T.C.
MANİSA CELAL BAYAR ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI
KONTROL VE KUMANDA SİSTEMLERİ BİLİM DALI

DOKUNMATİK EKRANLI CİHAZLAR İÇİN GÖRÜNTÜ İŞLEMEYE DAYALI ROBOTİK TEST OTOMASYON SİSTEMİ GELİŞTİRİLMESİ

YİĞİT KARABULUT

Danışman
Doç. Dr. SEZAİ TAŞKIN

MANİSA–2016

Yiğit KARABULUT	**DOKUNMATİK EKRANLI CİHAZLAR İÇİN GÖRÜNTÜ İŞLEMEYE DAYALI ROBOTİK TEST OTOMASYON SİSTEMİ GELİŞTİRİLMESİ**	**2016**

TEZ ONAYI

Yiğit KARABULUT tarafından hazırlanan **"Dokunmatik Ekranlı Cihazlar İçin Görüntü İşlemeye Dayalı Robotik Test Otomasyon Sistemi Geliştirilmesi"**adlı tez çalışması 09/12/2016 tarihinde aşağıdaki jüri üyeleri önünde Manisa Celal Bayar Üniversitesi Fen Bilimleri Enstitüsü **Elektrik Elektronik Mühendisliği Anabilim Dalı**'nda **YÜKSEK LİSANS** olarak savunulmuş ve **oybirliği** ile başarılı olarak kabul edilmiştir.

Danışman Doç. Dr. Sezai Taşkın
Manisa Celal Bayar Üniversitesi

Jüri Üyesi Yrd. Doç. Dr. Özkan Akın
Ege Üniversitesi

Jüri Üyesi Yrd. Doç. Dr. Hayati Mamur
Manisa Celal Bayar Üniversitesi

TAAHHÜTNAME

Bu tezin Manisa Celal Bayar Üniversitesi Mühendislik Fakültesi Elektrik Elektronik Mühendisliği Bölümü'nde, akademik ve etik kurallara uygun olarak yazıldığını ve kullanılan tüm literatür bilgilerinin referans gösterilerek tezde yer aldığını beyan ederim.

YİĞİT KARABULUT

İÇİNDEKİLER

İÇİNDEKİLER

SİMGELER VE KISALTMALAR DİZİNİ

OCR	Optical Character Recognization
OCV	Optical Character Verification
ADB	Android Debug Bridge
PID	Proportional-Integral-Derivative
SIM	Subscriber Identity Module
XML	Extensible Markup Language
MBT	Model Based Test
NASA	National Aeronautics and Space Administration
MRI	Magnetic Resonance Imaging
PET	Positron Emission Tomography
FOV	Field of View
RGB	Red-Green-Blue
CMYK	Cyan Magenta Yellow Key
JPEG	Joint Photographic Experts Group
CCD	Charge Coupled Device
FPS	Frame Per Second
XLSX	XML Spreadsheet

ŞEKİLLER DİZİNİ

Sayfa

ŞEKİLLER DİZİNİ

TABLO DİZİNİ

Sayfa

TEŞEKKÜR

Bu tezin yürütülmesi süresince, bilgi ve deneyimleri ile yol gösteren danışman hocam Doç. Dr. Sezai TAŞKIN'a, bu tez fikrinin ortaya çıkmasındaki katkılarından ve bu çalışmanın bir SAN-TEZ proje çalışması olarak sunulması aşamasına kadarki değerli katkılarından dolayı Vestel Digital Ar-Ge Merkezi'nden Bilgisayar Mühendisi İlker YILDIRIM'a, çalışmalarım sırasında destek olan mesai arkadaşlarıma, beni daima destekleyen aileme yürekten teşekkür ederim.

Yiğit KARABULUT
Manisa, 2016

ÖZET

Yüksek Lisans Tezi

Yiğit KARABULUT

Manisa Celal Bayar Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik Elektronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Sezai TAŞKIN

Tüketici elektroniğine geniş bir açıdan baktığımız zaman, tablet ve akıllı telefonlar gibi dokunmatik ekrana sahip cihazlara olan talebin artışı, bu cihazlara ait testlerin hızlı ve güvenli bir şekilde yapılması ihtiyacını doğurmuştur. Mevcut test platformlarının yapısında insan faktörü önemli bir rol oynadığı için çalışma sırasında birçok hata oluşabilmektedir. Bu sorunları gidermek için test platformlarının, test otomasyon sistemlerine dönüştürülmesine yönelik çalışmalar yapılmıştır. Robotlu test otomasyon sistemleri günümüzde elektronik cihazların test işlemlerinde önemli bir rol oynamaktadır.

Bu çalışmada, delta robot temelli bir test otomasyon sistemi tasarlanmıştır. Tasarlanan delta robot için donanım elemanları olarak LEGO® Mindstorms® NXT 2.0 servo motorları ve kontrolörü kullanılmıştır. Yazılım olarak ise MATLAB® R2016b kullanılmıştır. Robota ait elde edilen kinematik denklemler MATLAB® üzerinde çalıştırılmış ve robot açıları ile kartezyen koordinat düzlemi arasındaki dönüşüm sağlanmıştır. Sistemin kararlı bir şekilde çalışabilmesi için bir kamera sisteme eklenmiştir. Yapay görme algoritması olarak NCC (normalize çapraz korelasyon) algoritması kullanılarak robotun dokunmatik ekran üzerinden aldığı görüntüler şablon görüntülerle karşılaştırılmış ve sisteme geri besleme olarak verilmiştir. Tasarlanan robotik tabanlı test otomasyon sisteminde delta tipi robot kullanılması sayesinde farklı boyutlardaki dokunmatik ekranlı cihazlar için kullanılabilir özellikte olması sağlanmıştır. Robot, görüntü işleme tabanlı hareket yönlendirme yaptığından farklı işletim sistemine sahip cihazlar için performans, stress vb. gibi farklı test ihtiyaçları gerektiren uygulamaların yapılabilmesine olanak sağlamaktadır. Tasarlanan robot sistemindeki hareket kontrol elemanları olarak endüstriyel ürünler kullanılması durumunda çok daha hızlı ve profesyonel çözümler ortaya konulması mümkündür.

Anahtar Kelimeler: Dokunmatik Ekranlı Cihaz, Tüketici Elektroniği, Delta Robot, Robot Kinematiği, Yapay Görme, Görüntü İşleme, Test Otomasyon Sistemi

2016, 80 sayfa

ABSTRACT

M.Sc. Thesis

Development of Image Processing Based Robotic Test Automation System for Touch Screen Devices

Yiğit KARABULUT

Celal Bayar University
Graduate School of Applied and Natural Sciences
Department of Electrical and Electronics Engineering

Supervisor: Assoc. Prof. Dr. Sezai TAŞKIN

When we have a wide perspective on awareness of consumer electronics, increasing the demand of touch screen devices such as tablet, smart phone etc. has created a need for making the test of these devices quickly and reliably. The human factor plays an important role on structure of existing test platforms, and creates many errors during operation. So as to overcome these problems, some solutions have been developed. Especially, robotic based test automation systems are now playing an important role on testing electronic devices.

In this study a delta robot based test automation system is designed to perform for touch screen device testing. In order to design delta robot, LEGO® Mindstorms® NXT 2.0 servo motors and their controller are used as hardware elements. The designed delta robot system is controlled via MATLAB® R2016b development environment. Kinematic equations of the robot are run in MATLAB® and the transformation between angles of the robot arms and cartesian coordinate plane of end effector is provided by the equations. The images taken by the camera from touch screen are compared with the template images and given as feedback to the system by using NCC (normalized cross correlation) as computer vision algorithm. By capturing images via camera that is taken part in the system. Hence, the whole system works more stable in this condition. The designed delta robot can be used in different size of touch screen devices. Moreover, the robot can be used for different test methods including performance, stress etc. If professional products had been used as the motion control elements in the designed robot system, it woluld be possible to introduce much faster and more stable industrial solutions.

Keywords: Touch Screen Device, Consumer Electronics, Delta Robot, Robotics Kinematic, Computer Vision, Image Processing, Test Automation System

2016, 80 pages

1. GİRİŞ

Test sistemleri, bir sistemin belirlenmiş gereksinimleri karşıladığının doğrulanması işlemini yapan cihazlardır. Manuel ve otomatik test sistemleri olmak üzere iki versiyonu mevcuttur. Bu sistemler doğruluk, hız ve maliyet yönünden önemli farklılıklara sahiptir. Bu farklılıkların en önemli sebebi manuel sistemlerin kişilerin performansına dayalı olmasıdır [1, 2].

Günümüzde otomatik test sistemleri yüksek hız, doğruluk ve performans açısından tercih edilmektedir. Bu sistemlerde manuel test sistemlerinde bulunan testi gerçekleştiren operatörü taklit eden bir robot bulunmaktadır. Bu nedenle otomatik test platformunun doğruluk yüzdesi manuel test sistemlerine göre oldukça artmaktadır. Otomatik test platforlarını manuel test sistemlerinden ayıran bir diğer özellik ise sistem içindeki dahili bir kaynaktan gelen bir geri beslemeye göre çalışmasıdır. Bu geri besleme yöntemleri test sırasında doğrudan cihaz ile kurulabilecek bir iletişim bağlantısıyla olduğu gibi tablet veya cep telefonu gibi sistemlerin testlerinde görüntü işleme yönteminin kullanılması ve insanlı testlerde olduğu gibi ekran üzerinden alınan görüntülerin değerlendirilmesi ile yapılabilmektedir [3]. Test sistemi kullanıcıyı birebir taklit ederek testin başarısını arttıracağı gibi farklı kullanıcı profillerine ait özellikleri de (yaş, cinsiyet, hastalık) testlerde yansıtabilmektedir. Bu sayede farklı insan profillerine ait testlerin de aynı test platformunda gerçekleştirilmesi sağlanmaktadır [4].

Otomatik test platformlarında ağırlıklı olarak endüstriyel robotlar kullanılmaktadır. Bu konuda kontrolü diğer manipülatörlere göre daha basit olan kartezyen robotlar daha çok tercih edilmektedir. Kartezyen robotlardan farklı olarak test otomasyon sistemlerinde 6 serbestlik dereceli manipülatörler de kullanılabilmektedir. Bu robotlar erişim uzaylarının kartezyen robotlara göre daha küçük olmasına ve maliyetlerinin yüksek olmasına rağmen, hareket kabiliyetlerinin yüksek olması ve yaygın olarak kullanılmaları sebebiyle tercih edilebilmektedir. Bir diğer çözüm ise hareket uzayı daha dar olmasına karşın hız yönünden daha yüksek performanslı delta robotların kullanılmasıdır. Delta robotların test işlemleri gibi özellikle hız ve hassasiyet gerektiren işlemlerde oldukça başarılı olduğu görülmüştür [5].

Görüntü işleme teknolojisi hem endüstride hem de akademik çevrelerde çeşitli disiplinler arası çalışmalarda oldukça yaygın olarak kullanılmaktadır. İşlemcilerin güçlerinin hızla artmasıyla daha karmaşık algoritmalar çalıştırılabilmekte ve daha doğru çıktılar alınabilmektedir. Test sistemleri ile görüntü işleme teknolojisinin birleştirilmesi ve sisteme verdiği geri beslemeler diğer yöntemlere göre daha doğru sonuçlar vermektedir. Bu konuda optik karakter tanıma (OCR), optik karakter doğrulama (OCV) ve yapısal benzerlik [6] gibi metotların yanında şablon tanıma gibi yapısal tanıma algoritmalarıda kullanılmaktadır.

Literatürde elektronik cihazlar üzerinde yapılan testler belirli katmanlara ayrılmıştır. En önemli test türlerini performans testleri, fonksiyonel testler, yük ve stres testleri oluşturmaktadır [7, 8]. Bu test grupları elektronik cihazların son kullanıcı ile olan ilişkisini ölçmektedir. Bunun yanında güvenlik, uyumluluk gibi özellikleri de ölçecek testler de sıkça kullanılmaktadır.

Literatürde test otomasyon sistemleri ile ilgili yapılmış çalışmalara bakılırsa; özellikle cep telefonu ve tabletlerde bulunan dokunmatik ekranların yaygınlaşması ile performans ve uyumluluk testleri büyük önem kazanmıştır. Elbaum ve Diep tarafından yapılan çalışmada dokunmatik ekran testleri için hazırlanan kullanıcı profillerine dayanan test yazılımları incelenmiştir [9]. Yapılacak performans testleri için gerekli deney sayısının azaltılması için olasılıklı kullanıcı modelleri (Probabilistic User Models) Brooks ve Memon tarafından özetlenmiştir [10]. Performans testleri için gerekli işlem sayısının azaltılması test sırasında oluşabilecek hataları minimize ettiği gibi zaman açısından önemli bir avantaj sağlamaktadır. Çalışmada uygulanan algoritmalar için Event-Flow-Graph temelli modeller geliştirilmiştir. Fakat mevcut olayları (Event) temsil etmede bu sistem yetersiz kalmaktadır. Çalışmada Event-Flow-Graph temelli model yerine Markov modellerin uygulamalarda kullanımı önerilmiştir ve başarı yüzdesi verilmiştir. Cep telefonu ve tabletler üzerinde yapılan performans testlerinin, mekanik bir otomasyon sistemi ile gerçekleştirilmesi ise [11, 12] numaralı kaynaklarda irdelenmiştir. Bir kartezyen robot aracılığı ile verilen komutlara göre telefon üzerinde robotik bir kol hareket etmektedir. Bunun yanında telefon ile kullanıcı test sisteminin haberleşmesi Android Debug Bridge (ADB) üzerinden sağlanmaktadır. Bu sayede sistem kapalı çevrim çalışabilmesi için geri besleme almaktadır. ADB üzerinden sadece belirli işlemler için yanıt gelmesi ve bir hata durumunda cevap

alınamaması ADB kullanımının sakıncalı olabileceğini göstermektedir. Çalışmada ayrıca 3 tür kullanıcı profili üzerinde durulmuştur: (i)standart kullanıcı, (ii) akıllı kullanıcı ve (iii) iş amaçlı kullanım. Daha önceki veri setlerine dayanılarak otomatik olarak oluşturulmuş kullanıcı profilleri birçok farklı test için kullanılmıştır. Robotik tabanlı uzaktan kontrollü bir cep telefonu test istasyonu ile ilgili çalışmalar [13,14] de verilmiştir. Ekran üzerinde işlem yapılabilmesi için sistemde robotik bir kol ve kullanıcı parmağı yerine geçen bir aparat kullanılmıştır. Test senaryoları belli yerlerdeki ikonlar ve konumlara göre ayarlanmış olup bunlara robotik parmağın teması ile etkileşim olmaktadır. Çalışmada optik karakter tanıma (Optical Character Recognization - OCR) ve görüntü tanıma teknikleri (Image Recognition Techniques) kullanılmıştır. Ayrıca basit performans testlerinde ortaya çıkan gecikme süresinin (Latency Time) ölçülmesi ile ortaya çıkan sonuçların analizi üzerine çalışma [15] de verilmiştir. Uygulamalara ait gecikme zamanının ölçülmesi, ekran üzerinden bağımsız olarak alınan görüntülerle sağlanmıştır.

Bu çalışmada mevcut test otomasyon sistemlerine alternatif olabilecek görüntü işlemeye dayalı bir robotik test otomasyon sistemi ortaya konulmuştur. Tasarlanacak test ekipmanı kullanıcının dokunmatik ekran üzerinde yapacağı işlemleri birebir taklit edecek bir robotik sistemden oluşmaktadır. El hareketlerinin taklidi için bir delta robot tasarlanmıştır. Robota ait motor ve kontrolör donanımları için NXT Mindstorm® tabanlı ürünler kullanılmıştır. MATLAB® R2016b tabanlı olarak kontrol edilen sistemde motor kontrolü için PID kontrolör kullanılmaktadır. Delta robota ait kinematik denklemler MATLAB® ortamında hesaplanmaktadır. MATLAB® üzerinde bulunan "Image acquisition toolbox" platformuna bağlı sistem ile kamera üzerinden gelen görüntüler alınmakta ve uygun bir şekilde işlenmektedir.

Bölüm 2'de elektronik cihazlar üzerinde yapılabilecek testler ve bunlara ait özellikler sırasıyla verilmiştir. Özellikle son kullanıcıya ait testler üzerinde durulmuştur. Örnek test senaryo türleri ve nasıl çalıştığına ait bilgiler verilmiştir. Bölüm 3'de mevcut sistemlerde kullanılacak robotik sistem örnekleri ile delta robotlara ait genel bilgiler verilmiş, kinematik denklemleri ve sistem tasarımı üzerinde durulmuştur. Bölüm 4'de genel görüntü işleme metotlarından bahsedilmiş ve bu çalışmada kullanılan algoritmalara ait bilgiler verilmiştir. Bölüm 5'te kurulan test düzeneği hakkında bilgi verilmiş, test düzeneğinde kullanılan malzemeler tanıtılmış

ve uygulanan kontrol yöntemi açıklanmıştır. Bölüm 6'da ise test sisteminden elde edilen veriler analiz edilmiş ve konu hakkında ileride yapılabilecek çalışmalar önerilmiştir.

2. TEST

2.1 Giriş

Cep telefonu ve tabletlere yönelik talepler hızla artarken, bu konuda çalışan tüm firmalar rakiplerine göre daha performanslı ve fonksiyon sayısı daha yüksek cihazlar üretmek için büyük bir çaba göstermektedir. Tüketici tarafından bakıldığında ise kullanıcılar her zaman alacakları ürünlerin olabildiğince hızlı ve basit olmasını istemektedirler. Cihaz üzerinde bulacakları tek bir sorun veya kullanıma yönelik bir engel satılan cihaza olacak talepleri hızla düşürmektedir. Eğer tüketiciler alacakları ürün üzerinde tam anlamıyla bir memnuniyet duymazlarsa, hızlı bir şekilde rakip firmanın ürünlerine geçmektedirler. Bu durum üreticileri tüketiciye sunacakları cihazlar üzerinde ciddi testler yapmaya zorlamaktadır [16].

Geleneksel test yöntemlerine göre, mobil cihazlar üzerindeki testler belirli test senaryoları ve teknikleri gerektirmektedir. Geleneksel test yöntemleri ile yapılan testler kişisel bilgisayarlar üzerinde olduğu gibi belirli bir test ortamında olmakta ve önceden tanımlı verilere göre yapılmaktadır. Birçok sayıda ve çeşitte üretilen mobil teknolojiler, platformlar ve cihazların bulunması tam anlamıyla yeterli bir test stratejisinin geliştirilmesine engel olmaktadır. Mobil cihazların birçok farklı modeli ve çeşidinin olması önceden yapılan tanımlamaları geçersiz kılabilmektedir.

2.2 Test Katmanları

Mobil uygulamalar için çeşitli test katmanları sırasıyla Şekil 2.1'de gösterilmiştir.

Şekil 2.1 Test katmanlarına ait blok diyagram

- **Birim Testi**: Birim testleri fonksiyonel ve güvenilirlik testlerinden oluşmaktadır. Test senaryoları genellikle cihazın veya uygulamanın ilk çıktığı anda oluşturulur.

Bu testin genel amacı mobil uygulama ve cihazlarda olabildiğince fazla hatanın bulunabilmesidir. Birim testinin bir diğer adı da komponent testi olarak geçmektedir [17].

- **Entegrasyon Testi**: Entegrasyon testi elektronik cihaz içindeki modüllerin bir arada veya tek başlarına fonksiyonel olarak test edilmesidir. Cihaz içindeki komponentlerin yazılım ile uyumlu çalışıp çalışmadığının değerlendirmesini yapmaktadır. Birim testlerinden sonra uygulanmaktadır. Cihazlarda bulunan aktivitelerin birbirleri ile uyumları bu duruma örnek verilebilir.

- **Sistem Testi**: Sistem testleri entegrasyon testlerini geçmiş ve tüm gerekli komponentleri ile yazılımların uygun bir şekilde çalıştığı cihazlara uygulanmaktadır. Sistem tüm elemanları ile birlikte test edilmektedir. Bu durumda sisteme kara kutu (Black Box) testi uygulanır, sistemin iç yapısına ait herhangi bir durum göz önüne alınmaz, sadece giriş ve uygun çıkış değerleri incelenir.

- **Kullanıcı Arayüzü Testi**: Bu test cihazın kontrolü ve etkileşimini yeterli düzeyde olup olmadığını test etmektedir. Testlerde QWERTY klavye, butonlar, dokunmatik ekran, yazılım üzerindeki butonlar ve menüler kullanılmaktadır.

- **Bağlanım Testi**: Bağlanım testi fonksiyonel testlerin tekrar yapılmasından ileri gelir. Çıkan her yeni model için mevcut uygulamalar uyumluluk testlerini kapsamaktadır. Bu sayede yeni modellerin, eski modellerde bulunan tüm uygulama ve fonsiyonel yetenekleri başarı ile gerçekleştirdiğinden emin olunur. Yeni modellerin yanı sıra cihazlara ait yeni versiyonlarında bağlanım testinden geçmesi gerekmektedir.

- **Kabul Testi**: Kabul testleri çok sayıdaki test senaryosunun laboratuvar şartlarında hedef cihaz üzerinde denenmesidir. Bu testin amacı hedef cihazın tüm özelliklerinin ağır bir test durumunda yeterli seviyede çalışabilmesinin ölçülmesidir [17].

2.3 Test Kapsamları

Mobil uygulamalar için çeşitli test kapsamları sırasıyla Şekil 2.2'de gösterilmiştir [16].

Şekil 2.2 Test kapsamlarına ait blok diyagram

- **Fonksiyonel Test:** Fonsiyonel testlerde, mobil cihazların temel özelliklerine ait fonksiyonları test edilmektedir. Birçok çeşit ve modelde mobil cihaz olduğu için bu fonsiyonel testlerde buna uygun hazırlanmalıdır. Genellikle cihazların üretim aşamalarında yapılması gereken testlerdir. İşletim sistemi, tarayıcı, kablosuz ekipmanlar ve dil gibi farklılıklara göre fonsiyonel testin uygulaması gerçekleştirilir. Bu testlerin bir diğer ismi kara kutu (Black Box) testi olarak belirtilmiştir. Bunun anlamı cihaz iç çalışma prensiplerinin test konusunda herhangi bir etkisinin olmamasıdır.

- **Performans Testi**: Mobil uygulama ve cihazlar için bir diğer önemli test çeşidi performans testleridir. Bu testler neticesinde mobil cihazın içerisinde çalışan uygulamaların hızı ve kullanılabilirliği ölçülmektedir. Bu tür testler uygulamaların performans yönünden nerelerde ne kadar hafıza ve işlemci gücü tükettiği konusunda bilgiler vermektedir. Performans testleri test otomasyon sistemleri üzerinde diğer testlere göre uygulanması daha kolaydır. Performans testleri aşağıdaki bağlantı senaryolarına göre ayrı ayrı yapılmaktadır;

  - Sadece Wi-Fi
  - Sadece 2G/3G/4G
  - SIM kart olmadan
  - Uçak modu
  - USB modu
  - Bluetooth modu

- **Güvenlik Testi**: Şifre ve kullanıcı adlarının şifrelenmeden cihaz içinde tutulması kullanıcılar açısından büyük bir güvenlik tehdidi yaratmaktadır. Bunun yanında uygulama kurulumları sırasında arka planda çalışan zararlı yazılımlar güvenlik tehditlerin artmasına yol açmaktadır. Güvenlik testi bu gibi durumların güvenlik açısından ne gibi sorunlara neden olabileceğinin belirlenmesinde yardımcı olmaktadır.

- **Kullanılırlık Testi**: Kullanılırlık testleri mobil cihazların kaliteli olarak kullanıcıya sürülmesi açısından büyük önem taşımaktadır. Uygulama ağırlıklı olan bu testler daha çok kullanım kolaylığı açısından önemlidir. Kullanım kolaylığı mobil cihazların kullanıcı tarafından satın alınışında en büyük önemi taşır.

- **Yük ve Stres Testi**: Mobil cihazların bellek ve işlem gücü yıllar içinde büyük bir ivmeyle artmıştır ve artmaya devam etmektedir. Mobil cihaz sahipleri her zaman yüksek sayıda uygulamanın bir arada sistem çalışma hızını düşürmeyecek bir şekilde çalışmasını istemektedir. Stres testleri bu konuda üreticiye cihaz hakkında bilgi vermektedir. Yüksek işlemci kullanımlarında sistem donmasının veya kapanmalarının tespitinde stres testleri kullanılmaktadır. Test sırasında aynı işlem sürekli olarak yüksek hızda tekrarlatılarak çok miktarda verinin cihaz tarafından işlenmesi sağlanır.

Aşağıda bazı stres test teknikleri verilmiştir;

- Cihaz üzerindeki uygulamaları tam olacak dolacak şekilde çok büyük miktarda veri ile doldurulur. Örnek olarak rehberde bulunan kişi sayısı maksimum sayıda olacak şekilde arttırılır.
- 1. adımda verilen işlem sürekli olarak tekrarlanır, aynı boyuttaki veri sürekli olarak sisteme yüklenir.
- Aynı yükleme işlemi farklı hızlarda (çok yavaş veya çok hızlı) tekrarlanır.
- Uygulama üzerinde aynı işlemler uzun bir süre zarfında gerçekleştirilir ve işlemin bitmesinin ardından işlemci ve bellek kullanımının tekrar normale dönmesi beklenir.
- Rastgele bir şekilde çeşitli butonlara basılır veya dokunmatik ekran üzerinde işlemler yapılır.

- Birden çok uygulama aynı anda denenmekte ise sürekli olarak uygulamalar arası geçişler yapılmalıdır.

- **Yerelleştirme Testi**: Yerelleştirme testi cihaz üzerindeki yerel dil, saat ve para birimi gibi değerlerin cihaza ne kadar başarılı entegre olduğunun tespitinde kullanılmaktadır. Testte genellikle diğer dillere ait karakter ve yazı setlerinin başarısı göz önünde tutulmaktadır.

2.4 Test Teknikleri

Mobil uygulamalar için çeşitli test teknikleri sırasıyla Şekil 2.3'de gösterilmiştir.

Şekil 2.3 Test tekniklerine ait blok diyagram

- **Manuel Testler**: Belirli bir zamana kadar çoğu yazılım ve cihaz testi bir operatör tarafından elle yapılmıştır. Operatör elinde bulunan bir test senaryosunu (çoğunlukla excel tabanlı) adım adım uygulayarak testi gerçekleştirmektedir. Cihaz ile etkileşim ekran üzerinden olmaktadır ve sonuçlar buna göre kayıt edilmektedir. Test senaryolarının çok iyi bir şekilde dokümante edilmiş olması gerekmektedir. Bu testlerden iyi neticeler alınması operatörün test sırasındaki ruh hali, moral, motivasyon gibi durumlarına bağlıdır. Bu durumda insan kaynaklı birçok hataya neden olmaktadır [17].

- **Otomasyon Tabanlı Testler**: Test otomasyon sistemleri sadece yazılım tabanlı uygulamalar olabileceği gibi bir operatör gibi cihaz ile mekanik olarak etkileşime girebilecek robotik bir uygulamada olabilmektedir. Otomasyon sistemlerinde önceden tanımlanmış sonuçlar, alınan çıktılar ile karşılaştırılarak testin sonucunu belirlemektedir ve sonucu bir rapor halinde kayıt eder. Genellikle performans, stres

ve güvenilirlik testleri gibi operatör tarafından yapılması sakıncalı olan uzun süreli ve tekrarlı işlerde kullanılır. Otomasyona dayalı testler hızlı, tekrarlı ve tutarlı sonuçları nedeniyle tercih edilmektedir. Elle yapılan testlere göre test senaryoları iyi belirlenmelidir [18].

- **Risk Tabanlı Testler**: Risk tabanlı testler cihazların ve uygulamaların ne gibi durumlarda hangi oranlarda başarısızlığa uğrayacağının belirlenmesinde kullanılmaktadır. Genellikle uygulamaların ve işletim sisteminin güncellenmesinde bu gibi sorunlar meydana gelebilmektedir.

- **Senaryo Tabanlı Testler**: Senaryo tabanlı testler gerçek yaşamda kullanıcıların karşılaşabileceği durumlar göz önüne alınarak oluşturulan test durumlarıdır. Birçok uygulamanın cihaz üzerinde aynı anda çalışması durumunda, uygulamalar arasındaki geçişlerde ve uygulamalar üzerindeki işlemlerin birbirleri ile etkileşimi bu test senaryolarının oluşturulması ile gözlemlenmektedir.

- **Domain Tabanlı Testler**: Domain tabanlı testler giriş, çıkış ve diğer değişkenlerin durumlarına odaklanmıştır. Bu test sırasında test edilecek alan bazı alanlara bölünmüştür. Bölünen her alan için giriş ve çıkış değerleri incelenir, bu sayede olası hataların cihazın veya uygulamanın hangi bölümlerinde gerçekleştiği bulunur.

- **Rastgele Testler**: Rastgele testler veya maymun testleri yüksek oranda datanın ve test senaryosunun belirli bir sıra ile değil rastgele bir şekilde işletilmesi ile yapılmaktadır. Maymun terimi sistemde yapılan testlerin bir insan yerine başka bir canlının cihazı kullanımını kastetmektedir. Otomasyon tabanlı test sistemleri için geliştirilmiş bu test tekniği test anında çeşitli tuşlara basma veya dokunmatik ekranda işlemlerin rastgele bir şekilde yapılması ile uygulamaya yapacağı etkiler gözlemlenmektedir. Microsoft firmasının raporuna göre bu sayede %10 ile %20 arasındaki sistem hata miktarı bu testle bulunmuştur [19].

- **Emülatör Tabanlı Testler**: Bazı durumlarda gerçek bir mobil cihaz yerine aynı özelliklerini taklit edecek yazılım tabanlı emülatörler testlerde kullanılabilir [16].

Her ne kadar test otomasyon sistemleri üretici yönünden büyük yararlar sağlasada, sistemde kullanılacak test senaryo ve yöntemlerinin oluşturulması yüksek miktarda iş saati gerektirmektedir. Fakat otomasyon sistemleri kalite yönünden mobil cihazlarda büyük farklar yaratmaktadır. Bu durum yeni test senaryosu için oluşturulan algoritma ve yöntemlerinin her geçen gün ortaya çıkmasına neden olmaktadır.

Tablo 2.1 Test kapsamlarının karşılaştırılması [16]

Test Türleri	**Operatör ile Yapılan Testler**		**Test Otomasyon Sistemleri**
	Cihaz	Emülatör	
Birim	Hayır	Evet	Hayır
Entegrasyon	Hayır	Evet	Hayır
Sistem	Evet	Hayır	Hayır
Bağlanım	Evet	Hayır	Evet
Uyumluluk	Evet	Hayır	Evet
Kullanıcı Arayüzü	Evet	Hayır	Hayır
Performans	Evet	Hayır	Evet
Stres/Yük	Evet	Hayır	Evet
Güvenlik	Evet	Hayır	Evet
Senkronizasyon	Evet	Hayır	Hayır

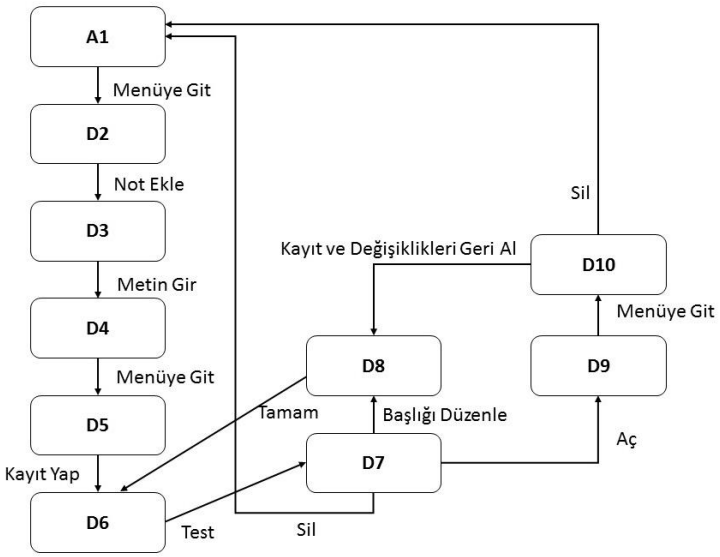
2.4.1 Yakala ve Oynat

Yakala ve oynat (Capture & Replay) test otomasyon sistemleri için oldukça popüler bir yöntemdir. Temel olarak kullanıcının test amaçlı yaptığı tuş ve ekran hareketleri sistemde kayıt edilmektedir. Ardından kayıt edilen test senaryosu tekrar oynatılır ve daha önceden alınan sistem çıktıları ile karşılaştırılır [20]. Bu yöntemin bir dezavantajı oluşturulan senaryonun esnek olmamasıdır. Cihaz veya uygulamanın değişmesi durumunda daha önceden kayıt edilen senaryolar bir daha kullanılamamaktadır. Bu durumda eski kayıtlar yeniden düzenlenmeli veya tekrar kayıt edilmelidir. Olabilecek tüm küçük değişiklikler sistemde hatalara neden olmaktadır. Buna rağmen senaryoların hızlı bir biçimde oluşturulmasına olanak vermektedir [21].

2.4.2 Model Tabanlı Test Senaryoları

Model tabanlı test (MTT) senaryoların son yıllarda test otomasyon sistemlerinde kullanımı oldukça yaygınlaşmıştır. MTT birçok mobil cihaz uygulamada başarıyla kullanılmaktadır [22-25]. Bu konuda geçmiş yıllara göre oldukça çok çalışma yapılmıştır. Test edilecek sisteme ait senaryolar bir model şeklinde hazırlamıştır. Bu sayede daha sistematik ve otomasyona yönelik bir test prosedürü ortaya çıkmıştır.

Mobil cihazlardaki uygulamalara ait senaryoların çıkarılmasında en popüler yöntemlerden biri olay akış grafikleridir (Event Flow Graph). Temel fikir uygulamada gerçekleşebilecek tüm olayları tek bir modelde göstermektir. Grafik oklardan ve noktalardan meydana gelmektedir [26]. Bunun anlamı bir noktadan diğerine bir ok ile gösterim varsa bir sonraki işlemin gerçekleştirilebileceğidir. Şekil 2.4'de bir uygulamaya ait örnek olay akış grafiği görülmektedir.



Şekil 2.4 Örnek bir olay akış grafiği

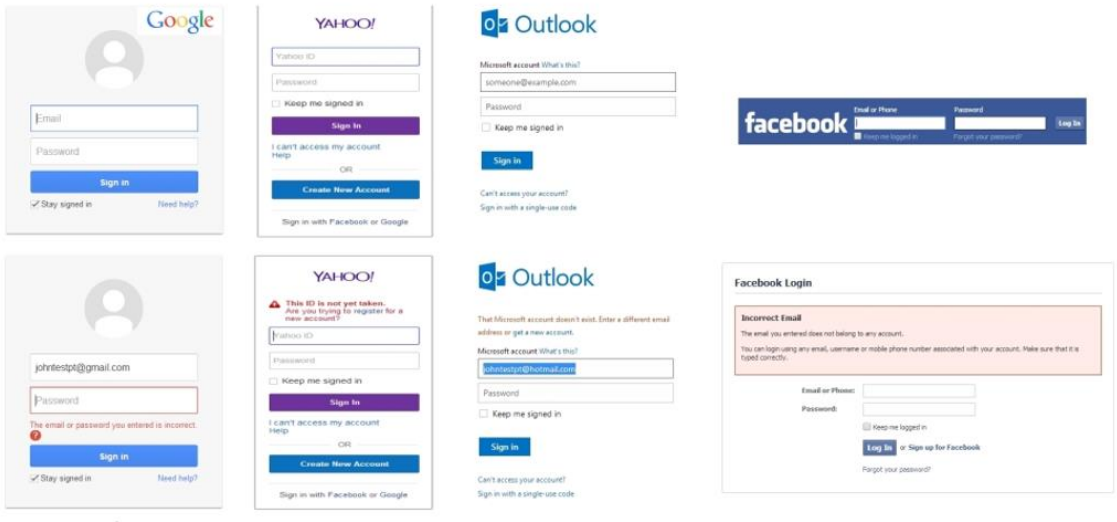
Olay akış grafikleri başarılı modeller oluşturmasına karşın, oluşturulan modellerin çok büyük olması bir sorun yaratmaktadır. Daha karmaşık uygulamalarda tüm modelin oluşturulması pratik açıdan büyük zorluklar oluşturabilmektedir.

2.4.3 Desen Tabanlı Test Senaryoları

Mobil cihazlara ve uygulamalara ait yazılımların birçoğu birbirine benzer bir şekilde kullanıcı desenlerine (Pattern) sahiptir. Mevcut bu desenleri kullanarak bir çok uygulama için ortak bir test senaryosu yaratılabilmektedir. Diğer mevcut yöntemler mobil cihazlara ait bu özellikten yararlanmazlar.

Üretilen mobil uygulamalar temelde pencereler ve görsel bileşenlerden oluşmaktadır. Bu görsel bileşenler uygulamalar içinde farklı şekillerde bulunmasına karşın uygulamada aynı işi yapmaktadır.

Şekil 2.5'de farklı uygulamalara ait giriş ekranı ve hatalı giriş sırasında çıkan ekran verilmektedir. Her ne kadar farklı ekranlar olarak görünsede ortak olan 2 adet metin giriş kutusu ve 1 adet buton her ekranda bulunmaktadır [27]. Ortak olarak bulunan bu öğeler desen test senaryoları olarak kullanılabilmektedir. Bu sayede her uygulama için farklı bir test senaryosuna gerek kalmamaktadır. Oluşturulan desenler model tabanlı senaryolara göre tekrar kullanılabilmektedir [28].

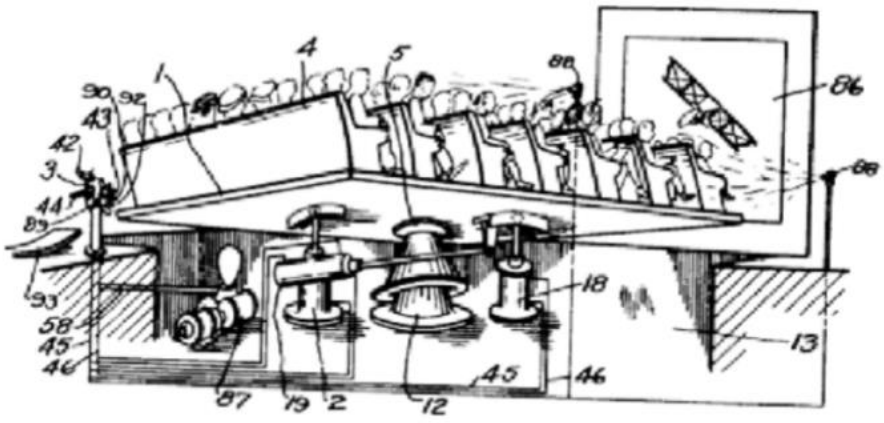


Şekil 2.5 Desen tabanlı bir test şablon örneği [27]

3. PARALEL ROBOTLAR

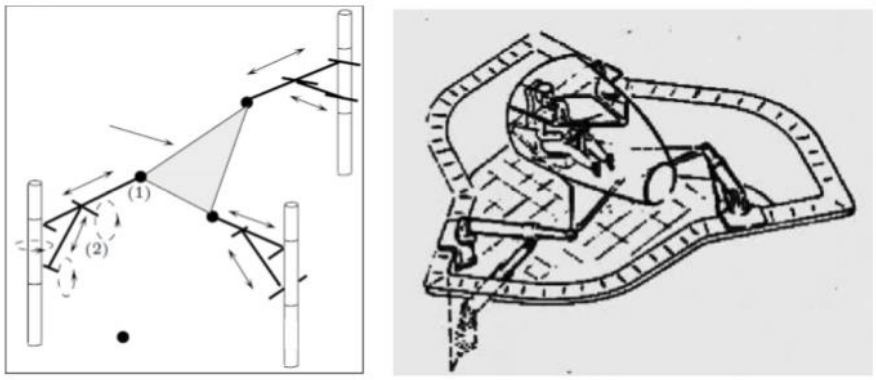
3.1 Paralel Robotların Tarihçesi

Paralel robotlar, bir adet hareketli bir adette sabit platformun birbirlerine seri olacak şekilde uzuvlar aracılığı ile bağlanmasından oluşmaktadır. Bu tip robotlarda uzuv sayıları 2 veya daha fazla sayıda olmaktadır [29]. Bu tip robotlara ait çalışmalar 17. yüzyıla kadar gitmektedir. İlk olarak Christopher Wren ardından 19.yüzyıl'da Cauchy, Lesque ve 1897 de Bricard parallel robotlar üzerine çalışmalar yapmıştır. 1928 yılında Gwinnett paralel robotlara ait ilk patenti almıştır. Bu robota ait çizim Şekil 3.1'de görülmektedir.



Şekil 3.1 Gwinnett paralel robotu [29]

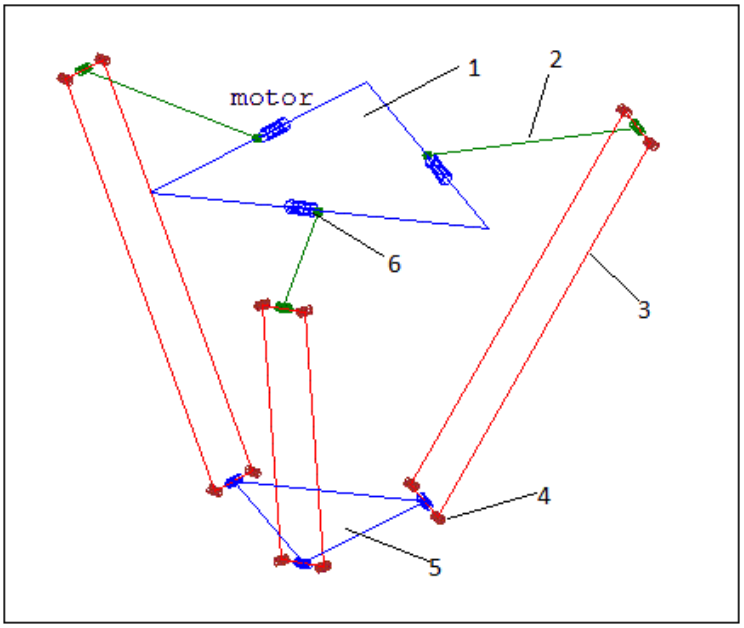
1947 yılında paralel robotlara ait ilk hareket denklemleri bulunmuştur. 1960'lı yıllardan sonra havacılık sektöründe pilotların eğitimleri için bu tip sistemlerden yararlanılmıştır. 1965'de ise ilk uçak simülatörü olan Steward platformu geliştirilmiştir [29]. Şekil 3.2'de bu uçak simülatörüne ait çizim verilmiştir.



Şekil 3.2 Steward uçak simülatörü [29]

3.2 Delta Robotlar

Delta robotlar paralel robotlar ailesine dahil olan bir robot çeşididir. Temel olarak Şekil 3.3'de görüldüğü gibi 3 adet koldan ve 2 adet üçgen platformdan oluşmaktadır. Robota ait kollar bir adet sabit platforma 120 derecelik eşit açılarla bağlanmıştır. Kolların diğer uçları sabit plakadan daha küçük oynar bir platforma bağlıdır.



Şekil 3.3 Delta robot bileşenleri 1. Sabit platform 2. Aktif kol 3. Bağımlı kol 4. Küresel eklemler 5. Yürüyen platform 6. Aktüatör [30]

Delta robotlar özellikle paketleme, dizme, sıralama ve toplama gibi işlemleri yüksek bir hassasiyet ve tekrarlanabilirlikle yerine getirmektedir. Delta robotlar sahip oldukları yüksek hız sayesinde bu işlemler için talep görmektedir.

Delta robotlar 1980 senesinin başlarında İsviçre'nin Ecole Polytechnique Federate de Lausanne araştırma enstitüsünde keşfedilmiştir. Küçük ve hafif objeleri çok yüksek hızlarda manipüle edebilmektir. Delta robot 3 boyutlu uzayda öteleme hareketlerini yapmak üzere tasarlanmıştır [31]. Yapılan bir çalışmada ise delta robotun dinamik modelini, robotun mevcut konumu ve hareketini kullanarak iteratif matris eşitlikleriyle elde etmişler ve çıkan sonuçların deneysel verilerle tutarlı olduğunu görmüşlerdir [32-35]. Delta robota ait kinematik denklemlerin çözülmesi amacıyla 2004 yılında üst ve alt platformların bir küre ile modellenmiş ve birbirleri ile kesiştirilmesi yöntemiyle yeni bir kinematik yöntem bulunmuştur. Bu kesişme robotun konumunu doğru bir şekilde vermektedir. Çıkan sonuçlar üzerine bir bilgisayar yazılımı geliştirilmiştir [33]. Başka bir çalışmada ise çalışma hacmi önceden belirlenmiş bir delta robotun optimum boyutlarının bulunması için genetik algoritmaya dayanan bir metot ortaya konulmuştur [36].

2005 yılında yapılan bir çalışmada delta robotların uygulama alanlarına bir yenisi daha eklenerek kalp masajı yapabilen bir delta robot geliştirilmiştir [34]. Delta robotların sağlık endüstrisinde de kullanılabileceği ortaya konulmuştur.

2009 yılında ise bir diğer robot üreticisi Japon FANUC (Fuji Automatic Numerical Control) firması M-1iA model delta robotu üretmiştir. Bunun yanında daha büyük ve ağır objeler üzerinde işlem yapabilmek için M-3iA robotunu 2010 yılında piyasaya sürmüştür. Şekil 3.4'te Fanuc M-1iA delta robotuna ait resim verilmiştir.

Şekil 3.4 Fanuc firmasına ait M-1iA delta robotu [37]

Delta robotlar elektronik sanayi başta olmak üzere ilaç sektörü, tıbbi uygulamalar ve paketleme sanayi gibi alanlarda yoğun bir şekilde kullanılmaktadır. Elektronik kartların dizgisinde, ilaçların sınıflandırılması, paketlenmesi ve kalite kontrol gibi işlemlerde yüksek hızı sayesinde tercih edilmektedir.

Delta robotlar diğer seri manipülatörlerle karşılaştırıldıklarında atalet kuvvetleri daha büyük olmaktadır. Bu nedenle robot yapıları daha hafif olmakta ve daha hızlı çalışabilmektedirler. Buna karşın tek bir gövde üzerine tüm eksenlerin bağlanması imalat sırasında birçok zorluğa neden olmaktadır [38].

3.3 Kinematik Analiz

Kinematik analiz eklemlerin açılarına göre hareketli platformun konumunun bulunması veya platformun belirli bir konumda olmasını sağlayacak açıların belirlenmesinde kullanılmaktadır. Kolların açılarına göre hareketli platformun

konumunun hesaplandığı analize ileri kinematik analiz aksi duruma ise ters kinematik analiz denmektedir [39].

Delta robotların merkezleri referans alındığında simetrik bir yapıya sahiptirler. Simetrik yapılarından dolayı kinematik çözüm için her 3 eklemden biri başlangıç olarak kabul edilebilir. Şekil 3.5'de görüldüğü gibi kolların bağlandığı sabit platformun merkezi kartezyen koordinat sistemine göre orijin olarak kabul edilir. Burada;

R_a : Sabit platformun yarıçapı

R_b : Hareketli platformun çapı

a : Aktif kolun uzunluğu

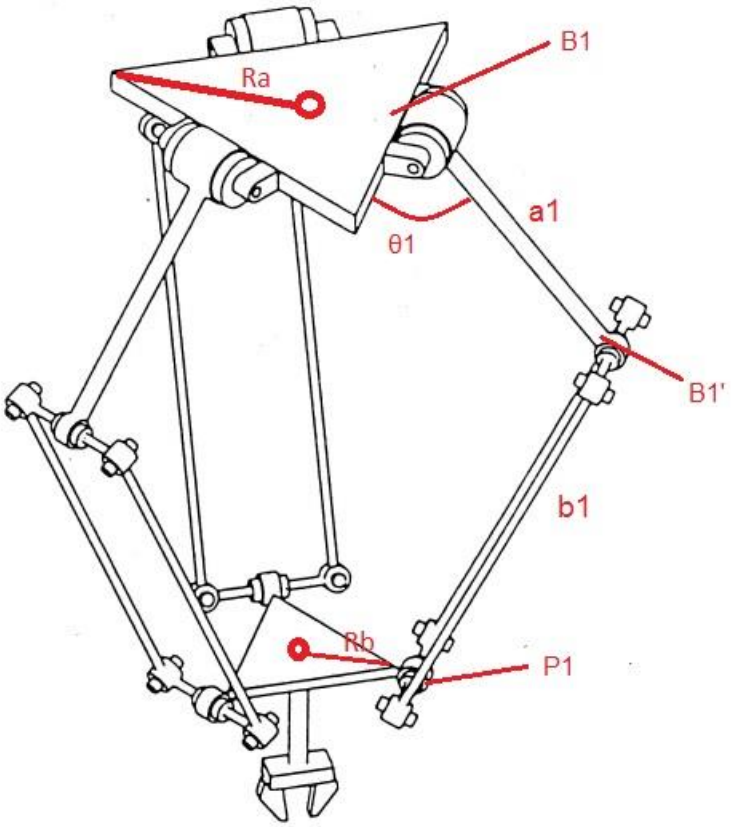
b : Bağımlı kolun uzunluğu

$\theta_1, \theta_2, \theta_3$: X-Y düzleminde aktif kollar ile sabit platform arasındaki açılar

B_i : Sabit eksene göre hareket eden parçanın geometrik konumu

P_i : Sabit eksene göre hareketli platformun geometrik konumu

B_i' : Aktif kol ile bağımlı kol arasındaki eklemin konumu



Şekil 3.5 Delta robota ait elemanların uzunlukları [33]

3.3.1 İleri Kinematik Analiz

İleri kinematik analiz, açıları bilinen delta robotun hareketli platfomunun kartezyen koordinat sistemine göre yerini bulunmasında kullanılmaktadır [39].

Şekil 3.6 Delta robotlara ait ileri kinematik dönüşüm

$$B'_{ix} = B_{ix} - a_i * \sin(\theta_{bi} + \beta_i) * \cos(\theta_{i1}) \qquad i = 1 \dots 3 \qquad (3.1)$$

$$B'_{iy} = B_{iy} - a_i * \cos(\theta_{bi} + \beta_i) * \cos(\theta_{i1}) \qquad i = 1 \dots 3 \qquad (3.2)$$

$$B'_{iz} = B_{iz} - a_i * \sin(\theta_{i1}) \qquad i = 1 \dots 3 \qquad (3.3)$$

Yukarıdaki bağıntılara göre aşağıdaki denklem yazılabilir, P noktaları hareketli platforma ait eklem koordinatlarını vermektedir.

$$(B'_{ix} - P_{xi})^2 + (B'_{iy} - P_{yi})^2 + (B'_{iz} - P_{zi})^2 = b_i^2 \qquad (3.4)$$

Her ekleme ait bağıntıları yazacak olursak;

$$(B'_{ix1} - P_{xi})^2 + (B'_{iy1} - P_{yi})^2 + (B'_{iz1} - P_{zi})^2 = b_1^2 \qquad (3.5)$$

$$(B'_{ix2} - P_{xi})^2 + (B'_{iy2} - P_{yi})^2 + (B'_{iz2} - P_{zi})^2 = b_2^2 \qquad (3.6)$$

$$(B'_{ix3} - P_{xi})^2 + (B'_{iy3} - P_{yi})^2 + (B'_{iz3} - P_{zi})^2 = b_3^2 \qquad (3.7)$$

Denklemlerin çözümü için B'ix noktalarının Rp kaydırılması durumunda B' den B'' koordinatları elde edilmektedir.

$$(x - B''_{x1})^2 + (y - B''_{y1})^2 + (z - B''_{z1})^2 = b_1^2 \qquad (3.8)$$

$$(x - B''_{x2})^2 + \left(y - B''_{y2}\right)^2 + (z - B''_{z2})^2 = b_2^2 \qquad (3.9)$$

$$(x - B''_{x3})^2 + (y - B''_{y3})^2 + (z - B''_{z3})^2 = b_3^2 \qquad (3.10)$$

$$w_i = B''^2_{ix} + B''^2_{iy} + B'^2_{iz} \qquad (3.11)$$

Yukarıda verilen denklemlerin çözümü neticesinde aşağıdaki ifadeler bulunmaktadır.

$$(x - B''_{x1})^2 + (y - B''_{y1})^2 + (z - B''_{z1})^2 = (x - B''_{x2})^2 + \left(y - B''_{y2}\right)^2 + (z - B''_{z2})^2 \quad (3.12)$$

$$(x - B''_{x1})^2 + (y - B''_{y1})^2 + (z - B''_{z1})^2 = (x - B''_{x3})^2 + \left(y - B''_{y3}\right)^2 + (z - B''_{z3})^2 \quad (3.13)$$

$$(x - B''_{x3})^2 + (y - B''_{y3})^2 + (z - B''_{z3})^2 = (x - B''_{x2})^2 + \left(y - B''_{y2}\right)^2 + (z - B''_{z2})^2 \quad (3.14)$$

Verilen denklemlerin çözülmesi ile;

$$x = \frac{j_1 - z * h_1}{k} \quad (3.15)$$

$$y = \frac{j_2 - z * h_2}{-k} \quad (3.16)$$

$$h_1 = \left(B''_{y3} - B''_{y1}\right)(B'_{z2} - B'_{z1}) - (B''_{y2} - B''_{y1})(B'_{z3} - B'_{z1}) \quad (3.17)$$

$$h_2 = (B''_{x3} - B''_{x1})(B'_{z2} - B'_{z1}) - (B''_{x2} - B''_{x1})(B'_{z3} - B'_{z1}) \quad (3.18)$$

$$j_1 = \frac{(w_2 - w_1)\left(B''_{y3} - B''_{y1}\right) - (w_3 - w_1)\left(B''_{y2} - B''_{y1}\right)}{2} \quad (3.19)$$

$$j_2 = \frac{(w_2 - w_1)\left(B''_{x3} - B''_{x1}\right) - (w_3 - w_1)\left(B''_{x2} - B''_{x1}\right)}{2} \quad (3.20)$$

$$k = \left(B''_{y3} - B''_{y1}\right)(B''_{x2} - B''_{x1}) - (B''_{y2} - B''_{y1})(B''_{x3} - B''_{x1}) \quad (3.21)$$

(3.15) ve (3.16) denklemleri (3.8) denkleminde yerine yazılırsa z ye bağlı ikinci dereceden bir denklem bulunur.

$$\left(\frac{h_{1+}^2 h_2^2}{g^2}+1\right)z^2 + \left(\frac{h_1 j_1+h_2 j_2}{k^2}+\frac{h_1 B'_{1x}}{k}-B'_{1z}\right)2z+\left(B''^{\,2}_{1x}+B'^{\,2}_{1z}-b_2^2+\frac{j_1^2+j_2^2}{k^2}+\frac{2j_1 B''_{1x}}{k}\right)=0 \quad (3.22)$$

Elde edilen (3.22) denklemi için ikinci dereceden bilinmeyen bir çözüm yapılırsa z eşitliği bulunmaktadır. Hesaplanan z kökleri için her zaman küçük reel köklerin alınması gerekmektedir. Eğer kökler kompleks çıkarsa verilen koordinatların geçerli olmadığı ve robotun hareketi yapamayacağı sonucu oluşmaktadır.

3.3.2 Ters Kinematik Analiz

İleri kinematik çözümün bulunmasında kullanılan (3.1) (3.2) (3.3) kadar olan denklemler (3.4) de yerine yazılırsa aşağıdaki (3.23) denklemi elde edilir.

Şekil 3.7 Delta robotlara ait ters kinematik dönüşüm

$$E_i \cos\theta_{i1} + F_i \sin\theta_{i1} = G_i \quad (3.23)$$

$$E_i = (y_i - B_{iy})cos(\theta_{bi} + \beta_i) - (x_i - B_{ix})sin(\theta_{bi} + \beta_i) \quad (3.24)$$

$$F_i = (z_i - B_{iz}) \quad (3.25)$$

$$G_i = \frac{(x_i - B_{ix})^2+(y_i - B_{iy})^2+(z_i - B_{iz})^2+a_i^2-b_i^2}{2a_i} \quad (3.26)$$

$$K_i = 1 \quad (3.27)$$

$$H_i = E_i^2 + F_i^2 - G_i^2 \tag{3.28}$$

Yukarıdaki (3.24) – (3.28) eşitlikleri (3.23) de yerine yazılır ise

$$\sin\theta_{i1} = \frac{F_i G_i + K_i E_i \sqrt{H_i}}{E_i^2 + F_i^2} \tag{3.29}$$

$$\cos\theta_{i1} = \frac{E_i G_i - K_i F_i \sqrt{H_i}}{E_i^2 + F_i^2} \tag{3.30}$$

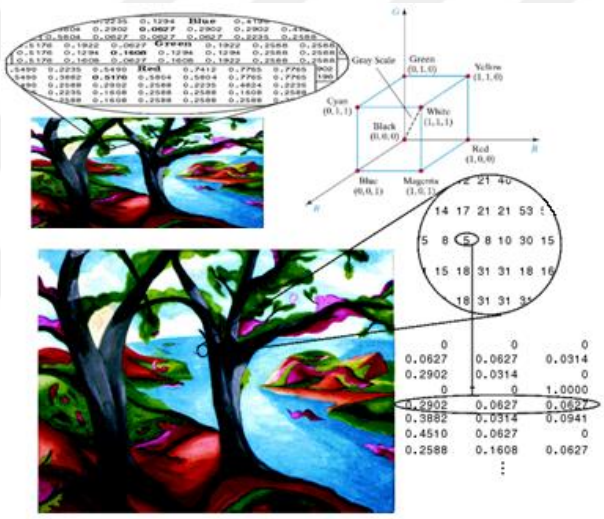
Sonuç olarak (3.29) ve (3.30) denklemlerinin arctan işleminden geçirilmesi ile θ_{i1} elde edilir.

$$\theta_{i1} = \tan^{-1}(\sin\theta_{i1} / \cos\theta_{i1}) \tag{3.31}$$

4. YAPAY GÖRME

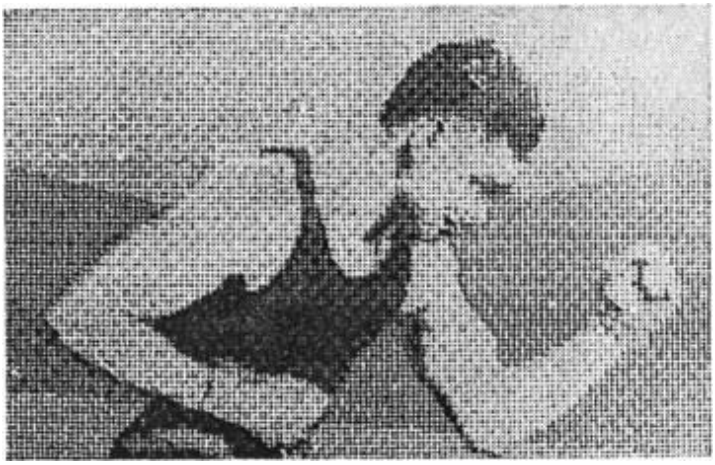
4.1 Yapay Görmeye Giriş ve Tarihçe

Yapay görme, bilgisayar algoritmalarının dijital bir görüntüde yaptığı işlemlere verilen bir isimdir. Analog görüntü işlemeye göre oldukça büyük avantajları vardır. Dijital görüntü işleme birçok uygulama alanına sahiptir. Akıllı sistemler/robotik, uzaktan algılama, fotoğrafçılık, tıbbi görüntüleme gibi alanlarda kullanılmaktadır. Resime ait değerler piksel adı verilen küçük noktalara ayrılmıştır. İçlerinde ait olduğu koordinatları ve renk bilgisini taşımaktadırlar. Şekil 4.1'de görüntünün piksellerden oluşan yapısı görülmektedir.



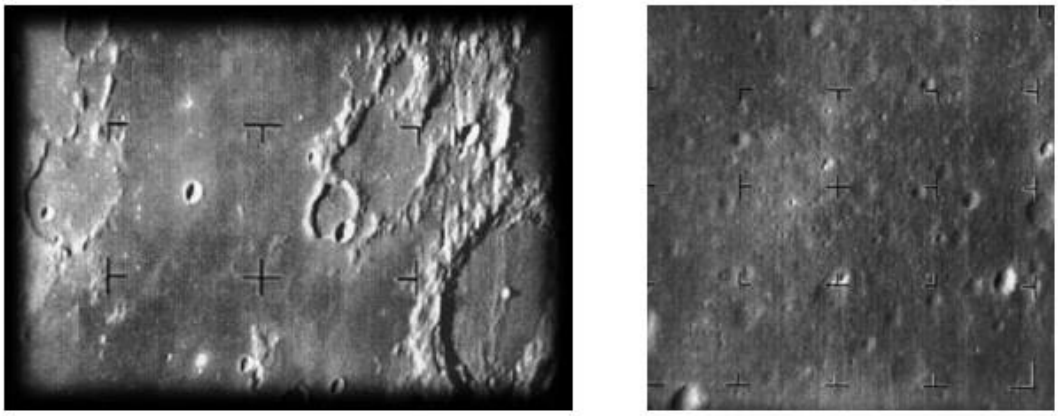
Şekil 4.1 Görüntünün piksel bazında incelenmesi [40]

İlk yapay görme çalışmaları 1920'lerde gazete resimlerinin iletimi için Atlas okyanusu altına döşenmiş hatlardan resimlerin iletilmesi ile başlamaktadır. Resimler sıkıştırılmış olarak gönderilmekte ve karşı tarafta özel bir alıcı sayesinde tekrar resim haline getirilmekteydi. Bu işlem 3 saat kadar sürmektedir. Bu uygulamaya ait bir görüntü Şekil 4.2'de verilmiştir.



Şekil 4.2 1920'lerde dijital olarak aktarılan bir resim [41]

1964 yılında ise dijital resimler üzerinde işlemler yapabilen ilk bilgisayar üretilmiştir. Üretilen bilgisayar NASA (Amerikan Ulusal Uzay Ajansı)'nın Ay yüzeyine indirdiği Ranger 7 uydusundan gelen resimlerim analizinde kullanılmıştır. İlerleyen yıllarda bilgisayar teknolojisinin gelişmesi ile masaüstü bilgisayarlarda yapay görüntüleme çalışmaları yoğunlaşmış ve bugünkü halini almıştır. Şekil 4.3 Ranger-7 uydusunun çektiği Ay resimlerini göstermektedir.



Şekil 4.3 Ranger-7 uydusunun çektiği ay resimleri [41]

4.2 Yapay Görme Sistemlerinin Avantajları

- **Üretim Devamlılığı:** Üretim tesislerinde makinelerin sürekli çalışması ve arızalanması gibi nedenlerden dolayı üretimin devamlılığı kesilebilmektedir. Bu gibi durumlar için yapay görmeye dayalı sistemler tercih edilirse işletme maliyetleri düşmekte ve devamlılık sağlanmaktadır.

- **Düşük İşçilik Giderleri:** Üretim miktarının yoğun olduğu işletmelerde işçi sayısıda bu duruma bağlı olarak artmaktadır. Nitelikli işler dışında kullanılacak yapay görme sistemleri çalışan sayısını azaltmaktadır. Özellikle montaj, kalite kontrol ve paketleme gibi işler bu teknolojiye dayalı robotik sistemlere yaptırılabilmektedir. Böylece işçilik maliyetleri büyük oranda düşmektedir.

- **Yüksek Üretim Kapasitesi ve Rekabet Faktörü:** İnsanlardan farklı olarak makineler 7/24 çalışabilmektedir. Uygun bakımları yapıldığı takdirde bu çalışma devamlılığı sürecek ve üretim kapasitesi artarak, firmanın rekabet gücüne katkı yapabilecektir.

- **Maksimum Kalite ve Minimum Hata:** Çalışanlardan kaynaklanan üretim hataları endüstrinin bir diğer sorunudur. Makineli otomasyon sistemleri ile kalite kontrol ve hata tespiti gibi operasyonlar mükemmele yakın bir şekilde gerçekleştirilebilmektedir. Yapay görme sistemlerinin bu gücü üretimde kaliteyi arttırarak, minimum hatanın elde edilmesine olanak sağlamaktadır.

- **İşçi Güvenliği:** Olabilecek iş kazaları için görüntü işleme sistemlerinden yararlanılabilir.

4.3 Yapay Görme Sistemlerinin Uygulama Alanları

Yapay görmenin yoğun olarak kullanıldığı alanlar şu şekilde özetlenebilir;

- **Uzaktan Algılama:** Uyduların Dünya'nın yüzeyinden aldığı görüntüler hava tahminleri, maden arama ve istihbarat gibi alanlarda kullanılır.

- **Endüstriyel Otomasyon:** Robot kontrolü, üretim kontrolü, kalite tespiti ve üretim güvenliği alanlarında kullanılır.

- **Sağlık:** Tıbbi alanda X-Ray, bilgisayarlı tomografi, MRI, PET taraması ve termal görüntüleme gibi işlemlerde kullanılır.

- **Bilimsel Konular:** Mikroskopla alınan verilerin analizinde, X-ray analizleri ve yüzey analizleri gibi konularda yararlanılır.

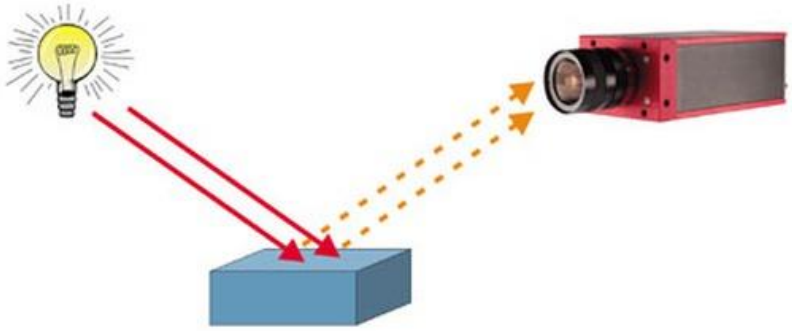
- **Savunma:** Hedef takibi, istihbarat, akıllı silahlar ve kılavuzlama gibi konularda yararlanılmaktadır.

4.4 Yapay Görme Sistemlerinin Bileşenleri

Uygulama alanları farklılık göstersede bir yapay görme sistemi için (i) Işıklandırma (ii) Görüntü alma (iii) Görüntü işleme olmak üzere üç ana faktör vardır.

4.4.1 Işıklandırma

Yapay görmeye dayalı bir sistem geçekleştirileceği zaman, seçilecek ışık kaynakları yakalanacak görüntüler üzerinde büyük bir etki yaratmaktadır. Bu önemli durum çoğu zaman göz ardı edilebilmektedir. Görüntü almada kullanılan kameralar objelerin kendisi yerine üzerinden yansıyan fotonların oluşturduğu görüntüyü kayıt etmektedir. Görüntü kalitesinin ışıklandırma tarafından bu denli etkilenmesi oluşturulacak sistemde uygun bir aydınlatmayı gerektirmektedir [42].



Şekil 4.4 Yapay görmede aydınlatma tekniği [42]

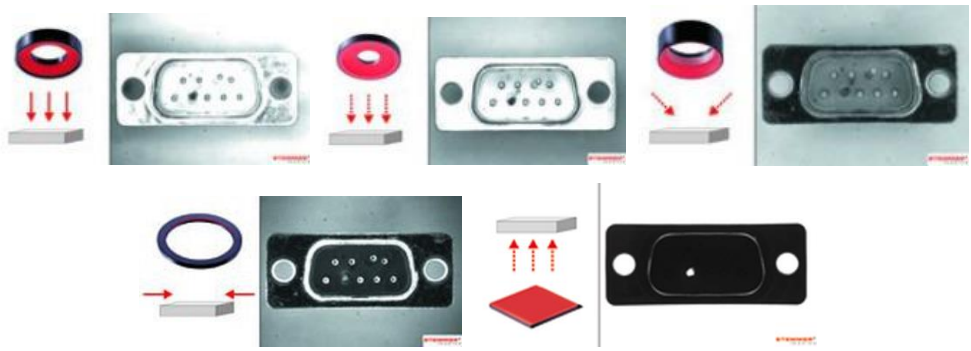
Kullanılan aydınlatma kaynakları şu şekildedir;

- LED aydınlatma
- Soğuk aydınlatma kaynakları
- Lazer aydınlatma
- Floresan ışık
- Halojen lambalar

Bunlardan LED aydınlatma en çok kullanılan aydınlatma türü olmaktadır. Led aydınlatma uzun ömürleri ile ucuz olması ve montaj sırasında kolaylık sağlaması sebebiyle tercih edilmektedir.

Temel aydınlatma teknikleri 5 şekilde olmaktadır, Şekil 4.5'de farklı ışıklandırma yöntemlerinin etkileri gösterilmiştir.

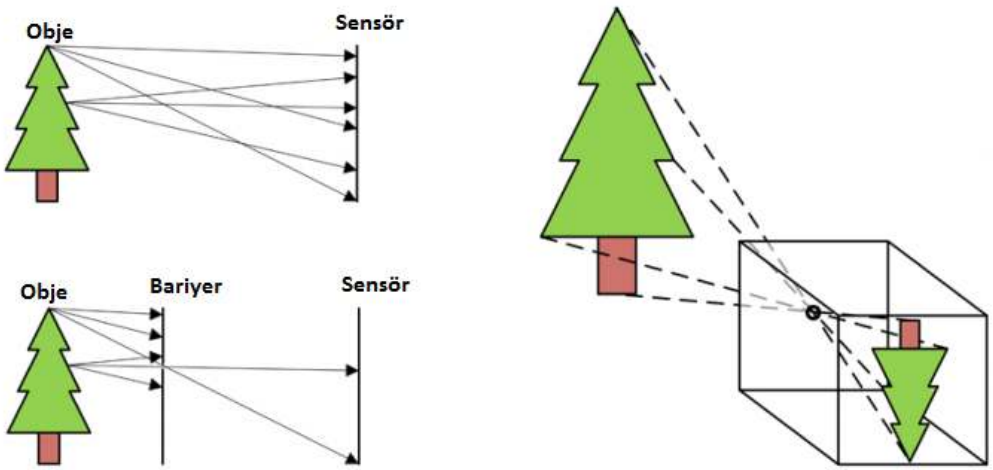
- Direk önden aydınlatma
- Ayrışık aydınlık alan aydınlatması
- Ayrışık karanlık alan aydınlatması
- Karanlık alan aydınlatması
- Arkadan aydınlatma



Şekil 4.5 Farklı ışıklandırmanın aynı cisim üzerindeki farklı etkileri [42]

4.4.2 Görüntü alma

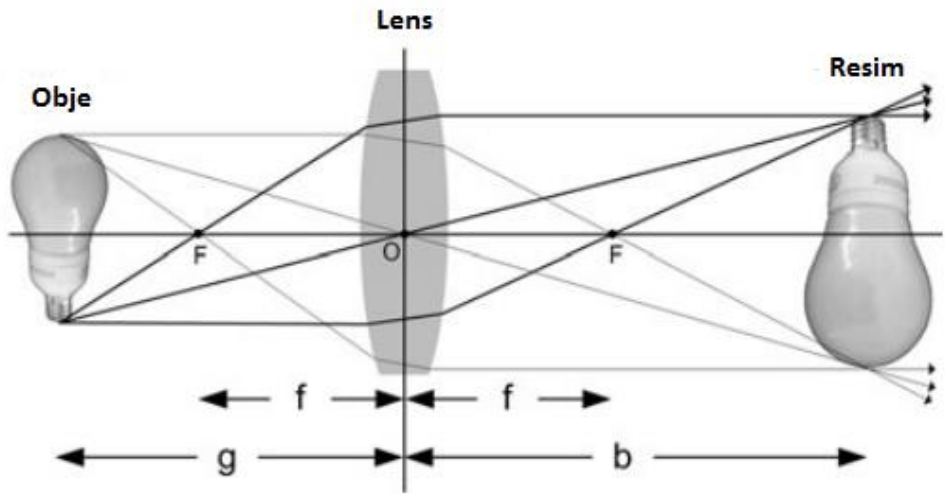
Objenin aydınlatılmasından sonra, sıra yansıyan ışınların kamera ile yakalanmasına gelmektedir. Fakat objeden yansıyan ışınların sensöre doğrudan gelmesi büyük sorunlar yaratmaktadır. Bunun için sensör ile cisim arasında bir tür bariyer olması gerekmektedir. Bariyer fikri pratikte güzel bir sonuç sağlasa da, sensöre ulaşan ışık miktarını önemli ölçüde düşürmektedir. Bu durumun üstesinden gelmek için odak ve lens kavramları devreye girmektedir [43]. Şekil 4.6'da odak kullanılması ve kullanılmamasının görüntü üzerindeki etkisi gösterilmiştir.



Şekil 4.6 Odak kullanılarak görüntünün elde edilmesi [43]

Optik bir görüntüleme sisteminin en temel bileşeni lens olarak adlandırılır. Camdan oluşmaktadır ve gelen ışığı kırarak sensör üzerine düşürmeye yardımcı olur.

Gelen ışınlar her noktada farklı bir açıda kırılarak, görüntünün tek bir noktada toplanması sağlanmaktadır. Paralel olarak gelen ışınların ise aynı yere denk gelecek şekilde kırıldığı görülmektedir. Bu noktaya odak noktası (Focal Point) denilmektedir ve F ile gösterilir. Optik orta nokta O ile gösterilir ve bu noktadan odak noktasına kadar olan mesafeye odak uzunluğu denilmektedir ve f ile sembolize edilir [43].



Şekil 4.7 Lens kullanılarak görüntünün elde edilmesi [43]

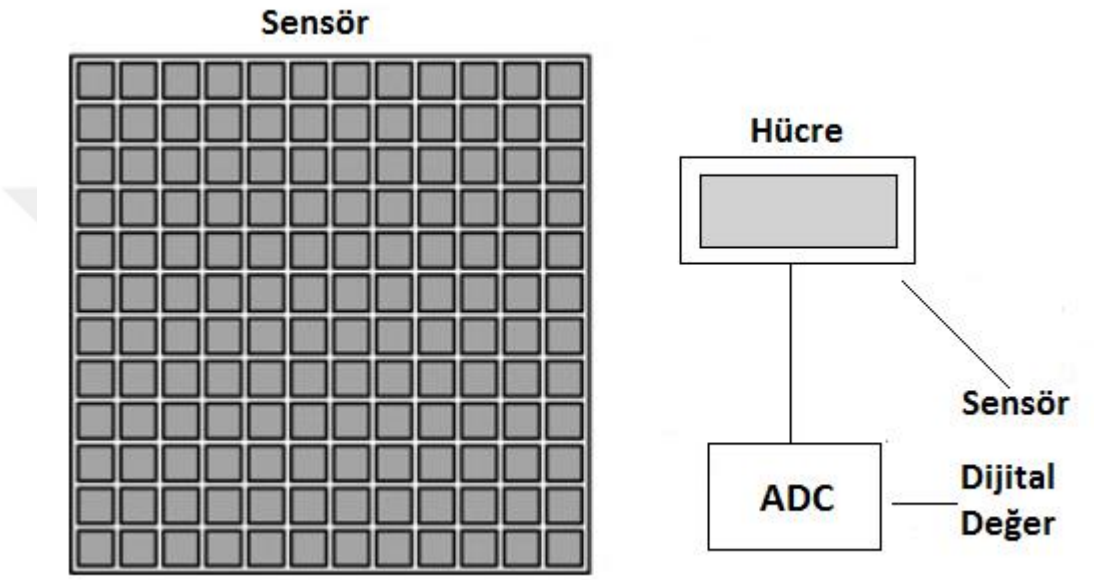
Bir diğer önemli parametre kameraların sahip oldukları görüş açısıdır (Field-of-View) . Birçok uygulamada bu parametre oldukça önemli sonuçlar doğurmaktadır. Bu duruma ait denklemler aşağıda verilmiştir.

$$FOV_x = 2 * \tan^{-1}\left(\frac{sensör\ genişliği/2}{f}\right) \quad (4.1)$$

$$FOV_y = 2 * \tan^{-1}\left(\frac{sensör\ yüksekliği/2}{f}\right) \quad (4.2)$$

Bir başka önemli parametre ise ışığın odaktan girmesine olanak veren odak açıklığıdır. Odak açıklığının miktarı odak önündeki mekanik bir düzenek ile sağlanmaktadır. Resimlerin karanlık veya aydınlık çıkmasını odak açıklığının miktarı sağlamaktadır.

Gelen ışınlar optik bir lensten geçtikten sonra sıra kameranın sensörü ile bu ışınların kaydedilmesindedir. Sensör x ve y yönünde düzgün bir şekilde sıralanmış hücrelerin biraraya gelmesiyle oluşmaktadır. Her bir hücre bir adet piksele karşılık gelmektedir. Hücrelere gelen ışık büyüklüğüne göre bir voltaj değerine döndürülür, ardından bu değer dijital olarak kayıt altına alınır. Işık miktarı arttıkça dijital değerde o derece artmaktadır. Şekil 4.8'de hücre yapısı ve dijital çevrim gösterilmiştir.



Şekil 4.8 Kamera sensörü ve görüntünün elde edilmesi

4.4.3 Görüntü işleme

Sensör ile elde edilen görüntüler dijital bir ortamda kayıt edilmektedir. Görüntüye ait her bir piksele ait değerler bir (x,y) koordinatında muhafaza edilmektedir. Her piksel değeri bir tavan değerine sahiptir ve bu değer genellikle 256 ve katları olarak tanımlanmaktadır. Bu değerin aşılması durumunda piksel değeri satürasyona uğrar ve tavan değerde kalır.

Piksellerin oluşturduğu bu görüntü eğer bir sisteme sokularak içindeki değerler veya görüntüye ait özellikler değiştirilirse ise yeni bir görüntü oluşur, bu durum

görüntü işleme olarak kabul edilmektedir. Temel olarak görüntü işleme amaçları şu şekilde olmaktadır;

- Görüntü iyileştirme
- Görüntü onarma
- Görüntü sıkıştırma
- Görüntü analiz etme
- Görüntü tanıma

Alınan görüntüler temelde şu türlerden oluşmaktadır;

- **İkili Görüntü:** Sadece siyah ve beyaz piksellerden oluşur, her piksel 1 bitlik yer kaplar.

- **Gri Tonlamalı Görüntü:** Sadece grinin tonlamalarından oluşmaktadır, çoğunlukla piksel başına 8 bit olarak tanımlanırlar.

- **Renkli Görüntü:** RGB, CMYK gibi renk modeli ile 3 renk katmanından oluşmaktadır.

Görüntü işlemeye ait konular son yıllarda çok çeşitli ve büyük miktarda olmasına rağmen, temel olarak şu şekilde özetlenebilir;

- **Görüntü İyileştirme:** Görüntünün kalitesini arttırarak, daha iyi bir görünüme getirme işlemidir. Görüntüyü karartma, daha açık hale getirme, kontrastını arttırma veya kenarları belirginleştirme (keskinleştirme) gibi işlemler uygulanır. Bu konuda alçak geçiren, yüksek geçiren filtreler, görüntünün histogramı üzerindeki işlemler görüntü iyileştirme tekniklerini oluşturur.

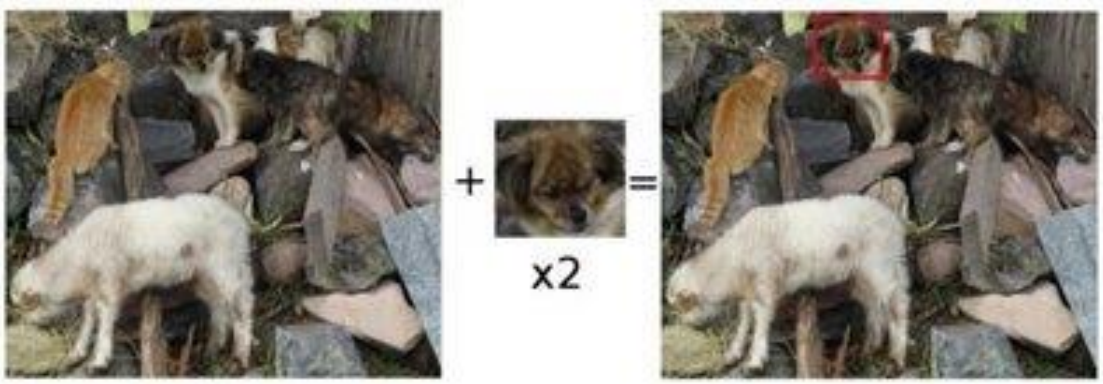
- **Görüntü Onarma:** Bozulmuş veya gürültüye maruz kalmış görüntüyü alarak daha temiz hale getirmek için yapılmaktadır. Kameradan kaynaklı olabilecek gürültüler

ile nesnenin o anki hareketi nedeniyle (motion blur) olan bulanıklıklar bu işlemler ile düzeltilir.

- **Görüntü Sıkıştırma:** Görüntü sıkıştırma mevcut bir dijital görüntünün içinde bulunan veri miktarının azaltılması amacıyla yapılan bir görüntü işleme tekniğidir. Bu tekniğin kullanılmasının ilk nedeni veri iletimindeki görüntü sayısını arttırabilmektir. Ayrıca görüntü içinde bulunan daha az anlamlı veriler gene bu yolla atılarak daha az verinin işlenmesi sağlanır. En çok kullanılan görüntü sıkıştırma algoritması JPEG formatıdır.

- **Nesne Takibi:** Zaman boyunca görüntüdeki nesne ve nesnelerin konumlarını belirleme ve takip etme işlemidir. Genellikle videolar üzerinde kullanılmaktadır. Nesnenin belli bir yönde hareket ettiği ve şeklini koruduğu durumlarda hareketli nesne takibi kolay bir işlemdir. Fakat belirli bir yönde hareket etmeyen veya birçok defa bir engelin arkasında kalan cisimlerde nesne takibi oldukça zordur.

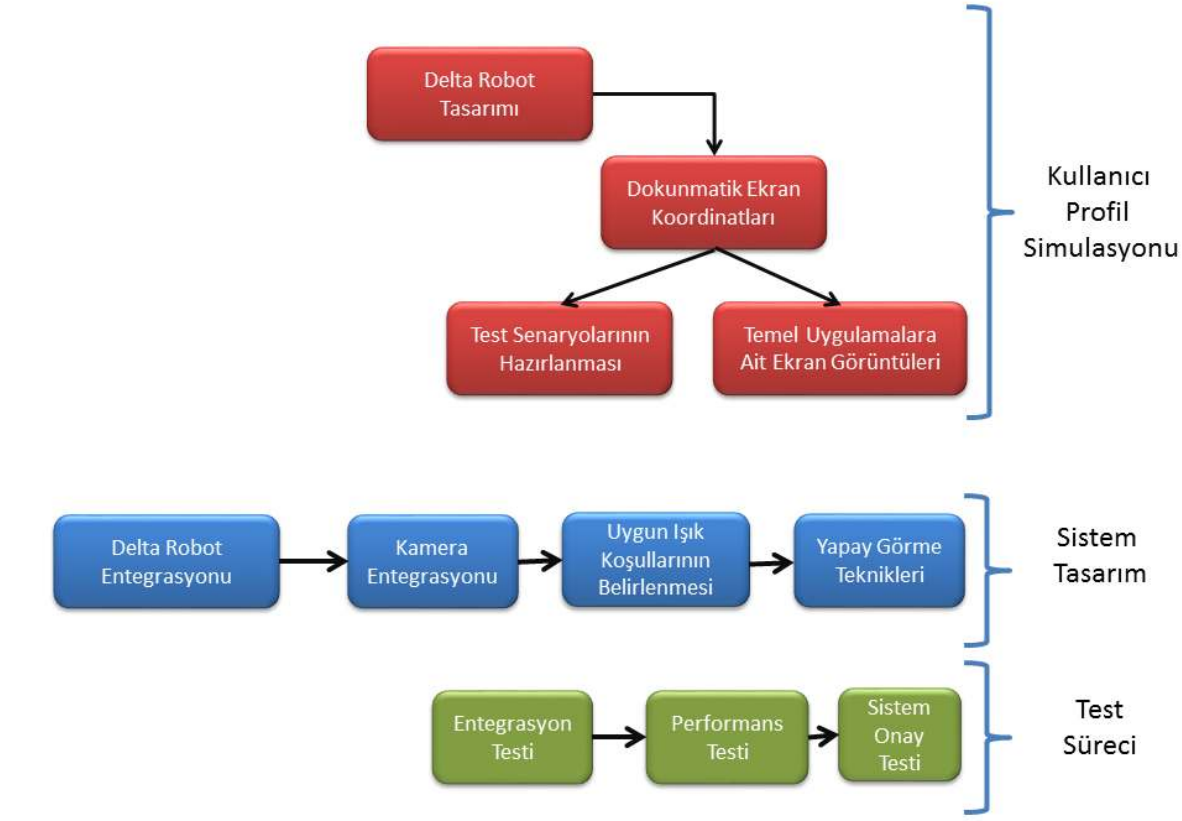
- **Şablon Eşleme:** Bir görüntü içinde, verilen bir şablonun görüntü içinde eşleşen parçalarının bulunması işlemidir. Karşılaştırılacak şablon önceden tanımlanmalıdır. Aşağıdaki şekilde görüldüğü gibi tanımlanan şablon görüntü içinde karşılaştırılarak bulunmuştur. Şekil 4.9'da bir şablon eşleme örneği verilmiştir.



Şekil 4.9 Yapay görmede şablon eşleştirilmesi [44]

5. DENEY DÜZENEĞİNİN OLUŞTURULMASI

Test otomasyon sisteminin oluşturulma aşamaları Şekil 5.1'de gösterilmiştir. Bu aşamaların içinde ilk olarak delta robotun tasarımı ve monte edilecek platformun üretilmesi gerekmektedir. Sistemde kullanılacak kameraya karar verilmesi ve uygun lens seçimi bir sonraki adımı oluşturur. Sistemin test işlemlerinde kullanılması için bazı temel test senaryolarının hazırlanması bir sonraki aşamayı oluşturmaktadır. Test senaryolarının düzgün bir şekilde çalıştığının kontrolü için kamera tarafından gelen görüntünün yapay görme algoritmaları ile doğrulanması bir diğer aşamayı oluşturmaktadır.



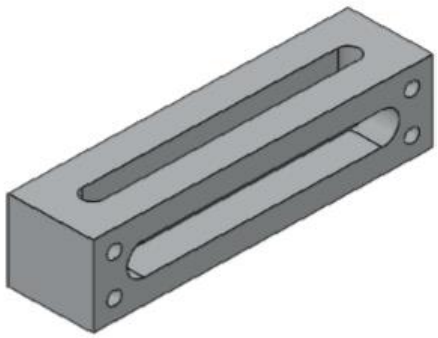
Şekil 5.1 Test otomasyon sisteminin oluşturulma aşamaları

5.1 Robot Platform Tasarımı

Delta robotlar, 6 eksenli manipülatörler ve kartezyen robotlardan farklı olarak asılması için bir platforma ihtiyaç duyarlar. Bu nedenle delta robotların tasarımında önce ilk olarak uygun platform tez kapsamında tasarlanmış ve üretilmiştir.

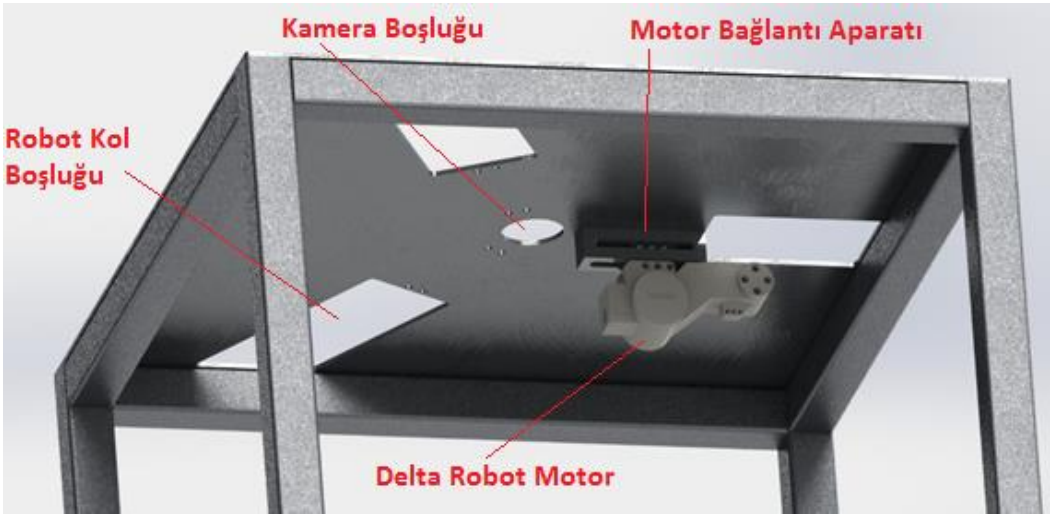
Platform üretiminde robotun ağırlığının taşınması ve hareket kabiliyetinin engellenmemesi dikkat edilecek hususlardır. Bu nedenle platform üretiminde 30x30 mm boyutlarında sigma profiller kullanılmıştır. Platform boyutları robotun çalışma uzayı göz önüne alındığı için 50x50x50 cm olacak şekilde ayarlanmıştır.

Delta robota ait motorların montajı için ise robot platformuna ek bir parça daha imal edilmiştir. Üretilen parçaya bağlanacak motorların merkeze olan uzaklığı 10 - 15 cm arasında bir kızak vasıtası ile değiştirilebilmektedir. Şekil 5.2'de üretilen bu montaj elemanı görülmektedir.



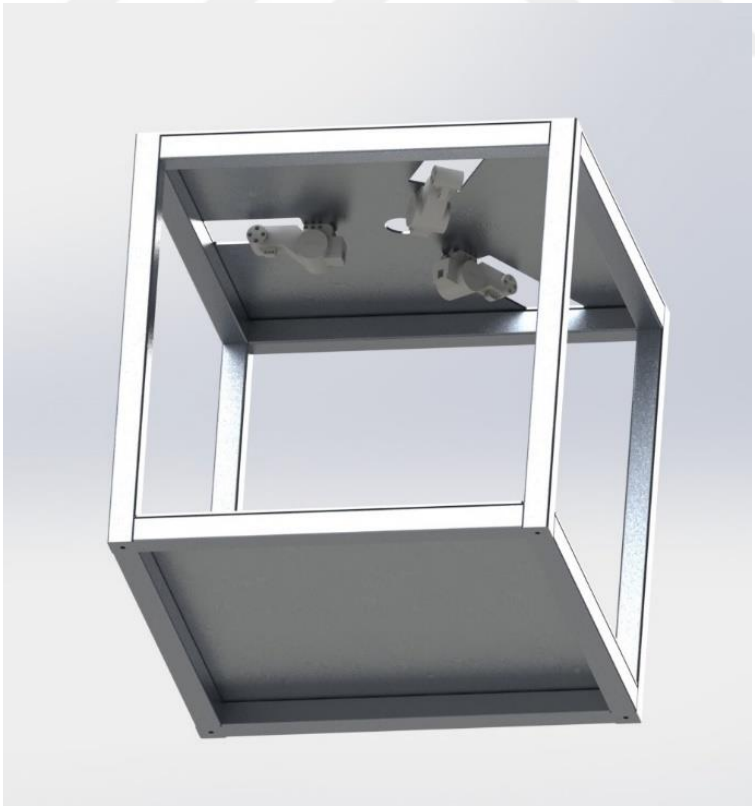
Şekil 5.2 Platform ile motorlar arasındaki kızaklı bağlantı elemanı

Tasarlanacak robota ait kolların rahatça çalışabilmesi için platformun üzerinde kolların yukarı ve aşağı hareketi için uygun boşluklar bırakılmıştır. Bu sayede çalışma sırasında robot yukarı yönde herhangi bir engelle karşılaşmayacaktır. Şekil 5.3'de robot platformunun son hali görülmektedir. Platformun üst kısmının merkezinde 4 cm çapında dairesel bir boşluk bırakılmıştır. Açılan bu boşluğa sistemde kullanılacak kamera yerleştirilmiştir. Bu sayede üzerinde işlem yapılacak dokunmatik ekrana kamera tarafından doğrudan bir görüş açısı sağlanmıştır.



Şekil 5.3 SolidWorks ortamında platform ile motor bağlantısının gösterimi

Yukarıda belirtilen özelliklere uygun robot platformu SolidWorks programında hazırlanmış ve üretilmiştir. Tasarlanan delta robota uygun platformun son hali Şekil 5.4'de verilmiştir.



Şekil 5.4 SolidWork ortamında motor bağlantılarının son hali

5.2 Delta Robot Tasarımı

Delta robot tasarımında pratik anlamda birçok farklı tasarıma olanak veren bir malzeme olan LEGO® Mindstorms® NXT 2.0 elemanları kullanılmıştır. Hem malzemelerin kolaylıkla temini hem de birleştirilmesinde kolaylık sağlaması açısında tercih edilmiştir.

Kullanılan lego seti içinde bulunan elemanlar şu şekildedir;

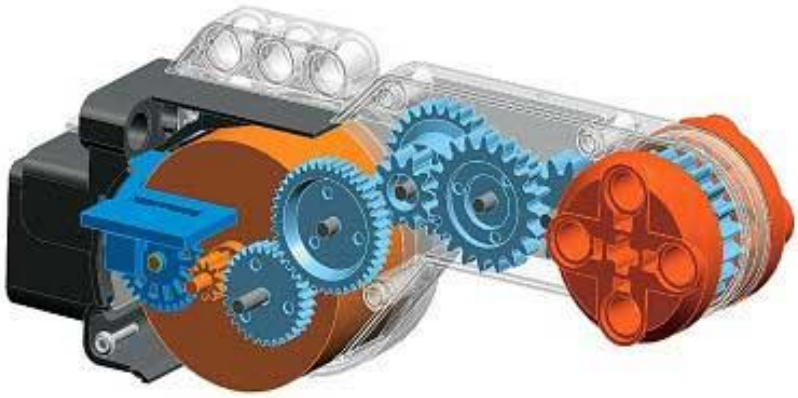
- 1 adet akıllı NXT lego birimi: 32 bit bir işlemciye sahiptir, 4 giriş ve 3 çıkışı bulunmaktadır, Bluetooth veya USB üzerinden haberleşebilmektedir.
- 3 adet servo motor
- 1 adet ultrasonik sensör, 2 adet temas sensörü, 1 adet renk sensörü
- Farklı uzunluk ve boyda lego bağlantı elemanları

Şekil 5.5 LEGO® Mindstorms® NXT 2.0 [45]

NXT akıllı birimi MATLAB®/Simulink, LabVIEW, Microsoft Visual Studio gibi birçok uygulama geliştirme ortamıyla haberleşebilmektedir. Tez kapsamında uygulama geliştirme ortamı olarak MATLAB® programı tercih edilmiştir. MATLAB® üzerinde bulunan yapay görme uygulamaları arabirimleri bu tercihte etkili olmuştur.

MATLAB® üzerinde doğrudan NXT akıllı birimini programlamaya uygun bir kütüphane bulunmamaktadır. Bu nedenle MATLAB® programına entegre şekilde çalışabilen RWTH Aachen Üniversitesinde geliştirilen açık kaynak kodlu bir kütüphane kullanılmıştır. MATLAB® üzerinde NXT Lego setini programlamaya olanak veren bu kütüphane, sensörlerin ve motorların kontrolü için birçok özellik barındırmaktadır.

Tez kapsamında motorların kontrölü kritik bir öneme sahiptir. Kullanılan kütüphane içinde motor kontrolü fonksiyonları verilen açıya ve hıza göre motoru hareket ettirebilme kabiliyetine sahiptir. NXT Lego motoru PID (Proportional Integral Derivative) kontrol yöntemiyle sürülmektedir. Ayrıca motor içinde bulunan bir enkoder ile konum bilgisi alınmakta ve PID kontrolüne geri besleme olarak verilmektedir. Bu şekilde verilen açıya uygun olarak hassas bir motor hareketi sağlanmaktadır. Kullanılan motorların hassasiyeti +/- 1 derece ve maksimum torku 50 Ncm olarak belirtilmiştir. Delta robot tasarımında 3 adet NXT Lego motoru kullanılmıştır.



Şekil 5.6 NXT motorun iç yapısı [46]

Delta robot tasarımında bir diğer önemli parametre motorların merkeze uzaklıkları ve kollara ait uzaklıklarıdır. Bu parametreler delta robota ait ters kinematik denklemlerin hesaplanmasında büyük önem taşımaktadır. Daha önceki çalışmalar baz alınarak uygun robot ölçüleri üretilen robot platformuna uygun olacak şekilde hesaplanmıştır [47]. Sabit platform ile aktif kol arasındaki açı negatif değerlere geçmeyecek şekilde sınırlandırılmıştır. Bu sayede kolun platfom tavanına çarpması engellenmiştir. Tasarlanan robota ait uzunluklar şekildedir;

- Sabit platform : 125 mm
- Bağımlı kol : 310 mm
- Aktif kol : 100 mm
- Hareketli platform : 30 mm

Kullanılan motorlarda sistemin daha stabil olması sağlamak ve mevcut tork miktarını arttırmak için bir dişli yapısı sisteme eklenmiştir.

Şekil 5.7 Motorlara ait dişli yapısı

- 1. dişlinin çapı (R1) : 25 mm
- 2. dişlinin çapı (R2) : 40 mm

$$\frac{R1}{R2} = \frac{T1}{T2} \tag{4.3}$$

Yukarıdaki verilen dişli tork hesabı kullanılarak hesaplama yapıldığında motor tarafından üretilen torkun robot kolda 1.6 katına çıktığı hesaplanmaktadır. Bununla birlikte motor devir miktarı aynı oranda düşmektedir. Delta robotlarda hareket esnasında robot kol açı değişimlerinin küçük olması sebebiyle robot hareketinde negatif bir etkisi yoktur.

Sistemde Grey Point firmasının BlackFly U305SC2 model bir kamerası kullanılmıştır. Kameranın boyut olarak küçük olması ve çözünürlüğünün yeterli olması sebebiyle robot platformunda tercih edilmiştir. Kamera USB 3.0 arayüzüne sahiptir, bu sayede enerjilendirme ve iletişim aynı birim üzerinden yapılabilmektedir.

Şekil 5.8 BlackFly U305SC2 kamerası [48]

Kameranın sahip olduğu özellikler şu şekildedir;

Tablo 5.1 BlackFly U305SC2 kamera özellikleri

Çözünürlük	808 x 608
Frame Sayısı	50 FPS
Megapiksel	0.5 MP
Renk Berraklığı	Renkli
Sensör İsmi	Sony ICX693
Sensör Modeli	CCD
Pozlama Çeşidi	Global Pozlama
Sensör Formatı	1/3"
Güç Gereksinimi	5-24V DC veya 5 V USB 3.0
Arabirim	USB 3.0

Kamera üzerinde Fujinon YV2.8×2.8SA-2 model bir lens kullanılmıştır. Lense ait özellikler aşağıda verilmiştir.

Şekil 5.9 Fujinon YV2.8×2.8SA-2 lensi [49]

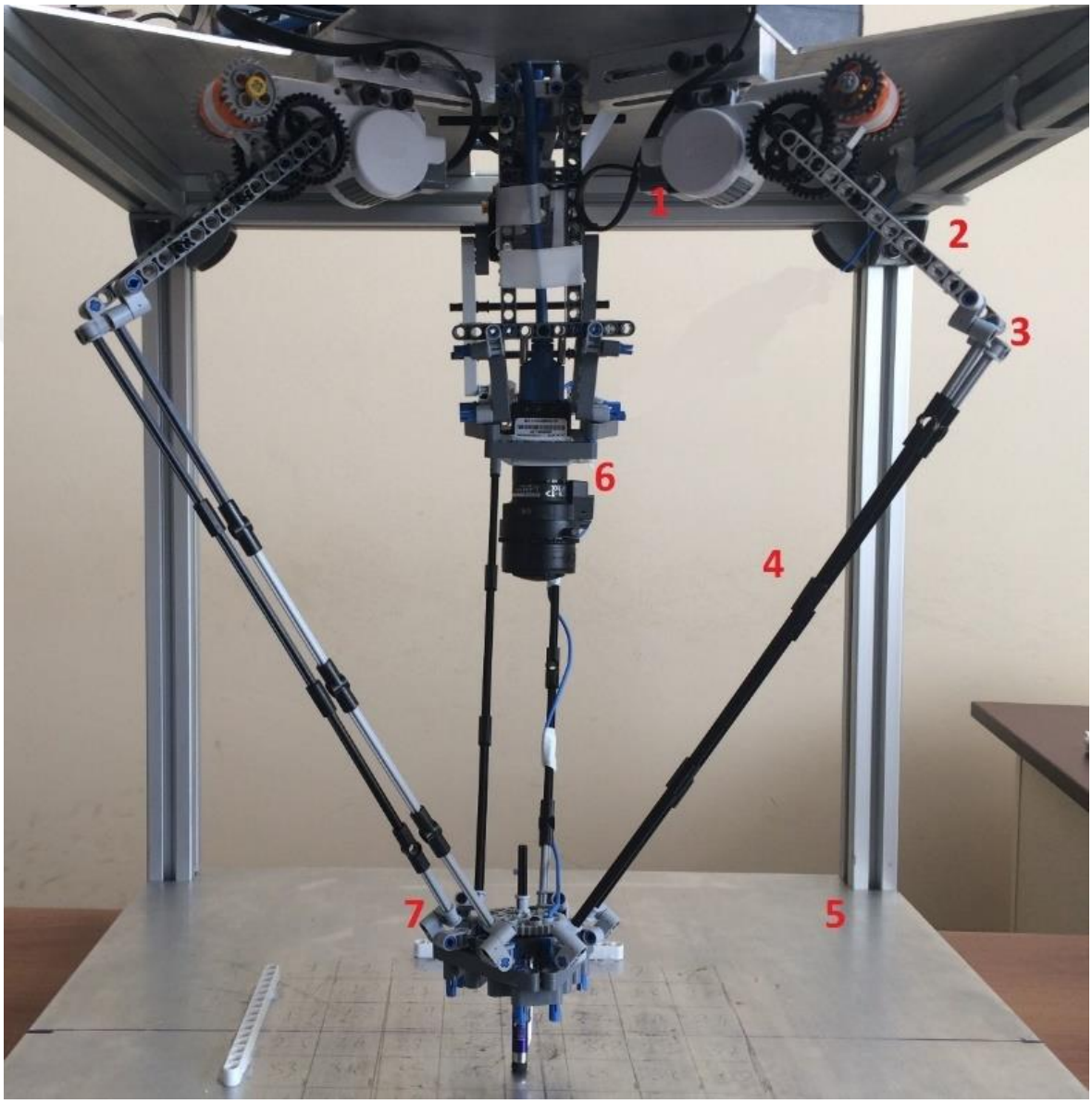
Tablo 5.2 Fujinon YV2.8×2.8SA-2 lens özellikleri

Model	Fujinon YV2.8×2.8SA-2
Odak uzunluğu	2.8mm-8mm
Optik Formatı	1/3"

Tasarlanan robotun son hali Şekil 5.10'daki gibidir.

1. Delta robot motorları
2. Robot aktif kol
3. Robot kol eklem yapısı

4. Robot bağımlı kol
5. Robot platformu
6. Sistemde bulunan kamera
7. Hareketli platform



Şekil 5.10 Deney düzeneğinin son hali

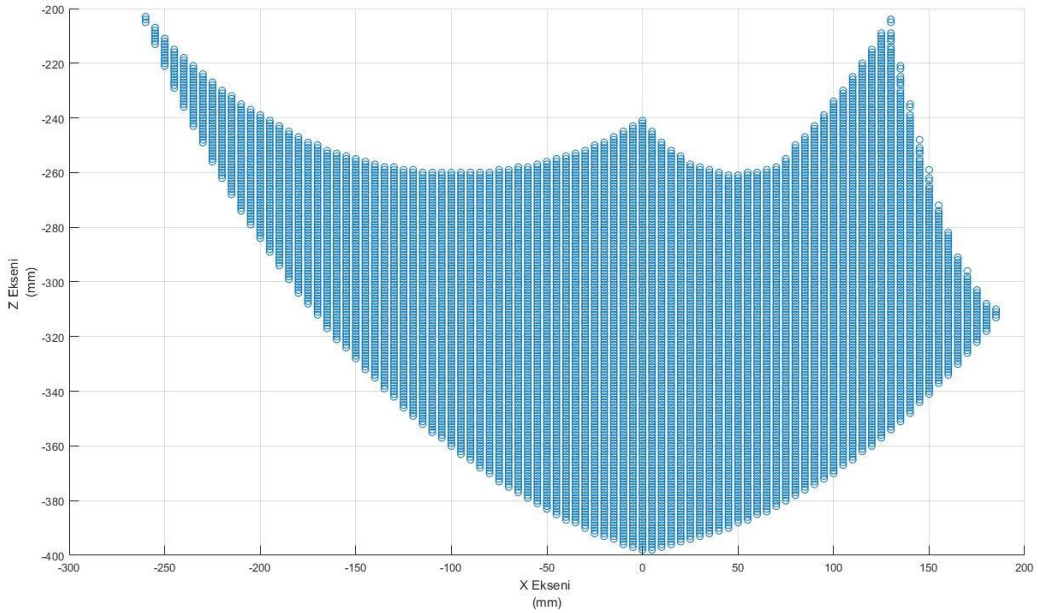
5.3 Delta Robot Çalışma Uzayı

Tasarlanan robota ait çalışma uzayı aralığı MATLAB® üzerinde oluşturulan bir arama algoritması kullanılarak bulunmuştur. Arama algoritmasında belirli aralıkta

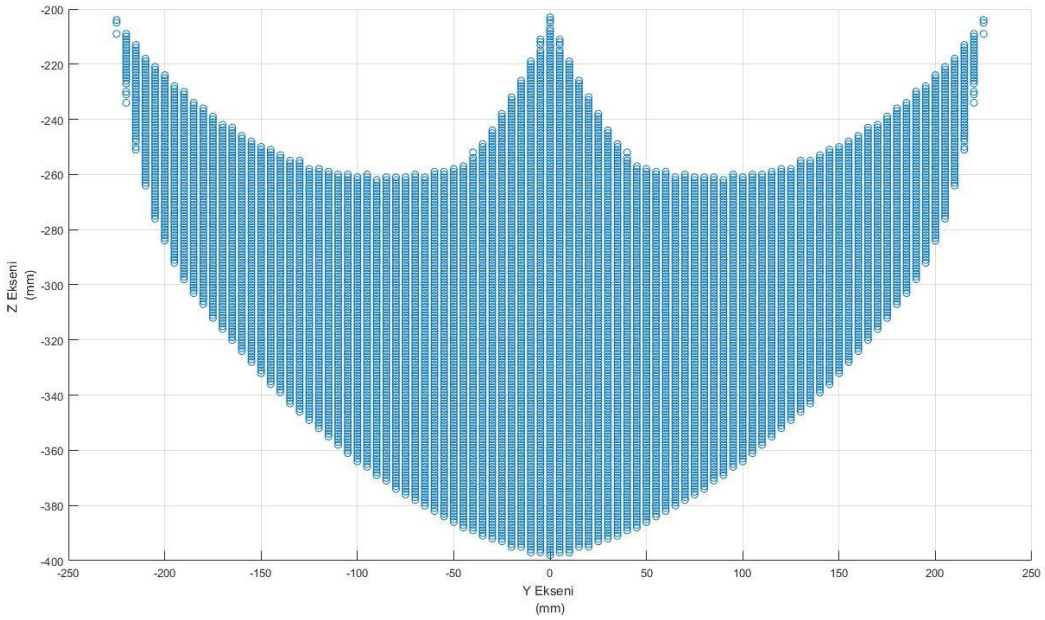
verilen x,y,z eksenlerinde koordinatların ters kinematik denklemlerle açısal değerleri bulunmuş ve bulunan açıların 0 ve 100 dereceleri arasında olan noktalarla çalışma uzayı oluşturulmuştur. Bulunan çalışma uzayında eksenlerin çalışma aralıkları şu şekildedir;

- X ekseninde 180 mm ile -266 mm arasında
- Y ekseninde 225 mm ile -225 mm arasında
- Z ekseninde -203 mm ile -398 mm arasında

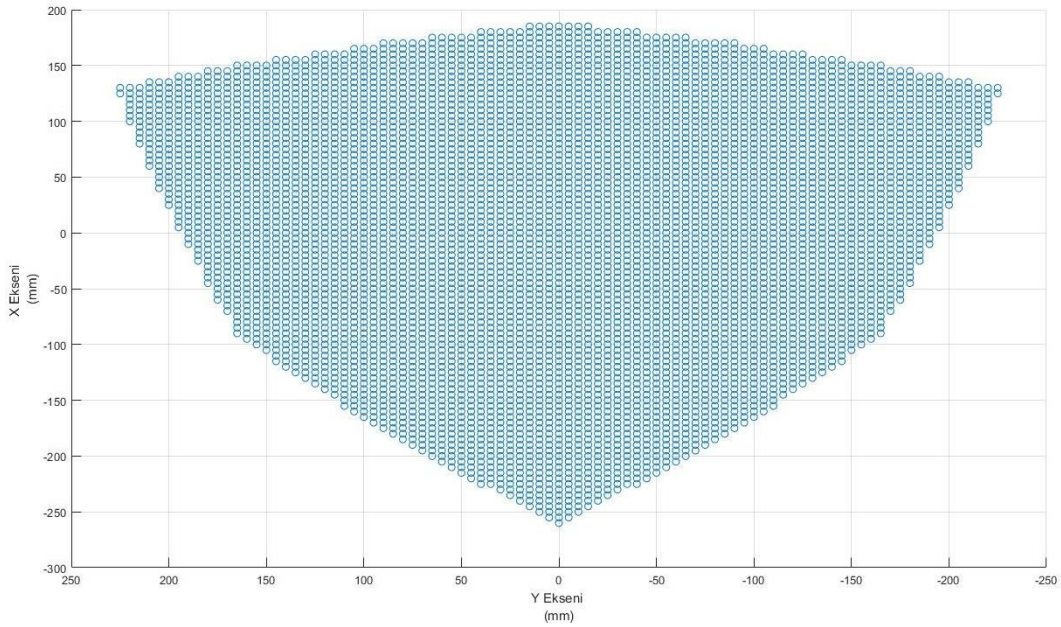
Çalışma uzayının kartezyen koordinat sisteminde eksenlerinin birbirleri arasındaki ilişkileri Şekil 5.11, 5.12 ve 5.13'de verilmiştir.



Şekil 5.11 X-Z eksenlerine ait çalışma uzayı



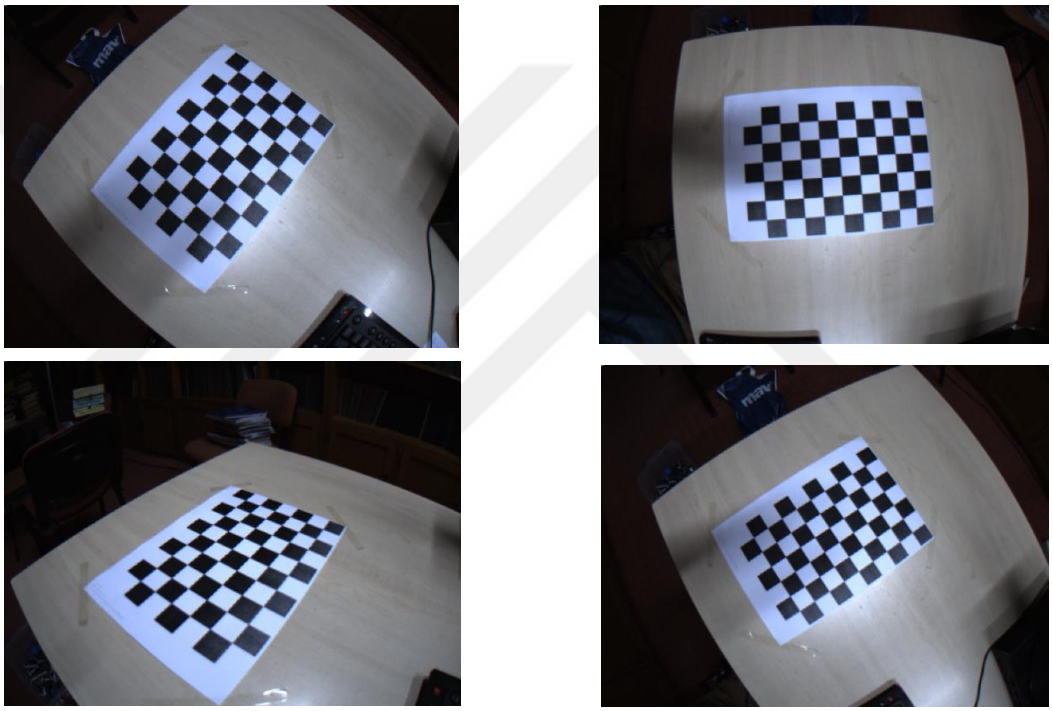
Şekil 5.12 Y-Z eksenlerine ait çalışma uzayı



Şekil 5.13 X-Y eksenlerine ait çalışma uzayı

5.4 Kamera Kalibrasyonu ve Yapay Görme Algoritmaları

Yapay görme uygulamalarında kullanılacak kameranın sistemde kullanılmasından önce kalibre edilmesi gerekmektedir. Kamera kalibrasyonu kameranın içsel, dışsal ve lensten kaynaklanan bozulmaların belirlenmesi işlemidir. Bulunan parametreler ile kameranın görüntüsü iyileştirilmektedir. Bu konuda en çok kullanılan yöntemlerden biri dama tahtası (checkerboard) deseni kullanılarak yapılan kalibrasyondur [50]. Bu yöntemde en az 10, en fazla 20 farklı açı ve yön olmak üzere kameraya desenin görüntüsü alınır.



Şekil 5.14 Kamera kalibrasyonunda kullanılan farklı açılardan çekilmiş görüntüler

Ardından MATLAB® üzerindeki "cameraCalibrator" uygulaması ile alınan görüntüler değerlendirilir. Bu işlemin ardından kameraya ait olan parametreler çıkarılır ve görüntünün düzeltilmesinde kullanılmaktadır. Şekil 5.15 ve 5.16'da kalibrasyon işlemi öncesinde ve sonrasında alınan görüntüler verilmiştir.

Şekil 5.15 Doğrudan kameradan alınan görüntü

Şekil 5.16 Doğrudan kameradan alınmış görüntünün kalibre edildikten sonraki hali

Test otomasyon sisteminde kullanılan yapay görme algoritmalarından biri şablon eşleştirme algoritmasıdır. Yapılan işlemlerin ardından ekran üzerinden alınan görüntü ile daha önceden belirlenen şablonların karşılaştırılması büyük önem taşımaktadır. Bu nedenle sistemde normalize 2-D çapraz korelasyon (NCC) yöntemi kullanılmıştır [51,52]. Algoritmanın adımları şu şekildedir;

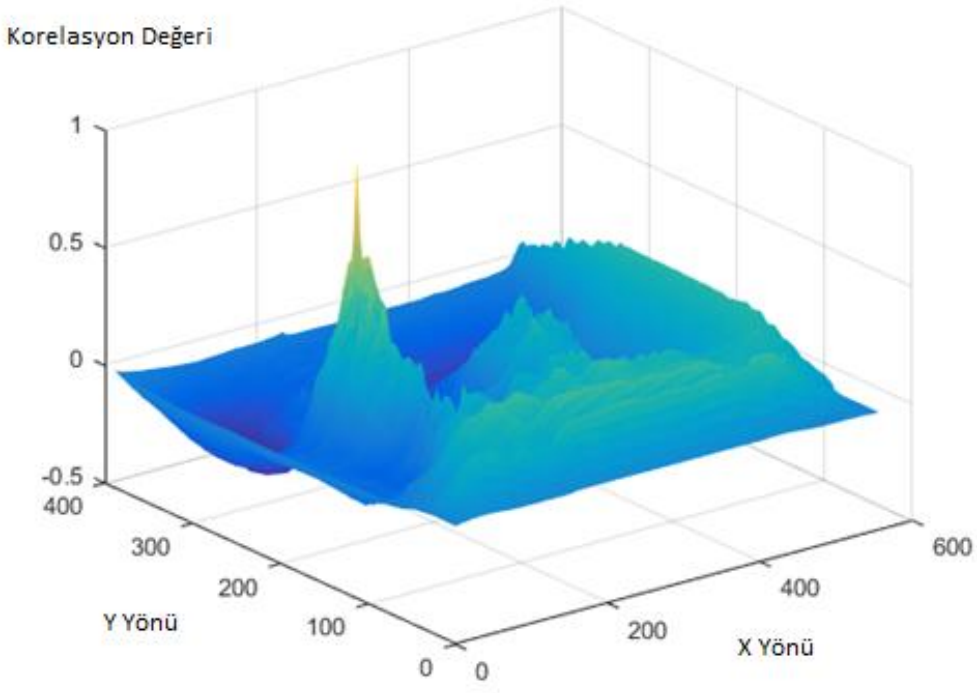
- Çapraz korelasyonun hesaplanması (görüntünün büyüklüğüne göre uzamsal veya frekans uzayında bu hesap yapılabilir)
- Belli bir alandaki değerlerin toplanması
- Korelasyon katsayısının bulunması için belli bölgede toplanan değerlerin kullanılması

Aşağıda yapay görme algoritmasına ait örnek bir uygulama verilmiştir. Alınan ekran görüntüsü ile daha önceden belirlenen şablon karşılaştırılmış ve uygun bölgenin korelasyon katsayısı hesaplanmıştır. Şekil 5.17'de verilen şablon Şekil 5.18'de alınan ekran görüntüsü içinde eşleştirilmiştir.

Şekil 5.17 Facebook uygulaması şablonu

Şekil 5.18 Facebook uygulaması ekranı

Hesaplanan korelasyon değeri 0.9985 olarak bulunmuştur, Şekil 5.19'da yüksek korelasyon değerine sahip bölge görülmektedir.



Şekil 5.19 NCC uygulamasının ardından çıkan korelasyon değerleri

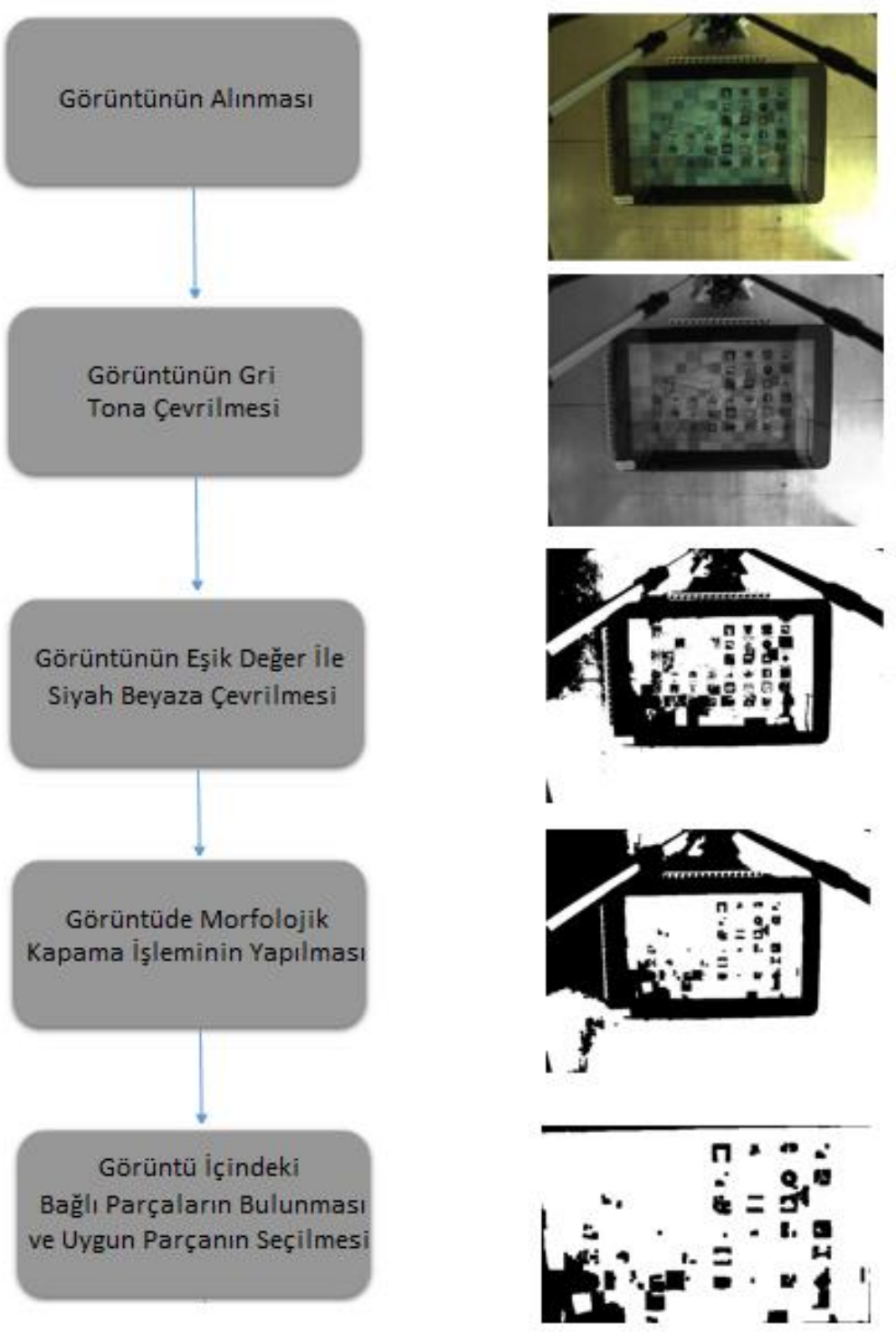
Bulunan bölge Şekil 5.20'de görüldüğü gibi çerçeve içerisine alınmıştır.

Şekil 5.20 Facebook uygulaması şablon eşleştirme işleminin sonucu

Kullanılan bir diğer yapay görme algoritması ise dokunmatik ekranın boyutundan bağımsız olarak yerinin bulunmasını sağlayan bir segmentasyon algoritmasıdır. Bu algoritma ile sabit bir konumu olmayan ve boyut olarak farklı cihazların platform üzerinde nerede bulunduğu kamera tarafından tespit edilebilmektedir. Test otomasyon sistemi için segmentasyon adımları şu şekildedir;

- Görüntü alınır ve gri tonlara çevrilir.
- Görüntü normalize edilir ve 0.3 eşik değeri ile filtrelenir.
- Morfolojik olarak 5 piksel genişliğinde bir kare eleman ile kapama işlemi yapılır.
- Görüntü içindeki bağlı elamanlar (connected component) bulunur. Elemanlar içinde düşey uzunluğunun, yatay uzunluğa oranı 0.8 ve 0.6 arasındaki elemanlar dışındakiler elenir. Kalan elemanlar içinde alan olarak en büyük eleman alınır. Alınan elemanın sahip olduğu çerçeve ilk resimden çıkarılır.

Şekil 5.21'de segmentasyon işlemine ait adımlar verilmiş ve sonuçlarıyla gösterilmiştir. Şekil 5.22'de işlemin ardından elde edilen görüntü gösterilmiştir.



Şekil 5.21 Segmentasyon işleminin blok diyagramı ve elde edilen ekran görüntüleri

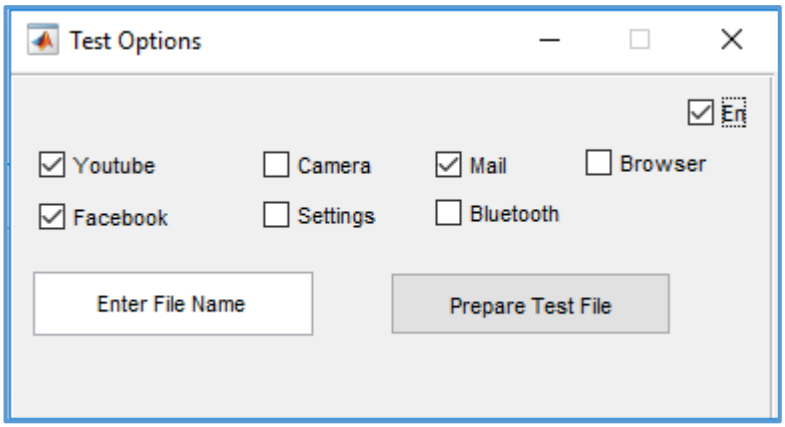
Şekil 5.22 Segmentasyon işlemi ardından elde edilen ekran görüntüsü

5.5 Test Senaryolarının Oluşturulması

Tez kapsamında tasarlanan delta robot tarafında dokunmatik ekranlı cihazlar üzerinde test yapılabilmesi için test senaryolarının hazırlanması gerekmektedir. Test senaryosu oluşturulması şu aşamalardan oluşmaktadır.

1. Robotun ekran üzerinde dokunacağı yerlerin koordinatlarının çıkarılması
2. Robotun yapacağı her işlem için uygun ekran görüntülerinin alınması
3. Alınan ekran görüntüleri üzerinde belirleyici şablonların çıkarılması

Test senaryolarının oluşturulması için MATLAB® ortamında bir arayüz hazırlanmıştır. Bu sayede arayüz içinden seçilen test uygulamaları için bir XLSX formatında dosya oluşturmaktadır. Hazırlanan test arayüzü istenirse Türkçe veya İngilizce olarak farklı dillerde kullanılabilmektedir.



Şekil 5.23 Test senaryosu oluşturulması için tasarlanan MATLAB GUI

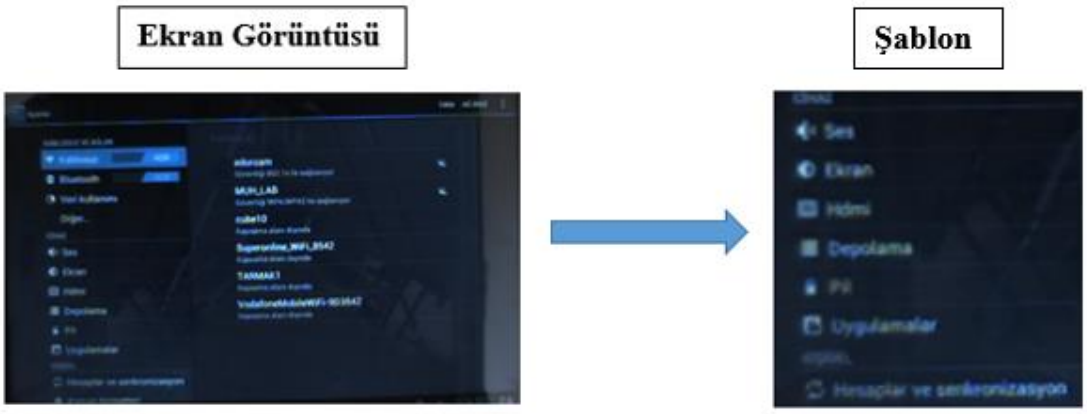
Hazırlanan koordinatlar, şablonlar ve sonraki adımın hangi durum olacağı otomatik olarak bir XLSX formatında excel dosyasına girilmektedir. Test edilecek cihaza ait test senaryosu tüm durumlar için ayrı ayrı dosyaya yazılması gerekmektedir. Excel dosyası içindeki yerleşim planı tablo 5.3'de görülmektedir.

Tablo 5.3 Test senaryosuna ait XLSX dosyası içeriği

X Ekseni (cm)	Y Ekseni (cm)	Z Ekseni (mm)	Şablon Dosyası
16,4	6,25	-345	youtube.bmp
-1	-1	-1	menu.bmp
18,3	9,2	-342	facebook.bmp
-1	-1	-1	menu.bmp
5,5	13,6	-340	tarayici.bmp
-1	-1	-1	menu.bmp
10,5	9,5	-345	mail.bmp
-1	-1	-1	menu.bmp
10,5	11,5	-345	bluetooth.bmp
-1	-1	-1	menu.bmp
14,3	13,3	-345	kamera.bmp
-1	-1	-1	menu.bmp
18	12,5	-340	ayarlar.bmp
-1	-1	-1	menu.bmp

Test sırasında kullanılacak şablonları hazırlanması için test ekranı üzerinden ekran görüntüsü alınması gerekmektedir. Ardından karşılaştırılacak anlamlı parçanın ekran görüntüsü içinden alınması ve ayrı bir dosyada saklanması işlemi gerleştirilmektedir.

- Ayarlar ekran görüntüsü şekil 5.24'deki gibidir. Buradan ayarlar ekranına özel bir şablon şu şekilde seçilmiştir.



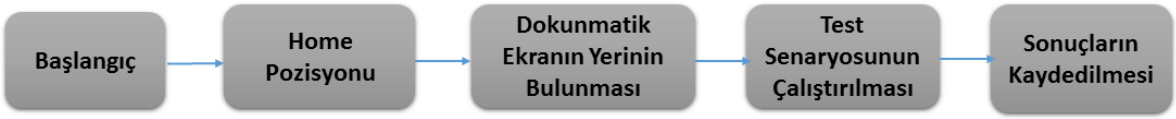
Şekil 5.24 Ayarlar ekranı ve kullanılan şablon

5.6 Test Otomasyon Sisteminin Çalışma Yapısı

Gerçekleştirilecek test işlemi belirlendikten sonra sisteme yüklenir, ardından sistemin enerjilendirilmesi yapılır. Robotik sistem ilk olarak home (başlangıç) pozisyonuna gider ve Şekil 5.25'de görüldüğü gibi kol açısı sistem ile paralel bir konuma geldiğinde robot motorlarının enkoderlarını sıfırlar. Bu aşamadaki adımlar Şekil 5.26'da gösterilmiştir.

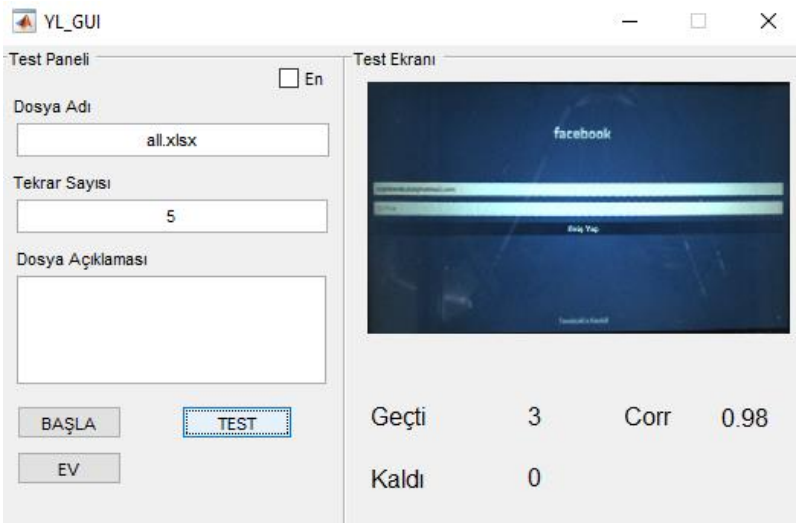
Şekil 5.25 Başlangıç pozisyonu

Test sırasında farklı boyutta dokunmatik ekranlı cihazların kullanılabilmesi için sistem ekran yerini otomatik olarak kamera ile tespit etmektedir. Bu segmentasyon işleminin ardından test senaryosu verilen döngü sayısı kadar çalıştırılmaktadır. Robot işleminin bitmesinin ardından test sonuçları bir XLSX formatında bir dosyaya kayıt edilmektedir.



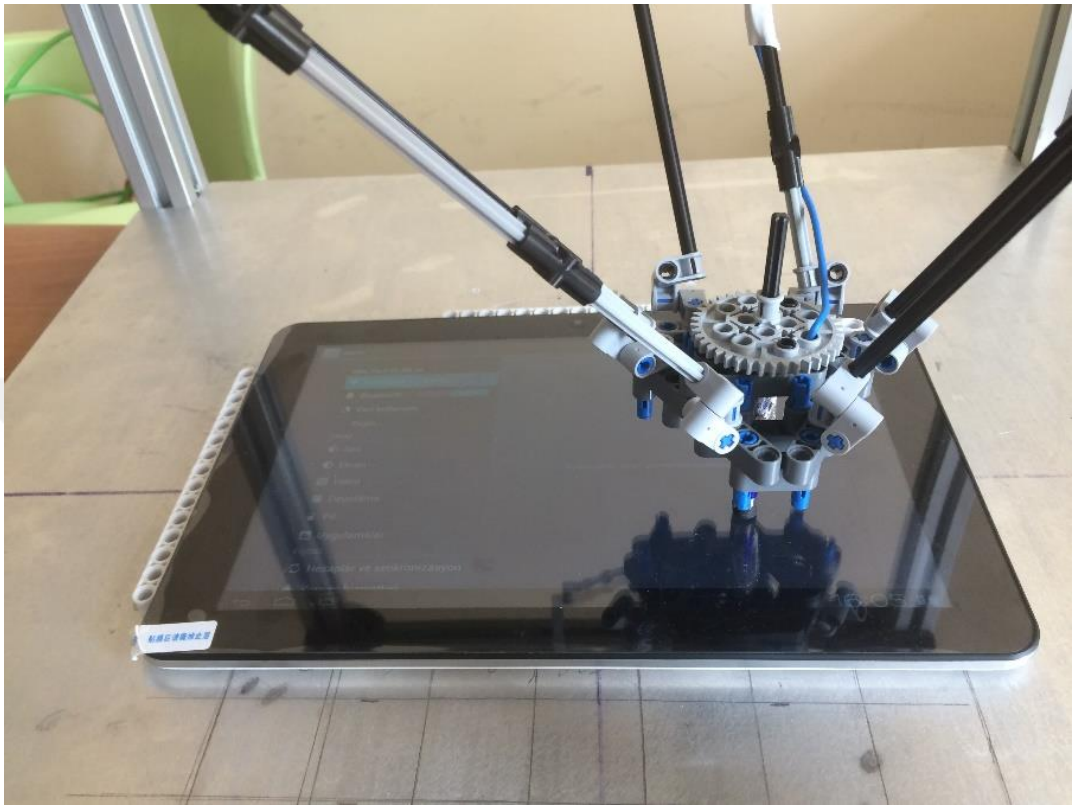
Şekil 5.26 Test işlemi aşamaları

Test uygulamasının yapılabilmesi için MATLAB® ortamında bir arayüz tasarlanmıştır. Bu arayüz ile test robotunun start alması, test robotunun başlangıç durumuna getirilmesi ve testin başlatılması için bir adet buton yerleştirilmiştir. Testin başlatılması için test XLSX dosyasının adının girilmesi, yapılacak testin tekrar sayısının verilmesi, verilerinin girilmesi gerekmektedir. Test sırasında anlık ekran görüntüleri görülebilmektedir. Bunun yanında başarılı ve başarısız test sayıları görülebilmekte ve her test adımının sonuçları bir dosyaya yazılmaktadır. Şekil 5.27'de tasarlanan arayüz gösterilmiştir.



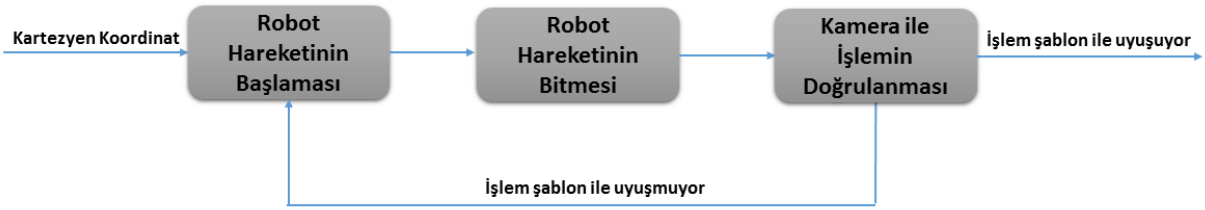
Şekil 5.27 MATLAB GUI test arayüz ortamı

Robotun ekran üzerinde yapacağı işlemler Şekil 5.28'de görüldüğü gibi 2 temel adımdan oluşmaktadır. İlk adım robotun verilen koordinata gitmesi ve basma işlemini gerçekleştirmesidir.



Şekil 5.28 Basma işleminin gerçekleştirilmesi

İkinci aşama ise yapılan işlemin ardında kamera ile ekranın kontrolünün yapılması ve işlemin doğrulanmasıdır. Eğer işlem doğrulanmaz ise önceki koordinatlara giderek işlem tekrarlanmaktadır. Bu süreç işlemin doğrulanmasına kadar sürmektedir. Bu sayede sistemde kamera ile bir geri besleme işlemi yapılmaktadır.

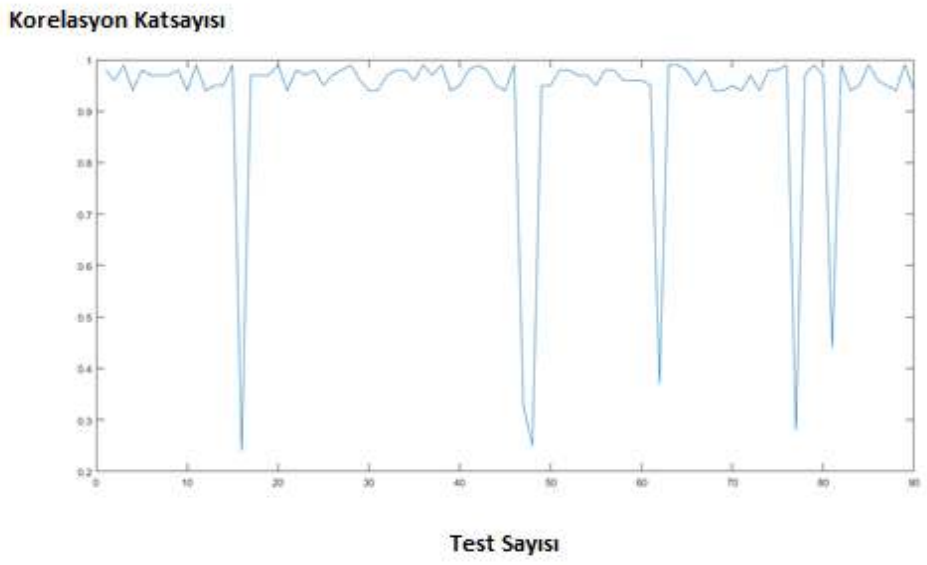


Şekil 5.29 Robot hareket aşamaları

6. SONUÇLAR VE ÖNERİLER

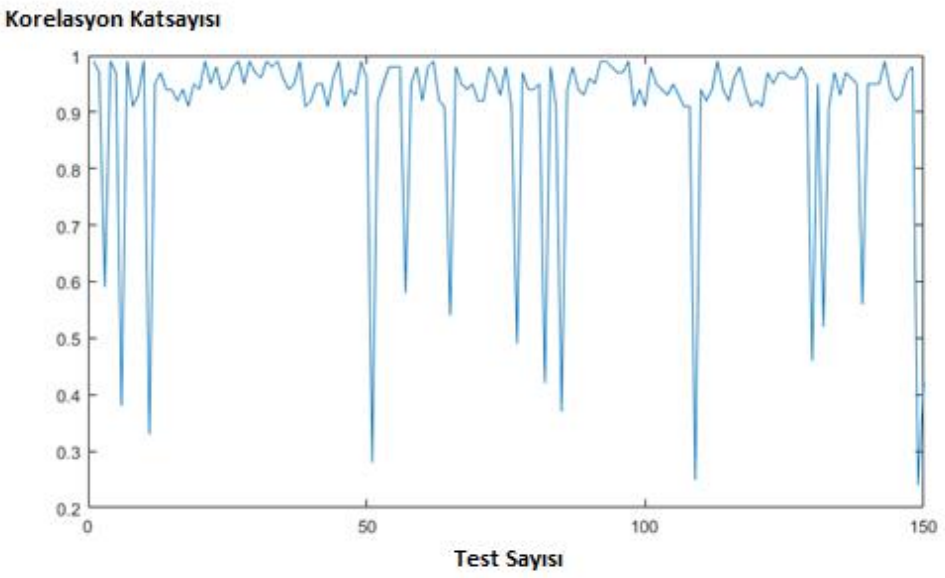
Bu çalışmada dokunmatik ekran cihazlar için bir test otomasyon platformu geliştirilmiştir. Geliştirilen platformda kullanılmak üzere bir delta robot tasarlanmıştır. Tasarlanan robota ait fiziksel uzunluklar, robotun hareket uzayının büyüklüğüne göre çıkarılmıştır. Ayrıca robota ait kinematik dönüşümler hesaplanmış ve sistemde kullanılmıştır. Robota ait tüm parçalar LEGO® Mindstorms® NXT 2.0 kullanılarak hazırlanmış. Sistemde kullanılmak üzere uygun test senaryoları hazırlanmıştır ve sistemde bazı temel testlere ait denemeler yapılmıştır. Ayrıca sisteme eklenen kamera aracılığı ile yapılan testlerin doğrulukları daha önceden hazırlanmış şablonlar ile karşılaştırılarak onaylanmıştır. Sistemdeki tüm elemanların kontrolü MATLAB® geliştirme ortamında yapılmıştır. Kurulan deney düzeneğinin çalışma performansının incelenmesi için bazı fonksiyonel testler gerçekleştirilmiştir.

- 1. Test senaryosu için youtube, facebook ve ayarlar olmak üzere 3 adet uygulama seçilmiştir. Uygulamaların açılıp açılmadığına dair kontrol kamera ile karşılaştırılmıştır. Test kapsamındaki senaryoda bu uygulamaların 30 kere açılıp kapanması sağlanmıştır. Eşik değer 0.85 olarak seçildiği için test sonucu olarak 84 defa başarılı, 6 defa başarısız bir açma kapama işlemi gerçekleşmiştir. Bu teste ait şablonların korelasyon katsayılarının dağılımı Şekil 6.1'de gösterilmektedir. Test adımlarına ait korelasyon ortalaması 0.9234 olarak bulunmuştur ve test işlemindeki başarı oranı % 93.33 olarak hesaplanmıştır.



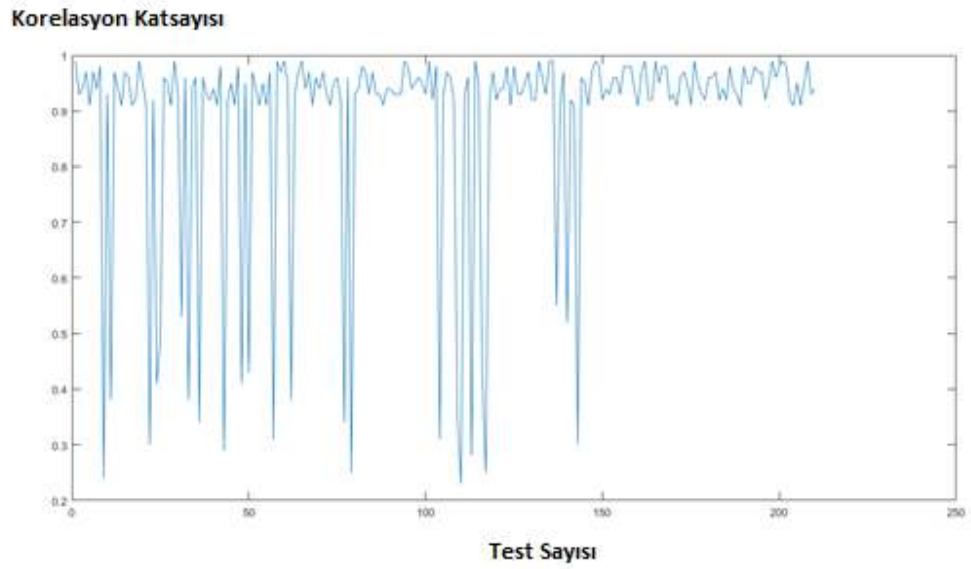
Şekil 6.1 1.teste ait korelasyon sonuçları

- 2. Test senaryosunda ayarlar, kamera, mail, bluetooth, tarayıcı olmak üzere 5 adet uygulama seçilmiştir. Test kapsamında senaryonun 30 kere açılıp kapanması sağlanmıştır. Eşik değer 0.85 olarak seçildiği için test sonucu olarak 134 defa başarılı, 16 defa başarısız bir açma kapama işlemi gerçekleşmiştir. Bu teste ait şablonların korelasyon katsayılarının dağılımı Şekil 6.2'de gösterilmektedir. Test adımlarına ait korelasyon ortalaması 0.9002 olarak bulunmuş ve test işlemindeki başarı oranı % 89.33 olarak hesaplanmıştır.



Şekil 6.2 2. teste ait korelasyon sonuçları

- 3. Test senaryosunda youtube, bluetooth, ayarlar, kamera, mail, bluetooth, tarayıcı olmak üzere 7 adet uygulama seçilmiştir. Test kapsamında senaryonun 30 kere açılıp kapanması sağlanmıştır. Eşik değer 0.85 olarak seçildiği için test sonucu olarak 186 defa başarılı, 24 defa başarısız bir açma kapama işlemi gerçekleşmiştir. Bu teste ait şablonların korelasyon katsayılarının dağılımı Şekil 6.3'de gösterilmektedir. Test adımlarına ait korelasyon ortalaması 0.8817 olarak bulunmuştur ve test işlemindeki başarı oranı % 88.57 olarak hesaplanmıştır.



Şekil 6.3 3. teste ait korelasyon sonuçları

Tablo 6.1 Test senaryolarına ait sonuçlar

Test Adı	**Tekrar Sayısı**	**Başarılı Test**	**Başarısız Test**	**Başarı Yüzdesi %**	**Korelasyon Yüzdesi %**
Youtube	50	46	4	92	91.66
Facebook	50	45	5	90	86.22
Mail	50	45	5	90	85.04
Ayarlar	50	44	6	88	85.64
Bluetooth	50	47	3	94	87.04
Kamera	50	43	7	86	85.76
Tarayıcı	50	47	3	94	87.80
1.senaryo	90	84	6	93.33	92.34
2.senaryo	150	134	16	89.33	90.02
3.sanaryo	210	186	24	88.57	88.17
Toplam	**800**	**721**	**79**	**90.12**	**87.94**

Bu çalışma ile tasarlanan test otomasyon sisteminin dokunmatik ekranlı cihazların performans testlerinde başarılı bir şekilde çalışabildiği gösterilmiştir. Özellikle testlerin hızlı ve kararlı bir şekilde yapılabilmesinde delta robotlu sistemlerin uygun olduğu görülmüştür. Tez kapsamında yapılan delta robota ait mekanik özellikler kullanılan malzeme nedeniyle hızlı gerçekleştirilen işlemlerde test sırasında sorun yaratabilmektedir. Bu sorunun önüne geçilebilmesi için robota ait kollar ve yürüyen platformun daha dayanıklı bir malzemeden seçilmesi gerekir. Ayrıca test platformunda bulunan kameranın çözünürlüğünün daha yüksek olması ile ekran üzerinden daha detaylı bilgi elde edilebilir ve farklı yapay görme algoritmaları kullanılabilir. Bunun

yanında çözünürlüğün artması ile daha küçük ekranlar üzerinde işlem yapılabilir. Bu çalışmada tasarlanan sistem aynı zamanda ilgili konulardaki mühendislik eğitimlerinde kullanılabilecek yapıdadır.

KAYNAKLAR

1. Koll S. M., Wattenhofer R, Welten S. A Personal Touch - Recognizing Users Based on Touch Screen Behavior, PhoneSense '12 Proceedings of the Third International Workshop on Sensing Applications on Mobile Phones, Article No. 1,2012

2. S. Shyam Sundar , Saraswathi Bellur , Jeeyun Oh , Qian Xu & Haiyan Jia, User Experience of On-Screen Interaction Techniques: An Experimental Investigation of Clicking, Sliding, Zooming, Hovering, Dragging, and Flipping, Human–Computer Interaction, 2014

3. Selvam R, Mobile Software Testing – Automated Test Case Design Strategies. International Journal on Computer Science and Engineering (IJCSE), April 2011, 3(4) 1450–1461s

4. Hwangbo, H., Yoon S., Jin, B., Han, Y., Ji, Y., A Study of Pointing Performance of Elderly Users on Smartphones, International Journal of Human-Computer Interaction, 2013, 29:9, 604-618s

5. Šindler, V., Uríček, J., Bulej, V., Delta robots - robots for high speed manipulation, Tehnicki Vjesnik, September 2011, 18(3):435-445s

6. Wang, Z., Bovik, A. C., Sheikh H. R., Simoncelli, E. P., Image quality assessment: From error visibility to structural similarity, IEEE Transactions on Image Processing, Apr. 2004, vol. 13, no. 4, 600-612s

7. Mulikita M., Mobile Application Testing, Serminar Informatik, 21.06.2012, 25s

8. Kim H., Choi B., Wong E., Performance Testing of Mobile Applications at the Unit Test Level, 2009 Third IEEE International Conference on Secure Software Integration and Reliability Improvement, 171-180s

9. Elbaum, S., Diep, M., Profiling Deployed Software: Assessing Strategies and Testing Opportunities , IEEE Transaction on Software Engineering, April 2005, vol. 31, no. 4, 312-327s

10. Brooks, P., Memon, A., Automated GUI Testing Guided by Usage Profiles, 22nd IEEE/ACM Int. Conf. Automated Software Engineering, 2007, 333-342s

11. Reger, G., Barringer, H., Rydeheard, D., A pattern-based Approach to Parametric Specification Mining, Proc. 28nd IEEE/ACM Int. Conf. Automated Software Engineering, 2013, 658-663s

12. Canfora, G., Mercaldo, F., Visaggio, C., A Case Study of Automating User Experience-Oriented Performance Testing on Smartphones, IEEE Int. Conf. Software Testing, Verification and Validation 2014, 66-69s

13. Dhanapal, K., Deepak, K., Sharma S., An Innovative System for Remote and Automated Testing of Mobile Phone Applications, Service Research and Innovation, Institute Global Conference, 2012, 44-54s

14. Kanstren, T., Aho, P., Lämsä A., Robot-Assisted Smartphone Performance Testing, 7th IEEE International Conference on Technologies for Practical Robot Applications (TEPRA'15) 2015

15. Ohmann, T., Herzberg, M., Fiss, S., Halbert, A., Behavioral Resource-Aware Model Inference, Proc. 29th IEEE/ACM Int'l. Conf. on Automated Software Engineering (ASE), 2014, 19-30s

16. Test Strategies for Smartphones and Mobile Devices, Macadamian Technologies, 2010

17. Zhifang, L., Bin, L., Xiaopeng, G., Test Automation on Mobile Device, Proceedings of the 5th Workshop on Automation of Software Test, AST '10, 2010,1-7s

18. Jensen, C. S., Prasad, M. R., Moller, A., Automated Testing with Targeted Event Sequence Generation, ISSTA ’13, 2013 July, 15–20s

19. Nyman, N., In Defense of Monkey Testing, Test Automation SIG Group, 2002, 34s

20. Arlt, S., Bertolini, C., Pahl, S., Schäf, M., Trends in Modelbased GUI Testing Advances in Computers, 2012, 86:183–222s

21. Li, K., Wu, M., Effective GUI Testing Automation: Developing an Automated GUI Testing Tool. Wiley, Hoboken, NJ, 2006, 445s

22. Takala, T., Katara, M., Harty J., Experiences of system-level modelbased GUI testing of an Android application, Software Testing, Verification and Validation (ICST), 2011 IEEE Fourth International Conference on, 377–386s

23. Paiva, A. C. R., Automated Specification Based Testing of Graphical User Interfaces. PhD thesis, 2007.

24. Xie, Q., Memon, A. M., Model-based testing of community-driven open-source GUI applications. In Software Maintenance, 2006. ICSM’06. 22nd IEEE International Conference on, IEEE, 2006, 145–154s

25. Xie, Q., Developing cost-effective model-based techniques for GUI testing, Proceedings of the 28th international conference on Software engineering, ACM, 2006, 997– 1000s

26. Moreira, R., Memon, A., A Pattern-Based Approach for GUI Modeling and Testing, Proceedings of the 24th annual International Symposium on Software Reliability Engineering (ISSRE 2013), 2013

27. Anders Toxboe. Design patterns. http://ui-patterns.com/patterns/, 2014

28. Lin, X., Lu, L., Su K., Yan, Y., An Approach of GUI Test Automation on Mobile Device, International Journal of Digital Content Technology and its Applications(JDCTA), 2013

29. Merlet, J.P., Parallel robots. Springer Science & Business Media, 2006, 401s

30. Delta Robot https://en.wikipedia.org/wiki/Delta_robot

31. Angeles, J., Fundamentals of Robotic Mechanical Systems: Theory, Methods and Algorithms, Springer, New York, 2003, 523s

32. Staicu, S., Carp-Ciocardia, D.C., Dynamic Analysis of Clavel's Delta Parallel Robot, IEEE International Conference on Robotics & Automation, 2003, 4116-4121s

33. Zsombo-Murray, P.J., An Improved Approach to the Kinematics of Clavel's DELTA Robot, McGill University, Department of Mechanical Engineering, Centre for Intelligent Machines, 2009.

34. Li, Y., Xu, Q., Dynamic Analysis of a Modified DELTA Parallel Robot for Cardiopulmonary Resuscitation, Department of Electromechanical Engineering, Faculty of Science and Technology University of Macau, 2005.

35. Staicu, S., Carp-Ciocardia, D.C., Dynamics of Delta Parallel Robot with Linear Actuators, IEEE International Conference on Mechatronics, 2005, 870-875s

36. Laribi, M.A., Romdhane, L., Zeghloul, S., Analysis and dimensional synthesis of the DELTA robot for a prescribed workspace, Mechanism and Machine Theory, 2007, 42, 859–870s

37. http://fanucrobotics.com

38. Khatib, O., Siciliano, B., Handbook of Robotics, Springer, 2008, 1611s

39. Zhang, D., Parallel Robotic Machine Tools, Springer, 2010, 219s

40. Training on Digital Image Processing Fundamentals Concept & Techniques in MATLAB, http://www.solutions4u-asia.com/emailc/digitalimageprocessing.html

41. Gonzalez, R.C., Woods, R.E., Digital Image Processing Second Edition, Prentice Hall, 2002, 827s

42. Illumination techniques for industrial image processing, http://www.stemmer-imaging.co.uk/en/technical-tips/illumination-techniques/

43. Moeslund, T. B., Introduction to Video and Image Processing Building Real Systems and Applications, Springer, 2012, 227s

44. Template Matching, http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html

45. https://images-na.ssl-images-amazon.com/images/I/71m0WJtJoiL._SL1500_.jpg

46. http://www.philohome.com/nxtmotor/motor1_3.jpg

47. Li, M., Bi, D., Xiao, Z., Mechanism Simulation and Experiment of 3-DOF Parallel Robot Based on MATLAB, International Power, Electronics and Materials Engineering Conference (IPEMEC 2015), 489-494s

48. http://www.edmundoptics.de/images/catalog/1006832.jpg

49. https://www.ptgrey.com/content/images/thumbs/0002592_fujinon-yv2828sa-2-28mm-8mm-13-cs-mount-lens_772.jpeg

50. Zhang, Z., A flexible new technique for camera calibration, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22(11):1330-1334s

51. Lewis, J. P., Fast Template Matching, Vision Interface, 1995, 120-123s

52. Haralick, Robert M., Shapiro, L. G., Computer and Robot Vision, Volume II, Addison-Wesley, 1992, 316-317s

EKLER

EK A. (Test Programı MATLAB kodları)

```
function varargout = YL_GUI(varargin)
% YL_GUI MATLAB code for YL_GUI.fig
%      YL_GUI, by itself, creates a new YL_GUI or raises the
existing
%      singleton*.
%
%      H = YL_GUI returns the handle to a new YL_GUI or the handle
to
%      the existing singleton*.
%
%      YL_GUI('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in YL_GUI.M with the given input
arguments.
%
%      YL_GUI('Property','Value',...) creates a new YL_GUI or raises
the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before YL_GUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to YL_GUI_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help YL_GUI

% Last Modified by GUIDE v2.5 09-Nov-2016 16:13:11

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @YL_GUI_OpeningFcn, ...
                   'gui_OutputFcn',  @YL_GUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before YL_GUI is made visible.
function YL_GUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to YL_GUI (see VARARGIN)

% Choose default command line output for YL_GUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes YL_GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the command line.
function varargout = YL_GUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
COM_CloseNXT all
warning('off','all');
```

```
imaqreset;

global map;
%%%%%%%%%%%%%%%
map=zeros(48,3);

map(1,:)=[80 -135 -320];
map(2,:)=[80 -110 -320];
map(3,:)=[80 -80  -320];
map(4,:)=[80 -50  -340];
map(5,:)=[80 -5 -340];
map(6,:)=[80 30 -340];
map(7,:)=[80 70 -340];
map(8,:)=[80 97 -340];

map(9,:)=[42 -133 -330];
map(10,:)=[50 -110 -330];
map(11,:)=[52 -77  -340];
map(12,:)=[56 -39  -340];
map(13,:)=[56 0 -340];
map(14,:)=[56 40 -340];
map(15,:)=[56 75 -340];
map(16,:)=[53 100 -340];

map(17,:)=[10 -135 -330];
map(18,:)=[10 -110 -340];
map(19,:)=[10 -75  -340];
map(20,:)=[10 -40  -340];
map(21,:)=[10 10 -340];
map(22,:)=[10 50 -340];
map(23,:)=[10 75 -340];
map(24,:)=[10 110 -340];

map(25,:)=[-25 -130 -340];
map(26,:)=[-25 -105 -340];
map(27,:)=[-25 -77  -340];
map(28,:)=[-25 -37  -340];
map(29,:)=[-25 10 -340];
map(30,:)=[-25 45 -340];
map(31,:)=[-25 80 -340];
map(32,:)=[-25 107 -340];

map(33,:)=[-50 -130 -330];
map(34,:)=[-57 -105 -340];
map(35,:)=[-57 -73  -340];
map(36,:)=[-57 -33  -340];
map(37,:)=[-60 5 -340];
map(38,:)=[-60 40 -340];
map(39,:)=[-55 80 -340];
map(40,:)=[-50 107 -340];

map(41,:)=[-85 -130 -330];
map(42,:)=[-95 -105 -340];
map(43,:)=[-90 -73  -340];
map(44,:)=[-100 -33  -340];
map(45,:)=[-100 5 -340];
map(46,:)=[-90 40 -340];
map(47,:)=[-90 77 -340];
map(48,:)=[-90 100 -330];
```

```
D=imread('C:\Users\yigit\Desktop\matlab_tez_gui\blank.jpg');
imshow(D,'Parent',handles.axes2);

%%%%%%%%%%%% Kamerayı aç %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load('C:\Users\yigit\Desktop\matlab_tez_gui\grey_calib.mat');

global vid;

vid = videoinput('pointgrey');

vid.FramesPerTrigger = 1;
vid.TriggerRepeat = Inf;
triggerconfig(vid,'manual');

start(vid);
trigger(vid)
%%%%%%%%%%%% NULL %%%%%%%%%%
for j=1:150
   frame = getsnapshot(vid);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global h;
global mA;
global mB;
global mC;
global mA_data;
global mB_data;
global mC_data;
global map;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%  ROBOT KOL BOSLUGA GIDER %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mA_data = mA.ReadFromNXT();
mB_data = mB.ReadFromNXT();
mC_data = mC.ReadFromNXT();

[q1,q2,q3] = InvDelta(135,-20,-320);
q1=ceil(q1*-1*1.4);
q2=ceil(q2*-1*1.4);
q3=ceil(q3*-1*1.4);

speed=12;

a_position = -1*q1 - mA_data.Position;
if(a_position > 0 )
    mA.Power = speed;%ceil(a_position/2);
else
    mA.Power = -speed;%ceil(a_position/2);
end
mA.TachoLimit = abs(a_position)+1;
```

```
b_position = -1*q2 - mB_data.Position;
if(b_position>0)
    mB.Power = speed;
else
    mB.Power = -speed;
end
mB.TachoLimit = abs(b_position)+1;

c_position = -1*q3 - mC_data.Position;
if(c_position>0)
    mC.Power = speed;
else
    mC.Power = -speed;
end
mC.TachoLimit = abs(c_position)+1;

mA.SendToNXT();
mB.SendToNXT();
mC.SendToNXT();

pause(1);
mA.WaitFor();
mB.WaitFor();
mC.WaitFor();

pause(2);
mA.Stop('brake');
mB.Stop('brake');
mC.Stop('brake');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% SEGMENTASYON %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global vid;
load('C:\Users\yigit\Desktop\matlab_tez_gui\grey_calib.mat');

frame = getsnapshot(vid);
frame_un = undistortImage(frame, cameraParams);
grey=rgb2gray(frame_un);
BW = im2bw(grey, 0.3);

SE = strel('square',5);
BW2 = imclose(BW,SE);

s  =
regionprops(BW2,'ConvexArea','BoundingBox','MajorAxisLength','MinorA
xisLength');

for i=1:size(s,1)
    if(s(i).ConvexArea>90000 && s(i).ConvexArea<150000 &&
(s(i).MinorAxisLength/s(i).MajorAxisLength)<0.8)
       %%%%Frame ROI %%%%%
       img_roi = s(i).BoundingBox;
      break;
   end
end

%%%%%%%%%% EKRANDA GÖSTER %%%%%%%%%%%%%%%%%%%%%%%%%
frame = getsnapshot(vid);
```

```
frame_un = undistortImage(frame, cameraParams);
frame_crop = imcrop(frame_un,img_roi);
imshow(frame_crop,'Parent',handles.axes2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
filename = get(handles.edit1,'String');
repeat = get(handles.edit2,'String');

[num,txt] = xlsread(filename);

pass=0;
fail=0;

for k=1:str2num(repeat)
    for i=1:size(num,1)

    image_txt = strjoin(txt(i,1));


    x = num(i,1);
    y = num(i,2);
    z = num(i,3);

    if x == -1

        x_new = -103;
        y_new = -94;
        z = -334;

        for j=1:6

            mA_data = mA.ReadFromNXT();
            mB_data = mB.ReadFromNXT();
            mC_data = mC.ReadFromNXT();

            speed=14;

            if j==1
               [q1,q2,q3] = InvDelta(0,0,-290);
            elseif(j==2)
               [q1,q2,q3] = InvDelta(y_new+10,x_new,-300);
            elseif(j==3)
               [q1,q2,q3] = InvDelta((y_new),x_new,z);
            elseif(j==4)
               [q1,q2,q3] = InvDelta(y_new+10,x_new,-300);
            elseif(j==5)
               [q1,q2,q3] = InvDelta(0,0,-290);
            else
               [q1,q2,q3] = InvDelta(140,0,-320);
            end

            q1=ceil(q1*1.4);
            q2=ceil(q2*1.4);
            q3=ceil(q3*1.4);

            a_position = q1 - mA_data.Position;
            if(a_position > 0 )
                mA.Power = speed;
```

```
            else
                mA.Power = -speed;
            end
            mA.TachoLimit = abs(a_position)+1;

            b_position = q2 - mB_data.Position;
            if(b_position>0)
                mB.Power = speed;
            else
                mB.Power = -speed;
            end
            mB.TachoLimit = abs(b_position)+1;

            c_position = q3 - mC_data.Position;
            if(c_position>0)
                mC.Power = speed;
            else
                mC.Power = -speed;
            end
            mC.TachoLimit = abs(c_position)+1;

            mA.SendToNXT();
            mB.SendToNXT();
            mC.SendToNXT();

            mA.WaitFor();
            mB.WaitFor();
            mC.WaitFor();
        end

    else
        x1=floor((x+1.5)/3);
        x2=floor((x+1.5)/3)+1;

        y1=floor((y+1.5)/3);
        y2=floor((y+1.5)/3)+1;

        blok = (y1-1) * 8 + x1;

        ratio_x=(x-(x1*3-1.5))/3;
        ratio_y=(y-(y1*3-1.5))/3;

        x_new=(map(blok+1,2)-map((blok),2))*ratio_x+map((blok),2);

        y_new=(map((blok+8),1)-map((blok),1))*ratio_y+map((blok),1);

        for j=1:5

            mA_data = mA.ReadFromNXT();
            mB_data = mB.ReadFromNXT();
            mC_data = mC.ReadFromNXT();

            if j==1
                [q1,q2,q3] = InvDelta(0,0,-280);
            elseif j==2
                [q1,q2,q3] = InvDelta((y_new+10),x_new,-290);
            elseif(j==3)
                [q1,q2,q3] = InvDelta(y_new,x_new,z);
            elseif(j==4)
```

```
            [q1,q2,q3] = InvDelta((y_new+10),x_new,-290);
        else
            [q1,q2,q3] = InvDelta(140,0,-320);
        end

        q1=ceil(q1*1.4);
        q2=ceil(q2*1.4);
        q3=ceil(q3*1.4);

        speed=20;

        a_position = q1 - mA_data.Position;
        if(a_position > 0 )
            mA.Power = speed;
        else
            mA.Power = -speed;
        end
        mA.TachoLimit = abs(a_position)+1;

        b_position = q2 - mB_data.Position;
        if(b_position>0)
            mB.Power = speed;
        else
            mB.Power = -speed;
        end
        mB.TachoLimit = abs(b_position)+1;

        c_position = q3 - mC_data.Position;
        if(c_position>0)
            mC.Power = speed;
        else
            mC.Power = -speed;
        end
        mC.TachoLimit = abs(c_position)+1;

        mA.SendToNXT();
        mB.SendToNXT();
        mC.SendToNXT();

        mA.WaitFor();
        mB.WaitFor();
        mC.WaitFor();
    end
end

pause(3);

frame = getsnapshot(vid);
frame_un = undistortImage(frame, cameraParams);
frame_crop = imcrop(frame_un,img_roi);
imshow(frame_crop,'Parent',handles.axes2);

a=rgb2gray(frame_crop); %%%% ana resim
b=rgb2gray(imread(image_txt));
c = normxcorr2(b,a);
set(handles.text11, 'String', num2str(max(c(:)),2));
%excel_data(index,:)={'18','12,5','-340','ayarlar.bmp'};
if max(c(:)) > 0.8
   pass=pass+1;
```

```
        set(handles.text7, 'String', int2str(pass));
    else
        fail=fail+1;
        set(handles.text8, 'String', int2str(fail));
    end




%     [ypeak, xpeak] = find(c==max(c(:)));
%     yoffSet = ypeak-size(b,1);
%     xoffSet = xpeak-size(b,2);
%     figure
%     hAx  = axes;
%     imshow(a,'Parent', hAx);
%     imrect(hAx, [xoffSet+1, yoffSet+1, size(b,2), size(b,1)]);

    end
end
%      frame = getsnapshot(vid);
%      J1 = undistortImage(frame, cameraParams);
%      cor = imcrop(J1,img_roi);
%      grey=rgb2gray(cor);
%      J = histeq(grey);

stop(vid);
clear('pointgrey');


%1< --- Executes during object creation, after setting all
properties.
function axes2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate axes2


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%  Home %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global h;
global mA;
global mB;
global mC;
global mA_data;
global mB_data;
global mC_data;

h = COM_OpenNXT();

COM_SetDefaultNXT(h);
```

```
mA = NXTMotor('A', 'Power', -20, 'SpeedRegulation', true);
mB = NXTMotor('B', 'Power', -20, 'SpeedRegulation', true);
mC = NXTMotor('C', 'Power', -20, 'SpeedRegulation', true);
mA.Stop('off');
mB.Stop('off');
mC.Stop('off');

mA.ActionAtTachoLimit = 'Holdbrake';
mA.SmoothStart = true;

mB.ActionAtTachoLimit = 'Holdbrake';
mB.SmoothStart = true;

mC.ActionAtTachoLimit = 'Holdbrake';
mC.SmoothStart = true;


pause(1);
mA.SendToNXT();
mB.SendToNXT();
mC.SendToNXT();
count = 0;
flag = 0;
pause(4);
while(1)
   mA_data = mA.ReadFromNXT();
   mB_data = mB.ReadFromNXT();
   mC_data = mC.ReadFromNXT();

   if( (count+5)<mB_data.TachoCount)
       mA.Stop('brake');
       mB.Stop('brake');
       mC.Stop('brake');
       pause(1);
       mA.ResetPosition();
       mB.ResetPosition();
       mC.ResetPosition();
       break;
   end

   count=mB_data.TachoCount;
   pause(0.1);

 end

 pause(1);


% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 if get(handles.checkbox1, 'Value') == true %%English

     set(handles.text2, 'String', 'File Name');
     set(handles.text3, 'String', 'Repeat Number');
     set(handles.text4, 'String', 'File Description');
```

```
        set(handles.text5, 'String', 'PASS');
        set(handles.text6, 'String', 'FAIL');
        set(handles.pushbutton1, 'String', 'START');
        set(handles.pushbutton2, 'String', 'TEST');
        set(handles.pushbutton3, 'String', 'HOME');
        set(handles.uipanel1, 'Title', 'Test Panel');
        set(handles.uipanel2, 'Title', 'Test Screen');

    else
        set(handles.text2, 'String', 'Dosya Adı');
        set(handles.text3, 'String', 'Tekrar Sayısı');
        set(handles.text4, 'String', 'Dosya Açıklaması');
        set(handles.text5, 'String', 'GEÇTİ');
        set(handles.text6, 'String', 'KALDI');
        set(handles.pushbutton1, 'String', 'BAŞLAT');
        set(handles.pushbutton2, 'String', 'TEST');
        set(handles.pushbutton3, 'String', 'EV');
        set(handles.uipanel1, 'Title', 'Test Paneli');
        set(handles.uipanel2, 'Title', 'Test Ekranı');

    end
% Hint: get(hObject,'Value') returns toggle state of checkbox1
```

EK B. (Kinematik Fonksiyonu MATLAB kodları)

```
function [q1,q2,q3] = InvKin (x,y,z)
% Compute inverse kinematics (using iterative method)
% [Q1,Q2,Q3] = INVKIN(X,Y,Z)
% X,Y,Z = Given Cartesian coordinates
% Q1,Q2,Q3 = Corresponding joint angles in degree

%close all
%hold on
%view(3)
%grid on
%axis ([-200 200 -200 200 0 400])

% x,y,z is the destination of platform's centor of gravity (C.G.)
point.
% Note : Suggest to use Lower Configuration to avoid collision.

% Setup length (mm)
L  = 125 ;              % Base
l  = 30 ;           % Platform
l1 = 90 ;               % Link1
l2 = 280 ;              % Link2

% Motor1
a1 = x^2 + y^2 + z^2 + 2*y*l/sqrt(3) - 2*y*L/sqrt(3) + l^2/3 -
2*l*L/3 + L^2/3 + l1^2 - l2^2 ;
b1 = 2*y*l1 + 2*l*l1/sqrt(3) - 2*l1*L/sqrt(3) ;
c1 = 2*z*l1 ;
% Select Configuration
%q1 = atan2(c1,b1) + acos(a1/sqrt(b1^2+c1^2)) ;     % Upper
Configuration
q1 = atan2(c1,b1) - acos(a1/sqrt(b1^2+c1^2)) ;      % Lower
Configuration

% Motor2
a2 = x^2 + y^2 + z^2 + x*l - x*L - y*l/sqrt(3) + y*L/sqrt(3) + l^2/3
- 2*l*L/3 + L^2/3 + l1^2 - l2^2 ;
b2 = sqrt(3)*x*l1 + 2*l*l1/sqrt(3) - 2*L*l1/sqrt(3) - y*l1 ;
c2 = 2*z*l1 ;
% Select Configuration
%q2 = atan2(c2,b2) + acos(a2/sqrt(b2^2+c2^2)) ;     % Upper
Configuration
q2 = atan2(c2,b2) - acos(a2/sqrt(b2^2+c2^2)) ;      % Lower
Configuration

% Motor3
a3 = x^2 + y^2 +z^2 - x*l + x*L - y*l/sqrt(3) + y*L/sqrt(3) + l^2/3
- 2*l*L/3 + L^2/3 + l1^2 - l2^2 ;
b3 = - sqrt(3)*x*l1 - y*l1 + 2*l*l1/sqrt(3) - 2*L*l1/sqrt(3) ;
c3 = 2*z*l1 ;
% Select Configuration
%q3 = atan2(c3,b3) + acos(a3/sqrt(b3^2+c3^2)) ;     % Upper
Configuration
q3 = atan2(c3,b3) - acos(a3/sqrt(b3^2+c3^2)) ;      % Lower
Configuration

% Simulation section
```

```
%plot3 ([0,L/2,-L/2,0] , [L/sqrt(3),-L/(2*sqrt(3)),-
L/(2*sqrt(3)),L/sqrt(3)] , [0,0,0,0] ,'r*-') ;
% Base
%plot3 ([0,0,x] , [L/sqrt(3),l1*cos(q1)+L/sqrt(3),y+l/sqrt(3)] ,
[0,l1*sin(q1),z] ,'g') ;                                          %
Motor1 to Platform
%plot3 ([L/2,sqrt(3)*l1*cos(q2)/2+L/2,x+l/2] , [-L/(2*sqrt(3)),-
l1*cos(q2)/2-L/(2*sqrt(3)),y-l/(2*sqrt(3))] , [0,l1*sin(q2),z] ,'g')
;          % Motor2 to Platform
%plot3 ([-L/2,-sqrt(3)*l1*cos(q3)/2-L/2,x-l/2] , [-L/(2*sqrt(3)),-
l1*cos(q3)/2-L/(2*sqrt(3)),y-l/(2*sqrt(3))] , [0,l1*sin(q3),z] ,'g')
;        % Motor3 to Platform
%plot3 ([x,x+l/2,x-l/2,x] , [y+l/sqrt(3),y-l/(2*sqrt(3)),y-
l/(2*sqrt(3)),y+l/sqrt(3)] , [z,z,z,z] ,'b*-') ;
% Platform

% Convert q into degree
q1 = q1*180/pi ;
q2 = q2*180/pi ;
q3 = q3*180/pi ;

q1 = ceil(q1);
q2 = ceil(q2);
q3 = ceil(q3);
```

ÖZGEÇMİŞ

Adı Soyadı : Yiğit Karabulut

Doğum Yeri ve Yılı : İzmir, 1988

Medeni Hali : Bekar

Yabancı Dili : İngilizce

E-posta : yigit.karabulut@cbu.edu.tr

Eğitim Durumu

Lise : İzmir Atatürk Anadolu Lisesi, 2006

Lisans : Ege Üniversitesi, Elektrik Elektronik Mühendisliği Bölümü, 2011

Mesleki Deneyim

Kentkart Ege Elektronik Ltd. Şti., Yazılım ve Uygulama Mühendisi
2011-2012

İzmir Üniversitesi, Elektronik ve Haberleşme Mühendisliği Bölümü, Araştırma Gör.
2013-2015

Celal Bayar Üniversitesi, Mühendislik Fakültesi, Araştırma Görevlisi
2015-(halen)