

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(YÜKSEK LİSANS TEZİ)

**3 SERBESTLİK DERECELİ (3R) BİR ROBOT
MANİPÜLATÖRÜNÜN KONTROLÜ VE GÖRÜNTÜ
İŞLEMEYE DAYALI NESNE TAŞINMASI**

Ekrem YAVUZ

Tez Danışmanı : Doç. Dr. Musa ALCI

Mekatronik Anabilim Dalı

Sunuş Tarihi : 17.06.2015

Bornova - İZMİR

2015

Ekrem YAVUZ tarafından Yüksek Lisans Tezi olarak sunulan “3 Serbestlik Dereceli (3R) Bir Robot Manipulatorin Kontrolü ve Görüntü İşlemeye Dayalı Nesne Taşınması” başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi’nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 17.06.2015 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:

İmza

Jüri Başkanı	: Doç. Dr. Musa ALCI	
Raportör Üye	: Yrd. Doç. Dr. Mustafa ENGİN	
Üye	: Yrd. Doç. Dr. Yavuz ŞENOL	

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Yüksek Lisans Tezi olarak sunduğum “3 Serbestlik Dereceli (3R) Bir Robot Manipülatörün Kontrolü ve Görüntü İşlemeye Dayalı Nesne Taşınması” başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

17/06/ 2015

İmza

Ekrem YAVUZ

ÖZET

3 SERBESTLİK DERECELİ (3R) BİR ROBOT MANİPÜLATÖRÜNÜN KONTROLÜ VE GÖRÜNTÜ İŞLEMeye DAYALI NESNE TAŞINMASI

YAVUZ, Ekrem

Yüksek Lisans Tezi, Mekatronik Anabilim Dalı

Tez Danışmanı: Doç. Dr. Musa ALCI

Haziran 2015, 154 sayfa

Bu tezde, TQ MA2000 robot manipülatörünün çalışma uzayı içerisinde tanımlanmış nesneleri görüntü işleme yöntemi ile algılaması ve nesneleri bulunduğu konumdan alıp başka bir konuma taşınması işlemi gerçekleştirilmiştir. Kameradan alınan görüntüden görüntü işleme ile nesnelerin konumunun tespit edilmesi, robotun tespit edilen konumdaki nesneleri alabilmesi için eklemlerinin alması gereken açı değerlerinin ters kinematik analizi ile hesaplanması ve robotun tespit edilen konuma kontrollü hareketi işlemleri sırasıyla gerçekleştirilmiştir.

Kamera, robotun tüm çalışma uzayını tepeden görecektir şekilde yerleştirilmiştir. Görüntü işleme tekniklerinin uygulanmasında OpenCV kütüphaneleri kullanılmış ve yazılım Code::Blocks geliştirme ortamında, C++ dilinde yazılmıştır. Robotun bilek kısmı, eklem kaçıklıklı ve artık yapılı olması nedeniyle kullanılmamış, ana eklemleri olan ilk üç eklem kullanılmıştır. Ters kinematik analizinde, analitik çözüm yöntemi tercih edilmiştir ve yazılımı görüntü işleme yazılımının devamına yazılmıştır. Robotun hareket kontrolü, L298 motor sürücü entegreli kartlar ve Arduino Mega 2560 mikrodenetleyicisi ile yapılmış, yazılımı Arduino IDE ortamında yazılmıştır. Robotun eklemlerinin alacağı açı değerleri Arduino'ya Code::Blocks ortamından seri port üzerinden gelmektedir.

Geliştirilen bu sistem, robotik ve görüntü işlemeye dayalı robot kontrol uygulamalarında eğitim amaçlı kullanılmaya elverişli durumdadır.

Anahtar sözcükler: Mekatronik, robot kontrolü, TQ MA2000, robot kol, pid kontrol, OpenCV, görüntü işleme, ileri kinematik, ters kinematik, Arduino

ABSTRACT**CONTROL OF 3 DOF (3R) ROBOT MANIPULATOR AND
MOVING OBJECTS BASED ON IMAGE PROCESSING**

YAVUZ, Ekrem

MSc in Mechatronics Eng.

Supervisor: Assoc. Prof. Dr. Musa ALCI

June 2015, 154 pages

In this thesis, detecting predefined objects in the workspace of TQ MA2000 robotic manipulator via image processing methods, and moving these objects from determined location to another location by the robot, was applied. Determining the location of objects using digital image processing techniques through the image captured by the camera, calculating the angle values of the robot's joints extracted from inverse kinematics analysis in order to take the objects into the determined location and motion control of the robot were performed respectively.

The camera was positioned on the top sighting the entire workspace of the robot. OpenCV libraries were used for the implementation of image processing techniques and the software was developed in the Code::Block development environment in C++ language. Robot wrist portion was not used due to redundant and the wrist offset, so the first three joints, which are the main joints, were used. Analytical solution technique was preferred in the inverse kinematics analysis and inverse kinematics analysis software is written subsequently. Motion control of robot was performed with L298 motor driver cards and Arduino Mega 2560 microcontroller, and software was developed in the Arduino IDE environment. Calculated angle values of robot's joints were sent to Arduino from the Code::Block environment via serial port.

This improved system is suitable for educational purposes in the robotics and robot control applications based on image processing.

Keywords: Mechatronic, robot control, TQ MA2000, robot arm, pid control, OpenCV, image processing, forward kinematic, inverse kinematic, Arduino

TEŞEKKÜR

Yüksek lisansım süresince gerek yaptığımız çalışmalarda gerekse diğer zamanlarda yardımlarını esirgemeyen ve tez çalışmamı severek tamamlamamı sağlayan danışman hocam Doç. Dr. Musa ALCI 'ya, çalışmalarımın her aşamasında yardımlarını esirgemeyip, sonsuz sabır göstererek, değerli bilgi ve katkılarıyla çalışmama ışık tutan ve yönlendiren hocam Prof. Dr. Erol UYAR 'a, çalışmalarım sırasında karşılaşılan problemlerin çözümüne bilgi ve tecrübeleriyle katkıda bulunan değerli meslektaşım Nail AKÇURA 'ya, değerli arkadaşlarım Doğan Can KARATAŞ 'a, Alican ALPKAYA 'ya ve Mücahid CANDAN 'a teşekkürü bir borç bilirim.

Ayrıca, çalışmalarım esnasında maddi manevi yardımlarını benden esirgemeyen aileme en içten şükranlarımı sunarım.

İÇİNDEKİLER

Sayfa

ÖZET	vii
ABSTRACT	ix
TEŞEKKÜR	xi
İÇİNDEKİLER DİZİNİ.....	xiii
ŞEKİLLER DİZİNİ	xvii
ÇİZELGELER DİZİNİ.....	xxiii
SİMGELER VE KISALTMALAR DİZİNİ	xxv
1. GİRİŞ	1
2. ÇALIŞMAYI OLUŞTURAN ANA HATLAR.....	4
2.1 Literatür Taraması	4
2.2 Robot	11
2.2.1 Robotların avantaj ve dezavantajları	12
2.2.2 Robotların kullanıldığı endüstriyel alanlar	13
2.2.3 Robotları tanımlamada kullanılan parametreler	14
2.2.4 Robotu oluşturan parçalar.....	15
2.2.5 Robotlarda kullanılan eklem yapıları.....	15
2.2.6 Robotlarda kullanılan bilek yapıları	16
2.2.7 Robotların sınıflandırılması.....	18
2.2.8 Robot programlama yöntemleri.....	29
2.2.9 Robot kinematik analizi.....	30

İÇİNDEKİLER (devam)

	<u>Sayfa</u>
2.3 Görüntü İşleme ve Esasları	49
2.3.1 Görüntü işleme adımları.....	49
2.3.2 Görüntü işlemede kullanılan terimler	50
2.3.3 Görüntü işlemede kullanılan teknikler	52
2.3.4 Renk modelleri.....	56
2.3.5 OpenCV görüntü işleme kütüphanesi	59
2.4 Kontrol Sistemleri.....	60
2.4.1 Kontrol nedir?	60
2.4.2 Kontrol sistemi çeşitleri	60
2.4.3 Kontrol sistemi çeşitleri	62
2.4.4 Kontrol yöntemleri.....	63
2.5 TQ MA2000 Robot Kolu	72
2.6 Arduino	77
2.6.1 Arduino nedir?	77
2.6.2 Arduino Mega 2560 R3 mikrodeneleyici kartı	77
2.7 L298 Motor Sürücü Entegresi ve Kartı.....	84
2.7.1 L298 motor sürücü entegresi.....	84
2.7.2 L298 entegreli motor sürücü kartı.....	87
3. YAPILAN ÇALIŞMALAR	90
3.1 Robotun Bağlantı Şeması.....	90

İÇİNDEKİLER (devam)

Sayfa

3.1.1 Motorların sürücü kartlarına bağlanması.....	90
3.1.2 Sürücü kartlarının Arduino'ya bağlanması	91
3.1.3 Potansiyometrelerin Arduino'ya bağlanması	91
3.2 OpenCV ile Görüntü İşleme	93
3.2.1 Görüntü işleme yazılımının çalıştırılması.....	93
3.2.2 Referans objelerinin yerleştirilmesi.....	93
3.2.3 Referans objelerinin ve taşınacak nesnelerin koordinatlarının tespiti	95
3.2.4 Tespit edilen koordinatlarının ekrana basılması	110
3.3 TQ MA2000 Robot Manipülatörünün İleri ve Ters Kinematik Analizi.....	114
3.3.1 TQ MA2000 robot manipülatörünün ileri kinematik analizi.....	114
3.3.2 TQ MA2000 robot manipülatörünün ters kinematik analizi	119
3.4 MATLAB Guide Ortamında Hazırlanmış Kinematik Analiz Arayüzü.....	128
3.5 Arduino Mikrodenetleyicisi ile Gerçekleştirilen İşlemler	132
3.5.1 Arduino'ya kütüphanelerin yüklenmesi	132
3.5.2 Giriş ve çıkış pinlerinin belirlenmesi.....	132
3.5.3 Potansiyometrelerin okunması ve anlamlandırılması.....	133
3.5.4 Seri haberleşmenin başlatılması	135
3.5.5 PWM frekansının ayarlanması	136
3.5.6 Seri porttan gelen verilerin okunması.....	137
3.5.7 PID kütüphanesinin kullanılması.....	138

İÇİNDEKİLER (devam)

	<u>Sayfa</u>
3.6 Robotta Bulunan Motorların Hareket Eğrileri	140
3.6.1 1. Motorun hareket eğrisi	140
3.6.2 2. Motorun hareket eğrisi	141
3.6.3 3. Motorun hareket eğrisi	142
3.6.4 Tüm eklemlerin aynı anda hareketi.....	143
4. SONUÇLAR VE TARTIŞMA	145
KAYNAKLAR DİZİNİ	147
ÖZGEÇMİŞ	153
EKLER.....	

ŞEKİLLER DİZİNİ

<u>Şekil</u>	<u>Sayfa</u>
2.1 Robot kolu ve insan kolu benzetimi	12
2.2 Prizmatik eklem	15
2.3 Döner eklem.....	16
2.4 Euler bileğinin yapısı	16
2.5 Eklem kaçıklı bileğin yapısı	17
2.6 a) Seri robot örneği, b) Paralel robot örneği	19
2.7 Kartezyen robot ve çalışma uzayı.....	20
2.8 Silindirik robot ve çalışma uzayı	21
2.9 Küresel robot ve çalışma uzayı.....	21
2.10 SCARA robot ve çalışma uzayı.....	22
2.11 Mafsallı robot ve çalışma uzayı.....	23
2.12 Açık döngü kontrol sistemi örneği	26
2.13 Kapalı döngü kontrol sistemi örneği	26
2.14 Bir noktanın bir koordinat sistemine göre tanımlanması.....	31
2.15 Bir noktanın iki farklı koordinat sistemine göre tanımlanması	32
2.16 Bir robotun uç işlevcisinin bir noktaya göre tanımlanması	32
2.17 Bir cismin referans bir koordinat sistemine göre yönelimi	33

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
2.18 Merkezleri çakışık, yönelimleri farklı iki koordinat sistemi.....	34
2.19 Merkezleri farklı, yönelimleri aynı iki koordinat sistemi	35
2.20 Merkezleri ve yönelimleri farklı iki koordinat sistemi	36
2.21 İki boyutlu uzayda dönme.....	37
2.22 X ekseninin θ açısı kadar döndürülmesi	39
2.23 Y ekseninin θ açısı kadar döndürülmesi	40
2.24 Z ekseninin θ açısı kadar döndürülmesi.....	40
2.25 DH değişkenleri olan a_{i-1} , α_{i-1} , d_i ve θ_i 'nin belirlenmesi.....	42
2.26 PUMA-560 robotu için dört farklı çözüm.....	45
2.27 Görüntünün yakalanması ve sayısallaştırılması aşamaları.....	49
2.28 İkili görüntü.....	51
2.29 Gri düzey görüntü	52
2.30 a) Gri düzey bir görüntü b) Bu görüntüye ait histogram.....	53
2.31 Açısal bozulmalar a) bozulma yok b) dış bükey bozulma c) iç bükey bozulma	53
2.32 Sobel filtresi örneği a) Filtrelemeden önce b) Filtrelemeden sonra.....	54
2.33 Eşikleme işlemi a) Eşiklemeden önce b) Eşiklemeden sonra	55

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
2.34 Etiketleme işlemi	55
2.35 RGB renk modeli	56
2.36 CMYK renk modeli	57
2.37 Renk özü (Hue) değerinin silindirik gösterimi	58
2.38 HSV renk modelinin konik biçimde gösterimi	59
2.39 Kontrol sistemlerinin temel gösterimi	60
2.40 Negatif geri beslemeli bir sistem	61
2.41 Pozitif geri beslemeli bir sistem	61
2.42 Açık çevrimli kontrol sistemi	62
2.43 Kapalı çevrimli kontrol sistemi	63
2.44 P (Oransal) kontrolörlü sistemin yapısı	64
2.45 P kontrolün, kontrolsüz bir sistem ile karşılaştırılması	65
2.46 PI (Oransal + Integral) kontrolörlü sistemin yapısı	66
2.47 PI kontrolün, kontrolsüz bir sistem ile karşılaştırılması	66
2.48 PD (Oransal + Türevsel) kontrolörlü sistemin yapısı	67
2.49 PD kontrolün, kontrolsüz bir sistem ile karşılaştırılması	68
2.50 PID (Oransal + Integral + Türevsel) kontrolörlü sistemin yapısı	69

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
2.51 PID kontrolün, kontrolsüz bir sistem ile karşılaştırılması.....	69
2.52 Ziegler - Nichols yöntemine göre K_c ve T_c nin bulunması	70
2.53 TQ MA2000 robotunun SolidWorks programında yapılan çizimi	72
2.54 TQ MA2000 robot kolunda kullanılan DC motorların yapısı.....	73
2.55 Motor kablolarını içeren 18 pinli konnektör	74
2.56 Potansiyometre kablolarını içeren 12 pinli konnektör	74
2.57 MA2000 robot kolunun aç ve potansiyometre ilişkisi.....	75
2.58 MA2000 robot kolunun çalışma uzayı benzetimi	76
2.59 Arduino Mega 2560 R3 kartının ön yüzü	78
2.60 Arduino Mega 2560 R3 kartının arka yüzü	78
2.61 Arduino IDE.....	83
2.62 L298 entegresi bacak yapısı	84
2.63 H köprüsünün yapısı	86
2.64 H köprüsü kullanarak motorların yön kontrolü.....	86
2.65 H köprüsünde frenleme yöntemi.....	87
2.66 L298 entegreli motor sürücü kartı.....	88
3.1 Görüntü işleme yazılımının çalıştırılması	93

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
3.2 Referans objelerinin yerleştirilmesi - 1.....	94
3.3 Referans objelerinin yerleştirilmesi - 2.....	94
3.4 Kameradan alınan orijinal görüntü	96
3.5 HSV renk uzayına çevrilmiş görüntü	97
3.6 Referans objelerinin eşiklenmesi.....	99
3.7 Taşınacak nesnelerin eşiklenmesi.....	100
3.8 Referans objelerinin filtrelenmesi.....	102
3.9 Taşınacak nesnelerin filtrelenmesi	103
3.10 Referans objelerinin kenarlarının tespiti.....	104
3.11 Taşınacak nesnelerin kenarlarının tespiti.....	105
3.12 Referans objelerinin orta noktalarının tespit edilmesi	108
3.13 Taşınacak nesnelerin orta noktalarının tespit edilmesi.....	109
3.14 Tespit edilen koordinatların ekrana basılması	110
3.15 Nesnelerin doğru algılandığı kontrolü	112
3.16 Ters kinematik analizi sonucu çıkan açı değerleri.....	113
3.17 TQ MA2000 robot manipülatörünün majör eklemlerine koordinat sistemlerinin yerleştirilmesi	115

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
3.18 TQ MA2000 robot manipülatörünün bileğine koordinat sistemlerinin yerleştirilmesi	115
3.19 MATLAB - Guide ortamında tasarlanan arayüz.....	129
3.20 MATLAB - Guide ortamında tasarlanan arayüzde ters kinematik analizi ..	130
3.21 MATLAB - Guide ortamında tasarlanan arayüzde ileri kinematik analizi..	131
3.22 1. Eklemin 0 - 90 derece hareketi	140
3.23 2. Eklemin 0 - 90 derece hareketi	141
3.24 3. Eklemin 0 - 90 derece hareketi	142
3.25 Tüm eklemlerin aynı anda 0 - 90 derece hareketi	143

ÇİZELGELER DİZİNİ

<u>Çizelge</u>	<u>Sayfa</u>
2.1 Euler bileğinin DH tablosu	17
2.2 Eklem kaçıklı bileğin DH tablosu.....	18
2.3 Robot çeşitlerinin karşılaştırılması	23
2.4 Elektrik, hidrolik ve pnömatik tahrik sistemlerinin karşılaştırılması	28
2.5 DH tablosu	43
2.6 CMYK renk modelinde diğer renklerin elde edilmesi	57
2.7 P, I ve D parametrelerinin sistem üzerindeki etkileri	63
2.8 P, I ve D katsayılarının hesaplanması.....	71
2.9 Arduino Mega 2560 R3 kartının genel özellikleri.....	79
2.10 H köprüsü doğruluk tablosu	87
3.1 18 pinli motor konnektörünün L298 sürücü kartlarına bağlanması.....	90
3.2 L298 - Arduino bağlantıları	91
3.3 Potansiyometre - Arduino bağlantısı	92
3.4 Referans objeler için tespit edilen H,S,V değerleri	98
3.5 Taşınacak nesneler için tespit edilen H,S,V değerleri	100
3.6 TQ MA2000 robot manipülatörü için oluşturulan DH tablosu	116
3.7 Birinci eklem için Aç - Potansiyometre ilişkisi.....	134

ÇİZELGELER DİZİNİ (devam)

<u>Çizelge</u>	<u>Sayfa</u>
3.8 Arduino Mega 2560 için Timer'lar ve kontrol ettiği pinler.....	136
3.9 PWM frekans seçimi.....	136
3.10 1. motor için parametreler.....	140
3.11 2. motor için parametreler.....	141
3.12 3. motor için parametreler.....	143

SİMGELER VE KISALTMALAR DİZİNİ

<u>Kısaltmalar</u>	<u>Açıklama</u>
AC	Alternatif Akım
CAD	Bilgisayar Destekli Tasarım
CAM	Bilgisayar Destekli Üretim
CCD	Yüklenme İliştirilmiş Araç
DC	Doğru Akım
DFD	Ayrıştırılmış Çerçeve Farkı
DH	Denavit Hartenberg
EEPROM	Elektronik Olarak Silinip Programlanabilir Salt Okunur Bellek
ICSP	Devre Üzerinden Programlama
IDE	Tümleşik Geliştirme Ortamı
ISA	Sanayi Standardı Mimari
I2C	Bir Çeşit Seri Haberleşme Protokolü
LED	Işık Yayan Diyot
MISO	Köleden Efendiye Veriyolu
MOSI	Efendiden Köleye Veriyolu
PID	Oransal - İntegral - Türevsel
PLC	Programlanabilir Lojik Kontrolör

SİMGELER VE KISALTMALAR DİZİNİ (devam)

<u>Kısaltmalar</u>	<u>Açıklama</u>
PSO	Parçacık Sürü Optimizasyonu
PUMA	Programlanabilir Üniversal Manipulatör Kol
PWM	Darbe Genişlik Modülasyonu
RBF	Radyal Tabanlı Fonksiyon
RX	Seri İletişimde Alıcı
SCK	Seri Saat
SCL	Saat Darbe Yolu
SDA	Veriyolu
SPI	Seri Çevresel Arayüz
SRAM	Durağan Rastgele Erişimli Bellek
SS	Köle Seçimi
TWI	İki Kablolu Arayüz
TX	Seri İletişimde Verici
UART	Üniversal Asenkron Alıcı Verici
USB	Üniversal Seri Veriyolu

1. GİRİŞ

İnsan yaşamını kolaylaştırmak, insanların yavaş veya güçsüz kaldığı işleri gerçekleştirmek amacıyla, insan müdahalesine ihtiyaç duymadan kendi kendine hareket edebilen makineler geliştirmek yıllardır süregelen birçok çalışmanın ana konusu olmuştur. Robotlara, insan duyusuna benzer özellikler (görme,dokunma,duyma vb.) yükleyebilmek için çeşitli sensörler de ilave edilerek robotların akıllı hale getirilmesi ve işlevselliğinin artırılması sağlanmıştır.

Robotların günümüzdeki en yaygın kullanım yeri endüstriyel üretim yapan alanlardır. Özellikle otomotiv endüstrisinde birbirinden farklı işlevleri yerine getirebilen robotlar kullanılmaktadır. Bu işlevlere örnek olarak yer ve yön değiştirme, taşıma, kaynak, parça montajı ve boyama örnek verilebilir. Robotların diğer kullanım alanları askeri uygulamalar , haberleşme ve tıp sektörüdür.

Robotların görmeye dayalı kontrolü, görüntü işleme teknikleri ile mümkündür. Görüntü işleme; dijital olarak kameradan alınan bir görüntünün istenen yapıya getirilmesi için özellikleri üzerinde değişiklikler yapılmasını sağlayan tekniktir. Bu teknik aracılığıyla görüntü üzerinde analiz yapmak mümkün olmaktadır. Görüntü işleme tekniği ile bir cismin boyutu, orta noktası, ağırlık merkezi, rengi, şekli, konumu vb. özellikleri rahatlıkla bulunabilir. Görüntüde özellikle istenen bölüm varsa bu ön plana çıkarılarak diğer kısımlar yok edilebilir, görüntüde bulunan bozukluklar çeşitli yöntemlerle düzeltilebilir.

Görüntü işlemenin günümüzde başlıca kullanım alanları savunma sanayisi, güvenlik sistemleri, kalite kontrol sistemleridir. Güvenlik sistemlerine örnek olarak trafik denetimi sistemleri (araç takibi, plaka tanıma, hız takibi) gösterilebilir.

Bu tez çalışmasında, 3 eklemlili bir robot kolunun çalışma uzayında (zeminde) bulunan nesnenin (topun) yerini görüntü işleme teknikleri ile tespit ederek ve ters kinematik analizi ile nesneyi yerinden almak için gerekli uzuv açıları hesaplanarak robotun nesneyi bulunduğu konumdan alıp, önceden öğretilmiş bir konuma bırakması amaçlanmıştır. Robotun çalışma uzayını tepe noktadan gören bir kameradan alınan görüntü, OpenCV kütüphaneleri kullanılarak C++ dilinde yazılan yazılım ile nesnenin orta noktasının tespitinde kullanılmıştır. Görüntü işleme tekniği ile bulunan nesnenin X-Y konumuna robotun ulaşabilmesi için gerekli uzuv açılarının hesaplatıldığı ters kinematik

problemin çözümü yine C++ dilinde yazılmıştır. Uzun açılarının hesaplanmasından sonra, bulunan açı değerleri seri port üzerinden robotun kontrollü hareketi için Arduino Mega 2560 mikrodeneleyici kartına aktarılmıştır. Arduino mikrodeneleyicisinin içine yazılmış olan kontrol yazılımı robotun belirtilen açılara en optimal şekilde ulaşmasını ve nesneyi tutup istenen konuma taşınmasını sağlamıştır.

Çalışılan tez konusunun içinde barındırdığı üç esas konu, parçalara ayrılarak Bölüm 2'de aktarılmıştır. Bölüm 2.1'de görüntü işleme ve robot kontrolü konusunda yapılmış olan çalışmalar hakkında yapılan literatür taraması verilmiştir. Bölüm 2.2'de robotun tanımı ve ortaya çıkışı, robotu oluşturan parçalar, robotların eklem yapıları, robotların sınıflandırılması, robotların avantaj ve dezavantajları, robot ileri ve ters kinematiğinin hesaplanması hakkında bilgi verilmiştir. Bölüm 2.3'de görüntü işlemenin tanımı, kameradan alınan görüntünün üzerinde uygulanan görüntü işleme teknikleri ve OpenCV görüntü işleme kütüphanesi üzerinde durulmuştur. Bölüm 2.4'de kontrol nedir, kontrol çeşitleri nelerdir, çalışmada kullanılan kontrol yönteminin esasları nelerdir sorularına cevaplar verilmiştir. Bölüm 2.5'de çalışmada kullanılan TQ MA2000 robot manipülatörünün özellikleri üzerinde durulmuştur. Bölüm 2.6'da robotun kontrol edilmesi için kullanılan Arduino Mega 2560 mikrodeneleyicisi, Bölüm 2.7'de ise robotta bulunan motorların sürülmesi için kullanılan L298 motor sürücü entegreli kartlar anlatılmıştır.

Bölüm 3'de tez çalışması sırasında yapılanlar 6 alt başlıkta anlatılmıştır. Bölüm 3.1'de robotun motorlarının, geri besleme elemanı olan potansiyometrelerinin, Arduino mikrodeneleyicisinin ve motor sürücü kartlarının bağlantıları verilmiştir. Bölüm 3.2'de nesnenin orta noktasının bulunabilmesi için kameradan alınan görüntünün hangi görüntü işleme adımlarından geçtiği üzerinde durulmuştur. Bölüm 3.3'de TQ MA2000 robot kolunun Denavit - Hartenberg yöntemi kullanılarak her bir eklem için yazılmış olan DH parametreleri kullanılarak eklemlerin aldıkları açılara göre uç işlevcinin bulunduğu konumu ve oryantasyonunu bilmemizi sağlayan ileri kinematik analizi ve analitik yöntemler kullanılarak uç işlevcinin istenen konuma gidebilmesi için eklemlerin alacağı açılarının hesaplanmasını sağlayan ters kinematik analizi yapılmış, orta noktası bulunan nesnenin bulunduğu yerden alınabilmesi için robot kolunun uzuvlarının alması gereken açı değerlerinin hesaplanması anlatılmıştır. Bölüm 3.4'te, ana sistemde kullanılmayan ancak çalışmanın yapılmasında yardımcı olması amacıyla Matlab Guide arayüzünde yazılmış olan ileri ve ters kinematik analizi programı

anlatılmıştır. Bölüm 3.5'de robotun nesneyi alabilmek amacıyla yaptığı hareketin Arduino ile nasıl kontrol edildiği anlatılmıştır. Bölüm 3.6'da robotta bulunan motorların verilen hedeflere ulaşması sırasında ortaya çıkan hareket eğrileri verilmiştir.

2. ÇALIŞMAYI OLUŞTURAN ANA HATLAR

2.1 Literatür Taraması

Literatürde robotik, görüntü işleme ve robot kinematiğinin analitik yöntemler ile çözümüne yönelik birçok çalışma bulunmaktadır. Bu tez çalışmasında yapılan görmeye dayalı robot kontrolü konusundaki literatürden bazıları, aşağıda verilmiştir:

M.A.P. Cisneros (2004), TQ MA2000 6R robot kolunun ucuna CCD kamera monte etmiştir. Kameradan aldığı görüntüleri gerçek zamanlı olarak işleyerek robotun çalışma uzayı içerisinde bulunan ve sürekli hareket eden oyuncak bir treni takip etmesini sağlamıştır. Çalışmasında robot kontrolü için neural network kullanmıştır.

F.A.T. Al-Saedi and A.H. Mohammed (2012), TQ MA2000 6R robot kolunun her bir eklemine PSO algoritmasına dayalı PID kontrolör tasarlamışlardır ve robotun optimal PID parametrelerini tespit etmişlerdir.

G.S. Huang et al (2011), yaptıkları çalışmada bir düzlemde bulunan nesnelerin görüntülerini kamera ile yakalayıp görüntüde bulunan nesneleri şekil olarak ayırtmışlar ve kenar belirleme algoritmaları ile yuvarlak olan nesneleri tespit etmişlerdir. Tespit edilen bu yuvarlak nesneleri robot kol kullanarak başka bir yere taşımışlardır.

C. Doignon et al (1994), 6 eklemlili robot kolun çalışma uzayında bulunan hareket halindeki cismin takip edilmesi üzerine çalışmışlardır. Çalışmada robotun uç işlevcisine monte edilmiş CCD kamera kullanmışlar ve bu kameradan alınan görüntüleri işleyerek gerçek zamanlı takip yapmışlardır.

L. T. Anh and J. B. Song (2010), yaptıkları çalışmada nesneleri yakalayabilen bir sistem geliştirmişlerdir. Robot kolun üzerine bağlı kamera ile yerde hareket halinde olan nesneleri takip ettirmişlerdir. İlk olarak görüntüyü almışlar ve alınan görüntüleri önceki görüntüler ile karşılaştırmışlardır. Hareket algılandığı zaman hareket halinde olan nesnenin konumu tespit etmişlerdir. Robot kolu hareket ettirmek için servo motor sürücü devresine gerekli komutları göndererek robota nesneyi yakalatmışlardır.

H. C. Chang ve T.M. Shih (2008), yaptıkları çalışmada üç eksenli bir robot kola bağlı silah ile hedef takibi yapan ve hedefi vuran bir sistem tasarlamışlardır. Robot kolun sürücü devresi için NI PXI-7356 hareket kontrol kartını kullanmışlardır. Hedef tespit edildikten sonra namlunun hedefe en kısa zamanda ulaşabilmesi için hesaplamalar yapmışlar ve bulanık mantık algoritmaları kullanmışlardır. Robot kolu servo motorlar ile kontrol edilmiştir.

M. Ogawa et al (2011), yaptıkları çalışmada gerçek zamanlı olarak bir insan ile hava hokeyi oynayabilen robot kol gerçekleştirmişlerdir. Masayı üstten gören bir kamera ile bilgisayara alınan görüntüler işlenmekte ve hokeyin yeri tespit edilmektedir. Daha sonra robot kol hokeyi karşılayacak şekilde konumlandırılmaktadır ve karşılık vermektedir. İki eksenli olan robot kolu hareket ettirmek için kontrol birimi tarafından iki adet açı ve hız bilgisi robot kola gönderilmektedir.

F. Basile et al (2012), yaptıkları çalışmada, birden fazla robot kolunun bulunduğu çalışma alanında bulunan nesnelerin yerlerini tespit etmişlerdir. Hedef nesnelerin koordinatlarını bulduktan sonra hangi robot hangi nesneye yakın bunu tespit etmişler ve nesnelerin o koordinatlardan alınabilmesi için ters kinematik hesabı yapmışlardır.

K. Mundhra et al (2002), dikey düzlemde serbest düşmekte olan bir topu yakalaması için insana dayalı algoritma ile geliştirilen bir mobil robot tasarlamışlardır. Dijital görüntü işlemeyi çok hızlı bir şekilde işletebilmek için kendileri DVRobot adında bir dijital görüntü işleme programı yazmışlardır.

K.P. Mondy et al (2005), ping pong oynayan robot kolu tasarımı ve geliştirmesi üzerine çalışma yapmışlardır. Topun hareketini arka plandaki değişimlerden arındırabilmek için DFD (Displaced frame difference) yöntemini kullanmışlardır. CCD video kameradan alınan gerçek zamanlı görüntüler üzerinde uygulanan bir dizi görüntü işleme tekniği ile ping pong topunun konumunu tespit etmişler ve robot kolun topa belirli bir açı ile vurarak masanın karşı tarafına gönderebilmesini sağlamışlardır.

S. van Delden et al (2007), robot koluna bağlı kamera ile robotun çalışma uzayındaki topun yönelimine bağlı olarak kameranın yöneliminin otomatik olarak gerçekleştirilmesini sağlamışlardır.

J.A. Akec et al (1998), robot kolun uç işlevcisine taktıkları kameradan sürekli olarak alınan görüntüyü işleyerek konveyör bant üzerindeki kıvrımlı izin robot kol tarafından takibini sağlamışlardır. 6 eklemlili bir robot kol kullanarak çalışmayı uygulamaya geçirmişlerdir.

L. Bustamante and J. Gu (2007), bulunduğu çevreyi tarayıp elektrik prizini algılayan ve elektrik prizinin yanına gidip fişini prize takarak kendi güç kaynağını şarj edebilen bir mobil robotun dizayn ve uygulamasını yapmışlardır. Sistemde bir adet robot kol, bir adet zoom kamera, bir adet standart web kamera ve bir adet lazer yakınlık sensörü kullanmışlardır. Görüntü işleme çalışmasında şekil fonksiyonlarını kullanmışlardır.

V.F. Mufioz et al (2004), laparoskopik ameliyatlarda kullanılan robotik kameraların hareketini, kartezyen hareket kontrolörü kullanarak sağlamışlardır. Robot kolun hareketinin yönelim planlamasını adaptif kontrolle gerçekleştirmişlerdir.

W. Choi et al (1999), mobil bir robotun kapalı bir alanda üzerindeki sensörler vasıtasıyla engellere çarpmadan hareket etmesini sağlamışlar ve kameradan aldıkları görüntüyü işleyerek kapalı alanın haritasını çıkarmışlardır. Çalışmada düşük maliyetli multi-sensör sistemi ve robotun otonom hareketini sağlamak için çeşitli algoritmalar kullanmışlardır.

J. Carusone and G.M.T. D'Eleuterio (1998), ortam ışığı, çalışma alanı ve hedef noktaları gibi dış etkenlerden etkilenmeyen neural - network tabanlı bir görme sistemi geliştirmiş ve buna bağlı olarak robot kolun hareketini sağlamışlardır. Robotun çalışma uzayında bulunan topun yerini, CCD kamera tarafından alınan görüntüyü işleyerek tespit etmişler ve topu yakalamışlardır.

M. Albayrak (1997), bilgisayar destekli kontrol yöntemi ile üç boyutlu uzayda hareket edebilen, üç serbestlik dereceli robot kol tasarımı, kontrolü ve simülasyonu nu yapmıştır. Robotun kontrolü için, bilgisayarın 16 bitlik ISA slotlarından birine yerleştirilebilecek yapıda; adres çözücü katı, programlanabilir giriş/çıkış katı, dört adet motor sürücü katı ve step motor frenleme katının bulunduğu devreleri tek bir kontrol kartı üzerinde toplamıştır.

T. S. Tonbul (2002), beş eklemlili robot kolunun ters kinematik analizini ve yörünge planlamasını yapmıştır. Robotun eklem açılarının, açısal hız ve

ivmelerinin zamana göre deęişimlerini Matlab programını kullanarak hesaplatmış ve EDUBOT üzerinde uygulamasını da gerçekleştirmiştir.

M. Kert (2006), kameradan aldığı görüntüleri işleyerek, dairesel kesitli nesneler ve bu nesneler arasındaki engellerin konumlarını bulmuştur. Görüntü işlemenin sonucunda elde edilen nesnelerin üç boyutlu koordinatlarını, geliştirdiği algoritmaya aktararak, robotun nesnelere hangi sıra ile erişeceğinin optimizasyonu yapmıştır.

Ç. Saygılı (2006), Scara tipi bir robotun tasarımını ve animasyonunu gerçekleştirmiştir. Robotun parçalarını Solidworks programında modellemiş ve parçaları monte ederek robotun katı modelini oluşturmuştur. Matlab programında robotun ileri ve ters kinematik analizlerini ve hesaplamalarını yapmıştır.

S. Çiçek (2006), malzemelerin rengine göre hassas bir şekilde taşıma yapan robot kolu tasarımı ve uygulamasını yapmıştır. Çalışmada önce 5 serbestlik dereceli, R/C servo motor kontrollü, alüminyum mekanik sistemli, mikrodenetleyici tabanlı robot kolu ile mikrodenetleyici tabanlı renk sensörü tasarlayarak, örnek bir platform inşa etmiş ve robot kolunun bu platform üzerinde çalışmasını incelemiştir.

S. B. Kuran (2007), gıda sektöründe paketleme amaçlı, yapay görme kılavuzlu robotik uygulama geliştirmiştir ve bu sistemin uygulanabilirliğini test etmiştir. Bu amaçlar için endüstriyel bir robot ile haberleşen ve PC de çalışan yapay görme yazılımı geliştirmiş, robot ile birlikte çalıştırılmasını gerçekleştirmiştir. Kurmuş olduğu sistem yapay görme yazılımı, kamera ve altı eksenli endüstriyel robottan oluşmaktadır.

M.A. Duran (2007), üç serbestlik dereceli PUMA tipi robot kolunun uç elemanının PID yöntemiyle kontrolünü sağlamıştır. Robota ait kinematik ve dinamik denklemleri elde etmiştir. Dinamik analiz sonucu elde edilen servo motor torklarını sisteme dış bozucu etki olarak dahil etmiştir. Yörünge planlamasını kartezyen koordinatlarında yapmıştır.

A. Bayrak (2007), beş eksenli bir robot kolunun ters kinematik hesaplamalarını Paden-Kahan alt problemleri olarak bilinen alt problemlere indirgeyerek çözmüş ve yörünge planlamasını yapmış, görüntü işleme teknikleri kullanarak robotun hareketi esnasında önüne çıkabilecek engeli tanımlayacak bir

sistem geliřtirmiş ve robot kolunun hareketini bilgisayar ortamında simüle etmiştir. Robotun ters kinematik hesaplamalarını, yörünge planlamasını, piksel-mesafe hesaplamasını ve üç boyutlu simülasyonunu Delphi 7 programlama dilini kullanarak gerçekleřtirmiştir. Görüntü işleme ve nesne tanımlama işlemlerini Matlab kullanarak gerçekleřtirmiştir.

Z. Bingöl vd. (2005), yaptıkları çalışmada 6R serbestlik dereceli bir robotun ters kinematik analizini yapay sinir ağıları ile çözmüşlerdir. Bu çalışmalarında analitik olarak çözümü olmayan eklem kaçıklıklı bir robot kolu için yapay sinir ağı ile ters kinematik çözümünü gerçekleřtirmişlerdir ve elde ettikleri sonuçları farklı çözüm yöntemlerinden elde edilen sonuçlar ile karşılařtırmışlardır.

B.H. Dinh et al (2008), bir robot kolunun ters kinematik problemini radyal tabanlı bir ağı (RBF) kullanarak çözmeye çalışmış ve görüntü işlemeye dayalı bir kontrol uygulaması yapmışlardır.

Y. Şahin (2006), çalışmasında SCARA tipi bir robotun uç noktasının PID kontrol yöntemi ile verilen yörüngeyi uygun bir şekilde izlemesini amaçlamıştır.

H. Bulut (2002), çalışmasında düzlemsel tip robot kolunun ileri kinematik, ters kinematik, dinamik ve klasik PD tipi denetim, klasik PID tipi denetim ve PD artı yerçekimi terimli denetimi için benzetim yazılımı geliřtirmiştir. Her bir denetim stratejisi için elde edilen denklemlerin benzetim yazılımı Matlab ortamında programlamış ve görsel canlandırmaları Delphi yazılımı ile gerçekleřtirmiştir.

M.T. Daş (2003), yaptığı çalışmada SCARA tipi dört serbestlik dereceli Serpent 1 tipi robotun programlanabilir mantık denetleyicisiyle (PLC) hareket denetimini sağlamıştır. Çalışmada Siemens S7-200 serisi PLC ve Serpent 1 (Practical Electronics) robot kullanılmaktadır. Mevcut sistem doğru akım servo motorları ve pnömatik sürücülerle hareket etmektedir. Öncelikle robotun orijinal konumlandırması çözümlenmiş daha sonra PLC kısmen SCARA robota uyarlanmıştır. Uygulama olarak montaj hattında olduğu düşünölen üç farklı boyut ve ağırlıktaki parçaların bir noktadan hedef olarak belirlenen başka bir noktaya taşınarak hareket denetiminin sağlanması üzerinde çalışmıştır.

D. Sun and J.K. Mills (1999), yörünge performansının geliştirilmesi ve tekrarlı işlemlerde yakınsamanın garanti altına alınması için adaptif öğrenme

kontrol tekniğini önermiştir. Fakat bu kontrol yöntemi sistemin detaylı bir şekilde dinamiğinin incelenmesini gerektirmektedir.

C. Semetay (2001), tezinde robotların matematiksel modelleri elde ederek PID ve kayan kipli kontrol yöntemlerinin robotlar üzerine uygulanması ile sistemin verdiği cevapları incelemiştir.

J.Jantzen (1998), yaptığı çalışmada PID kontrolörün dizaynında kontrol parametrelerinin (K_p , K_i , K_d) belirlenmesi üzerinde durmuş ve lineer olmayan sistemlerin PID kontrol parametrelerinin bulanık mantık kullanılarak daha kolay elde edilmesi için çalışmıştır.

Z. Wei and L. Cai (2010), Gömülü Linux sistemi kullanarak bir ortamın sürekli görüntüsünü alarak hareket tespiti yapmışlardır. Çalışmalarında S3C2440 ARM işlemcili bir elektronik kart kullanmışlardır. Ortamın sürekli olarak görüntülerini almış ve her görüntüyü bir önceki ile karşılaştırarak oluşan farkları incelemiş ve hareketleri tespit etmişlerdir.

M. Seelye et al (2010), çalışmalarında uzaktan kumandalı robot kol kullanarak, robot kola bağlı olan kamera ile bitkilerin büyümelerini incelemişlerdir. Bitkilerin kas gelişimlerini ayrıntılı olarak incelemişlerdir. Bitki büyüdükçe kamera açısını robot kol ile ayarlayarak istenilen bölgelerin gelişimini takip etmişlerdir. Uzaktan kumanda ederken veri haberleşmesini ZigBee haberleşme teknolojisini kullanarak gerçekleştirmişlerdir.

H. H. Rapp (2011), yaptığı çalışmada robot kola bağlı bir raket üzerinde pin pon topunu takip eden ve sektiren bir sistem tasarlamıştır. Bu sistemde iki adet kamera bulunmaktadır, kameralar Linux tabanlı çalışan bir bilgisayara bağlıdır. Bilgisayar bir adet endüstriyel bilgisayara bağlıdır. Robot kol endüstriyel bilgisayar tarafından kontrol edilmektedir. Kameralardan biri raketi üstten, diğeri yandan görecektir şekilde yerleştirilmiştir. Böylece topun yukarı doğru yükselmesi ile raketin vurma hızını, sağa sola kayması ile de raketin konumunu ayarlamıştır. Bu sistem yaklaşık olarak 30 dakika boyunca topu sektirmeye devam edebilmektedir.

A. Bayrak ve M. Sarıtaş (2008), yaptıkları çalışmada beş eksenli bir robot kolu ile hedef tespit edilmesi işlemini gerçekleştirmişlerdir. Bu işlemi yaparken robot kolun karşılaşılabileceği engellere çarpmasını önlemişlerdir. Önceden

belirlenen hedefler bilgisayara kayıt edilmiş ve ortamdan alınan görüntüde hedef tarama işlemini gerçekleştirmişlerdir. Hedef tespit edildikten sonra koordinat tespiti algoritmaları ile görüntü üzerinde hedefin koordinatları belirlenmiş ve robot kol ilgili koordinata konumlandırılarak hedef alınmıştır.

A. Bejo et al (2009), bilgisayar destekli bir mikroişlemci ile 6 eksenli robot kol için bir kumanda devresi geliştirmişlerdir. Aynı zamanda ters kinematik hesaplamalar yaparak, bilgisayar üzerinden kontrolü gerçekleştirmişlerdir.

R. C. Luo et al (2008), ALSOK Guard ROBOi insansı robot kullanarak güvenlik uygulamaları gerçekleştirmişlerdir. Robotun gözlerinde bulunan kameralar vasıtasıyla ortamın görüntüsü alınmış, müdahale edilmesi gereken nesneler tespit edilerek ters kinematik yöntemi ile müdahale edilmiştir. Toplamda 14 adet servo motor bulunan robot üzerinde mesafe ölçüm işlemleri için sensörler bulunmaktadır. Hedef nesneler eğer robotun kollarının çalışma aralığı dışında ise sensörler vasıtasıyla yakınlık ve uzaklık değiştirilerek nesneleri almayı başarmışlardır.

Deepthi ve Sankaraiah (2011), yaptıkları çalışmada Nokia firmasına ait bir mobil cihaz üzerinde Qt ve OpenCV ile birlikte görüntü işleme uygulaması gerçekleştirmişlerdir. Kameradan aldıkları görüntüyü, gri seviyeye dönüştürmüş, Gaussian Filtre ile yumuşatma işlemi yapmışlardır. Daha sonra histogram eşitleme yaparak kenar çıkarımı yapmışlardır. En son resimde istenilen kısmı kırparak mobil cihazın belleğine resim olarak kaydetmişlerdir.

Ali vd. (2012), yaptıkları çalışmada bilgisayara bağlı bir kameradan aldıkları görüntüde bulunan nesneleri şekil olarak incelemişlerdir. Yuvarlak, kare, dikdörtgen, kalp şeklinde olan nesneleri tanımlar ve bilgisayara bağlı bulunan robot kol ile istedikleri nesneleri alıp başka bir yere yerleştirmişlerdir.

Yukarıda belirtilen literatür taramaları göz önünde bulundurularak yapılan bu tez çalışmasında görüntü işlemeye dayalı robot kontrolü uygulaması üzerinde çalışılmıştır. Bu uygulamada, 3R robotun çalışma uzayı içerisinde bulunan topların yerleri görüntü işleme yöntemi ile tespit edilerek robotun bu topları istenen başka bir konuma taşınması amaçlanmış ve başarılı olunmuştur.

2.2 Robot

İnsanoğlu, hayatını kolaylaştırmak amacıyla teknolojik alanlarda birçok çalışma yapmıştır. Bilgisayar destekli tasarım (CAD) ve bilgisayar destekli üretim (CAM) alanlarındaki gelişmeler, endüstriyel otomasyon sistemlerinin hızla yaygınlaşmasını sağlamıştır. Robot, otomasyon sistemlerinin temelini oluşturan buluşlardan bir tanesidir.

Robot kelimesi, ilk olarak Çek yazar Karel Capek'in 1920 yılında yazdığı "R.U.R. - Rossum's Universal Robots" isimli tiyatro oyununda kullanılmıştır. Bu eserde robot diye isimlendirilen cihazlar, mekanik ve otonom olarak çalışan ve insan duyguları barındırmayan cihazlardır. Çek dilinde robot, köle veya işçi anlamına gelmektedir (Wikipedia, 2002).

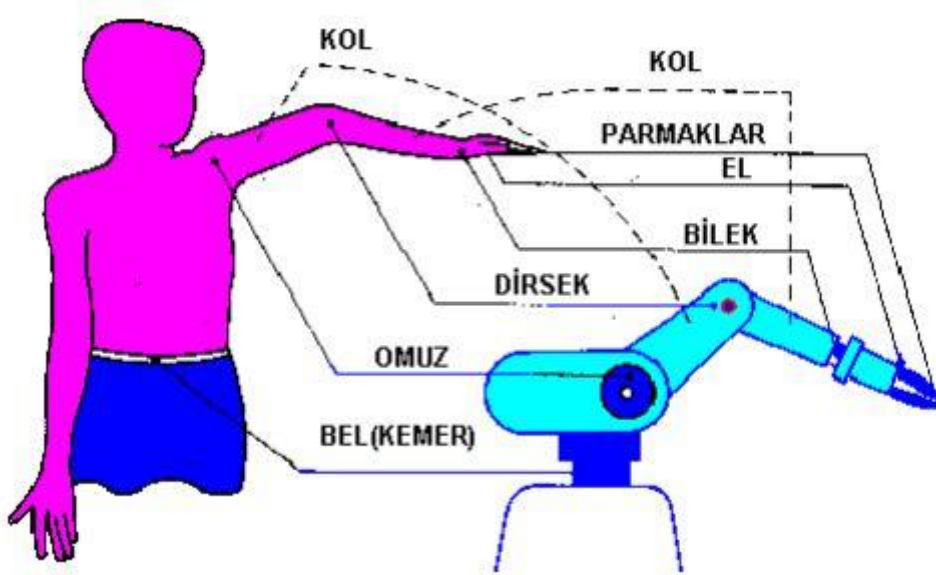
Robotik kelimesi ise ilk kez Isaac Asimov tarafından, 1941 yılında kullanılmıştır. Asimov ayrıca, tarihe "Üç Temel Robot Etik Yasası" olarak geçen kuralları koymuştur. Asimov, robot biliminin hızlı bir şekilde gelişeceği ve robotların insanların tüm işlerinde vazgeçilmez olacağını öngörmüştür (Wikipedia, 2001). Şu anda endüstride kullanılmakta olan yüz binlerce robotu düşünürsek, bu öngörüsünde ne derece başarılı olduğunu görürüz.

Robot bilimi matematik, kontrol, makine, elektrik ve bilgisayar mühendisliği gibi birçok bilim dalını içine alan disiplinler arası bir terbiyedir. Bu bilim dallarından makine mühendisliği robotların dinamik ve statik yapısını incelerken, matematik robotların hareketlerini tanımlayıp modelini çıkarır. Kontrol mühendisliği, kontrol algoritmaları geliştirerek sistemin istenilen hareketi gerçekleştirmesini sağlar. Elektrik mühendisliği, eyleyicilerin ve algılayıcıların tasarımıyla ilgilenir. Bilgisayar mühendisliği ise gerekli robot programlarının yapılmasında katkı sağlar. (Bingül ve Küçük,2009)

Amerikan Robot Enstitüsünün tanımına göre robot; "Çeşitli görevleri gerçekleştirmek için programlanmış hareketlerle malzeme,parça,takım ya da özel cihazları taşımak için tasarlanmış çok fonksiyonlu ve tekrar programlanabilir manipülatördür." Japonya'da ise robot; "Hafıza aygıtı takılmış, rutin işleri yaparak insanın yerini alabilen, belirlenmiş hareketleri otomatik olarak yerine getirebilen, gerekli ekipmanı tamamen sağlanmış makine" olarak tanımlanmaktadır. (Endüstri Mühendisliği Blogu, 2012)

Robot kavramının en kapsamlı tanımı ISO 8373 standardında belirtilmiştir. Bu standarda göre robot; "Endüstriyel uygulamalarda kullanılan, sabit veya hareketli olabilen, üç veya daha fazla programlanabilir eksene sahip, otomatik kontrollü, yeniden programlanabilir çok amaçlı manipulatördür." (Çengelci ve Çimen, 2005)

Robot kollar, insan koluna benzer şekilde tasarlanmış mekanik yapılardır. Şekil 2.1'de robot kolu ve insan kolunun benzerliği gösterilmiştir. İnsanların çalışmasının tehlikeli olabileceği veya çok hassas işçiliğin gerektiği uygulamalarda robot kollar tercih edilmektedir. Robot kollar tek başına bir sistem olabilecekleri gibi, daha kapsamlı bir robotun bir parçası olarak da tasarlanabilir. Örneğin mobil bir robotun ucuna bir robot kol eklenebilir ve böylece cisimler sabit bir robot kolun ulaşamayacağı uzaklıktaki yerlere rahatlıkla taşınabilir. Robot kolların çalışma kabiliyeti, barındırdıkları eksen sayısı ile doğru orantılıdır. Ancak eksen sayısı arttıkça robotun kontrolü ve programlanması da zorlaşmaktadır. Genellikle endüstriyel uygulamalarda en az üç, en fazla altı eksenli robot kolları tercih edilmektedir.



Şekil 2.1 Robot kolu ve insan kolu benzetimi (T.C. M.E.B., 2012)

2.2.1 Robotların avantaj ve dezavantajları

Her yeni teknolojinin insan hayatına sağladığı sayısız yarar kadar bazı zararları da bulunmaktadır. Bu bölümde bu avantaj ve dezavantajlardan bahsedilmiştir.

Robotların avantajları;

- Robotlar, girdiği bütün sektörlerde üretkenliği, güvenliği, verimliliği, kaliteyi ve tutarlılığı artırmıştır.
- Robotlar tehlikeli ortam şartlarında, hiçbir destek ve güvenlik önlemi gerektirmeksizin çalışabilirler.
- Robotlar yorulmadan, canı sıkılmadan ve tatil istemeden, her zaman aynı performans ve kararlılıkta çalışmaya devam ederler.
- Robotlar, başlangıçta kurulum maliyetleri yüksek olsalar da birkaç yılda kendilerini amorti ederler ve böylece bedavaya çalışan bir işçi konumuna gelirler.
- Bir insan ortalama 8 saat çalışabilecekken, robotlar bakım süreleri hariç 24 saat boyunca çalışabilir.
- İnsanlardan daha güçlüdür, daha ağır yükleri zorlanmadan kaldırabilir.
- Doğrulukları ve hassasiyetleri insanlara göre daha yüksektir. Yaklaşık olarak 0.01 mm hassasiyetle çalışabilen robotlar bulunmaktadır.

Robotların dezavantajları;

- Robotlar, endüstride insanların yerini alarak işsizliğe ve düşük maaşlara neden olurlar.
- Robotlar, çalışması sırasında oluşabilecek bir arıza durumunda operatörüne veya çalışma sahası içerisinde bulunan insanlara zarar verebilir.
- Robotlar, insanlar gibi görme, duyma, algılama, karar verme yeteneklerine sahip değildir. Bazı yetenekler robotlara sensörler, algılayıcı sistemler ve yazılımlar ile kazandırılmaya çalışılsa da insan gibi kendi kendine hızlı bir şekilde karar verme ve düşünebilmesi şu an için imkansızdır.
- Programlanarak çalıştıkları için, programın içine yazılmayan bir durum oluştuğunda hata verirler.
- Robotların parça ve bileşenleri pahalıdır.
- Yüksek yatırım maliyeti gerektirirler.

2.2.2 Robotların kullanıldığı endüstriyel alanlar

Robotlar nesne taşıma ve yerleştirme, paketlenme, yapıştırma, montaj, kesme, şekil verme, boyacılık, yüzey kaplama, taşlama, kaynaklama gibi

endüstriyel uygulamalarda sıklıkla kullanılmaktadır. Genel olarak robotların en sık kullanıldığı endüstri kolları şunlardır :

- Tarım, hayvancılık ve ormancılık,
- Gıda
- Üretim
- Yapı ve İnşaat sektörü
- Elektrik,su ve gaz dağıtım sektörü
- Maden
- Eğitim
- Tıp

2.2.3 Robotları tanımlamada kullanılan parametreler

İleriki konularda robotlar ile ilgili kullanılacak tanımlamalar aşağıda verilmiştir (Koca, 2007).

- **Eksen Sayısı:** Düzlemde bulunan bir noktaya ulaşabilmek için iki eksen gerekli iken; uzayda bulunan bir noktaya ulaşabilmek için üç eksen gereklidir. Robotun uç işlevcisi uzayda yani üç boyutta hareket ettiği için, yön kontrolü için en az üç eksene ihtiyaç vardır.
- **Serbestlik Derecesi:** Çoğunlukla robotun eksen sayısı ile aynı sayıdadır. Robotun uç işlevcisinin uzayda istenilen pozisyona gidebilmesi için üç, istenilen oryantasyonda bulunabilmesi için yine üç, toplamda altı serbestlik derecesine ihtiyacı vardır.
- **Çalışma Uzayı:** Robotun ulaşabildiği bütün noktaların toplamının oluşturduğu bölgedir.
- **Kinematik:** Nesnelerin devinimleriyle ilgilenir. Robot kinematiği ile robotun kuvvet, hız ve ivme analizleri yapılır.
- **Taşıma Kapasitesi:** Robotun taşıyabileceği en fazla yük miktarıdır.
- **Doğruluk:** Robotun gönderilen pozisyona ne kadar yaklaşabildiğini gösteren ölçüttür. Doğruluk; robotun hızı, çalışma uzayındaki yeri ve taşıdığı yükün ağırlığına göre değişebilir. Kalibrasyon işlemi uygulanarak robotun doğruluğu artırılabilir.
- **Tekrarlama Kabiliyeti:** Robotun daha önce ulaştığı bir pozisyona, ikinci bir sefer gönderildiğinde yine aynı hassasiyet ile ulaşabilmesi beklenir.

2.2.4 Robotu oluşturan parçalar

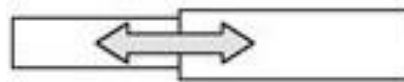
Robotlar temel olarak beş ana bölümden oluşmaktadır. Bunlar;

- **Gövde:** Bel, omuz ve eklemlerden oluşan ve sıralı olarak birbirine bağlı olan bölümdür.
- **Uç İşlevci:** Robotun yapması gereken görevleri gerçekleştiren bölümdür. Robottaki en aktif parçadır ve robotun en uç noktasında bulunur.
- **Yürütücüler:** Gövde ve uç işlevciyi hareket ettirmek için kullanılır. Kullandığı enerji kaynağına göre elektrikli, hidrolik ve pnömatik olan çeşitleri vardır. En çok elektrikli servo motorlar tercih edilmektedir.
- **Sensörler:** Akıllı robot teknolojilerinde kullanılırlar. Robotların kendi içerisinde bulunan mesafe ve konumlama sensörleri robotun aldığı mesafeyi ve verilen komutlara cevaplarını kontrol eder. Harici algılayıcılar ise robotların çalıştıkları alandaki iş parçalarını gözlemler. (T.C. M.E.B., 2012)
- **Kontrol Birimi:** Robotun hareketlerini denetleyen ve yöneten birimdir, diğer bir deyişle robotun beynidir. Robot bu birimden programlanır. Robota hem kontrol palsi gönderir hem de robottaki sensörlerden geri besleme olarak kapalı çevrim kontrolü sağlar.

2.2.5 Robotlarda kullanılan eklem yapıları

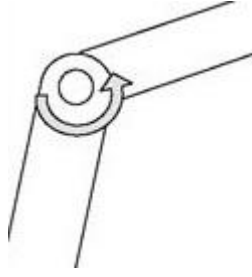
Bir robot yapısal olarak birbirine göre bağımsız hareket eden, prizmatik, döner, helikal, küresel ve silindirik hareketler gerçekleştiren eklemler ile, bu eklemleri birbirine birleştiren bağlardan oluşur. Robotikte en çok kullanılan eklem tipleri prizmatik ve döner eklem tipleridir.

- **Prizmatik (Prismatic - P) Eklemler:** İki uzuv arasında doğrusal hareket yapan eklem tipidir. Şekil 2.2'de örnek bir prizmatik eklem yapısı verilmiştir.



Şekil 2.2 Prizmatik eklem

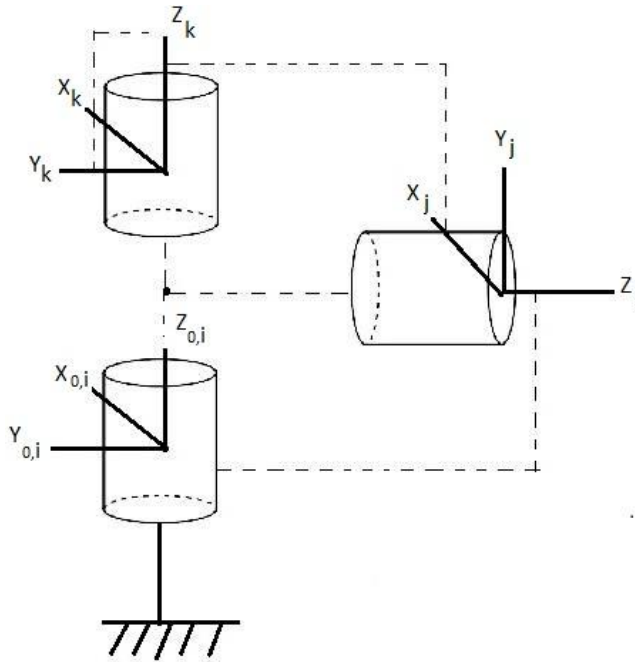
- **Döner (Revolute - R) Eklemler:** İki uzuv arasında dönme hareketi yapan eklem tipidir. Şekil 2.3'de örnek bir döner eklem yapısı verilmiştir.



Şekil 2.3 Döner eklem

2.2.6 Robotlarda kullanılan bilek yapıları

Endüstriyel robotlarda Euler ve eklem kaçıklıklı bilek olmak üzere genel olarak iki farklı bilek yapısı kullanılmaktadır. Euler bilekli yapıda üç eksen bir noktada kesişirken eklem kaçıklıklı bilek yapısında eksenlerin kesişmeleri d eklem kayması ve a bağ uzunluğu nedeniyle mümkün değildir. Şekil 2.4'de Euler bileğinin yapısı görülmektedir (Bingöl ve Küçük, 2009).



Şekil 2.4 Euler bileğinin yapısı

İleri bölümlerde anlatılacak olan ileri kinematik analizini, yeri gelmişken bilek yapıları için gösterelim. Konuyu anlamak için Bölüm 2.2.9.2'nin okunması gerekmektedir.

Tablo 2.1'de Euler bileğinin DH tablosu gösterilmiştir.

Tablo 2.1 Euler bileğinin DH tablosu

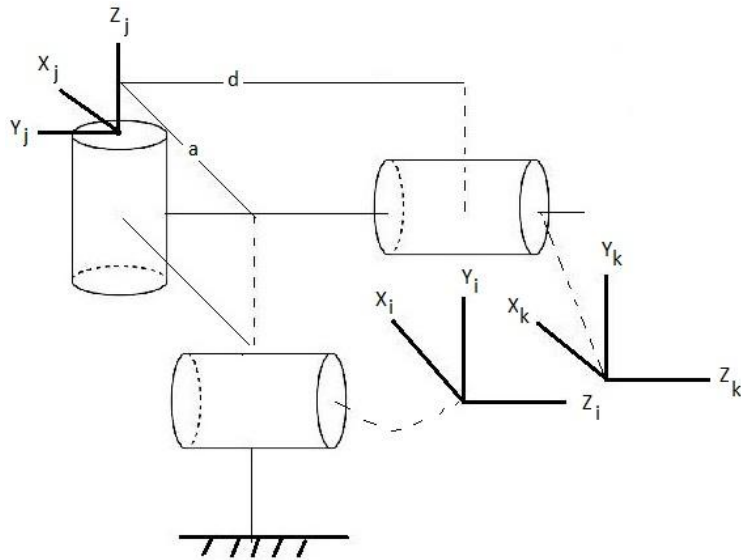
i	a_{i-1}	α_{i-1}	d_i	θ_i	Değişken
1	0	0	0	θ_i	θ_i
2	0	90	0	θ_j	θ_j
3	0	-90	0	θ_k	θ_k

Tablo 2.1'deki DH değişkenlerinden yararlanılarak Euler bileğinin ileri kinematik matrisleri aşağıdaki gibi bulunur.

$${}^0_k T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_j & -s\theta_j & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s\theta_j & c\theta_j & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_k & -s\theta_k & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s\theta_k & -c\theta_k & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Görüldüğü gibi matrislerin konum vektörleri $[0 \ 0 \ 0]^T$ şeklindedir. Bu yüzden Euler bilek yapısının, robotun konum bilgisine etkisi yoktur.

Şekil 2.5'te eklem kaçıklı bileğin yapısı gösterilmiştir.



Şekil 2.5 Eklem kaçıklı bileğin yapısı

Tablo 2.2'de eklem kaçıklı bileğin DH tablosu gösterilmiştir.

Tablo 2.2 Eklem kaçıklı bileğin DH tablosu

i	a_{i-1}	α_{i-1}	d_i	θ_i	Değişken
1	0	0	0	θ_i	θ_i
2	a	90	0	θ_j	θ_j
3	0	-90	d	θ_k	θ_k

Tablo 2.2'deki DH değişkenlerinden yararlanılarak eklem kaçıklı bileğin ileri kinematik matrisleri aşağıdaki gibi bulunur.

$${}^0_kT = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_j & -s\theta_j & 0 & a \\ 0 & 0 & -1 & 0 \\ s\theta_j & c\theta_j & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_k & -s\theta_k & 0 & 0 \\ 0 & 0 & 1 & d \\ -s\theta_k & -c\theta_k & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

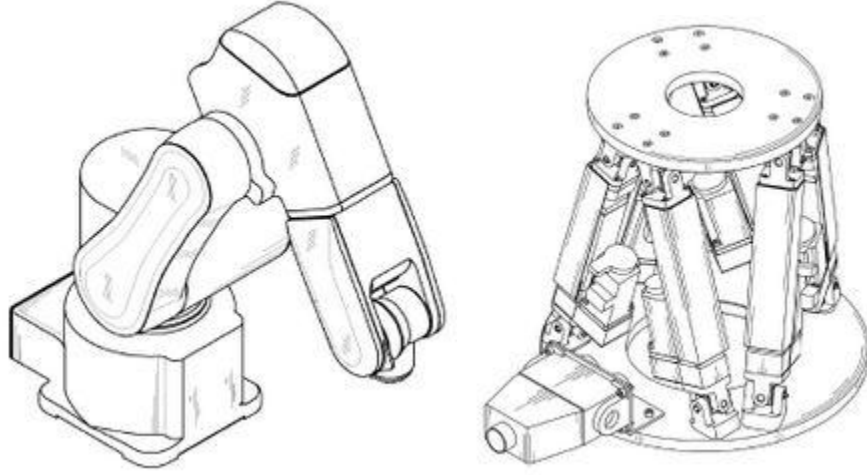
Görüldüğü gibi yukarıdaki 2. ve 3. matrisin konum vektörleri sıfır değildir. Bu nedenle eklem kaçıklı bilekler, robotun konum bilgisine etki ederler.

Eklem kaçıklı bileklerin kinematik analizi zordur. Bunun sebebi Bölüm 2.2.9.3'de anlatılacaktır. Endüstriyel uygulamalarda, analitik olarak hızlı bir şekilde analiz edilebilen Euler bilek yapısındaki robotlar tercih edilmektedir.

2.2.7 Robotların sınıflandırılması

Robotlar temelde seri robotlar (Şekil 2.6 a) ve paralel robotlar (Şekil 2.6 b) olmak üzere iki gruba ayrılır. Seri robotlar bir dizi eklemler ve bu eklemleri birbirine bağlayan bağlardan oluşur. Seri robotlar; geniş çalışma uzayına, az sayıda mekanik parçaya ve paralel robotlara göre daha basit kinematik denklemlere sahiptir. Fakat kendi ağırlıkları, kaldırabilecekleri yükün ağırlığına göre oldukça fazladır. İki eklem birbiri birden çok bağ ile bağlandığı paralel robotlar ise, seri robotlara göre daha sağlam bir yapıya sahiptir (Bingül ve Küçük, 2009). Paralel robotlarda bütün eyleyiciler aynı anda çalıştığı için kendi ağırlıklarından çok daha fazla yükü taşıyabilirler. Ancak mekanik yapılarının

karmaşıklığı nedeniyle kinematik denklemleri de oldukça karmaşıktır. Paralel robotlarda daha hassas işler yaptırılabilir. İnsan eli bütün olarak düşünüldüğünde paralel bir mekanizmadır. Parmaklar ise seri mekanizmaya örnek gösterilebilir.



Şekil 2.6 a) Seri robot örneği, b) Paralel robot örneği (Bonev, 2013)

Bu iki temel sınıflandırma dışında endüstriyel robotlar aşağıdaki 5 bölüme göre sınıflandırılabilir:

- Manipülâtörün (mekanik gövde) yapısına göre
- Hareket yöntemine göre
- Kontrol döngüsüne göre
- Kullandıkları enerji çeşidine göre
- Kesinlik derecelerine göre

2.2.7.1 Manipülâtörün yapısına göre robotlar

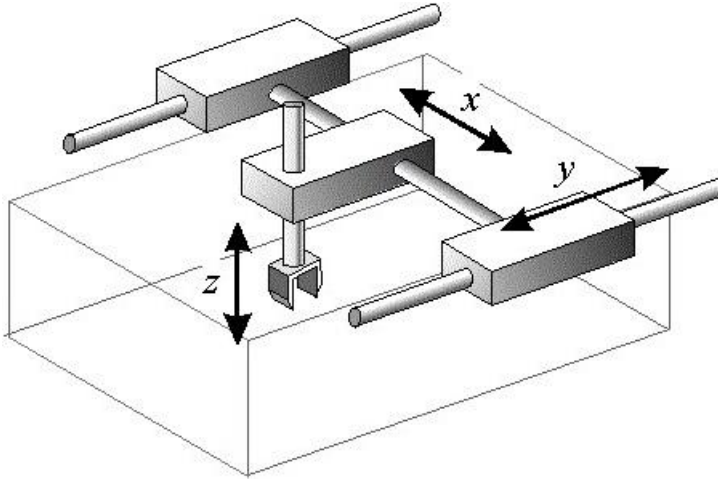
Bu tarz sınıflandırmada, robotun ilk üç ekleminin yapısı göz önüne alınır. Prizmatik ve döner eklemlerin çeşitli kombinasyonlarına göre oluşturulmuş ve endüstride yer tutmuş sınıflandırmalar aşağıda verilmiştir. Burada P harfi prizmatik ekleme, R harfi ise döner ekleme temsil etmektedir.

- Kartezyen Robotlar (PPP)
- Silindirik Robotlar (RPP)
- Küresel Robotlar (RRP)
- SCARA (Selective Compliant Assembly Robot Arm) Robotlar (RRP)

- Mafsallı Robotlar (RRR)

Kartezyen Robotlar

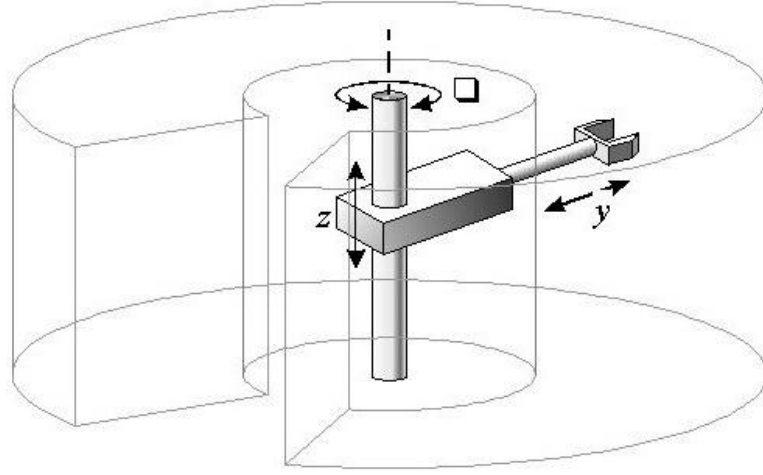
En kısıtlı hareket kabiliyetine sahip olan kartezyen robotun ana gövdesi, birbirine dik açılı üç tane prizmatik eklemden oluşur. Her bir eklem kartezyen koordinat sistemine (X,Y,Z) paralel hareket eder. Üç tane düzgün hatta hareket ettiği için hareket koordinatlarını tespit etmek kolaydır ve kinematik problemleri basittir. İstenilen bir pozisyona ulaşması için yapılacak hesaplar çok basit olarak bulunur. Çalışma uzayı dikdörtgen prizmadır ve robotun kapladığı alandan daha küçüktür. Bu robotlar genellikle CNC tezgahlarında, ağır yükleme kapasitesinin ihtiyaç olduğu işlerde ve mermer, cam ve ahşap işlemede kullanılır. Şekil 2.7'de kartezyen robotun yapısı ve çalışma uzayı gösterilmiştir.



Şekil 2.7 Kartezyen robot ve çalışma uzayı (Menevşe, 2013)

Silindirik Robotlar

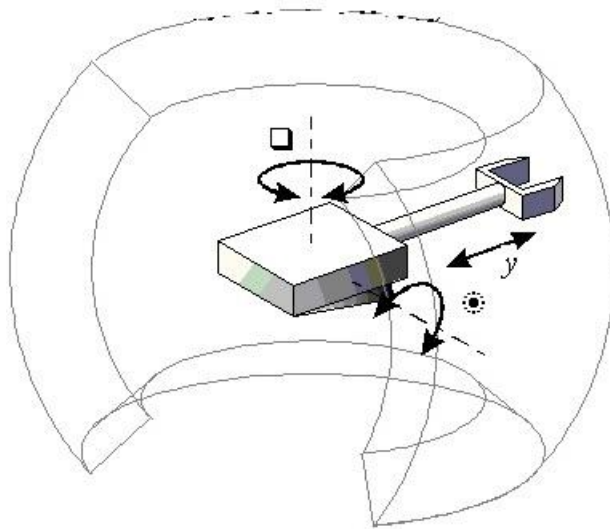
Bir tane döner, iki tane prizmatik eklemden oluşan robot çeşididir. Kartezyen robottaki prizmatik olan birinci eklem yerine bu robotta döner eklem gelmiştir. Çalışma uzayları dairesel bir silindire benzer olduğu için silindirik robot olarak adlandırılırlar. Mekanik yapıları basittir ve buna bağlı olarak programlanmaları kolaydır. Kartezyen robotlara göre hareket kabiliyeti daha yüksektir. Yapılması istenen işin, hedef ile aynı doğrultuda olmadığı durumlarda, yükleme, boşaltma ve montaj gibi işlemlerde kullanılmaya uygundur. Oldukça yüksek ağırlıkları taşıyabilirler. Şekil 2.8'de silindirik robotun yapısı ve çalışma uzayı gösterilmiştir.



Şekil 2.8 Silindirik robot ve çalışma uzayı (Menevşe, 2013)

Küresel Robotlar

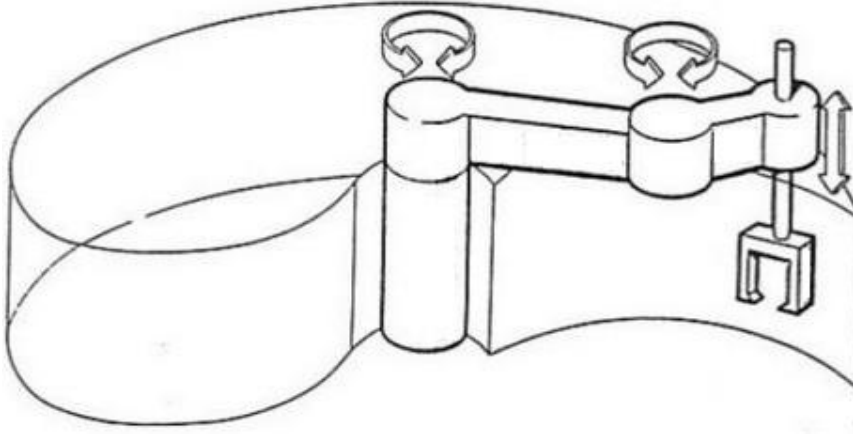
İlk iki eklemi döner, üçüncü eklemi ise prizmatik yapıdadır. Bu robotun yapısı askeri tankların kule kısmına benzemektedir. Küresel robotlar, mekanik bakımdan silindirik ve kartezyen robotlara göre daha zayıf ve daha karmaşıktır. Kolda bulunan eklemlerden sonuncusu doğrusal yönde hareket yapmakta, ilk eklem temele dik eksen etrafında, ikinci eklem ise ilk eksene dik ve temel eksene paralel bir eksenin etrafında dönmektedir. Çalışma uzayları küresel bir kabuk şeklindedir. Cilalamada, transfer işlemlerinde ve makine montajlarında kullanılmaya uygundur. Şekil 2.9'da küresel robotun yapısı ve çalışma uzayı gösterilmiştir.



Şekil 2.9 Küresel robot ve çalışma uzayı (Menevşe, 2013)

SCARA Robotlar

Eklem yapısı itibariyle küresel robotlara benzemesine rağmen eklemlerin geometrisi açısından tamamen farklıdır. SCARA robotta ilk iki eklem döner, üçüncü eklem ise prizmatiktir ve her üç eklem de birbirine paraleldir. Çalışma uzayı silindirikdir. Hız ve konum performansı çok iyi olduğundan genellikle elektronik kartlara malzeme montajı işleminde, parçaların test edilmesinde, nesne toplama ve yerleştirme işlemlerinde kullanılmaktadır. Taşıma kapasiteleri en fazla 20 kg.'a kadar olmasına rağmen dikey yönde çok hızlı hareket etmesi ve çalışma uzayı içerisinde istediği herhangi bir noktaya çok çabuk ulaşılabilmesinden dolayı özellikle montaj işlemlerinde sıklıkla kullanılırlar. Şekil 2.10'da SCARA robotun yapısı ve çalışma uzayı gösterilmiştir.

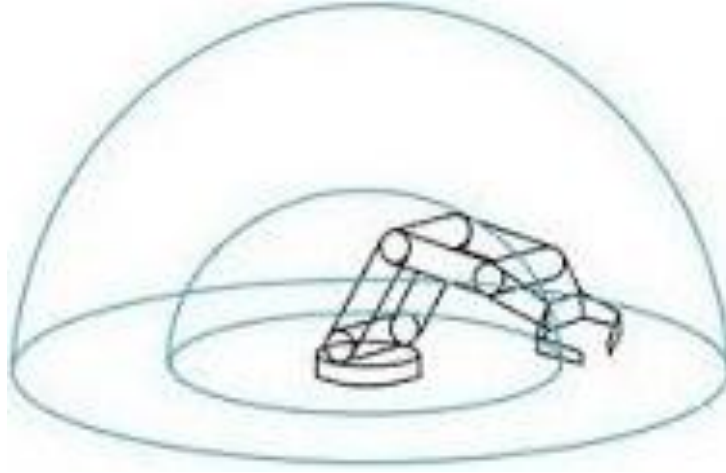


Şekil 2.10 SCARA robot ve çalışma uzayı (Menevşe, 2013)

Mafsallı Robotlar

Mafsallı robotlar, tamamı döner hareket yapan bel, omuz ve dirsek mafsallarından oluşur. İnsan kolunun hareketine en yakın hareket eden robot çeşididir ve bu yüzden anthropomorphic (insansı) robot olarak da adlandırılır. Karmaşık yapılarından dolayı kontrolleri zordur ve bu nedenle geç kullanılmaya başlanmıştır. En yüksek hareket kabiliyetine sahip olan bu robotlar, iş yapma yeteneği olarak da diğer robot çeşitlerinden daha kabiliyetli ve daha hızlıdır. Kullanılacağı uygulamaya göre eksen sayısı seçilir. Basit uygulamalarda 3 eklemlili bir robot yeterli olabilecek iken daha komplike bir uygulamada yeterli olmayabilir. Eklem sayısının artması robotun hareket serbestliğini artırmaktadır ve buna paralel olarak kontrolünü de zorlaştırmaktadır. Şekil 2.11'de mafsallı

robotun yapısı ve çalışma uzayı gösterilmiştir. Boyama, montaj, kaynaklama ve yüzey temizliği gibi geniş bir kullanım alanına sahiptirler.



Şekil 2.11 Mafsallı robot ve çalışma uzayı (Menevşe, 2013)

Robot çeşitlerinin karşılaştırılması

Tablo 2.3'de, yukarıda bahsedilen robot çeşitlerinin eklem tipleri, kullanım alanları ve kullanım sonuçlarına göre karşılaştırılması verilmiştir. Tablodan da görüleceği gibi robot çeşitleri arasında endüstride en sık kullanılan mafsallı robotlardır.

Tablo 2.3 Robot çeşitlerinin karşılaştırılması

Robot Tipi	<u>Eklem Tipleri</u> -Birinci eklem -İkinci eklem -Üçüncü eklem	Kullanım Alanları	<u>Kullanım Sonuçları</u> -Avantajları -Dezavantajları
Kartezyen Robotlar	Prizmatik Eklem Prizmatik Eklem Prizmatik Eklem	-CNC tezgahları, -Demiryolu ve köprü inşaatları, -Büyük makine montajları	-Kinematik modelleri basittir. - Rijit bir gövdeye sahiptir. -Hareket analizleri basittir.
			-Çalışması için büyük alan gerektirir. -Büyüklüğüne göre iş alanı küçüktür.

Tablo 2.3 Robot çeşitlerinin karşılaştırılması (devam)

Silindirik Robotlar	Döner Eklem Prizmatik Eklem Prizmatik Eklem	-İş ve hedefin aynı doğrultuda olmadığı işler, -Büyük makine montaj sanayisinde, -Basit montaj ve demontaj işleri	-Kinematik modelleri basittir. -Hareket analizi basittir.
			-İş alanları sınırlıdır. -Tozlu ve ıslak ortamda çalışmaları zordur.
Küresel Robotlar	Döner Eklem Döner Eklem Prizmatik Eklem	-Cilalama, -Transfer işlemleri, - Makine montajları, -Nükleer Santral	-Büyük alanlara uzanabilirler. -Zeminden uzaktaki nesneleri tutabilirler.
			-Kinematik modelleri karışıktır. -Hareket analizi zordur.
SCARA Robotlar	Döner Eklem Döner Eklem Prizmatik Eklem	-Elektronik kartlara malzeme montajı, - Parça test etme, - Malzeme toplama ve yerleştirme	-Çok hızlı hareket yeteneği vardır. -Yüksek doğrulukta çalışır.
			-Ağır yük taşıyamazlar
Mafsallı Robotlar	Döner Eklem Döner Eklem Döner Eklem	-Otomobil sanayisinde boyama, kaynaklama, montaj ve yüzey temizleme işleri, -Nükleer Santral, -Elektronik montaj sanayi	-Maksimum esnekliğe sahiptir. -İş alanı robot büyüklüğüyle doğru orantılıdır. -Cisimleri altlarından tutabilir.
			-Kinematik yapıları karmaşıktır. -Hareket analizi zordur.

2.2.7.2 Hareket yöntemine göre robotlar

Hareket yöntemine göre sınıflandırma, robotların belirlenen hedeflere nasıl ulaştıkları ile ilgilidir. Hareket yöntemine göre iki tür robot vardır:

- Noktadan Noktaya Hareket Eden Robotlar (Point to Point Robotic System - PTP)
- Sürekli Yörünge Kontrollü Robotlar (Continuous Path Robotic System - CP)

Noktadan Noktaya Hareket Eden Robotlar

Bu tür robotlarda, robot sayısal olarak tanımlanan noktaya gider ve hareketini durdurur. Robota sadece ulaşılması istenen koordinat değerleri gönderilir. Robotun hareketi sırasındaki hızının ve takip ettiği yörüngenin bir önemi yoktur.

Bu hareket yöntemi genellikle bir nesneyi bir yerden başka bir yere yerleştirme (tut ve yerleştir, pick and place) işlerinde kullanılan robotlarda ve nokta kaynağı yapan robotlarda kullanılmaktadır. (Bingül ve Küçük,2009)

Sürekli Yörünge Kontrollü Robotlar

Bu tür robotlarda, belirli bir yörünge takip edilir ve bu yörüngenin dışına çıkılmaz. Eksenlerin hareketleri farklı hızlarda ve aynı anda gerçekleşebilir. Robotun hareketi kontrol devresinin denetiminde gerçekleşir. Robot düzgün ve kesiksiz bir hareket yapar. Robotun hareketi boyunca geçtiği tüm noktalar kaydedilir.

Bu hareket yöntemi genellikle ark kaynağı, spreyci boyama, metal parçalarının pürüzsüz yüzey işlemleri ve karmaşık montaj uygulamalarında kullanılan robotlarda tercih edilir. (Çiçek, 2006)

2.2.7.3 Kontrol döngüsüne göre robotlar

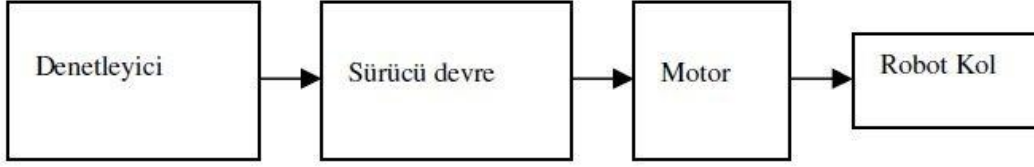
Robotlar kontrol döngüsüne göre ikiye ayrılırlar:

- Açık Döngü Kontrol Yöntemi
- Kapalı Döngü (Geri Beslemeli) Kontrol Yöntemi

Açık Döngü Kontrol Yöntemi

Açık döngü kontrol yönteminde, sistemin o andaki mevcut durumu ile alakalı herhangi bir bilgi denetleyiciye gelmemektedir. Açık döngü robot kontrolünde, robota belirli bir konuma gitmesi için komut gönderilir ancak gerçekte robotun o noktaya gidip gitmediği kontrol edilmez. Bu kontrol yöntemi, çok iyi planlanmış durumlarda ve çevrenin sistemi etkilemediği durumlarda kullanılabilir. Açık döngü kontrol yönteminde adım motorlar kullanılırsa istenilen

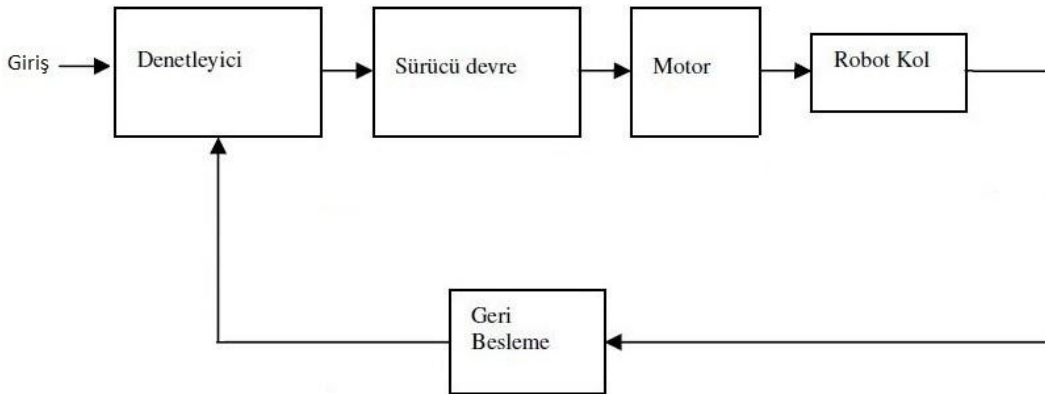
doğruluk oranını elde etmek mümkün olabilir. Kurulum maliyeti olarak kapalı döngü kontrol sistemlerinden daha ucuzdur. Açık döngü kontrol yöntemi ile kontrol edilen örnek bir sistem, Şekil 2.12'de verilmiştir.



Şekil 2.12 Açık döngü kontrol sistemi örneği

Kapalı Döngü (Geri Beslemeli) Kontrol Yöntemi

Bu kontrol yönteminde, sensörlerden alınan sistemin mevcut durumu hakkındaki bilgiler kontrolöre gönderilmekte ve kontrolör çıkıştaki hataya bağlı olarak girişi değiştirmekte ve istenen çıkışı yakalamaya çalışmaktadır. Kapalı döngü robot kontrol yönteminde, robot gitmesi gereken konuma olan uzaklığını yani hatayı sürekli olarak kontrol eder ve istenen konuma ulaşıp beklemeye geçer. Hangi zamanda hangi konumda olduğu sürekli olarak bilinir. Hassas ve kesin bir kontrol uygulaması yapılmak istendiğinde bu yöntem kullanılır. Kapalı döngü kontrol yöntemi ile kontrol edilen örnek bir sistem, Şekil 2.13'de verilmiştir.



Şekil 2.13 Kapalı döngü kontrol sistemi örneği

2.2.7.4 Kullandıkları enerji çeşidine göre robotlar

Robotlarda kullanılan motorlar, aşağıdaki üç şekilde tahrik edilir:

- Elektrik Tahrikli Motorlar
- Hidrolik Tahrikli Motorlar
- Pnömatik Tahrikli Motorlar

Elektrik Tahrikli Motorlar

Robotik alanında en fazla tercih edilen tahrik sistemi elektrik tahrikli motorlardır. Elektrik motorlarının yetenekleri fazladır ve bakım ihtiyaçları yok denecek kadar azdır. Boyut olarak küçük oldukları için az yer kaplarlar. Robotikte en sık kullanılan elektrik motorları çeşitleri şunlardır:

- DC Servo Motorlar: Düşük gerilimle yüksek tork ürettikleri için sıklıkla tercih edilirler. Üzerinde geri besleme bilgisi sağlayan enkoder mevcuttur.
- AC Servo Motorlar: Yüksek dinamik özelliklere sahiptirler.
- Adım Motorlar: Basit uygulamalarda kullanılırlar. Yüksek hassasiyete sahiptir. Geri besleme bilgisine ihtiyaç duymadan çalıştırılırlar.

Hidrolik Tahrikli Motorlar

Akışkanların hareket ettirilmesi mantığına göre çalışırlar. Hidrolik motorlarında genellikle basınçlı yağ kullanılır. Bu motorların tükettikleri düşük güce karşılık ürettikleri güç çok yüksektir bu yüzden ağır endüstride kullanılırlar. İçerisinde çok fazla bileşen barındırır (döner silindir, pompa, depo, soğutma sistemi). Yapıları doğrusal olmadığı için elektrik motorlarına göre kontrolleri daha zordur. Yavaş çalışırlar ve genellikle bulundukları ortamı yağ sızdırırlar.

Pnömatik Tahrikli Motorlar

Robotlarda kullanılan en basit tasarıma sahip motorlardır. Prensipde hidrolik motorlarına benzerdir ancak bu motorlarda akışkan olarak basınçlı yağ yerine basınçlı hava kullanılır. Hidrolik sistemlere göre daha düşük güç üretirler ve daha ucuzdurlar. Hareket eden robot pistonlarının ataletini hızla ortadan kaldıracak yüksek hava basıncının üretilmemesinden dolayı pnömatik sürücülerin kontrolü zordur. Basit uygulamalarda kullanılırlar (Bingöl ve Küçük, 2009).

Tablo 2.4'de tahrik sistemlerinin karşılaştırılması yapılmıştır.

Tablo 2.4 Elektrik, hidrolik ve pnömatik tahrik sistemlerinin karşılaştırılması

	ELEKTRİK TAHRİKLİ MOTORLAR	HİDROLİK TAHRİKLİ MOTORLAR	PNÖMATİK TAHRİKLİ MOTORLAR
AVANTAJLARI	-Hızlı ve hassastır. -Karmaşık kontrol tekniklerini uygulamaya uygundur. -Kolay kullanımlıdır. -Ucuzdur. -Neredeyse hiç bakım gerektirmez. -Az yer kaplar.	-Çok yüksek güçlere sahiptir. -Yağ basıncı azalmadığından eklemler hareket ettirilmeden sabit kalabilir. -Hassas kontrol yapılabilir.	-Hidrolik sistemlerden daha ucuzdur. -Yüksek hızlara çıkabilir. -Basit bir tasarımı vardır.
DEZAVANTAJLARI	-Dişli ve güç aktarma organlarına gereksinim duyar. -Düşük güçlüdür. -Oluşan elektrik arkı, kimi problemlere sebebiyet verebilir.	-Pahalıdır. -Yağ sızması ile çevreyi kirletebilir. -Yüksek hızlara çıkamaz. -İçerisinde çok bileşen barındırır.	-Havanın sıkıştırılabilirlik özelliğinden dolayı basınç kaybı oluşur. -Elektrikli sistemlere göre daha pahalıdır. -Sürekli bakım ister. -Kontrolü zordur.

2.2.7.5 Kesinlik derecelerine göre robotlar

Kesinlik derecesine göre robotlar aşağıdaki üç şekilde sınıflandırılırlar:

- Çözünürlük (resolution)
- Doğruluk (accuracy)
- Yinelenebilirlik (repeatability)

Çözünürlük

Çok küçük bir yer değiştirmeyi gerçekleştirme yeteneği olarak tarif edilir. Gerçekleşen teknolojik gelişmelere bağlı olarak robotların çözünürlükleri her geçen gün artmaktadır. Günümüz teknolojisinde 0.001 mm çözünürlüklere ulaşan robotlar mevcuttur ve bunlar ağırlıklı tıp sektöründe ameliyatlarda kullanılmaktadır.

Doğruluk

Daha önce de belirtildiği gibi doğruluk; robotun gönderilen pozisyona ne kadar yaklaşabildiğini gösteren ölçüttür.

Yinelenebilirlik

Robotun uç işlevcisinin, birbirinden farklı birçok işlemi gerçekleştirdikten sonra tekrar aynı konuma gelebilme ölçüsüdür. Özellikle tut ve yerleştir uygulamalarında büyük önem taşır. Elektronik devrelere malzeme montajı yapan robotlar buna örnek verilebilir. Günümüzde kullanılan endüstriyel robotların yinelenebilirliği 0.05 ile 0.005 mm arasındadır (Bingül ve Küçük, 2009).

- Tüm bu sınıflandırma yöntemleri göz önüne alındığında, bu tez çalışmasında kullanılan robot 1. sınıflandırmaya göre mafsallı robotlar sınıfında, 2. sınıflandırmaya göre noktadan noktaya hareket eden robotlar sınıfında, 3. sınıflandırmaya göre kapalı döngü kontrol sistemiyle kontrol edilen robotlar sınıfında, 4. sınıflandırmaya göre ise elektrik motorlu robotlar sınıfında yer almaktadır.

2.2.8 Robot programlama yöntemleri

Robotların programlanmasında kullanılan üç temel yöntem vardır. Bu yöntemler;

- Fiziksel Kurma
- Önceden Öğretme
- Yazılım ile Programlama

Bu yöntemlerden hangisinin seçileceğine, robotun yapısına ve kullanım amacına göre karar verilir.

2.2.8.1 Fiziksel kurma yöntemi

Operatör, elinde bulunan anahtarları açarak veya kapatarak robotun, manüel olarak hareket etmesini sağlar.

2.2.8.2 Önceden öğretim

Bu yöntemde, öğretim kutusu (teach pendant) denilen cihaz sayesinde robot istenilen noktalara hareket ettirilir. Ayrıca öğretim kutusu üzerinden robot programlanabilir ve robota ait tanımlama parametreleri girilebilir. Bu yöntem, özellikle tut ve yerleştir uygulamalarında büyük kolaylık sağlamaktadır. Nesne hangi noktadan alınacak ve hangi noktaya bırakılacak ise bu noktalar hafızaya alınarak robota öğretilir. Robotun belirlenen noktalara hareketini, istenildiği sayıda tekrar etmesi sağlanabilir. Bu yöntemin kullanılabilmesi için operatör olması şarttır.

2.2.8.3 Yazılım ile programlama

Bu yöntemde robotun gideceği noktalar, dönüş yapacağı konumlar, hareketi sırasında istenilen hız değeri gibi bilgiler önceden yazılmış olan bir program ile robota gönderilir. Robotun yapısına bağlı olarak (özellikle eklem sayısı) yazılacak olan programın uzunluğu ve karmaşıklığı artacaktır. Bu yöntemde, robota kamera ile görme gibi özellikler eklenerek ve buna uygun yazılımlar yazılarak, robotun otomatik olarak çalışması sağlanabilir. Arayüz programı tasarlanarak yazılıma görsellik katılabilir.

2.2.9 Robot kinematik analizi

Geometrik bir sistemin yapısını ve hareketini inceleyen bilim dalına kinematik denir. Robotik biliminin analitik araçlarının en önemlilerinden biri olan kinematik modelleme, robot biliminin birçok alanında kullanılmaktadır. Tasarlanan her robotun kinematik analizinin yapılması gerekmektedir. Robotların kinematik analizi ileri kinematik ve ters kinematik olmak üzere ikiye ayrılır. Bu iki yöntemden bahsetmeden önce, bir sistemin kinematik modelinin çıkarılabilmesi için bilinmesi gereken temel matematiksel tanımlardan bahsedelim. Bu matematiksel tanımlar, nesnenin konumu ve yönelimi, genel dönüşümler, öteleme ve dönme işlemleridir.

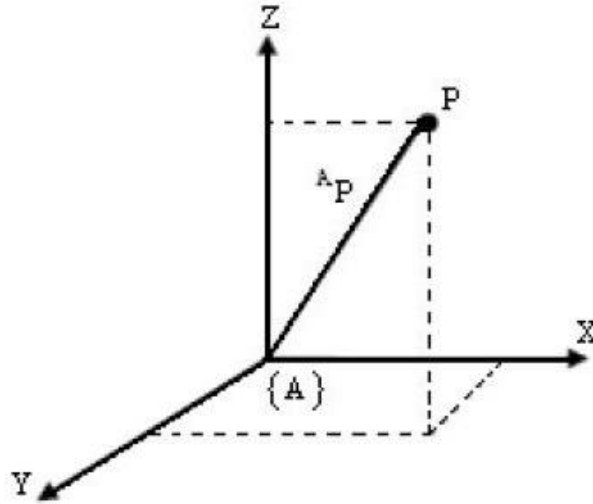
2.2.9.1 Temel matematiksel tanımlar

Robotlar, üç boyutlu uzayda çalışırlar. Robotun ve etrafında bulunan nesnelerin uzaydaki konumlarını ve birbirlerine göre olan yönelimlerini belirleyebilmek amacıyla robotun ve etrafında bulunan nesnelerin merkezlerine

birer adet koordinat sistemi yerleştirilir. Bu bölümde koordinat sistemleri arasında nasıl ilişki kurulacağı anlatılacaktır.

Konum Tanımı

Üç boyutlu uzayda bulunan bir noktanın konumu, yerleştirildiği koordinat sisteminin merkezine göre 3x1 boyutlu bir vektörle ifade edilir. Örnek olarak {A} koordinat sistemine göre bir P noktası, Şekil 2.14'de gösterildiği gibi tanımlanır.

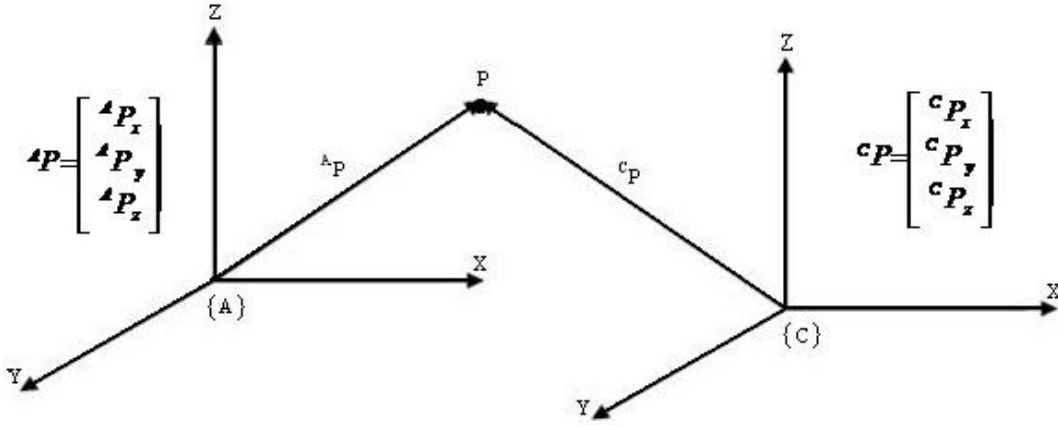


Şekil 2.14 Bir noktanın bir koordinat sistemine göre tanımlanması

Şekil 2.14'de görüldüğü gibi P noktasının A koordinat sisteminin merkezine olan uzaklığı, bir doğru ile gösterilir ve bu doğrunun uzunluğu denklem 2.1'de gösterildiği gibi (x,y,z) eksenlerindeki sayısal değerler ile ifade edilir.

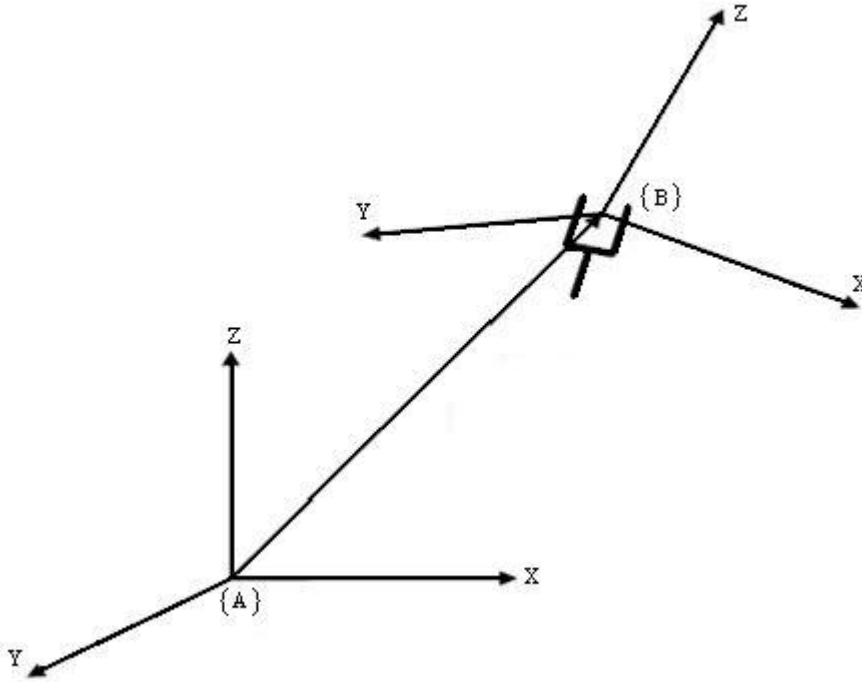
$${}^A P = \begin{bmatrix} {}^A P_x \\ {}^A P_y \\ {}^A P_z \end{bmatrix} \quad (2.1)$$

Bir nokta, kendisinden uzaklığı önemli olmaksızın aynı anda birden fazla koordinat sistemine göre tanımlanabilir. Örnek olarak bir P noktasının, birbirinden bağımsız {A} ve {C} koordinat sistemlerine göre tanımlanması Şekil 2.15'de gösterilmiştir.



Şekil 2.15 Bir noktanın iki farklı koordinat sistemine göre tanımlanması

Bir robotun uç işlevcisinin uzaydaki bir noktaya olan uzaklığını bulmak için, hem robotun uç işlevcisine hem de uzaydaki noktaya koordinat sistemi yerleştirilir. Daha sonra, yine önceki örneklerde gösterildiği gibi bir doğru çizilerek aralarındaki uzaklık ifade edilir. Şekil 2.16'da bu durum gösterilmiştir.



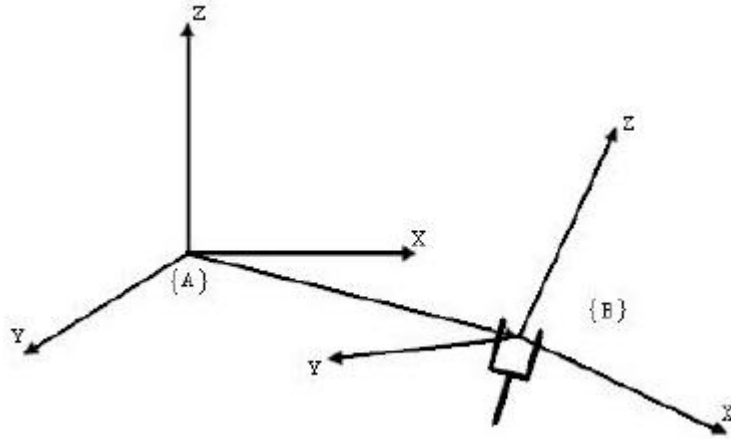
Şekil 2.16 Bir robotun uç işlevcisinin bir noktaya göre tanımlanması

Burada uç işlevcinin merkezinin A noktasına uzaklığı denklem 2.2'de gösterildiği gibi ifade edilir.

$${}^A P_B = \begin{bmatrix} {}^A P_{B_x} \\ {}^A P_{B_y} \\ {}^A P_{B_z} \end{bmatrix} \quad (2.2)$$

Yönelim Tanımı

Üç boyutlu uzayda, bir noktanın bir koordinat sistemine göre konumu ile birlikte yönelimi de tanımlanır. Yönelim, bir koordinat sisteminin başka bir koordinat sistemine göre dönme miktarıdır ve 3x3 boyutlu bir matrisle ifade edilir. Bir katı cismin yöneliminin başka bir referans koordinat sistemine göre tanımlanması katı cisme bir koordinat sistemi yerleştirilir. Şekil 2.17'de robotun uç işlevcisine {B} koordinat sistemi yerleştirilmiştir ve bir {A} referans koordinat sistemine göre yönelim tanımı gösterilmiştir (Bingül ve Küçük, 2009).



Şekil 2.17 Bir cismin referans bir koordinat sistemine göre yönelimi

Uç işlevcisine yerleştirilen {B} koordinat sistemini, {A} referans sistemi cinsinden ifade etmek için birim vektörler kullanılır. {B} koordinat sisteminin birim vektörleri denklem 2.3'de, {B} koordinat sisteminin birim vektörlerinin {A} koordinat sistemi cinsinden ifadesi denklem 2.4'te gösterilmiştir (Bingül ve Küçük, 2009).

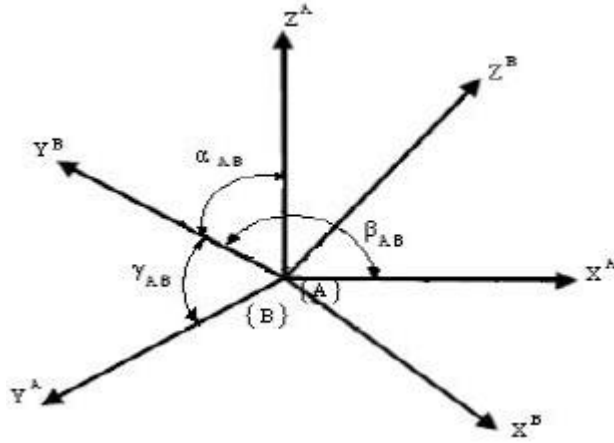
$$\{B\} = \hat{X}_B, \hat{Y}_B, \hat{Z}_B \quad (2.3)$$

$$\{A\} = {}^A \hat{X}_B, {}^A \hat{Y}_B, {}^A \hat{Z}_B \quad (2.4)$$

Denklem 2.4'teki ifadeyi 3x3 boyutlu bir matrisle denklem 2.5'deki gibi ifade edebiliriz. 3x3 şeklinde yazılan bu matrise dönme (rotation) matrisi denir. Bu matris $\{B\}$ koordinat sisteminin $\{A\}$ koordinat sistemine göre (x,y,z) eksenlerindeki dönme miktarlarını gösterir (Bingül ve Küçük, 2009).

$${}^A_B R = \begin{bmatrix} {}^A\hat{X}_B & {}^A\hat{Y}_B & {}^A\hat{Z}_B \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad (2.5)$$

Denklem 2.5'teki matrisin her bir kolonu birer birim vektördür. Bu birim vektörler $\{B\}$ koordinat sisteminin eksenlerinin doğrultusunu $\{A\}$ koordinat sistemine göre tanımlar.



Şekil 2.18 Merkezleri çakışık, yönelimleri farklı iki koordinat sistemi

Şekil 2.18'de verilen koordinat sistemlerinin merkezleri çakışık olmasına karşın yönelimleri farklıdır. Bu durumda $\{B\}$ koordinat sisteminde yer alan $\hat{X}_B$ birim vektörünün yönelimi $\{A\}$ koordinat sistemine göre denklem 2.6'daki gibi tanımlanır (Bingül ve Küçük, 2009).

$${}^A\hat{X}_B = \begin{bmatrix} r_{11} \\ r_{21} \\ r_{31} \end{bmatrix} = \begin{bmatrix} |X_B||X_A|\cos\gamma_{AB} \\ |X_B||Y_A|\cos\beta_{AB} \\ |X_B||Z_A|\cos\alpha_{AB} \end{bmatrix} = \begin{bmatrix} \hat{X}_B \cdot \hat{X}_A \\ \hat{X}_B \cdot \hat{Y}_A \\ \hat{X}_B \cdot \hat{Z}_A \end{bmatrix} \quad (2.6)$$

2.6'daki denklem sadece $\hat{X}_B$ birim matrisine göre yazılmıştır. Tüm yönelim matrisi denklem 2.7'deki gibi bulunur.

$${}^A_B R = \begin{bmatrix} {}^A\hat{X}_B & {}^A\hat{Y}_B & {}^A\hat{Z}_B \end{bmatrix} = \begin{bmatrix} \hat{X}_B \cdot \hat{X}_A & \hat{Y}_B \cdot \hat{X}_A & \hat{Z}_B \cdot \hat{X}_A \\ \hat{X}_B \cdot \hat{Y}_A & \hat{Y}_B \cdot \hat{Y}_A & \hat{Z}_B \cdot \hat{Y}_A \\ \hat{X}_B \cdot \hat{Z}_A & \hat{Y}_B \cdot \hat{Z}_A & \hat{Z}_B \cdot \hat{Z}_A \end{bmatrix} \quad (2.7)$$

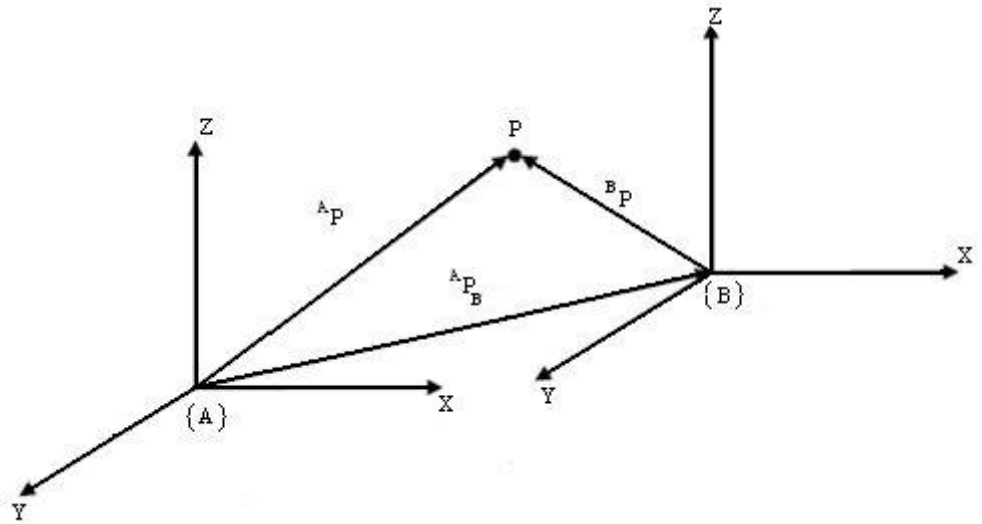
Denklem 2.7'yi daha açık bir şekilde denklem 2.8'deki gibi ifade edebiliriz.

$${}^A_B R = \begin{bmatrix} |X_B||X_A|\cos\gamma_{AB} & |Y_B||X_A|\cos\theta_{AB} & |Z_B||X_A|\cos\varphi_{AB} \\ |X_B||Y_A|\cos\beta_{AB} & |Y_B||Y_A|\cos\phi_{AB} & |Z_B||Y_A|\cos\delta_{AB} \\ |X_B||Z_A|\cos\alpha_{AB} & |Y_B||Z_A|\cos\psi_{AB} & |Z_B||Z_A|\cos\sigma_{AB} \end{bmatrix} \quad (2.8)$$

Genel Dönüşümler

Önceki bölümde, bir vektörün herhangi bir koordinat sistemine göre konum ve yönelimi incelenmişti. Bu bölümde iki farklı koordinat sisteminin birbirlerine göre konum ve yönelimleri incelenecektir.

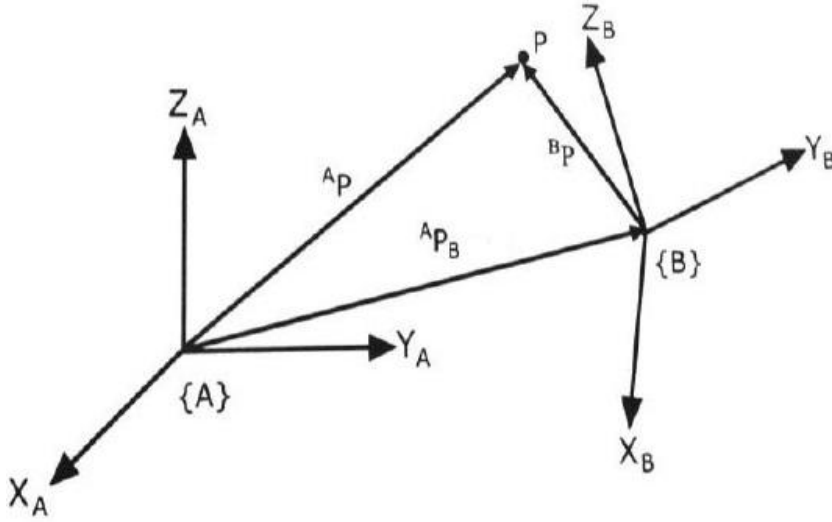
{A} koordinat sisteminin {B} koordinat sistemine uzaklığı denklem 2.2'de ${}^A P_B$ şeklinde bir matris ile gösterilmişti. Şekil 2.19'da gösterilen P noktasının, {A} koordinat sisteminin merkezine uzaklığının, {B} koordinat sistemine göre ifade edilişi denklem 2.9'da gösterilmiştir (Bingül ve Küçük, 2009).



Şekil 2.19 Merkezleri farklı, yönelimleri aynı iki koordinat sistemi

$${}^A P = {}^B P + {}^A P_B \quad (2.9)$$

Bu iki koordinat sisteminin Şekil 2.20'deki gibi hem yönelimleri hem de merkezleri farklı olursa, denklem 2.10'daki halini alır.



Şekil 2.20 Merkezleri ve yönelimleri farklı iki koordinat sistemi

$${}^A P = {}^A R {}^B P + {}^A P_B \quad (2.10)$$

${}^A P$ ile ${}^B P$ arasındaki ilişki tek bir matris ile gösterilmek istenirse, denklem 2.11'deki dönüşüm matrisi elde edilir.

$${}^A P = {}^A T {}^B P \quad (2.11)$$

$$\begin{bmatrix} {}^A P \\ 1 \end{bmatrix} = \begin{bmatrix} {}^A R & {}^A P_B \\ 000 & 1 \end{bmatrix} \begin{bmatrix} {}^B P \\ 1 \end{bmatrix}$$

Burada ${}^A T$, 4x4'lük bir matristir. Bu matrise homojen dönüşüm matrisi adı verilir. İçerisinde 3x3 boyutlu ${}^A R$ dönme matrisi ve 3x1 boyutlu ${}^A P_B$ konum vektörü bulunur. ${}^A T$ matrisi sembolik olarak denklem 2.12'deki gibi yazılır (Bingül ve Küçük, 2009).

$${}^A T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.12)$$

Öteleme İşlemi

Bir öteleme vektörü kullanılarak bir nokta belli bir koordinat sistemine göre tanımlanabilir. Bir P_2 noktasını $\{A\}$ koordinat sistemine göre tanımlamak için ${}^A P_1$ vektörü, ${}^A Q$ vektörü kadar ötelenir (Bingül ve Küçük, 2009).

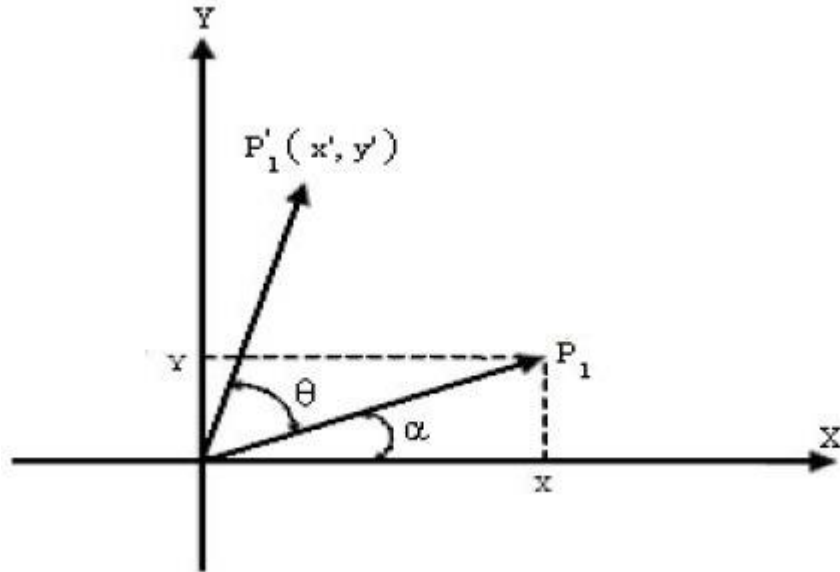
$${}^A P_2 = {}^A P_1 + {}^A Q \quad (2.13)$$

Denklem 2.13'deki ${}^A P_2$ vektörünü, homojen dönüşüm matrisi $D_{Q(q)}$ cinsinden yazarsak denklem 2.14'ü elde ederiz.

$${}^A P_2 = D_{Q(q)} {}^A P_1 = \begin{bmatrix} 1 & 0 & 0 & q_x \\ 0 & 1 & 0 & q_y \\ 0 & 0 & 1 & q_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.14)$$

Dönme İşlemi

Bir koordinat sisteminin herhangi bir eksen etrafında döndürülmesi iki veya üç boyutlu uzayda ifade edilir. İki boyutta dönmeyi ele alalım. İki boyut, aslında bir düzlemi ifade eder ve bu düzlem Şekil 2.21'de görüldüğü gibi sadece (x,y) gibi iki eksenenden oluşur (Bingöl ve Küçük, 2009).



Şekil 2.21 İki boyutlu uzayda dönme

Uzunluğu P olan P_1 vektörünün (x) eksenine yaptığı açı α olsun. Bu durumda P_1 vektörünün (x,y) eksenlerindeki izdüşümü sırayla,

$$x = P \cos \alpha \quad (2.15)$$

$$y = P \sin \alpha \quad (2.16)$$

olur. Şekil 2.19'da görüldüğü gibi bir P_1 vektörü θ kadar döndürülerek P'_1 vektörü elde edilsin. Bu dönme işlemi sonunda P'_1 vektörünün (x,y) eksenlerindeki izdüşümü sırayla,

$$x' = P \cos(\theta + \alpha) \quad (2.17)$$

$$y' = P \sin(\theta + \alpha) \quad (2.18)$$

olur. Bulunan bu trigonometrik ifadeler toplanırsa P'_1 vektörünün (x,y) eksenlerindeki izdüşümü,

$$x' = P(\cos\theta \cos\alpha - \sin\theta \sin\alpha) \quad (2.19)$$

$$y' = P(\sin\theta \cos\alpha + \cos\theta \sin\alpha) \quad (2.20)$$

şeklinde bulunur. Denklem 2.15 ve 2.16 , denklem 2.19 ve 2.20'de yerine konursa,

$$x' = x \cos\theta - y \sin\theta \quad (2.21)$$

$$y' = x \sin\theta + y \cos\theta \quad (2.22)$$

Yapılan bu dönme işleminin matris formunda yazımı denklem 2.23'deki gibidir. İki boyutta gerçekleştirilen dönme işlemi, 2x2 boyutlu bir matris ile gösterilir.

$$R(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \quad (2.23)$$

Öyleyse P'_1 ile P_1 arasındaki ilişki,

$$P'_1 = R(\theta)P_1 = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \cos \theta - y \sin \theta \\ x \sin \theta + y \cos \theta \end{bmatrix} \quad (2.24)$$

olur.

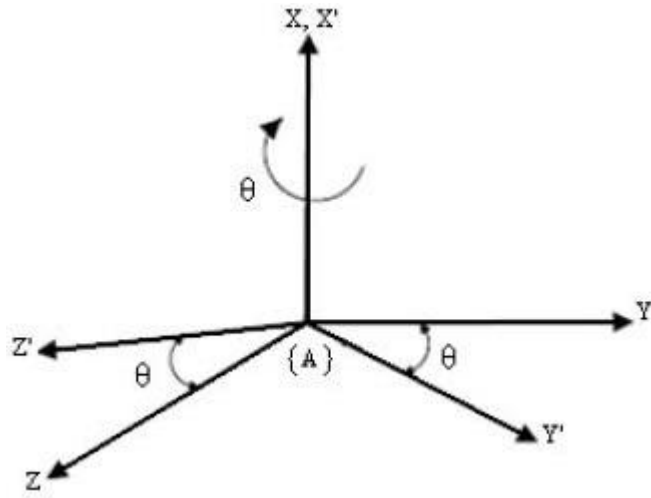
İki boyutta dönme işlemine benzer şekilde, üç boyutlu dönmeyi inceleyelim. 3x3 boyutlu bir R dönme matrisi kullanarak ${}^A P_1$ vektörü, yeni bir vektör olan ${}^A P_2$ olarak değiştirilebilir. Bu işlem denklem 2.25'deki gibi gösterilir (Bingül ve Küçük, 2009).

$${}^A P_2 = R_K(\theta) {}^A P_1 \quad (2.25)$$

Dönme operatörü olan $R_K(\theta)$, K ekseninde θ kadar dönme işlemi gerçekleştirir. Bu operatör, konum vektörü sıfır olan 4x4 boyutlu bir homojen dönüşüm matrisi olarak denklem 2.26'daki gibi gösterilir (Bingül ve Küçük, 2009).

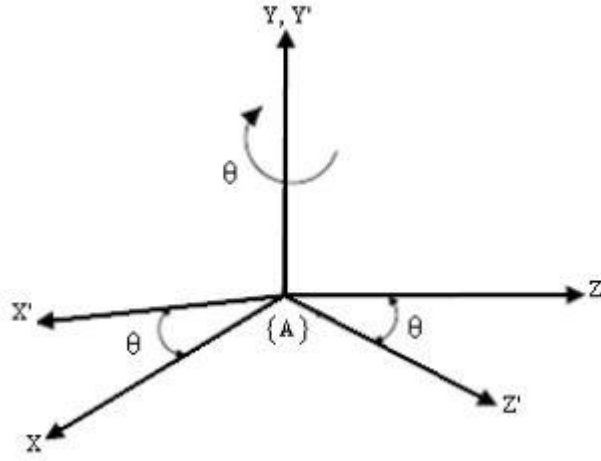
$$R_K(\theta) = \begin{bmatrix} 0 & 0 & 0 & 1 \\ {}^A_B R & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.26)$$

Buna göre x,y ve z eksenlerindeki dönme matrislerini hesaplırsak,



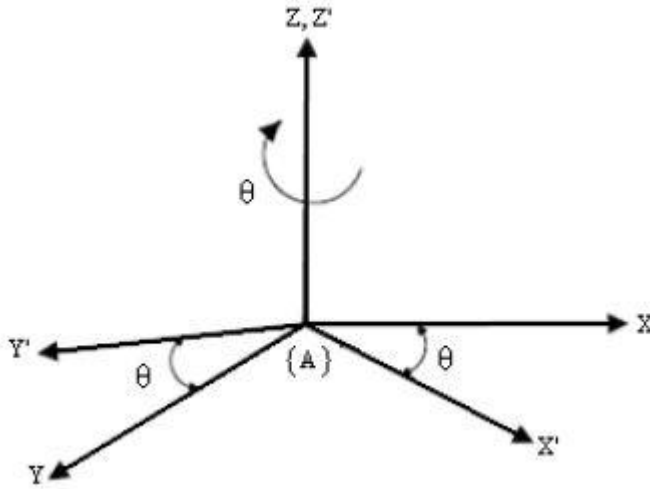
Şekil 2.22 X ekseninin θ açısı kadar döndürülmesi (Erşan, 2009)

$$R_X(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.27)$$



Şekil 2.23 Y ekseninin θ açısı kadar döndürülmesi (Erşan, 2009)

$$R_Y(\theta) = \begin{bmatrix} \cos \theta & 0 & -\sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.28)$$



Şekil 2.24 Z ekseninin θ açısı kadar döndürülmesi (Erşan, 2009)

$$R_Z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.29)$$

2.2.9.2 İleri kinematik analizi

İleri kinematik, bir robot manipülatördeki eklem açılarının ve link uzunluklarının bilinmesi durumunda, robotun uç işlevcisinin alacağı konum ve oryantasyonu hesaplama işlemine denir. Bir seri robot, ana çerçevesinden uç işlevcisine kadar birbirlerine prizmatik veya döner eklemlerle tutturulmuş seri bağlardan oluşur. İleri kinematik analizi sırasında, ana çerçeveden başlanarak tüm eklemlere birer koordinat sistemi eklenir. Komşu iki eklem arasındaki ilişki ${}^{i-1}_iT$ dönüşüm matrisi ile ifade edilir. Bu şekilde tüm sıralı eklemler arasındaki dönüşüm matrisleri bulunur ve bulunan bu matrislerin ardışık olarak çarpılması ile robotun ana çerçevesi ile uç işlevcisi arasındaki konum ve oryantasyon ilişkisi bulunur.

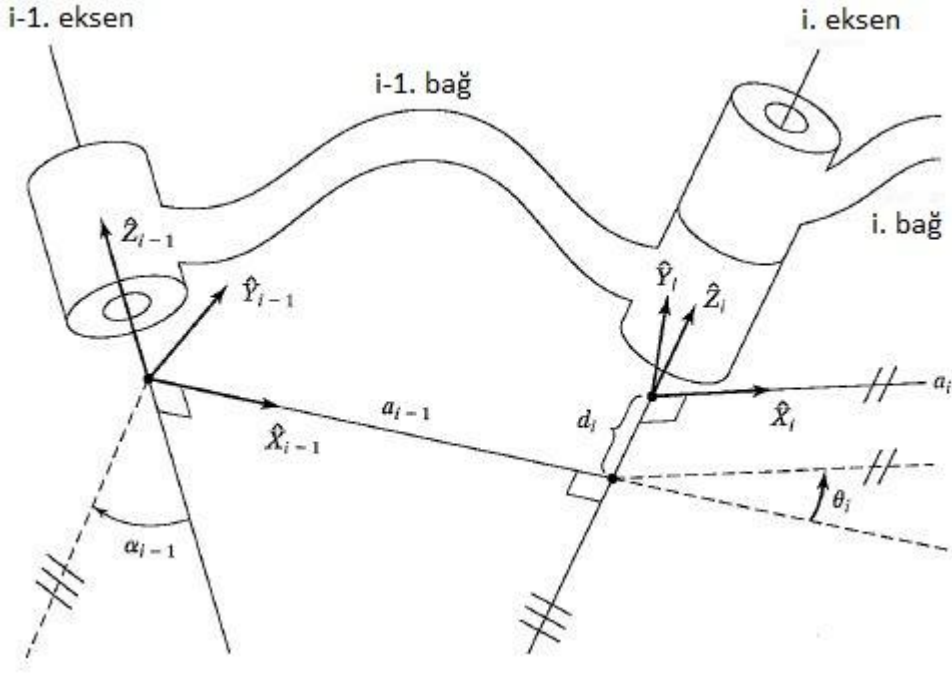
Denavit - Hartenberg Yöntemi

Robotların eklemleri arasındaki ilişkiyi ifade etmek için birçok yöntem geliştirilmiştir. Bu yöntemler arasında en etkili olan ve en sık kullanılan yöntem Denavit - Hartenberg (DH) yöntemidir.

Bu yöntemde, DH değişkenleri olarak adlandırılan dört değişken kullanılarak robot kinematiği çıkarılır. Bu değişkenler şunlardır:

- İki eksen arasındaki bağ uzunluğu (a_{i-1})
- (i-1) ile (i) eksenleri arasındaki açı (α_{i-1})
- Üst üste çakışan bağlar arasındaki eklem kaçıklığı (d_i)
- İki bağ arasında oluşan eklem açısı (θ_i)

Şekil 2.25'de bu değişkenler gösterilmiştir.



Şekil 2.25 DH değişkenleri olan a_{i-1} , α_{i-1} , d_i ve θ_i 'nin belirlenmesi (Craig, 1989)

DH değişkenlerinin bulunabilmesi izlenmesi gereken adımlar şunlardır:

1) Tüm eklemlere koordinat sistemlerinin yerleştirilmesi ve numaralandırılması gerekmektedir. Koordinat sistemleri aşağıdaki sıralamaya ve kurala göre yerleştirilir:

- Eklemlerin dönme ve kayma yönleri belirlenir. Döner eklemler için dönme yönü, prizmatik eklemler için kayma yönü Z eksenini olarak belirlenir.
- Birinci eksenin dönme yönü Z eksenini olarak belirlendikten sonra X eksenini kendi etrafında döndürüldüğünde, birinci Z eksenini ile ikinci Z ekseninin çakışmasını sağlayacak şekilde X eksenini belirlenir.
- Z ve X eksenleri bulunduktan sonra, sağ el kuralına göre Y eksenini bulunur.

2) Koordinat sistemleri yerleştirildikten sonra, sırada DH değişkenlerinin bulunması vardır. Değişkenler, aşağıdaki kurallara göre bulunur.

- a_{i-1} , Z_{i-1} eksenini ile Z_i eksenini arasında, X_{i-1} eksenini boyunca ölçülen uzunluktur.
- α_{i-1} , Z_{i-1} eksenini ile Z_i eksenini arasında, X_{i-1} eksenini boyunca ölçülen açıdır.
- d_i , X_{i-1} eksenini ile X_i eksenini arasında, Z_i eksenini boyunca ölçülen uzunluktur.
- θ_i , X_{i-1} eksenini ile X_i eksenini arasında, Z_i eksenini boyunca ölçülen açıdır.

3) DH değişkenleri bulunduğundan sonra, bu değişkenler kullanılarak DH tablosu oluşturulur. Tablonun toplam satır sayısı, robotun serbestlik derecesini belirtir. Tablodaki α_{i-1} ve a_{i-1} robotun hareketiyle değişmeyen parametrelerdir. d_i parametresi prizmatik eklemlerde, θ_i parametresi ise döner eklemlerde değişken olmaktadır.

Tablo 2.5 DH tablosu

Eksen No	D-H Değişkenleri				i. Ekleme Değişkeni
i	a_{i-1}	α_{i-1}	d_i	θ_i	d_i veya θ_i
1	a_0	α_0	d_1	θ_1	d_1 veya θ_1
2	a_1	α_1	d_2	θ_2	d_2 veya θ_2
3	a_2	α_2	d_3	θ_3	d_3 veya θ_3
4	a_3	α_3	d_4	θ_4	d_4 veya θ_4

4) Oluşturulan DH tablosunun satırları kullanılarak her bir ekleme ait dönüşüm matrisleri oluşturulur. Bu matrislerin oluşturulmasında denklem 2.30'de verilen genel dönüşüm matrisi kullanılır.

$${}^{i-1}T_i = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.30)$$

5) Ekleme sayısı adedinde olan dönüşüm matrislerinin çarpılmasıyla ana çerçeve ile uç işlevci arasındaki konum ve oryantasyon ilişkisini veren ileri kinematik matrisi elde edilir.

$${}^0_N T = {}^0_1 T {}^1_2 T {}^2_3 T {}^3_4 T \dots {}^{N-1}_N T \quad (2.31)$$

Bulunan 4x4 lük bu matrisin dokuz elemanı oryantasyon, üç elemanı ise konum bilgisi içerir.

$${}^{i-1}_i T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.32)$$

↓
Yönelim matrisi
↓
Konum matrisi

2.2.9.3 Ters kinematik analizi

Ters kinematik analizi; kartezyen uzayda, robotun ana çerçevesine göre verilen uç işlevcinin konum ve yönelim verileri yardımıyla eklem değişkenlerinin belirlenmesidir. Yani robotun istenen bir noktaya ulaşabilmesi için uzuv değişkenleri olan d ve θ değerlerinin bulunmasıdır. İleri kinematik analizin tersidir denilebilir.

Ters kinematik problemlerin genel özellikleri

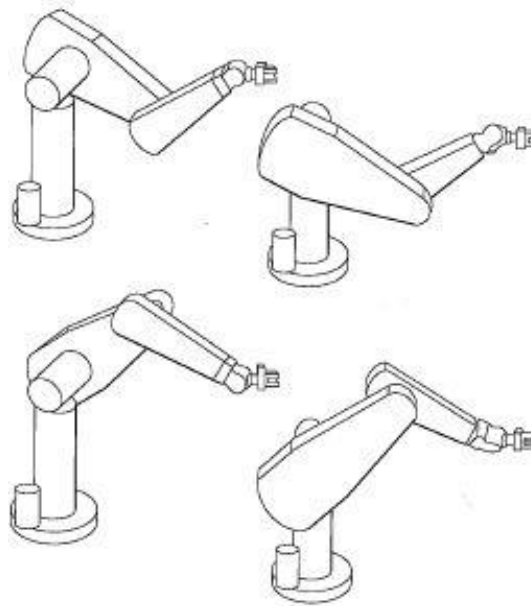
- İleri kinematik problemlerinin çözümü kolaydır ve her zaman çözümü vardır. Ancak ters kinematik problemleri doğrusal olmayan denklemler içerdikleri için her zaman çözümü bulunamamaktadır.
- Analitik olarak ters kinematik problemi çözülemeyen robotlarda nümerik ve iteratif çözümler denir. Ancak tamamen kesin sonuç üreten analitik çözüm bilgisayar ortamında çok hızlı hesaplatılırken, iteratif çözüm çok yavaş hesaplatılmaktadır. Bu yüzden robot tasarımı yapılırken analitik çözümünün mümkün olduğu robotlar tasarlanmalıdır. (Bingül ve Küçük, 2009)
- Problemin zorluğu, robotta bulunan eklemlerin yapısına göre değişmektedir. Prizmatik eklemler, doğrusal denklemler içerdikleri için ters kinematik analizi kolaylaştırırken döner eklemler içerisinde bulundurdıkları doğrusal olmayan denklemler nedeniyle çözümü zorlaştırır ve birden çok çözüm oluşturur. 6R bir robotta bulunacak

ters kinematik çözüm sayısı $26=64$ tür. Bu çözümlerin bir kısmı sanal, bir kısmı gerçek çözüm olabilir.

- Ters kinematik analizi sonucu bulunan her matematiksel çözüm, fiziksel çözümü temsil etmez.
- Ters kinematik problemin çözümünde robotta bulunan bilek yapısı da etkili olmaktadır. Euler bileğinin üç eklemi de bir noktada kesiştiğinden, bu bileğe sahip robotların ters kinematiği tamamen analitik olarak çözülebilir. Ancak eklem kaçıklıklı bileklerde a ve d değişkenleri bulunduğundan, bu bileğe sahip robotların neredeyse tamamı analitik olarak çözülemez. Ayrıca Euler bilek yapısında konum ve yönelim birbirinden ayrı gerçekleşirken eklem kaçıklıklı bilekte konum ve yönelim iç içe geçmiştir. Bu da eklem kaçıklıklı bilekli robotların ters kinematik problemlerinin analitik olarak çözülememesine sebep olmaktadır.

Çözümlerin tekliği

İçerisinde döner eklemlerin bulunduğu robotların ters kinematik problemlerinin çözümünde birden fazla çözüm kümesi oluşabileceğinden bahsetmiştik. Şekil 2.26'da PUMA-560 robotunun aynı noktaya dört farklı şekilde nasıl ulaştığı gösterilmiştir.



Şekil 2.26 PUMA-560 robotu için dört farklı çözüm (Craig, 1989)

Birden fazla çözümlerin olduğu robot türlerinde, çözümlerin ayıklanması ve en doğru çözümün bulunması için bilgisayar algoritmaları geliştirilebilir.

Ters kinematik problemin analitik çözümü

Ters kinematik problemin çözümünde analitik ve sayısal çözüm yöntemleri kullanılmaktadır. Analitik yöntem daha hızlı ve doğru çözümler ürettiği için bu tez çalışmasında analitik yöntem tercih edilmiştir.

Craig (1989) tarafından tanımlanan altı serbestlik derecesine sahip robotun ileri kinematiği denklem 2.33'deki gibi yazılır.

$${}^0T_6 = {}^0T_1 {}^1T_2 {}^2T_3 {}^3T_4 {}^4T_5 {}^5T_6 \quad (2.33)$$

${}^0T_6$ ileri kinematik matrisi, konum ve yönelim verilerini içeren matris elemanları cinsinden denklem 2.34'teki gibi yazılır. Denklem 2.33'teki eşitliğin her iki tarafı, birinci uzvun dönüşüm matrisinin tersi ile çarpılarak denklem 2.35'deki eşitlik sağlanır. Denklem 2.34'teki matrisin 1. kolonu uç işlevcinin normal vektörünü $(n = [n_x \ n_y \ n_z]^T)$, 2. kolonu kayma (sliding) vektörünü $(s = [s_x \ s_y \ s_z]^T)$ ve 3. kolonu ise yaklaşım (approaching) vektörünü $(a = [a_x \ a_y \ a_z]^T)$ göstermektedir (Bingül ve Küçük, 2009).

$${}^0T_6 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.34)$$

$$[{}^0T_1]^{-1} {}^0T_6 = {}^1T_2 {}^2T_3 {}^3T_4 {}^4T_5 {}^5T_6 \quad (2.35)$$

Ters kinematik problem çözümü, denklem 2.35'teki eşitliğin sağ ve sol tarafında yer alan karşılıklı matris elemanlarının birbirine eşitlenmesiyle yapılır. Eşitliğin iki tarafında da 12 adet denklem bulunur ve bu denklemlerden basit ifadeli olanları kullanılarak çözüm yapılır. Denklem 2.35'te sadece bir uzvun tersi, eşitliğin iki tarafı ile çarpılmıştı. Oluşan bu denklem ile çözüm bulunamazsa, diğer uzuvların da tersleri eklenerek çözümler denenir. Kullanılabilecek diğer denklemler aşağıda verilmiştir:

$$\begin{bmatrix} {}^0T & {}^1T \end{bmatrix}^{-1} {}^0T = {}^2T {}^3T {}^4T {}^5T \quad (2.36)$$

$$\begin{bmatrix} {}^0T & {}^1T & {}^2T \end{bmatrix}^{-1} {}^0T = {}^3T {}^4T {}^5T \quad (2.37)$$

$$\begin{bmatrix} {}^0T & {}^1T & {}^2T & {}^3T \end{bmatrix}^{-1} {}^0T = {}^4T {}^5T \quad (2.38)$$

$$\begin{bmatrix} {}^0T & {}^1T & {}^2T & {}^3T & {}^4T \end{bmatrix}^{-1} {}^0T = {}^5T \quad (2.39)$$

Ters kinematik çözüm gerçekleştirilirken bazı trigonometrik eşitliklerden yararlanılır. Bu eşitlikler aşağıda verilmiştir (Bingöl ve Küçük, 2009).

$$1. \quad \cos \theta = a \quad \text{ise} \quad \theta = \text{Atan} 2(\pm \sqrt{1-a^2}, a) \quad (2.40)$$

$$2. \quad \sin \theta = a \quad \text{ise} \quad \theta = \text{Atan} 2(a, \pm \sqrt{1-a^2}) \quad (2.41)$$

$$3. \quad \cos \theta = a \quad \text{ve} \quad \sin \theta = b \quad \text{ise} \quad \theta = \text{Atan} 2(b, a) \quad (2.42)$$

$$4. \quad a \sin \theta + b \cos \theta = 0 \quad \text{ise} \\ \theta = \text{Atan} 2(-b, a) \quad \text{veya} \quad \theta = \text{Atan} 2(b, -a) \quad (2.43)$$

$$5. \quad a \sin \theta + b \cos \theta = c \quad \text{ise} \\ \theta = \text{Atan} 2(a, b) + \text{Atan} 2(\pm \sqrt{a^2 + b^2 - c^2}, c) \quad (2.44)$$

$$6. \quad a \cos \theta_i + b \cos \theta_j = c \quad \text{ve} \\ a \sin \theta_i + b \sin \theta_j = d \quad \text{ise}, \\ \theta_i = \text{Atan} 2(d, c) + \text{Atan} 2(\pm \sqrt{c^2 + d^2 - s^2}, s) \\ \theta_j = \theta_i + \text{Atan} 2(\pm \sqrt{4a^2b^2 - t^2}, t) \quad (2.45)$$

Denklemden kullanılan kısaltmalar aşağıda verilmiştir.

$$s = \frac{a^2 - b^2 + c^2 + d^2}{2} \quad \text{ve} \quad t = c^2 + d^2 - a^2 - b^2$$

$$7. \quad a \sin \theta + b \cos \theta = c \quad \text{ve} \\ a \cos \theta - b \sin \theta = d \quad \text{ise}, \\ \theta = \text{Atan} 2(ac - bd, ad + bc) \quad (2.46)$$

$$8. \quad a \cos(\theta_i + \theta_j) + b \cos \theta_i = c \quad \text{ve} \\ a \sin(\theta_i + \theta_j) + b \sin \theta_i = d \quad \text{ise}, \\ \theta_j = \text{Atan} 2(\sin \theta_j, \cos \theta_j) \\ \theta_i = \text{Atan} 2(rd - sc, rc + sd) \quad (2.47)$$

Denklemden kullanılan kısaltmalar aşağıda verilmiştir.

$$\cos \theta_j = \frac{c^2 + d^2 - a^2 - b^2}{2ab}, \quad \sin \theta_j = \sqrt{1 - \cos^2 \theta_j},$$

$$r = a \cos \theta_j + b \quad \text{ve} \quad s = a \sin \theta_j$$

$$\begin{aligned} 9. \quad & a \sin(\theta_i + \theta_j) + b \cos(\theta_i + \theta_j) + c \cos \theta_i = d \quad \text{ve} \\ & b \sin(\theta_i + \theta_j) - a \cos(\theta_i + \theta_j) + c \sin \theta_i = e \quad \text{ise} \\ & \theta_j = \text{Atan} 2(a, b) + \text{Atan} 2(\pm \sqrt{c^2 + d^2 - r^2}, r) \\ & \theta_i = \text{Atan} 2(sd + te, -se + td) \end{aligned} \quad (2.48)$$

Denklemden kullanılan kısaltmalar aşağıda verilmiştir.

$$s = a \cos \theta_j - b \sin \theta_j, \quad t = b \cos \theta_j + a \sin \theta_j + c \quad \text{ve}$$

$$r = \frac{d^2 + e^2 - a^2 - b^2 - c^2}{2c}$$

$$\begin{aligned} 10. \quad & a \cos \theta_i \cos(\theta_j + \theta_k) + b \cos \theta_i \cos \theta_j + c \sin \theta_i = d \quad \text{ve} \\ & a \sin \theta_i \cos(\theta_j + \theta_k) + b \sin \theta_i \cos \theta_j + c \cos \theta_i = e \quad \text{ise,} \\ & \theta_i = \text{Atan} 2(\pm re - cd, \pm rd + ce) \end{aligned} \quad (2.49)$$

Denklemden kullanılan kısaltma aşağıda verilmiştir.

$$r = \sqrt{d^2 + e^2 - c^2}$$

$$\begin{aligned} 11. \quad & a \cos \theta_i \cos \theta_j \sin \theta_k - a \sin \theta_i \cos \theta_k - b \sin \theta_i = c, \\ & a \sin \theta_i \cos \theta_j \sin \theta_k + a \cos \theta_i \cos \theta_k + b \cos \theta_i = d \quad \text{ve} \\ & a \sin \theta_j \sin \theta_k = e \quad \text{ise,} \\ & \theta_k = \text{Atan} 2(\sqrt{1 - \cos^2 \theta_k}, \cos \theta_k) \\ & \theta_j = \text{Atan} 2\left(\frac{e}{a}, \pm \sqrt{c^2 + d^2 - b^2 - a^2 \cos^2 \theta_k - 2ab \cos \theta_k}\right) \\ & \theta_i = \text{Atan} 2(ad \cos \theta_j \sin \theta_k - [b + a \cos \theta_k]d, \\ & ac \cos \theta_j \sin \theta_k + [b + a \cos \theta_k]d) \end{aligned} \quad (2.50)$$

Denklemlerde kullanılan kısaltma aşağıda verilmiştir.

$$\cos \theta_k = \frac{c^2 + d^2 + e^2 - a^2 - b^2}{2ab}$$

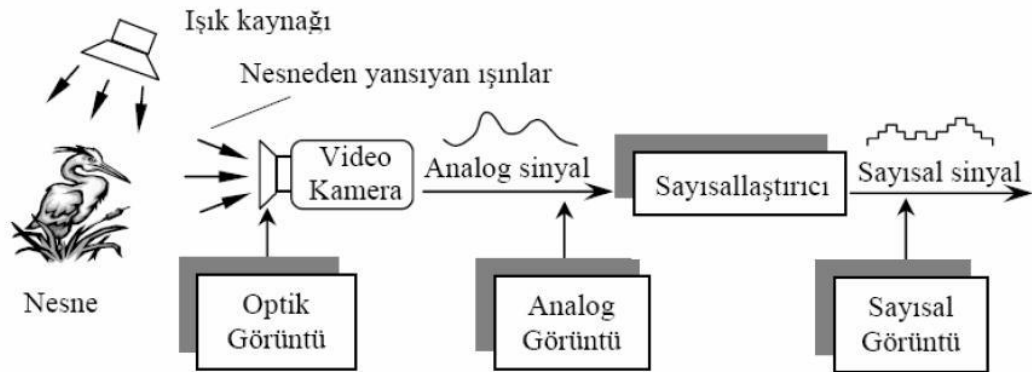
2.3 Görüntü İşleme ve Esasları

Görüntü işleme; kameralar vasıtasıyla alınan görüntülerin, elektronik ortamda bilgisayar ve çeşitli yazılımlar yardımıyla gereksinimlere uygun şekilde özelliklerinin ve yapılarının değiştirilmesi, geliştirilmesi ve analizlerinin yapılması işlemidir. Günümüz teknolojisi sayesinde, bir görüntünün girdi olarak kullanılıp istenilen özellikte bir başka görüntünün elde edilmesi mümkündür. Görüntü işleme tekniği ile dijital görüntüde bulunan bir cismin boyutu, orta noktası, ağırlık merkezi, rengi, şekli, parlaklığı, konumu vb. özellikleri tespit edilebilir, görüntüdeki gürültüler yok edilip görüntü kalitesi artırılabilir, görüntüde bulunan nesnelerden hareketli ve hareketsiz olanları ayırt edilebilir. Bu sayılanlardan başka birçok farklı amaç için görüntü işleme teknikleri kullanılmaktadır.

Görüntü işlemenin başlıca kullanım alanlarına savunma sanayisi, güvenlik sistemleri, kalite kontrol sistemleri sayılabilir. Güvenlik sistemlerine örnek olarak trafik denetimi sistemleri (araç takibi, plaka tanıma, hız takibi) gösterilebilir.

2.3.1 Görüntü işleme adımları

Şekil 2.27'de gerçek yaşamdan alınan analog görüntülerin dijital görüntüye dönüşüm adımları gösterilmiştir.



Şekil 2.27 Görüntünün yakalanması ve sayısallaştırılması aşamaları (Erşan, 2009)

Görüntü işlemede ilk adım, bir ışık kaynağı ile aydınlatılmış olan nesneden yansıyan ışınların kameraya aktarılmasıdır. Nesnenin tanımlanmasını sağlayan bu ışınlar, kamerada elektrik sinyallerine çevrilir. Böylece optik görüntü, analog

görüntüye çevrilmiş olur. Analog haldeki görüntü bir analog/dijital dönüştürücü yardımıyla dijital sinyallere dönüştürülür.

Dijital görüntünün elde edilmesinden sonraki adım, ön-işleme adıdır. Ön-işleme adımı, alınan görüntünün ileriki aşamalarda daha kolay ve başarılı bir şekilde işlenebilmesi için bazı küçük çaplı işlemlerden geçirilmesidir. Görüntüdeki gürültülerin yok edilmesi, görüntünün kontrast ayarının yapılması gibi işlemler ön-işleme adımına örnek olarak verilebilir.

Ön-işleme adımı bittikten sonra sıradaki adım, segmentasyon (parçalara ayırma) adıdır. Segmentasyon, bir görüntüyü her biri içerisinde farklı özelliklerin bulunduğu anlamlı bölgelere ayırma işlemidir. Örnek olarak; havadan çekilen bir fotoğrafta, yolda hareket eden araçları ve çevreyi yoldan ayırt etmek için segmentasyon yöntemi kullanılır. Segmentasyon yöntemleri, ayrıştırılmak istenilen özelliğe ve kullanılan görüntüye göre farklılık göstermektedir. Dolayısıyla tüm görüntülere uygulanabilecek genel bir segmentasyon yöntemi yoktur. Segmentasyon görüntü işlemenin en zor uygulamalarından biridir.

Segmentasyondan sonraki adım, görüntüde ilgilenilen kısımların ön plana çıkarılması işlemidir. Segmentasyon ile ayrıştırılan bölümlerden kullanılacak olan bölüm ön plana alınır. Son olarak , tanıma ve yorumlama adımı gelir. Bu aşamada, resmin içerisindeki nesnelerin veya bölgelerin önceden belirlenen tanımlamalara göre etiketlendirilmesi yapılır (Peker, 2009).

2.3.2 Görüntü işlemede kullanılan terimler

Piksel: Bir görüntünün temel bileşenidir. Piksel sayısı ile görüntüdeki ayrıntı sayısı doğru orantılıdır. Dijital görüntü, $M \times N$ boyutlu bir matristen meydana gelir. Denklem 2.51'de bu matrisin yazım formu gösterilmiştir. Matrisin her bir elemanı bir pikseli temsil etmektedir ve bu matris görüntünün çözünürlük değerini ifade eder. Görüntü kalitesinin artırılması için çözünürlüğün yani piksel sayısının artırılması gerekir.

$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0, N-1) \\ f(1,0) & f(1,1) & \dots & f(1, N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1, N-1) \end{bmatrix} \quad (2.51)$$

İkili görüntü: Görüntünün sayısallaştırılması sırasında tüm renkler göz ardı edilerek sadece siyah ve beyazın göz önünde bulundurulduğu yöntemdir. Görüntüdeki beyaz kısımlar aynen kalırken, beyaz ile siyah rengin arasında kalan tüm renkler siyah olarak atanır. Görüntüde sadece siyah ve beyaz renk olduğu için her piksel 1 bitten oluşur. Şekil 2.28'de ikili görüntü örneği gösterilmiştir.



Şekil 2.28 İkili görüntü (Wilson, 2000)

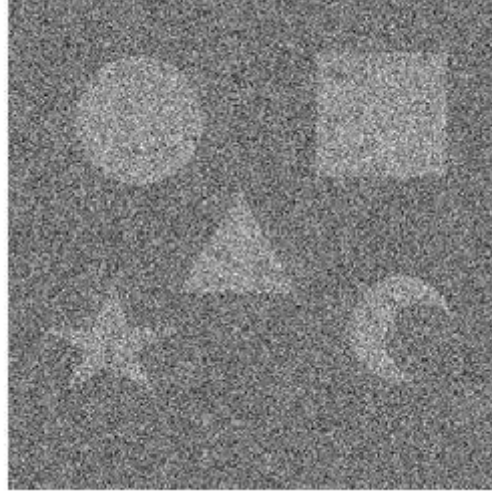
Gri düzey görüntü: İkili görüntüye ek olarak grinin tonlarının da kullanıldığı yöntemdir. 256 gri değer bir byte olarak tanımlanır (1 byte = 8 bit ve $2^8=256$). Tonlar 0'dan 255'e kadar olan kodlarla ifade edilir. 0 değeri siyahı, 255 değeri ise beyazı temsil eder. Arada kalan kodlar, diğer tonları ifade eder. Bunun tam tersi şekilde de seçim yapılabilir. Şekil 2.29'da Şekil 2.28'deki ikili görüntünün gri düzey hali gösterilmiştir.

Renkli görüntü: Bu yöntemde görüntüleme RGB (Kırmızı-Yeşil-Mavi) kodlanmış aynı nesneye ait üç adet gri düzey görüntünün bu sırayla ve üst üste ekrana gönderilmesi ile meydana gelir.

Ölçeklendirme: Görüntünün boyutu, kullanım ihtiyacına göre büyütülüp küçültülebilir. Bu işleme ölçeklendirme adı verilir.

Döndürme: Görüntüler, kullanım ihtiyacına göre istenilen bir açı miktarı kadar çevrilebilir. Bu işleme döndürme adı verilir.

Aynalama: Görüntüler, aynadaki görüntüye benzer şekilde ters çevrilebilir. Buna aynalama denir.



Şekil 2.29 Gri düzey görüntü (Wilson, 2000)

Renk düzeltmesi: Görüntünün ışık ve renk tonları ihtiyaca göre değiştirilebilir.

İşaretleme: Görüntünün belirli bir bölümünün, bir işlem için hedef gösterilmesine işaretleme denir.

Katman: Görüntü işleme yazılımlarında özellikle montaj çalışmalarında kullanılan, alttaki nesnenin görünmesini engellemeyen, saydam ve sanal yüzeylere katman denir.

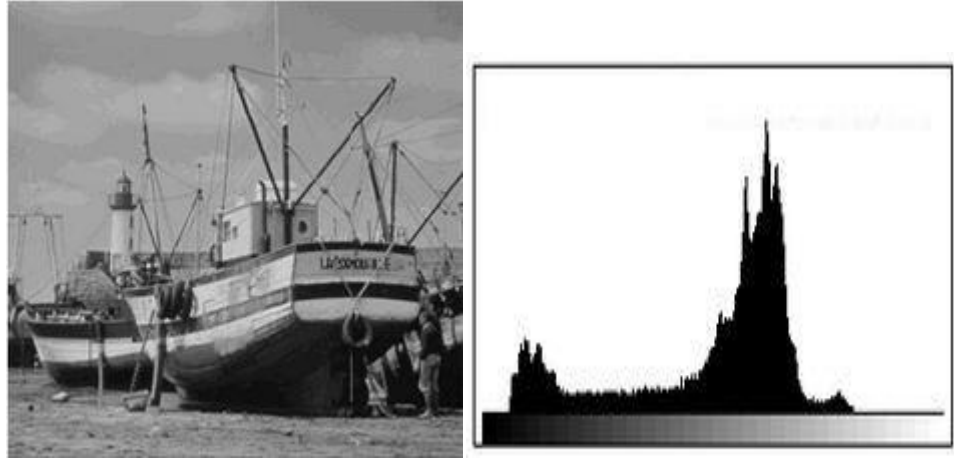
Maskeleye: Bazı bölümleri, çalışmanın dışında tutma işlemidir.

2.3.3 Görüntü işlemede kullanılan teknikler

Dijital görüntünün elde edilmesinden sonra, bu görüntünün işlenerek ihtiyaç duyulan hale getirilmesi adımına geçilir. Görüntü işleme uygulamalarında en sık kullanılan teknikler aşağıda verilmiştir.

Histogram: Görüntüdeki piksel değerlerinin ağırlığını belirten grafiksel bir gösterimdir. Görüntü iyileştirme, görüntü tamiri, görüntü segmentasyonu ve görüntü sıkıştırma gibi işlemlerin temelini oluşturur.

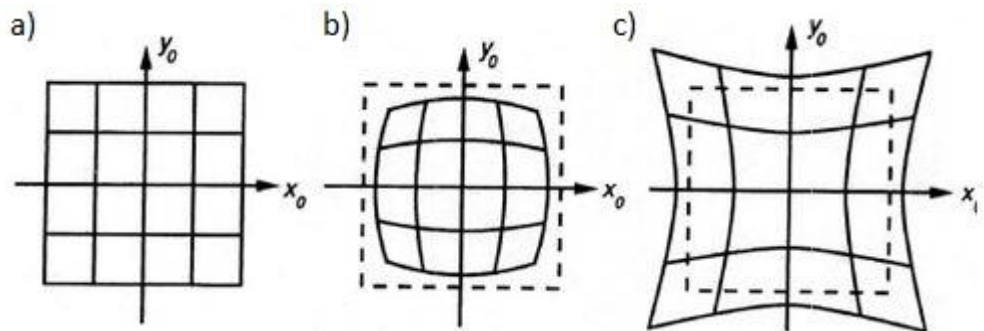
Şekil 2.30'da bir görüntü ve ona ait histogram grafiği gösterilmiştir.



Şekil 2.30 a) Gri düzey bir görüntü b) Bu görüntüye ait histogram

Şekil 2.30b'de x eksenini Şekil 2.30a'daki görüntünün parlaklık değerini, y eksenini ise bu parlaklık değerlerinin görüntüde tekrar etme sıklığını temsil etmektedir.

Görüntü tamiri: Görüntüleme sistemlerinde, sistem elemanlarının fiziksel hatalarından ve yetersizliklerinden dolayı oluşan görüntü bozulmaları vardır. Çok mercekli objektiflerde; ağırlık, malzeme kalitesi ve boyut gibi fiziksel durumların sonucu olarak oluşurlar ve bunlara açısal bozulma denir. Dışbükey ve içbükey bozulma olmak üzere ikiye ayrılır. Mercek sistemindeki elemanların görüntü eksenlerinin üst üste oturmamasından dolayı oluşan bozulmalara da teğetsel bozulma denir. Teğetsel bozulma, görüntüde geometrik hatalara neden olur. Teğetsel ve açısal bozulmaların düzeltilmesi için görüntü tamiri tekniği kullanılır. Şekil 2.31'de bir görüntünün bozulmamış ilk hali ve açısal bozulmalara uğramış halleri gösterilmiştir.



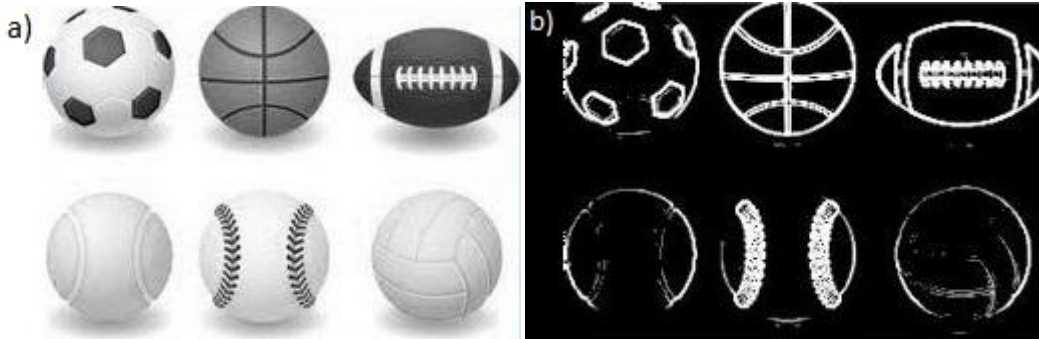
Şekil 2.31 Açısal bozulmalar a) bozulma yok b) dış bükey bozulma c) iç bükey bozulma

Görüntü iyileştirme: Görüntünün anlaşılabilirliğinin artırılması amacıyla yapılır. Görüntüdeki sağlam bölümler kullanılarak bozuk kısımların düzeltilmesi işlemidir. Bu işlemin gerçekleştirilebilmesi için çeşitli filtreler kullanılır. Görüntüdeki bulanıklık giderilmesi, zıtlıkların belirginleştirilmesi, kontrast germe gibi işlemler görüntü iyileştirme yöntemlerine örnek verilebilir.

Görüntü sıkıştırma: Bir görüntünün, içeriğinde kayba sebep olmadan kapladığı hafızanın azaltılması işlemidir.

Kontrast iyileştirme: Görüntünün parlaklık ve kontrast değerlerinin ihtiyaca göre ayarlanmasıdır.

Görüntü filtreleme: Filtreleme, girdi görüntünün işlenerek, resmin özelliklerinin değiştirilmesi sonucu yeni bir resmin oluşturulmasıdır. Görüntülerdeki hataların çoğu filtreleme yöntemi ile yok edilir, çeşitli efektler verilebilir. Örneğin kenar keskinleştirme, görüntü yumuşatma gibi efektler verilebilir. Çok kullanılan filtreleme yöntemlerinden bazıları şunlardır: Sobel filtresi, Prewitt filtresi, Roberts filtresi. Şekil 2.32'de bir görüntü ve ona ait Sobel filtresi uygulanmış hali gösterilmiştir.

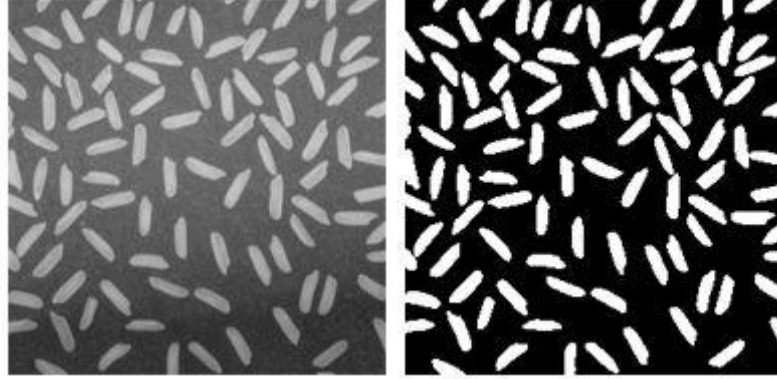


Şekil 2.32 Sobel filtresi örneği a) Filtrelemeden önce b) Filtrelemeden sonra

Bölümleme: Görüntüde istenen şekillerin ele alınıp incelenmesi ve sınıflandırılması için yapılan bir gruplandırma işlemidir. Eşikleme ve etiketleme gibi bölümleme teknikleri vardır. Görüntüde ayrıştırma, arka planın temizlenmesi ve görüntüdeki farklı nesnelerin tespiti için kullanılır.

Eşikleme: Bu yöntemde öncelikle 0-255 arası bir değer, eşik değeri olarak seçilir. Daha sonra gri düzey görüntünün tüm pikselleri bu eşik değeri ile karşılaştırılır. Eşik değerinin altında kalan pikseller 0, eşik değerinin üzerinde kalan pikseller ise 1 ile kodlanarak ikili kodlu bir görüntü elde edilmiş olur. Elde

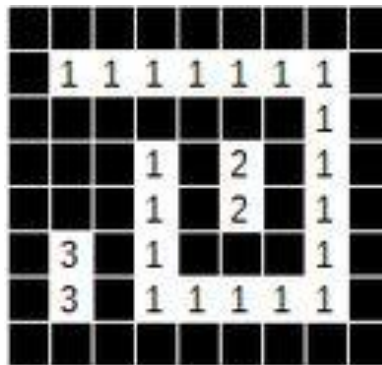
edilen bu görüntüde siyah kısımlar zemini, beyaz kısımlar ise nesneyi temsil etmektedir. Şekil 2.33'de bir görüntünün eşikleme işleminden önce ve sonraki durumu gösterilmiştir.



Şekil 2.33 Eşikleme işlemi a) Eşiklemeden önce b) Eşiklemeden sonra

Etiketleme: İkili kodlu bir görüntünün içerisinde birbiriyle hiçbir komşuluğu olmayan nesnelerin değişik renklere boyanarak birbirlerinden ayrılması işlemine etiketleme denir (Jain et al, 1995).

Etiketleme işlemi, ikili resmin (0,0) noktasından başlayarak piksel piksel taranması ile yapılır. İkili resim içinde nesneler beyaz, zemin siyah renktedir. Tarama sırasında beyaz piksele rastlandığında bu pikselin 3x3 matris şeklindeki piksel komşularına bakılır. Bu piksel komşuları arasında önceden etiketlenmiş bir piksele rastlanırsa beyaz piksele de aynı etiket değeri verilir. Beyaz pikselin komşulukları arasında önceden etiketlenmiş bir piksel değeri yoksa, son etiketleme değeri bir artırılarak beyaz piksele verilir ve etiketleme tüm resme uygulanır (Şekil 2.34). Etiketleme işlemi sonucunda resimdeki nesneler farklı renge boyanır ve resimdeki renkler sayıldığında nesne sayısı bulunur (Kert, 2006).



Şekil 2.34 Etiketleme işlemi

2.3.4 Renk modelleri

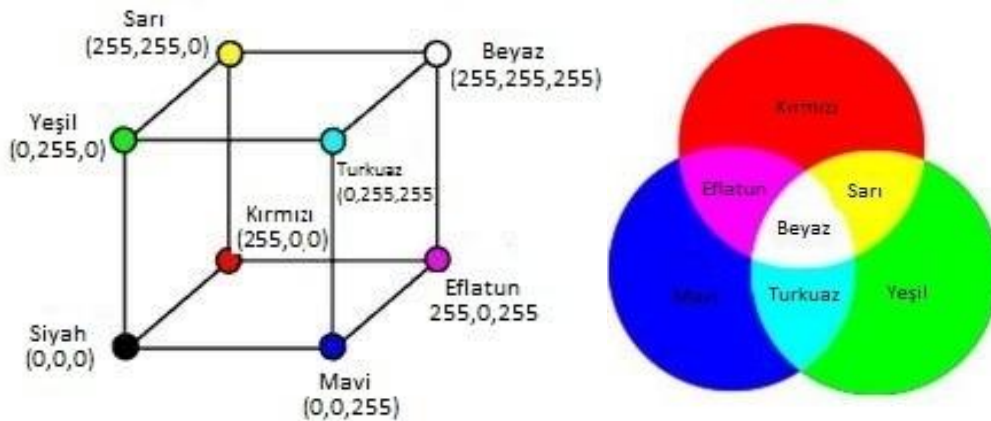
Literatürde kullanılan renk modellerinden bazıları şunlardır;

- RGB renk modeli
- CMYK renk modeli
- HSV renk modeli
- YIQ renk modeli
- YUV renk modeli
- 11_12_13 renk modeli
- L*a*b renk modeli

Bu modellerden en çok kullanılanları aşağıda açıklanmıştır.

2.3.4.1 RGB renk modeli

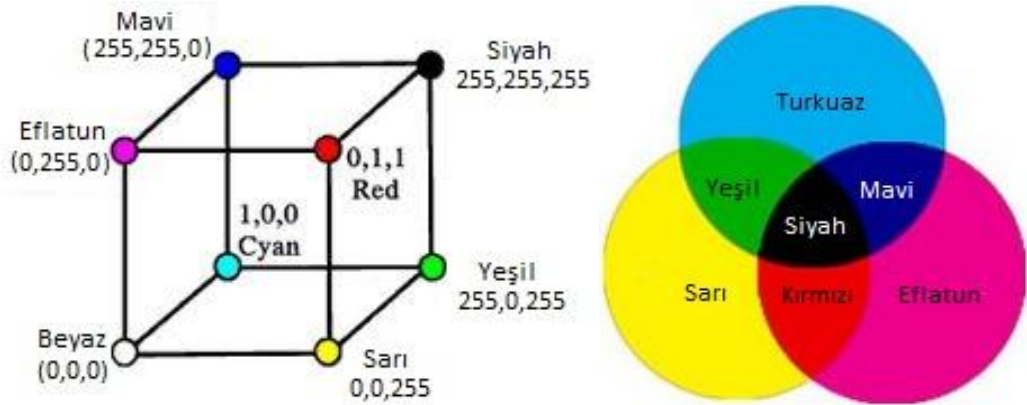
İsmi 'Red-Green-Blue' (Kırmızı-Yeşil-Mavi) kelimelerinin baş harflerinden alan bu renk uzayı, en çok kullanılan uzaydır. RGB, toplamsal bir metottur. Yani kırmızı, yeşil ve mavinin dışındaki diğer tüm renkler bu üç rengin belirli oranlarda karıştırılması ile elde edilir. Beyaz, bu üç rengin hepsini içerirken siyah ise hiçbirini içermemektedir. RGB renk modelinde arka plan siyah olarak kabul edilir. RGB renk modeli, internet ve sayısal ortamlarda yapılan çalışmalarda kullanılmasına karşın baskı ortamında kullanılmamaktadır. Bunun sebebi, görüntüyü elde etmekte faydalanan cihazlara bağımlı olmasıdır. Şekil 2.35'de RGB renk modelinin ifade edilişi gösterilmiştir.



Şekil 2.35 RGB renk modeli

2.3.4.2 CMYK renk modeli

İsmi 'Cyan Magenta Yellow Key' (Turkuaz Eflatun Sarı Siyah) kelimelerinin baş harflerinden alır. Başlangıçta kullanılan renk sayısı C, M, Y olmak üzere 3'tü. Siyah bu renklere sonradan eklenmiştir. Bu üç renk teoride %100 oranlarında karışarak siyahı verirken, pratikte tam olarak siyah rengi vermemesinden ve bu üç rengin karışımının meydana getirdiği yüksek maliyetten dolayı sonradan eklenmiştir. Bu renk modelinde RGB renk modelinin aksine, diğer renkleri elde etmek için eksiltme işlemi uygulanır. Üst üste binen boya, siyah renge yaklaşır. Arka plan veya kağıt beyazdır. Diğer renklerin elde edilmesinde, hangi renk için hangi ana renklerin emilmesi veya yansıtılması gerektiği Tablo 2.6'da gösterilmiştir. Bu işlem için renklere yansıtıcı olmayan bazı pigmentler eklenerek o rengin görülmemesi sağlanır. Bu renk modeli genellikle yazıcılarda, matbaalarda ve yüksek seviyeli baskı gerektiren alanlarda kullanılır. Şekil 2.36'de CMYK renk modelinin ifade edilişi gösterilmiştir.



Şekil 2.36 CMYK renk modeli

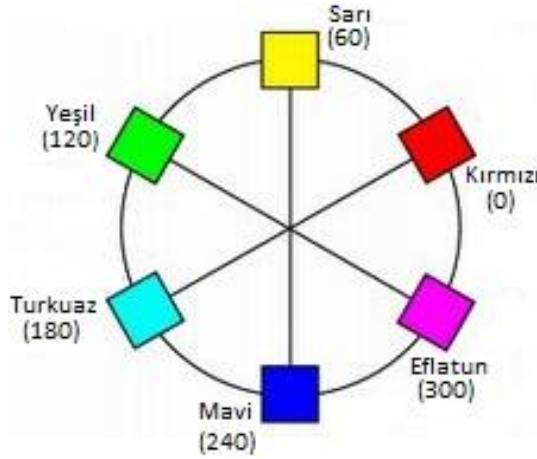
Tablo 2.6 CMYK renk modelinde diğer renklerin elde edilmesi (Peker, 2009)

Ana Renk	Emilme	Yansıtma
Turkuaz	Kırmızı	Mavi ve Yeşil
Eflatun	Yeşil	Mavi ve Kırmızı
Sarı	Mavi	Kırmızı ve Yeşil
Siyah	Hepsi	Hiçbiri

2.3.4.3 HSV renk modeli

İsmi 'Hue, Saturation, Value' kelimelerinin baş harflerinden alır. HSV renk uzayı renkleri sırasıyla renk özü, doygunluk ve parlaklık olarak üç kısımda

inceler. Renk özü, rengi belirler. Silindirik gösterimde açısal bir değerdir ve bu nedenle 0-360 arası değer alır. Şekil 2.37'de renk özünün silindirik gösterimi yapılmıştır. Doygunluk, rengin canlılık miktarını belirten parametredir. Yüksek doygunluk değerleri canlı renkler ortaya koyarken, düşük doygunluk değerleri rengin griye dönmesine sebep olur. Parlaklık, görüntünün içindeki beyaz rengin oranını temsil eder.



Şekil 2.37 Renk özü (Hue) değerinin silindirik gösterimi

HSV renk modeli RGB renk modeline göre insan görüşüne daha yakın yapıdadır. HSV renk modeli, RGB renk modelinin silindirik koordinat sisteminde gösterimi olarak da tanımlanabilir. HSV renk modeli, aygıta bağımlıdır bu nedenle HSV modelinde tanımlı bir renk, rengi üreten aygıta göre farklılık gösterebilir.

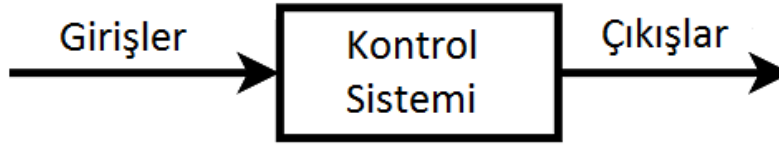
HSV renk modeli, ilk tanımlandığında konik bir biçimdeydi. Ancak sonradan, gerçek zamanlı koordinat denetimi için o zamanın teknolojisi yeterli gelmediğinden silindir biçime çevrildi. Ayrıca; konik biçimde, aydınlık düzeyi azaldıkça koninin genişliği azalırken silindir biçimde sıfır aydınlık düzeyinde bile yüksek doygunluk düzeyleri tanımlanabilir, ve bunlar geçersiz renkler olabilir. Bu nedenlerden dolayı görüntü işlemede konik biçim tercih edilir. Şekil 2.38'de HSV renk modelinin konik biçimi gösterilmiştir (Özcan, 2010).

2.4 Kontrol Sistemleri

2.4.1 Kontrol nedir?

Bir sistemin çıkışını istenen değere ulaştırmak, ya da sistemin önceden mevcut bir durumunun devamlılığını sağlayabilmek için sistem üzerinde yapılan işlemlerin tümüne kontrol denir. Bir sistemin genel olarak çıkışlarını, bozucu etkilere rağmen istenen değerlere ulaştırmak için gerekli kontrol işlemlerini gerçekleştirmek üzere kurulan sistemlere kontrol sistemleri denir. Bir kontrol sisteminin amacı, giriş bilgilerini kullanarak, çıkışları istediğimiz değere ulaştırmak ve bu değerde sabit tutmaya çalışmaktır (Çelikel, 2006).

Şekil 2.39'da kontrol sistemlerinin temel yapısı gösterilmiştir.



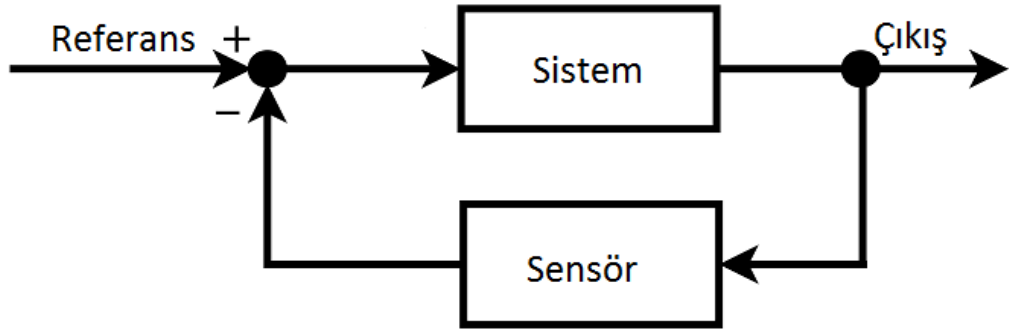
Şekil 2.39 Kontrol sistemlerinin temel gösterimi

2.4.2 Kontrol sistemi çeşitleri

Kontrol teorisinde sıkça kullanılan terimlerden bazıları aşağıda açıklanmıştır.

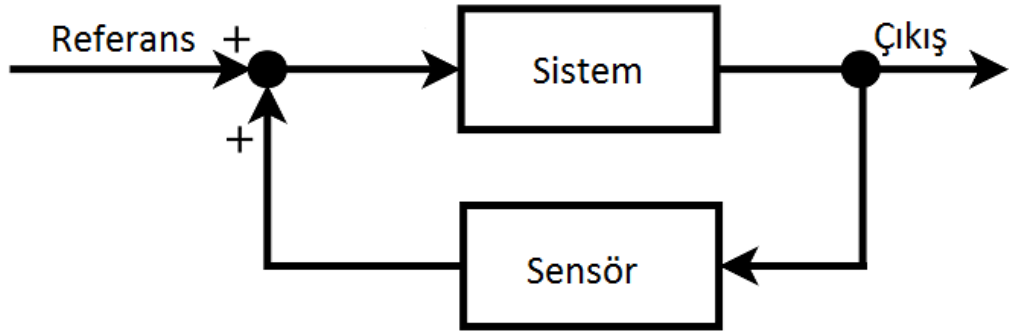
Geri besleme: Geri besleme, algılayıcılardan alınan verilerin kullanılarak bir sonraki hareketlerin düzenlenmesi işlemidir. Diğer bir ifadeyle, bir sistemin çıkışını istenen duruma ulaştırmak ve o halde kararlı bir şekilde kalabilmesini sağlamak için, sistemin o andaki durumu ile istenen durumun sürekli bir şekilde karşılaştırılması işlemidir. Bir sürecin bir basamağındaki değişimin, önceki bir basamağı da etkilemesidir denilebilir. İki çeşit geri besleme tekniği vardır.

Negatif geri besleme: Sistemin çıkışı ile ulaşılmak istenen referans değeri arasındaki farkın yani hatanın, sisteme giriş olarak verildiği kontrol yöntemidir. Giriş değerine göre hareket ayarlanmış olur. Şekil 2.40'da negatif geri beslemeli bir sistem gösterilmiştir.



Şekil 2.40 Negatif geri beslemeli bir sistem

Pozitif geri besleme: Sistemin çıkışı ile ulaşılması istenen referans değerinin toplamının, sisteme giriş olarak verildiği kontrol yöntemidir. Giriş değerine göre hareket fazlaşmış olur. Şekil 2.41'de pozitif geri beslemeli bir sistem gösterilmiştir.



Şekil 2.41 Pozitif geri beslemeli bir sistem

Gecikme zamanı (t_g): Çıkışın, referans değerinin yarısına ulaşana kadar geçen süredir. 1. derece sistemler için t_g , zaman sabitine eşittir.

Yükselme Zamanı (t_y), (t_r): Çıkışın, referansın %10-90 , %5-95 veya %0-100 değerine ulaşana kadar geçen süredir. Genelde aşırı sönümlü (1. derece) sistemlerde %0-100, titreşimli sistemlerde %10-90 alınır.

Tepe Zamanı (t_t), (t_p): Cevabın referansı aşarak ilk tepe yaptığı zamandır.

Maksimum Aşım (M_p): Cevabın referans değeri en çok aştığı miktardır. Eğer nihai değere ulaşamayıp kalıcı durum hatası oluşuyorsa yüzde olarak verilir.

Oturma Zamanı (t_o), (t_s): Cevap eğrisindeki titreşimlerin %5 veya %2' ye düştüğü süredir. 1. derece sistemlerde oturma zamanı yükselme zamanına eşittir.

Kalıcı Durum Hatası: Sürekli halde, çıkış ile referans girişi arasındaki farktır.

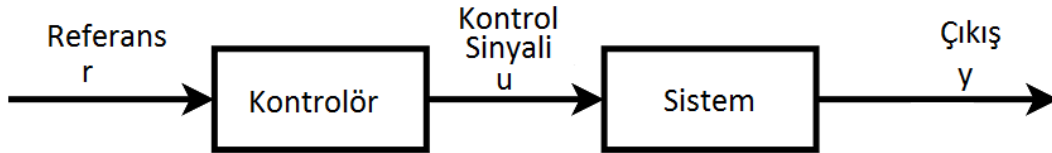
- Verilen bir sistemin bir tipte giriş sinyalinde küçük bir hata verirken diğer bir girişte hata gösterebilir. Bu açık çevrim transfer fonksiyonuna bağlıdır.
- Kalıcı duruma sıfır veya en küçük hata ile ulaşmalıdır.
- Kapalı çevrim kontrol sistemi oluşturularak kalıcı durum hatası açık çevrime göre daha küçük hatalar oluşturulabilir.

2.4.3 Kontrol sistemi çeşitleri

Kontrol sistemleri açık çevrimli kontrol sistemi ve kapalı çevrimli kontrol sistemi olmak üzere ikiye ayrılır.

2.4.3.1 Açık çevrimli kontrol sistemi

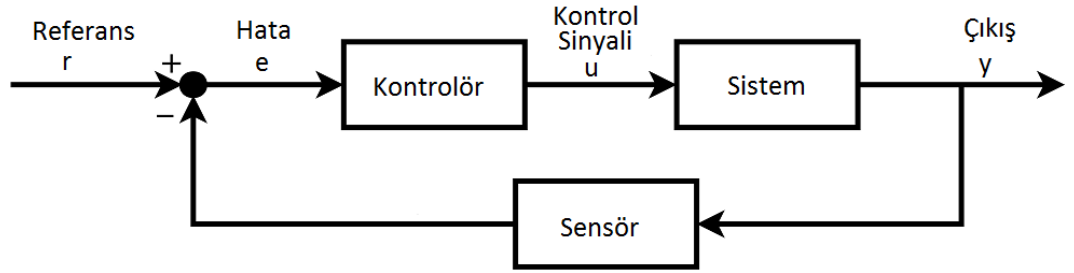
Açık çevrim kontrol sisteminde referans değerine bağlı olarak bir kontrol sinyali üretilir. Sistemin çıkışında oluşan bilgi, girişi etkilemez. Yani geri besleme yoktur. Şekil 2.42'de açık çevrimli bir kontrol sistemi gösterilmiştir.



Şekil 2.42 Açık çevrimli kontrol sistemi

2.4.3.2 Kapalı çevrimli kontrol sistemi

Kapalı çevrim kontrol sisteminde geri besleme kullanılır. Yani sistemin çıkış bilgileri sensörler ile ölçülür, referans değeri ile işleme sokulur ve kontrolöre giriş olarak verilir. Böylece daha kararlı bir kontrol elde edilmiş olur. Şekil 2.43'de kapalı çevrimli bir kontrol sistemi gösterilmiştir.



Şekil 2.43 Kapalı çevrimli kontrol sistemi

2.4.4 Kontrol yöntemleri

Kontrol sistemlerinin davranışlarına göre birbirinden farklı kontrol yöntemleri bulunmaktadır. Bu kontrol yöntemleri P (Proportional - Oransal), I (Integral - İntegral) ve D (Derivative - Türevsel) parametrelerinin kombinasyonları ile oluşturulmuştur. Bu parametrelerin sisteme etkilerini açıklayalım.

P parametresi: Sistemin çıkışında bulunan o anki mevcut hatanın belirli bir kazanç değeri ile çarpılıp sisteme giriş olarak verilmesi mantığını taşır.

I parametresi: Kontrol işleminin başlamasından, I parametresinin hesaplandığı ana kadar geçen tüm hataların toplamına orantılı olarak etki gösterir.

D parametresi: Hatanın değişimi ile doğru orantılı olarak etki gösterir.

Bu parametrelerin sistem üzerindeki etkileri Tablo 2.7'de gösterilmiştir.

Tablo 2.7 P, I ve D parametrelerinin sistem üzerindeki etkileri

Parameter	Yükselme Zamanı	Aşım Değeri	Yerleşme Zamanı	Kalıcı Durum Hatası
K_p	Azalı	Artar	Küçük değişim	Azalı
K_i	Azalı	Artar	Artar	Yok olur
K_d	Küçük değişim	Azalı	Azalı	Etkisi yok

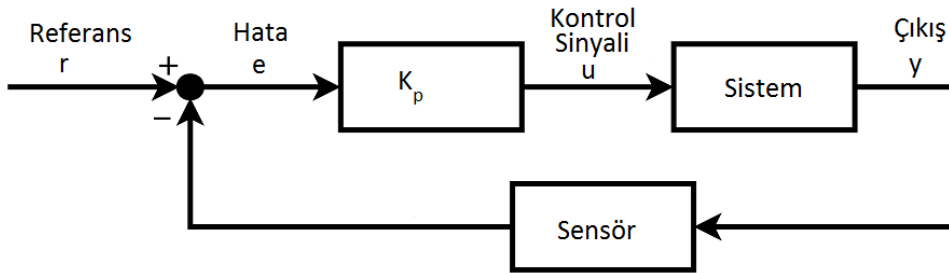
Hangi kontrol parametrelerinin kullanılacağına, kontrol edilecek sistemin karakteristiğine göre karar verilir. Aşağıda bu parametreler kullanılarak oluşturulan kontrol tipleri açıklanmıştır.

2.4.4.1 P (Oransal) Kontrol

Oransal kontrol, basit bir yapıda olması sebebi ile endüstride pek çok uygulama alanı bulmaktadır. Kontrolör çıkışı; o andaki hata değeri ile bir kazanç değerinin çarpılmasıyla hesaplanır. Bu kazanç değeri kontrolör çıkışının sistem hatasına karşı hassasiyetini gösterir. P kontrol, denklem 2.52'de gösterildiği şekilde formülize edilir. Şekil 2.44'de oransal kontrolörlü bir sistemin yapısı gösterilmiştir.

$$u(t) = K_p e(t) \quad (2.52)$$

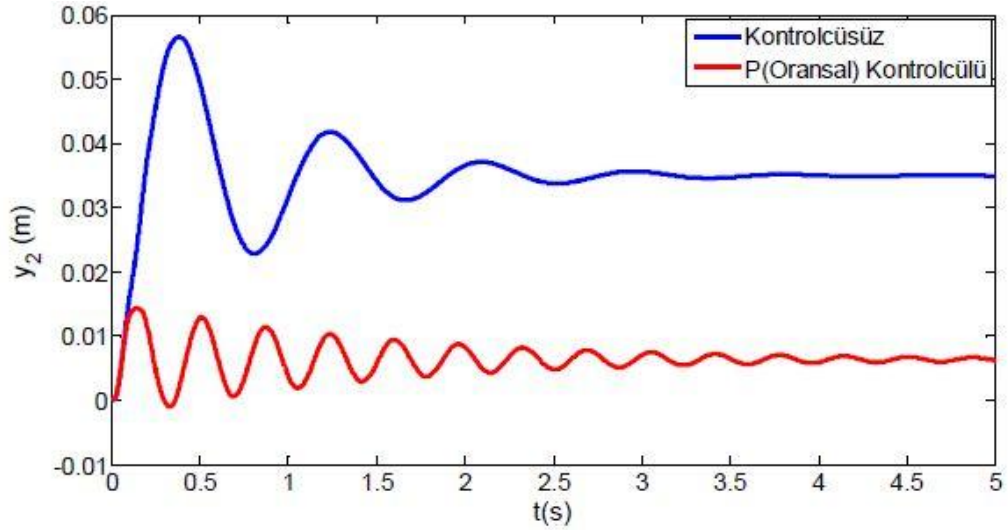
Burada K_p oransal kazanç olarak ifade edilir ve sabit bir değerdir.



Şekil 2.44 P (Oransal) kontrolörlü sistemin yapısı

Bu kontrolör tipinde kazanç değeri büyüdükçe kontrolörün tepkisi artarken, hata azaldıkça daha düşük bir kontrol etkisi gösterilir ve sistemin referansa yumuşak bir şekilde yaklaşması sağlanır. Buna karşılık hata küçüldükçe kontrol etkisi çok zayıflayacak ve sistemin çıkışı referans değere tam olarak oturamadığından kalıcı durum hatası oluşacaktır. K_p değerinin artırılmasıyla kalıcı durum hatası azaltılabilir ancak bu durumda aşım miktarı artacak ve belirli bir değerden sonra salınım başlayacaktır.

Şekil 2.45'de P kontrolün, kontrolsüz bir sistem ile karşılaştırılması gösterilmiştir.



Şekil 2.45 P kontrolün, kontrolsüz bir sistem ile karşılaştırılması (Çelikel, 2006)

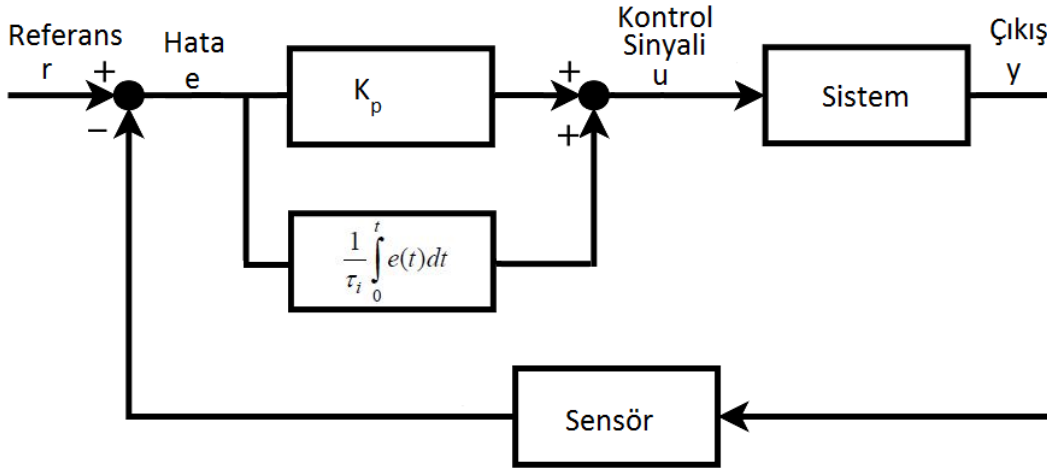
Şekilde de görüldüğü gibi P kontrolör ile kontrol edilen sistemin cevabı, kontrolsüz sistemin cevabına göre daha iyi çıkmıştır. Aşım değeri, yükselme zamanı ve kalıcı durum hatası azalmıştır. Buna karşılık salınım ve oturma zamanı artmıştır.

2.4.4.2 PI (Oransal + İntegral) Kontrol

Oransal kontrolde ortaya çıkan kalıcı durum hatasını yok etmenin yolu, kontrolcüye hatanın integrali ile orantılı olarak çalışan bir parametre eklemektir. Integral, hatanın birikimi ile orantılıdır. Hata değerleri toplanıp belirli bir büyüklüğe ulaştığı anda sisteme büyük bir düzeltme etkisi ile etki eder. I kontrol, kontrol sistemlerinde genellikle P kontrol ile birlikte kullanılırlar. PI kontrolörler denklem 2. 53'te gösterildiği gibi formülize edilir. Şekil 2.46'de PI kontrolörlü bir sistemin yapısı gösterilmiştir.

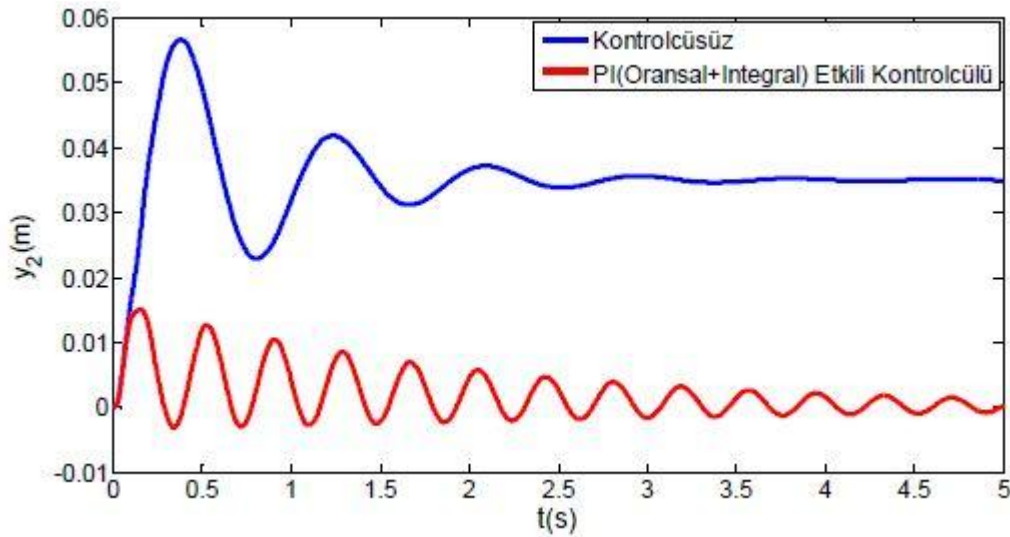
$$u(t) = K_p \left(e(t) + \frac{1}{\tau_i} \int_0^t e(t) dt \right) \quad (2.53)$$

Burada K_p oransal kazanç ve τ_i integral sabitidir.



Şekil 2.46 PI (Oransal + Integral) kontrolörlü sistemin yapısı

Şekil 2.47'de PI kontrolün, kontrolsüz bir sistem ile karşılaştırılması gösterilmiştir.



Şekil 2.47 PI kontrolün, kontrolsüz bir sistem ile karşılaştırılması (Çelikel, 2006)

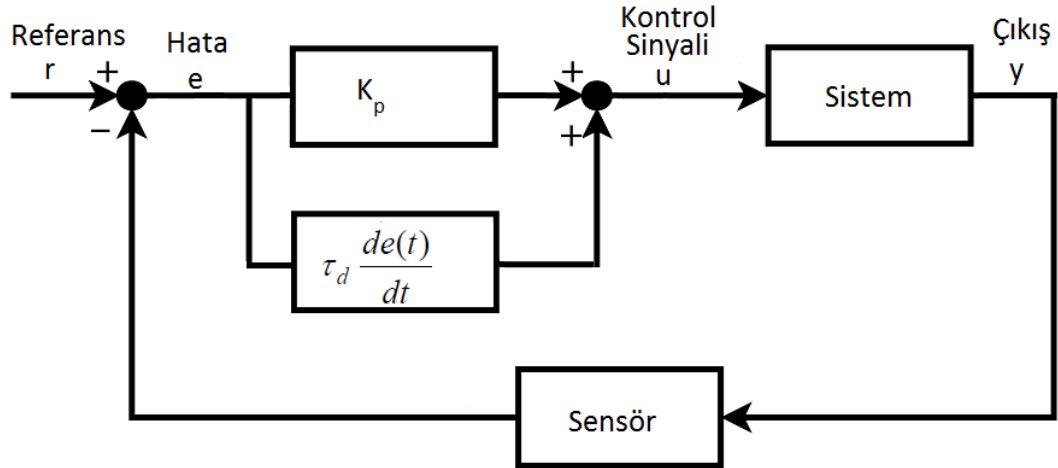
Şekilde de görüldüğü gibi PI kontrolör ile kontrol edilen sistemin cevabı, kontrolsüz sistemin cevabına göre daha iyi çıkmıştır. Aşım değeri, yükselme zamanı ve kalıcı durum hatası azalmıştır. Buna karşılık salınım ve oturma zamanı artmıştır. PI kontrolörü P kontrolör ile kıyaslırsak, kalıcı durum hatasının PI kontrolörde sıfıra inmiş olduğu, ancak aşım değerinin artmış olduğu görülür.

2.4.4.3 PD (Oransal + Türev) Kontrol

Türev parametresinin en önemli özelliği önsezidir. Sistemde büyük bir hata oluşmadan durumu fark eder ve oluşması muhtemel bu hatayı önler. Sabit bir sayının türevi sıfır olduğundan, zamanla değişmeyen bir hata üzerinde etkisi yoktur. Bu nedenle D kontrol, kontrol sistemlerinde yalnız başına kullanılamaz. PD kontrolörler denklem 2. 54'te gösterildiği gibi formülize edilir. Şekil 2.48'de PD kontrolörlü bir sistemin yapısı gösterilmiştir.

$$u(t) = K_p \left(e(t) + \tau_d \frac{de(t)}{dt} \right) \quad (2.54)$$

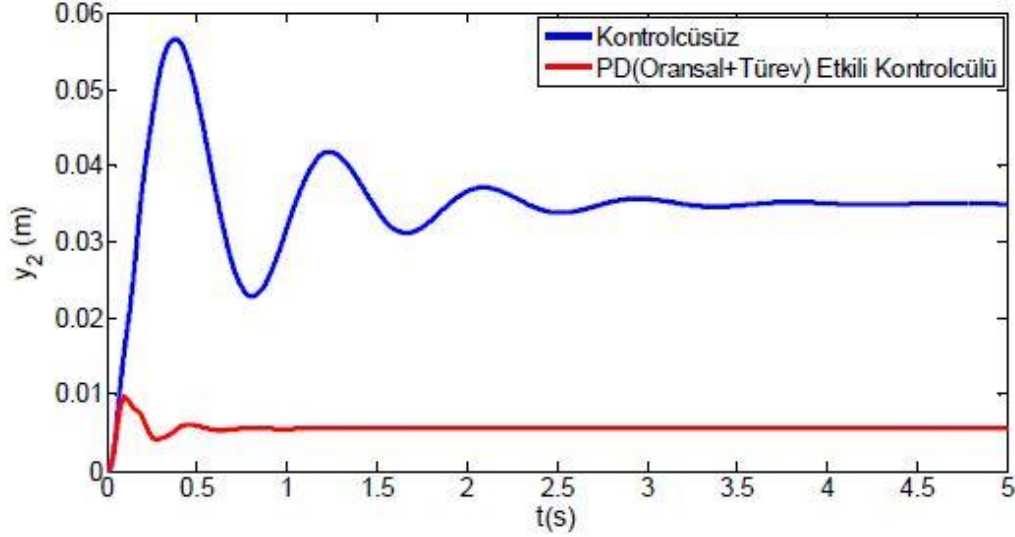
Burada K_p oransal kazanç ve τ_d türev sabitidir.



Şekil 2.48 PD (Oransal + Türevsel) kontrolörlü sistemin yapısı

Şekil 2.49'da PD kontrolün, kontrolsüz bir sistem ile karşılaştırılması gösterilmiştir.

Şekilde de görüldüğü gibi PD kontrolör ile kontrol edilen sistemin cevabı, kontrolsüz sistemin cevabına göre daha iyi çıkmıştır. Aşım değeri, yükselme zamanı ve kalıcı durum hatası azalmıştır. PD kontrolörü P kontrolör ile kıyaslırsak, salınımın ve oturma zamanının azaldığı görülür. PD kontrolörü PI kontrolör ile kıyaslırsak salınımın ve oturma zamanının azaldığı ancak kalıcı durum hatasının arttığı görülür.



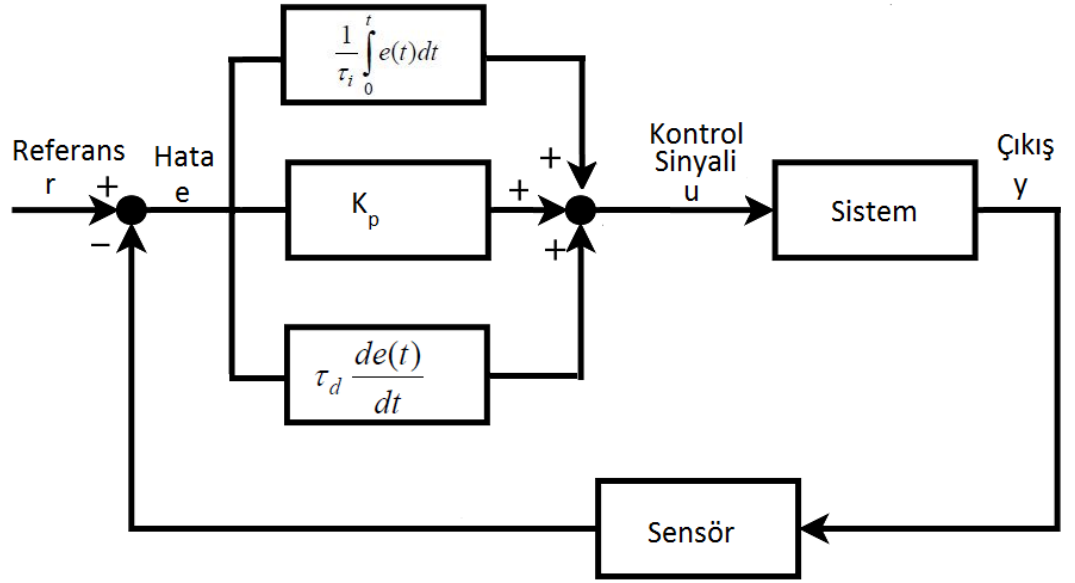
Şekil 2.49 PD kontrolün, kontrolsüz bir sistem ile karşılaştırılması (Çelikel, 2006)

2.4.4.4 PID (Oransal + Integral + Türev) Kontrol

Endüstride en yaygın kullanım alanına sahip kontrol sistemleridir. Bunun sebebi hem lineer hem de nonlineer sistemlerde etkili bir kontrol sağlamasıdır. P, I ve D parametrelerinin avantajları ön plana çıkarken, dezavantajları yok olmaktadır. Örnek olarak, P kontrolcünün sebep olduğu kalıcı durum hatası I kontrolcü ile yok edilirken, I kontrolcünün sebep olduğu yüksek oturma zamanı ve P kontrolcünün sebep olduğu aşım değeri de D kontrolcü ile azaltılacaktır. Böylece kalıcı durum hatası olmayan, hızlı ve kararlı bir kontrolör elde edilmiş olur. PID kontrolörler denklem 2.55'te gösterildiği gibi formülize edilir. Şekil 2.50'de PID kontrolörlü bir sistemin yapısı gösterilmiştir.

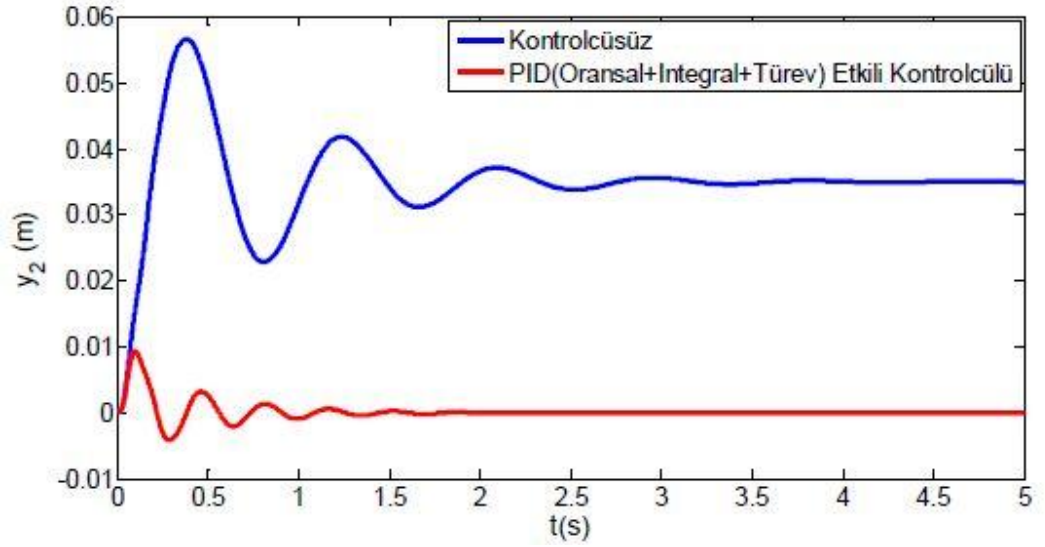
$$u(t) = K_p \left(e(t) + \frac{1}{\tau_i} \int_0^t e(t) dt + \tau_d \frac{de(t)}{dt} \right) \quad (2.55)$$

Burada K_p oransal kazanç, τ_i integral sabiti ve τ_d türev sabitidir.



Şekil 2.50 PID (Oransal + Integral + Türevsel) kontrolörlü sistemin yapısı

Şekil 2.51'de PID kontrolün, kontrolsüz bir sistem ile karşılaştırılması gösterilmiştir.



Şekil 2.51 PID kontrolün, kontrolsüz bir sistem ile karşılaştırılması (Çelikel, 2006)

Şekilde de görüldüğü gibi PID kontrolörün cevabı, diğer tüm sistemlere göre çok daha iyi çıkmıştır. Aşım, yükselme zamanı, oturma zamanı ve salınım olabildiğince azalmış, kalıcı durum hatası sıfıra indirilmiştir. Mükemmele yakın bir kontrol elde edilmiştir.

PID sistemin dezavantajı, belirlenen P, I ve D katsayılarının sabit olmasıdır. Ancak uygulamada, sistemler her zaman aynı şekilde çalışmazlar. Dolayısıyla aynı P,I ve D parametreleri her koşulda optimal bir kontrol sağlayamaz. Bu tez çalışmasında, robot kolunun her bir eklemünde, farklı açılı aralıkları için farklı P,I ve D parametreleri tespit edilmiş ve böylece daha düzgün ve kararlı bir kontrol elde edilmiştir.

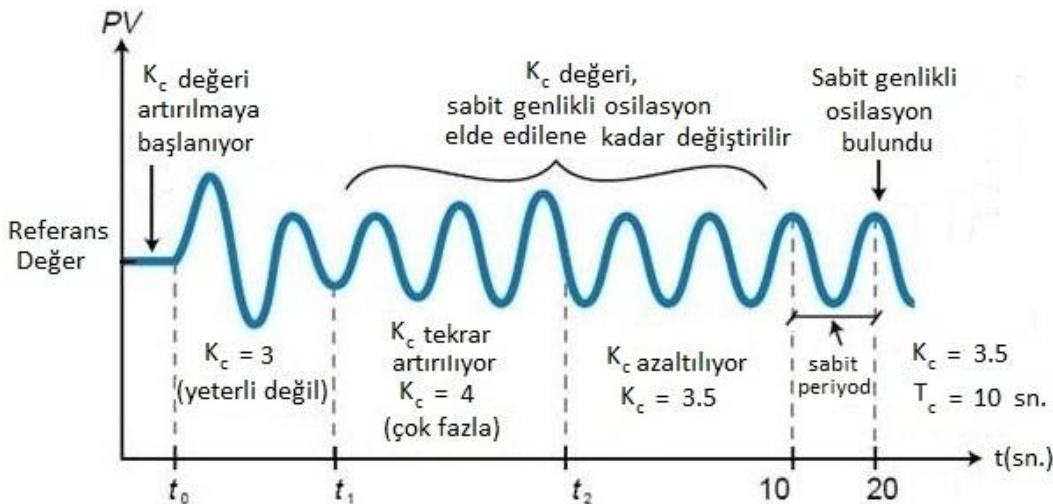
P,I ve D parametrelerinin tespitinde kullanılan birçok yöntem bulunmaktadır. Bu yöntemlerden en kolay ve etkili olanı Ziegler - Nichols yöntemidir.

Ziegler - Nichols Yöntemi:

Ziegler - Nichols (ZN) yöntemi yaygın olarak kullanılan bir yöntemdir. Bunun sebebi, matematiksel modeli olmayan ya da matematiksel modelinin çıkarılması çok zor olan sistemlerde kullanılabilmesidir. Çünkü ZN yönteminde matematiksel model gerekmez. Deneysel yaklaşım yöntemidir.

ZN yöntemine göre K_p , K_i , K_d parametrelerinin hesaplanması aşağıdaki sıralamada yapılır (Ziegler and Nichols, 1942).

1. K_d ve K_i sıfır olarak ayarlanır.
2. K_p değeri, robot sabit genlikte bir salınım hareketine ulaşana kadar artırılır.
3. Ulaşılan bu salınımdaki kazanç değerine K_c denir. Osilasyonun periyodu da T_c olarak adlandırılır. Şekil 2.52'de T_c 'nin bulunması gösterilmiştir.



Şekil 2.52 Ziegler - Nichols yöntemine göre K_c ve T_c nin bulunması

4. K_p , K_i , K_d değerleri Tablo 2.8'deki gibi hesaplanır.

Tablo 2.8 P, I ve D katsayılarının hesaplanması

Kontrolör	K_p	K_i	K_d
P	$0.5 * K_c$	0	0
PI	$0.45 * K_c$	$1.2 * K_p / T_c$	0
PD	$0.8 * K_c$	0	$0.125 * K_p * T_c$
PID	$0.6 * K_c$	$2 * K_p / T_c$	$0.125 * K_p * T_c$

2.5 TQ MA2000 Robot Kolu

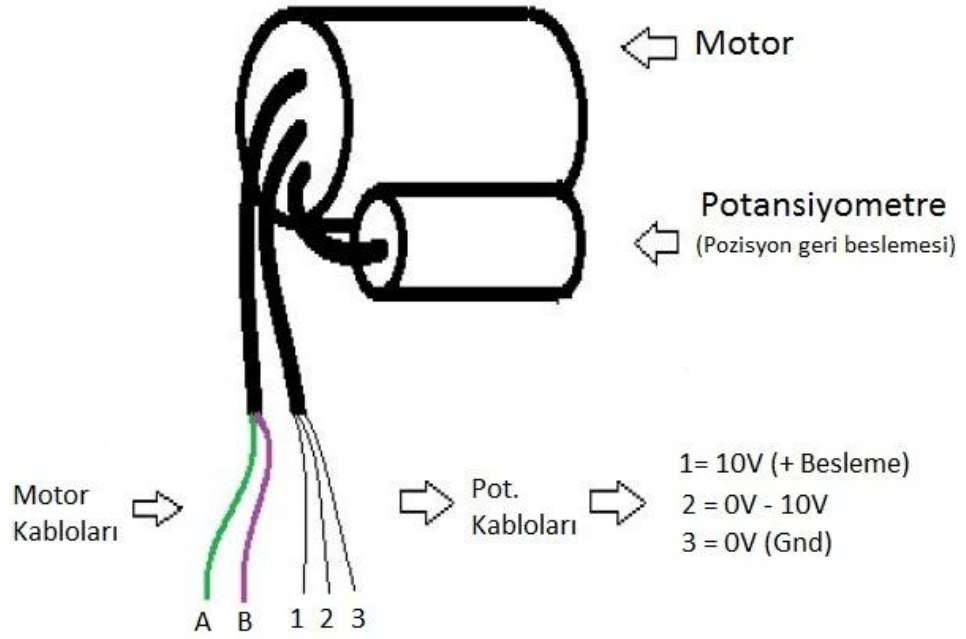
Bu tez çalışmasında, nesnelerin taşınması amacıyla 6 döner eklemlili (6R) bir robot kol olan TQ MA2000 kullanılmıştır. MA2000 robotunun çalışma uzayı yarıçapı 500 mm olan bir çemberdir. Bu robot, 3 adet ana eklem (bel, omuz, dirsek) ve üç adet yardımcı eklemden (eğme, sapma, döndürme) oluşmuştur. Şekil 2.53'te TQ MA2000 robotu için SolidWorks programında yapılan çizimi gösterilmiştir.



Şekil 2.53 TQ MA2000 robotunun SolidWorks programında yapılan çizimi

DC Motorlar:

Bütün eklemler Şekil 2.54'te gösterilen yapıdaki Maxon marka fırçalı DC motorlar ile tahrik edilmektedir. Bu motorlar yüksek performanslı Alnico mıknatıslar ile üretilmiştir. Alnico mıknatıs metal alüminyum, kobalt, demir ve diğer metal elementlerden imal edilmektedir. Bu DC motorların tamamında, pozisyon geri beslemesini sağlayabilmek için potansiyometreler bulunmaktadır. Potansiyometrelerden ölçülen direnç değerleri, açı değerleri ile orantılıdır. Potansiyometreye voltaj uygulandığında, eklemin kol açısına bağlı olarak bir çıkış oluşacaktır. Bu çıkış değeri mikroişlemcilerin analog kanallarında yorumlanarak robotun eklem açılarının kontrolü yapılır.



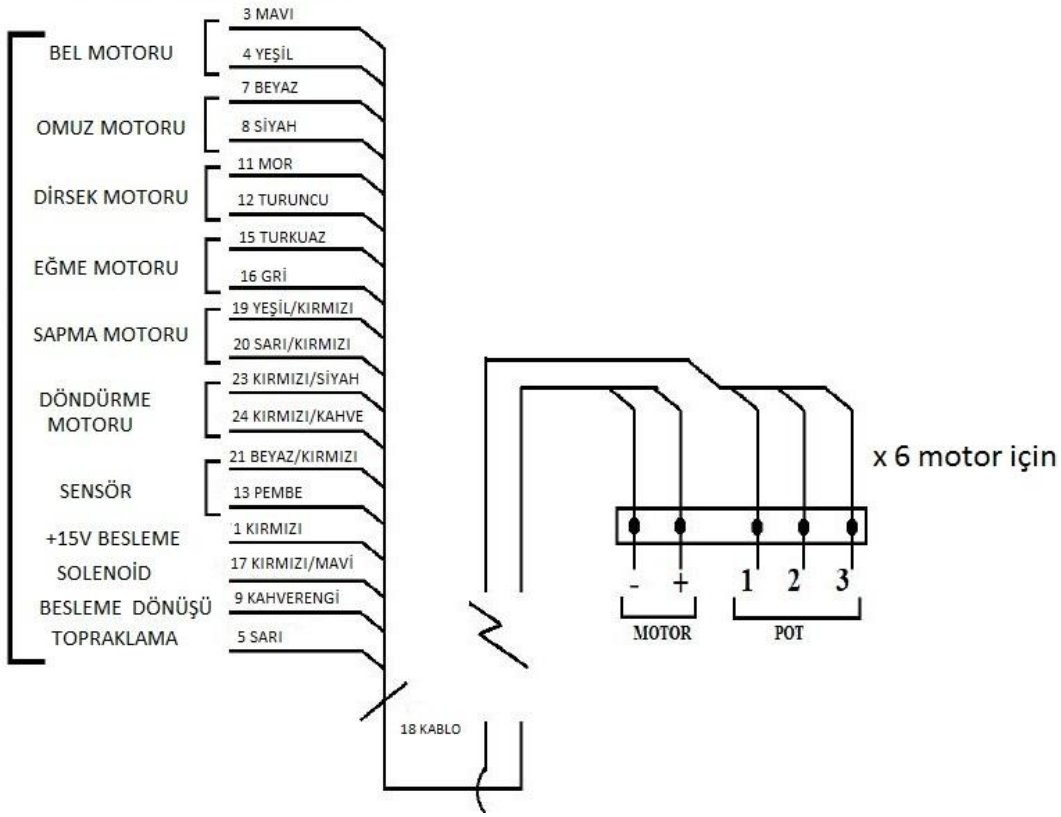
Şekil 2.54 TQ MA2000 robot kolunda kullanılan DC motorların yapısı

MA2000 Bağlantı Şeması:

MA2000 robotunda bulunan 6 adet motorun kabloları ve besleme uçları 18 pinli bir konnektörde birleştirilmiştir. Şekil 2.55'de bu konnektör ve iç yapısı gösterilmiştir.

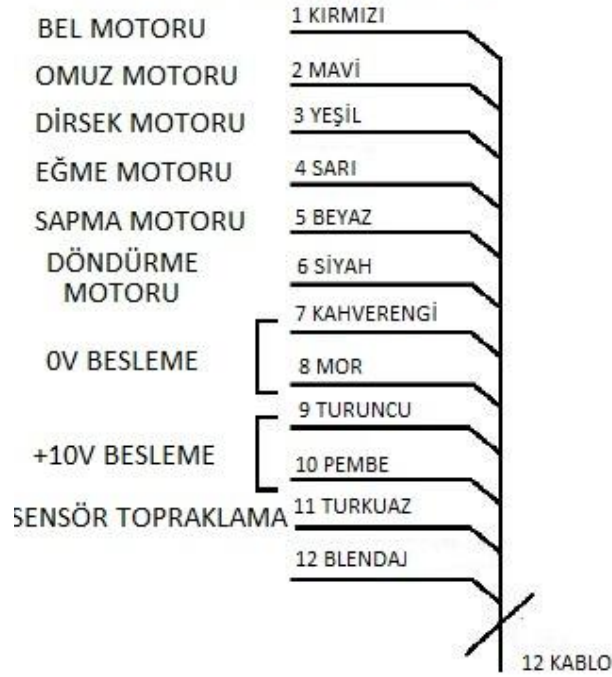
MA2000 robotunun motor pozisyon bilgilerinin alındığı potansiyometrelerin kabloları da 12 pinli bir konnektörde birleştirilmiştir. Şekil 2.56'da bu konnektör ve iç yapısı gösterilmiştir.

18 PİNLİ KONNEKTÖR



Şekil 2.55 Motor kablolarını içeren 18 pinli konnektör

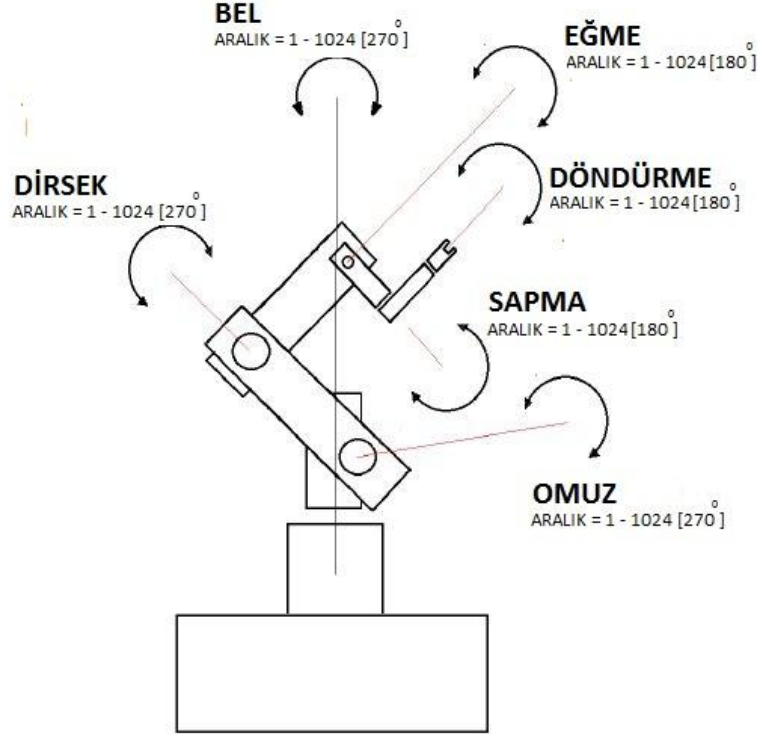
12 PİNLİ KONNEKTÖR



Şekil 2.56 Potansiyometre kablolarını içeren 12 pinli konnektör

Eklemlerin Çalışma Aralığı:

Şekil 2.57'de MA2000 robot kolundaki 6 eklemin potansiyometrelerinin analog değerleri ve eklemlerin açısıl çalışma aralığı gösterilmiştir.



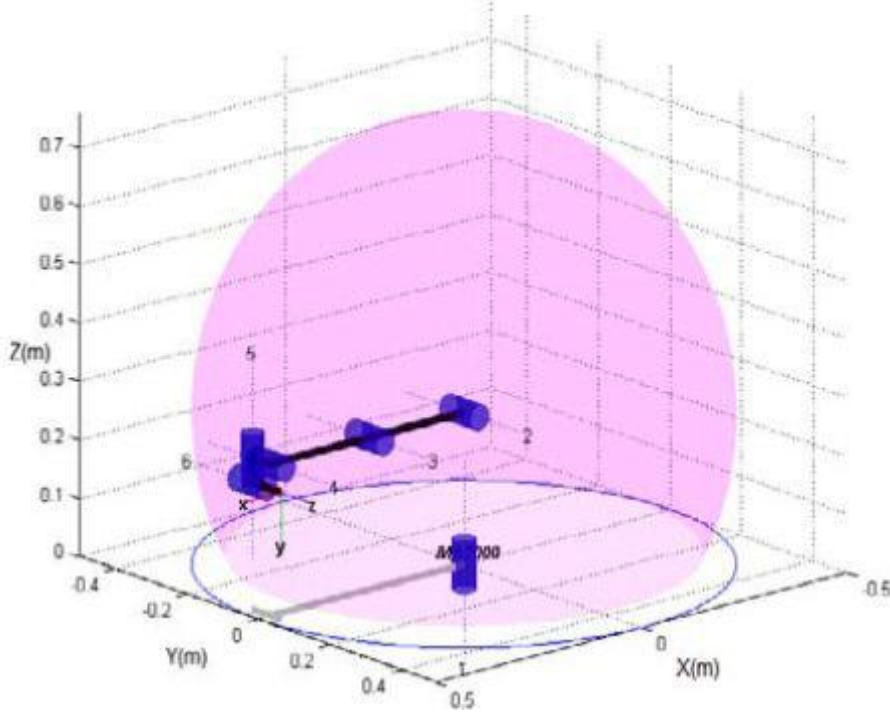
Şekil 2.57 MA2000 robot kolunun açı ve potansiyometre ilişkisi (Quinn, 2013)

Çalışma Uzayı:

MA2000 robot kolunun çalışma uzayı Şekil 2.58'de gösterildiği gibi MATLAB programında simüle edilmiştir.

Tutucu:

Robotun ucuna pnömatik bir tutucu eklenmiştir ve bu tutucu 3 bar hava basıncında çalışmaktadır. Bu tutucu sayesinde nesneler bir yerden başka bir yere taşınabilmektedir.



Şekil 2.58 MA2000 robot kolunun çalışma uzayı benzetimi (Al-Saedi and Mohammed, 2012)

MA2000 robot kolunun ileri ve ters kinematik analizleri ileriki bölümlerde detaylı olarak anlatılacaktır.

2.6 Arduino

2.6.1 Arduino nedir?

Arduino; kısa bir şekilde bir giriş - çıkış kartı olarak tanımlanabilir. Processing/Wiring dilinin bir uygulamasını içeren geliştirme ortamından, açık kaynak kodlu olarak geliştirilmiştir. Esnek ve kolay kullanımlıdır. Arduino üzerindeki mikroişlemci (AtmegaXX), Arduino'nun kendine özgü programlama dili ile Arduino IDE ortamında programlanır ve karta yüklenir.

Arduino'nun birçok uygulamada tercih edilmesine yol açan bazı özellikleri şunlardır:

- Açık kaynaklı bir donanıma sahiptir yani kartların devre tasarımları tamamen açıktır.
- Açık kaynaklı bir yazılıma sahiptir. Arduino IDE platformunda yazılım yazılır.
- Program geliştirmek kolaydır. Kart üzerinde yazılımın testi rahatlıkla yapılabilir.
- Arduino programlamada kullanılan yazılım dili basittir. Ayrıca internet ortamında birçok kaynak da programlamada yol göstericidir.
- Arduino'nun çok geniş bir kütüphane desteği vardır.
- Çok yaygın bir kullanımı olduğu için, yaptığınız uygulamaya benzer bir uygulamanın daha önce yapılmış olması muhtemeldir. Bu nedenle yardım alabileceğiniz kaynaklar mutlaka vardır.
- Arduino'ya özel hazırlanmış eklentiler (shield) sayesinde birçok yapı ilave edilebilir.
- Ucuzdur.

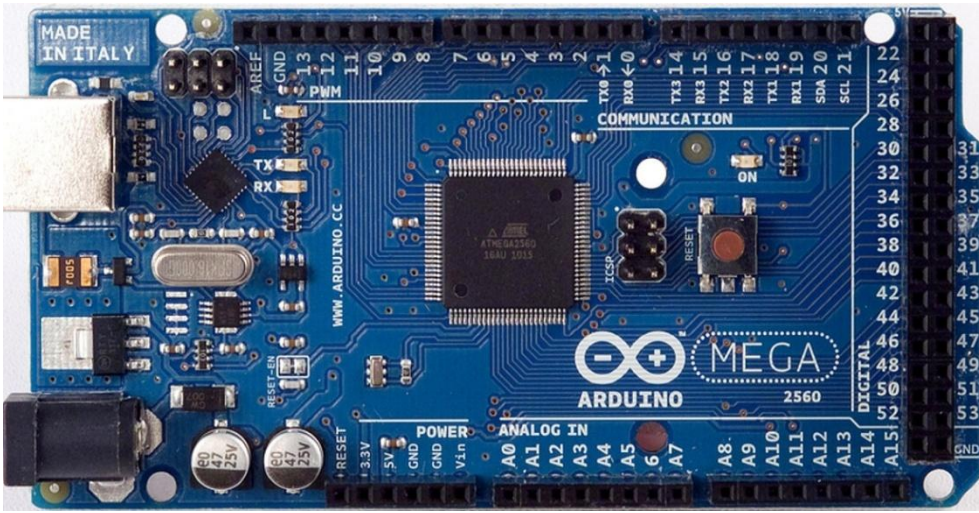
Arduino'nun birbirinden farklı birçok kartı bulunmaktadır. Uygulamadaki gereksinimlere göre kart seçimi yapılır. Bu çalışmada Arduino Mega 2560 R3 mikrodnetleyici kartı kullanılmıştır.

2.6.2 Arduino Mega 2560 R3 mikrodnetleyici kartı

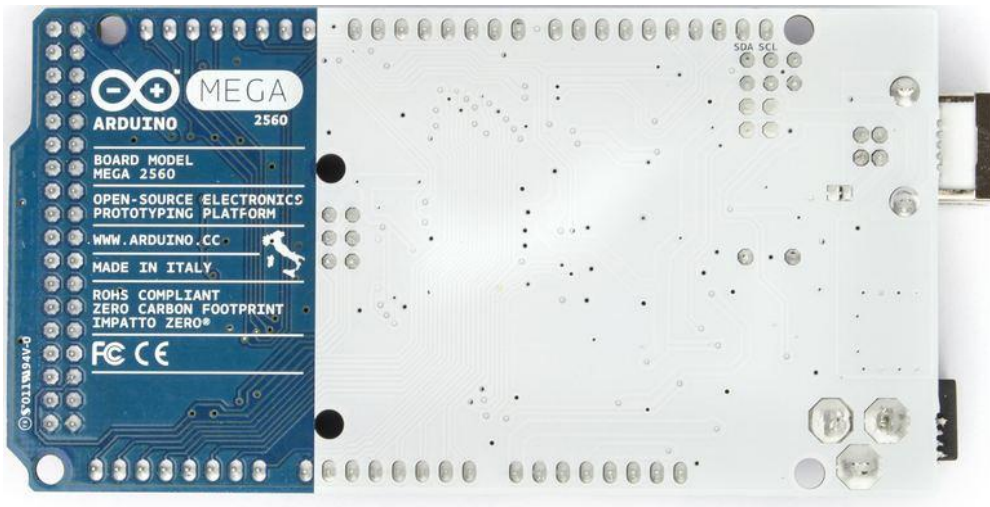
Arduino Mega 2560, ATmega2560 tabanlı bir mikrodnetleyici kartıdır. Bu kartta;

- 54 adet dijital giriş ve çıkış (bunlardan 15 tanesi PWM çıkışı üretebilmekte),
- 16 tane analog giriş,
- 4 UART(donanımsal seri port)
- 16 MHZ kristal osilatör,
- USB bağlantısı,
- Adaptör girişi,
- ICSP bağlantısı,
- Bir adet reset butonu vardır.

Şekil 2.59'da Arduino Mega 2560 R3 kartının ön yüzü, Şekil 2.60'da arka yüzü gösterilmiştir.



Şekil 2.59 Arduino Mega 2560 R3 kartının ön yüzü



Şekil 2.60 Arduino Mega 2560 R3 kartının arka yüzü

Arduino Mega 2560 R3, Arduino Mega'nın birkaç özellik eklenmiş halidir. Eklenen bu özellikler;

- USB üzerinden programlanmayı ve haberleşmeyi sağlayan FTDI ve ATmega8U2 yerine ATmega16U2 çipi sayesinde çok daha hızlı programlanabilmekte ve yeni çip üzerindeki hafızadan dolayı daha hızlı veri transferi yapabilmektedir.
- Linux ve Mac işletim sistemleri için USB sürücüyeye olan ihtiyaç ortadan kalkmıştır. Windows için ise Arduino IDE ile birlikte gelen "inf" dosyasını windows'a tanıtmak gerekmektedir.
- Bilgisayara klavye, joystick veya mouse olarak tanıtılabilir.
- Ekstra olarak birer tane SDA ve SCL pinleri konulmuştur.
- Reset pininin yanında konumlandırılmış IOREF ve hiçbir bağlantısı olmayan, genel kullanım amacıyla yerleştirilmiş isimsiz yeni pinler mevcuttur. IOREF pini Arduino eklentilerini kart üzerinden beslemek için kolaylık sağlamaktadır.

Genel Özellikleri:

Mega 2560 R3 kartının genel özellikleri Tablo 2.9'da özetlenmiştir.

Tablo 2.9 Arduino Mega 2560 R3 kartının genel özellikleri

Mikrodenetleyici	ATmega2560
Çalışma Gerilimi	5V
Giriş Voltajı (önerilen)	7-12V
Giriş Voltajı (limit)	6-20V
Dijital Giriş/Çıkış Sayısı	54 (15'ü PWM çıkışı)
Analog Giriş Sayısı	16
Giriş/Çıkış Pinlerinin Akımı	40 mA
3.3V Pini Akımı	50 mA
Flash Bellek	256 KB (8 KB'ı bootloader a ayrılmıştır)
SRAM	8 KB
EEPROM	4 KB
Osilatör Frekansı	16 MHz

Güç Pinleri:

Arduino Mega 2560 R3, USB bağlantısı ile beslenebileceği gibi üzerinde yer alan adaptör jak ile de beslenebilir. Kart, harici güç girişi üzerinden 6 ile 20 volt gerilim aralığında çalışabilen ac-dc çevirici adaptör ile veya batarya ile beslenebilir. Ancak 7 voltun altındaki gerilim değerlerinde çıkış pinleri gerilim değerleri 5 voltun altına düşebileceği ve 12 voltun üzerinde besleme voltajlarında regülatör devresindeki ısınma karta zarar verebileceğinden dolayı tavsiye edilen besleme voltaj aralığı 7 - 12 volt aralığıdır.

Güç pinleri aşağıda verilmiştir.

- **V_{IN}:** Kartın harici güç kaynağı ile beslenmesi durumunda pozitif ucun bağlanacağı pindir. Güç jakının pozitif ucu bu pine bağlıdır.
- **5V:** Regüle edilmiş 5 volt çıkış pinidir. Kart; güç yakından, V_{IN} pini üzerinden veya USB ile beslenebilmektedir. Ancak kartın bu 5V pini üzerinden veya 3.3V pini üzerinden beslenmesi karta zarar verebilir. Bundan kaçınılmalıdır.
- **3.3V:** Kart üzerinde regüle edilmiş 3,3 voltluk çıkış pinidir. Maksimum 50mA'lık akım basabilmektedir.
- **GND:** Kartın toprak pinidir.
- **IOREF:** Arduino kartının çalışma voltajını gösteren pindir. Arduino eklentileri tarafından kullanılır ve voltaj referans değerini gösterir.

Hafıza:

Atmega2560 mikrodenetleyicisi, kod depolamak için 256KB flash hafızaya (8 KB'ını bootloader kullanır), 8KB SRAM'e ve 4 KB EEPROM'a sahiptir.

Giriş ve Çıkış Pinleri:

Arduino Mega kartındaki 54 dijital pinden her biri giriş veya çıkış olarak ayarlanıp kullanılabilir. Bu pinlerin her biri 5 voltta çalışmakta ve

maksimum 40mA akım sağlayabilmektedir. Bu pinlerin hepsi 20- 50 kOhm pull-up direncine sahiptir.

Bu 54 pinden bazıları, özel birtakım fonksiyona sahiptir. Bunlar;

- **Seri İletişim:**

Serial: 0 (RX) - 1 (TX)

Serial 1: 19 (RX) - 18 (TX)

Serial 2: 17 (RX) - 16 (TX)

Serial 3: 15 (RX) - 14 (TX)

Seri iletişimde RX pinleri alıcı, TX pinleri ise gönderici pinleri olarak kullanılmaktadır. 0 ve 1 pinleri USB üzerinden seri iletişimde kullanılmak üzere USB - TTL dönüştürücü olan Atmega16U2 çipine bağlıdır. Yazılan program, bu pinler üzerinden mikrodenetleyiciye yüklenmektedir.

- **Dış Kesmeler:**

2 (kesme 0),

3 (kesme 1),

18 (kesme 5),

19 (kesme 4),

20 (kesme 3),

21 (kesme 2).

Bu pinler düşen kenar, yükselen kenar veya düşük ve yüksek voltaj değerlerinde kesme üretmesi için ayarlanıp kullanılabilir.

- **PWM:**

2 - 13 arasındaki 12 pin ve 44 -46 arasındaki 3 pin PWM çıkışı olarak kullanılabilir. Bu pinler üzerinden 8-bit PWM çıkışı üretilebilir.

- **SPI:**

50 (MISO),

51 (MOSI),

52 (SCK),
53 (SS).

Bu pinler SPI iletişiminde kullanılabilmektedir.

- **LED:**

Kart üzerinde 13 numaralı pine bağlı bir led bulunmaktadır. Pin aktif hale getirilirse (high) led yanar, pin pasif hale getirilirse (low) led söner.

- **TWI:**

Kart, 20 (SDA) ve 21 (SCL) pinleri üzerinden TWI (I2C protokolü ile) iletişimi desteklemektedir.

Mega2560 16 analog giriş pinine sahiptir. Bu pinlerin her biri 10 bit çözünürlüktedir (1024 farklı değer). Varsayılan referans değeri olarak 0-5 volt aralığıdır. Fakat istenildiğinde AREF pinine istenilen referans voltaj değeri girilebilir. AREF pinine voltaj değeri girildiğinde analog girişler, 0 ile gösterilen referans voltaj değerine göre değerlendirilir (AREF : Analog-Referans : Analog girişler için referans değeri).

Kartın Programlanması:

Arduino IDE ortamı üzerinden programlanır. Bootloader'ı (karta yazılımın yüklenmesini sağlayan küçük bir kod parçası) içerisinde gelir. Bootloader'ı atlanıp ICSP üzerinden direkt olarak mikrodenetleyici programlanabilir. Programın yazıldığı IDE programından bir görüntü Şekil 2.61'de gösterilmiştir.

USB Aşırı Akım Koruması:

Arduino Mega 2560 R3'ün USB'den gelecek aşırı akımlara karşı koruması vardır. 500 mA'in üzerinde bir akım değeri uygulanırsa üzerinde bulunan sigortası atar.

Fiziksel Özellikleri ve Eklenti(Shield) Uyumluluğu:

Arduino Mega2560 R3'ün boy ve en uzunlukları 100mmx50mm'dir. Bu kart, Arduino için üretilmiş tüm eklentilere uyumludur.



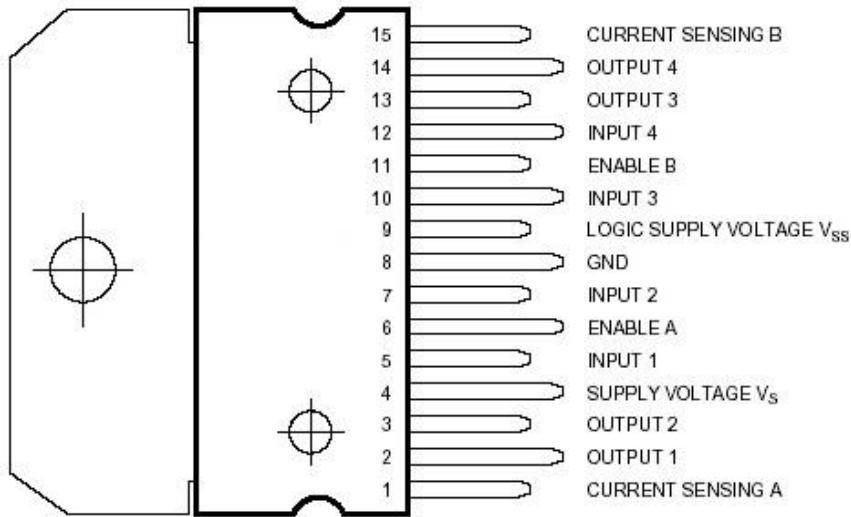
Şekil 2.61 Arduino IDE

Bu tez çalışmasında kullanılan Arduino Mega 2560 R3 kartının bağlantı şemaları ve karta yazılan yazılımın içeriği ileriki bölümlerde anlatılacaktır.

2.7 L298 Motor Sürücü Entegresi ve Kartı

2.7.1 L298 motor sürücü entegresi

L298, iki kanallı bir motor sürücü entegresidir. Aynı anda birbirinden bağımsız olarak 2 adet motor sürmeye imkan tanır. Yüksek akıma karşı korumalıdır, kanal başına 2 ampere kadar akım verebilir. İçerisinde 2 adet H köprüsü bulunur. Toplam 15 adet bacağı bulunmaktadır. Bunlardan Input1, Input2, Output1, Output2, Enable A ve Current Sensing A A köprüsü için; Input3, Input4, Output3, Output4, Enable B ve Current Sensing B B köprüsü içindir. V_{ss} , V_{ss} ve Gnd bacakları ise besleme bacaklarıdır. Şekil 2.63'de L298 entegresinin bacak yapısı gösterilmiştir.



Şekil 2.62 L298 entegresi bacak yapısı

Bu bacakların nasıl kullanılacağını açıklayalım.

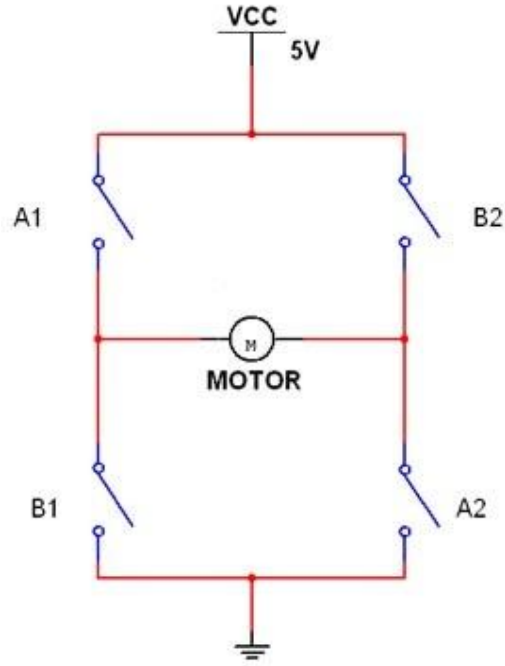
- Input1, Input2 (5, 7): A kanalının giriş bacaklarıdır. Mikroişlemciden çıkan sinyal bilgileri bu giriş bacaklarına bağlanır. Eğer Input1'e 5V, Input2'ye 0V verildiğinde motor saat yönünde dönüyorsa, tam tersi şekilde besleme verildiğinde saatin tersi yönde dönecektir. Her iki pini de aynı voltaj değeri ile beslersek motor dönmez.
- Input3, Input4 (10, 12): B kanalının giriş bacaklarıdır. Kullanımı A kanalı ile aynıdır.

- Output1, Output2 (2, 3): A kanalının çıkış bacaklarıdır. Motorlardan gelen kablolar bu bacaklara bağlanır.
- Output3, Output4 (13, 14): B kanalının çıkış bacaklarıdır. Kullanımı A kanalı ile aynıdır.
- Enable A, Enable B(6, 11): A ve B kanallarının kullanılabilmesi için bu bacakların +5V ile beslenmeleri gerekir.
- Current Sensing A, Current Sensing B (1, 15): A ve B kanallarının kullanılabilmesi için bu bacaklar toprağa bağlanır.
- Supply Voltage V_s (4): Çıkışlardan kaç volt alınmak isteniyorsa bu bacağı o voltaj değeri bağlanır. En fazla 46 volt verilebilir.
- VSS (9): Bu bacak, L298' in besleme bacağıdır ve +5 volt ile beslenmelidir.
- Gnd (8): Bu bacak, L298' in topraklama bacağıdır.

L298 entegresinin kullanılabilmesi için H köprüsü mantığının bilinmesi gerekmektedir. Bu nedenle detaylı olarak anlatılmasında fayda vardır.

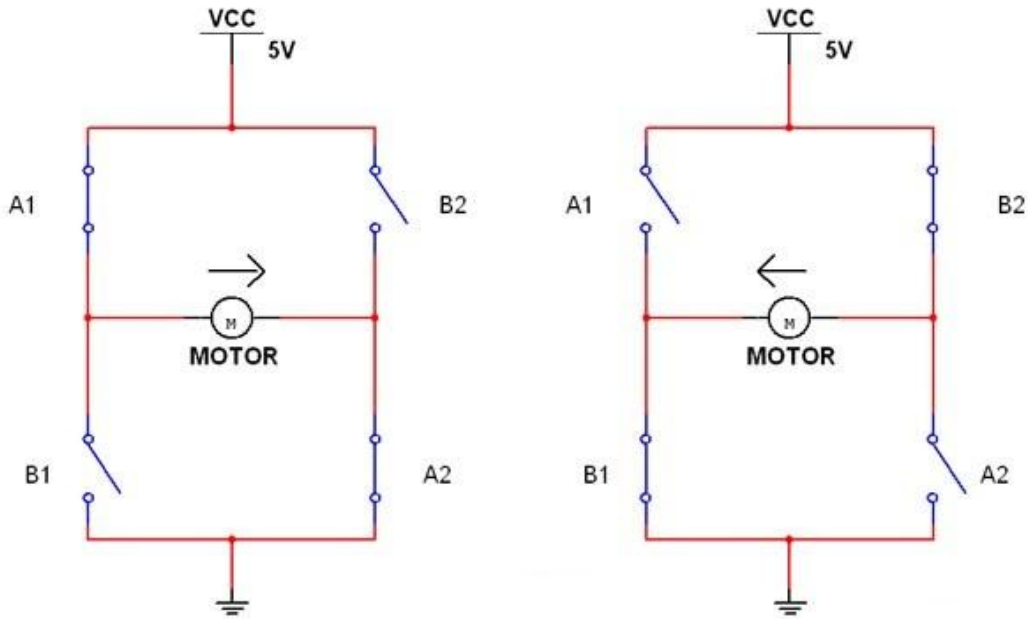
H köprüsü:

H köprüsü, DC bir motoru hem ileri hem de geri yönde sürmeyi sağlayan bir yapıdır. 4 adet transistör (triyak ya da tristör de kullanılır) ile anahtarlama yapılarak kullanılır. Akımın yönünün dijital olarak kontrol edilmesi mantığını taşır. Transistör, triyak ya da tristör kullanılmasından dolayı düşük akımlarla yüksek akımlar kontrol edilebilir. Yapı olarak H harfine benzemesinden dolayı H köprüsü adı verilmiştir. Şekil 2.63'de H köprüsünün yapısı gösterilmiştir.



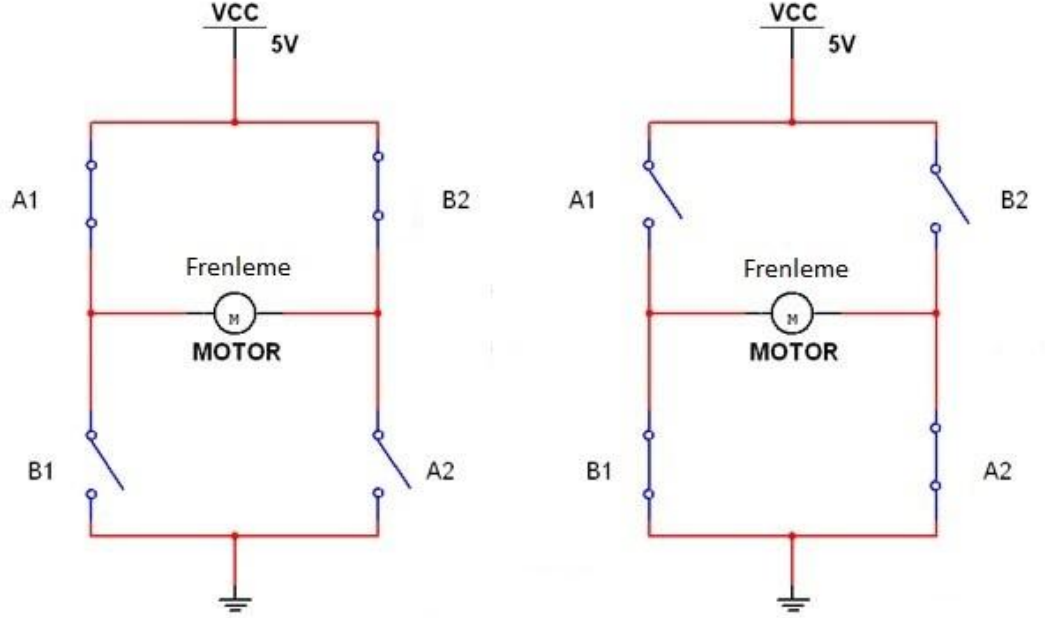
Şekil 2.63 H köprüsünün yapısı

H köprüsü kullanılarak motorların yönlerini kontrol etmek için, transistörler çapraz olarak aktif edilir. Örnek olarak; A1 - A2 transistörleri aktif hale getirilirse motor saat yönünde, B1 - B2 transistörleri aktif hale getirilirse motor saatin tersi yönünde hareket edecektir. Şekil 2.64'de bu durumlar gösterilmiştir.



Şekil 2.64 H köprüsü kullanarak motorların yön kontrolü

Eğer A1 - B2 transistörleri birlikte aktif edilirse motor kilitlenir ve elektronik frenleme yapılmış olur. Aynı durum B1 - A2 transistörlerinin aktif edilmesi halinde de geçerlidir. Şekil 2.65'de bu durum gösterilmiştir.



Şekil 2.65 H köprüsünde frenleme yöntemi

H köprüsünün doğruluk tablosu tablo 2.10'da gösterilmiştir.

Tablo 2.10 H köprüsü doğruluk tablosu

A1	A2	B1	B2	Dönme Yönü
1	1	0	0	saat yönü
0	0	1	1	saatin tersi yönü
1	0	0	1	frenleme
0	1	1	0	frenleme
1	0	1	0	kısa devre
0	1	0	1	kısa devre

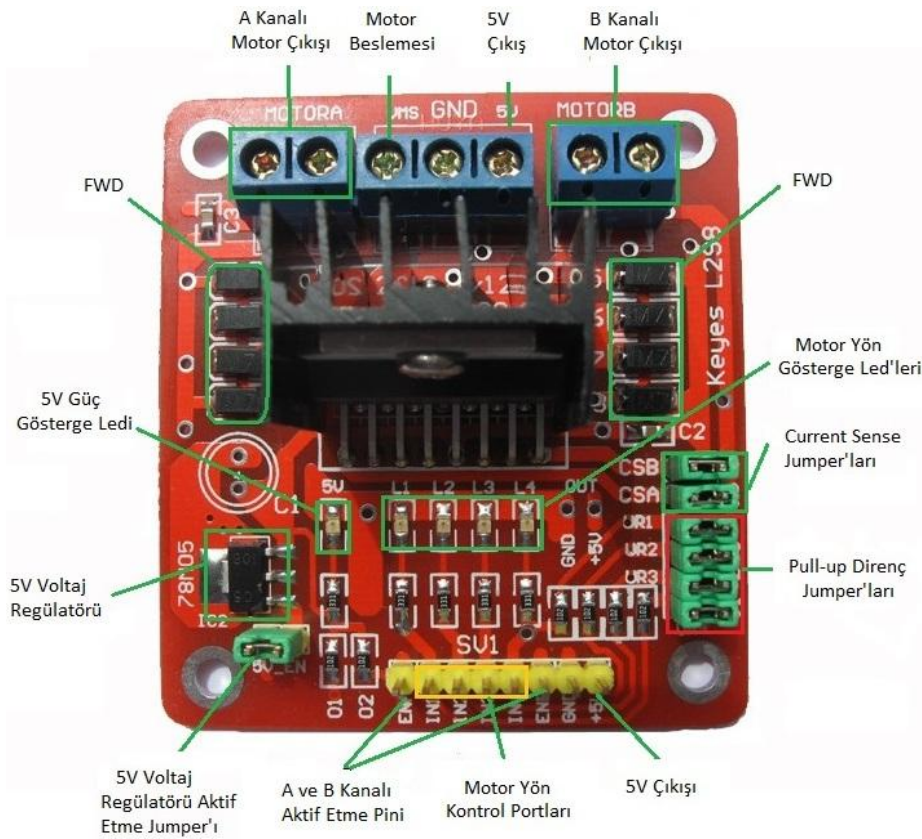
2.7.2 L298 entegreli motor sürücü kartı

Bu motor sürücü kartı, L298 entegresinin kolay kullanımı için kart haline getirilmiş şeklindedir. Şekil 2.66'da L298 entegreli motor sürücü kartı gösterilmiştir.

Kompakt bir yapıdadır. L298 entegresinin özelliklerinin hepsini taşır, birkaç ekstra özellik de eklenmiştir. Aşağıda eklenen bu ekstra özellikler verilmiştir.

Özellikleri:

- Motorun hangi yönde döndüğünün anlaşılması için, iki kanalda da dönüş yönüne göre yanan ikişer adet led bulunmaktadır.
- Karta entegre edilmiş regülatör mevcuttur.
- Yüksek sıcaklığa ve kısa devreye karşı koruması bulunmaktadır.
- Karta entegre edilmiş soğutucu mevcuttur.
- Akım okuma pinleri dışa verilmiş haldedir.



Şekil 2.66 L298 entegreli motor sürücü kartı

L298 motor sürücü kartının pin bağlantı bacakları L298 entegresi ile aynı sayıda ve yapıdadır. Ek olarak üzerinde jumper takılmış pinler bulunmaktadır. Bu pinler farklı özellikleri aktif etmede kullanılır. Aşağıda bu pinlerin özellikleri verilmiştir.

- CSA: A kanalının akım çıkış pinidir. Bu pinde bulunan jumper çıkartılarak A kanalındaki motorun çektiği akım değeri analog olarak okunabilir.
- CSB: B kanalının akım çıkış pinidir. Bu pinde bulunan jumper çıkartılarak B kanalındaki motorun çektiği akım değeri analog olarak okunabilir.
- VR1: Input1 girişini 5V'a çeken pull-up direncine bağlı olan jumperdir. Bu jumper kullanılırsa, Input1 girişine sürekli olarak 5V gelir.
- VR2: Input2 girişini 5V'a çeken pull-up direncine bağlı olan jumperdir. Bu jumper kullanılırsa, Input2 girişine sürekli olarak 5V gelir.
- VR3: Input3 girişini 5V'a çeken pull-up direncine bağlı olan jumperdir. Bu jumper kullanılırsa, Input3 girişine sürekli olarak 5V gelir.
- VR4: Input4 girişini 5V'a çeken pull-up direncine bağlı olan jumperdir. Bu jumper kullanılırsa, Input4 girişine sürekli olarak 5V gelir.
- 5V-EN: 7805 regülatörünü aktifleştirmek için kullanılan jumperdir. Takıldığında kartın regülatörü aktif hale gelir ve bu pinden 5V çekilebilir.

L298 entegreli motor sürücü kartı, sağladığı birçok avantaj nedeniyle bu tez çalışmasında kullanılmıştır. L298 motor sürücü kartının bu tez çalışmasında nasıl kullanıldığı ileriki bölümlerde ayrıntılı olarak anlatılacaktır.

3. YAPILAN ÇALIŞMALAR

Bu bölümde; robotun ve diğer birimlerinin bağlantıları, nesnenin orta noktasının bulunabilmesi için kameradan alınan görüntünün hangi görüntü işleme adımlarından geçtiği, TQ MA2000 robot kolunun ileri ve ters kinematik analizi ve orta noktası bulunan nesnenin bulunduğu yerden alınabilmesi için uzuvlarının alması gereken açı değerlerinin hesaplanması ve robotun nesneyi alabilmek amacıyla yaptığı hareketin kontrolü üzerinde durulmuştur.

3.1 Robotun Bağlantı Şeması

Çalışmada kullanılan robota ait üç adet motorun, motorların geri besleme elemanı olarak kullanılan üç adet potansiyometrenin, motorları sürmede kullanılan üç adet L298 motor sürücü kartının ve tüm sistemi denetim altında tutan Arduino mikrodenetleyici kartının Proteus programında çizilen bağlantı şeması Ek 1'de verilmiştir.

3.1.1 Motorların sürücü kartlarına bağlanması

TQ MA2000 robotunda 6 adet eklem ve buna bağlı olarak 6 adet motor bulunmasına rağmen, bu çalışmada ana eklemler olan ilk üç eklem kullanılmıştır. L298 motor sürücü kartlarının her birinde aynı anda 2 adet motor sürülebilmektedir. Bu nedenle 3 motoru sürmek için 2 adet sürücü kartı yeterli olmaktadır. Ancak ileriki çalışmalarda tüm eklemlerim kullanılabileceği göz önünde bulundurularak 3 adet sürücü kartı kullanılmış ve böylece aynı anda 6 motorun sürülebilmesine imkan sağlanmıştır. Bu çalışmada kullanılan 3 motorun her biri birer adet L298 sürücü kartına bağlanmıştır. Tablo 3.1'de motor kablolarının L298'e bağlantısı gösterilmiştir.

Tablo 3.1 18 pinli motor konnektörünün L298 sürücü kartlarına bağlanması

	L298 OUT1	L298 OUT2
Motor 1	3	4
Motor 2	7	8
Motor 3	11	12

3.1.2 Sürücü kartlarının Arduino'ya bağlanması

L298 motor sürücü kartlarının A kanallarına bağlı motorların kontrol edilebilmesi için her sürücü kartının 3'er adet pini kullanılır. Bu pinler; IN1, IN2 ve ENA'dır. ENA pinlerine Arduino'dan lojik 1 (5V) gönderildiğinde A kanalları kullanıma hazır hale gelmektedir. IN1 ve IN2 pinleri ise yön ve hız kontrolünün yapılmasını sağlar. Sürücü kartlarının IN1 pinlerine Arduino'nun PWM uçları bağlanmış ve PWM frekansları 4 KHZ'e set edilmiştir. Eklemlerin bulunduğu pozisyon ile referans pozisyonlar arasındaki hataların büyüklüğünü göre PWM sinyallerinin doluluk boşluk oranı (duty cycle) değiştirilerek hız kontrolü sağlanmaktadır. Burada tüm motorlar birbirinden farklı PWM sinyalleri ile sürülmektedir. Sürücü kartlarının IN2 pinlerine Arduino'dan lojik 1 bilgisi gönderilerek motorlar bir yönde döndürülürken, lojik 0 (0V) bilgisi gönderilerek diğer yönde döndürülmektedir. L298 sürücü kartlarının Arduino'ya bağlantı şeması Tablo 3.2'de gösterilmiştir.

Tablo 3.2 L298 - Arduino bağlantıları

	L298 ENA	L298 IN1	L298 IN2	L298 GND
1.Sürücü Kartı)	ARD 8	ARD 10	ARD 9	ARD GND
2.Sürücü Kartı)	ARD 7	ARD 5	ARD 4	ARD GND
3.Sürücü Kartı)	ARD 6	ARD 3	ARD 2	ARD GND

3.1.3 Potansiyometrelerin Arduino'ya bağlanması

Motorlardan geri besleme alarak uzuvların anlık pozisyonlarını bilmemizi sağlayan potansiyometreler, Arduino mikrodenetleyicisinin analog girişlerine bağlanarak okunur. Tablo 3.3'de bu bağlantı gösterilmiştir.

Tablo 3.3 Potansiyometre - Arduino bağlantısı

12 PINLI KONNEKTOR	ARDUINO
Motor 1 Pot. (1-Mavi)	A0
Motor 2 Pot. (2-Beyaz)	A1
Motor 3 Pot. (3-Turuncu)	A2
Potansiyometre Pozitif Beslemesi (9-10 Mavi)	+5V
Potansiyometre Negatif Beslemesi (7-Beyaz)	GND

Arduino'nun analog olarak okuduğu pozisyon bilgisi, Arduino içerisinde dijitalle çevrilerek 0-1024 aralığında bir değer olarak kullanıcıya gösterilir. Potansiyometre 0 k Ω iken 0 bilgisi gelirken, potansiyometre maksimum direnç değerine alındığında 1024 bilgisi gelir. Bu bilgilerin yorumlanması ve kullanılması, kodların açıklandığı bölümde anlatılacaktır.

Arduino kartı, USB port aracılığı ile bilgisayar üzerinden beslenmektedir. Aynı zamanda görüntü işleme ile nesnenin yeri tespit edildikten sonra ters kinematik analizi ile hesaplanan uzuv açıları, USB port üzerinden Arduino'ya gönderilmektedir.

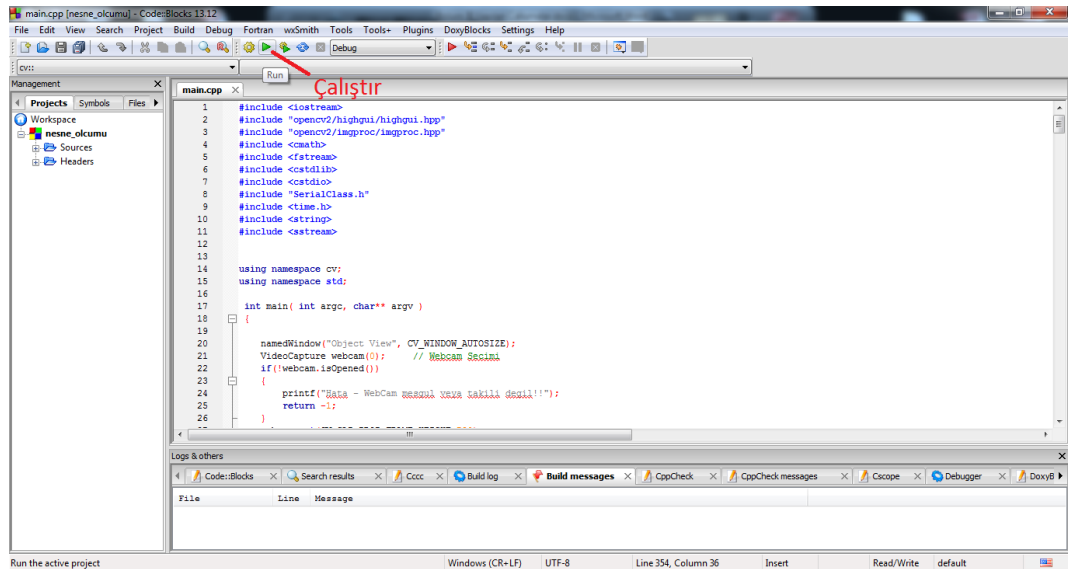
3.2 OpenCV ile Görüntü İşleme

Robotun çalışma uzayının tamamını tepe noktadan gören Logitech C270 HD kamera ile alınan görüntüler, aşağıda anlatılacak işlemlerden geçirilerek nesnelerin pozisyonları tespit edildi.

Görüntü işleme yazılımı, Code::Blocks geliştirme ortamında, OpenCV kütüphaneleri kullanılarak yazıldı.

3.2.1 Görüntü işleme yazılımının çalıştırılması

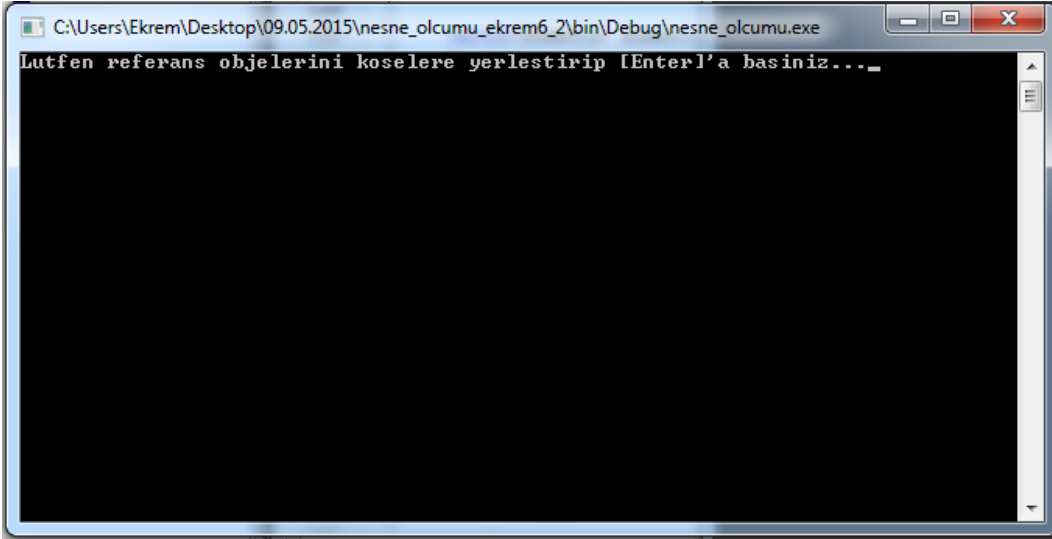
Code::Blocks ortamında, Şekil 3.1'de gösterilen "Run" butonuna basılarak program çalıştırılır.



Şekil 3.1 Görüntü işleme yazılımının çalıştırılması

3.2.2 Referans objelerinin yerleştirilmesi

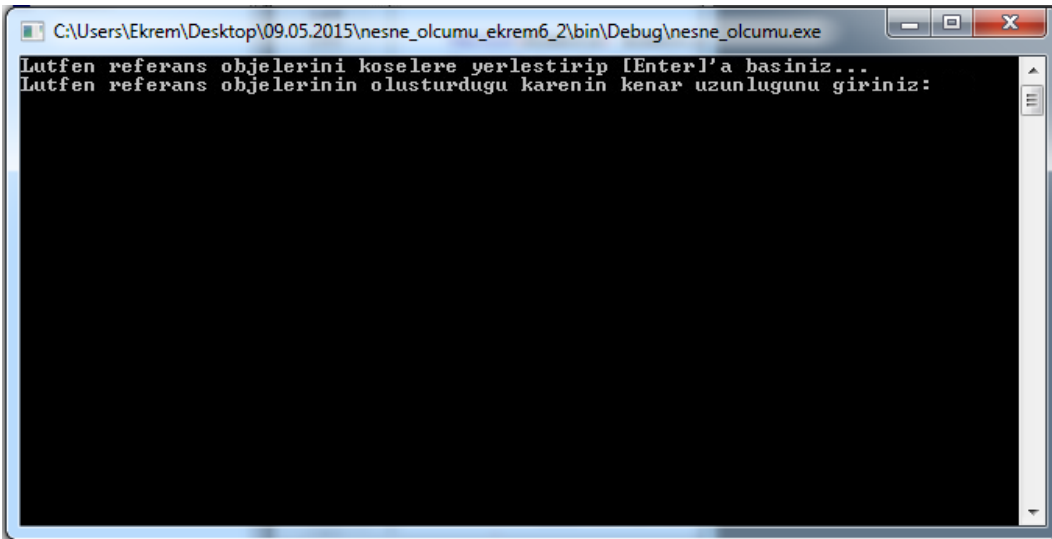
Program ilk çalıştırıldığında, ekranda Şekil 3.2'de gösterildiği gibi bir pencere açılır ve referans olarak kullanılacak objelerin yerleştirilmesi istenir. Bu çalışmada seçilen referans objeleri mavi renkli iki adet kapaktır. Bu objeler; robotun merkezinden eşit uzaklıkta ve çapraz bir şekilde yerleştirilir. Referans objeleri önceden yerleştirilmiş ise bu adım "Enter" tuşuna basılarak atlanır.



Şekil 3.2 Referans objelerinin yerleştirilmesi - 1

Referans objeler yerleştirildikten sonra "Enter" tuşuna basılır.

İkinci adım olarak Şekil 3.3'de gösterildiği gibi kullanıcıdan referans objelerinin oluşturduğu karenin bir kenar uzunluğu istenir. Yerleştirilen bu iki adet obje, bir karenin çapraz iki köşesinde gibi düşünülür ve bu karenin bir kenar uzunluğu programa girilir. Objelerin robotun çalışma uzayının dışına yerleştirilmesi daha uygun olacağı için karenin kenar uzunluğu genellikle 80 ya da 100 cm olarak seçilir.



Şekil 3.3 Referans objelerinin yerleştirilmesi - 2

3.2.3 Referans objelerinin ve taşınacak nesnelerin koordinatlarının tespiti

Referans objelerinin oluşturduğu karenin kenar uzunluğu yazılıp "Enter" tuşuna basıldığında, arka planda görüntü işleme adımları çalışmaya başlar. Bu adımlar sıra ile anlatılacaktır.

3.2.3.1 Görüntünün alınması

Öncelikle kamera aktif hale gelir ve 720x720 piksel çözünürlüğünde bir adet görüntü alır. Bu görüntünün alınması ve boyutlandırılması için yazılan kodlar şunlardır:

- Görüntünün alınması işleminden önce, yazılmış olan aşağıdaki kod yardımı ile kameradan alınacak görüntünün çözünürlüğü 720x720 piksele ayarlanır.

```
webcam.set(CV_CAP_PROP_FRAME_HEIGHT,720);
webcam.set(CV_CAP_PROP_FRAME_WIDTH,720);
```

- Kameradan webcam.read fonksiyonu ile görüntü alınır ve 'img' adlı matrisin içine atılır.

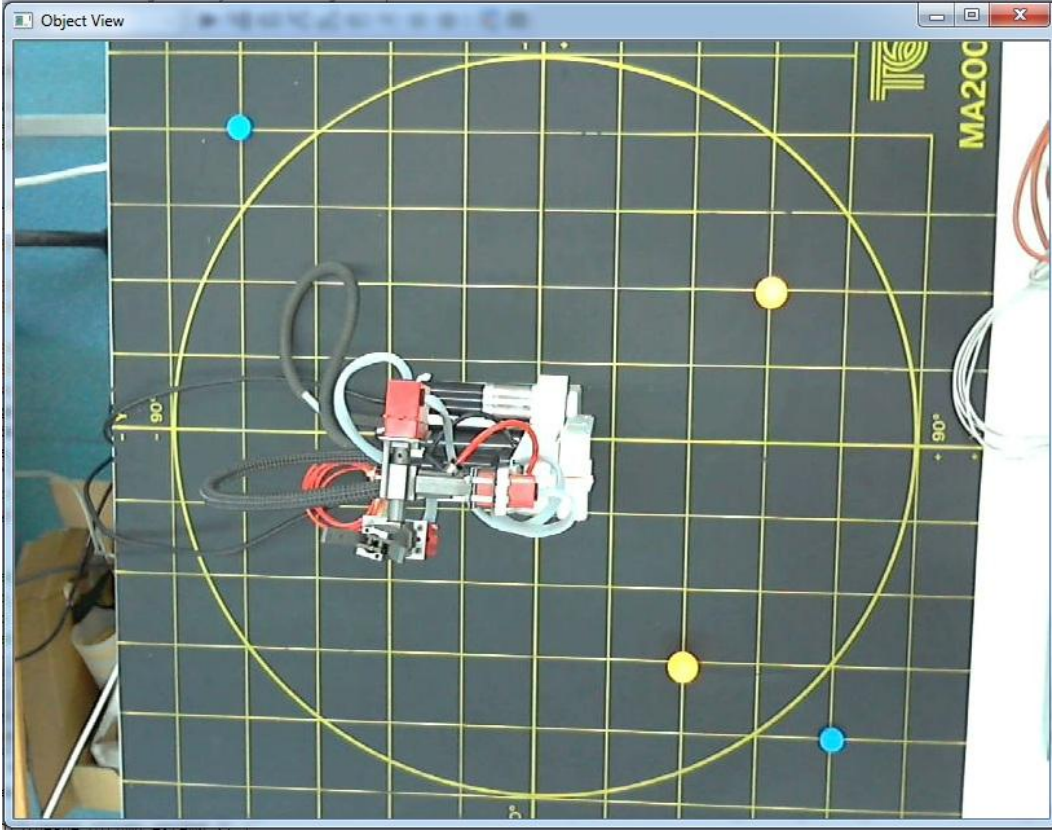
```
webcam.read(img);
```

webcam.read fonksiyonunun çalıştırılması ile kameradan alınan görüntü Şekil 3.4'de gösterilmiştir.

3.2.3.2 Görüntünün renk uzayı dönüşümü

OpenCV, renk uzayı olarak BGR formatını kullanmaktadır. Dolayısıyla kameradan alınan görüntü BGR formatındadır. Görüntü işleme tekniklerinde genellikle HSV renk uzayı tercih edilir. Bunun nedeni, HSV uzayında renklerin çok daha detaylı olarak tanımlanabilmesidir.

- H (Hue) değeri ile renk seçilir. OpenCV'de H parametresi 0-180 arası değer alır.



Şekil 3.4 Kamradan alınan orijinal görüntü

- S (saturation) ile rengin doygunluğu yani canlılığı seçilir. OpenCV'de S parametresi 0-255 arası değer alır.
- V (value) ile rengin parlaklığı seçilir. OpenCV'de V parametresi 0-255 arası değer alır.

Bu üç parametre ile tam olarak istenen renk, ton ve parlaklıktaki nesne görüntüden ayrıştırılabilir.

OpenCV'de renk dönüşümü işlemleri 'cvtColor' fonksiyonu ile yapılır. Bu fonksiyon şu şekilde kullanılır:

```
void cvtColor(src, dst, code)
```

- src: Giriş görüntüsü
- dst: Çıkış görüntüsü
- code: Dönüşüm yapılacak renk uzaylarıdır. Bazı renk dönüşümü kodları şunlardır:

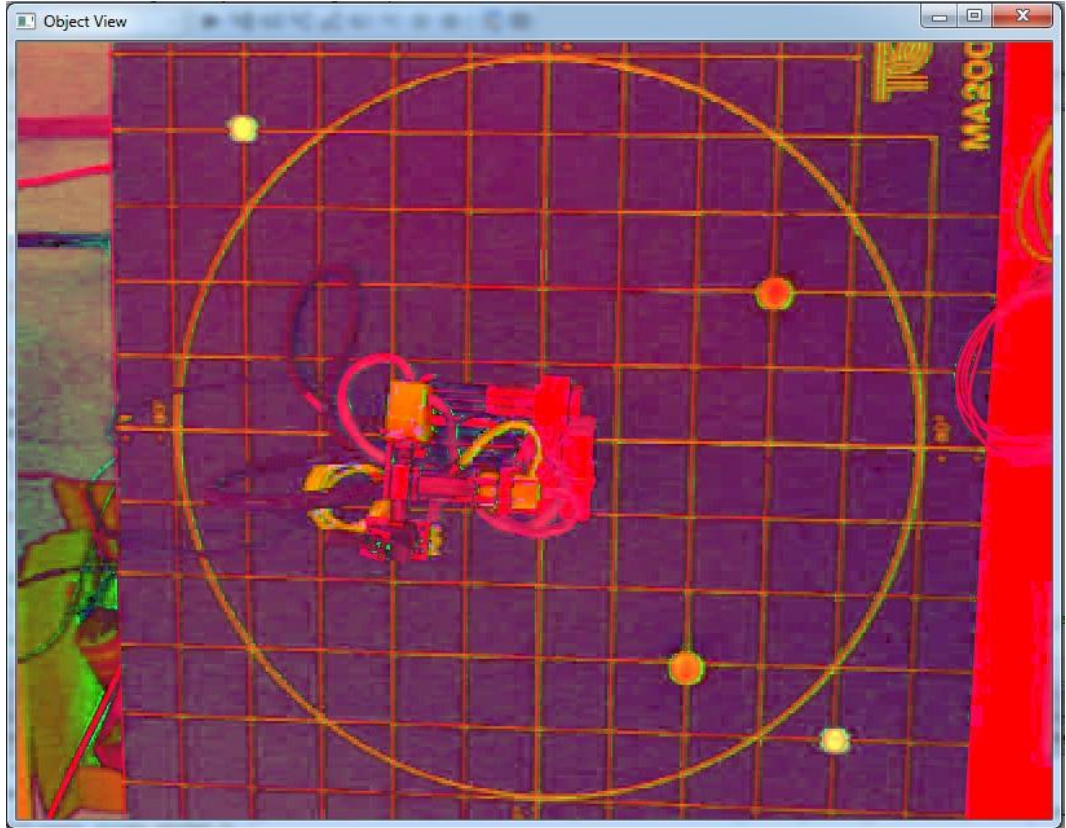
- RGB - GRAY renk dönüşümü için;
CV_BGR2GRAY, CV_RGB2GRAY, CV_GRAY2BGR, CV_GRAY2RGB
- RGB - CIE XYZ renk dönüşümü için;
CV_BGR2XYZ, CV_RGB2XYZ, CV_XYZ2BGR, CV_XYZ2RGB
- RGB - HSV renk dönüşümü için;
CV_BGR2HSV, CV_RGB2HSV, CV_HSV2BGR, CV_HSV2RGB

Bu fonksiyon çalışmada şu şekilde kullanılmıştır:

```
cvtColor(img,img_hsv,COLOR_BGR2HSV);
```

Burada 'img' dönüştürülecek olan BGR renk uzayındaki görüntü matrisi, 'img_hsv' ise dönüşüm sonucu oluşacak HSV renk uzayındaki görüntü matrisidir.

cvtColor fonksiyonunun çalıştırılması sonucu Şekil 3.4'de verilen görüntü, Şekil 3.5'deki yeni halini alır.



Şekil 3.5 HSV renk uzayına çevrilmiş görüntü

3.2.3.3 Görüntünün eşiklenmesi

Görüntünün HSV renk uzayına çevrilmesinden sonraki adım, görüntünün eşiklenmesi (threshold) adımıdır. Bu işlemin amacı, görüntüden istenilen özel bir bölümün ayrıştırılmasıdır. Bu çalışmada ayrıştırılması gereken iki özel bölüm vardır. Bunlar; referans olarak kullanılan objeler ve robotun taşınması istenen nesnelerdir. Bu nedenle iki farklı eşikleme işlemi yapılır.

Eşikleme işlemi H, S, V parametrelerine göre yapılır. Eşiklenecek olan nesnenin alt ve üst H,S,V değerleri denenerek bulunur.

OpenCV'de eşikleme işlemi "inRange" fonksiyonu ile yapılır. Bu fonksiyon;

```
void inRange(src, lowerb, upperb, dst)
```

şeklinde kullanılır. Burada;

- src: Giriş görüntüsü,
- lowerb: Alt H,S,V değerleri,
- upperb: Üst H,S,V değerleri,
- dst: Çıkış görüntüsüdür.

Referans objelerinin eşiklenmesi

Referans olarak kullanılan objeler için H,S,V değerleri Tablo 3.4'teki gibi bulunmuştur.

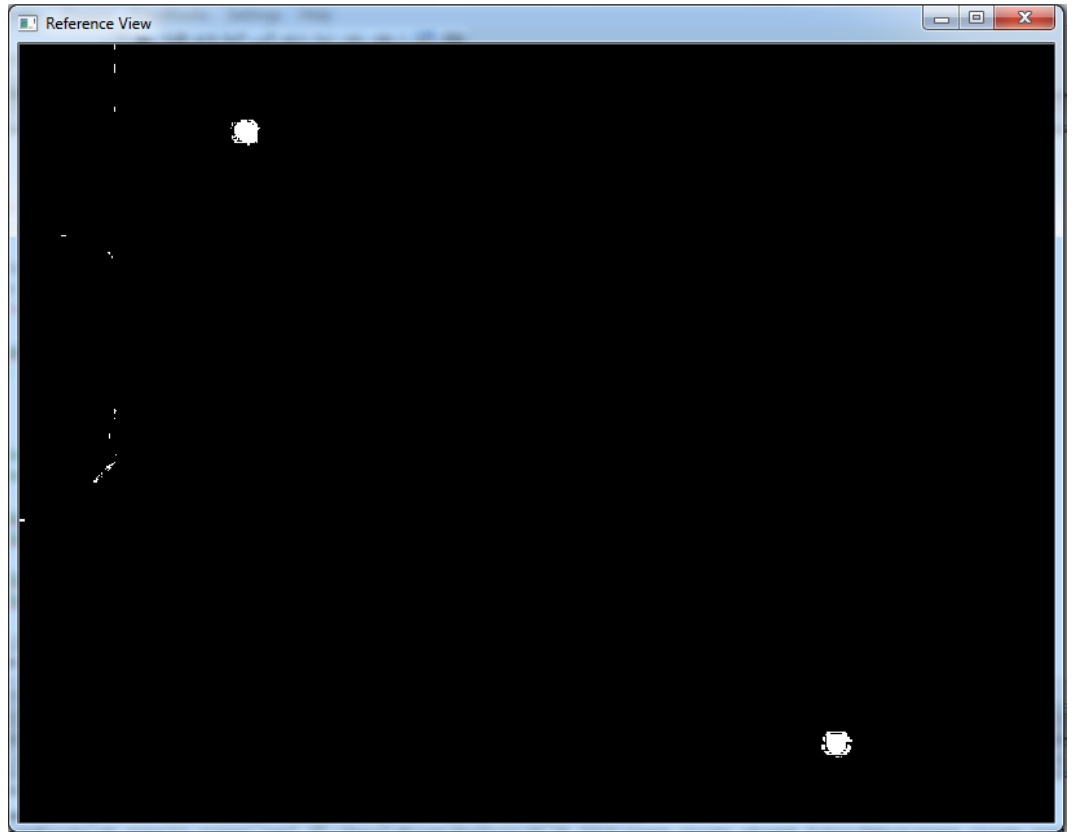
Tablo 3.4 Referans objeler için tespit edilen H,S,V değerleri

	Alt Değer	Üst Değer
H	90	146
S	160	255
V	47	255

inRange fonksiyonu, referans objelerinin eşiklenmesi için şu şekilde kullanılmıştır:

```
inRange(img_hsv, Scalar(H_ref_Alt, S_ref_Alt, V_ref_Alt), Scalar(H_ref_Ust, S_ref_Ust, V_ref_Ust), ref_thres_img);
```

inRange fonksiyonunun çalıştırılması sonucu Şekil 3.5'de verilen görüntü, Şekil 3.6'daki yeni halini alır.



Şekil 3.6 Referans objelerinin eşiklenmesi

Taşıyacak nesnelerin eşiklenmesi

Robot tarafından taşıyacak nesneler için H,S,V değerleri Tablo 3.5'teki gibi bulunmuştur.

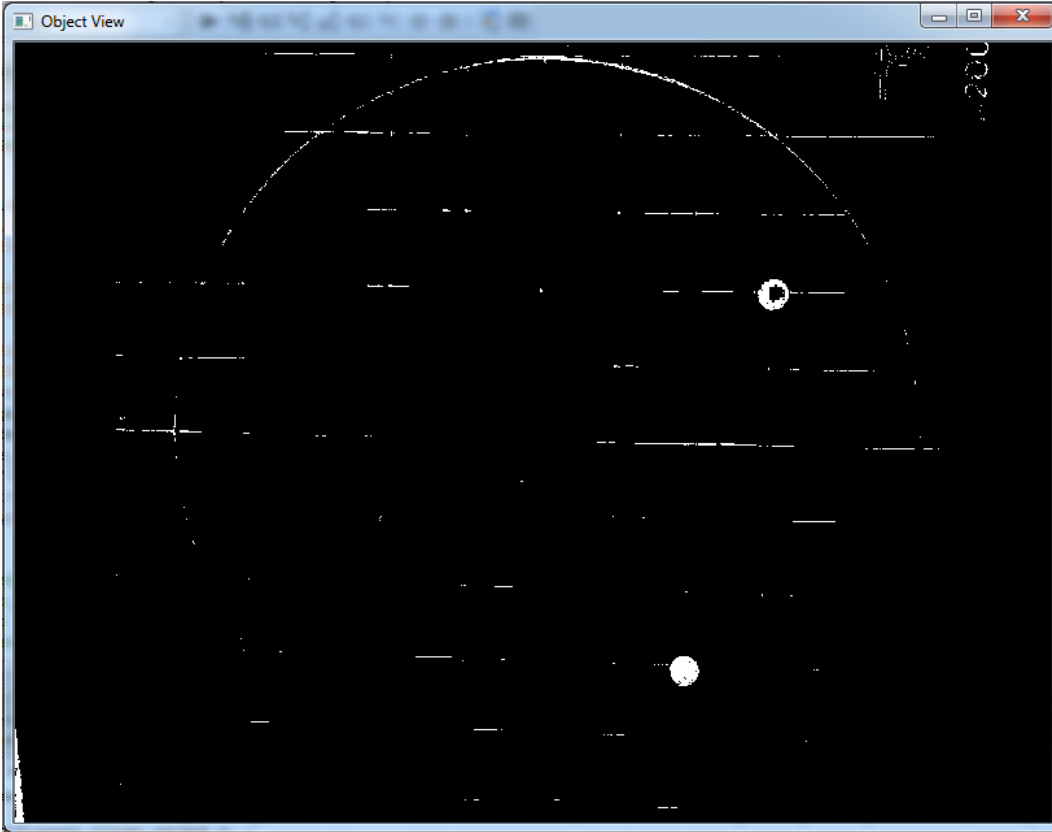
InRange fonksiyonu, taşıyacak nesnelerin eşiklenmesi için şu şekilde kullanılmıştır:

```
inRange(img_hsv,Scalar(H_Alt,S_Alt,V_Alt),Scalar(H_Ust,S_Ust,V_Ust), thres_img);
```

Tablo 3.5 Taşınacak nesneler için tespit edilen H,S,V değerleri

	Alt Değer	Üst Değer
H	19	31
S	55	255
V	243	255

inRange fonksiyonun çalıştırılması sonucu Şekil 3.5'de verilen görüntü, Şekil 3.7'deki yeni halini alır.



Şekil 3.7 Taşınacak nesnelerin eşiklenmesi

Eşikleme işlemi sonucunda elde edilen her iki görüntü de gürültü içermektedir. Bu gürültünün giderilmesi için filtreleme işlemi yapılmalıdır.

3.2.3.4 Görüntünün filtrelenmesi

Eşikleme işlemi için fonksiyona verilen, ayrıştırılmak istenen nesneye ait alt ve üst H,S,V değer aralığına giren tüm nesneler, eşiklemeden sonra yeni görüntüde ortaya çıkmaktadır. Bu nesnelerden bazıları istenen, bazıları ise istenmeyen nesnelerdir. Bu istenmeyen nesnelere gürültü adı verilir. Gürültülerin görüntüden yok edilmesi gerekmektedir. Bunun için çeşitli fonksiyonlar bulunmaktadır. Bu çalışmada "erode (erozyon)" fonksiyonu kullanılmıştır.

OpenCV'de erode fonksiyonu;

```
void erode(src, dst, element)
```

şeklinde kullanılır. Burada;

- src: Giriş görüntüsü
- dst: Çıkış görüntüsü
- element: Yapılacak olan erozyonun özelliklerinin verildiği matristir. Değeri değiştirilmezse başlangıçta 3x3 olarak tanımlıdır. "getStructuringElement" komutu ile özellikleri değiştirilir.

getStructuringElement komutu;

```
getStructuringElement(erosion_type, Size(2*erosion_size+1, 2*erosion_size+1),  
Point(erosion_size, erosion_size ) );
```

şeklinde kullanılır. Burada;

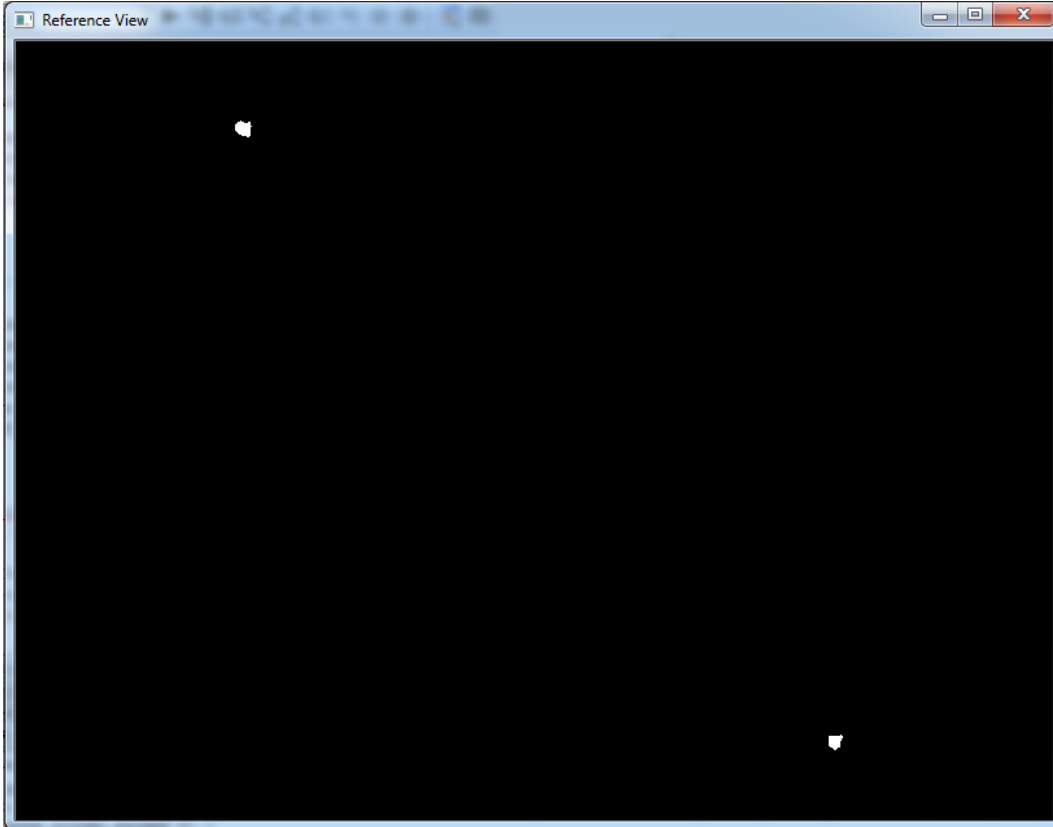
- erosion_type: Erozyonun tipidir. Aşağıdaki dikdörtgen, çapraz ya da elips şekillerinden biri seçilir.
 - MORPH_RECT
 - MORPH_CROSS
 - MORPH_ELLIPSE
- size: Erozyon matrisinin boyutudur. Değer büyüdükçe aşındırma artar.
- point: Erozyonun dayanak noktasıdır.

Referans objelerin filtrelenmesi

erode fonksiyonu referans objelerin filtrelenmesinde şu şekilde kullanılmıştır:

```
int ref_erosion_size = 3;
Mat ref_element = getStructuringElement( MORPH_ELLIPSE, Size(
2*ref_erosion_size + 1, 2*ref_erosion_size+1 ), Point( ref_erosion_size,
ref_erosion_size ) );
erode(ref_thres_img,ref_filtered_img,ref_element);
```

erode fonksiyonun çalıştırılması sonucu Şekil 3.6'da verilen görüntü, Şekil 3.8'deki yeni halini alır.



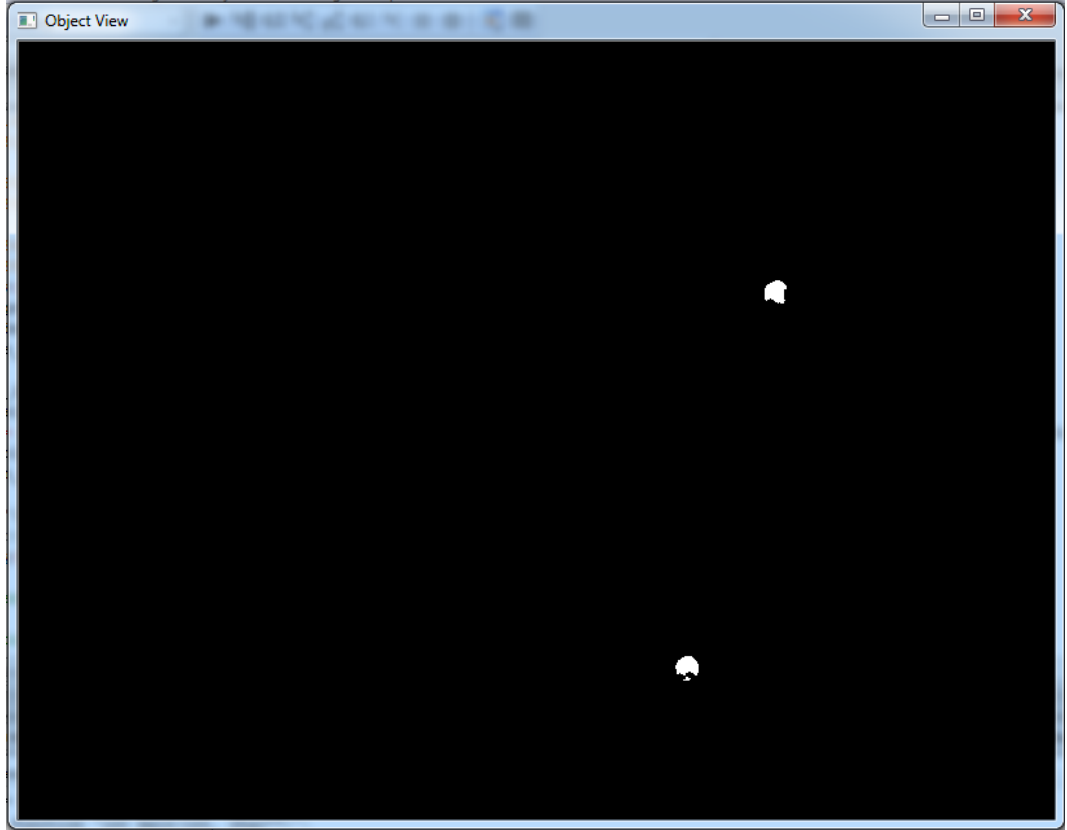
Şekil 3.8 Referans objelerinin filtrelenmesi

Taşınacak nesnelerin filtrelenmesi

erode fonksiyonu taşınacak nesnelerin filtrelenmesinde şu şekilde kullanılmıştır:

```
int erosion_size = 3;
Mat element = getStructuringElement( MORPH_ELLIPSE, Size( 2*erosion_size
+ 1, 2*erosion_size+1 ), Point( erosion_size, erosion_size ) );
erode(thres_img,filtered_img,element);
```

erode fonksiyonun çalıştırılması sonucu Şekil 3.7'de verilen görüntü, Şekil 3.9'daki yeni halini alır.



Şekil 3.9 Taşınacak nesnelerin filtrelenmesi

Filtreleme işlemi ile elde edilen görüntülerde görüldüğü gibi referans objeler ve taşınacak nesneler çok net elde edilmiştir. Sırada bu nesnelerin kenarlarının tespit edilmesi işlemi vardır.

3.2.3.5 Kenarların tespit edilmesi

OpenCV'de kenar tespiti için kullanılabilecek fonksiyonlardan bir tanesi "Canny" fonksiyonudur. Canny fonksiyonu başarılı sonuçlar verdiği için bu çalışmada kullanılmıştır.

OpenCV'de Canny fonksiyonu şu şekilde kullanılır:

```
void Canny(src, dst, lowThreshold, highThreshold, aperture_size)
```

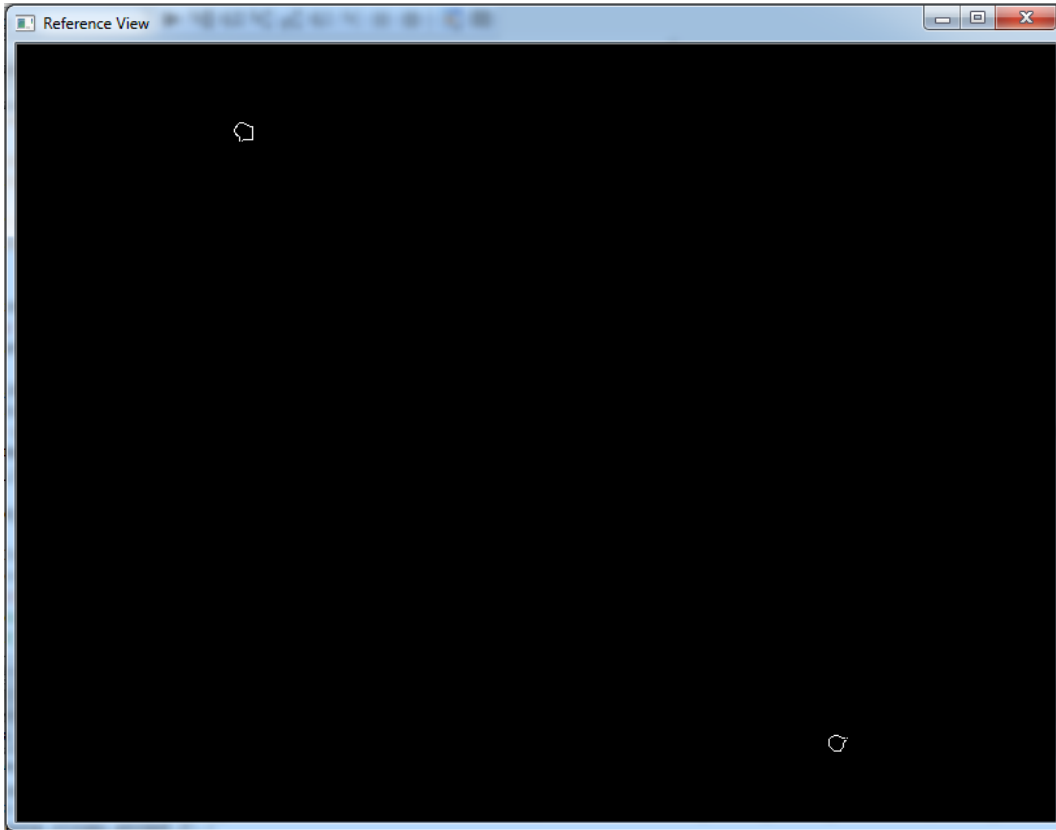
- src: Gri-düzey haldeki giriş görüntüsü
- dst: Çıkış görüntüsü
- lowThreshold: Hysteresis alt eşik değeri
- highThreshold: Hysteresis üst eşik değeri
- aperture_size: Sobel operatörü için açıklık değeridir.

Referans objelerinin kenarlarının tespit edilmesi

Canny fonksiyonu referans objelerinin kenarlarının tespit edilmesinde şu şekilde kullanılmıştır:

```
Canny(ref_filtered_img, ref_algilanan_kenarlar, 60, 255);
```

Canny fonksiyonun çalıştırılması sonucu Şekil 3.8'de verilen görüntü, Şekil 3.10'daki yeni halini alır.



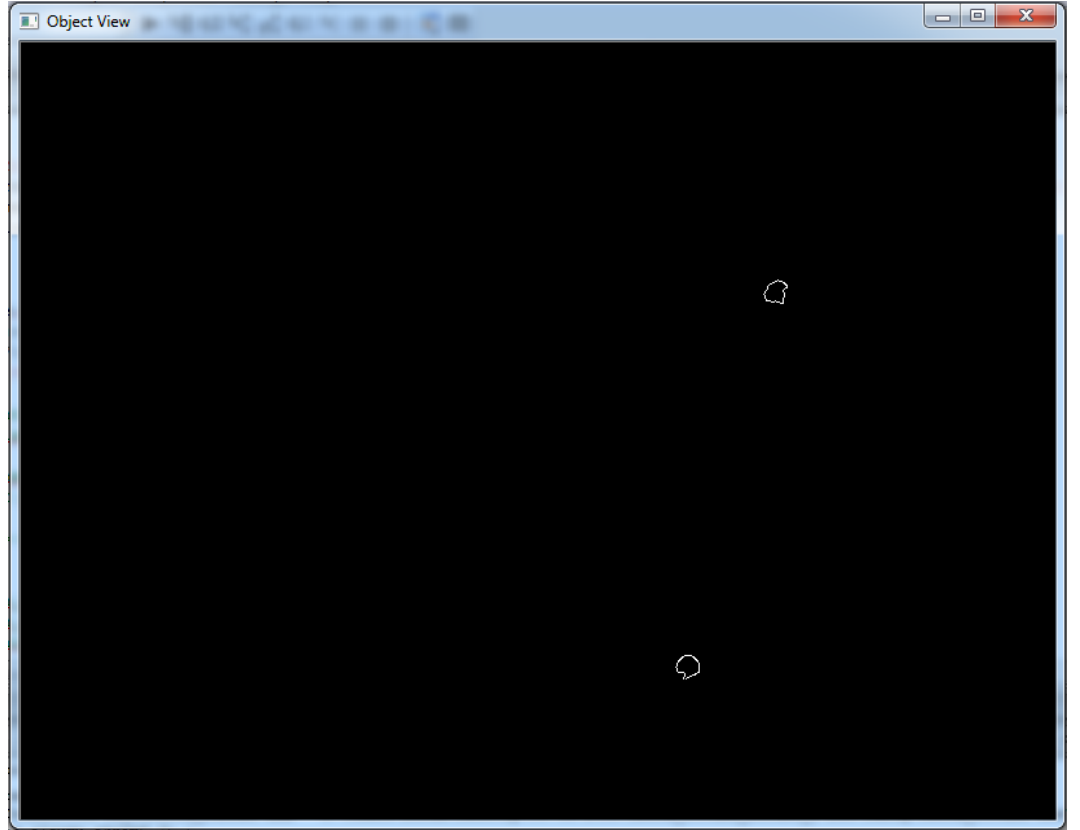
Şekil 3.10 Referans objelerinin kenarlarının tespiti

Taşınacak nesnelerin kenarlarının tespit edilmesi

Canny fonksiyonu taşınacak nesnelerin kenarlarının tespit edilmesinde şu şekilde kullanılmıştır:

```
Canny(filtered_img,algılanan_kenarlar,60,255);
```

Canny fonksiyonun çalıştırılması sonucu Şekil 3.9'da verilen görüntü, Şekil 3.11'deki yeni halini alır.



Şekil 3.11 Taşınacak nesnelerin kenarlarının tespiti

Canny fonksiyonu ile referans objelerinin ve taşınacak nesnelerin kenarları tespit edilmiştir. Artık son adım olarak, orta noktaların bulunması kalmıştır.

3.2.3.6 Orta noktaların tespit edilmesi

Orta noktaların tespit edilebilmesi için öncelikle çevre vektörlerinin (contour vectors) elde edilmesi gerekmektedir. OpenCV'de çevre vektörleri 'findContours' fonksiyonu ile bulunur. Bu fonksiyon;

```
void findContours(image, contours, mode, method, offset);
```

şeklinde kullanılır. Burada;

- image: İkili görüntü tipindeki giriş resmi. Genellikle Canny fonksiyonundan çıkan görüntüdür.
- contours: Tespit edilen çevre vektörleri çıkış olarak verilir.
- mode: Çevre alım modu seçimi yapılır.
 - CV_RETR_EXTERNAL,
 - CV_RETR_LIST,
 - CV_RETR_CCOMP,
 - CV_RETR_TREE olmak üzere 4 tane mod vardır.
- method: Çevre yaklaşım methodu seçimi yapılır.
 - CV_CHAIN_APPROX_NONE,
 - CV_CHAIN_APPROX_SIMPLE,
 - CV_CHAIN_APPROX_TC89_L1,
 - CV_CHAIN_APPROX_TC89_KCOS olmak üzere 4 tane method vardır.
- offset: Bütün çevre vektörlerinin girilen miktar kadar kaydırılmasını sağlar.

Çevre vektörlerinin bulunmasından sonra sırada moment hesabı vardır. Bunun için örnek bir kod;

```
vector<Moments> mu(contours.size() );
for( int i = 0; i < contours.size(); i++ )
{ mu[i] = moments( contours[i], false ); }
```

şeklindedir.

Moment hesabından sonra sırada ağırlık merkezi tespiti ve ağırlık merkezine nokta konulması vardır. Bunun için örnek bir kod;

```
vector<Point2f> mc( contours.size() );
for( int i = 0; i < contours.size(); i++ )
{ mc[i] = Point2f( mu[i].m10/mu[i].m00 , mu[i].m01/mu[i].m00 ); }
```

şeklindedir.

Referans objelerinin orta noktalarının tespit edilmesi

Referans objelerinin çevre vektörlerinin, momentlerinin ve ağırlık merkezlerinin tespiti ve ağırlık merkezlerine nokta koyma işlemi aşağıda verilen kodlar ile gerçekleştirilmiştir.

```
vector<vector<Point>>ref_algilanan_kontourler;
findContours(ref_algilanan_kenarlar,ref_algilanan_kontourler,CV_RETR_TREE,
CV_CHAIN_APPROX_SIMPLE,Point(0,0));

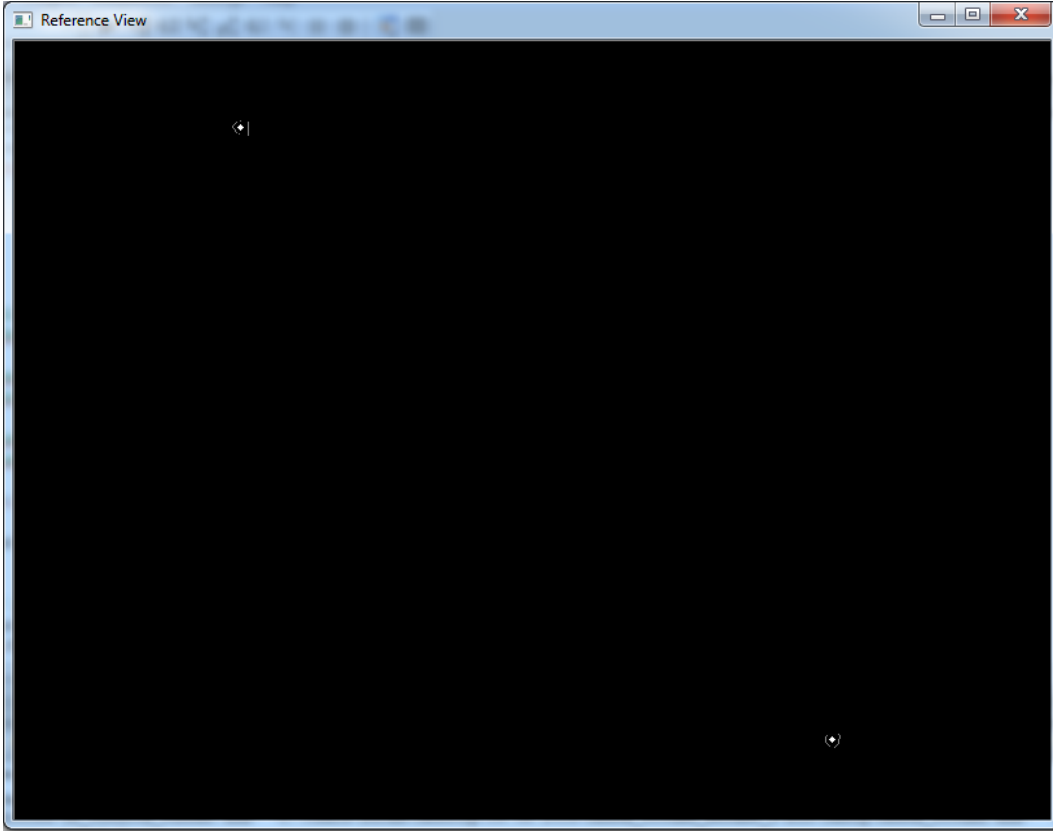
vector<Moments> ref_momentler(ref_algilanan_kontourler.size());
for(int i = 0; i < ref_algilanan_kontourler.size(); i++ )
{
    ref_momentler[i] = moments(ref_algilanan_kontourler[i]);
}
int ref_x[ref_algilanan_kontourler.size()];
int ref_y[ref_algilanan_kontourler.size()];

Point2d ref_noktalar[ref_algilanan_kontourler.size()/2];
for (int i=0; i < ref_algilanan_kontourler.size(); i++)
{
    ref_x[i] = ref_momentler[i].m10/ref_momentler[i].m00; // m10 = x
    ref_y[i] = ref_momentler[i].m01/ref_momentler[i].m00 // m01 = y
}

for(int i = 0; i < ref_algilanan_kontourler.size(); i++)
{
    int ref_noktasayisi = i/2;
    int ref_koordinat_noktalari = 2*ref_noktasayisi;
    ref_noktalar[ref_noktasayisi] = Point2d(ref_x[ref_koordinat_noktalari],
ref_y[ref_koordinat_noktalari]);

    line(ref_algilanan_kenarlar,ref_noktalar[ref_noktasayisi],ref_noktalar[ref_noktasayisi],Scalar(255,255,255),4); // merkez çiz
}
```

Bu kodların çalıştırılması ile Şekil 3.10'da verilen görüntü, Şekil 3.12'deki yeni halini alır.



Şekil 3.12 Referans objelerinin orta noktalarının tespit edilmesi

Taşıyacak nesnelerin orta noktalarının tespit edilmesi

Taşıyacak nesnelerin çevre vektörlerinin, momentlerinin ve ağırlık merkezlerinin tespiti ve ağırlık merkezlerine nokta koyma işlemi aşağıda verilen kodlar ile gerçekleştirilmiştir.

```
vector<vector<Point>> algilanan_kontourler;
findContours(algilanan_kenarlar,algilanan_kontourler,CV_RETR_TREE,CV_CHAIN_APPROX_SIMPLE,Point(0,0));

vector<Moments> momentler(algilanan_kontourler.size());
for(int i = 0; i < algilanan_kontourler.size(); i++ )
{
    momentler[i] = moments(algilanan_kontourler[i]);
}
double x[algilanan_kontourler.size()];
double y[algilanan_kontourler.size()];

Point2d noktalar[algilanan_kontourler.size()/2];
for int(int i=0; i < algilanan_kontourler.size(); i++)
{
    x[i] = momentler[i].m10/momentler[i].m00; // m10 = x
```

```

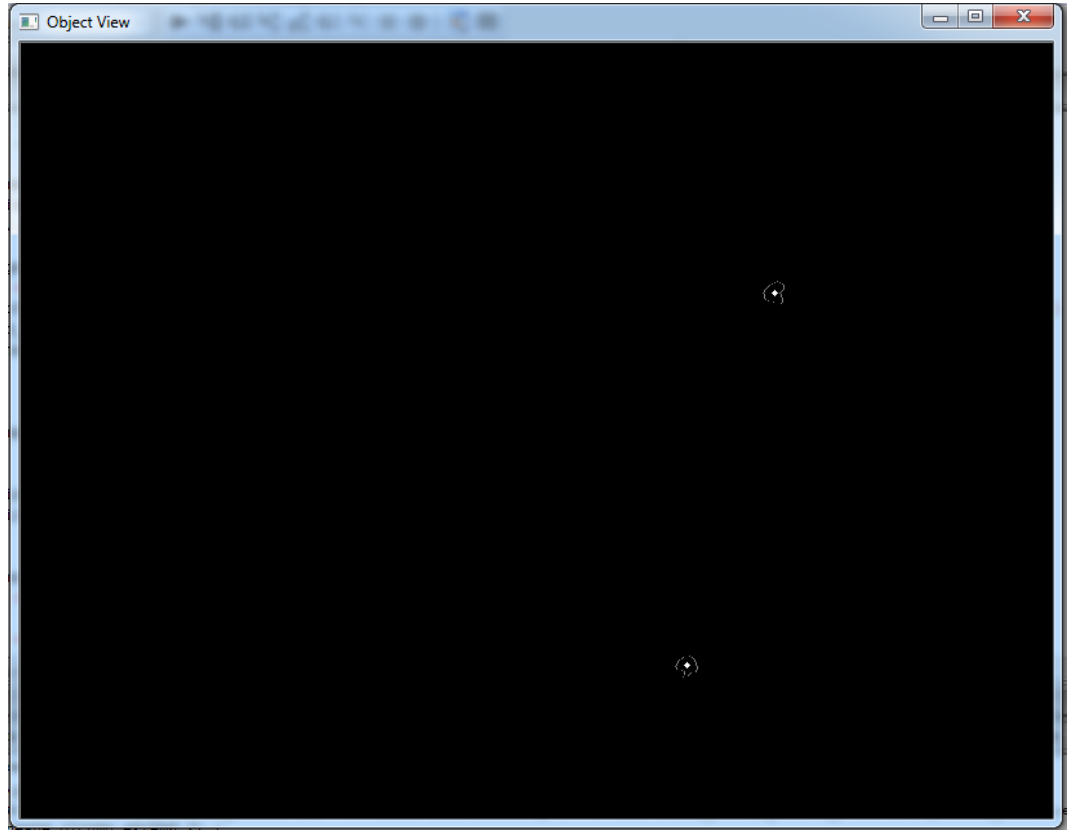
    y[i] = momentler[i].m01/momentler[i].m00 // m01 = y
}

for(int i = 0; i < algilanan_kontourler.size(); i++)
{
    int noktasayisi = i/2;
    int koordinat_noktalari = 2*noktasayisi;
    noktalar[noktasayisi] =
Point2d(x[koordinat_noktalari],y[koordinat_noktalari]);

line(algilanan_kenarlar,noktalar[noktasayisi],noktalar[noktasayisi],Scalar(255,255
,255),4); // merkez çiz
}

```

Bu kodların çalıştırılması ile Şekil 3.11'de verilen görüntü, Şekil 3.13'deki yeni halini alır.



Şekil 3.13 Taşınacak nesnelerin orta noktalarının tespit edilmesi

Arka planda gerçekleşen görüntü işleme adımları bunlardır. Bu adımlar yaklaşık olarak 1 saniyede tamamlanmaktadır.

3.2.4 Tespit edilen koordinatlarının ekrana basılması

Şekil 3.3'te kullanıcıdan referans objelerinin oluşturduğu karenin kenar uzunluğu istenmişti. Bu bilgi girilip 'Enter' tuşuna basıldığında, arka planda Bölüm 3.2.3'de verilen adımlar çalışır. Bu adımlar tamamlandığında ekrana Şekil 3.14'de gösterildiği gibi referans objelerinin, taşınacak nesnelerin ve robotun merkezinin piksel olarak koordinatları basılır.

```

C:\Users\Ekrem\Desktop\09.05.2015\nesne_olcumu_ekrem6_2\bin\Debug\nesne_olcumu.exe
Lutfen referans objelerini koselere yerlestirip [Enter]'a basiniz...
Lutfen referans objelerinin olusturdugu karenin kenar uzunlugunu giriniz: 80

0.Referans:
X0: 538      Y0: 631
1.Referans:
X1: 66       Y1: 174

Robot merkez (x,y): (302,402)
Hesaplanan 1 piksel uzunlugu: 0.172206cm

0.Nesne:
X0: 478      Y0: 517.333
1.Nesne:
X1: 193.188  Y1: 583.539

0.Nesne x,y : 30.3082 cm , 19.8611 cm
1.Nesne x,y : -18.738 cm , 31.2621 cm

```

Şekil 3.14 Tespit edilen koordinatların ekrana basılması

Kullanıcının girmiş olduğu kenar uzunluğu bilgisi kullanılarak piksel uzayından gerçek uzaya geçiş yapılır. Burada kullanılan yöntem, koordinat düzleminde herhangi iki nokta arasındaki uzaklık;

$$|AB| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (3.1)$$

formülü ile bulunur.

Şekil 3.14'deki değerler denklem 3.1'de yerine koyarsak;

$$|AB| = \sqrt{(538 - 66)^2 + (631 - 174)^2}$$

olarak bulunur. Bu iki referans objesinin orta noktaları arasındaki piksel olarak uzaklık miktarıdır.

Kullanıcının girdiği değer, karenin bir kenar uzunluğunun değeri idi. Buna göre karenin köşegen uzunluğu $= 80\sqrt{2}$ olarak bulunur. Buna göre bir pikselin karşılık geldiği uzunluk değeri;

$$\frac{80\sqrt{2}}{\sqrt{(538-66)^2 + (631-174)^2}} = 1 \text{ piksel uzunluğu} = 0.172206 \text{ cm}$$

olarak bulunur.

1 pikselin gerçek uzunluğu ve robotun merkez noktası bilindiğine göre, artık taşınacak nesnelerin robotun merkezine olan uzaklıkları hesaplanabilir. Şekil 3.14'de görüldüğü gibi önce piksel değeri olarak bulunan taşınacak nesnelerin koordinatları, daha sonra cm olarak hesaplanmış ve ekrana basılmıştır. Bu hesaplamaları gösterelim.

- 0. nesne olarak belirtilen nesnenin x ve y yönlerinde robotun merkezine olan uzaklığı

x yönünde $478-302 = 176$ piksel,

y yönünde $517 - 402 = 115$ pikseldir.

Her bir pikselin uzunluğu 0,172206 cm olarak bulunmuştu. Buna göre;

0. nesnenin robotun merkezine x yönünde uzaklığı $= 176*0.172206 = 30.3$ cm,

0. nesnenin robotun merkezine y yönünde uzaklığı $= 115*0.172206 = 19.8$ cm

olarak bulunur.

- Benzer şekilde 1. nesne olarak belirtilen nesnenin x ve y yönlerinde robotun merkezine olan uzaklığı

x yönünde $(193-302)*0.172206 = -18.7$ cm

y yönünde $(583 - 402)*0.172206 = 31.2$ cm

olarak bulunur.

Bu hesaplamaların doğru bir şekilde yapılabilmesi için kameradan alınan görüntüden referans objelerinin her ikisinin de net bir şekilde algılanması gerekmektedir. Ancak bazı durumlarda bu her seferinde mümkün olmamaktadır. Referans objeler için H,S,V değerleri belirlenirkenki ışık yoğunluğu her zaman aynı kalmamaktadır ve ışık yoğunluğu görüntü işlemede çok önemli bir etkidir.

Bu nedenle kullanıcıya, referans objelerinin ve taşınacak nesnelerin doğru algılanıp algılanmadığı sorusu Şekil 3.15'de gösterildiği gibi soruldu. Kullanıcı E ya da H harflerinden birini yazarak cevabını belirtir.

```

C:\Users\Ekrem\Desktop\09.05.2015\nesne_olcumu_ekrem6_2\bin\Debug\nesne_olcumu.exe
Lutfen referans objelerini koselere yerlestirip [Enter]'a basiniz...
Lutfen referans objelerinin olusturdugu karenin kenar uzunlugunu giriniz: 80
0.Referans:
X0: 538      Y0: 631
1.Referans:
X1: 66       Y1: 174
Robot merkez (x,y): (302,402)
Hesaplanan 1 piksel uzunlugu: 0.172206cm
0.Nesne:
X0: 478      Y0: 517.333
1.Nesne:
X1: 193.188  Y1: 583.539
0.Nesne x,y : 30.3082 cm , 19.8611 cm
1.Nesne x,y : -18.738 cm , 31.2621 cm
Nesne dogru algilandi mi? [E/H]

```

Şekil 3.15 Nesnelerin doğru algılandığı kontrolü

Kullanıcı H tuşuna basarak doğru algılanmadığını belirtirse, program otomatik olarak kendini kapatır. Kullanıcı E tuşuna basarak nesnelerin doğru algılandığını belirtirse, taşınacak olan nesneler için ters kinematik analizi yapılır ve bulunan uzuv açı değerleri USB port üzerinden Arduino'ya gönderilir. Şekil 3.16'da tespit edilen iki nesne için ters kinematik ile hesaplanmış uzuv açıları gösterilmiştir.

Burada her iki nesne için de 4 adet çözüm kümesi olduğuna dikkat edelim. Bu çözüm kümelerinden bazıları, fiziksel olarak sisteme uymamaktadır. Bunların ayrıştırılması ve en doğru çözüm kümesinin bulunması ileriki bölümlerde anlatılacaktır.

```

C:\Users\Ekrem\Desktop\09.05.2015\nesne_olcumu_ekrem6_2\bin\Debug\nesne_olcumu.exe
Lutfen referans objelerini koselere yerlestirip [Enter]'a basiniz...
Lutfen referans objelerinin olusturdugu karenin kenar uzunlugunu giriniz: 80

0.Referans:
X0: 538      Y0: 631
1.Referans:
X1: 66       Y1: 174

Robot merkez (x,y): (302.402)
Hesaplanan 1 piksel uzunlugu: 0.172206cm

0.Nesne:
X0: 478      Y0: 517.333
1.Nesne:
X1: 193.188   Y1: 583.539

0.Nesne x,y : 30.3082 cm , 19.8611 cm
1.Nesne x,y : -18.738 cm , 31.2621 cm
Nesne dogru algilandi mi? [E/H] E

Tum Cozum Kumeleri 0.Nesne =
39.5747 76.4722 70.6885
39.5747 170.887 -86.698
-153.101 -170.887 70.6885
-153.101 -76.4722 -86.698

En Dogru Cozum Icin Aci Degerleri 0.Nesne : 39.5747 76.4722 70.6885

Tum Cozum Kumeleri 1.Nesne =
127.239 76.6907 70.1025
127.239 170.358 -86.112
-65.3622 -170.358 70.1025
-65.3622 -76.6907 -86.112

En Dogru Cozum Icin Aci Degerleri 1.Nesne : 127.239 76.6907 70.1025

Gonderilen Aci Degerleri 0.Nesne : 39.5747 76.4722 70.6885
Gonderilen Aci Degerleri 1.Nesne : 127.239 76.6907 70.1025

Process returned 0 (0x0)   execution time : 48.518 s
Press any key to continue.

```

Şekil 3.16 Ters kinematik analizi sonucu çıkan açı değerleri

En doğru çözüm için bulunan açı değerleri Arduino'ya gönderilir ve bu program sona erer. Buradan sonra iş Arduino'nun robotu kontrol etmesine kalmıştır. Ancak bu konuya geçmeden önce, TQ MA2000 robot manipülatörü için ileri ve ters kinematik analizlerinin nasıl yapıldığını anlatalım.

3.3 TQ MA2000 Robot Manipülatörünün İleri ve Ters Kinematik Analizi

İleri kinematik analizi; bir robot manipülatördeki eklem açılarının ve link uzunluklarının bilinmesi durumunda, robotun uç işlevcisinin alacağı konum ve oryantasyonun hesaplanması işlemidir.

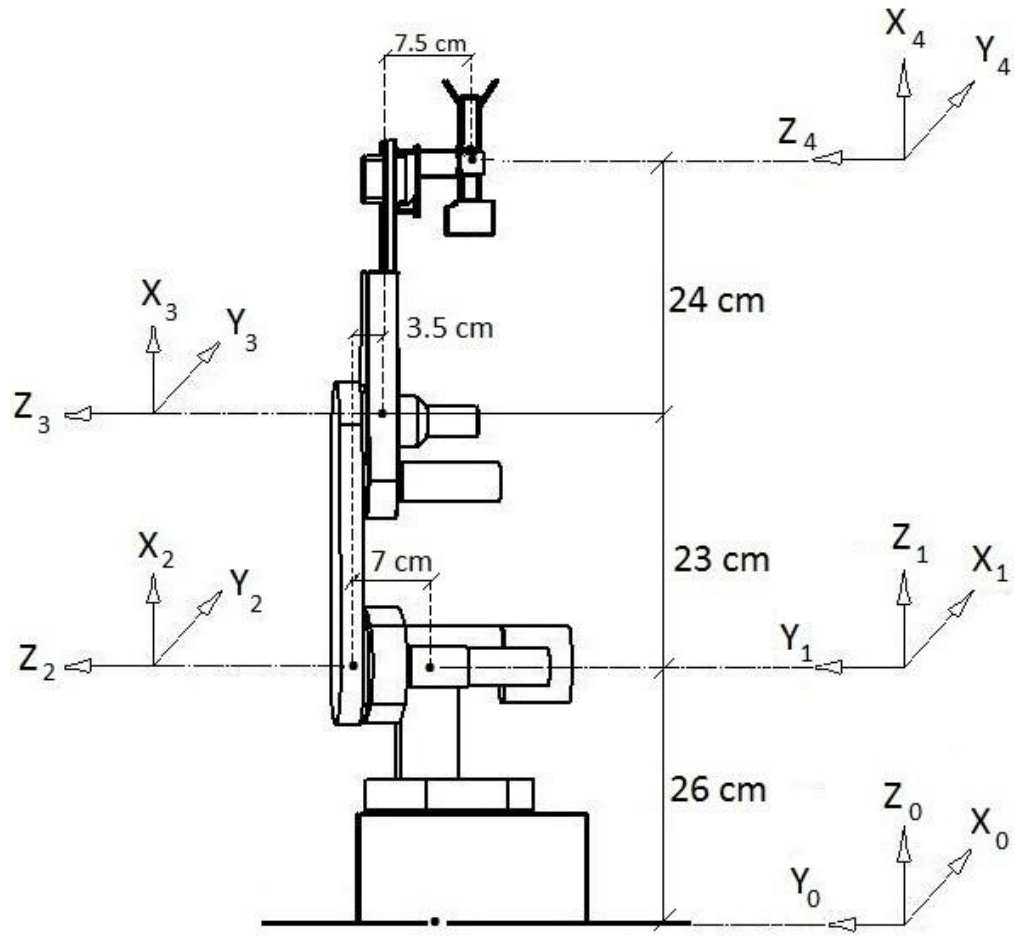
Ters kinematik analizi; kartezyen uzayda, robotun ana çerçevesine göre verilen uç işlevcinin konum ve yönelim verileri yardımıyla eklem değişkenlerinin belirlenmesi işlemidir.

3.3.1 TQ MA2000 robot manipülatörünün ileri kinematik analizi

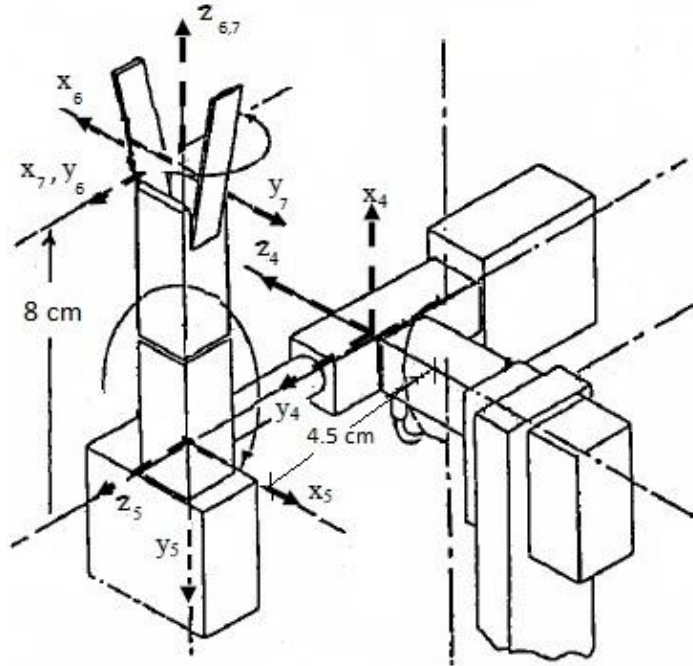
Çalışmada kullanılan TQ MA2000 robot manipülatörü 6 ekleme sahiptir. Bu eklemlerden ilk üçü majör eklemdir (bel, omuz, dirsek) ve uygulamada bunlar kullanılmıştır. Son üç eklem ise bilek kısmıdır. Uygulamada bilek kullanılmadığı halde ileriki çalışmalar da düşünülerek ileri kinematik analizi yapılmıştır. İleri kinematik analizinin yapılmasında Denavit - Hartenberg yöntemi kullanılmıştır. Bu yöntem Bölüm 2.2.9.2'de anlatılmıştı.

DH yöntemini adım adım uygulayalım.

1. Tüm eklemlere koordinat sistemlerinin yerleştirilmesi ve numaralandırılması yapılır. Şekil 3.17 ve 3.18'de TQ MA2000 robot manipülatörüne yerleştirilen koordinat sistemleri gösterilmiştir.



Şekil 3.17 TQ MA2000 robot manipulatörünün majör eklemlerine koordinat sistemlerinin yerleştirilmesi



Şekil 3.18 TQ MA2000 robot manipulatörünün bileğine koordinat sistemlerinin yerleştirilmesi

2. Koordinat sistemleri yerleştirildikten sonra, sırada DH değişkenlerinin bulunması ve bu değişkenler kullanılarak DH tablosunun oluşturulması vardır. Tablo 3.6'da oluşturulan DH tablosu görülmektedir.

Tablo 3.6 TQ MA2000 robot manipülatörü için oluşturulan DH tablosu

Eksen No	D-H Değişkenleri				i. Eklem Değişkeni
i	a_{i-1}	α_{i-1}	d_i	θ_i	d_i veya θ_i
1	0	0	26	θ_1	θ_1
2	0	$-\pi/2$	7	$\theta_2 - \pi/2$	θ_2
3	23	0	-3.5	θ_3	θ_3
4	24	0	-7.5	θ_4	θ_4
5	0	$-\pi/2$	4.5	$\theta_5 + \pi/2$	θ_5
6	0	$\pi/2$	8	θ_6	θ_6
7	0	0	0	$\pi/2$	0

3. Oluşturulan DH tablosunun satırları kullanılarak her bir ekleme ait dönüşüm matrisleri oluşturulur. Bu matrislerin oluşturulmasında denklem 2.30'da verilen genel dönüşüm matrisi kullanılır.

Tablonun 1. satırı kullanılarak; ${}^0_1T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Tablonun 2. satırı kullanılarak; ${}^1_2T = \begin{bmatrix} s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ c\theta_2 & -s\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Tablonun 3. satırı kullanılarak; ${}^2_3T = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & a_2 \\ s\theta_3 & c\theta_3 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Tablonun 4. satırı kullanılarak; ${}^3_4T = \begin{bmatrix} c\theta_4 & -s\theta_4 & 0 & a_3 \\ s\theta_4 & c\theta_4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Tablonun 5. satırı kullanılarak; ${}^4_5T = \begin{bmatrix} -s\theta_5 & -c\theta_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ -c\theta_5 & s\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Tablonun 6. satırı kullanılarak; ${}^5_6T = \begin{bmatrix} c\theta_6 & -s\theta_6 & 0 & 0 \\ 0 & 0 & -1 & -d_6 \\ s\theta_6 & c\theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Tablonun 7. satırı kullanılarak; ${}^6_7T = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

4. Eklem sayısı adedinde olan dönüşüm matrislerinin çarpılmasıyla ana çerçeve ile uç işlevci arasındaki konum ve oryantasyon ilişkisini veren ileri kinematik matrisi elde edilir.

$${}^0_7T = {}^0_1T {}^1_2T {}^2_3T {}^3_4T {}^4_5T {}^5_6T {}^6_7T \quad (3.2)$$

$${}^0_7T = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.3)$$

- Denklem 3.3'deki ileri kinematik matrisinin ilk sütunu 3x1 boyutlu normal vektörüdür ve elemanları $[n_x \ n_y \ n_z]^T$ şu şekilde bulunmuştur:

$$n_x = \cos(\theta_2 + \theta_3 + \theta_4) * \cos(\theta_1) * \cos(\theta_6) - \sin(\theta_6) * (\cos(\theta_5) * \sin(\theta_1) - \sin(\theta_2 + \theta_3 + \theta_4) * \cos(\theta_1) * \sin(\theta_5))$$

$$n_y = \sin(\theta_6) * (\cos(\theta_1) * \cos(\theta_5) + \sin(\theta_2 + \theta_3 + \theta_4) * \sin(\theta_1) * \sin(\theta_5)) + \cos(\theta_2 + \theta_3 + \theta_4) * \cos(\theta_6) * \sin(\theta_1)$$

$$n_z = \cos(\theta_2 + \theta_3 + \theta_4) * \sin(\theta_5) * \sin(\theta_6) - \sin(\theta_2 + \theta_3 + \theta_4) * \cos(\theta_6)$$

- Denklem 3.3'deki ileri kinematik matrisinin ikinci sütunu 3x1 boyutlu sliding (kayma) vektörüdür ve elemanları $[s_x \ s_y \ s_z]^T$ şu şekilde bulunmuştur:

$$s_x = -\cos(\theta_6) * (\cos(\theta_5) * \sin(\theta_1) - \sin(\theta_2 + \theta_3 + \theta_4) * \cos(\theta_1) * \sin(\theta_5)) - \cos(\theta_2 + \theta_3 + \theta_4) * \cos(\theta_1) * \sin(\theta_6)$$

$$s_y = \cos(\theta_6) * (\cos(\theta_1) * \cos(\theta_5) + \sin(\theta_2 + \theta_3 + \theta_4) * \sin(\theta_1) * \sin(\theta_5)) - \cos(\theta_2 + \theta_3 + \theta_4) * \sin(\theta_1) * \sin(\theta_6)$$

$$s_z = \sin(\theta_2 + \theta_3 + \theta_4) * \sin(\theta_6) + \cos(\theta_2 + \theta_3 + \theta_4) * \cos(\theta_6) * \sin(\theta_5)$$

- Denklem 3.3'deki ileri kinematik matrisinin üçüncü sütunu 3x1 boyutlu approach (yaklaşma) vektörüdür ve elemanları $[a_x \ a_y \ a_z]^T$ şu şekilde bulunmuştur:

$$a_x = \sin(\theta_1) * \sin(\theta_5) + \sin(\theta_2 + \theta_3 + \theta_4) * \cos(\theta_1) * \cos(\theta_5)$$

$$a_y = \sin(\theta_2 + \theta_3 + \theta_4) * \cos(\theta_5) * \sin(\theta_1) - \cos(\theta_1) * \sin(\theta_5)$$

$$a_z = \cos(\theta_2 + \theta_3 + \theta_4) * \cos(\theta_5)$$

- Denklem 3.3'deki ileri kinematik matrisinin dördüncü sütunu 3x1 boyutlu pozisyon vektörüdür ve elemanları $[p_x \ p_y \ p_z]^T$ şu şekilde bulunmuştur:

$$p_x = d_6 * (\sin(\theta_1) * \sin(\theta_5) + \sin(\theta_2 + \theta_3 + \theta_4) * \cos(\theta_1) * \cos(\theta_5)) - d_2 * \sin(\theta_1) - d_3 * \sin(\theta_1) - d_4 * \sin(\theta_1) + a_3 * \sin(\theta_2 + \theta_3) * \cos(\theta_1) + a_2 * \cos(\theta_1) * \sin(\theta_2) + d_5 * \cos(\theta_2 + \theta_3 + \theta_4) * \cos(\theta_1)$$

$$p_y = d_2 * \cos(\theta_1) + d_3 * \cos(\theta_1) + d_4 * \cos(\theta_1) + a_3 * \sin(\theta_2 + \theta_3) * \sin(\theta_1) - d_6 * \cos(\theta_1) * \sin(\theta_5) + a_2 * \sin(\theta_1) * \sin(\theta_2) + d_5 * \cos(\theta_2 + \theta_3 + \theta_4) * \sin(\theta_1) + d_6 * \sin(\theta_2 + \theta_3 + \theta_4) * \cos(\theta_5) * \sin(\theta_1)$$

$$p_z = d_1 + a_3 * \cos(\theta_2 + \theta_3) + a_2 * \cos(\theta_2) - d_5 * \sin(\theta_2 + \theta_3 + \theta_4) + d_6 * \cos(\theta_2 + \theta_3 + \theta_4) * \cos(\theta_5)$$

Bulunan ileri kinematik matrisinin doğruluk kontrolü için, DH tablosunda bulunan a , d , α ve θ değerleri ileri kinematik matrisinde yerine konur. θ değerleri sıfır olarak kabul edilir ve böylece başlangıç pozisyonunda (zero position) robotun uç işlevcisinin merkezine göre yeri tespit edilir. Bu işlemler yapılırsa ileri kinematik matrisi;

$${}^0_7T = \begin{bmatrix} 1 & 0 & 0 & 4.5 \\ 0 & 1 & 0 & -4 \\ 0 & 0 & 1 & 81 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

şeklinde bulunur. Burada konum vektörü

$$P = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} 4.5 \\ -4 \\ 81 \end{bmatrix}$$

dir.

Başlangıç pozisyonunda robotun uç işlevcisinin yeri, robotun merkezinden itibaren ölçülürse x yönünde 4.5 cm, y yönünde -4 cm ve z yönünde 81 cm olarak bulunmuştur. Ölçülen değerler ile ileri kinematik analizi ile hesaplanan değerler birebir tutmuştur. Buna göre ileri kinematik analizi doğru yapılmıştır.

3.3.2 TQ MA2000 robot manipülatörünün ters kinematik analizi

Ters kinematik analizinde, birinci derece ters alma işlemi yapılır. Bunun için denklem 3.2'nin her iki tarafı da $\begin{bmatrix} 0 & 1 \end{bmatrix}^{-1}$ matrisi ile çarpılır ve denklem 3.4 elde edilir. Birinci derece ters alma işlemi yeterli olmaz ise ikinci veya üçüncü derece ters alma işlemleri de gerçekleştirilebilir.

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}^{-1} {}^0_7T = {}^1_2T {}^2_3T {}^3_4T {}^4_5T {}^5_6T {}^6_7T \quad (3.4)$$

Denklem 3.4'deki ${}^0_7T$ matrisi sembolik bir matristir ve denklem 3.5'de gösterilmiştir.

$$\begin{bmatrix} {}^0_7T \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.5)$$

${}^0_1T$ matrisinin ters alma işlemi şu şekilde gerçekleştirilir:

$$\begin{bmatrix} {}^0_1T \end{bmatrix}^{-1} = \begin{bmatrix} {}^0_1R^T & -{}^0_1R^T {}^0P_1 \\ 0 & 1 \end{bmatrix}$$

Buna göre;

$$\begin{bmatrix} {}^0_1R \end{bmatrix} = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 \\ s\theta_1 & c\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} {}^0_1R \end{bmatrix}^T = \begin{bmatrix} c\theta_1 & s\theta_1 & 0 \\ -s\theta_1 & c\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$-\begin{bmatrix} {}^0_1R \end{bmatrix}^T {}^0P_1 = -\begin{bmatrix} c\theta_1 & s\theta_1 & 0 \\ -s\theta_1 & c\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ d_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -d_1 \end{bmatrix}$$

$$\begin{bmatrix} {}^0_1T \end{bmatrix}^{-1} = \begin{bmatrix} c\theta_1 & s\theta_1 & 0 & 0 \\ -s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & -d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.6)$$

olarak bulunur. Denklem 3.4 de bulunan matrisler θ_4 , θ_5 ve θ_6 sıfır kabul edilerek (robotun bilek kısmı kullanılmadığı için) yerine konursa;

$$p_x * \cos(\theta_1) + p_y * \sin(\theta_1) = (9 * \cos(\theta_2 + \theta_3))/2 + 32 * \sin(\theta_2 + \theta_3) + 23 * \sin(\theta_2)$$

(3.7)

$$p_z - 26 = 32 * \cos(\theta_2 + \theta_3) - (9 * \sin(\theta_2 + \theta_3))/2 + 23 * \cos(\theta_2) \quad (3.8)$$

$$p_y * \cos(\theta_1) - p_x * \sin(\theta_1) = -4 \quad (3.9)$$

şeklinde 3 bilinmeyenli $(\theta_1, \theta_2, \theta_3)$ 3 denklem bulunmuş olur. Bu üç denklemin çözümü için şu adımlar izlenir.

1. Denklem 2.44'te verilen trigonometrik eşitlik, denklem 3.9'a uygulanırsa;

$a = p_x$, $b = p_y$, $c = -4$ olur. Buna göre θ_1 şöyle hesaplanır:

$$\theta_1 = \text{atan2}(-p_x, p_y) \pm \text{atan2}(\sqrt{(-p_x)^2 + (p_y)^2 - (-4)^2}, -4) \quad (3.10)$$

olarak bulunur.

2. Denklem 2.48'te verilen trigonometrik eşitlik, denklem 3.7 ve 3.8'e uygulanırsa;

$$i = 2, j = 3$$

$$a = -4.5, b = 32, c = 23$$

$$d = p_z - 26$$

$$e = p_x * \cos\theta_1 + p_y * \sin\theta_1$$

$$s = -4.5 * \cos\theta_3 - 32 * \sin\theta_3$$

$$t = 32 * \cos\theta_3 - 4.5 * \sin\theta_3 + 23$$

$$r = \left(\frac{(d)^2 + (e)^2 - (-4.5)^2 - 32^2 - 23^2}{2 * 23} \right)$$

olur. Buna göre θ_2 ve θ_3 şöyle hesaplanır:

$$\theta_3 = \text{atan2}(-4.5, 32) \pm \text{atan2}(\sqrt{23^2 + (p_z - 26)^2 - r^2}, r) \quad (3.11)$$

ve

$$\theta_2 = \text{atan2}(s * d + t * e, -s * e + t * d) \quad (3.12)$$

olarak bulunur.

Görüldüğü gibi bilinmeyenler θ_1, θ_2 ve θ_3 'ün bulunabilmesi için tek gereken, robotun ulaşması istenen pozisyonun koordinatları (p_x, p_y ve p_z) dir. Bu koordinat bilgisi, görüntü işleme yöntemi ile bulunmuştur.

Örnek bir ters kinematik analizi yapmak istersek; Şekil 3.16'daki '0. nesne' olarak belirtilen nesnenin koordinatlarını kullanalım. Görüntü işleme yöntemi ile nesnenin x ve y koordinatları tespit edilmişti. Nesne zeminde bulunduğu için orta noktasının yerden yüksekliği, kendi yüksekliğinin yarısına eşittir ve bu 2 cm olarak ölçülmüştür.

- **θ_1 ' in çözümü**

$$P_{0.Nesne} = \begin{bmatrix} 30.3082 \\ 19.8611 \\ 2 \end{bmatrix}$$

Buna göre;

$a = 30.3082$, $b = 19.8611$, $c = -4$ olur. Bu değerler denklem 3.10'da yerine konursa;

$$\theta_1 = \frac{\text{atan2}(-30.3082, 19.8611) \pm \text{atan2}(\sqrt{(-30.3082)^2 + (19.8611)^2 - (-4)^2}, -4)}{2}$$

$$= -0.9907 \pm 1.6814 \text{ rd}$$

$$\theta_1 = \left\{ \begin{array}{c} 0.6907 \\ -2.6721 \end{array} \right\} \text{ rd} = \left\{ \begin{array}{c} 39.57 \\ -153.1 \end{array} \right\} \text{ dg} \quad (3.13)$$

θ_1 için iki adet çözüm kümesi bulundu. θ_3 ve θ_2 'nin çözümlerinin bu iki çözüm kümesine göre ayrı ayrı yapılması gerekir.

- **$\theta_1 = 0.6907 \text{ rd}$ için θ_3 çözümü**

$$a = -4.5, b = 32, c = 23, d = -24$$

$$e = 30.3082 * \cos(0.6907) + 19.8611 * \sin(0.6907) = 36.0146$$

$$r = \frac{(-24)^2 + (36.0146)^2 - (-4.5)^2 - 32^2 - 23^2}{2 * 23} = 6.5174$$

olarak bulunur. Bu değerler denklem 3.11'de yerine konursa;

$$\theta_3 = atan2(-4.5, 32) \pm atan2(\sqrt{23^2 + (-24)^2 - 6.5174^2}, 6.5174)$$

$$= -0.1397 \pm 1.3735 \text{ rd}$$

$$\theta_3 = \left\{ \begin{array}{c} 1.2337 \\ -1.5132 \end{array} \right\} \text{ rd} = \left\{ \begin{array}{c} 70.68 \\ -86.7 \end{array} \right\} \text{ dg} \quad (3.14)$$

- **$\theta_1 = -2.6721 \text{ rd}$ için θ_3 çözümü**

a, b, c, d değerleri önceki çözümdeki ile aynıdır.

$$e = 30.3082 * \cos(-2.6721) + 19.8611 * \sin(-2.6721) = -36.0146$$

$$r = \frac{(-24)^2 + (-36.0146)^2 - (-4.5)^2 - 32^2 - 23^2}{2 * 23} = 6.5174$$

olarak bulunur. Bu değerler denklem 3.11'de yerine konursa;

$$\theta_3 = atan2(-4.5, 32) \pm atan2(\sqrt{23^2 + (-24)^2 - 6.5174^2}, 6.5174)$$

$$= -0.1397 \pm 1.3735 \text{ rd}$$

$$\theta_3 = \left\{ \begin{array}{c} 1.2337 \\ -1.5132 \end{array} \right\} \text{ rd} = \left\{ \begin{array}{c} 70.68 \\ -86.7 \end{array} \right\} \text{ dg} \quad (3.15)$$

Dikkat edilirse, e değeri önceki e değerinin eksi katı çıkmıştır. Ancak r hesaplanırken e değerinin karesi alındığı için e değerinin değişmesi r 'yi değiştirmemiştir. a, b, c ve d değerleri de aynı olduğu için çözüm kümeleri aynı çıkmıştır.

- **$\theta_1 = 0.6907 \text{ rd}$ ve $\theta_3 = 1.2337 \text{ rd}$ için θ_2 çözümü**

$$d = -24$$

$$e = 30.3082 * \cos(0.6907) + 19.8611 * \sin(0.6907) = 36.0146$$

$$s = -4.5 * \cos(1.2337) - 32 * \sin(1.2337) = -31.6874$$

$$t = 32 * \cos(1.2337) - 4.5 * \sin(1.2337) + 23 = 29.3372$$

olarak bulunur. Bu değerler denklem 3.12'de yerine konursa;

$$\theta_2 = \text{atan2}(-31.6874 * -24 + 29.3372 * 36.0146, 31.6874 * 36.0146 + 29.3372 * -24)$$

$$= 1.3347 \text{ rd}$$

$$\theta_2 = \{1.3347\} \text{ rd} = \{76.47\} \text{ dg} \quad (3.16)$$

- **$\theta_1 = 0.6907 \text{ rd}$ ve $\theta_3 = -1.5132 \text{ rd}$ için θ_2 çözümü**

$$d = -24$$

$$e = 30.3082 * \cos(0.6907) + 19.8611 * \sin(0.6907) = 36.0146$$

$$s = -4.5 * \cos(-1.5132) - 32 * \sin(-1.5132) = 31.6879$$

$$t = 32 * \cos(-1.5132) - 4.5 * \sin(-1.5132) + 23 = 29.3346$$

olarak bulunur. Bu değerler denklem 3.12'de yerine konursa;

$$\theta_2 = \text{atan2}(31.6879 * -24 + 29.3346 * 36.0146, -31.6879 * 36.0146 + 29.3346 * -24)$$

$$= 2.9825 \text{ rd}$$

$$\theta_2 = \{2.9825\} \text{ rd} = \{170.88\} \text{ dg} \quad (3.17)$$

- **$\theta_1 = -2.6721 \text{ rd}$ ve $\theta_3 = 1.2337 \text{ rd}$ için θ_2 çözümü**

$$d = -24$$

$$e = 30.3082 * \cos(-2.6721) + 19.8611 * \sin(-2.6721) = -36.0146$$

$$s = -4.5 * \cos(1.2337) - 32 * \sin(1.2337) = -31.6874$$

$$t = 32 * \cos(1.2337) - 4.5 * \sin(1.2337) + 23 = 29.3372$$

olarak bulunur. Bu değerler denklem 3.12'de yerine konursa;

$$\theta_2 = \text{atan2}(-31.6874 * -24 + 29.3372 * -36.0146, 31.6874 * -36.0146 + 29.3372 * -24)$$

$$= -2.9825 \text{ rd}$$

$$\theta_2 = \{-2.9825\} \text{ rd} = \{-170.88\} \text{ dg} \quad (3.18)$$

- **$\theta_1 = -2.6721 \text{ rd}$ ve $\theta_3 = -1.5132 \text{ rd}$ için θ_2 çözümü**

$$d = -24$$

$$e = 30.3082 * \cos(-2.6721) + 19.8611 * \sin(-2.6721) = -36.0146$$

$$s = -4.5 * \cos(-1.5132) - 32 * \sin(-1.5132) = 31.6879$$

$$t = 32 * \cos(-1.5132) - 4.5 * \sin(-1.5132) + 23 = 29.3346$$

olarak bulunur. Bu değerler denklem 3.12'de yerine konursa;

$$\theta_2 = \text{atan2}(31.6879 * -24 + 29.3346 * -36.0146, -31.6879 * -36.0146 + 29.3346 * -24)$$

$$= -1.3347 \text{ rd}$$

$$\theta_2 = \{-1.3347\} \text{ rd} = \{-76.47\} \text{ dg} \quad (3.19)$$

Ters kinematik probleminin yazılımsal çözümü, C++'da yazılmıştır. Görüntü işleme kodlarının devamına yazılmıştır. Görüntü işleme yöntemi ile

bulunan taşınacak nesnelerin pozisyonları, ters kinematik analizleri yapılarak θ_1 , θ_2 ve θ_3 için çözüm kümeleri bulunur.

Analitik olarak yapılan ters kinematik çözümünde, her bir döner eklemin ekstradan çözüm kümesi getireceğinden önceki bölümlerde bahsetmiştik.. Tamamı döner eklemlili olan bir robotta bulunacak çözüm kümesi sayısı $2^{(n-1)}$ formülü ile hesaplanır. n eklem sayısını ifade etmektedir. Çalışmada kullanılan robotta, bulunan ters kinematik çözümü sayısı $2^{(3-1)} = 4$ tane olmuştur. Eğer robotta bulunan eklemlerin tamamı kullanılsa idi çok daha fazla çözüm kümesi çıkacaktı. Bulunan bu 4 çözüm kümesi aşağıda verilmiştir:

$$\text{ÇK1} = \begin{Bmatrix} 0.6907 \\ 1.3347 \\ 1.2337 \end{Bmatrix} rd = \begin{Bmatrix} 39.57 \\ 76.47 \\ 70.68 \end{Bmatrix} dg$$

$$\text{ÇK2} = \begin{Bmatrix} 0.6907 \\ 2.9825 \\ -1.5132 \end{Bmatrix} rd = \begin{Bmatrix} 39.57 \\ 170.88 \\ -86.7 \end{Bmatrix} dg$$

$$\text{ÇK3} = \begin{Bmatrix} -2.6721 \\ -2.9825 \\ 1.2337 \end{Bmatrix} rd = \begin{Bmatrix} -153.1 \\ -170.88 \\ 70.68 \end{Bmatrix} dg$$

$$\text{ÇK4} = \begin{Bmatrix} -2.6721 \\ -1.3347 \\ -1.5132 \end{Bmatrix} rd = \begin{Bmatrix} -153.1 \\ -76.47 \\ -86.7 \end{Bmatrix} dg$$

Bulunan Çözüm Kümelerinin Analizi

Bulunan bu çözüm kümelerinin bazıları sanaldır. Yani gerçekte robotun fiziksel olarak bu açı değerlerine ulaşmasını sağlamamaktadır. Gerçek olan çözüm kümelerinden ise bazıları robotu daha uzun yollardan sonuca götürmeye çalışmaktadır. Bu yüzden eleme yapılarak en doğru çözüm kümesinin bulunması gerekmektedir.

- Sanal çözümleri yok etme:

Birinci eleme yöntemimiz sanal çözümleri tespit edip bunları çözüm kümelerinden çıkarmaktır.

2. çözüm kümesini incelersek

$$\text{ÇK2} = \begin{Bmatrix} 0.6907 \\ 2.9825 \\ -1.5132 \end{Bmatrix} rd = \begin{Bmatrix} 39.57 \\ 170.88 \\ -86.7 \end{Bmatrix} dg$$

olarak bulunmuş. Ancak robotun 2. eklemine yapabileceği θ_2 açısı $(+135, -135)$ derece aralığındandır. Dolayısıyla 2. çözüm kümesi, fiziksel olarak sistemi sağlamamıştır. Aynı şekilde 3. çözüm kümesini incelersek;

$$\text{ÇK3} = \begin{Bmatrix} -2.6721 \\ -2.9825 \\ 1.2337 \end{Bmatrix} rd = \begin{Bmatrix} -153.1 \\ -170.88 \\ 70.68 \end{Bmatrix} dg$$

θ_2 açısı -170.88 derece çıktığı için bu çözüm kümesi de fiziksel olarak sistemi sağlamamaktadır.

Elimizde 2 adet çözüm kümesi daha kalmıştır. Kalan bu çözüm kümeleri, sistemi fiziksel olarak sağlamaktadır. Dolayısıyla farklı bir eleme yöntemine daha ihtiyaç vardır.

- En kısa yolu seçme:

Taşınacak nesnenin bulunduğu konum $(30.3082, 19.8611)$, 1. bölgede kalmaktadır. Dolayısıyla robotun bel eklemine yani θ_1 açısının birinci bölge aralığı olan $(0,90)$ derece arasında kalması daha doğru bir çözüm getirir. ÇK4'de ise robotun beli 3. bölgeye dönmüş, omuz ve dirsek açılarını ise negatif değerlere getirerek nesneyi tersten almaya çalışmıştır. Gerçekte istenilen ise nesneye her zaman pozitif omuz ve dirsek açıları ile ulaşılmasıdır. Dolayısıyla uygulanan bu en doğru çözümü bulma yöntemleri sonucunda ÇK1 en doğru çözüm olarak bulunmuştur.

$$\text{En Doğru Çözüm Kümesi} = \text{ÇK1} = \begin{Bmatrix} 0.6907 \\ 1.3347 \\ 1.2337 \end{Bmatrix} rd = \begin{Bmatrix} 39.57 \\ 76.47 \\ 70.68 \end{Bmatrix} dg$$

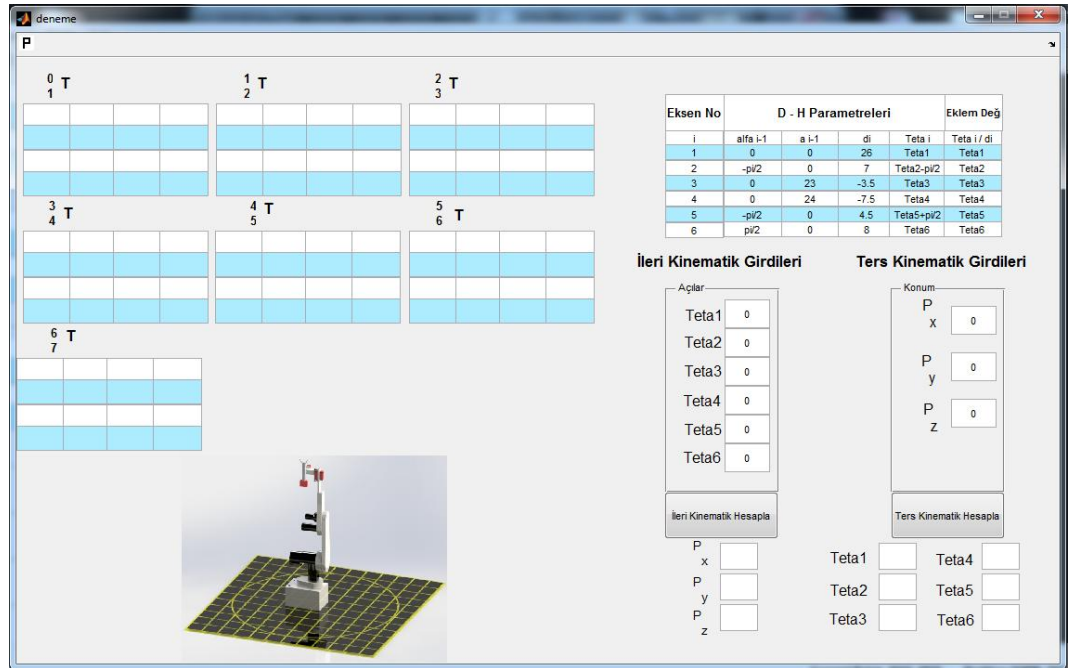
Anlatılan bu en doğru çözümü bulma yöntemi, C++'da yazılmıştır ve bu kodlar ekler kısmında verilmiştir.

Şekil 3.16'da görüldüğü gibi, nesnenin doğru algılanıp algılanmadığı kullanıcıya sorulduktan sonra, kullanıcı E(evett) cevabını verirse, tespit edilen taşınacak nesnenin ters kinematik analizleri yapılarak çözüm kümeleri bulunur, ve bulunan bu çözüm kümeleri de 'en doğru çözüm kümesi' algoritmasına sokularak nihai çözüm kümesi tespit edilir. Tespit edilen çözüm kümesindeki açı değeri, Arduino mikrodeneleyicisine USB port üzerinden gönderilir. Buradan sonra, robotun kontrollü hareketinin gerçekleştirilmesi işlemi vardır.

3.4 MATLAB Guide Ortamında Hazırlanmış Kinematik Analiz Arayüzü

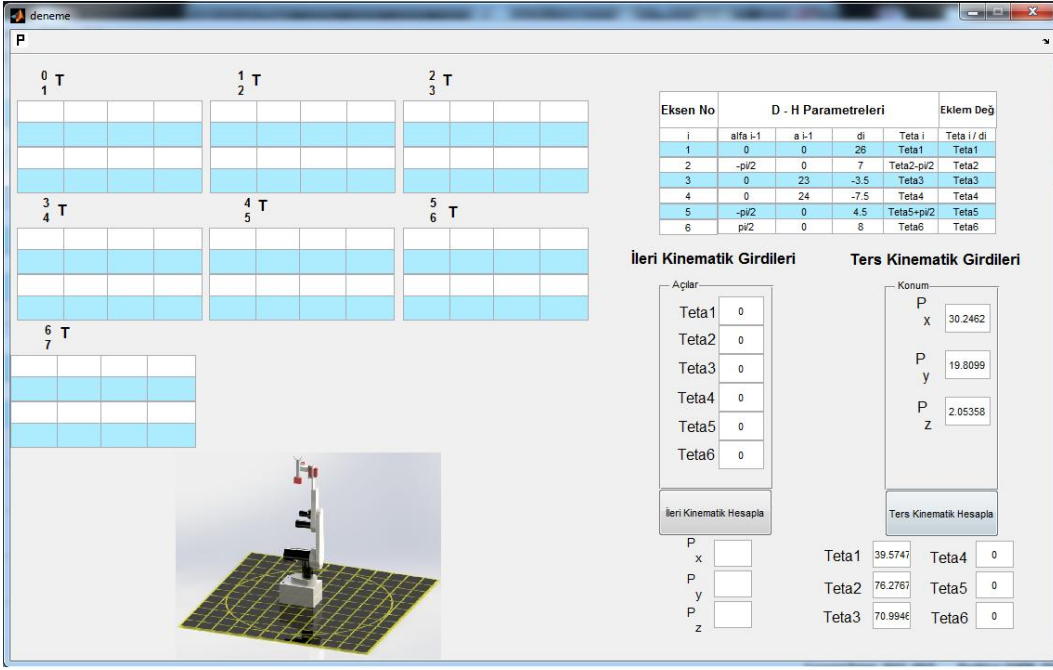
TQ MA2000 robot manipülatörünün ileri ve ters kinematik analizini hızlı bir şekilde yaptırabilmek amacıyla MATLAB - Guide ortamında bir arayüz tasarlandı. Bu arayüz sistemin genel çalışmasında kullanılmamakla birlikte, yapılan deneme çözümleri sırasında çok yararlı olmuştur. Ayrıca ileri kinematik

analizinde, tüm eklemlerin ileri yön matrisleri ayrı ayrı görülebilmesi de önemli bir fayda sağlamıştır. Şekil 3.19'da tasarlanan bu arayüz görülmektedir.



Şekil 3.19 MATLAB - Guide ortamında tasarlanan arayüz

Arayüze ilk girildiğinde Şekil 3.19'da görüldüğü gibi TQ MA2000 robot manipülatörünün DH tablosundan başka herhangi bir veri yoktur. Ters kinematik girdileri bölümüne, Şekil 3.16'da 0. nesne olarak tanımlanmış olan nesnenin konumu girilip ters kinematik hesapla butonuna basılırsa Şekil 3.20'deki gibi bir çözüm elde edilmektedir.



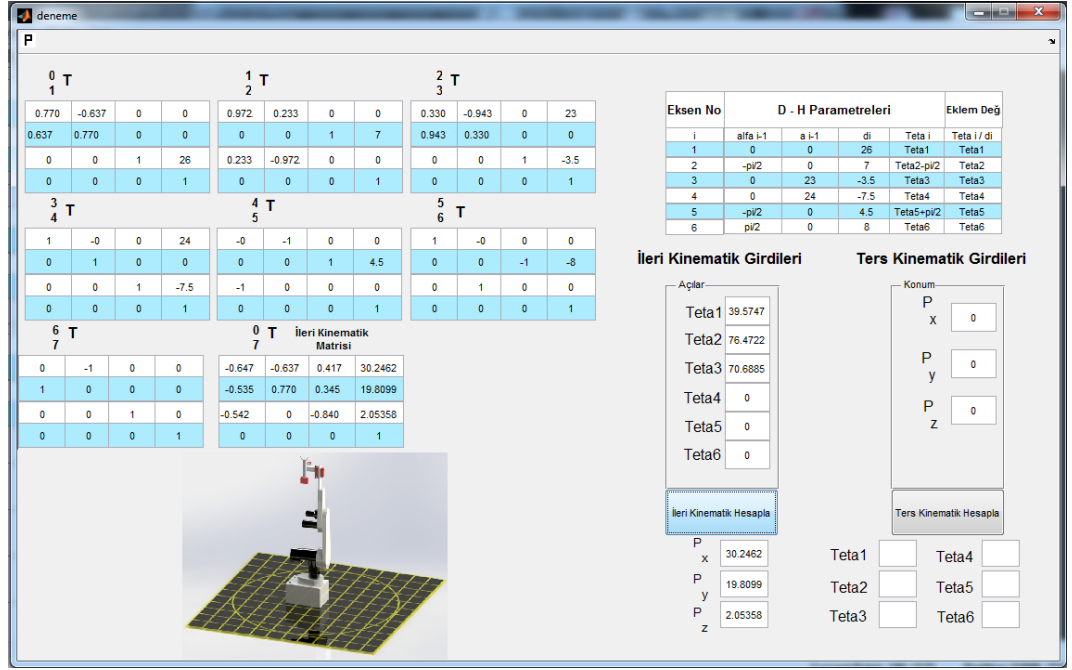
Şekil 3.20 MATLAB - Guide ortamında tasarlanan arayüzde ters kinematik analizi

Görüldüğü gibi bulunan açı değerleri, Şekil 3.16'deki değerler ile örtüşmüştür.

Ters kinematik analizi sırasında, arayüzün sol tarafındaki matrisler boş kalmıştır. Bu matrisler ileri kinematik analizinde kullanılan her bir eklem için ileri yön matrisini ifade etmektedir. İleri yön kinematik analizi yapmak için, Şekil 3.20'de elde edilen açı değerlerini ileri kinematik girdileri bölümüne yazıp, ileri kinematik hesapla butonuna basılırsa Şekil 3.21'deki gibi bir çözüm elde edilir.

İleri kinematik analizi ile elde edilen P_x , P_y , P_z konum bilgilerinin, Şekil 3.20'deki verilerle birebir aynı çıktığına dikkat edilmelidir.

Bu arayüzde yapılan ileri kinematik analizinde, robotta bulunan 6 eklem de kullanılabilmesi sağlanmıştır. Ancak ters kinematik analizinde, çözümlerin çok yüksek sayılara ulaşmasının engellenmesi için, ana çalışmada olduğu gibi burada da ilk 3 eklem (bel, omuz, dirsek) kullanılarak çözüm yaptırılmıştır. Bilekte bulunan eklemler sabit, yani açı değerleri sıfır olarak kabul edilmiştir.



Şekil 3.21 MATLAB - Guide ortamında tasarlanan arayüzde ileri kinematik analizi

3.5 Arduino Mikrodenetleyicisi ile Gerçekleştirilen İşlemler

Arduino mikrodenetleyicisi, sistemin ana kontrol elemanı olarak kullanılmıştır. Arduino içerisine yazılan yazılımın amacı, robotun istenilen pozisyona kontrollü hareketinin sağlanmasıdır. Bunun gerçekleştirilebilmesi için yapılan işlemler ve yazılan kodlar, adım adım anlatılacaktır.

3.5.1 Arduino'ya kütüphanelerin yüklenmesi

Bu çalışmada, Arduino'nun içerisinde tanımlı olan kütüphanelerden farklı olarak, 'PID' ve 'multimap' kütüphaneleri kullanılmıştır. Bu kütüphanelerin Arduino içerisine eklenmesi şu şekilde yapılır.

```
#include <PID_v1.h>
#include <multiMap.h>
```

3.5.2 Giriş ve çıkış pinlerinin belirlenmesi

Arduino'ya kütüphanelerin eklenmesinden sonra sırada giriş ve çıkış pinlerinin belirlenmesi vardır. Çalışmada kullanılan 3 adet L298 motor sürücü kartının her birine, Arduino'dan 3'er adet çıkış gitmektedir (bkz. Tablo 3.2). Ayrıca motorlardan geri besleme almamızı sağlayan potansiyometreler de Arduino'ya bağlanmıştır (bkz. Tablo 3.3). Tablo 3.2 ve Tablo 3.3'de gösterilen bağlantılar, aşağıda verilen kod yardımı ile çalışır hale getirilir.

```
int m1en = 8;           //motor1 aktif etme pini
int m1direction = 9;    //motor1 yon secim pini
int m1speed = 10;       //motor1 pwm pini
int m2en = 7;           //motor2 aktif etme pini
int m2direction = 4;     //motor2 yon secim pini
int m2speed = 5;        //motor2 pwm pini
int m3en = 6;           //motor3 aktif etme pini
int m3direction = 2;     //motor3 yon secim pini
int m3speed = 3;        //motor3 pwm pini
int m1pot = A0;          //motor1 potansiyometre analog okuma pini
int m2pot = A1;          //motor2 potansiyometre analog okuma pini
int m3pot = A2;          //motor3 potansiyometre analog okuma pini

pinMode(m1en,OUTPUT);    //cikis atamaları
pinMode(m1direction,OUTPUT);
pinMode(m1speed,OUTPUT);
pinMode(m2en,OUTPUT);
```

```
pinMode(m2direction,OUTPUT);
pinMode(m2speed,OUTPUT);
pinMode(m3en,OUTPUT);
pinMode(m3direction,OUTPUT);
pinMode(m3speed,OUTPUT);
```

Kodlarda da görüldüğü gibi öncelikle hangi pin numaralarının kullanılacağı belirlenmiş ve bu pin numaraları isimlendirilmiştir. Daha sonra dijital olan çıkışlar, 'pinMode' komutu ile çıkış olarak atanmıştır.

3.5.3 Potansiyometrelerin okunması ve anlamlandırılması

Arduino'da analog okuma, 'analogRead' komutu ile yapılmaktadır. Analog olarak okunacak olan A0,A1 ve A2 pinleri aşağıdaki gibi analogRead komutu ile okunarak daha önceden tanımlanmış 'potvalue' değişkenlerinin içerisine atılmıştır.

```
potvalue1 = analogRead(m1pot);           //motor1 potansiyometre oku
potvalue2 = analogRead(m2pot);           //motor2 potansiyometre oku
potvalue3 = analogRead(m3pot);           //motor3 potansiyometre oku
```

Potansiyometreler lineer olmayan bir çalışma grafiğine sahiptir. Bu nedenle lineerizasyon(lineere yaklaştırma) işlemi yapılmıştır. Bunu gerçekleştirebilmek için, her eklemden belirli açı değerlerine karşılık okunan potansiyometre değerleri not edilmiş ve bunlar 'multiMap' fonksiyonuna sokularak küçük aralıklarda lineerleştirme yapılmıştır. Seri porttan gelen, eklemlerin ulaşması gereken açı değerleri bu fonksiyona sokulur. Fonksiyon çıktı olarak, bu açı değerlerine karşılık gelen potansiyometre değerlerini verir.

Örnek olarak birinci eklemi göz önüne alalım. Bu eklemden bulunan potansiyometre, diğer eklemlerdeki potansiyometrelere göre daha lineer bir yapıdadır. Bu nedenle 5 adet ölçüm alınması yeterli görülmüştür. Alınan bu ölçümler Tablo 3. 7'de gösterilmiştir.

Tablo 3.7 Birinci eklem için Açık - Potansiyometre ilişkisi

Ölçülen Açık Değeri	Okunan Potansiyometre Değeri
-160	17
-90	234
0	512
90	790
160	1006

Açık değerlerinin ölçümü için BOSCH GLM-80 dijital açıölçeri kullanılmıştır. Bu açıölçer 0.1 derece hassasiyet ile açı ölçümleri yapabilmektedir. Potansiyometre değerleri ise Arduino'nun analog okuma kanallarından alınmıştır. Analog okuma aralığı (0,1024) dır.

Ölçülen bu açı ve potansiyometre değerleri multiMap fonksiyonuna sokulur. MultiMap fonksiyonunun yapısına bakarsak;

```
multiMap<float>(float val, float in,float out, size);
```

şeklindedir. Burada float val, fonksiyonun çalıştırılması sonucu çıkacak değerdir. Float in ve float out değerleri ise ölçüm ile elde edilen değerlerdir. Burada bizim için float in açı değerleri, float out ise potansiyometre değerleridir. Size(boyut) ise dizinin boyutudur yani alınan ölçüm miktarıdır. Birinci eklem için bu değer 5 olarak belirlenmiştir.

Örnek olarak; fonksiyona 135 derece şeklinde bir değer girsin. Fonksiyon, verilen referans açı ölçüm değerlerine bakar ve en yakın olan aralığı bulur. Bu aralık (90,160) derecedir. Fonksiyon bu aralığı lineer kabul ederek hesaplamayı yapar. Bu derece aralığına karşılık gelen potansiyometre aralığı (790,1006) dır. Buna göre fonksiyona verilen 135 derece girişi için çıkacak olan potansiyometre değeri 929'dur.

Aşağıdaki kod bloğunda her üç eklem için de gerçekleştirilen açı - potansiyometre dönüşümü verilmiştir. İkinci ve üçüncü eklemlerdeki potansiyometreler daha az lineer olduğu için daha fazla ölçüm alınmış ve

lineerleştirme aralığı küçültülmüş, böylece daha yüksek bir hassasiyet sağlanmıştır.

```
//motor1 mapping (-160, 160 derece)
float out1[] = {17,234,512,790,1006}; // 5 olcum alindi
float in1[] = {-160,-90,0,90,160};
m1mapans = multiMap<float>(m1_aci_gelen, in1, out1, 5);

//motor2 mapping (-45, 120 derece)
float out2[] =
{26,44,59,74,88,103,117,133,148,163,179,195,210,225,241,255,271,287,303,320,
335,352,367,380,395,408,424,439,457,471,487,503,516,533}; // 34 olcum alindi
float in2[] = {-45,-40,-35,-30,-25,-20,-15,-10,-
5,0,5,10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,95,100,105,110,115,12
0};
m2mapans = multiMap<float>(m2_aci_gelen, in2, out2, 34);

//motor3 mapping (-135, 135 derece)
float out3[] =
{102,117,132,147,162,177,192,207,222,237,252,267,282,297,312,327,341,356,37
3,386,400,417,432,444,463,477,487,512,529,543,559,572,589,603,618,637,650,6
64,685,703,714,732,747,758,775,793,804,818,828,847,862,877,893,910,925}; //
55 olcum alindi
float in3[] = {-135,-130,-125,-120,-115,-110,-105,-100,-95,-90,-85,-80,-75,-70,-
65,-60,-55,-50,-45,-40,-35,-30,-25,-20,-15,-10,-
5,0,5,10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,95,100,105,110,115,12
0,125,130,135};
m3mapans = multiMap<float>(m3_aci_gelen, in3, out3, 55);
```

3.5.4 Seri haberleşmenin başlatılması

Arduino ile görüntü işleme ve ters kinematik analizinin yazıldığı geliştirme ortamı (Code::Blocks) arasındaki haberleşme USB port üzerinden yapılmıştır. Arduino'da seri haberleşme 'Serial.begin' komutu ile başlatılır. Komutun yanında parantez içerisinde, haberleşmenin yapılacağı 'baud rate, data bit, parity, stop bit' gibi bilgiler verilir. Varsayılan data bit(veri boyutu) değeri 8, parity(eşlik) değeri none (yok), ve stop bit(durdurma biti) değeri 1 olarak tanımlıdır. Bu bilgiler varsayılan halleri ile kullanılmış, baud rate değeri şu şekilde tanımlanmıştır:

```
Serial.begin(9600);
```

3.5.5 PWM frekansının ayarlanması

Arduino mikrodnetleyicinin PWM frekanslarının kontrolü için, Timer içeriklerinin değiştirilmesi gerekmektedir. Tablo 3.8'de hangi Timer'ın hangi pinleri etkilediği gösterilmiştir.

Tablo 3.8 Arduino Mega 2560 için Timer'lar ve kontrol ettiği pinler

Timer	Kontrol Ettiği Pinler
Timer 0	4,13
Timer 1	11,12
Timer 2	9,10
Timer 3	2,3,5
Timer 4	6,7,8

Bu çalışmada kullanılan 3 adet motorun PWM sinyalleri, Tablo 3.2'de görüldüğü gibi (IN1 uçları) 3,5 ve 10 numaralı pinlerden verilmiştir. Bu pinleri kontrol eden Timer'lar ise Timer2 ve Timer3'tür. Tablo 3.9'da PWM frekans seçimi için Timer'lara atanması gereken değerler verilmiştir.

Tablo 3.9 PWM frekans seçimi

Parametre	Bölen	Frekans (Hz)
0x01	1	31372.55
0x02	8	3921.16
0x03	32	980.39
0x04	64	490.20
0x05	128	245.10
0x06	256	122.65
0x07	1024	30.64

Bu çalışmada PWM frekansları, 4 kHz değerine set edilmek istenmiştir. Bunun için Timer2 ve Timer3'ün değeri 0x02 olarak belirlenmiştir. PWM frekanslarını ayarlamak için yazılan kodlar şunlardır:

```
TCCR2B = TCCR2B & 0b11111000 | 0x02; // Motor1 PWM 4 kHz'e set edildi.
TCCR3B = TCCR3B & 0b11111000 | 0x02; // Motor2 ve Motor3 PWM 4 kHz'e
set edildi.
```

3.5.6 Seri porttan gelen verilerin okunması

Code::Blocks geliştirme ortamında ters kinematik analizi ile bulunan açı değerleri, Arduino mikrodenetleyiciye USB port üzerinden gönderildi. Arduino'ya gelen bu verilerin okunması ve parçalanması aşağıda verilen kod bloğu ile yapılmıştır.

```
while(Serial.available() > 0)
{
    char gelen = Serial.read();
    seri_veri += gelen;

    if(gelen == ' ' && refcounter == 0)
    {
        refcounter+=1;
        m1_aci_gelen = seri_veri.toFloat();
        seri_veri = "";
    }
    else if(gelen == ' ' && refcounter == 1)
    {
        refcounter+=1;
        m2_aci_gelen = seri_veri.toFloat();
        seri_veri = "";
    }
    else if(gelen == '\n')
    {
        refcounter = 0;
        m3_aci_gelen = seri_veri.toFloat();
        seri_veri = "";
    }
}
```

Burada yapılan, yeni bir satır gelene kadar olan verileri okuyup bu verileri aralarındaki boşluklara göre parçalama esasına dayanmaktadır. İlk boşluğa kadar olan veri 1. eklem alması gereken açı değerini, 2. boşluğa kadar olan veri 2. eklem alması gereken açı değerini ve yeni satıra kadar olan veri de 3. eklem alması gereken açı değerini temsil etmektedir.

3.5.7 PID kütüphanesinin kullanılması

Arduino'da PID kontrolör oluşturmak için 'PID()' fonksiyonu kullanılır. Fonksiyonun yapısı şu şekildedir:

```
PID(&Input, &Output, &Setpoint, Kp, Ki, Kd, Direction)
```

Burada kullanılan parametreleri açıklayalım.

- **Input:** Kontrol edilmeye çalışılan değişkendir.
- **Output:** PID kontrolör ile ayarlanacak olan değişkendir. Bloğun çıkışıdır.
- **Setpoint:** Kontrol edilmeye çalışılan değişkenin ulaştırılması istenen değerdir.
- **Kp,Ki,Kd:** Ayar parametreleridir. PID kontrolörün sisteme nasıl etki edeceği bu parametreler ile belirlenir.
- **Direction:** Belirli bir hata ile karşılaşıldığında, çıkışın hangi yönde etki edeceğini belirler.

Bu çalışmada kullanılan 3 adet motorun her biri için farklı PID kontrolör kullanılmıştır. Bu kontrolörler şu şekilde tanımlanmıştır:

```
PID pid1(&potvalue1,&error1,&m1_git,Kp1,Ki1,Kd1,DIRECT);
PID pid2(&potvalue2,&error2,&m2_git,Kp2,Ki2,Kd2,DIRECT);
PID pid3(&potvalue3,&error3,&m3_git,Kp3,Ki3,Kd3,DIRECT);

pid1.Compute();
pid2.Compute();
pid3.Compute();
```

'Compute()' fonksiyonu her döngüde çalıştırılarak PID kontrolörünün çıkış değerini yeniden hesaplaması sağlanır.

Yukarıda verilen kodu 1. motor örneğinden giderek açıklarsak;

- **potvalue1:** Kontrol altında tutulacak olan potansiyometre değerinin okunduğu değişkendir.
- **error1:** Referans değer ile girişin o andaki değeri arasındaki farkın tutulduğu değerdir.
- **m1_git:** Motorun ulaşması istenen potansiyometre değeridir.

PID kontrolörlerden error1, error2 ve error3 hata bilgileri çıkar. Bu bilgiler 'SetOutputLimits()' fonksiyonu ile (-255,255) aralığına şu şekilde sınırlandırılır:

```
pid1.SetMode(AUTOMATIC);
pid1.SetOutputLimits(-255,255);
pid2.SetMode(AUTOMATIC);
pid2.SetOutputLimits(-255,255);
pid3.SetMode(AUTOMATIC);
pid3.SetOutputLimits(-255,255);
```

Bu sınırlandırmanın yapılmasının amacı, error bilgilerinin motorlara PWM sinyali olarak verilecek olmasıdır. Bu şekilde hata miktarı arttıkça motorlar daha hızlı çalışacakken, hata miktarı azaldıkça motorlar da yavaşlayacak ve böylece iyi bir kontrol gerçekleştirilmiş olacaktır. PWM sinyalinde (-255,255) in karşılığı (-5V,5V) tur. Bu şekilde motorlara gidecek olan voltaj değeri $\pm 5V$ olarak sınırlandırılmış olur.

Ayrıca 'SetMode()' fonksiyonu kullanılarak PID kontrolörün otomatik ya da manüel olarak kullanılacağı belirtilir. Bu çalışmada PID kontrolör otomatik modda kullanılmıştır.

Robotta bulunan 2. ve 3. eklem (omuz ve dirsek eklemleri) en çok yüke maruz kalan eklemlerdir. Özellikle kol yere paralel konuma yaklaştıkça üzerine düşen yük miktarı artmaktadır. Bu nedenle bu eklemler için seçilen P, I, D parametreleri her şartta doğru bir kontrol sağlamamaktadır. Bunun önüne geçmek için, çeşitli açı setleri oluşturulmuş ve bu açı setleri içerisinde diğer açı setlerinden bağımsız olarak P, I, D parametreleri belirlenmiştir. Böylece robotun her açı değerinde, optimal şekilde hareket etmesi amaçlanmıştır. Parametrelerin tespitinde deneysel yöntemler kullanılmıştır.

Bulunan bu parametrelerin, PID kontrolöre gönderilmesi için 'SetTunings()' fonksiyonu kullanılmıştır. Aşağıda bu fonksiyonun kullanımı verilmiştir:

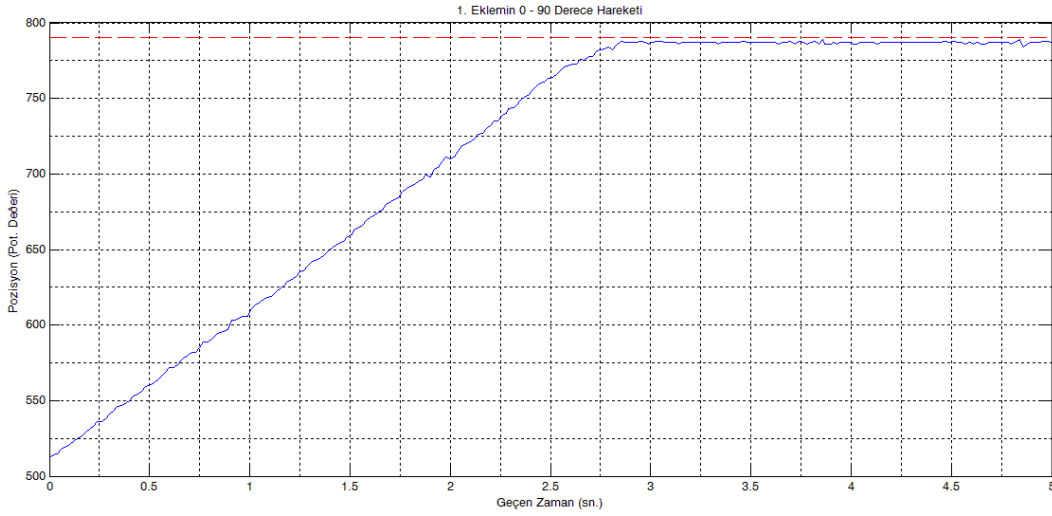
```
pid2.SetTunings(Kp2,Ki2,Kd2);
pid3.SetTunings(Kp3,Ki3,Kd3);
```

Arduino mikrodeneleyicisine yazılan kodun tam hali, ekler bölümünde verilmiştir.

3.6 Robotta Bulunan Motorların Hareket Eğrileri

Robottaki eklemleri tahrik etmek için kullanılan motorlar, Arduino içerisinde oluşturulan PID kontrolörler ile kontrol edilmiştir. Motorların, verilen bir referans değere hareketi sırasında ortaya çıkan (pozisyon - zaman) grafikleri Şekil 3.22-24-24'te verilmiştir.

3.6.1 1. Motorun hareket eğrisi



Şekil 3.22 1. Eklemin 0 - 90 derece hareketi

Şekil 3.22'de görüldüğü gibi, 1. eklem 0 derece (pot. değeri 512) pozisyonundan harekete başlamış ve 90 derece (pot. değeri 790) pozisyonunda hareketini tamamlamıştır. Ekleme bağlı motor, bu hareket sırasında Tablo 3.10'daki parametreleri kullanmıştır. Tabloda görüldüğü gibi D parametresi bu motorda kullanılmamış, PI kontrol yapılmıştır. Bunun nedeni, motorun PI kontrol ile optimal bir şekilde kontrol edilebilmiş olmasıdır.

Tablo 3.10 1. Motor için parametreler

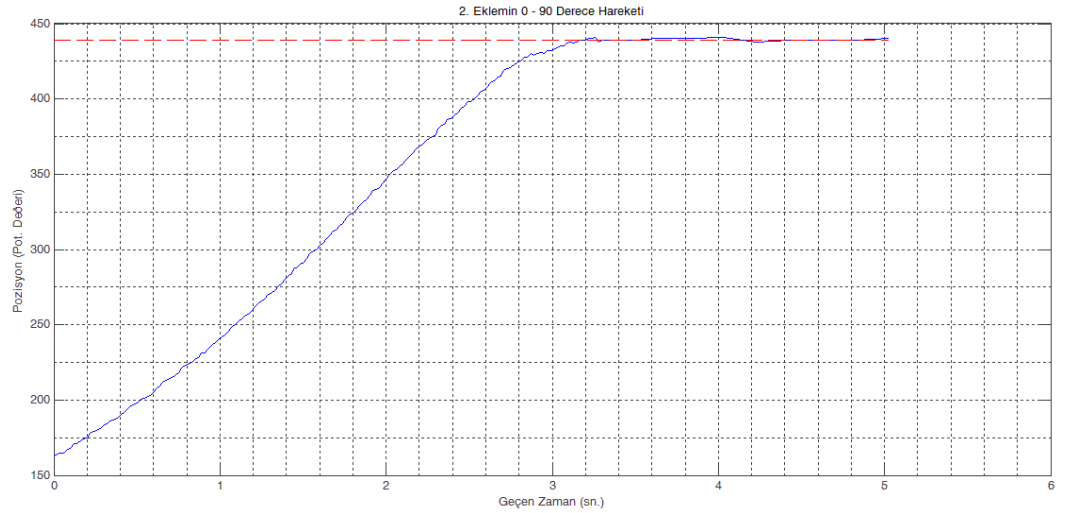
P	7
I	0.01
D	0

Oluşan grafiği incelersek,

- Aşım oluşmamıştır.
- Oturma zamanı yaklaşık 2.8 sn. dir.
- Çok küçük bir kalıcı durum hatası vardır (1 derece civarı)

Motor istenilen pozisyona kısa bir sürede ve tatmin edici bir doğrulukta oturmuştur. Parametreler seçilirken aşım oluşmamasına özen gösterilmiştir. Aksi takdirde osilasyon oluşmakta ve robotun verilen referansa oturma süresi uzamaktadır. Oluşan kalıcı durum hatası, 1 derecenin altında bir değer olduğu için robotun doğruluğuna etkisi tolere edilebilir miktarlardadır.

3.6.2 2. Motorun hareket eğrisi



Şekil 3.23 2. Eklemin 0 - 90 derece hareketi

Şekil 3.23'de görüldüğü gibi, 2. eklem 0 derece (pot. değeri 163) pozisyonundan harekete başlamış ve 90 derece (pot. değeri 439) pozisyonunda hareketini tamamlamıştır. Ekleme bağlı motor, bu hareket sırasında Tablo 3.11'deki parametreleri kullanmıştır.

Tablo 3.11 2. motor için parametreler

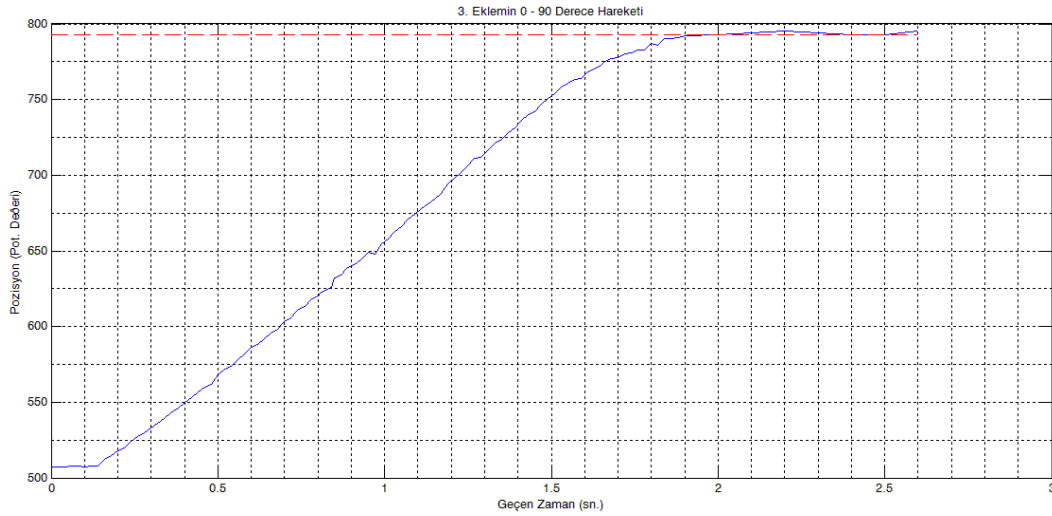
P	7
I	0.02
D	0.2

Oluşan grafiği incelersek,

- Aşım oluşmamıştır.
- Oturma zamanı yaklaşık 3.2 sn. dir.
- Kalıcı durum hatası yoktur.

1. eklem robotun bel eklemidir. Robotun tüm ağırlığını üzerinde taşımasına rağmen, yaptığı hareketin yönünden dolayı yer çekiminden ve ağırlıktan etkilenmemektedir. Ancak 2. eklem için bu durum daha farklıdır. 2. eklem, robotun omuz eklemidir ve kendi ağırlığına ek olarak 3. eklem (dirsek) ve bileğin ağırlığını da taşımaktadır. Ayrıca 2. eklemin hareketi yer çekimine karşı olduğu için eklemlerin ağırlığından dolayı zaman içerisinde bir miktar kaymalar yaşanmaktadır. Oluşan bu hatanın ve bu hataya karşı üretilen PWM sinyalinin büyüklüğü eklemi harekete geçirebilecek dereceye geldiğinde, eklem referans değere tekrar oturmaya çalışır.

3.6.3 3. Motorun hareket eğrisi



Şekil 3.24 3. Eklemin 0 - 90 derece hareketi

Şekil 3.24'de görüldüğü gibi, 3. eklem 0 derece (pot. değeri 512) pozisyonundan harekete başlamış ve 90 derece (pot. değeri 793) pozisyonunda hareketini tamamlamıştır. Ekleme bağlı motor, bu hareket sırasında Tablo 3.12'deki parametreleri kullanmıştır.

Tablo 3.12 3. motor için parametreler

P	5
I	0.25
D	0.5

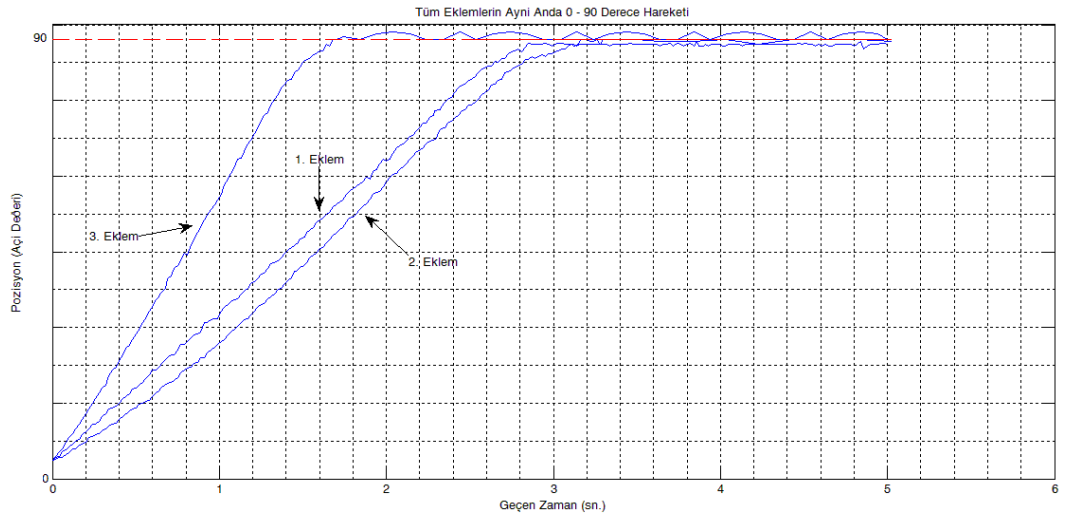
Oluşan grafiği incelersek,

- Aşım oluşmamıştır.
- Oturma zamanı yaklaşık 1.9 sn. dir.
- Kalıcı durum hatası yoktur.

2. eklem için bahsedilen yer çekiminin etkisiyle kayma durumu, 3. eklem için de geçerlidir ancak 3. eklem taşıdığı yükün daha hafif olmasından dolayı, kayma daha uzun sürelerde gerçekleşmektedir ve eklem kendini toparlaması daha kolay olmaktadır.

3.6.4 Tüm eklemlerin aynı anda hareketi

Tüm eklemlere aynı anda, sıfır konumundan (tüm açılar 0), $\theta_1 = \theta_2 = \theta_3 = 90$ derece konumuna hareket etmesi için bir girdi verilirse, Şekil 3.25'deki hareket grafikleri elde edilir.



Şekil 3.25 Tüm eklemlerin aynı anda 0 - 90 derece hareketi

Verilen grafiklerde anlık olarak kimi bölgelerde dalgalanmalar göze çarpmaktadır. Gerçekte bu dalgalanmalar olmamasına karşın, pozisyon bilgilerinin potansiyometrelerden analog olarak okunması nedeniyle gürültü de işin içine girmiştir. Robotun hareketi sırasında bu gürültüler çok etkili olmamaktadır. Ancak robot bir konuma oturduktan sonra, analog okumada sürekli olarak gerçekleşen dalgalanmalar nedeniyle meydana gelen hatalar, PID kontrolörün I parametresi tarafından toplanmakta ve bir süre sonra sistemin hareket etmesine neden olmaktadır. Bu durum sistemin kararlılığını önemli ölçüde etkilemektedir. Bazı yumuşatma fonksiyonları ile gürültüler azaltılsa da, sonunda yine kalıcı durum hatalarının oluşmasının önüne geçilememiştir. Bu nedenle yapılabilecek en iyi çözüm, geri besleme bilgisinin potansiyometre ile değil, enkoder ile alınmasıdır. Enkoder dijital bilgi veren bir cihaz olduğu için gürültülerden etkilenmemekte ve çok daha hassas çalışmaktadır. Ayrıca çözünürlüğü de potansiyometreye göre daha yüksektir. Tüm bunlara karşın, çalışmada kullanılan TQ MA2000 robot manipülatöründeki tüm motorların pozisyon geri beslemeleri potansiyometre ile alınmıştır. Robot kapalı bir yapıda olduğu için potansiyometreler enkoder ile değiştirilememiştir.

4. SONUÇLAR VE TARTIŞMA

Bu tez çalışmasında, altı eklemlili TQ MA2000 robot manipulatörünün çalışma uzayı içerisindeki, daha önceden özellikleri tanımlanmış nesneleri görüntü işleme yöntemleri ile algıladıktan sonra bulunduğu konumdan alıp başka bir konuma taşıması işlemi gerçekleştirilmiştir.

Çalışmanın görüntü işleme aşamasında, piksel uzayından gerçek uzaya geçebilmek yani 1 pikselin gerçekteki uzunluğunu hesaplayabilmek için aralarındaki uzaklığın bilindiği 2 adet referans nesne kullanılmıştır. Yapılan görüntü işleme analizi sonucunda, x ve y eksenlerindeki hatanın yaklaşık 0.1 - 0.3 cm civarında olduğu görülmüştür. Taşınacak nesneler zeminde bulunduğu için z ekseninin görüntü işleme ile bulunmasına gerek olmamıştır, bu nedenle tek bir kameranın kullanılması yeterli olmuştur.

Çalışmanın yapıldığı ortamdaki aydınlatma sabitlenemediği için, işlenecek olan görüntü üzerinde gürültüler meydana gelmiştir. Görüntü işleme analizi için seçilmiş olan renk uzayı HSV' nin V değeri parlaklık ile doğrudan ilgilidir. Önceden belirlenmiş olan ve taşınacak nesnelerin özelliklerini belirten HSV parametreleri, anlık olarak değişebilen ortam aydınlığı nedeni ile çok nadir de olsa nesnenin algılanmasını sağlayamamıştır. Ortam ışığının sabitlenmesi ya da bunun mümkün olmadığı durumlarda kullanıcıdan bağımsız olacak şekilde ve ortam ışığının değişimlerine uyum sağlayacak adaptif filtreleme algoritmaları geliştirilebilir.

Robotun ileri kinematik analizi için Denavit - Hartenberg yöntemi kullanılmıştır. Bu yöntemin temel adımı olan DH parametreleri ve ileri yön matrisleri bulunarak ileri kinematik analizi gerçekleştirilmiştir. Robotun bilek kısmı, eklem kaçıklıklı ve artık yapılı olması nedeniyle kullanılmamış, ana eklemleri olan ilk üç eklem (bel, omuz, dirsek) kullanılmıştır. Robotun bilek kısmı olmadan da bu çalışmada istenilen işlemlerin gerçekleştirilebildiği görülmüştür. Bunun sebebi pozisyon kontrolü yaparak, oryantasyona gerek duyulmadan taşınacak nesnelere ulaşabilmemizdir. Ters kinematik analizi için analitik çözüm yöntemi tercih edilmiştir. Bu yöntem ile yapılan çözümler kesin cevaplar verse de, tüm robot tiplerinde kullanılmaya uygun değildir. Özellikle bu robottaki gibi eklem kaçıklıklı bileklerde analitik yöntem ile çözümler bulunamamaktadır. Bu yöntem yerine, ileri kinematik çözümünden yola çıkılarak arama tablosu yöntemi ya da iteratif çözüm yöntemleri tercih edilebilir.

Robot kolu hareketi gerçekleştirilirken, uç işlevcisinin hedef noktaya tam doğrulukla yaklaşmadığı ve hedef noktada sabit kalamadığı görülmüştür. Bunun nedeni, robot eklemlerinin pozisyon geri besleme elemanı olan potansiyometrelerin hassasiyetinin düşük olması ve bu potansiyometrelerin analog olarak okunması sırasında karşılaşılan gürültülerdir. Bu gürültüler, PID kontrolöre hata olarak girmekte ve kontrolör biriken bu hatalara bir süre sonra tepki göstermektedir. Bu güçlüklerin aşılabilmesi için en doğru yol, pozisyon geri beslemeleri için enkoder kullanılmasıdır. Enkoder hem çok daha yüksek çözünürlüğe sahip bir elemandır, hem de dijital olarak okunduğu için gürültülerden etkilenmemektedir. Ancak çalışmada kullanılan robotun potansiyometrelerinin bulunduğu bölümlerin açılmayacak yapıda olması nedeniyle potansiyometreler enkoder ile değiştirilememiştir. Mevcut sistemde robotun hedefe yaklaşma doğruluğu %90-95 civarlarındadır.

Çalışmada kullanılan TQ MA2000 robot manipülatörünün mekaniksel ve elektriksel özelliklerin bilinmemesinden ötürü, robotun hız kinematiği ve dinamiği çıkartılamamıştır.

Yapılabilecek sonraki çalışmalarda, robotun bilek kısmı da kullanılarak sisteme oryantasyon kabiliyeti eklenebilir. Tüm robot yapılarında uygulanamayan analitik çözüm yöntemi yerine sayısal çözüm yöntemleri tercih edilebilir. Robotun eklemlerinin pozisyon geri beslemesi için potansiyometre yerine enkoder tercih edilebilir. Zeminin düz olmadığı çalışmalar için sisteme ilave edilecek ikinci bir kamera ile derinlik algısı elde edilebilir. Böylece nesnenin z eksenindeki konumu da ölçülebilir. Görüntü işleme sırasında bozucu etki olan ortam ışığının etkisinin azaltılması için adaptif filtreleme yöntemleri uygulanabilir.

Bu çalışma, robotik ve görüntü işlemeye dayalı robot kontrol uygulamalarında eğitim amaçlı kullanılmaya elverişli durumdadır.

KAYNAKLAR DİZİNİ

- Akec, J.A., Steiner, S.J., Stenger, F.**, 1998, An Experimental Visual Feedback Control System for Tracking Applications Using a Robotic Manipulator, Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society, Vol.2, Aachen, Germany, 1125- 1130 p.
- Albayrak, M.**, 1997, Üç Boyutlu Uzayda Hareket Edebilen Robot Kolun, Bilgisayar Destekli Kontrolü, Tasarımı ve Uygulaması, Yüksek Lisans Tezi, Gazi Üniversitesi Fen bilimleri Enstitüsü, 67 s.
- Ali, H., Seng, T.C., Hoi, L.H., Elshaikh, M.**, 2012, Development of Vision-Based Sensor of Smart Gripper for Industrial Applications, IEEE 8th International Colloquium on Signal Processing and its Applications, Melaka, 300-304 p.
- Alpat, A.**, 2012, “Arduino Mega 2560’a Genel Bakış”, <http://arduinoturkiye.com/arduino-mega-2560-nedir>, (Erişim Tarihi : 11.05.2015)
- Al-Saedi, F.A.T., Mohammed A.H.**, 2012, Design and Implementation of PSO-PID Controller for MA2000 Robotic Manipulator, IJCSET, Vol 2 Issue 9, 1427-1433 p.
- Anh, L.T., Song, J.B.**, 2010. Object Tracking and Visual Servoing using Features Computed from Local Feature Descriptor, International Conference on Control, Automation and Systems, Gyeonggi-do, Korea, 1044-1048 p.
- Basile, F., Caccavale, F., Chiacchio, P., Coppola, J., Curatella, C.**, 2012, Task-Oriented Motion Planning for Multi-Arm Robotic Systems, Robotics and Computer-Integrated Manufacturing, 569-582 p.
- Bayrak, A.**, 2007, Beş Eksenli Bir Robot Kolunun Simülasyonu ve Kontrolü, Yüksek Lisans Tezi, Gazi Üniversitesi Fen Bilimleri Enstitüsü, 101 s.
- Bayrak, A., Sarıtaş, M.**, 2008, Beş Eksenli Bir Robot Kolu Simülasyonu ve Engel / Hedef Takibi, ELECO, Bursa, 1-5 s.
- Bejo, A., Pora, W., Kunieda, H.**, 2009, Development of a 6-Axis Robotic Arm Controller Implemented on Low-Cost Microcontroller, 6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Pattaya, Choburi, 328-331 p.

KAYNAKLAR DİZİNİ (devam)

- Bingül, Z., Ertunç, H.M., Oysu, C.,** 2005, Comparison of Inverse Kinematics Solutions Using Neural Network for 6R Robot Manipulator with Offset, Computational Intelligence Methods and Applications, ICSC Congress, Vol. 0, Istanbul, Turkey, 5 p.
- Bingül Z., Küçük S.,** 2009, Robot Kinematığı, Birsen Yayınevi, İstanbul, 348s.
- Bonev, I.,** 2013, “The truth about Foxconn’s Foxbot industrial robots”, <http://coro.etsmtl.ca/blog/?p=73> (Erişim tarihi: 03 Mayıs 2015)
- Bulut, H.,** 2002, Control and Animation of a Three Link Robotic Arm Using Direct Kinematics and Inverse Kinematics Methods, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, 81 s.
- Bustamante, L., Gu, J.,** 2007, Localization of Electrical Outlet for a Mobile Robot using Visual Servoing, Canadian Conference on Electrical and Computer Engineering, Vancouver, BC, Canada, 1211-1214 p.
- Carusone, J., D' Eleuterio, G.M.T.,** 1998, The Feature CMAC: A Neural-Network-Based Vision system for Robotic Control, Proceeding of the IEEE International Conference on Robotics and Automation, Vol.4, Downsview, Ont., Canada, 2959-2964 p.
- Chang, H.C., Shih, T.M.,** 2008, Visual Servo Control of a Three Degree of Freedom Robotic Arm System, IEEE International Conference on Advanced Robotics and its Social Impacts, Taipei, Taiwan, 1-6 p.
- Choi, W., Ryu, C., Kim, H.,** 1999, Navigation of a Mobile Robot using Mono-Vision and Mono-Audition, Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Vol.4, Tokyo, Japan, 686-691 p.
- Cisneros, M.A.P.,** 2004, Intelligent Model Structures In Visual Servoing , The Degree of Doctor of Philosophy, University of Manchester, Institute of Science and Technology - UMIST, Manchester, England, 320 p.
- Çelikel, H. K.,** 2006, Taşıt Süspansiyon Sistemlerinin Bulanık Mantık Ayarlı PID Kontrolü, Yüksek Lisans Tezi, İstanbul Üniversitesi Fen Bilimleri Enstitüsü, 76 s.
- Çengelci, B., Çimen, H.,** 2005, Endüstriyel Robotlar, Makine Teknolojileri Elektronik Dergisi, 2005 (2), Türkiye, 69-78 s.

KAYNAKLAR DİZİNİ (devam)

- Çiçek, S.**, 2006, Renge Göre (Kırmızı, Yeşil, Mavi) Malzeme Taşıyan Robot Kolu Tasarımı Ve Uygulaması, Yüksek Lisans Tezi, Gazi Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Bilgisayar Eğitimi, 144 s.
- Craig, J.J.**, 1989, Introduction to Robotics: Mechanics and Control, USA, 400 p.
- Daş, M.T.**, 2003, Motion Control of a SCARA Robot With a PLC Unit, Gaziantep Üniversitesi Fen Bilimleri Enstitüsü, 49 s.
- Deepthi, R.S., Sankaraiah, S.**, 2011, Implementation of Mobile Platform Using Qt and Open CV for Image Processing Applications, IEEE Conference on Open Sysyems (ICOS), Langkawi, Malaysia, 284-289 p.
- Dinh, B.H., Dunnigan, M.W., Reay, D.S.**, 2008, A practical approach for position control of a robotic manipulator using a radial basis function network and a simple vision system, WSEAS Transaction on Systems and Control, Vol.3, 289-298 p.
- Doignon, C., Abba, G., Ostertag, E.**, 1994, Recognition and Localization of Solid Objects by a Monocular Vision System for Robotics Tasks, Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, Vol.3, Munich, Germany, 2007-2014 p
- Duran, M. A.**, 2007, Puma Tipi Bir Manipulatörün Kontrolü, Yüksek Lisans Tezi, Selçuk Üniversitesi Fen bilimleri Enstitüsü, 84 s.
- Endüstri Mühendisliği Blogu**, 2012, “Endüstriyel Robotlar”, <http://endustrimuhendisligi.blogspot.com.tr/2012/12/endustriyel-robotlar.html>, (Erişim Tarihi: 25 Nisan 2015)
- Erşan, E.**, 2009, Bir Robotun Görmeye Dayalı Kontrolü, Yüksek Lisans Tezi, Marmara Üniversitesi Fen Bilimleri Enstitüsü, 90s
- Huang, G.S.**, 2011, Robotic Arm Grasping and Placing Using Edge Visual Detection System, 8th Asian Control Conference (ASCC), Kaohsiung, Taiwan, 960-964 p.
- Jain, R.C., Kasturi, R., Schunck B.G.**, 1995. **Machine Vision**, ISBN 007-0320-187, Mc Graw-Hill.
- Jantzen, J.**, 1998, Tuning Of Fuzzy PID Controllers, Technical University of Denmark, Department of Automation, Denmark, 22 p.

KAYNAKLAR DİZİNİ (devam)

- Kert, M.**, 2006, Gerçek Görüntüden Elde Edilen Koordinatlarla Robot Kol Hareket Optimizasyonu, Yüksek Lisans Tezi, Mustafa Kemal Üniversitesi Fen Bilimleri Enstitüsü, 70 s.
- Koca, H.**, 2007, Robot Manipülatör Denetimi, Yüksek Lisans Tezi, Gazi Üniversitesi Fen Bilimleri Enstitüsü Elektrik Elektronik Mühendisliği, 108s.
- Kuran, S. B.**, 2007, Yapay Görme Kılavuzlu Robot ile Gıda Tutma ve Paketleme, Yüksek Lisans Tezi, Eskişehir Osmangazi Üniversitesi Fen Bilimleri Enstitüsü, 76 s.
- Luo, R.C., Tsai, C.H., Lai, C.C., Chang, C.M.**, 2008, Autonomous Security Robot Services Using Eye-in Hand Visual Servo Systems, IEEE International Conference on Advanced Robotic and its Social Impacts, Taipei, Taiwan, 1-6 p.
- Menevşe, S.**, 2013, “Endüstriyel Robotlar & Sınıflandırılması”, <http://www.elektrikport.com/teknik-kutuphane/endustriyel-robotlar-siniflandirilmesi/11295> (Erişim Tarihi: 26 Nisan 2015)
- Modi, K.P, Sahin, F., Saber, E.**, 2005, An Application of Human Robot Interaction: Development of a Ping-Pong Playing Robotic Arm, IEEE International Conference on System, Man and Cybernetics, Vol.2, Waikoloa, HI, USA, 1831-1836 p.
- Mufioz, V.F., Garcia-Morales, I., Morales, J., Gomez-DeGabriel, J.M., Fernandez- Lozano, J., Garcia-Cerezo, A.**, 2004, Adaptive Cartesian Motion Control Approach for a Surgical Robotic Cameraman, Proceeding of the IEEE International Conference on Robotics and Automation, New Orleans, LA, USA, April, 3069-3070 p.
- Mundhra, K., Suluh, A., Sugar, T., McBeath, M.**, 2002, Intercepting a Falling Object: Digital Video Robot, Proceeding of the IEEE International Conference on Robotics and Automation, Washington DC, USA, 2060-2065 p.
- Ogawa, M., Shimizu, S., Kadogawa, T., Hashizume, T.**, 2011, Development of Air Hockey Robot Improving with the Human Players-Arm Control of Effective Quick Draw Taking An Opponent’s Eye Movement into Account, 37th Annual Conference on IEEE Industrial Electronics Society (IECON), Melbourne, VIC, 3364-3369 p.

KAYNAKLAR DİZİNİ (devam)

- Özcan, T.**, 2010, Hareketli Nesnelerde Yüz Tespitine Yönelik Bir Uygulama, Yüksek Lisans Tezi, Trakya Üniversitesi Fen Bilimleri Enstitüsü, 99 s.
- Peker, M.**, 2009, Görüntü İşleme Tekniği Kullanılarak Gerçek Zamanlı Hareketli Görüntü Tanıma, Yüksek Lisans Tezi, Sakarya Üniversitesi Fen Bilimleri Enstitüsü, 94 s.
- Quinn, C.**, 2013, Robotic Arm Controller, Dundalk Institute of Technology Sustainable Design Engineering, 61 p.
- Rapp, H.H.**, 2011. A Ping-Pong Ball Catching and Juggling Robot, a Real-Time Framework for Vision Guided Acting of an Industrial Robot Arm, Proceeding of the 5th International Conference on Automation Robotics and Applications, Wellington, New Zealand, 430-435 p.
- Saygılı, Ç.**, 2006, Scara Tipi Bir Robotun Tasarımı ve Animasyonu, Yüksek Lisans Tezi, Selçuk Üniversitesi Fen bilimleri Enstitüsü, 136 s.
- Seelye, M., Gupta, G.S., Seelye, J., Mukhopadhyay, S.C.**, 2010, Camera-in-hand Robotic System for Remote Monitoring of Plant Growth in a Laboratory, IEEE Instrumentation and Measurement Technology Conference (I2MTC), Austin, TX, 809-814 p.
- Semetay, C.**, 2001, Robot kontrolü ve kinematiği, Yüksek Lisans Tezi, İstanbul Üniversitesi Fen Bilimleri Enstitüsü, Makine Mühendisliği Anabilim Dalı, 106 s.
- Sun, D., Mills , J.K.**, 1999, High-accuracy trajectory tracking of industrial robot manipulator using adaptive-learning schemes, American Control Conference, Vol3, San Diego, USA, 1935-1939 p.
- Şahin, Y.**, 2006, SCARA Tip Bir Robotun Yörünge Kontrolünde PID Kontrol Uygulaması, Yüksek Lisans Tezi, Selçuk Üniversitesi Fen Bilimleri Enstitüsü Makine Anabilim Dalı, 71s
- T.C. Milli Eğitim Bakanlığı**, 2012, Tekstil Teknolojisi - Robotik , Ankara, 101s
- Tonbul, T. S.**, 2002, Beş Eksenli Bir Robot Kolunun Ters Kinematik Hesaplamalarının Ve Yörünge Planlamasının Yapılması, Yüksek Lisans Tezi, Gazi Üniversitesi Fen Bilimleri Enstitüsü, 103 s.

KAYNAKLAR DİZİNİ (devam)

- Van Delden, S., Farr, R., Hensley, S.,** 2007, An Automated Camera Orientation Recovery Algorithm for Eye-in-hand Robotic Manipulator, IEEE International Workshop on Robotic and Sensors Environments, Ottawa, Ont., Canada, 1-6 p.
- Wei, Z., Cai, L.,** 2010, Design for Motion Detection System Based on Embedded Linux, International Conference on Multimedia Technology, Ningbo, 1-3 p.
- Wikipedia Contributors,** 2001, “Three Laws of Robotics”, Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/wiki/Three_Laws_of_Robotics, (Erişim Tarihi: 25 Nisan 2015)
- Wikipedia Contributors,** 2002, “R.U.R.”, Wikipedia, The Free Encyclopedia, <http://en.wikipedia.org/wiki/R.U.R.>, (Erişim Tarihi: 25 Nisan 2015)
- Wilson, R.,** 2000, “Grey Level Segmentation”, <http://www.dcs.warwick.ac.uk/~rgw/sira/node7.html> (Erişim Tarihi: 04 Mayıs 2015)
- Ziegler, J. G., Nichols, N. B.,** 1942, Optimum Settings for Automatic Controllers, Transactions of the ASME, Vol. 64, 759 - 768 p

ÖZGEÇMİŞ

Adı Soyadı : Ekrem YAVUZ

Doğum tarihi : 20/09/1990

Doğum yeri : Ödemiş/İZMİR (TR)

Adres : Erzene Mah. 56 Sokak No:24 Umut Apt. A Blok Daire:1
Bornova/İZMİR

Telefon : (+90) 5069302694

E-mail : ekrem.yavuz@ege.edu.tr
ekremyavuz35@windowslive.com

Eğitim :

- 2012–2015 Ege Üniversitesi Fen Bilimleri Enstitüsü
Mekatronik Anabilim Dalı
CGPA: 3.71/4
- 2008–2012 Kocaeli Üniversitesi Mekatronik
Mühendisliği Bölümü
CGPA: 2.95/4
- 2004–2008 Aydın Ortaklar Anadolu Öğretmen Lisesi
Grade: 88.2/100

İş Deneyimi :

- İş Yeri : LST - Lazer Sistemleri Teknolojisi
Pozisyon : Yazılım Mühendisi
Çalışılan Tarih : 17.12.2012 - 12.09.2013
- İş Yeri : Ege Üniversitesi Fen Bilimleri
Enstitüsü Mekatronik Anabilim
Dalı

Pozisyon : Arařtırma Grevlisi
Çalışılan Tarih : 27.12.2013 - Halen

Çalışmalar :

- Erol Uyar, Nail Akçura, Ekrem Yavuz, Mcahid Candan, “3 Eksenli CNC İşleme Tezgahı Tasarımı ve Kontrol”, TOK 2014, Kocaeli niversitesi, Bildiri no:16, 34-37 s.
- Nail Akçura, Erol Uyar, Mcahid Candan, Ekrem Yavuz, “Çevre Haritalandırma ve Obje Geçme Algoritması Entegreli Bir Mobil Aracın Bulanık Mantık ile İç Mekan Navigasyon Kontrol”, TOK 2014, Kocaeli niversitesi, Bildiri no:122, 563-566 s.
- Mcahid Candan, Erol Uyar, Nail Akçura, Ekrem Yavuz, “Remote Controlled Electro-Pneumactical Climbing Robot For Cleaning Of Skyscrapers”, International Journal of Computer and Information Technology, Volume 04 – Issue 01, 75 - 78 p., January 2015
- Erol Uyar, Mcahid Candan, Ekrem Yavuz, Nail Akçura, 2014, “Dzlemsel Elektro-Pnmatik Maniplatr Tasarımı Ve Kontrol”, HPKON 2014, İstanbul, 567 - 581 s.
- Erol Uyar, Ekrem Yavuz, Mcahid Candan, Nail Akçura, 2015, “Dzlemsel Elektro-Pnmatik Maniplatr Tasarımı Ve Kontrol”, Mhendis ve Makina, Cilt 56 Sayı 662 s:56-62, Mart 2015.

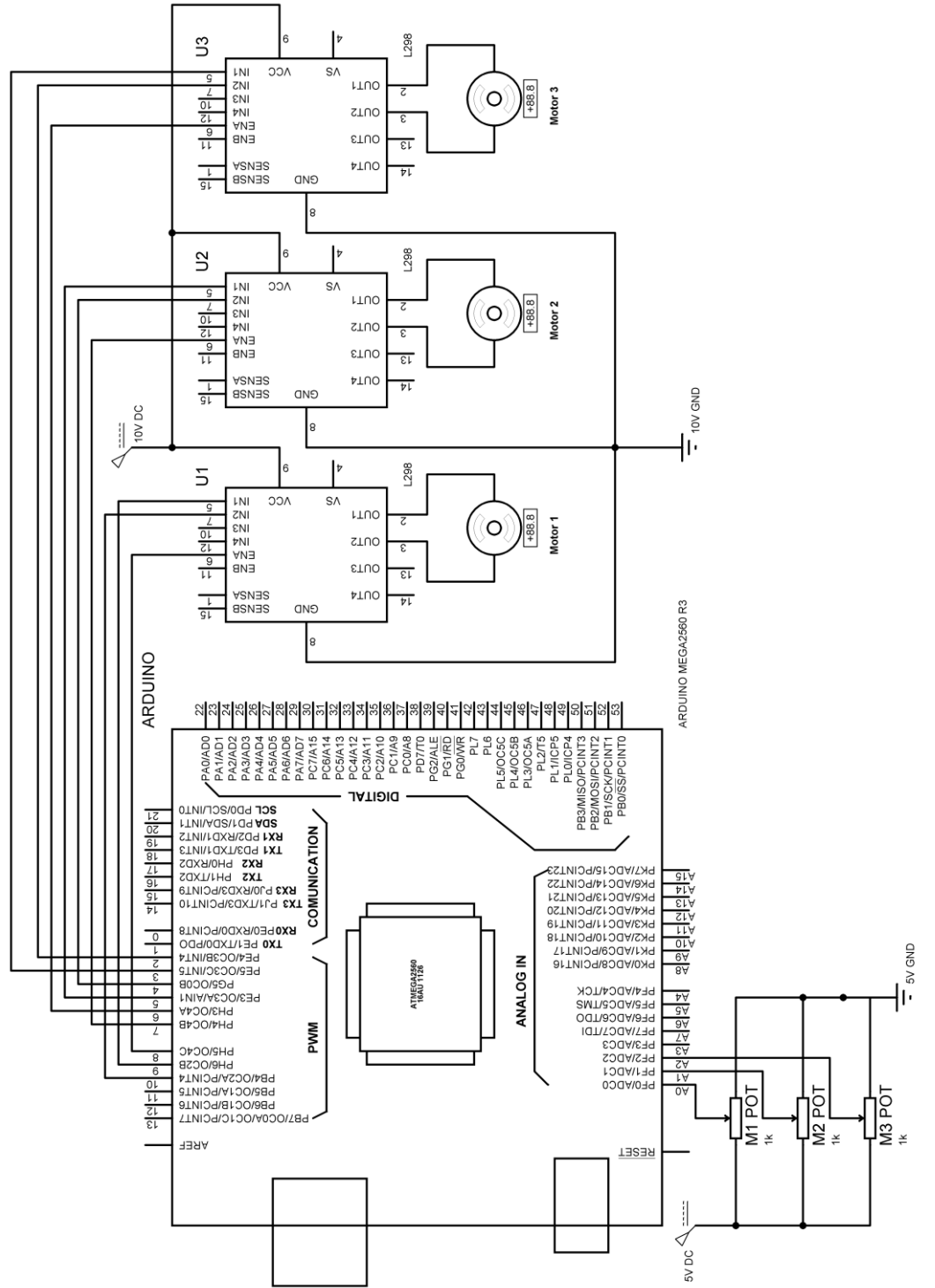
EKLER

Ek 1 Robot ve Kontrol Elemanlarının Bağlantı Şeması

Ek 2 Görüntü İşleme ve Ters Kinematik Analizi Yazılımı

Ek 3 Arduino Robot Kontrol Yazılımı

Ek 1 Robot ve Kontrol Elemanlarının Bağlantı Şeması



Ek 2 Görüntü İşleme ve Ters Kinematik Analizi Yazılımı

```
#include <iostream>
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <cmath>
#include <fstream>
#include <cstdlib>
#include <cstdio>
#include "SerialClass.h"
#include <time.h>
#include <string>
#include <sstream>

using namespace cv;
using namespace std;

int main( int argc, char** argv )
{
    namedWindow("Object View", CV_WINDOW_AUTOSIZE);
    VideoCapture webcam(0);      // Webcam Secimi
    if(!webcam.isOpened())
    {
        printf("Hata - WebCam mesgul veya takili degil!!");
        return -1;
    }
    webcam.set(CV_CAP_PROP_FRAME_HEIGHT,720); // görüntü
yüksekligi ayarla
    webcam.set(CV_CAP_PROP_FRAME_WIDTH,720);  // görüntü genisligi
ayarla

    double ref_reel_uzaklik = 0;

    cout << "Lutfen referans objelerini koselere yerlestirip
[Enter]'a basiniz...";
    getchar();
    cout << "Lutfen referans objelerinin olusturdugu karenin kenar
uzunlugunu giriniz: ";
    cin >> ref_reel_uzaklik;
    cout << endl;

    //Referans Algilama Bolumu
    //Referans Kapak

    Mat img;
    webcam.read(img);
    int H_ref_Alt = 90; // Hue Alt Siniri
    int H_ref_Ust = 146; // Hue Üst Siniri

    int S_ref_Alt = 160; // Satürasyon Alt Siniri
    int S_ref_Ust = 255; // Satürasyon Üst Siniri

    int V_ref_Alt = 47; // Parlaklik Alt Siniri
    int V_ref_Ust = 255; // Parlaklik Üst Siniri

    Mat img_hsv;
    cvtColor(img,img_hsv,COLOR_BGR2HSV);
    Mat ref_thres_img;
```

```

    inRange(img_hsv, Scalar(H_ref_Alt, S_ref_Alt, V_ref_Alt),
    Scalar(H_ref_Ust, S_ref_Ust, V_ref_Ust), ref_thres_img);

    Mat ref_filtered_img;
    int ref_erosion_size = 3; // filtre gücünü ayarla
    Mat ref_element = getStructuringElement( MORPH_ELLIPSE, Size(
2*ref_erosion_size + 1, 2*ref_erosion_size+1 ), Point(
ref_erosion_size, ref_erosion_size ) );
    erode(ref_thres_img, ref_filtered_img, ref_element);
    Mat ref_algilanan_kenarlar;

    Canny(ref_filtered_img, ref_algilanan_kenarlar, 60, 255); //
kenar algilat

    //agirlik merkezlerini bul
    vector<vector<Point> > ref_algilanan_kontourler;

    findContours(ref_algilanan_kenarlar, ref_algilanan_kontourler, CV_RE
TR_TREE, CV_CHAIN_APPROX_SIMPLE, Point(0, 0));

    vector<Moments>
ref_momentler(ref_algilanan_kontourler.size());
    for(int i = 0; i < ref_algilanan_kontourler.size(); i++ )
    {
        ref_momentler[i] = moments(ref_algilanan_kontourler[i]);
    }

    int ref_x[ref_algilanan_kontourler.size()];
    int ref_y[ref_algilanan_kontourler.size()];
    Point2d ref_noktalar[ref_algilanan_kontourler.size()/2];
    for(int i = 0; i < ref_algilanan_kontourler.size(); i++)
    {
        ref_x[i] = ref_momentler[i].m10/ref_momentler[i].m00; //
m10 = x , m01 = y
        ref_y[i] = ref_momentler[i].m01/ref_momentler[i].m00;
    }

    int ref_x_top = 0;
    int ref_y_top = 0;
    double ref_x_fark = 0;
    double ref_y_fark = 0;
    double ref_pixel_uzunlugu = 0;
    double ref_oranti = 0;

    // bulunan referans objeleri numaralandir
    for(int i = 0; i < ref_algilanan_kontourler.size(); i++)
    {
        if(i%2 == 0)
        {
            printf("%d.Referans:\n", i/2);
            printf("X%d: %d\t\tY%d:
%d\n", i/2, ref_y[i], i/2, ref_x[i]);
            ref_x_top += ref_x[i];
            ref_y_top += ref_y[i];
        }
    }

    // bulunan referans objelerinin orta noktalarını çiz
    for(int i = 0; i < ref_algilanan_kontourler.size(); i++)
    {
        int ref_noktasayisi = i/2;

```

```

        int ref_koordinat_noktaları = 2*ref_noktasayisi;
        ref_noktalar[ref_noktasayisi] =
Point2d(ref_x[ref_koordinat_noktaları],ref_y[ref_koordinat_noktala
ri]);

line(ref_algilanan_kenarlar,ref_noktalar[ref_noktasayisi],ref_nokt
alar[ref_noktasayisi],Scalar(255,255,255),4); // merkez çiz
    }

    cout << endl;
    // referans objelerin arasındaki uzaklığın orta noktasını bul
    ref_x_top /= 2;
    ref_y_top /= 2;
    ref_x_fark = abs(ref_x[2] - ref_x[0]);
    ref_y_fark = abs(ref_y[2] - ref_y[0]);
    ref_x_fark = pow(ref_x_fark,2);
    ref_y_fark = pow(ref_y_fark,2);
    ref_pixel_uzunlugu = std::sqrt(ref_x_fark + ref_y_fark);
    ref_oranti =
((ref_reel_uzaklik*std::sqrt(2))/(ref_pixel_uzunlugu));

    // referans objelere göre hesap yap, robotun orta noktasını
    // ve bir piksel uzunluğunu belirle
    cout << "Robot merkez (x,y): (" << ref_y_top << "," <<
ref_x_top << ")" << endl;
    cout << "Hesaplanan 1 piksel uzunluğu: " << ref_oranti <<
"cm" << endl << endl;

    // Nesne Algılama Bolumu
    // Nesne Turuncu Top

    int H_Alt = 19; // Hue Alt Siniri
    int H_Ust = 31; // Hue Üst Siniri

    int S_Alt = 55; // Satürasyon Alt Siniri
    int S_Ust = 255; // Satürasyon Üst Siniri

    int V_Alt = 243; // Parlaklık Alt Siniri
    int V_Ust = 255; // Parlaklık Üst Sinir

    Mat thres_img;
    inRange(img_hsv, Scalar(H_Alt, S_Alt, V_Alt), Scalar(H_Ust,
S_Ust, V_Ust), thres_img);

    Mat filtered_img;
    int erosion_size = 3; // filtre gücünü ayarla
    Mat element = getStructuringElement( MORPH_ELLIPSE, Size(
2*erosion_size + 1, 2*erosion_size+1 ), Point( erosion_size,
erosion_size ) );
    erode(thres_img,filtered_img,element);
    Mat algılanan_kenarlar;

    Canny(filtered_img,algılanan_kenarlar,60,255); // kenar
algılat

    //ağırlık merkezlerini bul
    vector<vector<Point>> algılanan_kontourler;

    findContours(algılanan_kenarlar,algılanan_kontourler,CV_RETR_TREE,
CV_CHAIN_APPROX_SIMPLE,Point(0,0));

```

```

vector<Moments> momentler(algilanan_kontourler.size());
for(int i = 0; i < algilanan_kontourler.size(); i++)
{
    momentler[i] = moments(algilanan_kontourler[i]);
}

double x[algilanan_kontourler.size()];
double y[algilanan_kontourler.size()];
Point2d noktalar[algilanan_kontourler.size()/2];
for(int i = 0; i < algilanan_kontourler.size(); i++)
{
    x[i] = momentler[i].m10/momentler[i].m00; // m10 = x , m01
= y
    y[i] = momentler[i].m01/momentler[i].m00;
}

// bulunan nesneleri numaralandir
for(int i = 0; i < algilanan_kontourler.size(); i++)
{
    if(i%2 == 0)
    {
        cout << i/2 << ".Nesne: " << endl;
        cout << "X" << i/2 << ": " << y[i] << "\t\t" << "Y" <<
i/2 << ": " << x[i] << endl;
    }
}

// bulunan nesnelerin orta noktalarini ciz
for(int i = 0; i < algilanan_kontourler.size(); i++)
{
    int noktasayisi = i/2;
    int koordinat_noktalari = 2*noktasayisi;
    noktalar[noktasayisi] =
Point2d(x[koordinat_noktalari],y[koordinat_noktalari]);

line(algilanan_kenarlar,noktalar[noktasayisi],noktalar[noktasayisi
],Scalar(255,255,255),4); // merkez çiz
}

double reel_uzaklik_x[algilanan_kontourler.size()];
double reel_uzaklik_y[algilanan_kontourler.size()];

//nesnelerin robotun merkezine olan uzakliklerini hesapla
for(int i = 0; i < algilanan_kontourler.size(); i++)
{
    if(i % 2 == 0)
    {
        reel_uzaklik_x[i] = (x[i] - ref_x_top)*ref_oranti;
        reel_uzaklik_y[i] = (y[i] - ref_y_top)*ref_oranti;
    }
}

//nesnelerin robotun merkezine olan uzakliklerini ekrana
bastir
for(int i = 0; i < algilanan_kontourler.size(); i++)
{
    if(i % 2 == 0)
        cout << endl << (i/2)+1 << ".Nesne x,y : " <<
reel_uzaklik_y[i] << " cm , " << reel_uzaklik_x[i] << " cm " <<
endl;
}

```

```

    }

    imshow("Object View",algılanan_kenarlar); // ekranda algılanan
    nesneleri goster
    namedWindow("Reference View",CV_WINDOW_AUTOSIZE);
    imshow("Reference View",ref_algılanan_kenarlar); // ekranda
    referans nesnelerini goster
    webcam.release(); //webcam i serbest birak
    cout << endl <<"Lutfen goruntu penceresinde [Enter]'a basiniz"
    << endl;
    while(waitKey(0) !=13){;}

    printf("Nesne dogru algilandi mi? [E/H] "); //nesnelerin
    tespiti edilmesi kontrolu
    char choice;
    cin >> choice;
    while(1)
    {
        if(choice == 'E' || choice == 'e') // dogru algilandi ise
        devam et
            break;
        else if(choice == 'H' || choice == 'h') //dogru
        algılanmadi ise bitir
            return -1;
    }

    double tetal[4];
    double teta2[4];
    double teta3[4];
    double py = reel_uzaklik_x[0]; // y uzakligi
    double px = reel_uzaklik_y[0]; //x uzakligi
    double pz = 2; // z uzakligi yani yukseklik
    tetal[0] = atan2(-px,py) + atan2(sqrt(pow(px,2) + (pow(py,2) -
16)), -4);
    tetal[1] = tetal[0];
    //tetal matrisi 4 elemanli olmasi icin cogaltildi.
    tetal[2] = atan2(-px,py) + atan2(-sqrt(pow(px,2) + (pow(py,2) -
16)), -4);
    tetal[3] = tetal[2];
    //tetal matrisi 4 elemanli olmasi icin cogaltildi.

    //tetal[0] icin teta3[0] ve teta3[1] bulunuyor.
    double a1 = -4.5;
    double b1 = 32;
    double c1 = 23;
    double d1 = pz-26;
    double e1 = px*cos(tetal[0])+py*sin(tetal[0]);
    double r1 = (pow(d1,2)+pow(e1,2)-pow(a1,2)-pow(b1,2) -
pow(c1,2))/(2*c1);
    teta3[0] = atan2(a1,b1)+ atan2(sqrt(pow(c1,2)+pow(d1,2) -
pow(r1,2)), r1);
    teta3[1] = atan2(a1,b1)+ atan2(-sqrt(pow(c1,2)+pow(d1,2) -
pow(r1,2)), r1);

    //teta3[0] icin teta2[0] bulunuyor
    double s1 = a1*cos(teta3[0])-b1*sin(teta3[0]);
    double t1 = b1*cos(teta3[0])+a1*sin(teta3[0])+c1;
    teta2[0] = atan2((s1*d1)+ (t1*e1), -(s1*e1)+(t1*d1));

    //teta3[1] icin teta2[1] bulunuyor
    double s2 = a1*cos(teta3[1])-b1*sin(teta3[1]);

```

```

double t2 = b1*cos(teta3[1])+a1*sin(teta3[1])+c1;
teta2[1] = atan2((s2*d1)+ (t2*e1),-(s2*e1)+(t2*d1));

//teta1[1] icin teta3[2] ve teta3[3] bulunuyor.
double e2 = px*cos(teta1[2])+py*sin(teta1[2]);
double r2 = (pow(d1,2)+pow(e2,2)-pow(a1,2)-pow(b1,2)-
pow(c1,2))/(2*c1);
teta3[2] = atan2(a1,b1)+ atan2(sqrt(pow(c1,2)+pow(d1,2)-
pow(r2,2)),r2);
teta3[3] = atan2(a1,b1)+ atan2(-sqrt(pow(c1,2)+pow(d1,2)-
pow(r2,2)),r2);

//teta3[2] icin teta2[2] bulunuyor
double s3 = a1*cos(teta3[2])-b1*sin(teta3[2]);
double t3 = b1*cos(teta3[2])+a1*sin(teta3[2])+c1;
teta2[2] = atan2((s3*d1)+ (t3*e2),-(s3*e2)+(t3*d1));

//teta3[3] icin teta2[3] bulunuyor
double s4 = a1*cos(teta3[3])-b1*sin(teta3[3]);
double t4 = b1*cos(teta3[3])+a1*sin(teta3[3])+c1;
teta2[3] = atan2((s4*d1)+ (t4*e2),-(s4*e2)+(t4*d1));

for(int i=0;i<4;i++)
//teta1 acisi 180 den büyük ise tersini al.
//orn : 252 derece yerine -108 derece al.
{
    if (teta1[i]>M_PI)
    {
        teta1[i]=teta1[i]-2*M_PI;
    }
}

double C[4][3] =
{{teta1[0]*180/M_PI,teta2[0]*180/M_PI,teta3[0]*180/M_PI},
{teta1[1]*180/M_PI,teta2[1]*180/M_PI,teta3[1]*180/M_PI},
{teta1[2]*180/M_PI,teta2[2]*180/M_PI,teta3[2]*180/M_PI},
{teta1[3]*180/M_PI,teta2[3]*180/M_PI,teta3[3]*180/M_PI}};

cout << "Tum Cozum Kumeleri=" << endl;
//cozum kumeleri ekrana bastir
for(int i=0; i < 4; ++i)
{
    for (int j=0; j < 3; ++j)
    {
        cout << C[i][j] << " ";
    }
    cout << endl;
}

double sol[1][3]; //en
dogru cozum kumesini sol degiskenine aktar

if(px>=0 && py>=0)
{
    for(int n = 0; n <3; n++)
    {

```

```

        if((0<tetal[n] && tetal[n]<(M_PI/2) || -
2*M_PI<tetal[n] && tetal[n]<(-3*M_PI/2)) && 0<teta2[n] && teta2[n]
<2*M_PI/3 && 0 < teta3[n] && teta3[n] <2*M_PI/3)
        {
            for(int i = 0; i<3;i++)
                sol[0][i]=C[n][i];
        }
    }

    if (px<=0 && py>=0)
    {
        for(int n=0; n<3; n++)
        {
            if(((M_PI/2)<tetal[n] && tetal[n]<M_PI || (-
3*M_PI/2)<tetal[n] && tetal[n]<(-M_PI)) && 0<teta2[n] &&
teta2[n]<2*M_PI/3 && 0<teta3[n] && teta3[n]<2*M_PI/3)
            {
                for(int i = 0; i<3;i++)
                    sol[0][i]=C[n][i];
            }
        }
    }

    if (px<=0 && py<=0)
    {
        for(int n=0; n<3; n++)
        {
            if((M_PI<tetal[n] && tetal[n]<(3*M_PI/2) || -
M_PI<tetal[n] && tetal[n]<(-M_PI/2)) && 0<teta2[n] &&
teta2[n]<2*M_PI/3 && 0<teta3[n] && teta3[n]<2*M_PI/3)
            {
                for(int i = 0; i<3;i++)
                    sol[0][i]=C[n][i];
            }
        }
    }

    if (px>=0 && py<=0)
    {
        for(int n=0; n<3; n++)
        {
            if(((3*M_PI/2)<=tetal[n] && tetal[n]<=(2*M_PI) || (-
M_PI/2)<=tetal[n] && tetal[n]<=0 ) && 0<=teta2[n] &&
teta2[n]<=2*M_PI/3 && 0<=teta3[n] && teta3[n]<=2*M_PI/3)
            {
                for(int i = 0; i<3;i++)
                    sol[0][i]=C[n][i];
            }
        }
    }

    printf("\nEn Dogru Cozum Icin Aci Degerleri: %lf %lf
%lf",sol[0][0],sol[0][1],sol[0][2]);

    stringstream strs1;
    //float acil degeri char a cevrilip mlacistr ye atiliyor
    strs1 << sol[0][0];
    string temp_str1 = strs1.str();
    char* mlacistr = (char*) temp_str1.c_str();

```

```

        stringstream str2;
//float aci2 degeri char a cevrilip m2acistr ye atiliyor
        str2 << sol[0][1];;
        string temp_str2 = str2.str();
        char* m2acistr = (char*) temp_str2.c_str();

        stringstream str3;
//float aci3 degeri char a cevrilip m3acistr ye atiliyor
        str3 << sol[0][2];;
        string temp_str3 = str3.str();
        char* m3acistr = (char*) temp_str3.c_str();

        char* space = " "; //aci degerleri arasina bosluk koymak icin
        char* newline = "\n"; //aci degerleri gonderilince yeni
        satira gecmek icin

        int bufferSize = strlen(mlacistr) + strlen(space) +
        strlen(m2acistr) + strlen(space) + strlen(m3acistr) +
        strlen(newline);
        char* concatString = new char[ bufferSize ]; //char aci1 aci2
        aci3 degerleri birlestiriliyor
        strcpy( concatString, mlacistr );
        strcat( concatString, space );
        strcat( concatString, m2acistr );
        strcat( concatString, space );
        strcat( concatString, m3acistr );
        strcat( concatString, newline );
        printf("\nGonderilen Aci Degerleri : %s",concatString);
//ekrana dogru aci degerlerini bastir

        Serial* SP1 = new Serial("\\\\.\\COM14"); //topu almak icin
        serial baslat
        int acilength = strlen(concatString); //aci degerlerinin
        toplam string uzunlugu
        bool WriteSuccess = SP1->
>WriteData(concatString,acilength);//topun alinacagi konumu gonder
        delete SP1; //top alinacak konum gitti seriali sil
        Sleep(10000); //topu almasi icin 10 saniye bekle
        Serial* SP2 = new Serial("\\\\.\\COM14"); //topu birakmak icin
        serial baslat
        WriteSuccess = SP2->WriteData("90 60 60\n",9); //topun
        birakilacagi konum
        delete SP2; //top alinacak konum gitti seriali sil
        Sleep(10000); //topu birakmasi icin 10 saniye bekle
        Serial* SP3 = new Serial("\\\\.\\COM14"); //reset pozisyonuna
        gecmek icin serial baslat
        WriteSuccess = SP3->WriteData("0 0 0\n",6); //reset pozisyonu
        delete SP3; //reset konumu gonderildi seriali sil
        Sleep(10000);

        return 0;
}

```

Ek 3 Arduino Robot Kontrol Yazılımı

```
#include <PID_v1.h>
#include <multiMap.h>

int m1en = 8;           //motor1 aktif etme pini
int m1direction = 9;    //motor1 yon secim pini high ileri low geri
int m1speed = 10;       //motor1 pwm pini
int m2en = 7;           //motor2 aktif etme pini
int m2direction = 4;    //motor2 yon secim pini high ileri low geri
int m2speed = 5;        //motor2 pwm pini
int m3en = 6;           //motor3 aktif etme pini
int m3direction = 2;    //motor3 yon secim pini high ileri low geri
int m3speed = 3;        //motor3 pwm pini
int m1pot = A0;         //motor1 potansiyometre analog okuma pini
int m2pot = A1;         //motor2 potansiyometre analog okuma pini
int m3pot = A2;         //motor3 potansiyometre analog okuma pini

double error1=0,m1_aci_gelen=0,Kp1=7.0,Ki1=0.01,Kd1=0;
double error2=0,m2_aci_gelen=0,Kp2,Ki2,Kd2;
double error3=0,m3_aci_gelen=0,Kp3,Ki3,Kd3;

double potvalue1 = 0;
double potvalue2 = 0;
double potvalue3 = 0;

int refcounter = 0;
String seri_veri = "";

float m1mapans; //mapping den gelen float pot degerleri
float m2mapans;
float m3mapans;

//yuvarlama icin
int m1mapansint; //mapping den gelen float degerleri round edilip
int e cevrilerek bu degiskenlere atilacak
int m2mapansint;
int m3mapansint;

double m1_git; //pid bloga girecek olan pot degerleri double a
cevrilecek
double m2_git;
double m3_git;

PID pid1(&potvalue1,&error1,&m1_git,Kp1,Ki1,Kd1,DIRECT); // 1.
motor icin PID kontrolor olusturuldu
PID pid2(&potvalue2,&error2,&m2_git,Kp2,Ki2,Kd2,DIRECT); // 2.
motor icin PID kontrolor olusturuldu
PID pid3(&potvalue3,&error3,&m3_git,Kp3,Ki3,Kd3,DIRECT); // 3.
motor icin PID kontrolor olusturuldu

void setup()
{
  RobotInitialize();      // baslangic parametrelerini yukle
}

void loop()
{
```

```

    RobotSerialReceive(); // robotun eklemlerinin gidecegi aci
    degerlerini USB porttan al
    RobotMapping(); // gelen aci degerlerini multiMap
    fonksiyonuna sokarak aci - potansiyometre donusumunu yap.
    RobotPIDParameters(); // gelen aci degerlerine gore, onceden
    belirlenmis olan P,I,D degerlerini kontrolorlere yukle
    RobotRun(); // robotu harekete gecir
}

void RobotRun()
{
    potvalue1 = map(analogRead(m1pot), 0, 1023, 1023, 0); //motor1
    potansiyometre oku
    potvalue2 = analogRead(m2pot); //motor2
    potansiyometre oku
    potvalue3 = map(analogRead(m3pot), 0, 1023, 1023, 0); //motor3
    potansiyometre oku

    pid1.Compute(); // 1. PID kontrolorun cikisini hesapla
    pid2.Compute(); // 2. PID kontrolorun cikisini hesapla
    pid3.Compute(); // 3. PID kontrolorun cikisini hesapla

    Serial.print("m1_git=");
    Serial.print(m1_git); // 1. eklemin gidecegi potansiyometre
    degerini seri porta bas.
    Serial.print("\t|| ");

    Serial.print("m2_git=");
    Serial.print(m2_git); // 2. eklemin gidecegi potansiyometre
    degerini seri porta bas.
    Serial.print("\t|| ");

    Serial.print("m3_git=");
    Serial.print(m3_git); // 3. eklemin gidecegi potansiyometre
    degerini seri porta bas.
    Serial.print("\t|| ");

    Serial.print("Error1=");
    Serial.print(m1_git-potvalue1); // 1. eklemin hata miktarı
    Serial.print("\t|| ");

    Serial.print("Error2=");
    Serial.print(m2_git-potvalue2); // 2. eklemin hata miktarı
    Serial.print("\t|| ");

    Serial.print("Error3=");
    Serial.print(m3_git-potvalue3); // 3. eklemin hata miktarı

    Serial.print("\n");

    // motor 1 hata degerine gore hareket edilecek yon ve hiz
    belirlemeleri
    if(error1<-3)
    {
        digitalWrite(m1direction, HIGH);
        analogWrite(m1speed, error1);
    }
    if(error1>= -3 && error1<=3)
    {

```

```

        digitalWrite(m1direction, LOW);
        analogWrite(m1speed, 0);
    }
    if (error1 > 3)
    {
        digitalWrite(m1direction, LOW);
        analogWrite(m1speed, error1);
    }

    // motor 2 hata degerine gore hareket edilecek yon ve hiz
    belirlemeleri
    if (error2 < -3)
    {
        digitalWrite(m2direction, LOW);
        analogWrite(m2speed, -error2);
    }
    if (error2 >= -3 && error2 <= 3)
    {
        digitalWrite(m2direction, LOW);
        analogWrite(m2speed, 0);
    }
    if (error2 > 3)
    {
        digitalWrite(m2direction, HIGH);
        analogWrite(m2speed, -error2);
    }

    // motor 3 hata degerine gore hareket edilecek yon ve hiz
    belirlemeleri
    if (error3 < -5)
    {
        digitalWrite(m3direction, LOW);
        analogWrite(m3speed, -error3);
    }
    if (error3 >= -5 && error3 <= 5)
    {
        digitalWrite(m3direction, LOW);
        analogWrite(m3speed, 0);
    }
    if (error3 > 5)
    {
        digitalWrite(m3direction, HIGH);
        analogWrite(m3speed, -error3);
    }
    delay(1);
}

void RobotInitialize()
{
    TCCR2B = TCCR2B & 0b11111000 | 0x02; // Motor1 pwm 4 khz'e set
    edildi.
    TCCR3B = TCCR3B & 0b11111000 | 0x02; // Motor2 ve Motor3 pwm 4
    khz'e set edildi.

    pinMode(m1en, OUTPUT); // cikis atamalari
    pinMode(m1direction, OUTPUT);
    pinMode(m1speed, OUTPUT);
    pinMode(m2en, OUTPUT);
    pinMode(m2direction, OUTPUT);
    pinMode(m2speed, OUTPUT);
    pinMode(m3en, OUTPUT);

```

```

pinMode(m3direction, OUTPUT);
pinMode(m3speed, OUTPUT);

Serial.begin(9600);           //seri haberlesme baslat

digitalWrite(m1en, HIGH);     //motor1 aktif et
digitalWrite(m2en, HIGH);     //motor2 aktif et
digitalWrite(m3en, HIGH);     //motor3 aktif et

pid1.SetMode(AUTOMATIC);      // PID kontrolorlerin calisma
modlariinin secimi
pid2.SetMode(AUTOMATIC);
pid3.SetMode(AUTOMATIC);

pid1.SetOutputLimits(-255,255); //PID kontrolorlerin cikis
limitlerinin belirlenmesi
pid2.SetOutputLimits(-255,255);
pid3.SetOutputLimits(-255,255);
}

void RobotSerialReceive()
{
    while(Serial.available() > 0) //seri haberlesme basladi ise
verileri al
    {
        char gelen = Serial.read();
        seri_veri += gelen;

        if(gelen == ' ' && refcounter == 0) // ilk bosluga kadar olan
verileri 1. eklem aci degeri olarak tanimla
        {
            refcounter+=1;
            m1_aci_gelen = seri_veri.toFloat();
            seri_veri = "";
        }
        else if(gelen == ' ' && refcounter == 1) // ilk bosluk ile
ikinci bosluk arasindaki verileri 2. eklem aci degeri olarak
tanimla
        {
            refcounter+=1;
            m2_aci_gelen = seri_veri.toFloat();
            seri_veri = "";
        }
        else if(gelen == '\n') // ikinci bosluktan yeni satira
kadar olan verileri 3. eklem aci degeri olarak tanimla
        {
            refcounter = 0;
            m3_aci_gelen = seri_veri.toFloat();
            seri_veri = "";
        }
    }
}

void RobotMapping()
{
    //motor1 mapping (-160, 160 derece)
    float out1[] = {17,234,512,790,1006}; // 5 olcum alindi
    float in1[] = {-160,-90,0,90,160};
    m1mapans = multiMap<float>(m1_aci_gelen, in1, out1, 5);
}

```

```

m1mapansint=(int) (round(m1mapans));
m1_git=(double)m1mapansint;

//motor2 mapping (-45, 120 derece)
float out2[] =
{26,44,59,74,88,103,117,133,148,163,179,195,210,225,241,255,271,28
7,303,320,335,352,367,380,395,408,424,439,457,471,487,503,516,533}
; // 34 olcum alindi
float in2[] = {-45,-40,-35,-30,-25,-20,-15,-10,-
5,0,5,10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,95,100,10
5,110,115,120};
//int val2 = (int)m2_aci_gelen;
m2mapans = multiMap<float>(m2_aci_gelen, in2, out2, 34);
m2mapansint=(int) (round(m2mapans));
m2_git=(double)m2mapansint;

//motor3 mapping (-135, 135 derece)
float out3[] =
{102,117,132,147,162,177,192,207,222,237,252,267,282,297,312,327,3
41,356,373,386,400,417,432,444,463,477,487,512,529,543,559,572,589
,603,618,637,650,664,685,703,714,732,747,758,775,793,804,818,828,8
47,862,877,893,910,925}; // 55 olcum alindi
float in3[] = {-135,-130,-125,-120,-115,-110,-105,-100,-95,-90,-
85,-80,-75,-70,-65,-60,-55,-50,-45,-40,-35,-30,-25,-20,-15,-10,-
5,0,5,10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,95,100,10
5,110,115,120,125,130,135};
m3mapans = multiMap<float>(m3_aci_gelen, in3, out3, 55);
m3mapansint=(int) (round(m3mapans));
m3_git=(double)m3mapansint;

}

void RobotPIDParameters()
{
    // motor 2 Kp, Ki,Kd seçim bölümü
    if(m2_git<=116 && m2_git>=26) //M2 -15.1,-45 derece arası
    {
        Kp2=7;
        Ki2=0.02;
        Kd2=0.2;
    }

    else if(m2_git<=210 && m2_git>=117) //M2 15,-15 derece arası
    {
        Kp2=7;
        Ki2=0.01;
        Kd2=0.1;
    }

    else if(m2_git<=352 && m2_git>=211) //M2 60,15.1 derece arası
    {
        Kp2=7;
        Ki2=0.02;
        Kd2=0.2;
    }

    else if(m2_git<=439 && m2_git>=353) //M2 90,60.1 derece arası
    {
        Kp2=7;
        Ki2=0.02;
        Kd2=0.2;
    }
}

```

```

}

else if(m2_git<=533 && m2_git>=440) //M2 120,90.1 derece arasi
{
    Kp2=7;
    Ki2=0.05;
    Kd2=0.3;
}

// motor 3 Kp, Ki,Kd seçim bölümü
if(m3_git<=176 && m3_git>=102) //M3 -110.1,-135 derece arasi
{
    Kp3=5;
    Ki3=0.25;
    Kd3=0.5;
}

else if(m3_git<=236 && m3_git>=177) //M3 -90.1,-110 derece arasi
{
    Kp3=5;
    Ki3=0.25;
    Kd3=0.5;
}

else if(m3_git<=326 && m3_git>=237) //M3 -60.1,-90 derece arasi
{
    Kp3=5;
    Ki3=0.25;
    Kd3=0.5;
}

else if(m3_git<=462 && m3_git>=327) //M3 -15.1,-60 derece arasi
{
    Kp3=4;
    Ki3=0.25;
    Kd3=0.3;
}

else if(m3_git<=559 && m3_git>=463) //M3 15,-15 derece arasi
{
    Kp3=5;
    Ki3=0.01;
    Kd3=0.01;
}

else if(m3_git<=703 && m3_git>=560) //M3 60,15.1 derece arasi
{
    Kp3=4;
    Ki3=0.25;
    Kd3=0.3;
}

else if(m3_git<=793 && m3_git>=704) //M3 90,60.1 derece arasi
{
    Kp3=5;
    Ki3=0.25;
    Kd3=0.5;
}

else if(m3_git<=847 && m3_git>=794) //M3 110,90.1 derece arasi
{

```

```
Kp3=5;
Ki3=0.25;
Kd3=0.5;
}

else if(m3_git<=925 && m3_git>=848) //M3 135,110.1 derece arasi
{
    Kp3=5;
    Ki3=0.25;
    Kd3=0.5;
}

pid2.SetTunings(Kp2,Ki2,Kd2); // 2. eklem icin secilen
parametreleri 2. kontrole yukle
pid3.SetTunings(Kp3,Ki3,Kd3); // 3. eklem icin secilen
parametreleri 3. kontrole yukle
}
```