Aşağıda, bu sistemlerin kullanıldığı bazı çalışmalar bulunmaktadır. "Computer Vision Techniques in Manufacturing" adlı (Zhang ve diğ., 2023) çalışmasında, üretim hattında ürünlerin boyutlarının belirlenmesi için bir bilgisayarla görme sistemi geliştirilmiştir. Kamera ile alınan görüntüler işlenmiş ve ürünlerin boyutları otomatik olarak ölçülmüştür. Bu çalışma, üretim süreçlerinde kalite kontrol ve doğruluk açısından bilgisayarla görme sistemlerinin önemini vurgulamaktadır.

Görüntü işleme ve nesne tanıma, bilgisayarla görme alanında önemlidir. Kontur algılama ve nesne tanıma teknikleri, nesnelerin boyutlarının belirlenmesinde kritik rol oynar. "Topological Structural Analysis of Digitized Binary Images by Border Following" adlı (Suzuki ve Abe, 1985) çalışmasında, dijitalleştirilmiş ikili görüntülerde sınır takibi yöntemiyle topolojik yapıların analizini sunar. Kontur algılama ve nesne tanıma algoritmalarının temelleri bu çalışmada ele alınmıştır.

Gerçek zamanlı video işleme sistemleri, kamera görüntülerinin anlık olarak işlenmesini ve analiz edilmesini sağlar. Kamera görüntüleri, üretim süreçlerinde kalite kontrol ve boyut ölçümleri için anlık olarak işlenmiştir (Huang ve diğ., 2018).

Alüminyum profil üretiminde kalite kontrol süreçlerinin etkinliği, bilim ve teknolojinin ilerlemesiyle birlikte önemli ölçüde artmıştır. Büyük boyutlu ve karmaşık yapılı, bükümlü profillerin ölçümleri, geleneksel manuel test yöntemleriyle artık endüstrinin ihtiyaçlarını karşılayamamaktadır. Görüntü işleme teknolojisi, hata payları ve hassasiyetlerin istenen aralıklarda tutulabilmesi durumunda, temaslı ölçümlere kıyasla daha üstün bir yöntem olarak değerlendirilmektedir. Bu bağlamda, alüminyum profillerin ölçümünde görüntü işleme teknolojilerinin kullanımı hem kalite kontrol süreçlerinin iyileştirilmesi hem de üretim verimliliğinin artırılması açısından büyük bir potansiyele sahiptir (Yoldaş ve Sungur, 2020).

Ekstrüzyon yöntemi ile üretilen pvc profillerin boyut ve şekilsel olarak kalite unsurunu etkileyecek birçok faktör bulunmaktadır. Bazıları enjeksiyon basıncı, kalıp sıcaklığı, çekici ve kesici hızı, soğutma zamanı, şeklindedir. Ekstrüzyon yöntemi ile üretilen kesitlerinin et kalınlığı ve conta boşlukları, üretilen profil kalitesini direkt olarak etkilemektedir. Bundan dolayı, üretilen profillerin tolerans değerinde olup

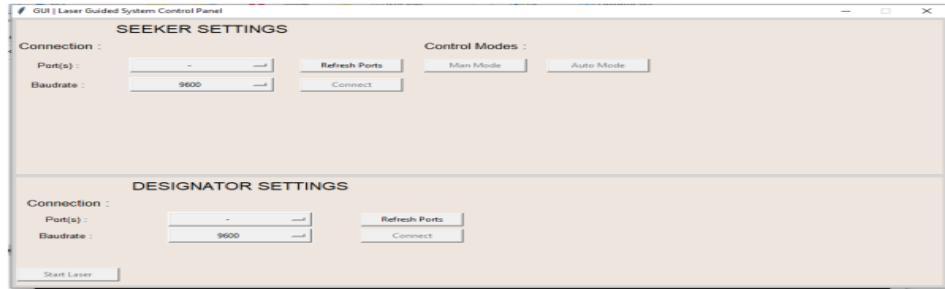
olmadığı bilinmek zorundadır. Bu tür sorunları ortadan kaldırmak için görüntü işleme tekniği ile algoritma oluşturulmuştur (Akdoğan, 2018).

2.4 Lazer Kullanılarak Yapılan Görüntü İşleme Uygulamaları

Bu altbölümde, lazer kullanılarak yapılan yazılımlardan bahsedilecek olup, ekran tasarımları gösterilip entegre edilen sistemlerden bahsedilecektir.

2.4.1 Lazer Işığını Takip Eden Araç Prototip Tasarımı

Lazer ışığını takip eden araç prototip tasarımını yapılan (Tasalı, 2023) tezinde, lazerden çıkan ışık ile birlikte işaretlenmiş bir nesneyi, nesneye kitlenerek takip eden lazer güdümlü bir araç prototip tasarımı gerçekleştirmiştir. Sistemde makine öğrenmesi ve görüntü işleme teknikleri kullanmıştır. Yüz tanıma algoritmaları ve sürücüsüz araç kullanımı gibi uygulamalar entegre etmiştir. Uzaktan kontrolü GUI Laser Guided System Control Panel isimli bir kullanıcı arayüzü tasarlamış ve bu, sisteme entegre edilip uzaktan kontrolü sağlamıştır.



Şekil 2.13: GUI| Laser Guided System Control Panel arayüzüne ait örnek görsel.

2.4.2 Görüntü İşleme Yöntemi ile Optik İşaret Tanıma ve Yönlendirme Sistemi

Görüntü işleme yöntemi ile optik işaret tanıma ve yönlendirme sistemi yapılan (Küçükkara, 2019) çalışmasında Python yazılım aracını kullanılmış ve kütüphane olarak ise OpenCV kütüphanesi kullanılmıştır. Optik form, üzerinde bulunan karmaşık

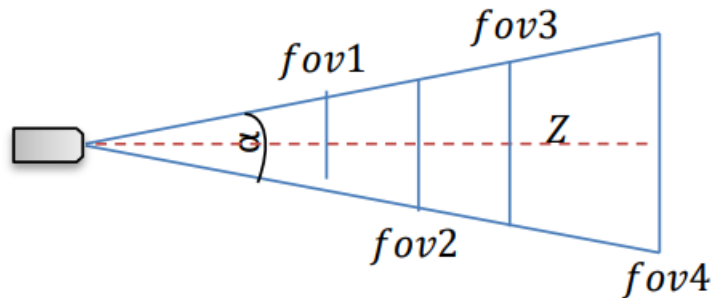
olarak konumlandırılmış ve yazılmış olan kitapçık türü, cevap anahtarı, öğrenci numarası vb. gibi alanlardan oluşmaktadır. Öğrenci bilgileri ve yapılan karekod, Python yazılım dilinde geliştirilen ayrıca bir optik işaret uygulaması yapılmıştır. Yazılımda karekod ile kağıdın düz mü yoksa ters mi olduğu saptanmış, sonrasında öğrenci bilgileri ve diğer bilgiler kontrol edilip sonuçları izlenmiştir. Sonuçlara göre %99,1 oranında başarılı bir sonuca varılmıştır.

2.5 Kamera ile Görüntü İşleme Kullanılarak Mesafe Ölçümü

Bu altbölümde, farklı tipteki kameralar göz önüne alınarak yapılan mesafe ölçüm sistemlerinden bahsedilecektir. Ek olarak, kameraların avantaj ve dezavantajlarıyla ortaya çıkan durumlardan da bahsedilecektir.

2.5.1 Mono Kameralar ile Mesafe Ölçümü

Günümüzde birçok araştırmada birçok bilgi gibi mesafe bilgisinin de görüntünün içinde saklı olduğunu kabul edilmiştir. Mesafe ölçümünün görüntü işleme ile farklı şekillerde yapılmasından ziyade uzman kişiler tarafından piksel odaklı olarak yapılmasına yönelinmiştir. Mono türlerindeki kameralarda yapılan çalışmalarda iki tür hata görülmüştür. Birincisi hedef belirtilmesi veya ölçülecek nesnenin boyutları hakkında bilgi sahibi olmanın gerekmesidir. Çünkü bu türdeki yöntemler, cisimin görüntünün içinde kapladığı alan hesabı üzerinden çıkarım yapmaktadır. Resimdeki görüntüde anlaşıldığı üzere, alanı arttıkça farklı türde hesaplamalar ve sonuçlar elde edilmiştir.



Şekil 2.14: Mono bir kameraya ait görüş alanının mesafe ile değişimi.

Şekil 2.5.1.1’de $-\alpha$ - kamera görüş açısını göstermektedir. Buna göre oluşturulan denklemde FoV değeri şu şekilde hesaplanmıştır:

$$F_0V = \tan a_2 \times z \times 2 \quad (2.3)$$

FoV değerindeki değişim hedefteki nesnenin kamera uzaklığına ve kameranın görüş açısına bağlı olarak değişiklik gösterebilir.

FoV değerinin artmasıyla birlikte, görüntüde metre sayısına düşen piksel sayısında azalma olacaktır.

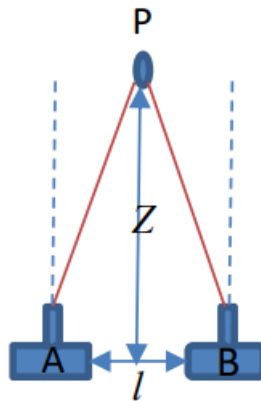
$$ppM = \frac{wp_x}{f_0v} \quad (2.4)$$

Bu formülde wp_x yatayda bulunan bir kameranın çözünürlüğünü göstermektedir. Kameranın yatay çözünürlüğü sabit varsayılır ise PPM’nin azalmasıyla birlikte piksel genişliği artacaktır ve bu durum aşağıdaki denklemde ifade edilmiştir (Yanık ve Turan, 2021).

$$p_x = \frac{1}{ppM} \quad (2.5)$$

2.5.2 Stereo Kameralar ile Mesafe Ölçümü

Stereo kameralarda, belirlenen iki farklı nokta ile görüntüdeki yer değiştirme biliniyorsa o cisime ait derinlik yaklaşımı yapılabilir. Bu durum üçgenleme metodu ile açıklanabilir.



Şekil 2.15: Üçgenleme metodu.

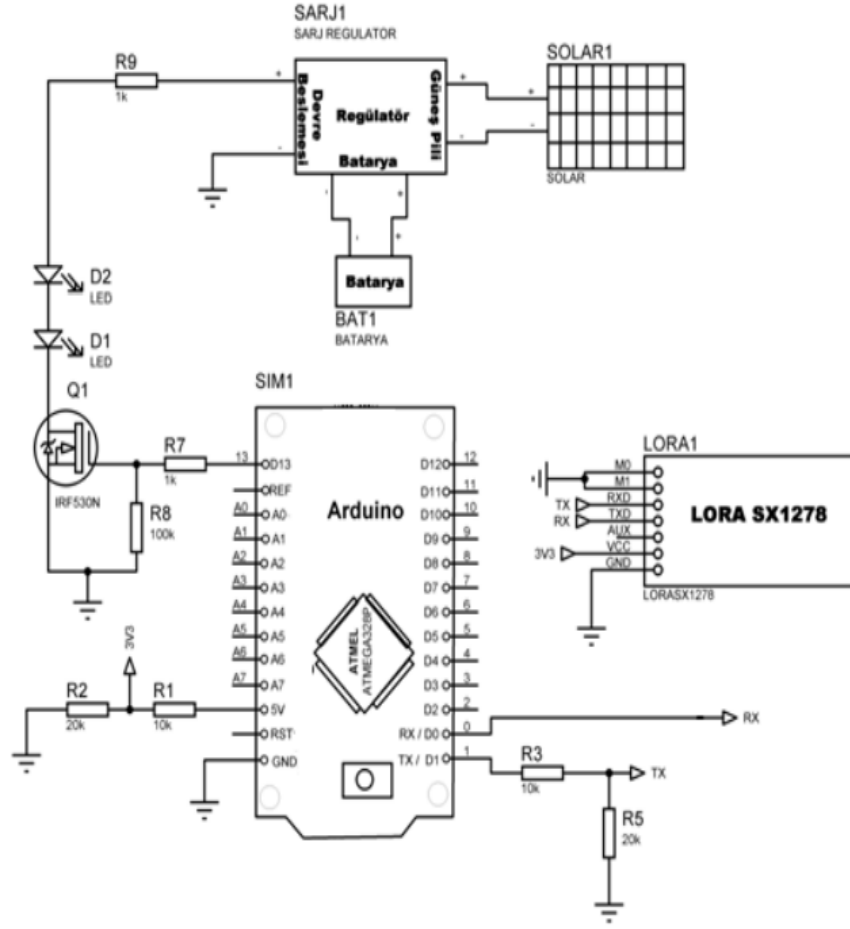
Bu yöntemde, kameralar arasındaki mesafe ve kamera çözünürlüğü ne kadar fazla ise başarı oranı o kadar fazla olacaktır.

A-B aynı özellikte kameraları, P hedef nesneyi ve 1 iki kamera arasındaki mesafeyi, Z ise hedefteki nesnenin kameralara uzaklığını göstermektedir (Yanık ve Turan, 2021).

2.6 Arduino Python Haberleşmesi ile Yapılan Projeler

Arduino ile güneş enerjili sera otomasyonu projesi yapan (Aygör, 2019) bu projenin donanım kısmını, internet kontrollü güneş enerjisi ile çalışan lamba, güneş enerjisi ile çalışan sensörler, nem sensörü, ışık sensörlerinin olduğu sensör parametreleri ve ölçüm sonuçlarının alındığı merkez istasyon arasındaki haberleşmeyi sağlayan bir devreden oluşturmuştur. Merkez istasyon devresi şu şekilde işlevleri gerçekleştirmiştir. Merkez istasyon devresi, kullanıcı aracılığı ile lambanın yakılıp söndürülmesini, sera aydınlatmasının açılıp kapatılabilmesini, sera parametrelerinin görülebilmesi için cep telefonu üzerinden Android uygulama ile takip edilebilmesini sağlar.

Parametrelerin, LORA veri transfer modülü ile aktarımı sağlamıştır. Veri, ilk önce Raspberry pi kartına, merkez istasyon içine bağlı olan bir LORA veri transfer modülü sayesinde alınmıştır. Alınan veri, internet aracılığı ile kullanıcının GET metodu ile hostuna alınmıştır. İnternet hostundan gelen veri, PHP yazılımı ile MYSQL veri tabanından ayrı ayrı değerlerin web sayfasında gösterimini sağlamıştır.



Şekil 2.16: İnternet kontrolü ile tasarlanan lamba devresi (Aygör, 2019).

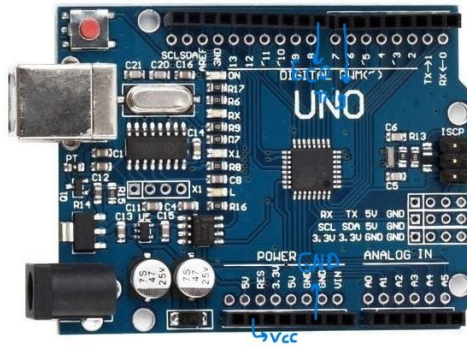
3. KAMERA VE GÖRÜNTÜ İŞLEME KULLANILARAK BORU VE PROFİL ÇAPININ ÖLÇÜLMESİ PROJESİNDEKİ KULLANILAN KODLAR

Boru ve profil ölçmek amacı ile yapılan yazılımda, ön çalışma ile yapılan işlemler ve uygulanan adım kodlarından bahsedilmektedir.

3.1 Ön Çalışma Arduino Kodları

Projede Arduino UNO kodları, sensör verilerini Python yazılım dili ile haberleşip verilerin aktarılmasını sağlamıştır. HC-SR04 sensörü kullanılmıştır. Bu sensörün kullanılma amacı, üretilen boru veya profil ile arasındaki mesafeyi öğrenebilmektir. Python ile Arduino UNO arasında seri bağlantı olarak 9600 baud rate hız ile com 3 portu üzerinden yapılmıştır. Kullanılan ultrasonik mesafe sensöründe 4 adet bacak bulunmaktadır.

Vcc 5V kaynağını, Gnd topraklama bacağını, trig sensörün ses dalgası gönderen kısmını ve echo ise gönderilen ses dalgasını alan kısmını göstermektedir. Ultrasonik mesafe sensörü için proje bağlantısı bacakları Şekil 3.1’de gösterildiği gibidir.



Şekil 3.1: Arduino HC-SR04 ultrasonik sensör bağlantısı.

```
#define trigPin 7  
#define echoPin 8
```

Şekil 3.2: Arduino kodunun yazım şekli.

Trig pini burada tetik olarak kullanılmış olup echo pini ise yansı pini olarak kullanılmıştır. Arduino pin seçimi, boş pinlere göre yapılmış olup karmaşıklığı önlemek için birbirlerine yakın pinler Şekil 3.2’de gibi seçilmiştir. Şekil 3.3’te serial haberleşme pini belirtilmiş olup giriş ve çıkış pinleri de belirtilmiştir.

```
void setup() {  
  Serial.begin(9600);  
  pinMode(trigPin, OUTPUT);  
  pinMode(echoPin, INPUT);  
}
```

Şekil 3.1: Arduino haberleşme kodu.

Haberleşme kodundan sonra, sürekli olarak çalışan döngü kodları yazılmıştır. Döngü kodları void loop() içerisine yazılmıştır. Duration ve distance adında iki uzun tamsayı (long) değişken tanımlanmıştır. Bu değişkenler, ses dalgasının sensörden yansiyarak geri gelmesi ve bu sürenin mikro saniye cinsinden ölçümü ile oluşan mesafenin saklanması sağlamaktadır. digitalWrite (trigPin, LOW) kod satırında, trigPin pinine düşük (LOW) bir sinyal gönderilmektedir. Bu, sensörün tetiklenmesini başlatmak için gerekli olan bir sinyaldir. delayMicroseconds (2) fonksiyonu 2 mikro saniye (µs) bekleme sağlar. Sensörü tetiklemek için bu kısa süre tercih edilmiştir. Bunun sebebi, hemen tetiklenmesinin istenmesidir. digitalWrite (trigPin, HIGH) kod satırında trigPin pinine yüksek (HIGH) bir sinyal gönderilmiştir. Bu, sensörün ses dalgası göndermeye başlamasını tetikler. delayMicroseconds (10) fonksiyonunda 10 mikro saniye (µs) bekleme yapılır. Bu süre, ses dalgasının gönderilmesi için gereklidir. digitalWrite (trigPin, LOW) kod satırında trigPin pinine tekrar düşük (LOW) bir sinyal gönderilir. Bu, ses dalgasının gönderiminin sonlandırılması içindir. duration = pulseIn (echoPin, HIGH) kod satırında pulseIn fonksiyonu, echoPin pinindeki yüksek (HIGH) sinyalin ne kadar süre boyunca devam ettiğini ölçer ve bu süreyi duration değişkenine kaydeder. $distance = duration \times 0.034 \div 2$ kod satırında sensörün ölçtüğü süreyi (milisaniye cinsinden) sesin hızıyla (yaklaşık 0.034 cm/milisaniye) çarparak ve ikiye bölerek nesnenin sensöre olan mesafesi hesaplanmaktadır. Serial.print("Mesafe: ") kod satırında ve Serial.println(distance) kod satırında; Ölçülen mesafe seri bağlantı üzerinden bilgisayara gönderilmiştir. delay(5) kod satırında kodun tekrar çalışması için 5 milisaniye beklenmiş ve kodlar bu sıra ile tekrardan çalışmıştır.

```

void loop() {
  long duration, distance;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;

  Serial.print("Mesafe: ");
  Serial.println(distance);

  delay(5);
}

```

Şekil 3.2: Arduino UNO void Loop () döngüsündeki kodlar.

Yapılan test ve denemelerde Arduino kartı ve yazılımı süre bakımından geciktiği ve kamera ile senkron gitmediği için bu yazılım ile birlikte kullanılan kodlar ve mesafe sensörü iptal edilmiştir. Yerine OpenCV ile çap ölçümü yapılan yazılım yapılmıştır.

3.2 Ön Çalışma Python Yazılımı Kodları

```

import serial
import cv2
import numpy as np
import tkinter as tk
from PIL import Image, ImageTk

```

Şekil 3.5: Projede Python yazılımında kullanılan kütüphaneler.

Kullanılan kütüphaneler şu şekilde açıklanabilir.

Serial kütüphanesinin kullanım amacı seri port üzerinden haberleşmeyi sağlamaktır. Bu kod satırı, projede kullanılan Arduino UNO ile haberleşmenin sağlanması için yazılmıştır ve baud rate hızı olarak 9600 tercih edilmiştir. Arduino ve Python yazılımında her ikisinin de baud rate hızları aynıdır ve Arduino UNO için com bağlantısı com3 olarak seçilmiştir.

Cv2 kütüphanesi, görüntü işleme (OpenCV), "Open Source Computer Vision Library" kelimelerinin kısaltmasıdır. Projede bilgisayar kamerası üzerinden görüntü alınması için kullanılmıştır. Diğer bir satırdaki Numpy kütüphanesi bilimsel hesaplamalar için kullanılan bir Python kütüphanesidir. Projedeki amacı Cv2 kütüphanesi ile görüntü işleme algoritmasında hız ve verimliliğin artırılmasıdır.

Tkinter kütüphanesi, kullanıcı arayüz tasarımı için kullanılmıştır. PIL ve Image, ImageTk için ise PIL (Pillow olarak da bilinir), Python için bir resim işleme kütüphanesidir. Buradaki kullanım amacı ImageTK modülünün PIL ile oluşturulan resimleri Tkinter içerisinde uygun bir formata dönüştürmesidir. Kütüphaneler, import kodu ile içe aktarılmıştır.

```
# Arduino ile seri haberleşme
ser = serial.Serial('COM3', 9600)

# Kamera başlatma
cap = cv2.VideoCapture(0)
```

Şekil 3.6: Projede kullanılan Arduino ile seri haberleşme ve video başlatma kodları.

serial.Serial('COM3', 9600) kod satırında, serial kütüphanesi aracılığıyla bir seri bağlantı başlatılmıştır. Bağlanmak istenilen seri port com3 olarak belirlenmiştir. 9600 ise iletişim hızı olarak belirlenmiştir. ser ise bir değişken olarak tanımlanmıştır, seri bağlantıyı temsil eden nesneyi içinde saklamasını sağlayan kod satırıdır. Cap=cv2.VideoCapture(0) kod satırında, cv2 kütüphanesi aracılığıyla bilgisayardaki entegre kamera aygıtı başlatılmıştır. Parantez içine yazılmış olan (0) ise kameranın indisini belirtir. Birden fazla kamera kullanılacak olsa idi bu indis 1 şeklinde değiştirilerek kullanılabilirdi. Projede bir adet kamera kullanıldığı için indis 0 olarak tercih edilmiştir.

```
# Tkinter penceresini başlatma
root = tk.Tk()
root.title("Lazer Tespiti ve Mesafe Ölçümü")
```

Şekil 3.7: Tkinter penceresi oluşturulması için kullanılmış kodlar.

Root=tk.TK() kod satırı ile yeni bir Tkinter penceresi oluşturulmuştur. Tkinter penceresinin başlığı ise lazer tespit ve mesafe ölçümü olarak belirlenmiştir.

```
# Kamera görüntüsü için Canvas
canvas = tk.Canvas(root, width=640, height=480)
canvas.pack(pady=20)
```

Şekil 3.8: Projede Tkinter arayüz penceresi oluşturma.

tk.Canvas(root, width=640, height=480) kod satırında, tkinter kütüphanesi aracılığıyla root penceresi üzerinde bir tuval (canvas) oluşturulmuştur. root

parametresi, tuvalin hangi pencere üzerinde oluşturulacağı belirtilmiştir. Burada root penceresi üzerine bir tuval eklenmiştir. width ve height kodları ise piksel cinsinden pencere büyüklüğünü göstermektedir (Şekil 3.8).

```
# Mesafe bilgisi için etiket
distance_label = tk.Label(root, text="Lazerler Arası Mesafe: 0 mm")
distance_label.pack(pady=10)
```

Şekil 3.9: Projede mesafe bilgisi için etiketlendirme.

Bu kod tkinter kütüphanesi aracılığı ile root penceresi arasında bir etiket (label) oluşturmuştur. text="Lazerler Arası Mesafe: 0 mm"; bu parametre, etiketin üzerinde gösterilecek metni belirtir. Burada etiket üzerinde "Lazerler Arası Mesafe: 0 mm" metni yer almıştır. (Şekil 3.9)

```
# Mavi çizginin boyu için etiket
line_length_label = tk.Label(root, text="Mavi Çizgi Boyu: 0 mm")
line_length_label.pack(pady=10)
```

Şekil 3.10: Ekranda mavi çizgi çizilmesi.

Bu kod, Tkinter ile bir etiket oluşturur ve bu etiketi root penceresine ekler. Etiket, "Mavi Çizgi Boyu: 0 mm" metniyle başlar ve bu metin, pencerenin üst kısmında veya belirlenen yerde gösterilmiştir. Kod satırında mavi çizginin boyunu ölçmek için tasarlanmıştır. İki adet çizgi çekilmiştir. Birinci çizgi iki adet lazer arasına çekilmiş olan kırmızı çizgidir. Diğer bir çizgi ise mavi çizgidir. Bu çizgi, gri cisim üzerine çizilmesi için tasarlanmıştır. İki çizgi arasındaki fark tolerans değerini göstermektedir.

```

def update_gui():
    global frame

    ret, frame = cap.read()
    if not ret:
        print("Kamera okunamadı!")
        root.after(10, update_gui)
        return

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    green_lower = np.array([40, 40, 40])
    green_upper = np.array([70, 255, 255])
    red_lower = np.array([0, 40, 40])
    red_upper = np.array([10, 255, 255])

    mask_green = cv2.inRange(hsv, green_lower, green_upper)
    mask_red = cv2.inRange(hsv, red_lower, red_upper)

    combined_mask = cv2.bitwise_or(mask_green, mask_red)
    contours, _ = cv2.findContours(combined_mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    if len(contours) > 0:
        largest_contour = max(contours, key=cv2.contourArea)
        (x, y, w, h) = cv2.boundingRect(largest_contour)
        cv2.line(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)

        # Mavi çizginin boyunu hesaplama
        line_length_mm = w # Bu değeri HC-SR04 sensöründen alınan veri ile güncelleyebilirsiniz.
        line_length_label.config(text=f"Mavi Çizgi Boyu: {line_length_mm} mm")

    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    img = Image.fromarray(frame)
    imgtk = ImageTk.PhotoImage(image=img)
    canvas.imgtk = imgtk
    canvas.create_image(0, 0, anchor=tk.NW, image=imgtk)

    if ser.in_waiting:
        distance_str = ser.readline().decode('latin1').strip() # Değişiklik burada
        try:
            distance_mm = float(distance_str)
            distance_label.config(text=f"Lazerler Arası Mesafe: {distance_mm:.2f} mm")
        except ValueError:
            print("Geçersiz mesafe değeri:", distance_str)

    root.after(10, update_gui)

update_gui()
root.mainloop()

cap.release()
ser.close()

```

Şekil 3.11: Projedeki ana kodlar.

def update_gui() kod satırında, update_gui adında bir fonksiyon tanımlanmıştır. global frame, frame adında bir değişkenin global değişken olduğu belirtilmiştir. Yani bu fonksiyon içinde frame değişkenini kullanırken, globalde tanımlanan frame değişkenine erişilmiştir. ret, frame = cap.read(), cap adlı kamera nesnesinden bir kare okur. Okuma başarılıysa ret True olur ve okunan kare frame değişkenine atanır şekilde dizayn edilmiştir. if not ret: print("Kamera okunamadı!") root.after(10, update_gui) return kod satırlarında, eğer kamera okuma işlemi başarısızsa, bir hata mesajı yazdırılır ve update_gui fonksiyonu tekrar çağrılır. Ardından fonksiyon sonlandırılır. Renk Filtrelemesi yapılmıştır. Görüntüyü HSV renk uzayına dönüştürmüştür. Yeşil ve kırmızı renk aralıklarına göre maske oluşturulmuştur. Büyük Kontur Bulma ile görüntüdeki konturları bulmuştur ve en

büyük konturu seçilmiştir. Mavi çizginin boyunu hesaplayarak konturun sınırlayıcı dikdörtgeninin genişliğini, yani mavi çizginin boyunu hesaplamıştır. `line_length_label.config(text=f"Mavi Çizgi Boyu: {line_length_mm} mm")` kod satırı ile hesaplanan mavi çizgi boyunu etiket üzerinde göstermek için etiketin metni güncellenmiştir. Bu kod satırının ardından görüntüyü Tkinter formatına dönüştürür ve canvas üzerinde gösterilir.

Kod satırı, `if ser.in_waiting:distance_str = ser.readline().decode('latin1').strip()` mesafe verisinin sürekli olarak okunması sağlanmıştır. Sonrasında Seri porttan okunan mesafe değerini sayıya dönüştürmeye çalışır. Eğer bu dönüşüm sırasında bir hata oluşursa, bu hatayı yakalar ve bir hata mesajı yazdırır. `update_gui` fonksiyonu ile 10ms sonra tekrardan çağırıp bir döngü oluşturur ve sürekli olarak çağırılması sağlanır. Kod tamamlandığında, kamera bağlantısı ve seri port bağlantısı sonlandırılmıştır.

Yazılan Python kodlarında lazer ile boru çapına denk geldiği ve yanlış ölçüm sonuçları alındığı görülmüştür. Borunun çapının değiştiği yerlerde lazer konumlaması zor ve sabit olduğu için çap ölçümü sağlıklı yapılamamıştır. Bunun yerine yeni Python kodları oluşturuldu.

4. ÖN ÇALIŞMADAKİ EKRAN KODLARI

```
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 210  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 210  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 210  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 210  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 210  
mesafe değeri: Mesafe: 210  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 210  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 209  
mesafe değeri: Mesafe: 209
```

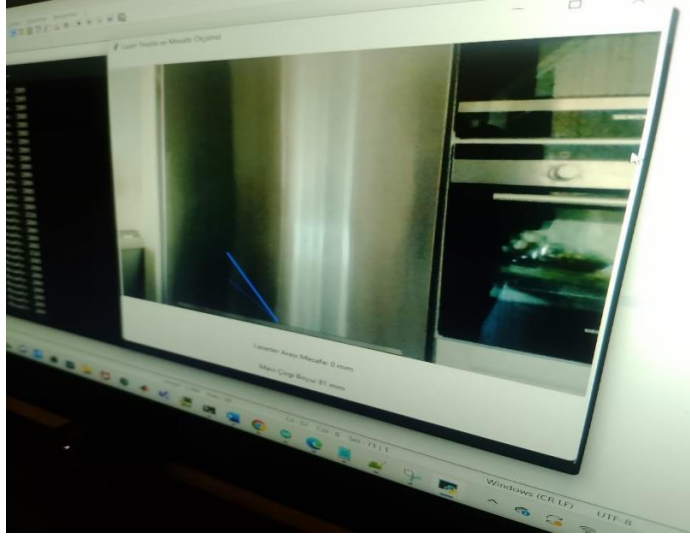
Şekil 4.1: Terminaldeki ekran ölçüm çıktıları.

Şekil 4.1’de ilk tasarımdaki ekran çıktıları verilmiştir. İki lazer arasındaki mesafe mm cinsinden gösterilmiştir.



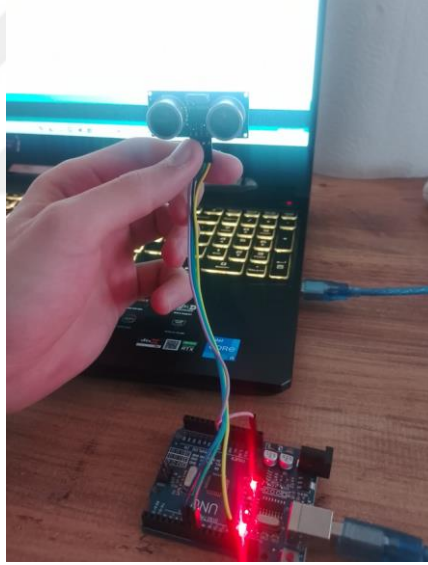
Şekil 4.2: Gerçek ölçüm.

Şekil 4.2’de mezura ile lazerler arası ölçülen mesafenin ölçümü gösterilmiştir.



Şekil 4.3: GUI penceresi Tkinter görüntüsü.

Lazerler arasındaki mesafenin GUI ekranı üzerindeki görüntüsü verilmiştir.



Şekil 4.4: Ultrasonik mesafe sensörü.

Ön çalışmadaki sistemde kullanılan elektronik cihazlardan ultrasonik mesafe sensörü sistemde kamera ve cisim arasındaki mesafe ile gerçek zamanda ölçüm doğruluğunu artırmak için kullanılmıştır.

5. BORU PROFİL ÇAP ÖLÇÜM PROJESİNDEKİ PYTHON KODLARI

5.1 Kullanılan Kütüphaneler ve Ne için Kullanıldıkları

```
import cv2
import numpy as np
import tkinter as tk
from tkinter import ttk
from PIL import Image, ImageTk
```

Şekil 5.1: Yazılımda kullanılan kütüphane kodları.

5.1.1 OpenCV (cv2) kütüphanesi

OpenCV, bilgisayarla görme ve görüntü işleme görevlerini gerçekleştirmek için kullanılan popüler bir kütüphanedir. OpenCV şu şekilde kullanılır.

Görüntü Okuma ve Yazma: OpenCV, görüntüleri yükleyip işlemek, kaydetmek için işlevler sunar.

Görüntü Dönüştürme: Görüntüleri gri tonlamaya çevirme, eşikleme (thresholding) gibi dönüştürme işlemlerini yapar.

Kontur Bulma: Görüntüdeki konturları bulma ve işleme yeteneğine sahiptir.

Görüntü Çizimleri: Görüntü üzerinde çizim yaparak nesneleri vurgulama (örneğin, daire, dikdörtgen çizme).

OpenCV'nin tercih edilme sebeplerinden birkaç tanesi şu şekildedir. OpenCV, C/C++ tabanlı olduğundan dolayı hızlı ve verimlidir, bilgisayar görmesi bakımından geniş bir yelpazeye sahiptir, platformlardaki bağımsızlığı sebebi ile Windows, linux, MacOS gibi platformlarda çalıştırılabilir.

5.1.2 NumPy (np) kütüphanesi

NumPy, büyük çok boyutlu dizi ve matrislerle çalışmayı kolaylaştıran, bu diziler üzerinde yüksek seviyede matematiksel işlemler yapmaya olanak tanıyan bir kütüphanedir.

Ne İşe Yarar? Dizi İşlemleri: Görüntüler, esasen büyük boyutlu matrisler/diziler olarak temsil edilir ve NumPy bu tür veri yapılarını verimli bir şekilde işlemek için kullanılır.

Matematiksel İşlemler: Matematiksel işlemleri hızlandırır ve kolaylaştırır.

Neden Tercih Edilir? Verimlilik: NumPy, optimize edilmiş C kodu kullanarak işlemleri hızlandırır.

Kolay Entegrasyon: Diğer bilimsel kütüphanelerle (Pandas, SciPy) ve özellikle OpenCV ile mükemmel uyum içinde çalışır.

5.1.3 Tkinter (tk) kütüphanesi

Tkinter GUI (arayüz) geliştirmek için kullanılan standart bir kütüphanedir. Kolay kullanımı ve zengin görsel arayüz oluşturma imkanlarından dolayı tercih edilmiştir.

5.1.4 PIL (Pillow) kütüphanesi

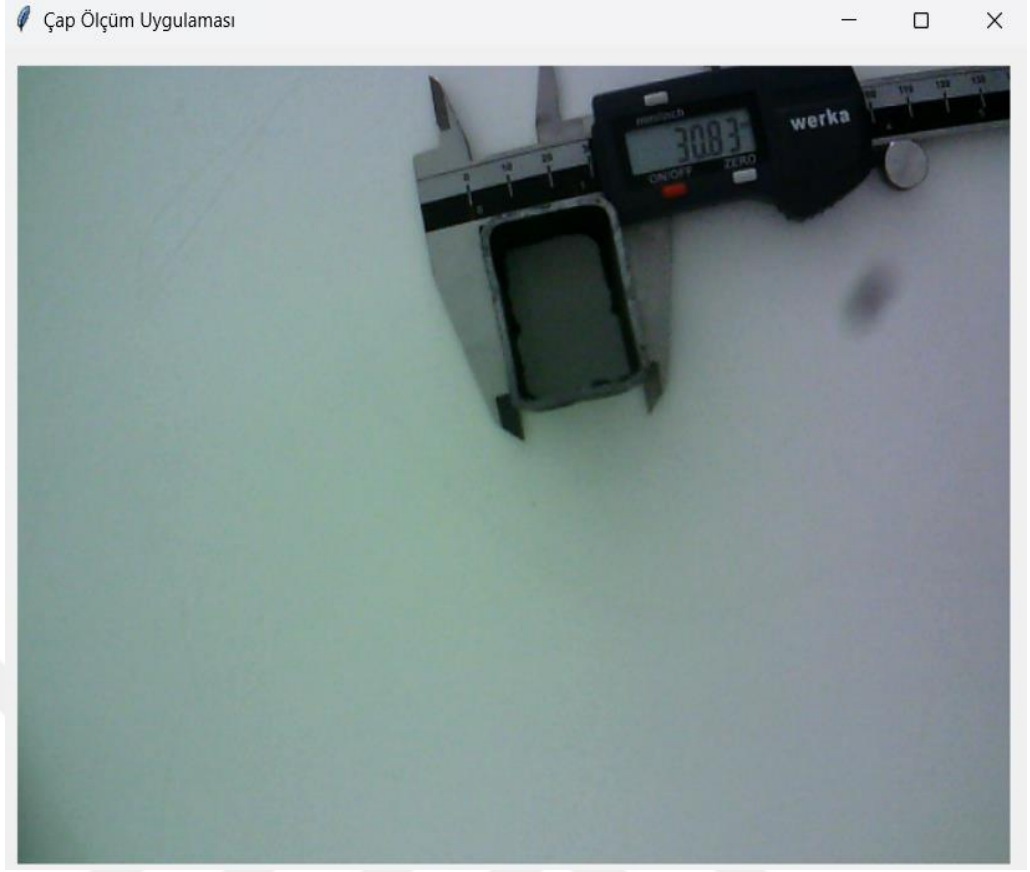
Pillow, Python için bir görüntü işleme kütüphanesi olan PIL'in (Python Imaging Library) devamı niteliğindedir. Görüntüleri yükleme, işleme ve dönüştürme yeteneklerine sahiptir. Görüntüleri kaydetme, dönüştürme, yeniden boyutlandırma gibi işlemleri yapar, tkinter ile kullanılabilen görüntü nesneleri oluşturur.

6. PROJE EKLAN ÇIKTILARI



Şekil 6.1: Profil uzunluğu kumpas ölçüsü.

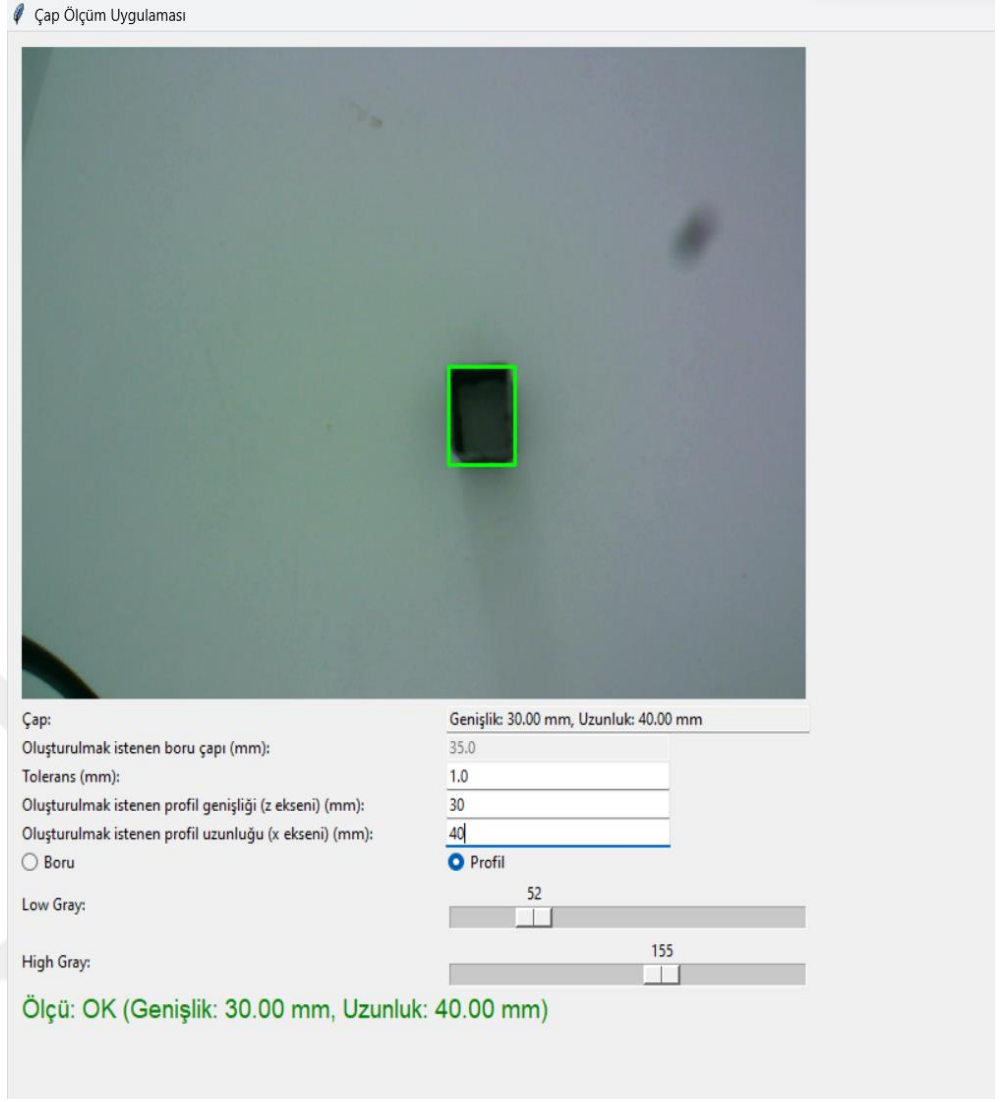
Bir parça profilin kalibrasyonu ve doğru sonuçları almak için kumpas yardımı ile genişlik ve uzunluk olarak ölçüm yapılmıştır. Yapılan ölçüm sonuçlarında tolerans değeri kullanıcı talebine göre değişmektedir. Ondalık değer de girilebilmektedir.



Şekil 6.2: profil genişliği kumpas ölçüsü.

Oluşturulmak istenen profil genişlik ve uzunluk olarak girilmiştir. Kamera açısı ve mesafesi burada önemli noktalardan biridir. Kamera kalibrasyona göre kumpas ile ölçülen değer için mesafesi ve açısı ayarlanmıştır.

Girilen değerler doğrultusunda tolerans değeri kullanıcı tarafından üretilmek istenen değerler doğrultusunda değişebilir ve buna göre ölçüm yapılabilmektedir. Girilen tolerans değeri doğrultusunda üretilmek istenen $\pm$ eklenip çıkartılarak değer aralığı çap için ok olarak yazılmaktadır. Aksi halde üretim tolerans dışında ise uyarı ile farkı kırmızı ile yazmaktadır.



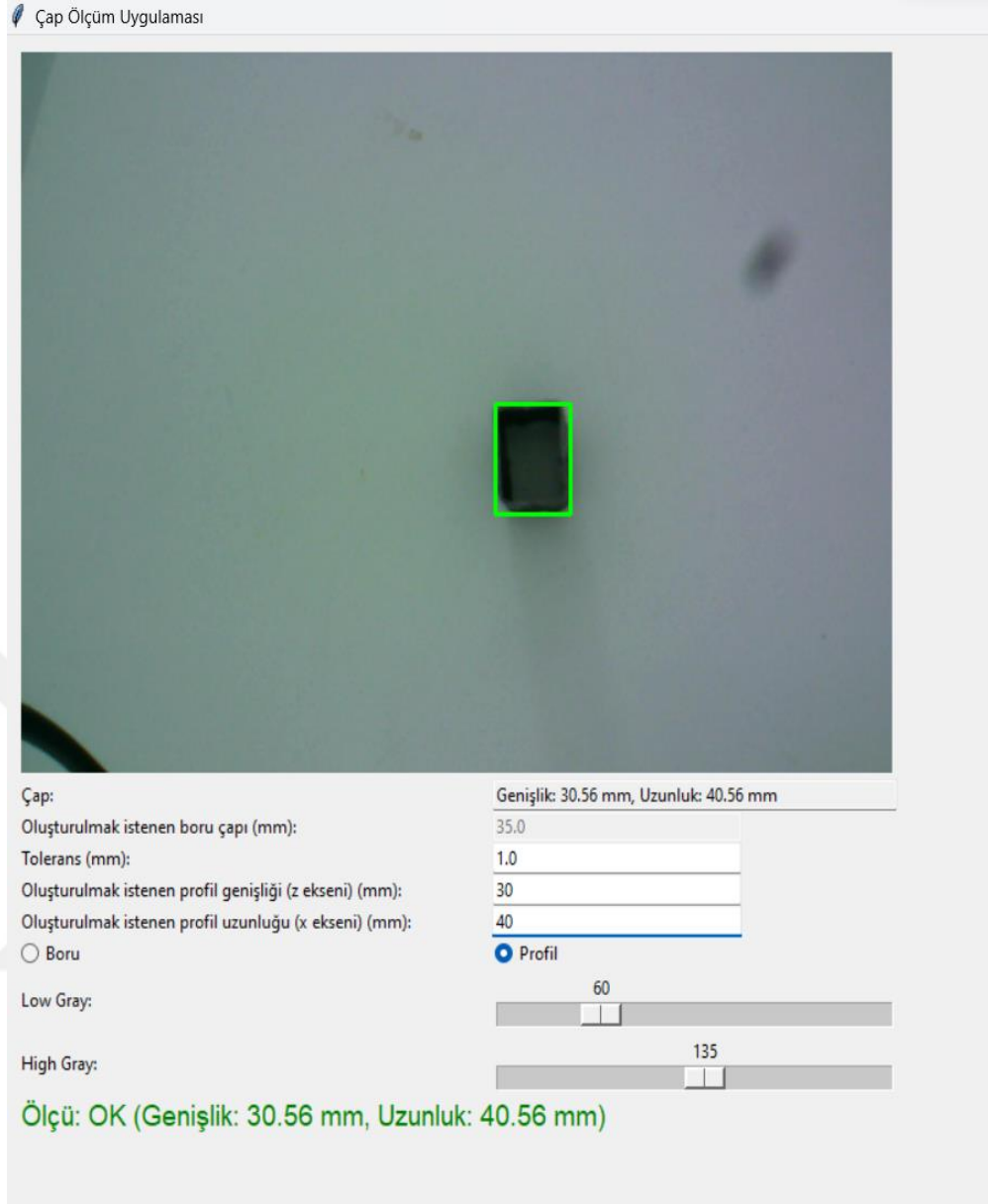
Şekil 6.3: Profil ölçüm sonucu.

Ölçüm sonuçları kalibrasyonlar şu şekilde yapılmıştır;

Açı: cisime dik

Mesafe : 37 cm olarak ayarlanmıştır.

Genişlik ve uzunluk olarak tam ölçüm sonucu alınmıştır. Ortama göre low ve high gray tonları ayarlanmıştır. Low ve high aralığı ortama ve ışık şiddetine göre farklılık gösterebilmektedir.



Şekil 6.4: Ölçüm sonucu 2.

Low ve high gray değerler aralıklarını biraz daha yaklaştırılıp kumpas ile ölçülen sonuca daha yakın bir değer ölçülmüştür. % 1,86 değerinde bir hata payı olarak sonuç alınmıştır.



Şekil 6.5: Kare profil genişlik kumpas ölçümü.

Kalibrasyon yapılabilmesi için kumpas yardımı ile profil parçasının ölçümü yapılmıştır. Şekilde genişlik ölçümü gösterilmiştir. GUI ekranı üzerinde dikdörtgen profillerde genişlik ve uzunluk ölçüleri 3 eksen üzerinden z ve x ekseni ile GUI ekranı üzerinde belirtilmiştir. Bu sayede doğru şekilde kalibrasyon yapılabilmesi sağlanmıştır.



Şekil 6.6: Kare profil uzunluk kumpas ölçümü.

Kare profil için kumpas ile şekilde uzunluk ölçümü yapılmıştır. Bu sayede kalibrasyon işlemi için kolaylık sağlanmıştır. Ve kamera mesafesi olarak ayarlama yapılması sağlanmıştır.

Çap Ölçüm Uygulaması

Genişlik: 31.11 mm, Uzunluk: 31.11 mm

Çap:

Oluşturulmak istenen boru çapı (mm): 35.0

Tolerans (mm): 1.0

Oluşturulmak istenen profil genişliği (z eksen) (mm): 30

Oluşturulmak istenen profil uzunluğu (x eksen) (mm): 30

☐ Boru ☒ Profil

Low Gray: 60

High Gray: 135

Ölçü: Tolerans Dışı (Genişlik: 31.11 mm, 0.11 mm büyük, Uzunluk: 31.11 mm, 0.11 mm büyük)

Şekil 6.7: Kare profil ölçüm sonucu.

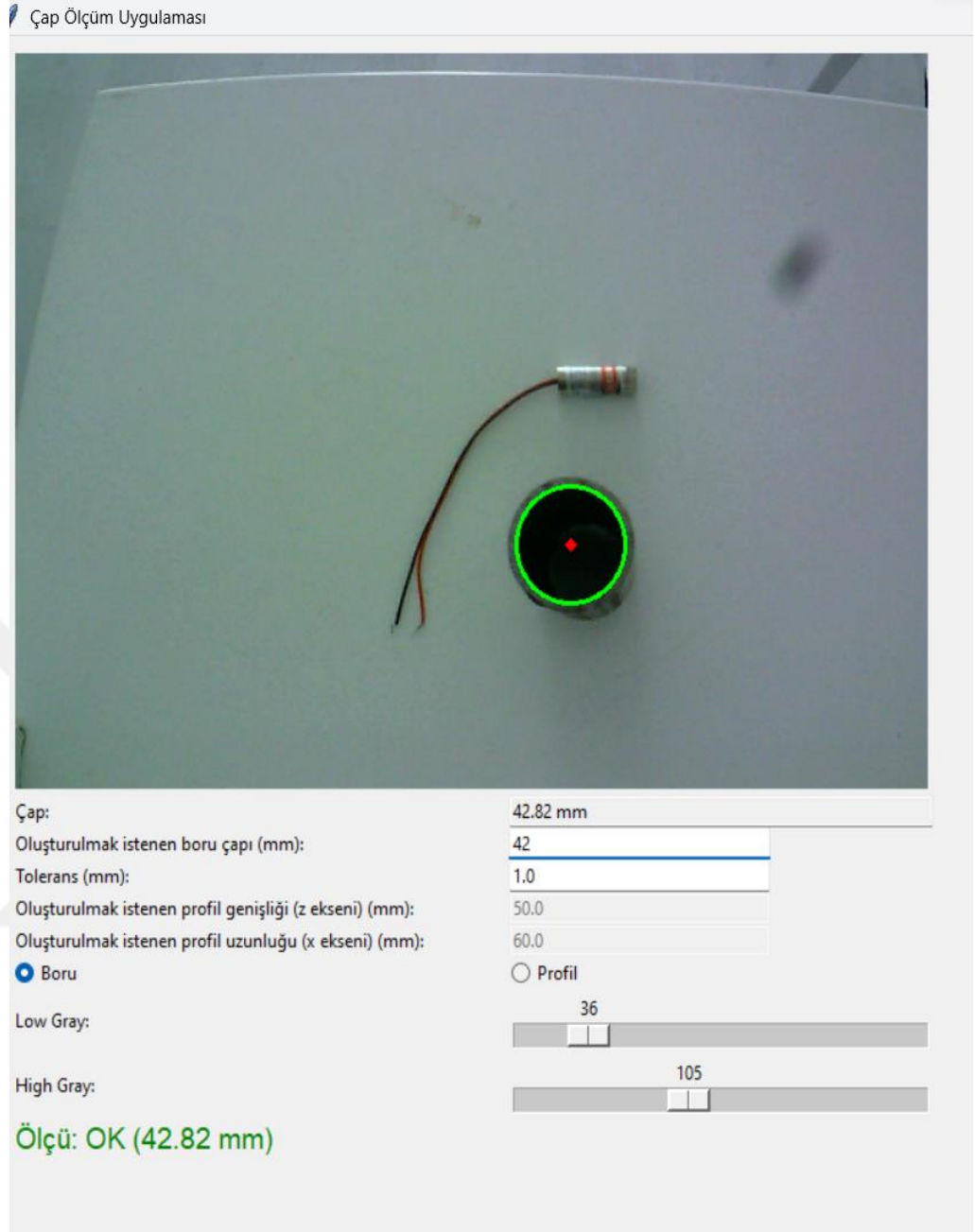
Kare profil ölçüm sonuçları yukarıdaki şekilde gibidir. Kalibrasyon kamera uzaklığı için önceki ölçümdeki ile aynı uzaklık 37 cm olarak ayarlanmıştır, değiştirilmemiştir. Low ve high parametreleri değiştirilmemiştir, 60-135 olarak girilmiştir. Oluşturulmak istenen profil genişlik 30 mm ve uzunluk 30mm olarak değiştirilmiş olup tolerans değeri 1mm olarak girilmiştir. Bu ölçüm sonuçlarına göre en yüksek hata payı %1.5 olarak hesaplanmıştır.



Şekil 6.8: Kumpas boru ölçüm sonucu.

Alınan boru parçası çapı kumpas yardımı ile yukarıdaki şekilde görüldüğü gibi ölçülmüştür. Kalibrasyon için 1 seferlik ölçüm ve kamera mesafesi ayarlanmıştır.

Ortam koşullarına göre low, high gri renk skalası aralığı belirlenmiştir. Boru çapı kumpasla ölçüldüğü gibi 42 mm'dir ve tolerans değeri kullanıcının gireceği rakam aralığındadır. Değerler girildikten sonra kamera başlatılmıştır ve kamera mesafesi ölçüm sonuçlarına göre ayarlanmıştır. Kamera mesafesi 37 cm, kamera açısı dik olacak şekilde konumlandırılmış ve ölçümler alınmıştır.

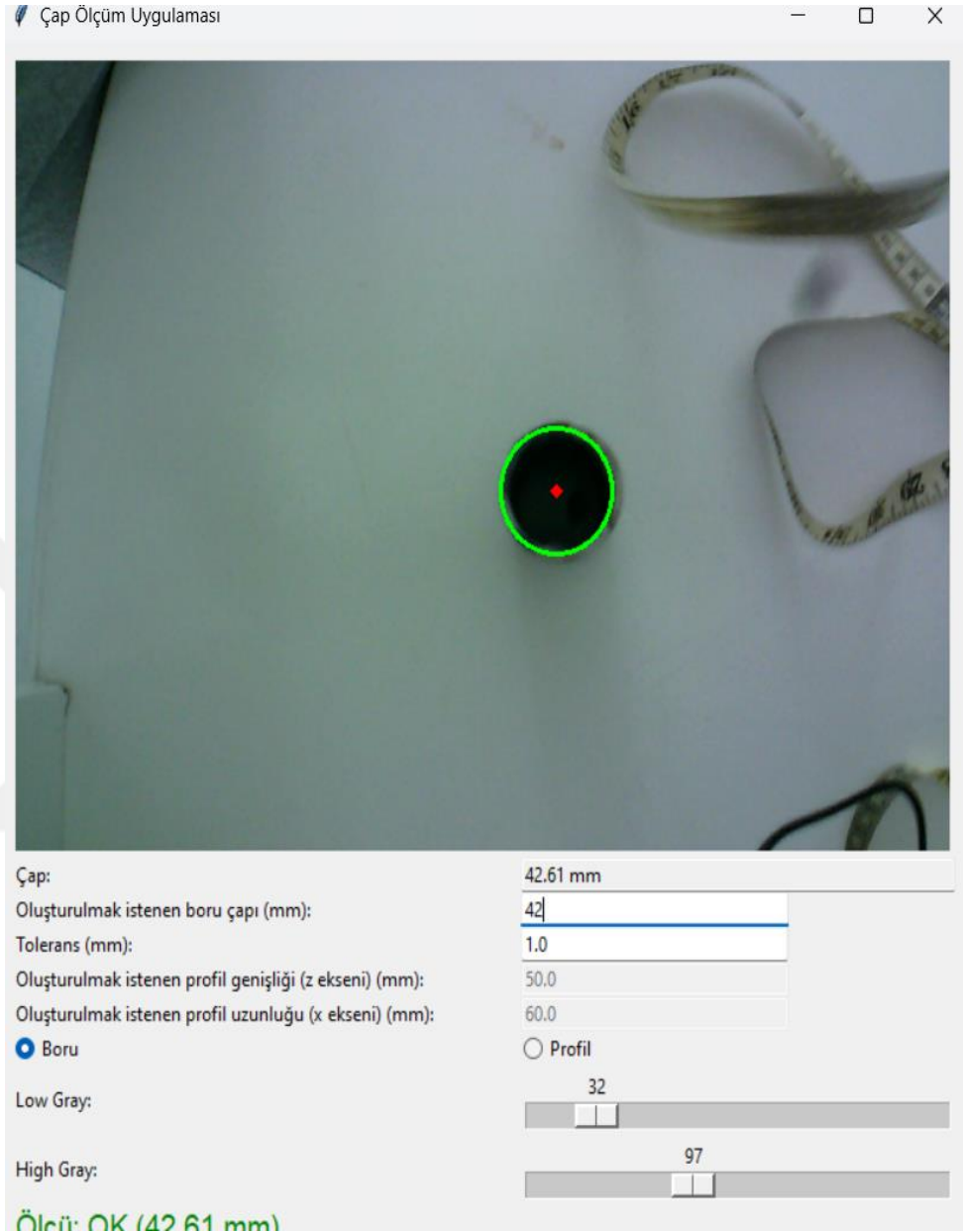


Şekil 6.9: Boru ölçüm sonucu ilk deneme.

Yukarıdaki şekilde görüldüğü gibi değerler girilmiş ve profil butonu seçilmediği için de aktif durumdadır, bu durum kullanıcının hatalı bilgi girmesini ve yanlış ölçüm sonuçları almasını engellemiştir.

Low high gray aralıkları 36-105 olarak seçilmiş ve mesafe 37 cm olarak cisim tam karşına gelecek şekilde kamera konumlandırılmıştır. Ölçüm sonucu 42.82 şeklinde çıkmıştır. Kumpas ile ölçüm sonucundan %0.72 değerinde bir sapma olmuştur.

Low-high gray aralıklarında boruyu tam olarak kavramadığı görüldüğü için oynamalar yapıp ikinci deneme yapılmıştır.



Şekil 6.10: Boru ölçüm sonucu ikinci deneme.

Boru için ikinci deneme de ise low hig gray aralığı 32-97 olarak ayarlanmış ve boru dışı tam olarak çerçeve içine alındığı görülmüştür, kamera mesafesi 37 cm, ve cisime dik bir şekilde konumlandırılmıştır. Buna göre ölçüm sonuçları 42.61mm olarak bulunmuştur. Kumpas ölçümüne göre sapma değeri % 0.23 olarak hesaplanmıştır.



Şekil 6.11: Kamera ile cisim arasındaki mesafe.

Kamera cisim arasındaki mesafe kalibrasyon için ölçülmüştür. Yapılan denemelerde genel olarak 37 cm'den oynatılmamıştır ve bu şekilde ölçüm sonuçları elde edilmiştir. Kumpas ölçümüne göre profillerde ölçüm hataları %1- %1.90 arasında değişmektedir. Bu durum kamera uzaklığı salınım, ortam ışığı yazılımı etkilemektedir. Mesafe sabit tutulup low-high gray skalasında oynama yapılmıştır.

7. TARTIŞMA

Bu yüksek lisans tezi kapsamında geliştirilen yazılım, boru ve profil malzemelerinin ölçümünü otomatikleştirmek amacıyla gerçek zamanlı kamera görüntü işlemlerini kullanarak önemli avantajlar sunmaktadır. Algoritmanın en önemli avantajları arasında, manuel ölçüm gereksinimini ortadan kaldırarak zaman ve maliyet tasarrufu sağlanması bulunmaktadır. OpenCV ve Tkinter kullanılarak geliştirilen algoritma, nesnelerin doğru ve hızlı bir şekilde tanımlanmasını ve ölçülmesini sağlamaktadır. Ayrıca, yazılımın yüksek çözünürlüklü kameralarla kullanılması, ölçüm hassasiyetini artırarak küçük detayların algılanmasını mümkün kılmaktadır. Öte yandan, yazılımın bazı zayıf yönleri de bulunmaktadır. Örneğin, kullanılan gri tonlama ve eşikleme teknikleri, belirli ışık koşullarında ve yüzey özelliklerinde hassasiyet kaybına yol açabilir. Ayrıca, hareketli üretim hatlarında kullanılan gerçek zamanlı görüntü işlemlerinde, yazılımın performansı ve stabilitesi test edilmelidir. Bu durum, yazılımın daha dinamik ve karmaşık üretim süreçlerine entegrasyonu sırasında önemli bir zorluk oluşturabilir.

8. SONUÇLAR VE ÖNERİLER

Bu yüksek lisans projesi, endüstriyel üretim süreçlerinde kullanılan boru ve profil malzemelerinin ölçümünü otomatikleştirmek için gerçek zamanlı kamera görüntü işlemlerini kullanan bir yazılım geliştirmeyi hedeflemektedir. Geliştirilen yazılım, belirlenen tolerans aralıklarında çap ve profil boyutlarının doğruluğunu kontrol etmek amacıyla kamera görüntülerinden elde edilen verileri işler. Bu yöntem, üretim süreçlerinde manuel ölçüm gereksinimini ortadan kaldırarak hem zaman hem de maliyet açısından önemli tasarruflar sağlamayı amaçlamaktadır.

Tablo 8.1: Fabrika ortamında kamera ölçümünün gerçek zaman ölçümü ile karşılaştırma tablosu.

Deneme	Kumpas ölçüsü	Kamera cisim uzaklığı	Ortam aydınlığı	Kamera ölçü sonucu
Boru 1	42.91 mm	31 cm	Gün ışığı	42.66 mm
Boru 2	58.64 mm	31 cm	Gün ışığı	58.84 mm
Boru 3	34.02 mm	31 cm	Gün ışığı	34.29 mm
Boru 4	48.64 mm	31 cm	Gün ışığı	48.78 mm
Profil 1	Kısa kenar: 40.31 mm Uzun kenar: 60.43 mm	31 cm	Beyaz ışık ile desteklendi	Kısa kenar: 40.56 mm Uzun kenar: 60.00 mm
Profil 2	Kısa kenar: 41.48 mm Uzun kenar: 80.19 mm	31 cm	Beyaz ışık ile desteklendi	Kısa kenar: 43.33 mm Uzun kenar: 80.00 mm

Fabrika ortamında deneme sonuçları yukarıdaki tabloda verilmiştir. Denemeler önce gündüz vakti yapılmıştır, bundan dolayı herhangi bir aydınlatmaya gerek duyulmamıştır. Sonra, daha karanlık bir ortama geçilmiş ve kameranın net görüntü almasını sağlamak amacıyla telefon ışığı ile desteklenmiştir. Boru ve profil denemeleri gerçekleştirilmiştir. Kamera ve cisim arası uzaklık 31 cm olarak ölçüm sonuçları doğrulanmıştır ve cisim kumpas ile ölçülüp kamera ölçümüyle kıyaslanmıştır.

İlk yapılan yazılımda iki lazer kullanılmıştır ve bu yazılımdan vazgeçilmesinin nedenlerinden bir tanesi lazerlerin konumu değişen boru çapına göre konum almamasındandır. Lazerler sabit bir şekilde başlatılıyor ve kamera bu lazerlere göre ölçüm alıyordu fakat yazılımın ilerisi hedeflendiğinden dolayı ilerleyen hat üzerinde bu şekilde bir durum çalışmayacaktı. Bundan dolayı lazer ve arduino kullanılan yazılımdan vazgeçilip OpenCV Tkinter arayüzü kullanılarak proje sonuca ulaşmıştır.

Yazılım, OpenCV ve Tkinter kütüphanelerini kullanarak geliştirilmiştir. OpenCV, görüntü işleme ve bilgisayarla görme görevlerinde kapsamlı araçlar sunarak nesnelerin tanımlanmasını ve boyutlarının ölçülmesini sağlar. Tkinter ise kullanıcı dostu bir arayüz oluşturmak için kullanılmıştır. Kamera görüntüleri gri tonlamaya dönüştürülmüş, ardından eşikleme ve kontur algılama teknikleriyle nesnelerin boyutları belirlenmiştir. Elde edilen bu veriler, belirlenen tolerans aralıklarıyla karşılaştırılmış ve sonuçlar kullanıcıya görsel ve metinsel olarak sunulmuştur.

Yazılımda daha yüksek çözünürlüklü kamera kullanılması mevcut kameranın çözünürlüğünün artırılması, ölçümlerin hassasiyetini artırabilir. Özellikle küçük çaplı boru ve profillerin daha doğru ölçülmesi sağlanabilir. Yüksek çözünürlüklü kameralar, daha küçük detayların algılanmasına olanak tanır ve ölçüm doğruluğunu artırır.

Farklı uygulama alanlarında kullanım sağlayabilir. Yazılımın yalnızca boru ve profil ölçümleriyle sınırlı kalmaması, farklı malzemeler ve ürünlerin kalite kontrolü için de uyarlanması kullanım alanını genişletecektir. Örneğin, plastik, cam ve kompozit malzemelerin ölçümü için de kullanılabilir.

Sabit bir ortamda gerçekleştirilen ölçümlerden, hareketli üretim hatlarında gerçek zamanlı kalite kontrolüne geçiş yapılabilir. Bu, üretim süreçlerini daha dinamik ve verimli hale getirerek üretim hızını artırabilir ve hata oranını düşürebilir.

Yazılımın mevcut üretim yönetim sistemleriyle entegre edilmesi, veri akışının daha sorunsuz ve hızlı olmasını sağlar. Bu, üretim hattında anlık geri bildirimler ve ayarlamalar yapmayı mümkün kılar. Ayrıca, entegre sistemler, üretim verimliliğini artırarak operasyonel maliyetleri düşürebilir.

Robotik sistemlerle entegre çalışarak, otomatik düzeltme ve ayar işlemleri yapılabilir. Bu, üretim sürecindeki insan müdahalesini minimize eder ve hata oranını

düşürür. Örneğin, robotik kolların kullanılması, hatalı ürünlerin otomatik olarak ayıklanmasını veya düzeltme işlemlerinin yapılmasını sağlayabilir.

Bu proje, endüstriyel ölçüm ve kalite kontrol süreçlerine önemli katkılar sunmaktadır. Geliştirilen yazılımın daha da iyileştirilmesi ve farklı uygulama alanlarına adapte edilmesi, üretim süreçlerinde verimliliği ve doğruluğu artıracaktır. Bu bağlamda, önerilen geliştirme ve entegrasyon stratejileri, endüstriyel uygulamalarda daha geniş bir kabul görmesini sağlayacaktır. Yazılımın kullanımının yaygınlaştırılması, üretim süreçlerinde kalite kontrolün daha güvenilir ve hızlı bir şekilde gerçekleştirilmesine olanak tanıyacaktır.

Bu projede elde edilen sonuçlar, gelecekte yapılacak çalışmalar için bir temel oluşturmaktadır. İlerleyen aşamalarda, yazılımın daha geniş bir yelpazede malzeme türleri ve üretim süreçleri için adaptasyonu hedeflenebilir. Ayrıca, kullanıcı deneyiminin iyileştirilmesi için daha sezgisel ve kullanıcı dostu arayüzler geliştirilebilir. Bunun yanı sıra, yapay zeka ve makine öğrenmesi algoritmalarının entegrasyonu, yazılımın ölçüm ve hata tespit becerilerini daha da ileriye taşıyacaktır. Sonuç olarak, bu projenin çıktıları, endüstriyel üretim süreçlerinde kalite kontrolün daha hızlı, doğru ve verimli bir şekilde gerçekleştirilmesine katkı sağlayacak önemli bir adım olarak değerlendirilmektedir.

9. KAYNAKLAR

- Akdoğan, M., “Karmaşık şekilli mamüllerin görüntü işleme teknikleri kullanılarak gerçek zamanlı boyut kontrolü”, Doktora Tezi, *Selçuk Üniversitesi Fen Bilimleri Enstitüsü*, Konya, (2018).
- Asaad, R. R., Ali, R. I., Shaban, A. A., and Salih, M. S., “Object Detection Using the ImageAI Library in Python”, *Polaris Global Journal of Scholarly Research and Trends*, 2(2), 1-9, (2023).
- Atilgan, M., “Sayısal görüntü işleme teknikleri kullanılarak iki boyutlu görüntü üzerinde yapılan üç boyutlu manipülasyonun analizi”, Yüksek Lisans Tezi, *Ankara Üniversitesi Sağlık Bilimleri Enstitüsü*, Ankara, (2017).
- Aygör, Y., “Arduino ile güneş enerjili sera otomasyonu”, Yüksek Lisans Tezi, *Çanakkale Onsekiz Mart Üniversitesi Fen Bilimleri Enstitüsü*, Çanakkale, (2019).
- Çobanoğlu, B., Herkes için Python, Pusula, (2021).
- Duman, B., “Görüntü işleme tekniklerinin eklemeli imalatla kullanımı”, *4th international congress on 3d printing (additive manufacturing) technologies and digital industry*, Antalya, (2019).
- Dwyer, P. G., “Review article: Total War or traditional war”, *The International History Review*, 31(1), 72-84, (2009).
- Gonzalez R. C. and Woods R. E., *Digital Image Processing*, 3th edition, New Jersey, USA: Pearson, 1-7, (2007).
- Güngörmüş, A., “Görüntü işleme teknikleri kullanarak petek üzerindeki arı larvasının konumunun ve özelliklerinin tespiti”, Yüksek Lisans Tezi, *Balıkesir Üniversitesi Fen Bilimleri Enstitüsü*, Balıkesir, (2020).
- Huang, J., Guo, M., Wang, S. and Chen, S., “A motion sensor with on-chip pixel rendering module for optical flow gradient extraction”, *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, Florence, Italy, 1-5, (2018).
- İlgin, S., “Python nedir? [online]”, (10 April 2024), <https://istanbulbogazicienstitu.com/python-nedir>, (2023).
- İlgin, U., “Python’da görüntü işleme uygulamaları [online]”, (12 April 2024), https://www.researchgate.net/publication/349427117_PYTHON'DA_GORUNTU_IS_LEME_UYGULAMALARI, (2020).
- Jahne, B., *Digital Image Processing*, ISBN 3-540-24035-7, Berlin, Germany: Springer, 7-19, (2005).

Jaykumaran, R., “Simple edge defect detection in concentric circles using OpenCV and YOLO [online]”, (15 May 2024), <https://medium.com/@jaykumaran2217/simple-edge-defect-detection-in-concentric-circles-using-opencv-and-yolo-8bfed4c181b6>, (2023).

Küçükpara, Z., “Görüntü işleme yöntemi ile optik işaret tanıma ve değerlendirme sistemi”, Yüksek Lisans Tezi, *Necmettin Erbakan Üniversitesi Fen Bilimleri Enstitüsü*, Konya, (2019).

Musthaq, S. and Mir A. H., “Digital image forgeries and Passive Image Authentication Techniques: A Survey”, *International Journal of Advanced Science and Technology*, 73, 15-32, (2014).

Özgül, F., *Python Kılavuzu*, İstanbul, Türkiye : Pusula Yayıncılık, (2015).

Pathak, A. R., Pandey, M. and Rautaray, S., “Application of deep learning for object detection”, *International Conference on Computational Intelligence and Data Science*, 132, Bhubaneswar, India, 1706–1717, (2018).

Perihanoğlu, G. M., “Dijital görüntü işleme teknikleri kullanılarak görüntülerden detay çıkarımı”, Yüksek Lisans Tezi, *İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü*, İstanbul, (2015).

Pulungan, A. B., Nafis, Z., Anwar, M., Hastuti, H., Hamdani, H. and Myori, D. E. M., “Object Detection with a Webcam Using the Python Programming Language”, *Journal of Applied Engineering and Technological Science*, 2(2), 103-111, (2021).

Suzuki, S. ve Abe, K., “Topological structural analysis of digitized binary images by border following”, *Computer Vision, Graphics, and Image Processing*, 32-46, (1985).

Sevli, O., *Python*, İstanbul, Türkiye : Kodlab yayın dağıtım yazılım Ltd. Şti., (2021).

Sharma, A., Pathak, J., Prakash M. and Singh, J. N., “Object detection using OpenCV and Python”, *3rd International Conference on Advances in Computing, Communication Control and Networking 2021*, Greater Noida, India, 501-505, (2021).

Tasalı, M., “Lazer ışığını takip eden araç prototipi tasarımı”, Yüksek Lisans Tezi, *Aksaray Üniversitesi Fen Bilimleri Enstitüsü*, Elektrik Elektronik Mühendisliği Anabilim Dalı, Aksaray, (2023).

Tarı, M., “Görüntü işleme teknikleri ile enjektör üretiminde kalite kontrolü”, Yüksek Lisans Tezi, *Konya Teknik Üniversitesi Lisansüstü Eğitim Enstitüsü*, Bilgisayar Mühendisliği Anabilim Dalı, Konya, (2020).

Topal, A. Yiğit T., “İnsansız Kara Araçları İçin LIDAR Teknolojisi Kullanılarak 3B Ortam Haritalama Sistemi”, *International Journal of 3D Printing Technologies and Digital Industry*, 5, 171-186, (2021).

Uğurlu, B., “Titanyum implantının tribokorozyon özelliklerinin incelenmesi ve sayısal görüntü işleme teknikleri kullanılarak tribokorozyonun analizi”, Yüksek Lisans Tezi, *Alanya Alaaddin Keykubat Üniversitesi Lisansüstü Eğitim Enstitüsü*, Metalurji ve Malzeme Anabilim Dalı, Alanya, (2022).

Yanık, H. ve Turan, B., “Mesafe ölçümünde image-metre: görüntü işleme ile mesafe tespit yöntemlerinde eksik ne?”, *International Conference on Data Science and Applications*, Türkiye, 480-486, (2021).

Yoldaş, M. ve Sungur, C., “Alüminyum Ekstrüzyon Profillerinin Hassas Kesit Ölçümlerinin Görüntü İşleme Teknolojisi ile Gerçekleştirilmesi”, *European Journal of Science and Technology*, 190-195, <https://doi.org/10.31590/ejosat.803308>, (2020).

Zhang, L. Zhou, U. Ve Konz, N., ‘‘ Computer Vision Techniques in Manufacturing’’, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 53, 105-117, (2023).



EKLER

EK A

OpenCV kullanılarak kamera görüntüsü alınır, işlenir (gri tonlama, eşikleme, kontur bulma) ve üzerinde çizimler yapılır. NumPy kullanılarak görüntü verileri ve matematiksel işlemler daha verimli bir şekilde gerçekleştirilir. Tkinter kullanılarak kullanıcı arayüzü oluşturulur. Bu, kullanıcıların görüntü üzerinde belirli ayarları yapmasına ve sonuçları görmesine olanak tanır. Pillow kullanılarak OpenCV'den alınan görüntüler, Tkinter canvas'ında gösterilebilir formata dönüştürülür.

```
# Bilinen bir nesnenin çapı (mm cinsinden) ve piksellerdeki çapı
KNOWN_DIAMETER_MM = 50.0 # Bu değer bilinen bir nesneye göre ayarlandı
KNOWN_DIAMETER_PIXELS = 100.0 # Bu değer kalibrasyon sonucu elde edildi

# Ölçek faktörü (mm/piksel)
SCALE_FACTOR = KNOWN_DIAMETER_MM / KNOWN_DIAMETER_PIXELS
```

Şekil A.1: Kod dizimi.

```
# Bilinen bir nesnenin çapı (mm cinsinden) ve piksellerdeki çapı
KNOWN_DIAMETER_MM = 50. # Bu değer bilinen bir nesneye göre ayarlandı
KNOWN_DIAMETER_PIXELS = 90.0 # Bu değer kalibrasyon sonucu elde edildi
```

```
# Ölçek faktörü (mm/piksel)
```

```
SCALE_FACTOR = KNOWN_DIAMETER_MM / KNOWN_DIAMETER_PIXELS
```

Bu kod dizini kullanılmasının amacı bilinen boyutlardaki bir nesneyi baz alarak, diğer cisimlerde doğru sonuçlar elde edilmesine katkı sağlamalarından dolayıdır.

$KNOWN_DIAMETER_MM = 50.0$ bu fiziksel olarak bilinen bir cismin çapıdır. Herhangi bir cisim varsayılabilir. Örneğin bir daire olabilir.

$KNOWN_DIAMETER_PIXELS = 100.0$ kod satırı bilinen aynı nesnenin bir görüntüdeki piksel cinsinden çapıdır. Bu değer, kalibrasyon sonucu elde edilir ve nesnenin görüntüdeki boyutunu ifade eder.

$SCALE_FACTOR = KNOWN_DIAMETER_MM / KNOWN_DIAMETER_PIXELS$ Bu ölçek faktörü, görüntüdeki pikselleri gerçek dünya birimlerine (milimetre) dönüştürmek için kullanılmıştır.

Ölçek faktörü formülü ise şu şekildedir: ölçek faktörü = bilinen çap (mm) / bilinen çap(piksel). Bu formül, bilinen fiziksel bir boyutu (mm) aynı boyutun görüntüdeki piksel karşılığına bölerek, her bir pikselin fiziksel dünyadaki karşılığını milimetre olarak belirler. Ölçüm süreci şu şekilde gerçekleşmiştir, bilinen çaplı bir nesne seçilmiştir, referans nesnenin bir görüntüsünü alır ve OpenCV ile bir kontur bulunur, nesnenin piksel cinsinden çapı ölçülür, bilinen çap ve görüntüdeki çap kullanılarak ölçek faktörü hesaplanır. Bu ölçek faktörü, görüntüdeki diğer nesnelerin boyutlarını hesaplamak için kullanılır. Bu şekilde bir yöntemin seçilmesinin sebebi ise bilinen bir cisim ile fiziksel boyutlarını kullanarak ölçek faktörünü hesaplamanın doğruluğunu artırmayı sağlamaktır. Bu şekilde başka bir cismin çapını doğru şekilde ölçmeye yardımcı olmaktadır.

```
class DiameterMeasurementApp:
    def __init__(self, root, camera_index=1):
        self.root = root
        self.root.title("Çap Ölçüm Uygulaması")

        self.cap = cv2.VideoCapture(camera_index)

        self.frame = ttk.Frame(root, padding=10)
        self.frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))

        self.canvas = tk.Canvas(self.frame, width=640, height=480)
        self.canvas.grid(row=0, column=0, colspan=4)

        self.label_diameter = ttk.Label(self.frame, text="Çap: ")
        self.label_diameter.grid(row=1, column=0, sticky=tk.W)

        self.diameter_var = tk.StringVar()
        self.diameter_entry = ttk.Entry(self.frame, textvariable=self.diameter_var, state='readonly')
        self.diameter_entry.grid(row=1, column=1, colspan=3, sticky=(tk.W, tk.E))
```

Şekil A.2: diameterMeasurementApp sınıfı.

DiameterMeasurementApp sınıfı çap ölçüm uygulamasının ana yapısını ve işlevselliğini tanımlar. Bu sınıf, bir kamera aracılığıyla alınan görüntüleri işleyerek, görüntüdeki nesnelerin çapını hesaplamak ve kullanıcıya geri bildirim sağlamak için bir kullanıcı arayüzü oluşturur.

diameterMeasurementApp sınıfı, çap ölçüm uygulamasını tanımlar, `_init_` metodu, sınıfın yapıcı metodudur ve sınıf örneği oluşturulduğunda çağırılır.

Root parametresi Tkinter ana pencere nesnesini temsil eder. Camera_index parametresi, kullanılacak kamera aygıtı indeksini belirler

`self.root.title("Çap Ölçüm Uygulaması")` , uygulama penceresi başlığını temsil eder.

`self.cap = cv2.VideoCapture(camera_index)` , OpenCV kullanarak belirtilen indeks ile kamera yakalama nesnesi oluşturulur. Bu nesne, kameradan görüntü almayı sağlar.

```
self.frame = ttk.Frame(root, padding=10)
```

```
self.frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S)) ,
```

Tkinter `ttk.Frame` kullanılarak ana pencere içinde bir çerçeve oluşturulur.

`padding=10` ile çerçevenin etrafına 10 piksel boşluk eklenir. grid yöntemi kullanılarak çerçeve, ana pencere içinde bir ızgara düzeninde yerleştirilir.

```
self.canvas = tk.Canvas(self.frame, width=640, height=480)
```

```
self.canvas.grid(row=0, column=0, columnspan=4)
```

`tk.Canvas` kullanılarak bir tuval oluşturulur. Bu tuval, kameradan alınan görüntüleri görüntülemek için kullanılacaktır. Tuvalin genişliği 640 piksel ve yüksekliği 480 piksel olarak ayarlanır. grid yöntemi ile tuval, çerçevenin içinde yerleştirilir ve dört sütunu kaplar (`columnspan=4`).

```
self.label_diameter = ttk.Label(self.frame, text="Çap: ")
```

```
self.label_diameter.grid(row=1, column=0, sticky=tk.W) self.diameter_var =  
tk.StringVar() self.diameter_entry=ttk.Entry(self.frame,  
textvariable=self.diameter_var, state='readonly') self.diameter_entry.grid(row=1,  
column=1, columnspan=3, sticky=(tk.W, tk.E))
```

ttk.Label kullanılarak "Çap:" metnini içeren bir etiket oluşturulur ve çerçevenin içine yerleştirilir.

tk.StringVar kullanılarak bir değişken oluşturulur. Bu değişken, giriş alanının değerini tutar.

ttk.Entry kullanılarak bir giriş alanı oluşturulur ve *textvariable* parametresi ile *self.diameter_var* değişkenine bağlanır. Bu alan sadece okunabilir (*state='readonly'*) olarak ayarlanır.

Giriş alanı, çerçevenin içinde yerleştirilir ve üç sütunu kaplar (*columnspan=3*).

```
# Hedef çap ve tolerans için girişler
self.target_diameter_label = ttk.Label(self.frame, text="Oluşturulmak istenen boru çapı (mm):")
self.target_diameter_label.grid(row=2, column=0, sticky=tk.W)
self.target_diameter_var = tk.DoubleVar(value=35.0)
self.target_diameter_entry = ttk.Entry(self.frame, textvariable=self.target_diameter_var)
self.target_diameter_entry.grid(row=2, column=1, sticky=(tk.W, tk.E))

self.target_tolerance_label = ttk.Label(self.frame, text="Tolerans (mm):")
self.target_tolerance_label.grid(row=3, column=0, sticky=tk.W)
self.target_tolerance_var = tk.DoubleVar(value=1.0)
self.target_tolerance_entry = ttk.Entry(self.frame, textvariable=self.target_tolerance_var)
self.target_tolerance_entry.grid(row=3, column=1, sticky=(tk.W, tk.E))

# Profil ölçümleri için girişler
self.target_width_label = ttk.Label(self.frame, text="Oluşturulmak istenen profil genişliği (z ekseni) (mm):")
self.target_width_label.grid(row=4, column=0, sticky=tk.W)
self.target_width_var = tk.DoubleVar(value=50.0)
self.target_width_entry = ttk.Entry(self.frame, textvariable=self.target_width_var, state='disabled')
self.target_width_entry.grid(row=4, column=1, sticky=(tk.W, tk.E))

self.target_length_label = ttk.Label(self.frame, text="Oluşturulmak istenen profil uzunluğu (x ekseni) (mm):")
self.target_length_label.grid(row=5, column=0, sticky=tk.W)
self.target_length_var = tk.DoubleVar(value=60.0)
self.target_length_entry = ttk.Entry(self.frame, textvariable=self.target_length_var, state='disabled')
self.target_length_entry.grid(row=5, column=1, sticky=(tk.W, tk.E))
```

Şekil A.3: hedef çap ve tolerans için girişler.

Bu bölümde, hedef çap ve tolerans için kullanıcıdan giriş almak, ayrıca profil ölçümleri için genişlik ve uzunluk girişlerini sağlamak amacıyla etiketler ve giriş alanları oluşturulmuştur. Bu işlevsellik, çap ölçüm uygulamasının kullanıcıdan hedef değerleri ve toleransları almasını sağlar, böylece ölçülen nesnelerin bu değerlere göre kontrol edilmesi mümkün olur.

Hedef çap etiket ve giriş alanı için kullanılan etiketler şu şekilde dir, *ttk.Label* kullanılarak "Oluşturulmak istenen boru çapı (mm):" metnini içeren bir etiket oluşturulur ve çerçevenin ikinci satırında ilk sütuna (*row=2, column=0*) yerleştirilir. *tk.DoubleVar* kullanılarak bir değişken oluşturulur ve bu değişkenin başlangıç değeri

35.0 mm olarak ayarlanır , *ttk.Entry* kullanılarak bir giriş alanı oluşturulur ve *textvariable* parametresi ile *self.target_diameter_var* değişkenine bağlanır. Giriş alanı, çerçevenin ikinci satırında ikinci sütuna (*row=2, column=1*) yerleştirilir ve yatay olarak genişlemesi sağlanır (*sticky=(tk.W, tk.E)*).

Tolerans etiketi ve giriş alanı için kullanılan etiketler şu şekildedir, *ttk.Label* kullanılarak *"Tolerans (mm):"* metnini içeren bir etiket oluşturulur ve çerçevenin üçüncü satırında ilk sütuna (*row=3, column=0*) yerleştirilir. *tk.DoubleVar* kullanılarak bir değişken oluşturulur ve bu değişkenin başlangıç değeri 1.0 mm olarak ayarlanır. *ttk.Entry* kullanılarak bir giriş alanı oluşturulur ve *textvariable* parametresi ile *self.tolerance_var* değişkenine bağlanır. Giriş alanı, çerçevenin üçüncü satırında ikinci sütuna (*row=3, column=1*) yerleştirilir ve yatay olarak genişlemesi sağlanır (*sticky=(tk.W, tk.E)*).

Profil için uzunluk ve genişlik olarak iki faktörde incelendi iki tarafında ölçüm sonuçları alındı, bunlar eksenler yardımı ile z ve x eksenleri aracılığı ile tanıtıldı.

Profil genişliği için giriş alanları yaratılmasını sağlayan kodlar; *ttk.Label* kullanılarak *"Oluşturulmak istenen profil genişliği (z eksen) (mm):"* metnini içeren bir etiket oluşturulur ve çerçevenin dördüncü satırında ilk sütuna (*row=4, column=0*) yerleştirilir. *tk.DoubleVar* ,kullanılarak bir değişken oluşturulur ve bu değişkenin başlangıç değeri 50.0 mm olarak ayarlanır. *ttk.Entry* kullanılarak bir giriş alanı oluşturulur ve *textvariable* parametresi ile *self.target_width_var* değişkenine bağlanır. Bu alan, başlangıçta disabled durumunda (yani, kullanıcı tarafından düzenlenemez) ayarlanır. Giriş alanı, çerçevenin dördüncü satırında ikinci sütuna (*row=4, column=1*) yerleştirilir ve yatay olarak genişlemesi sağlanır (*sticky=(tk.W, tk.E)*).

Profil uzunluğu ve etiket girişleri kodlara göre tanımları şu şekildedir.

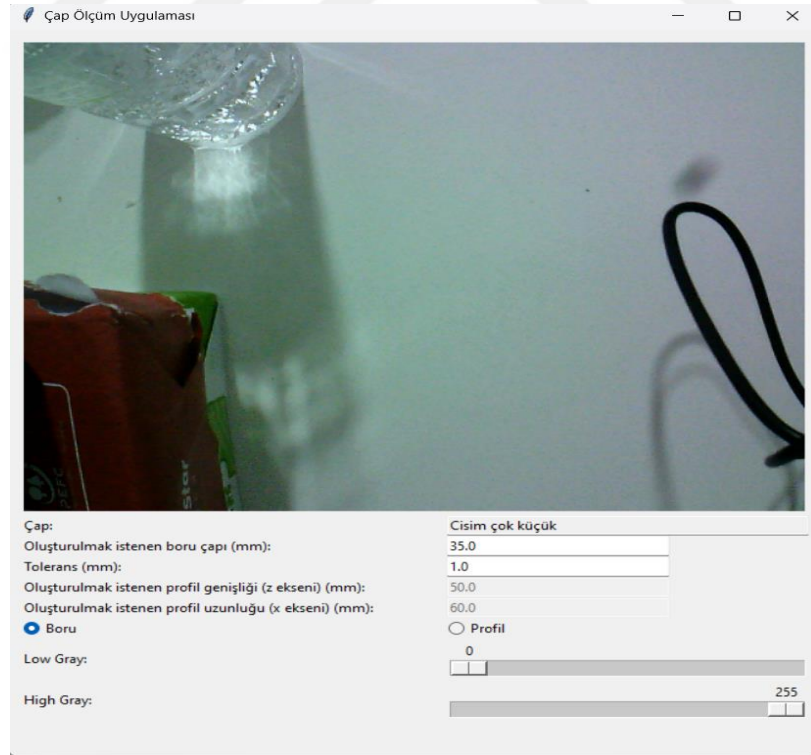
ttk.Label kullanılarak *"Oluşturulmak istenen profil uzunluğu (x eksen) (mm):"* metnini içeren bir etiket oluşturulur ve çerçevenin beşinci satırında ilk sütuna (*row=5, column=0*) yerleştirilir.

tk.DoubleVar kullanılarak bir değişken oluşturulur ve bu değişkenin başlangıç değeri 60.0 mm olarak ayarlanır.

`ttk.Entry` kullanılarak bir giriş alanı oluşturulur ve `textvariable` parametresi ile `self.target_length_var` değişkenine bağlanır. Bu alan da başlangıçta disabled durumunda ayarlanır.

Giriş alanı, çerçevenin beşinci satırında ikinci sütuna (`row=5, column=1`) yerleştirilir ve yatay olarak genişlemesi sağlanır (`sticky=(tk.W, tk.E)`).

Yazılımda kullanıcıdan hedef çap, profil genişliği, uzunluğu ve tolerans değerlerini almak için etiketler ve giriş alanları kullanılmıştır. Bu sayede kullanıcının ihtiyaç duyduğu bilgileri kolayca girilebilmesi sağlanır. Giriş alanlarının başlangıç değerleri, varsayılan veya tipik ölçüm değerlerini yansıtacak şekilde ayarlanmıştır, böylece kullanıcıların hızlı bir şekilde başlaması sağlanmıştır. Çap ve profil ölçümleri arasındaki geçişi kolaylaştırmak için giriş alanları başlangıçta devre dışı bırakılmıştır ve sadece ilgili ölçüm türü seçildiğinde etkinleştirilmiştir. Bu, kullanıcı arayüzünü daha temiz ve kullanımı kolay hale getirir. DoubleVar kullanılarak girişlerin sayısal (ondalıklı) değerler olması sağlanmıştır. Bu, ölçüm ve toleransların hassas bir şekilde girilmesine olanak tanır.



Şekil A.4: Çap ölçüm uygulaması GUI ekranı ve girişleri.

Etiketler ve giriş alanları belirli bir düzen doğrultusunda yerleştirilmiştir. Buda arayüzde kodun okunabilirliği ve anlaşılabilirliği bakımından kolaylık sağlamıştır. Düzen grid komutu ile bileşenlerin pencere içinde belirli bir düzenle yerleştirilmesi sağlanmıştır.

Kodların devamında *tk.StringVar* kullanılarak bir değişken oluşturulur ve başlangıç değeri "boru" olarak ayarlanır. Bu değişken, seçilen ölçüm türünü saklar.

ttk.Radiobutton kullanılarak "Boru" ve "Profil" seçenekleri için radyo düğmeleri oluşturulur. Her iki düğme *self.measurement_type* değişkenine bağlanır ve bu değişkenin değeri ilgili seçime göre "boru" veya "profil" olarak güncellenir.

Radyo düğmeleri *command* parametresi *self.update_measurement_type* metodu ile çağrılır, bu da seçim değiştiğinde arayüzde gerekli güncellemeleri yapar.

Radyo düğmeleri, çerçevenin altıncı satırında yan yana yerleştirilir (*row=6, column=0* ve *row=6, column=1*). Bu şekilde hangi seçenek seçilir ise diğer seçeneğin de aktif olmasını sağlamaktadır. Boru seçilir ise oluşturulmak istenen boru çapı ve tolerans değerleri girilmesi gerekmekte, profil seçeneği seçilir ise profil genişliği, profil uzunluğu ve tolerans olarak girilmesi gerekmektedir. Seçeneğe göre diğer seçenekler de aktif olmaktadır.

Çap:
Oluşturulmak istenen boru çapı (mm): 35.0
Tolerans (mm): 1.0
Oluşturulmak istenen profil genişliği (z eksen) (mm): 50.0
Oluşturulmak istenen profil uzunluğu (x eksen) (mm): 60.0
☐ Boru
☒ Profil
Low Gray: 0
High Gray: 255

Şekil A.5: Profil seçeneği seçildiği zaman.

Şekil A.6: Boru seçeneği seçildiği zaman.

Yukarıdaki şekillerde boru ve profil seçenekleri hangisi seçilir ise diğeri de aktif olmaktadır. Bunun nedeni kullanıcının yanlış bir değer ile yanlış ölçüm sonuçları engellenmek istenmesidir.

```
# Seçim için radyo düğmeleri
self.measurement_type = tk.StringVar(value="boru")
self.boru_radiobutton = ttk.Radiobutton(self.frame, text="Boru", variable=self.measurement_type, value="boru", command=self.update_measurement_type)
self.boru_radiobutton.grid(row=6, column=0, sticky=tk.W)
self.profil_radiobutton = ttk.Radiobutton(self.frame, text="Profil", variable=self.measurement_type, value="profil", command=self.update_measurement_type)
self.profil_radiobutton.grid(row=6, column=1, sticky=tk.W)

# Gri ton aralıkları için kaydırma çubukları
self.low_gray_var = tk.IntVar(value=0)
self.high_gray_var = tk.IntVar(value=255)

self.low_gray_label = ttk.Label(self.frame, text="Low Gray:")
self.low_gray_label.grid(row=7, column=0, sticky=tk.W)
self.low_gray_scale = tk.Scale(self.frame, from_=0, to=255, orient=tk.HORIZONTAL, variable=self.low_gray_var)
self.low_gray_scale.grid(row=7, column=1, columnspan=3, sticky=(tk.W, tk.E))

self.high_gray_label = ttk.Label(self.frame, text="High Gray:")
self.high_gray_label.grid(row=8, column=0, sticky=tk.W)
self.high_gray_scale = tk.Scale(self.frame, from_=0, to=255, orient=tk.HORIZONTAL, variable=self.high_gray_var)
self.high_gray_scale.grid(row=8, column=1, columnspan=3, sticky=(tk.W, tk.E))

# Sonuçlar için etiket
self.result_label = ttk.Label(self.frame, text="", font=("Helvetica", 14))
self.result_label.grid(row=9, column=0, columnspan=4, sticky=tk.W)

self.update_image()
```

Şekil A.7: Radyo düğmeleri, gri ton aralıkları ve sonuçlar etiketleri kodları.

Boru ve profilleri belirli bir renk skalasına göre arayüz üzerinden ayarlanmaktadır, ortam ışık ve birçok faktör etken olduğundan dolayı çubuklar ile ekrandan anlık olarak bakılıp yapılmıştır. Ve bu skalaya göre Şekil A.8’de görüldüğü gibi çerçeve içine alır.



Şekil A.8: ölçüm esnasında çerçeve içine alınmış profil.

Gri ton değişiklikleri, *tk.IntVar* kullanılarak iki değişken oluşturulur: *self.low_gray_var* ve *self.high_gray_var*. Başlangıç değerleri sırasıyla 0 ve 255 olarak ayarlanır. Bu değişkenler, gri ton aralıklarını ayarlamak için kullanılmıştır.

Kaydırma çubukları ve etikler sayesinde ayarlanabilir ve anlaşılabilir olması sağlanmıştır. *ttk.Label* kullanılarak "Low Gray:" ve "High Gray:" metinlerini içeren etiketler oluşturulur ve çerçevenin yedinci ve sekizinci satırında ilk sütuna yerleştirilir (*row=7, column=0* ve *row=8, column=0*).

tk.Scale kullanılarak kaydırma çubukları oluşturulur ve *from_=0, to=255* parametreleri ile 0'dan 255'e kadar bir aralıkta ayarlanır. *orient=tk.HORIZONTAL* parametresi ile yatay olarak yerleştirilirler. Kaydırma çubukları, *variable* parametresi ile ilgili değişkenlere (*self.low_gray_var* ve *self.high_gray_var*) bağlanmıştır. Kaydırma çubukları, çerçevenin yedinci ve sekizinci satırında ikinci sütuna yerleştirilir ve yatay olarak genişlemesi sağlanır (*row=7, column=1* ve *row=8, column=1, sticky=(tk.W, tk.E)*).

Sonuç ve etiketler için ise ilk olarak *ttk.Label* kullanılarak sonuçları göstermek için boş bir etiket oluşturulur. Başlangıçta etikette herhangi bir metin yoktur (*text=""*).

font=("Helvetica", 14) parametresi ile etiketin yazı tipi ve boyutu ayarlanır.

Etiket, çerçevenin dokuzuncu satırında dört sütun boyunca genişletilerek yerleştirilir (*row=9, column=0, columnspan=4*) ve sola hizalanır (*sticky=tk.W*).

self.update_image() metodu çağrılarak görüntülerin güncellenmesi sağlanır. Bu metod, kameradan alınan görüntüyü işleyerek kullanıcıya gösterilmesini sağlamıştır. Bu sayede kullanıcı anlık olarak geri bildirim almıştır.

```
def update_measurement_type(self):  
    if self.measurement_type.get() == "boru":  
        self.target_diameter_entry.config(state='normal')  
        self.target_width_entry.config(state='disabled')  
        self.target_length_entry.config(state='disabled')  
    elif self.measurement_type.get() == "profil":  
        self.target_diameter_entry.config(state='disabled')  
        self.target_width_entry.config(state='normal')  
        self.target_length_entry.config(state='normal')
```

Şekil A.9: boru-profil seçim deaktif kodları.

update_measurement_type sınıfı *self* parametresi ile sınıfın bir üyesi olarak tanımlanmıştır. Bu, sınıfın diğer üyelerine ve bileşenlerine erişim sağlar. Kullanıcı arayüzünde ölçüm türünün değiştirilmesi durumunda gerekli bileşenlerin etkinleştirilmesi veya devre dışı bırakılması işlevini yerine getirir. Ölçüm türü "boru" veya "profil" olarak seçildiğinde, ilgili giriş alanları etkinleştirilir veya devre dışı bırakılır.

self.measurement_type.get() ifadesi, şu anda seçili olan ölçüm türünü alır. *self.measurement_type*, radyo düğmeleri tarafından belirlenen bir *tk.StringVar* değişkenidir. *if* ve *elif* yapıları kullanılarak, ölçüm türünün "boru" veya "profil" olup olmadığı kontrol edilir.

Boru ölçümü seçildiğinde,

self.target_diameter_entry.config(state='normal'): Boru çapı için giriş alanı etkinleştirilir. *self.target_width_entry.config(state='disabled')*: Profil genişliği için giriş alanı devre dışı bırakılır.

self.target_length_entry.config(state='disabled'): Profil uzunluğu için giriş alanı devre dışı bırakılır.

Profil ölçümü seçildiğinde,

self.target_diameter_entry.config(state='disabled'): Boru çapı için giriş alanı devre dışı bırakılır. *self.target_width_entry.config(state='normal')*: Profil genişliği için giriş alanı etkinleştirilir.

self.target_length_entry.config(state='normal'): Profil uzunluğu için giriş alanı etkinleştirilir.

Kodların bu şekilde yazılmasının sebebi gereksiz girişlerin ve yanlış alanların önüne geçilmek istenmesidir. Örneğin, profil ölçümü yapılırken boru çapı girişinin etkin olması kafa karışıklığına yol açabilir. Bu tür hataları önlemek için yalnızca ilgili alanlar etkinleştirilir.

```
def update_image(self):  
    ret, frame = self.cap.read()
```

Şekil A.10: update image kütüphanesi.

Update_image sınıfında *self.cap.read()* metodu, kameradan bir kare yakalar. Ret değişkeni, kare yakalamanın başarılı olup olmadığını belirtir, frame ise yakalanan görüntüdür.

```
}  
if ret:  
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  
  
    low_gray = self.low_gray_var.get()  
    high_gray = self.high_gray_var.get()  
  
    # Gri ton aralığına göre eşikleme  
    _, thresholded = cv2.threshold(gray_frame, low_gray, high_gray, cv2.THRESH_BINARY_INV)
```

Şekil A.11: gri tonlama ve eşikleme.

Resimdeki kod satırlarında Görüntü gri tonlamaya dönüştürülür. Bu, kontur algılamayı ve eşiklemeyi kolaylaştırılmıştır. Kullanıcının belirlediği gri ton aralıklarına göre eşikleme yapılır. Eşiklenen görüntüde belirli gri tonlarının dışındaki tüm pikseller beyaza çevrilir.

```

# Konturları bulma
contours, _ = cv2.findContours(thresholded, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

if contours:
    largest_contour = max(contours, key=cv2.contourArea)

    if self.measurement_type.get() == "boru":
        (x, y), radius = cv2.minEnclosingCircle(largest_contour)
        center = (int(x), int(y))
        radius = float(radius)

        if radius > 10: # Minimum bir boyut filtresi uygulayın
            diameter = radius * 2 * SCALE_FACTOR
            target_diameter = self.target_diameter_var.get()
            tolerance = self.tolerance_var.get()

            lower_bound = target_diameter - tolerance
            upper_bound = target_diameter + tolerance

            if lower_bound <= diameter <= upper_bound:
                message = f"Ölçü: OK ((diameter:.2f) mm)"
                color = "green"
            else:
                if diameter < lower_bound:
                    deviation = lower_bound - diameter
                    message = f"Ölçü: Tolerans Dışı ((diameter:.2f) mm, (deviation:.2f) mm küçük)"
                else:
                    deviation = diameter - upper_bound
                    message = f"Ölçü: Tolerans Dışı ((diameter:.2f) mm, (deviation:.2f) mm büyük)"
                color = "red"

            self.diameter_var.set(f"((diameter:.2f) mm)")
            self.result_label.config(text=message, foreground=color)

```

Şekil A.12: konturları bulma.

Eşiklenen görüntüdeki konturlar bulunur. *cv2.RETR_EXTERNAL*, sadece dış konturları algılar. Bulunan konturlar arasından en büyük olanı seçilir. Eğer *self.measurement_type* "boru" ise: En büyük konturu kapsayan en küçük çember bulunur. Çemberin merkezi ve yarıçapı hesaplanır. Çap, yarıçapın iki katı olarak hesaplanır ve ölçek faktörü ile çarpılır. Hedef çap ve tolerans alınır ve çap kontrol edilir.

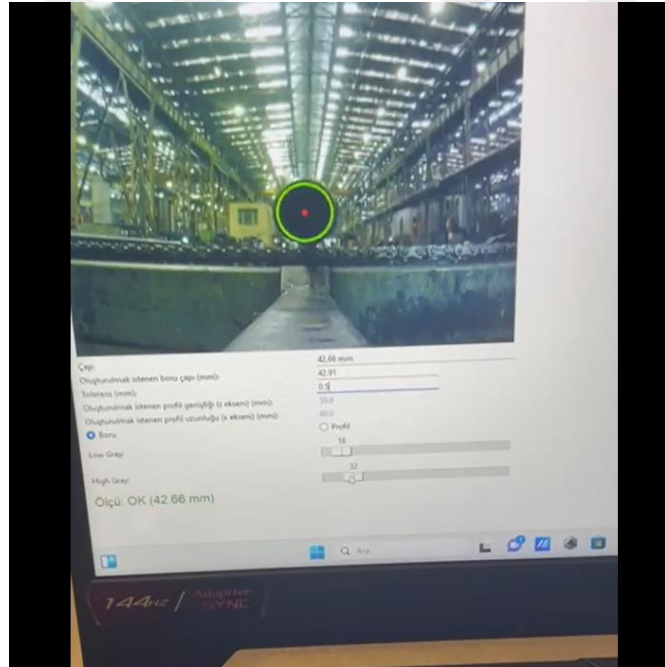
Eğer *self.measurement_type* "profil" ise: En büyük konturu kapsayan dikdörtgen bulunur. Dikdörtgenin genişliği ve uzunluğu hesaplanır ve ölçek faktörü ile çarpılır.

Genişlik ve uzunluk kontrol edilerek hedef değerlerle karşılaştırılır. Görüntü BGR'den RGB'ye dönüştürülür, ardından PIL ve ImageTk kullanılarak tkinter canvasında görüntülenir. 10 milisaniye sonra *update_image* metodunun tekrar çalışmasını sağlar.

EK B



Şekil B.1: Boru denemesi1 kumpas ölçümü.



Şekil B.2: Boru denemesi1 kamera ölçümü.



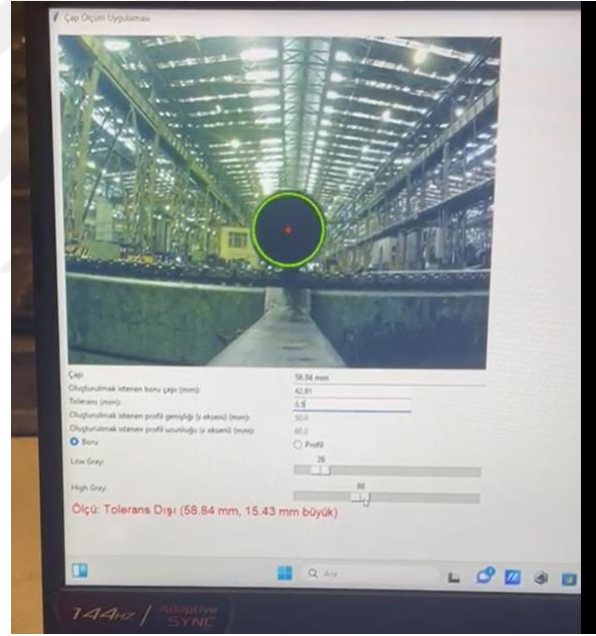
Şekil B.3: Kamera ve boru konumlanması.



Şekil B.4: Kamera ve boru arasındaki mesafe.



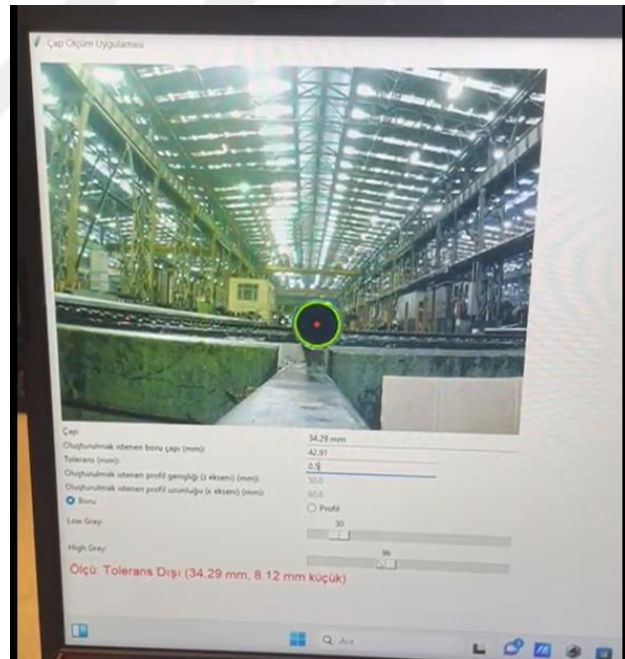
Şekil B.5: Boru denemesi 2 kumpas ölçümü.



Şekil B.6: Boru denemesi 2 kamera ölçümü.



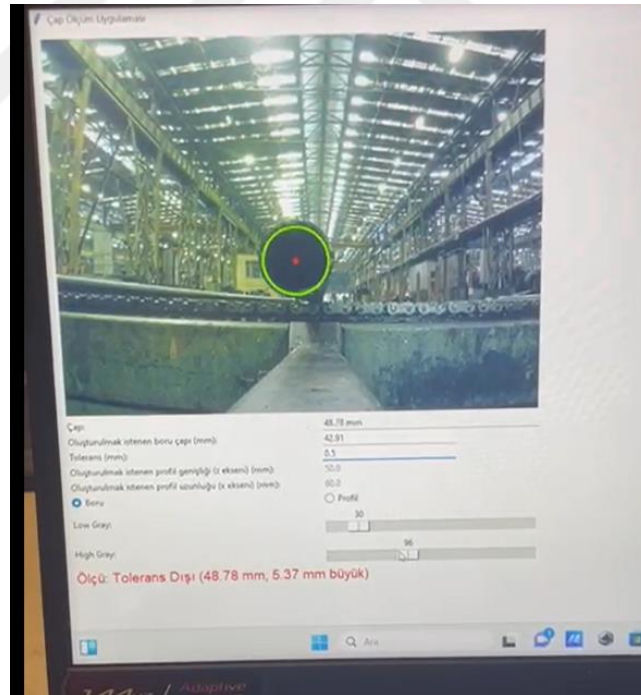
Şekil B.7: Boru denemesi 3 kumpas ölçümü.



Şekil B.8: Boru denemesi 3 kamera ölçümü.



Şekil B.9: Boru denemesi 4 kumpas ölçümü.



Şekil B.10: Boru denemesi 4 kamera ölçümü.



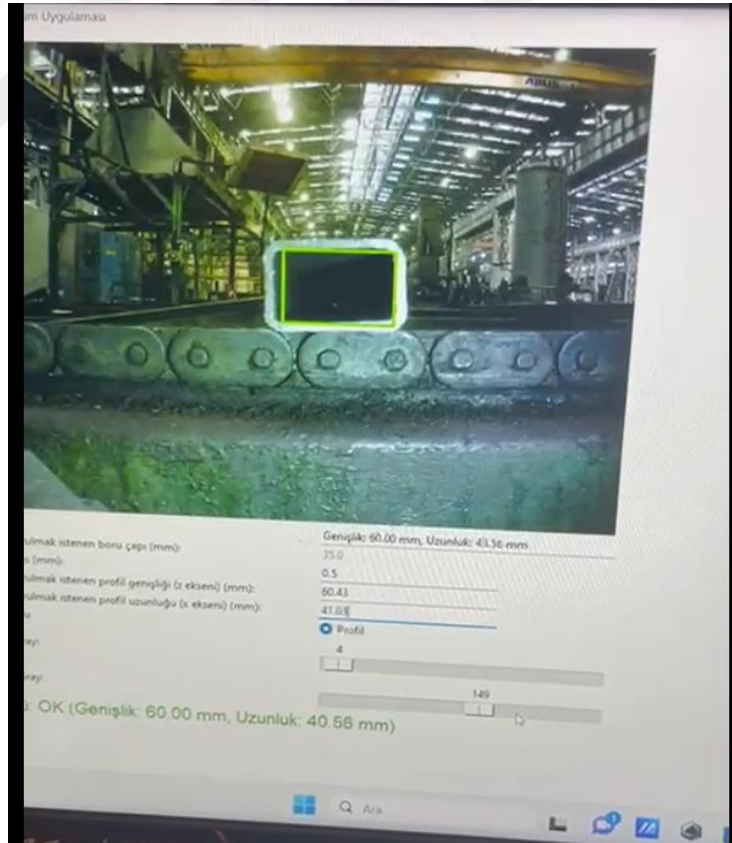
Şekil B.11: Profil denemesi 1 kısa kenar kumpas ölçümü.



Şekil B.12: Profil denemesi 1 uzun kenar kumpas ölçümü.



Şekil B.13: Profil denemesi 1 kamera uzaklığı.



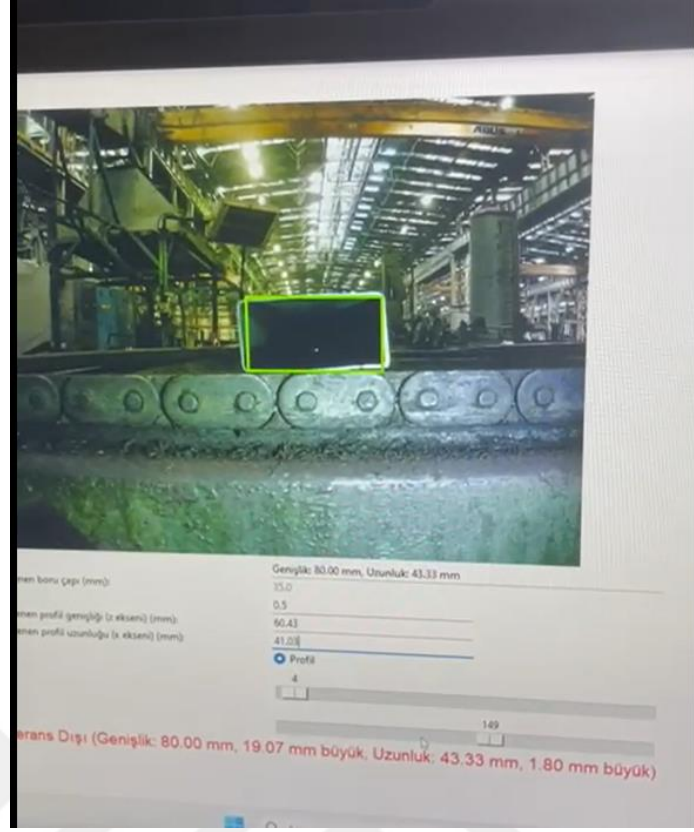
Şekil B.14: Profil denemesi 1 kamera ölçümü.



Şekil B.15: Profil denemesi 2 uzun kenar kumpas ölçümü.



Şekil B.16: Profil denemesi 2 kısa kenar kumpas ölçümü.



Şekil B.17: Profil denemesi 2 kamera ölçümü.