TEK BAKIŞ AÇISI DERİNLİK KESTİRİMİ YÖNTEMİ İLE ROBOT DENETİMİ

ENGİN KARATAŞ

YÜKSEK LİSANS TEZİ

ELEKTRİK VE ELEKTRONİK MÜHENDİSLİĞİ

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

AĞUSTOS 2010

ANKARA

Fen Bilimleri Enstitüsü onayı

_____

Prof. Dr. Ünver KAYNAK

Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Prof. Dr. Mehmet Önder EFE

Anabilim Dalı Başkanı

Engin KARATAŞ tarafından hazırlanan TEK BAKIŞ AÇISI DERİNLİK KESTİRİMİ YÖNTEMİ İLE ROBOT DENETİMİ adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Doç. Dr. Veysel GAZİ

Tez Danışmanı

Tez Jüri Üyeleri

Başkan : Prof. Dr. Mehmet Önder EFE _____

Üye : Doç. Dr. Veysel GAZİ _____

Üye : Yrd. Doç. Dr. Nilay SEZER UZOL _____

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Engin KARATAŞ

Üniversitesi : **TOBB Ekonomi ve Teknoloji Üniversitesi**
Enstitüsü : **Fen Bilimleri Enstitüsü**
Anabilim Dalı : **Elektrik ve Elektronik Mühendisliği**
Tez Danışmanı : **Doç. Dr. Veysel GAZİ**
Tez Türü ve Tarihi : **Yüksek Lisans - Ağustos 2010**

ENGİN KARATAŞ

TEK BAKIŞ AÇISI DERİNLİK KESTİRİMİ YÖNTEMİ İLE ROBOT DENETİMİ

ÖZET

Bu tez çalışmasında tek bakış açısı derinlik kestirimi yöntemi ile sürü robotlarda dizilim denetimi ve geri besleme doğrusallaştırılması yöntemi ile hareketli robotlar için bir hedefin etrafında çember takibi üzerinde çalışılmıştır. Nesneleri üç boyutlu algılayabilmek için stereo görüş gereklidir. Diğer bir deyişle, üç boyutlu algılama için insanlar için bir çift göze ve robotlar için de bir çift kameraya ihtiyaç vardır. Mesafe ölçümü içinse bir özelliği, bu çalışmada yüksekliği, bilinen bir nesnenin geliştirilen tek bakış açısı derinlik kestirimi yöntemi ile bu ölçümün nasıl yapılabileceği anlatılmıştır. Deneylerde Khepera 3 mini robotlar kullanılmış ve öncelikle robotlar üzerinde mevcut olmadığı için uygun kamera, servo motorlar ve diğer parçaların entegrasyonu yapılmıştır ve kamera için uygun sürücü derlenmiştir. Görüntü işleme için OpenCV kullanılmıştır. Gerekli OpenCV kütüphaneleri çapraz derlenip bir flash bellek yardımı ile robot üzerinde kullanılabilmesi sağlanmıştır. Üç ve dört robot ile yapılan çalışmalarda tasarlanan denetleyiciler ile basit bir kamera kullanarak dizilim denetimi uygulaması yapılmış ve robotların başarılı biçimde dizilimin bozulmadan hedeflere ulaştıkları gözlenmiştir. Tek bir robot üzerinde yapılan çalışmada ise geri besleme ile doğrusallaştırma yöntemi kullanarak robotun önceden yarıçapı belirlenen bir çember üzerine bir hedef etrafında dönmesi sağlanmıştır. Bu çalışma daha sonra sürü robotlarda kullanılmak üzere tasarlanan ve erkinlerin bir birini hedef olarak gördükleri uygulamalar için bir ön çalışma niteliği taşımaktadır.

Anahtar Kelimeler: Dizilim Denetimi, Geri Besleme Doğrusallaştırılması, Robot Görüşü, OpenCV, Çok Erkinli Sistemler

University	**: TOBB University of Economics and Technology**
Institute	**: Institute of Natural and Applied Sciences**
Science Programme	**: Electrical and Electronics Engineering**
Supervisor	**: Associate Professor Veysel GAZİ**
Degree Awarded and Date	**: M.S. - August 2010**

ENGİN KARATAŞ

ROBOT CONTROL USING SINGLE VIEW DEPTH ESTIMATION METHOD

ABSTRACT

In this thesis formation control in a multi-robot system using single view depth estimation method and circling around a target for mobile robots by feedback linearization were studied. In order to sense objects in three dimension stereo vision is necessary. In other words, two eyes for humans and two cameras for robots are needed for three dimensional sensing. For measuring distance a feature of an object, which is height in this study, is known and this information is used to explain how to measure an object's distance by single view depth estimation method that's developed. Khepera 3 mini robots were used in the experiments and since they were not equipped with the necessary equipments like webcams, servo motors, etc. , these parts were chosen appropriately and integrated to the robots. A driver for chosen webcam was also compiled. OpenCV was used for image processing. Necessary OpenCV libraries were cross-compiled and added on robots via a flash disk. In these studies, which are done on 3 or 4 robots, it was observed that robots can successfully go to their destination points without breaking formation, using their designed controllers and a simple webcam mounted on them. In the study using a single robot using the feedback linearization techniques a controller for circling around a target with a predefined radius was implemented. This study constitutes a preliminary study towards a multi-agent system application in which robots constitutes targets to each other.

Keywords: Formation Control, Feedback Linearization, Robot Vision, OpenCV, Multi Agent Systems

TEŞEKKÜR

Yüksek lisansım boyunca her zaman yanımda olan, iyi günümüzde bizimle birlikte sevinen, kötü günümüzde bizlere destek olan ve dinleyen, çalışma azmine hayran olduğum değerli hocam Doç. Dr. Veysel GAZİ'ye ve üzerimdeki emekleri için daima minnet duyacağım değerli hocalarım Prof. Dr. Mehmet Önder EFE'ye ve Yrd. Doç. Dr. Nilay SEZER UZOL'a,

Beraber çalışmaktan gurur duyduğum Sürü Sistemler Araştırma Laboratuvarı'ndaki çalışma arkadaşlarım olan Murat İlter Köksal, Salih Burak Akat, Ömer Çayırpunar, Mirbek Turduev, Yunus Ataş, Abdel-Razzak Merheb, Esma Gül ve Sabahat Duran'a,

Deney sonuçlarını çıkarmamda yardımcı olduğu için Andaç Töre Şamiloğlu'na, bizim için bizimle sabahlayan arkadaşım Murat Kırtay'a,

Ve her zaman beni destekleyen ve bugünlere getiren aileme teşekkürlerimi sunarım.

Bu çalışma Avrupa Komisyonu tarafından 045269 sözleşme numaralı 6. Çerçeve Programı özel amaçlı araştırma projesi kapsamında ve TÜBİTAK (Türkiye Bilimsel ve Teknolojik Araştırma Kurumu) tarafından 104E170 ve 106E122 sayılı projeler kapsamında ve desteklenmiştir.

İÇİNDEKİLER

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

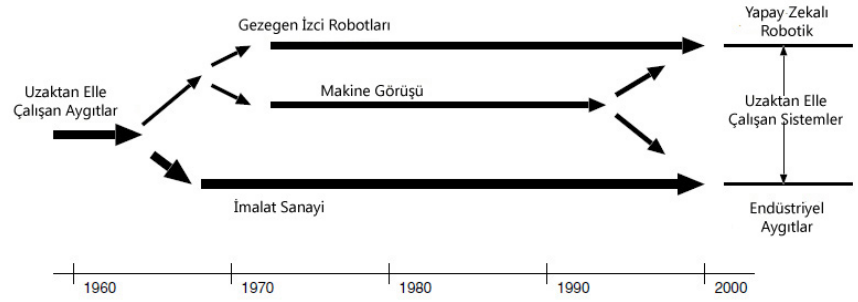
Şekil **Sayfa**

BÖLÜM 1

1. GİRİŞ

1.1. Robotik

Robotların tarihçesine bakıldığında 1960'lı yılların başlarında robotların evrilme sürecinde kollara ayrıldığı görülmüştür [1]. Üretimde kullanılacak robotlar için mühendisler robot kolları tasarladılar. Otomasyonu da sağlayan bu robot kolları bir kez programlandığında çok az bakım yapılarak haftalarca hatta aylarca çalışabilme kapasitesine sahiptiler. Şekil 1.1.'de 1960'lı yıllardan sonrasında robotların gelişimindeki yol ayrımları gösterilmektedir.



Şekil 1.1.: Robotların Gelişimini Gösteren Zaman Çizelgesi (Şekil [1] kaynağından alıntıdır.)

Toplu üretim yapılan montaj hatlarında robotlar herhangi bir problemi algılamak zorunda değildiler. Diğer bir deyişle bu robotlar kör ve algısızdılar. Uzay programlarında kullanılan robotlar ise zaman çizelgesinin farklı bir yol ayrımında idiler. Fazlaca özelleşmiş ve eşsiz olan bu robotlar gezegen izci[1] robotlarıydı. Üretimde kullanılan ve otomatik hareket eden robotların aksine izci robotlar, radyo haberleşmesinin olmadığı ayın karanlık tarafına geçişte beklenmeyen durumlarla karşı karşıya kalabiliyorlardı. Apollo 17'de astronot ve jeolog Harrison Schmitt ayın üzerinde hiç beklemediği turuncu bir kayaya rastlamıştır [1]. İdeal olarak böyle bir durumda robot kendini tehlikeye atmamak için durabilmeli ve araştırmayı kesebilmelidir. Asgari özelliklere sahip bir izci robot çevresini algılayabilecek kaynaklara, bu kaynaklardan gelen girdileri yorumlayabilecek bir yapıya ve değişen ortamlara tepki verecek şekilde hareketlerini uyarlayabilecek yöntemlere sahip

[1] ing: planetary rover

olmalıdır. Akıllı denebilecek robotların yapıldığı 1980'lerin sonlarına kadar ise insanlar eş zamanlı ortama ayak uydurabilmek için uzaktan çalıştırılabilme[2] mantığıyla çalışan robotları kullanmışlardır. Tehlikeli olan radyoaktif elementleri işleyebilmek için kuvveti yansıtabilen uzaktan elle çalışılan[3] robotlar kullanılmıştır [1].

1.2. Çok Erkinli Sistemler

Akıllı robotların gelişimine paralel olarak ama ondan bağımsız şekilde birden çok robottan yada birimden oluşan sistemleri ele alan çok erkinli sistemler gelişim göstermiştir. Çok erkinli sistemler[4], erkin adı verilen etkileşim halinde olan ve hesap yapabilen birden çok elemandan oluşan sistemlerdir [2]. Bir bilgisayar yazılımı yada bir robot gibi hesaplama yeteneğe sahip herhangi bir birim olan erkin iki önemli özelliğe sahiptir [3]. İlk olarak, belirli bir ölçüde de olsa bir erkin kendi kendine bağımsız eylemde bulunabilme özelliğine sahip olmalıdır. Diğer bir deyişle daha önceden belirlenen amaçları gerçekleştirebilmek için yapması gerekenlere karar verebilmelidir. İkinci olarak ise diğer erkinler ile etkileşim halinde olabilme yetisine sahip olmalıdır. Etkileşim halinde olmak basitçe veri alışverişi olarak algılanmamalıdır. Günlük hayattaki sosyal aktivitelerimizdeki etkileşime benzer olarak erkinler de kendi aralarında yardımlaşabilmeli, belirli bir düzene uyabilmeli veya uzlaşabilmelidirler [2].

Çok erkinli sistemler bilgisayar bilimlerinin nispeten yeni bir alt alanıdır. Her ne kadar ilk çalışmalar 1980'ler civarında başlamış olsa da geniş çapta farkına varılması 1990'lı yılların ortalarını bulmaktadır. Daha sonraları ise uluslararası ilginin muazzam bir şekilde arttığı bir alan olmuştur. [4] kaynağında çok erkinli sistemlerin karakteristik özellikleri şu şekilde tanımlanmıştır:

1. Her erkin sadece eksik bilgiye sahiptir ve yetenekleri kısıtlanmıştır.
2. Sistemin denetimi dağıtıktır.
3. Veriler belirli bir merkezde bulunmamalı, dağıtılmış halde bulunmalıdır.
4. Hesaplama, eşzamansız olarak yapılmalıdır.

Çok erkinli sistemler, erkinler bazında, birbirleriyle ve herbirinin çevresiyle olan eylemde bulunma şekilleri bazında farklılıklar gösterebilmektedirler.

[2]ing: teleoperation
[3]ing: telemanipulator
[4]ing: multiagent systems

1.3. Bilgisayarla Görüş

Çok erkinli sistemlerde etkileşim çeşitlerinden biri de bilgisayarla görüş[5] kullanılarak yapılandır. Bilgisayarla görüş, genel veya özel amaçlı bilgisayarlar tarafından çevremizdeki dünyadan otomatik olarak alınan görüntü veya görüntü kümelerinden faydalı bilgilerin çıkarılma yöntemlerini teorik ve algoritmik bazda geliştiren bir bilim dalıdır [5]. Faydalı bilgiler ile kastedilen genel bir nesnenin algılanması, bilinmeyen bir nesnenin üç boyutlu tanımlanması, gözlenen bir nesnenin konum ve yönelimi yada bir nesnenin uzaysal herhangi bir özelliğinin ölçülmesi olabilir.

Bir görüntü bir nesnenin, iki veya üç boyutlu bir sahnenin veya başka bir görüntünün uzaysal temsilidir. Görüntü, gerçek veya sanal olabilir. Bilgisayarla görüşte görüntü denildiğinde genelde akla gelen video görüntüsü, sayısal görüntü veya bir resim gibi kaydedilmiş görüntüdür. Gözlemlenen sayısal görüntünün birimi pikseldir. Bir piksel kendine ait konum ve değer bilgilerine sahiptir. Bilgisayarla görüşte ortaya çıkan temel sorunlardan biri de tek başına bir pikselin bilgisinin, ne nesnenin algılanmasında ne de şeklinin, konumunun ve yöneliminin tanımlanmasında pek bir faydasının olmamasıdır. Nesnelerin bilgisayarla tanınması genellikle bir dizi karmaşık işlemlerin başarılı bir şekilde uygulanması ile mümkün olabilmektedir. Bunun için kestirme bir yol yoktur. Nesnelerin tanınabilmesi için gereken işlemler ise koşullandırma, etiketleme, gruplandırma, çıkarım yapma ve eşleştirme olarak sıralanabilir [5].

1.4. Tezin Amacı ve Konusu

Bu tez çalışmasında çoklu robot erkinleri üzerinde basit bir kamera kullanarak, geliştirilen tek bakış açısı derinlik kestirimi yöntemi ile robot erkinlerinin denetiminin sağlanması amaçlanmıştır. Yapılan deneylerin bir kısmı, 3 ve 4 robot erkininin kullanıldığı dizilim denetimi; diğer kısmı da yine çok robot erkinli sistemlerde kullanılması için amaçlanan fakat bu tez çalışmasında tek bir robot erkini üzerinde denenen geri besleme doğrusallaştırması yöntemi ile robot denetimidir.

Bölüm 2.'de tek bakış açısı derinlik kestirimi yönteminin teorik kısmına değinilmiş, basit bir kameranın modellenmesi gösterilmiş ve kamera kalibrasyon matrisinden elde edilen katsayılarla robotların tek bakış açısı derinlik kestirimi yöntemi ile çevrelerindeki nesnelerin uzaklık ve açılarının nasıl bulunabileği denklemler halinde ifade edilmiştir.

[5] ing: computer vision

Bölüm 3.'te teorisi anlatılmış olan tek bakış açısı derinlik kestirimi yönteminin uygulanacağı Khepera 3 robotları öncelikle tanıtılmıştır. Daha sonra deney çalışmalarındaki amaçlara uygun olarak takılan servo motorlara neden ihtiyaç duyulduğu anlatılmıştır. Servo motorlarla beraber eklenen Li-Po pil, voltaj regülatörü, USB Hub ve flash belleğin ayrı ayrı fonksiyonları açıklanmıştır. Deneylerde kullanılan basit kameranın Khepera 3 robotunu tasarlayan K-Team firmasının robot üzerine entegre etmiş olduğu kameraya göre üstünlükleri, özellikleri yine bu bölümde gösterilmiştir. Ayrıca, kameraların çalışmasına imkan veren arm-linux işletim sistemi için kamera sürücüsünün hazırlanmasında nelere dikkat edileceğine değinilmiştir. Khepera 3 robotu deneyin yapılabileceği şekilde donanıma sahip olduktan sonra robotlar üzerinde görüntü işlenmesini sağlayan OpenCV kütüphaneleri ile bu kütüphaneleri kullanarak denenen farklı görüntü işleme yöntemlerinden bahsedilmiştir.

Bölüm 4.'te ilk olarak erkin modellenmesi yapılmış daha sonra dizilim denetimi için problem tanımı ifade edilmiştir. Deneylerde dizilim denetiminde kullanılan lider takipçi yapısı açıklanmış, bu yapı içindeki robotların görevleri açıklanmış ve her birinin denetleyicileri de ayrı ayrı anlatılmıştır. Daha sonra Khepera 3 robotları tüm yönlü robotlar olmadıklarından denetleyicilerin robotlar üzerinden çalışabilmesi için gerekli değişiklere değinilmiştir. Tüm robotlar için ortak olan üst kısma eklenen servoların nasıl denetlendiği de burada açıklanmıştır. En son olarak da robotlarla yapılan deneyler gösterilmiş ve elde edilen grafikler yorumlanmıştır.

Bölüm 5.'de çok erkinli sistemlerde uygulanılması düşünülen ancak burada tek bir robot üzerinden denenen geribesleme doğrusallaştırması yöntemi ile çember takibinin nasıl yapıldığı gösterilmiştir. Yapılan deneyler ve elde edilen grafikler yorumlanmıştır.

BÖLÜM 2

2. TEK BAKIŞ AÇISI DERİNLİK KESTİRİMİ YÖNTEMİ

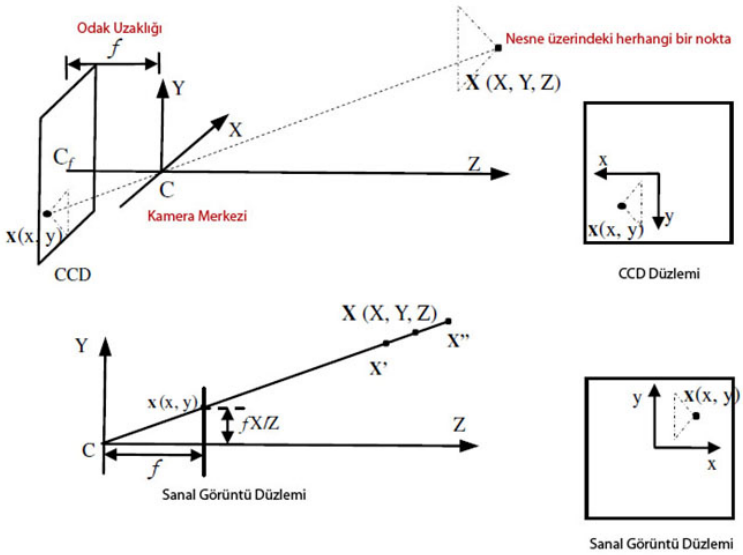
Çevremize baktığımızda nesneleri 3 boyutlu algılarız. Nesneleri 3 boyutlu algılayabilmek için ise stereo görüş gereklidir. Başka bir şekilde ifade etmek istersek, 3 boyutlu algılayabilmek için insanlar bir çift göze, robotlar da benzer şekilde belirli bir mesafe ile birleştirilmiş bir çift kameraya ihtiyaç duyarlar. Derinlik algılamasının tek kamera ile yapılabileceğini anlatan bu çalışmada kameranın görevi, robotun çevresinde seçilen nesnelerle yani diğer robotlarla arasındaki mesafesini ve açısını ölçmektir. Mesafe ve açı ölçümü içinse bir özelliği, bu çalışmada yüksekliği, bilinen bir nesnenin geliştirilen tek bakış açısı derinlik kestirimi yöntemi ile bu ölçümün nasıl yapılabileceği bu bölümün konusudur.

Robotların denetimi için basit bir CCD (charge coupled device-yük birleşik aygıt) kamera kullanılmıştır. Burada basit bir kamera kullanılmasının altını çizmek gerekmektedir çünkü çalışmanın temel amaçlarından biridir. Her-yönlü[1], PTZ[2] veya yüksek çözünürlüklü ve pahalı özel bir kamera kullanılmamıştır. Herhangi bir bilgisayarcıdan temin edilebilecek uygun bir kamera ile robotun kendi çevresinden toplayacağı bazı verileri toplayarak özellik çıkarımı yapılabilir [6]. Burada robot kontrolü üzerine özelleştiğimiz için kullanacağımız özellik, yaklaşık mesafe yani derinlik ve yaklaşık açı bulmak için geliştirilecek bir yaklaşım olacaktır. Kameradan elde edilecek mesafe ve açı bilgisi robotların birbirlerine olan mesafeleri ve açıları olarak yorumlanacaktır. Bunun için öncelikle kullanacağımız kameranın modellenmesi, modelden çıkarılacak kalibrasyon matrisi ve kameranın iç parametreleri, daha sonra da bu matris kullanılarak nasıl uzaklık ve açı bilgilerine ulaşabileceğimize dair bir yöntem üzerinde durulacaktır. Bu bölümde [6], [7] ve [8] kaynaklarındaki çalışmalar baz alınmıştır ve de oradaki anlatımlar takip edilecektir.

2.1. Kamera Modellenmesi

Bu çalışmada kullanılan kamera Şekil 2.1.'de görüldüğü gibi modellenebilir. Şekil 2.1.'deki kamera geometrisi modeli eski tip iğne deliği kameraları yada daha rahat bir şekilde insan gözünün yapısını esas alarak anlaşılabilir.

[1] ing: omni-vision
[2] Pan-Tilt-Zoom



Şekil 2.1.: Kamera geometrisinin modellenmesi (Şekil [6] kaynağından alıntıdır.)

Görüntü izdüşümünün merkezi olan C noktasına *kamera merkezi* denir [9]. Bu noktaya *optik merkez* de denir. Kamera merkezinden içeri giren ışığın kamera içinde oluşturduğu görüntünün düzlemi *CCD düzlemi* olarak adlandırılır. CCD düzleminde kamera merkezinin izdüşümü olan, C_f, odak noktasından kamera merkezi, C, noktasını *kameranın asıl ekseni* denen bir doğru birleştirir. Şekil 2.1.'de de görüldüğü gibi bu doğru Z eksenidir. Kameranın görüşünün, kamera içindeki izdüşümü olan *CCD düzlemi*, kamera merkezinden odak uzaklığı, f, kadar uzaklıktadır. Kamera modellenmesindeki işlemlerimizi daha kolaylaştırmak için CCD düzleminde ters olarak oluşan görüntüyü kamera merkezinden f odak uzaklığı mesafesi kadar öne taşıdığımızda düz olarak oluşturulan düzleme *Sanal Görüntü Düzlemi (SGD)* denir.

Amacımız kameranın görüş alanındaki 3-boyutlu bir $\mathbf{X} = (X, Y, Z) \in \mathbb{R}^3$ noktasını SGD üzerinde koordinatı $\mathbf{x} = (x, y)$ olan bir noktaya eşleştirmektir. Bu eşleştirmeyi yapabilmek için ihtiyacımız olan matrise *kalibrasyon matrisi* [9] denir. Kalibrasyon matrisine ulaşabilmek için kamera geometrisi modelindeki üçgen benzerlikleri kullanılabilir. CCD düzlemi ve SGD'de x ve y koordinat eksenlerinin birbirlerine gore 180 derece döndürülmüş olduğuna dikkat edilmelidir. $\mathbf{x} = (x, y)$ noktasının koordinatlarını Şekil 2.1.'deki şu üçgen benzerliğiyle bulunabilir :

$$\frac{f}{Z} = \frac{x}{X} = \frac{y}{Y} \tag{2.1}$$

Eğer gerçek dünyadaki ve görüntü düzlemindeki noktaları homojen vektörler

cinsinden ifade edersek, birbirleri arasında bir doğrusal eşleştirme yapılabilir :

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \rightarrow Z \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = P \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.2}$$

$$P = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.3}$$

P matrisine *kamera izdüşüm matrisi* [9] denir. Şekil 2.1.'de, (x, y) koordinat eksenlerinin merkezi, SGD ile Z ekseninin kesiştiği noktaya yerleştirilmiştir. Gerçekte ise durum biraz daha farklıdır. Kameranın merkezi, kameradan gelen görüntünün merkezinden biraz kayık olabilir. Kayma miktarı, kameranın ideal olmamasından kaynaklanır ve bu bozulma ile ilgili detaylı bilgi Ek-A'da bulunmaktadır. Bu kayma miktarı da (p_x, p_y) olarak modele şu şekilde eklenebilir.

$$\begin{aligned} x &= \frac{fX}{Z} + p_x \\ y &= \frac{fY}{Z} + p_y \end{aligned} \tag{2.4}$$

Buradan *x* ve *y*'nin yeni değerleriyle beraber (2.2) eşitliği şu şekilde güncellenir:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \rightarrow Z \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.5}$$

(2.5) eşitliğinden alt matris olarak çekebileceğimiz K' matrisine

$$K' = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

kamera kalibrasyon matrisi [9] denir.

2.2. Uzayda Niceleme

Bir kameranın CCD düzlemi yada SGD, çözünürlüğü ne olursa olsun piksel adı verilen çoğu zaman kare birimlerden oluşurlar. Geniş açılı kameralar olan balık gözü kameralarda olduğu gibi bu piksellerin her zaman kare olması gerekmez ama bu tez çalışmasında kullanılan basit kamera kare piksellere sahiptir ve pikselleri kare olmayan kameralar üzerinde durulmayacaktır. Şekil 2.2.'de balık gözü lens ile normal lens arasındaki fark görülmektedir. Fotoğrafın merkezinden kenarlara doğru uzaklaştıkça çapsal bozulma artışı gözlemlenmekte ve görüntü sanki bir kürenin üzerine sarılmış izlenimi vermektedir [10].

Şekil 2.2.: Balık gözü lens (solda), Normal lens (sağda) (Şekil [10] kaynağından alıntıdır.)

Kare piksellere sahip bir kamera üzerinde görüntü tamsayı piksellerden oluştuğu için gerçek dünyadan alınan 3 boyutlu verilerin gerçek sayı değerleri, tam sayı değerlere dönüştürülmelidir. Bu işlem gerçek sayıları yuvarlayarak tam sayılara dönüştürecek bir yuvarlama işleçi ile ifade edilebilir. [6].

$$\begin{pmatrix} \bar{x} \\ \bar{y} \\ 1 \end{pmatrix} = q \left(Z^{-1} \left[K | 0 \right] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \right) \tag{2.7}$$

(2.7) eşitliğinde $(\bar{x}, \bar{y})$, gerçek sayı değerlerine sahip (x, y) değerlerinin yuvarlanıp tam sayıya dönüştürülmüş halleridir. (2.7) eşitliğinin sağ tarafı (2.5) eşitliğinin tekrar yazılmış halidir. $q(\cdot) : \mathbb{R} \rightarrow \mathbb{Z}$ olarak tanımlanan yuvarlama işleçidir.

Eğer kamera görüntüsü üzerinde birim uzunluk başına düşen piksel sayısı x ve y

eksenleri üzerinde sırasıyla m_x ve m_y ise K' matrisi şu şekilde tekrar yazılabilir :

$$K = \begin{bmatrix} fm_x & 0 & m_x p_x \\ 0 & fm_y & m_y p_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_x & 0 & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

(2.8) eşitliğinde $a_x = fm_x$ ve $a_y = fm_y$ sırasıyla x ve y eksenlerinde kameranın odak uzaklığının piksel boyutlarındaki değerleridir. Ayrıca (x_0, y_0) da (2.4) eşitliğinde geçen kamera merkezinin piksel olarak ifadesidir.

Daha genel bir ifade yazmak istenirse kare olmayan pikselleri de hesaba katıp *eğiklik parametresi* olan *s* parametresini de K matrisine dahil etmek gerekmektedir. Son hali ile K matrisi şu şekilde yazılır :

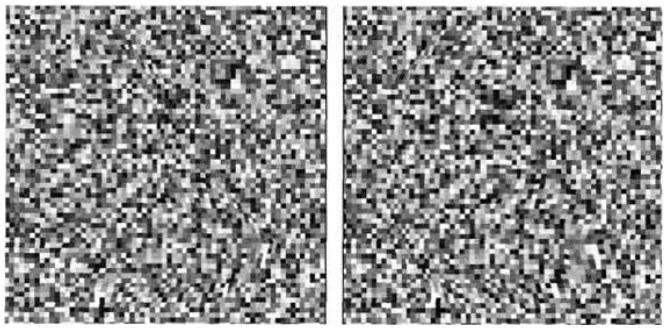
$$K = \begin{bmatrix} a_x & s & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

Normal kameralar için eğiklik parametresi, *s*, sıfır değerini alır. Bu tez çalışmasında denenen bütün kameraların pikselleri kare olarak gözlemlenmiştir ve *s* değeri bu yüzden sıfır olarak alınmıştır. a_x, a_y, x_0, y_0 değerleri her kameraya göre ve o kamera içinde çalışılan çözünürlüğe göre değişir. Bu değerleri bulmak için MATLAB programı için yazılan *Kamera Kalibrasyon Araç Kiti* [11] kullanılmıştır. Bu araç kitinin kullanımı ve değerlerin nasıl bulunduğu Ek-A'da gösterilmiştir. Kamera kalibrasyonu için detaylı bilgiler [12] [13] [14] kaynaklarından edinilebilir. Kamera modellenmesi için daha detaylı bilgiler ise [9] [15] kaynaklarından bulunabilir.

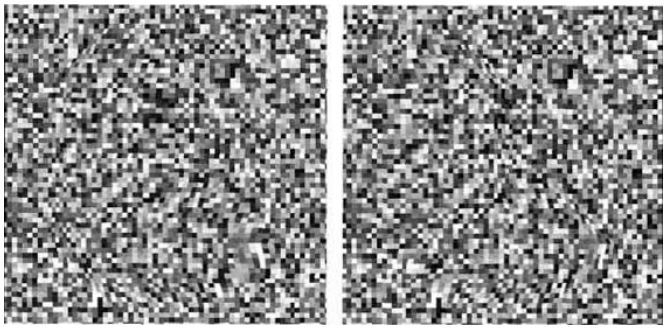
Çalışmalarımızda kullandığımız kameralar basit kameralar oldukları için profesyonel kameraların firmaları tarafından hazırlanan el kitapçıklarındaki özelliklerde belirtilen f, m_x, m_y gibi değerleri ayrı ayrı edinebilme imkanı olmamıştır. Kullanılan kamera markalarına ait internet sitelerinde en fazla f, odak uzaklığına rastlanmıştır ki bu tek başına yeterli değildir. Zaten f, m_x, m_y değerleri ayrı olarak değil de $a_x = fm_x$ ve $a_y = fm_y$ değerlerinde olduğu gibi çarpım halinde kullanılmıştır. Kamera Kalibrasyon Araç Kiti ile bulunan değerler a_x, a_y, x_0, y_0 değerleridir. f, m_x, m_y değerleri kalibrasyon ile elde edilememiştir ve üretici firma tarafından sağlanması gerekmektedir. Derinlik, burada uzaklık, hesabı için kullanılan yaklaşımda a_x, a_y, x_0, y_0 değerlerinin nasıl kullanılacağı bir sonraki bölümde anlatılmıştır.

2.3. Tek Görüş Açısı Derinlik Kestirimi Yaklaşımı

Daha önce belirtildiği gibi ayrıca (2.5)'den çıkarılabilecek bir sonuç olarak tek bir kamera bir nesnenin yönünü bulmakta işe yarasa da uzaklık bulmada genel olarak başarılı değildir. Bunun sebebi ise insanların iki göze sahip olmalarıyla benzerdir. Eğer bir nesneye ait görüntü, stereo görüntü çifti olarak doğru bir şekilde beyne gelirse ancak o zaman insan beyni nesneyi 3 boyutlu olarak canlandırabilmektedir. Burada doğru olarak gelmesinden kasıt, sağ gözün ve sol gözün görmesi gerekeni görmesi şeklinde açıklanabilir [9].



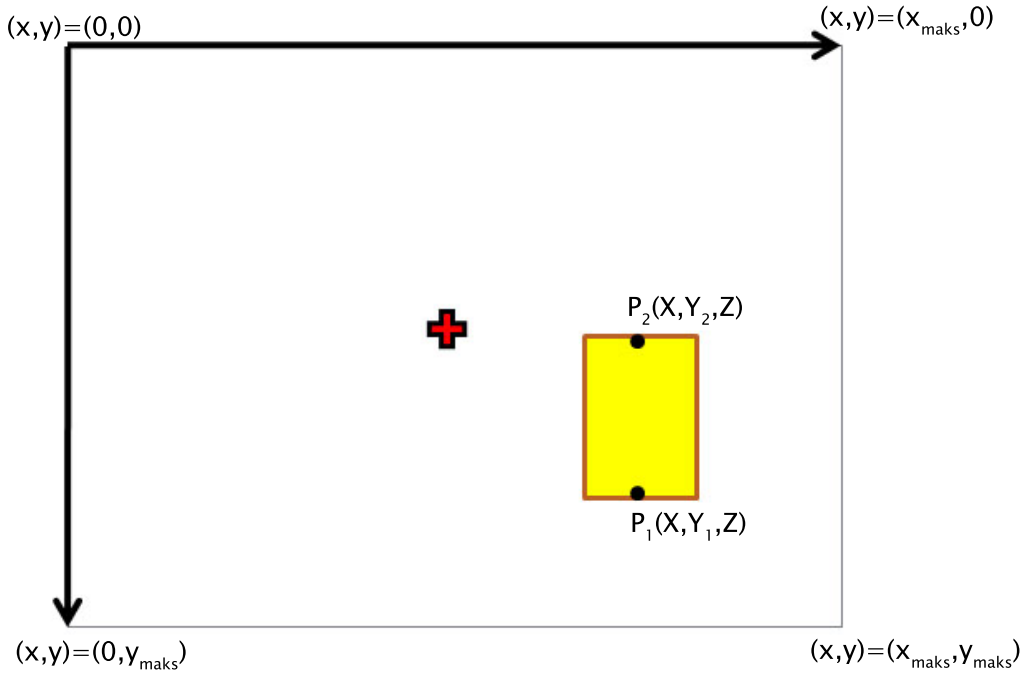
Şekil 2.3.: Asıl stereo görüntü çifti (Şekil [9] kaynağından alıntıdır.)



Şekil 2.4.: Yer değiştirilmiş stereo görüntü çifti (Şekil [9] kaynağından alıntıdır.)

Şekil 2.3.'de soldaki görüntü sol göz için, sağdaki görüntü sağ göz içindir. Bu şekile şaşı bakılırsa yani sağ göz soldaki görüntüyü sol göz de sağdaki görüntüyü görecektir. Şekil 2.3.'e bu şekilde bakıldığında dışarı doğru çıkık bir *"L"* harfi görülecektir. Eğer bu görüntü çiftini birbirleri arasında yer değiştirirsek, Şekil 2.4. elde edilir. Buradaki şekilde görülen ise içeri doğru girmiş bir *"L"* harfidir. İnsan beyni 2 boyutlu bir ekrandan veya kağıttan dahi sağ ve sol göze doğru görüntüler geldiğinde 3 boyutlu derinlik hissine sahip olarak algılayabilmektedir.

Bu çalışmada iki kamera kullanılmadığından derinlik kestirimi yapabilmek için nesneye ait bir özellik gerekmektedir. Uzaklığını bulmak istediğimiz nesne için bu özellik nesnenin gerçek boyutlarda *yüksekliği* olarak seçilmiştir. Önceden yüksekliklerinin farkını bildiğimiz iki nokta eğer robot kamerasının lensi yüksekliğinde veya buna yakın bir düzlemde ise SGD ve kamera kalibrasyon matrisinin parametreleri ile nesnenin uzaklığını yaklaşık olarak bulabiliriz. $P_1(X, Y_1, Z)$ ve $P_2(X, Y_2, Z)$ robot üzerinde veya çevrede yükseklik farkını bildiğimiz iki nokta olsunlar (Bkz. Şekil 2.5.). Kameranın görüntü düzleminde x koordinat ekseni sağa doğru pozitif değer alırken, y koordinat ekseni aşağıya doğru pozitif değer almaktadır. Bu yüzden Y_1 değeri Y_2 değerinden daha büyüktür. Ayrıca P_1P_2 robotun hareketine dik olarak seçilmiştir.



Şekil 2.5.: Uzaklık kestirimi için nokta belirlenmesi

(2.5) ve (2.9) eşitliklerine göre ;

$$\begin{aligned} \bar{y_1} &= q(Z^{-1}(a_y Y_1 + y_0 Z + sX)) \\ \bar{y_2} &= q(Z^{-1}(a_y Y_2 + y_0 Z + sX)) \end{aligned} \tag{2.10}$$

(2.10) eşitliğinde $(\bar{x}, \bar{y_1})$ ve $(\bar{x}, \bar{y_2})$ sırasıyla P_1 ve P_2 noktalarının görüntü üzerinde piksel cinsinden koordinatına denk gelmektedir, $y_1 > y_2$. Yuvarlama işleçimiz $q(\cdot) : \mathbb{R} \to \mathbb{Z}$ olarak tanımlanmıştı. Yuvarlama işlemi sırasında kalan artık kısım için de şu şekilde bir yuvarlama kalan işleçi tanımlanır :

$\tilde{q}(\cdot) : \mathbb{R} \to [0, 1), \tilde{q}(\xi) = \xi - q(\xi), \forall \xi \in \mathbb{R}.$

(2.10) eşitliğindeki $\bar{y_1}$ ve $\bar{y_2}$ gibi $\tilde{y_1}$ ve $\tilde{y_2}$ de şu şekilde yazılır :

$$\begin{aligned} \tilde{y_1} &= \tilde{q}(Z^{-1}(a_y Y_1 + y_0 Z + sX)) \\ \tilde{y_2} &= \tilde{q}(Z^{-1}(a_y Y_2 + y_0 Z + sX)) \end{aligned} \tag{2.11}$$

(2.10) eşitliğinden $\bar{y_1}$ ve $\bar{y_2}$ farkını alırsak :

$$\bar{y_1} - \bar{y_2} = Z^{-1} a_y (Y_1 - Y_2) + (\tilde{y_2} - \tilde{y_1}) \tag{2.12}$$

$\bar{h} = \bar{y_1} - \bar{y_2}$, $\tilde{h} = \tilde{y_1} - \tilde{y_2}$ ve $H = Y_1 - Y_2$ olarak ifade edersek, (2.12) eşitliğinden Z (gerçekte nesne ile robot arasındaki Z-ekseni boyunca mesafe) şu şekilde çekilir :

$$Z = a_y \frac{H}{\bar{h} + \tilde{h}} \tag{2.13}$$

Burada $\bar{h}$, görülen nesnenin SGD üzerinde piksel cinsinden izdüşümünün tamsayı halini; $\tilde{h}$, görülen nesnenin SGD üzerinde piksel cinsinden izdüşümünün gerçel değeri ile gerçel değerinin yuvarlama işleçinden sonra kalan tam sayı kısmının farkını; H ise gerçek dünyada görülen nesnenin gerçekteki yüksekliğidir. $\bar{h}$ ve $\tilde{h}$ kamera üzerinden alınan değerler iken H değeri nesnenin önceden bilinen yüksekliğidir.

(2.7) eşitliğinden ;

$$\bar{x} = q(Z^{-1} a_x X + x_0) \tag{2.14}$$

olduğu görülmektedir. Aynı şekilde $\tilde{x}$ de şu şekilde yazılır :

$$\tilde{x} = \tilde{q}(Z^{-1} a_x X + x_0) \tag{2.15}$$

Tamsayılama işleçine girmeden önce gerçek x değeri $x = \bar{x} + \tilde{x}$ olduğundan ;

$$x = Z^{-1} a_x X + x_0 \tag{2.16}$$

olarak ifade edilebilir. Buradan da X değerini çekersek ;

$$\begin{aligned} X &= Z\frac{x - x_0}{a_x} \\ &= a_y \frac{H}{\bar{h} + \tilde{h}} \frac{x - x_0}{a_x} \\ &= \frac{a_y H(\tilde{x} + \bar{x} - x_0)}{a_x(\tilde{h} + \bar{h})} \end{aligned} \tag{2.17}$$

Buradan gerçek mesafe olan D ise şu şekilde ifade edilir :

$$D = \sqrt{Z^2 + X^2} = \frac{a_y H}{\bar{h} + \tilde{h}} \sqrt{1 + \left(\frac{\bar{x} + \tilde{x} - x_0}{a_x}\right)^2} \tag{2.18}$$

Yuvarlama işleçinden dolayı ve a_y, a_x, x_0, y_0 değerlerinin kendi hata payları yüzünden D uzaklığında oluşan hatalar önemlidir. Bu değerlerden gelen hata payları tezin gelecek kısımlarında anlatılmıştır. Daha önce de belirtildiği gibi yukarıdaki çıkarsamalar için [6] [7] [8] kaynakları takip edilmiştir.

BÖLÜM 3

3. KAMERANIN ROBOTLARA ENTEGRASYONU

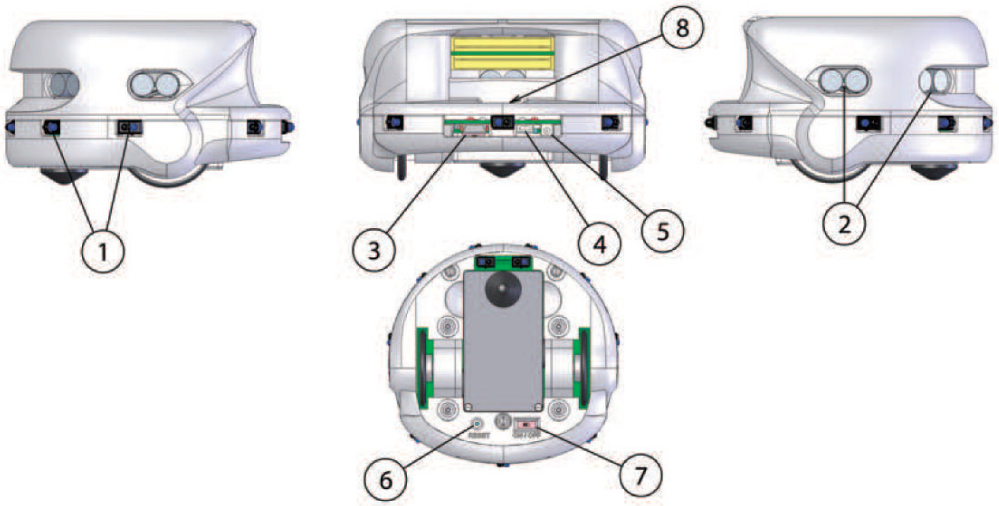
Bu bölümde, bir önceki bölümde anlatılan tek bakış açısı derinlik yaklaşımının pratikte robot üzerine nasıl uygulandığı anlatılmıştır. Yaklaşım ve sonraki bölümlerdeki teorilere uygun olabilmesi için robot üzerinde bazı eklemeler ve değişiklikler yapılması zorunlu olmuştur. Bu değişikliklerden sonra farklı görüntü işleme teknikleri denenmiş ve en uygunu ile teori robot üzerinde gerçeklenmiştir.

Öncelikle kullanılan Khepera 3 robotundan daha sonra kamera ayağı olarak kullanılan motorlardan ve görüntü işlemede kullanılan OpenCV ve denenen görüntü işleme teknikleri sırayla bahsedilecektir. En son olarak da pratikteki konulara ve de sorunlara değinilecektir.

3.1. Khepera 3 Robotunun Genel Özellikleri

Khepera 3, K-Team firması tarafından üretilen yükseltilebilen gömülü işletim sistemine, yakın ve uzak alanda nesne algılamasını sağlayan çoklu algılayıcılara, çabuk değiştirilebilir pil paket sistemine, iyi bir kademeli dişli takımı kör konumlandırmasına sahip bir robottur [16]. Küçük ve birimsel yapıya sahip bu robot, genişleyebilen veri yolu sistemi ile bir çok farklı parça ile birleştirilip kullanılabilir. Bu yapısıyla sürü sistemleri deneyleri için pratikte ideal bir robot olduğu söylenebilir.
Robotun özellikleri Şekil 3.1.'de gösterilmiştir ve numaralandırma robotun şu kısımlarına denk gelmektedir :

1. Kızılötesi algılayıcılar
2. Sesötesi Algılayıcılar
3. Ana Seri Bağlantı
4. miniAB USB Bağlantısı
5. Güç Bağlantısı
6. Yeniden Başlatma
7. Açma / Kapama
8. Dış Güç Destek Kablo Bağlantısı



Şekil 3.1.: Khepera 3 robotunun farklı açılardan görünüşü (Şekil [17] kaynağından alıntıdır.)

Şekil 3.1.'de görülen kızılötesi ve sesötesi algılayıcılar sırasıyla yakın ve uzak mesafede nesne algılanması veya robotların engelden yada birbirlerinden kaçınması için kullanılabilir. Buradaki çalışmada bu algılayıcılar kullanılmamış, yerine robota entegre edilen kamera kullanılmıştır. Fakat bu kamera algılayıcıların yerini tam olarak alması mümkün değildir çünkü kızılötesi algılayıcıların çalıştığı kısa mesafelerde, uzaklık ölçümü için kullanılan nesne çok yakına geldiği için kamera ile ölçüm yapılamamaktadır. Ayrıca, sesötesi algılayıcıların çalıştıkları uzak mesafelerde de uzaklık ölçümü için kullanılan nesne çok küçülmekte neredeyse görüntüden kaybolmaktadır. Robotların hareketini engellememek için dış güç destek kablosu da kullanılmamıştır. Bu yüzden bu parçaların detaylı bilgilerine değinilmemiştir. Ayrıntılı bilgi için [16] ve [17] kaynaklarına bakılabilir.

Khepera 3, temel olarak içinde işletim sistemini barındıran KoreBot adlı ana işlemci kartıyla beraber yada bu kart üzerine takılmadan kullanılabilir [16]. KoreBot içinde bir çeşit açık kaynak kodlu gömülü linux işletim sistemi olan Ångström linux işletim sistemi yer almaktadır. Bilgisayar üzerinde kullanılan linuxlerdeki bir çok paket ve kütüphane uygun yöntemler kullanılarak robotun içindeki linux işletim sistemine atılabilir. Genelde kişisel bilgisayarlarda kullanılan linuxlerden farklı olmasının asıl sebebi ise farklı mimariye sahip ARM işlemci türü olan Intel PXA255 XScale ARM işlemcisi ile çalışıyor olmasıdır.

Farklı mimariye sahip işlemci üzerinde çalıştığı için kendi kişisel bilgisayarlarımızda derlediğimiz kodlar doğrudan robot içine atılamamaktadır. Bunun için çapraz derleyici dediğimiz kendi bilgisayarımızda kurulan ama kodları hedef mimari için derleyen korebot-oe-tools adı verilen bir derleyici kullanılmıştır. Hazırlanan kodlarda çapraz derleyici standart C kütüphaneleri dışında 3 farklı kütüphane kullanmışlardır. Bunlar :

1. KoreBot Kütüphanesi
2. Khepera 3 Araç Kiti Kütüphanesi
3. OpenCV Kütüphanesi

Genellikle algılayıcılar, robot tekerlerini kontrol eden motorlar ve kendi algoritmalarımızı denemek için Khepera 3 Araç Kiti Kütüphanesi kullanılmıştır. Daha sonra ihtiyaç olduğunda kameradan görüntü almak, görüntü işlemek ve robota kontrol sinyali olarak gönderebilmek için OpenCV kütüphanesi kullanılmıştır. Khepera 3 Araç Kiti Kütüphanesi'nden daha önce robotu kontrol etmek için de kullanılan ama bu çalışmada sadece kameranın altındaki servo motorları kontrol etmek için kullanılan KoreIOLE genel girdi çıktı işlem kartını kontrol etmek için kullanılan KoreBot kütüphanesi kullanılmıştır.

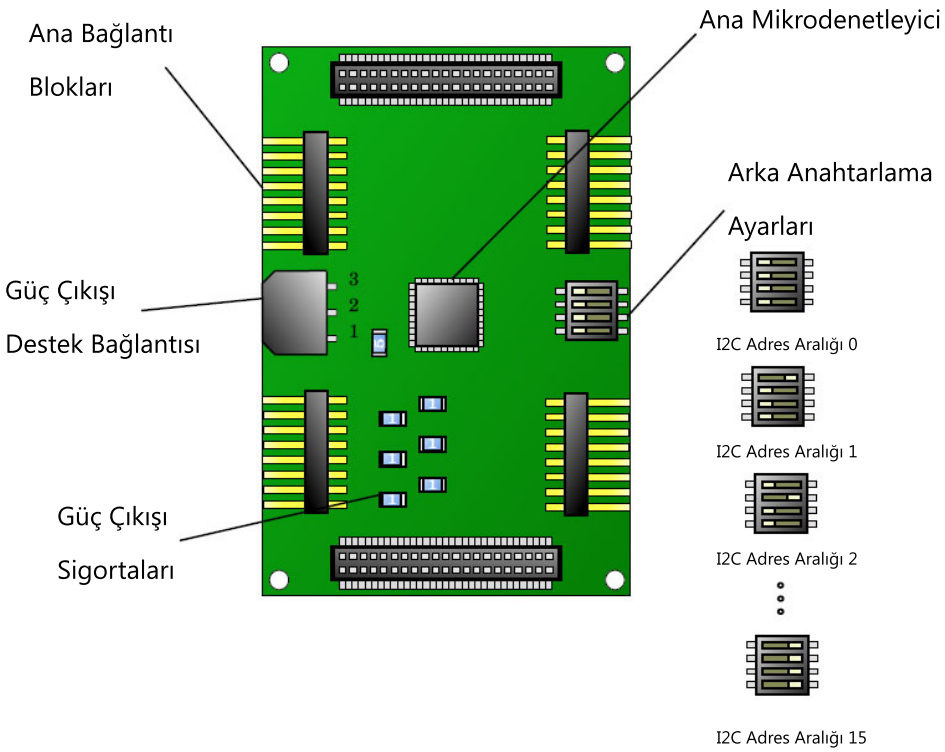
3.2. Khepera 3 ile Servo Motor Denetimi

Dizilim deneylerinde robot sürüsü belirli bir yönde ilerlerken her bir robotun izlemesi gereken hedefler yada robotlar bulunmaktadır. Özellikle bir robotun kamerası sabitken göremeyeceği sağındaki veya solundaki robotu izleyerek ileri doğru hareket etmesini gerektiren durumlar söz konusudur. Böyle durumları çözebilmek için kameranın robot üzerinde kontrollü bir şekilde döndürülebilmesi gerekmektedir. Kameraların döndürülebilmesi için de açıları kontrol edilebilir olduğundan çeşit olarak servo motorlar kullanılmıştır.

Kamera ayağı olarak kullanılan servo motorlar, temel olarak robotun hareketinden hemen hemen bağımsız olacak şekilde kamerayı her yöne döndürebilmek ve veri alabilmek için kullanılmışlardır. Khepera üzerinden servo motorları kontrol edebilmek için K-Team tarafından hazırlanan genişleme kartlarından biri olan KoreIOLE kartı kullanılmıştır. KoreIOLE kartı bir çok genel işleve sahip bir karttır.

KoreIOLE üzerindeki bütün dijital girdi çıktı pinleri standart girdi çıktı olarak kullanılabildiği gibi düşük frekanslı *darbe genişlik modülasyonu (PWM)* sinyali

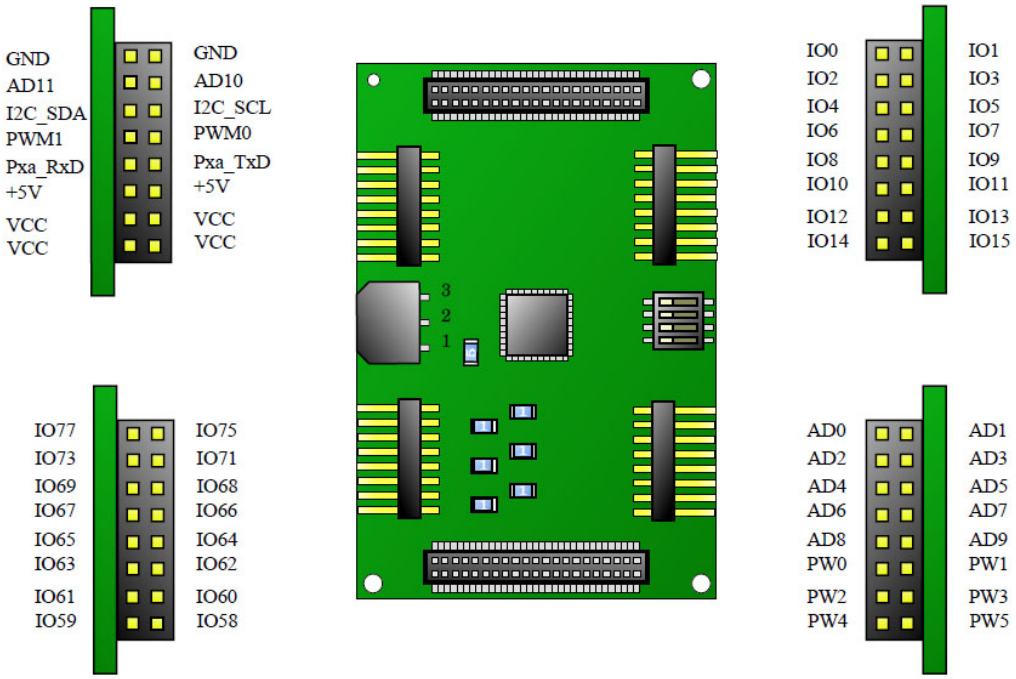
üretmek için kullanılabilir [18]. Kendi üzerinden diğer cihazlarla haberleşebilmek için I2C veri yolu olarak da işlev görür. İşletim sistemini üzerinde bulunduran Korebot kartına ait darbe genişlik modülasyonu için PXA PWM, seri haberleşmeler için PXA TxD ve RxD ile genel amaçlar için kullanılabilecek PXA GPIO sinyallerini bulundurur.



Şekil 3.2.: KoreIOLE donanımı (Şekil [18] kaynağından alıntıdır.)

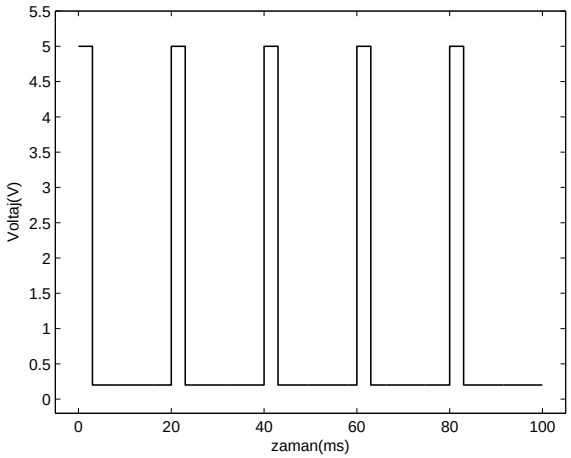
Şekil 3.2.'de görüldüğü gibi KoreIOLE merkezindeki bir mikrodenetleyici ve güç çıktı devrelerinden oluşmuştur. Ana mikrodenetleyici haberleşmeyi ve temel girdi çıktı işlevlerini yerine getirmeyi sağlamaktadır. Bir diğer modül de 1 ampere kadar üzerinden akım sürebilen transistörlere sahip güç çıktı devreleridir.

Şekil 3.3.'de daha önceden bahsi geçen sinyaller için ayrılan pinler görülmektedir. Ayrıca AD pinleri analogdan dijitale sinyal dönüştürmek için kullanılan pinlerdir. Servo motor kontrolü için darbe genişlik modülasyonu kullanılmıştır. KoreIOLE üzerindeki IO genel amaçlı girdi çıktı pinleri Korebot kütüphanesi kullanılarak 3 amaçla yani girdi, çıktı ve darbe genişlik modülasyonu için kullanılabilir. IO pinleri darbe genişlik modülasyonuna ayarlandığında çıktıdan alınacak sinyalin frekansı da ayarlanabilmektedir. Her ne kadar darbe genişlik modülasyonu için kullanılan 50 Hz frekansına çekilse de temiz bir sinyal elde edilememiştir. Tipik bir darbe genişlik



Şekil 3.3.: KoreIOLE pin dağılımı (Şekil [18] kaynağından alıntıdır.)

modulasyonu sinyali Şekil 3.4.'deki gibidir. Temiz sinyalden kasıt, çıktı pininden doğrudan alınan sinyalin Şekil 3.4.'deki gibi keskin iniş çıkışlara sahip olmamasıdır.



Şekil 3.4.: Darbe Genişlik Modülasyonu Sinyali

Çizelge 3.1. : KoreIOLE PWM Sınırları

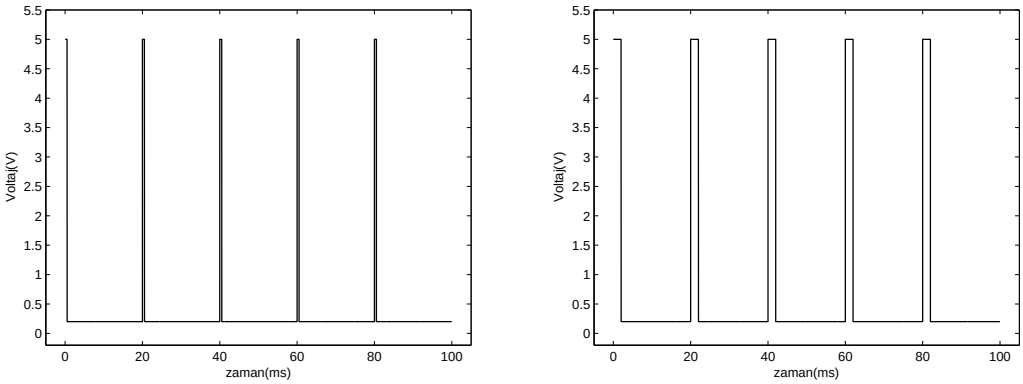
		periyot	register değeri
Darbe Genişlik Modülasyonu Sinyali	50 Hz	20 ms	—
Minimum Oran	1%	0.2ms	1
Maksimum Oran	10%	2ms	255

Servo motorların ancak temiz bir darbe genişlik modülasyonu sinyali verildiğinde düzgün çalıştığı gözlenmiştir, aksi takdirde motorda çok fazla titreme olmaktadır. Bu yüzden herhangi bir IO pini ayarlandığında frekansı sıfıra çekilmelidir. Ancak bu şekilde darbe genişlik modülasyonu moduna geçilir. KoreIOLE kartının üretebildiği darbe genişlik modülasyonu sinyalinin özellikleri Çizelge 3.1.'deki gibidir :

0.2 ms genişliğinde bir sinyal çoğu servo motorun çalışmaya başladığı sinyal genişlğinin altında ve sorun yaratmazken, üst sınır olarak konulan 2 ms servo motorların sinyal aralığı olarak üst sınırlarından daha küçük bir değerdir. Bunu şu şekilde ifade edebiliriz. Çizelge 3.1.'de kontrol girdisi olarak karta 1-255 arasında bir değer verilebildiği görülmektedir. Ancak kullanılan servo motorlar 1-40 değer aralığında çalışmamaktadır. Her servo motorun kendine has çalıştığı bir sinyal aralığı vardır. Robot üzerinde kullanılan servo motor 45-255 değer aralığında çalışmakta ve en fazla 145 derecelik bir dönüş sağlamaktadır. Motor girdisi olarak verilen 45 değerini zaman değerine çevirirsek $44 * (2 - 0.2)/254 + 0.2 = 0.512ms$ değerini elde ederiz. Aynı şekilde 255 değeri için de hesaplarsak $254 * (2 - 0.2)/254 + 0.2 = 2ms$ değeri olan üst sınır elde etmiş oluruz. Kullanılan servo motora ait firma dökümanlarında, servo motorunun hangi sinyal aralığında çalıştığı gösterilmemektedir. Benzer servo motorların firma dökümanlarına bakıldığında sinyal aralığının 550ms-2450ms olduğu görülmektedir [19]. Buna göre servo motorun 180 derece dönüş yapabilmesi KoreIOLE kartı ile mümkün değildir. Şekil 3.5.'de Khepera 3'e verilebiliecek minimum ve maksimum PWM sinyallleri görülmektedir.

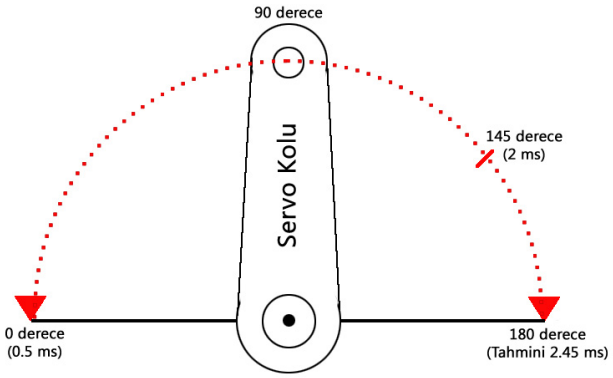
0-145 derece arasında kontrol edilebilen servo motora ait servo kolu da Şekil 3.6.'da gösterilmiştir.

Motor olarak SpringRC SM-S3317M Modeli bir servo motor kullanılmıştır. Robot üzerine takılan servo motorlar Şekil 3.7.'de görülmektedir. Şekil 3.7.'de de görüldüğü gibi her bir servo motordan üç renk kablo çıkmaktadır. Bunlardan kırmızı renk, servo motorun +5V'a takılan ucu; siyah renk, toprak bağlantısı ucu; beyaz

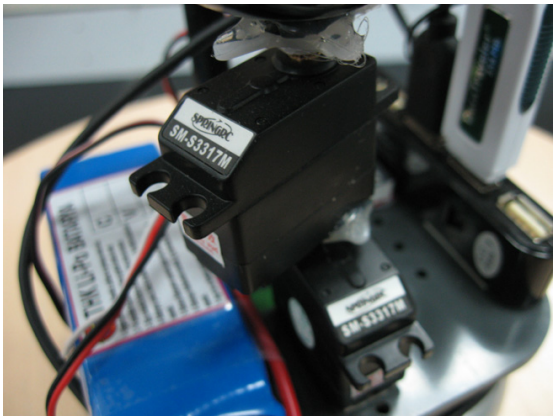


(a) Minimum PWM Sinyali (b) Maximum PWM Sinyali

Şekil 3.5.: Khepera 3 için PWM Sinyali Sınırları

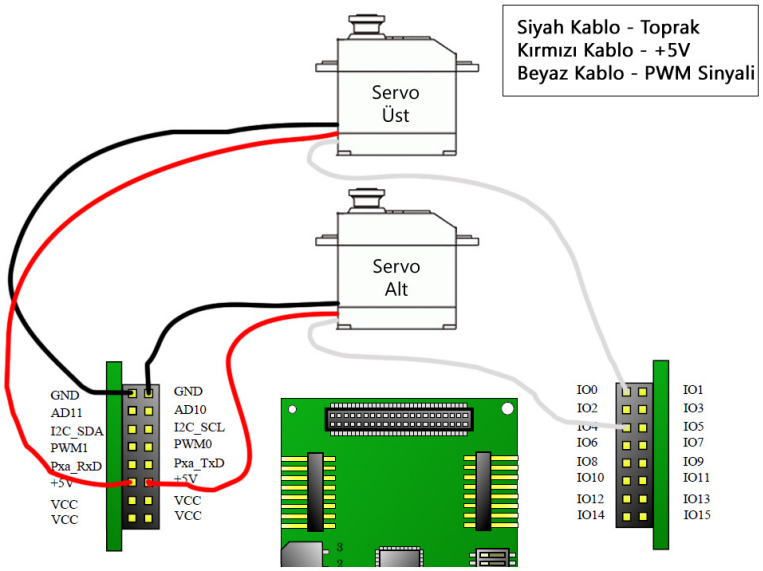


Şekil 3.6.: Servo Kol



Şekil 3.7.: Robot Üzerindeki SM-S3317M Servo Motorlar

renk de PWM sinyali için ayrılan kablo ucudur. Robot üzerindeki servo motorların KoreIOLE kartına hangi pinlerden bağlandığı Şekil 3.8.'de görülmektedir.



Şekil 3.8.: Servo Motorlar ile KoreIOLE Bağlantısı

Servo motorun 45-255 değer aralığı içinde hangi girdiye göre ne kadar döndüğü yaklaşık olarak ölçülmüştür ve kodlama yapılırken servo motorlar için *çözünürlük* olarak kullanılmıştır. Daha sonra servo motorun dönebildiği maksimum açının yarısında $(140°/2 = 70°)$ girdi değerine bakılmış ve bunun yaklaşık olarak 145-155 değer aralığında olduğu görülmüştür. Motorların üst üste muntazam bir şekilde istenen şekilde birleştirilememektedir çünkü metal dişlilere sahip servo motorun 24 dişlisi vardır ve iki dişli arasındaki 15 derecelik açı istenilen şekilde üst üste birleşmesine mani olmaktadır. Bu yüzden motorların istenen şekle en yakın şekilde birleştirilmesi sağlanmış ve daha düzgün olması için de kodlama içinde her servoya birbirine yakın değerlerde olan *hiza* değeri atanmıştır. Temel olarak motorların ilklendirilmesi Çizelge 3.2. 'deki gibidir:

Servo motorun denetiminde kullanılan metod için Ek-B'deki $servo\_control$ metoduna bakılabilir.

Robot üzerinde kullanılan servo motoru KoreIOLE'den alınabilen sinyal ile en fazla 145 derece döndürebilmek mümkün olmuştur. Kullanılan kamera açısının 60 derece yatay görüş açısına sahip olmasına ve toplamda 200 derecelik görüş elde edilmesine rağmen robotun sağında veya solunda geride kalan nesneleri görebilmesi için daha

Çizelge 3.2. : Khepera 3 Servo Motor İlklendirme Komutları

```
kio_ConfigIO(koreio,alt_motor_pin,2) //GPIO pini olan alt motor pini
(pin-4) '2' ile PWM sinyali pini olarak atanır.

kio_ConfigIO(koreio,ust_motor_pin,2) //GPIO pini olan üst motor pini
(pin-0) '2' ile PWM sinyali pini olarak atanır.

kio_ChangePWM_freq(koreio,0)    //PWM sinyali frekansı '0' değerine
çekilerek standart PWM sinyali üretilir.

kio_ChangePWM_ratio(koreio,alt_motor_pin,alt_motor_hiza) //Pin-4'e
alt motor hiza değeri olan(150) verilir, servo kolu ortalanır.

kio_ChangePWM_ratio(koreio,ust_motor_pin,ust_motor_hiza) //Pin-0'a
alt motor hiza değeri olan(150) verilir, servo kolu ortalanır.
```

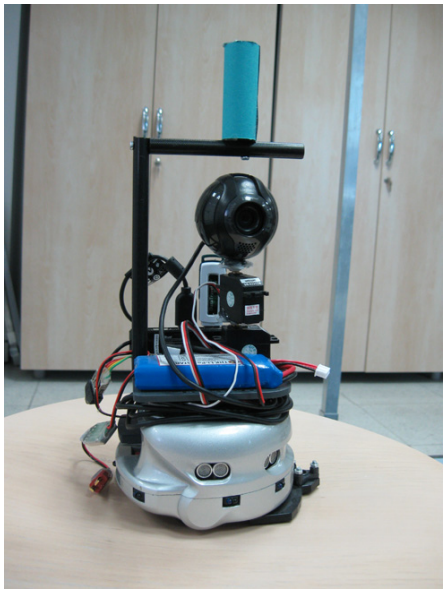
fazla görüş açısına ihtiyacı olduğu görülmüştür. Bu yüzden iki servo motor üst üste takılmış ve görüş açısına 145 derece daha eklenmiştir. Böylece toplamda yaklaşık 340 derecelik bir görüş açısı elde edilmiştir. Kullanılan servo motorun çalışmaya başladığı 0.5 ms sinyal genişliğinde ve o civarda verilen sinyal genişliklerinde motorda titreme görüldüğü için güvenli bir aralık olarak en fazla motoru 120 derece döndürecek şekilde darbe genişlik modülasyonu sinyali kullanılmıştır. Güvenli çalıştırma aralığı olarak seçilen bu aralıkta çalışmalarda robot 300 derece görüş açısı ile çalıştırılmıştır.

3.3. Khepera 3 Robotuna Sonradan Eklenen Parçalar

Daha önceki kısımlarda anlatılan parçalar yani işlemci ile işletim sistemini barındıran KoreBot ve servo motor denetimini sağlayan KoreIOLE kartları K-Team firmasının ürettiği ve robotlar ilk satın alınırken yada sonradan yollanmış parçalardı. Bu kısımda anlatılacak parçalar ise tez çalışmasına özel olarak robotun üzerine entegre edilmiş parçalardır. Khepera 3 robotunun entegre edilen parçalardan sonraki hali Şekil 3.9.'daki gibidir.

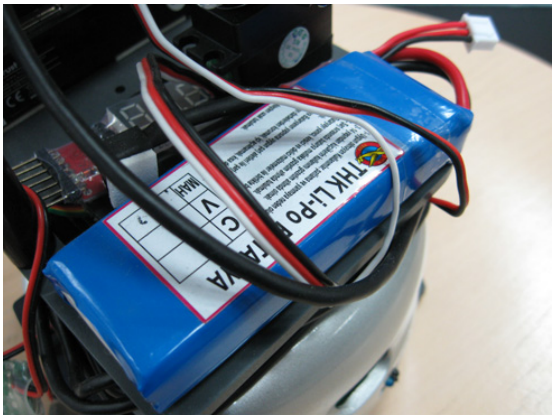
3.3.1. Li-Po Pil

Li-Po pil, eklenmesi gerekli olan parçalardandır. Khepera 3 robotunun altında bulunan kendi pili robotun tekerlerini çeviren servo motorlar için, KoreBot, haberleşme vb. görevler için kullanılmaktadır. Dışarıdan eklenen parçalardan servo motorların her birinin 70-150 mA akım çektiği ölçülmüştür. Kameranın kendisinin de 500 mA çektiği özelliklerinde yazmaktadır. Fazladan gelen bu akım yükünü Khepera 3'ün kendi pili karşılayamamaktadır. Li-Po pil takılmaz ise robotun üzerine takılan servo



Şekil 3.9.: Khepera 3 Robotunun En Son Hali

motorların ilklendirme aşamasında robot kilitlenmektedir. Kullanılan Li-Po piller 2 hücreli olup her biri en fazla 4.2V, toplamda 8.4V değerine kadar şarj edilebilmektedir. Zaman içinde düşen voltaj değeri dikkat edilmelidir. Herbir hücrenin voltajının 3.5V değeri altında düşmemesine özen gösterilmiştir. Bu değerden sonra voltaj düşüş hızı artmaktadır. Deneyler sırasında farklı li-po piller denenmiştir. 1300 mAh, 1700 mAh ve 2200 mAh'lik piller denenmiştir. Şekil 3.10.'de görülen li-po pil 7.4V 1700 mAh 20C özelliklerine sahip bir pildir.



Şekil 3.10.: Li-Po Pil

3.3.2. Turnigy UBEC 3A Voltaj Regülatörü

Dışarıdan destek güç olarak bağlanan Li-Po pilin voltajı 7.4V'dur. Khepera 3'ün dışarıdan güç beslemesi ise KoreBot üzerinden 5V verilerek yapılabilmektedir. Aradaki voltaj farkını düşürebilmek ve de 5V değerine sabitleyebilmek için Turnigy UBEC[1] voltaj regülatörü kullanılmıştır. Voltaj değeri düşse bile bu parça sayesinde robotun 5V besleme ile düzgün çalışması sağlanmıştır. Şekil 3.11.'de bu parça görülmektedir.

Şekil 3.11.: Turnigy UBEC 3A Voltaj Regülatörü

3.3.3. USB Hub

Kameranın takılabilmesi için gerekli USB bağlantısı aslında yoktur. Bu bağlantıyı kurabilmek için K-Team'in hazırlamış oldugu kamera kartının bozulup kablolarının, USB bağlantı ucuna lehimlenmesi gerekmektedir. Bu şekilde hazırlanan tek USB çıkışını artırabilmek için USB Hub denilen bir çoklayıcı gereklidir. Şekil 3.12.'de 4 USB girişi olan USB Hub'a takılan flash bellek ve kamera görülmektedir.

3.3.4. Flash Bellek

Şart olmamakla beraber düzenekte kullanılması gereken parçalardan biri de dışarıdan takılacak bir flash bellektir. Khepera 3'ün işletim sistemini barındıran 32 MB kapasiteli flash belleğin, 1 MB'lık kısmı kernel için, 31 MB'lık kısmı işletim sistemi için ayrılmıştır. İşletim sistemi 31 MB'lık kısmın yaklaşık 29 MB'lık kısmını doldurmakta ve geriye 2 MB kadar bir yer kalmaktadır. Derlenen OpenCV kütüphaneleri 5 dosyadan oluşur ve yaklaşık 4 MB yer kaplarlar. Khepera 3 robotunun

[1] Ultimate Battery Eliminator Circuit

Şekil 3.12.: USB Hub

kalıcı hafızasında kalan 2 MB'lık kısım yeterli olmadığı için dışarıdan flash bellek takılmıştır. Şekil 3.13.'de kullanılan flash bellek görülmektedir.

Şekil 3.13.: USB Flash Bellek

USB flash bellek USB Hub'a takıldığı anda arm-linux işletim sistemi tarafından tanınmaktadır. Fakat, OpenCV kütüphaneleri flash belleğe konulduğu zaman bu kütüphanelerin okunmasında sorun çıkmaktadır. Sorunun kaynağı ise OpenCV kütüphanelerinin bir linux işletim sisteminde derlenmesi ve hedef işletim sisteminin de arm-linux işletim sistemi olmasıdır. Günlük hayatta kullandığımız flash bellekler genelde Windows işletim sisteminin FAT32 yada NTFS dosya sistemi yapısındadır. OpenCV kütüphaneleri flash bellek üzerinden okunmaya çalışılırken linux işletim sistemine özgü sembolik bağlantı[2] oluşturur. FAT32 yada NTFS dosya sistemleri

[2]ing: Symbolic Link

üzerinde sembolik bağlantı oluşturulamamaktadır. Flash bellekler öncelikle linux dosya sistemi olan *ext2* formatı ile formatlanmıştır. Bu aşamada da sorun yaşanmış; Khepera 3, ext2 dosya formatındaki flash hafızayı görmemiştir. Bunun sebebi ise Khepera 3 için işletim sistemi görüntüsü hazırlanırken ext2 format desteğinin fabrika ayarlarında açık olmamasıdır. Ext2 format desteği açılıp Khepera 3'ün kerneli tekrar derlenip robot çalıştırıldığında OpenCV kütüphaneleri sorunsuz çalışabilmiştir.

USB flash bellekler üzerine değinilebilecek bir diğer konu ise flash belleklerin robot çalışırken zaman zaman robotun kitlenmesine sebep olmasıdır. Robotlar deneylerin ilk aşamalarında genelde aldıkları görüntüyü kaydederek çalışmışlardır. Pilin bitmesi yada robotun doğru çalışmadığı görüldüğü zaman kullanıcı tarafından durdurulduğu gibi durumlarda robotun kaydetmekte olduğu görüntüyü yarım olarak kaydetmesi gibi bir durum söz konusudur. Yarım kaydedilen bir görüntü USB flash bellek üzerinde görülmemekte bu yüzden de silinememektedir. Silinemediği ve sorunlu olarak flash bellekte durduğu için robotun bir sonraki çalışmasında aynı isim ve numara ile bir görüntü kaydedilmeye çalışıldığında çakışma olmakta ve robot kitlenmektedir. Doğrudan silinemediği için dosya sisteminin tamir edilmesi gerekmektedir. O da şu şekilde yapılır:

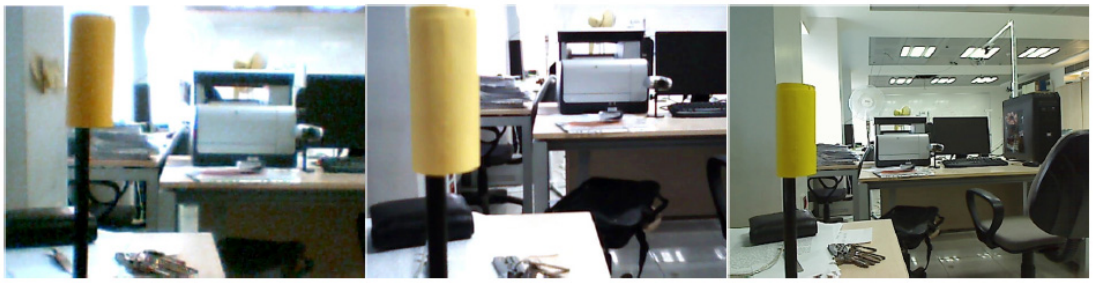
umount /dev/sda2: /dev/sda2 konumunda olan flash bellek umount komutu ile çıkartılır.

fsck.ext2 /dev/sda2: ext2 dosya sistemi yapısındaki flash bellek fsck komutu ile tamir edilir.

USB flash belleğin robota takılmasında iki amaç vardır. Bir tanesi OpenCV kütüphanelerini saklamak diğeri de robotun nasıl gördüğünü anlayabilmek için robotun kamerayla aldığı görüntüleri kaydetmesidir. Aslında robotun kalıcı hafızasında yer olmamasına karşın geçici hafızası olan *ram* kısmında yaklaşık 32 MB'lık bir boşluk vardır. OpenCV kütüphaneleri kablosuz bağlantı kurulup robotun içine atılırsa dışarıdan USB flash bellek kullanılmadan robotun kodları çalışabilmektedir. Bu da denenmiş ama kullanılmamıştır çünkü toplamda yaklaşık 4 MB yer kaplayan 5 OpenCV kütüphanesinin robotun geçiçi belleğine yollanması 3-4 dakika almaktadır. Her seferinde her robot için bu yollama işleminin zor olması sebebiyle çalışmalar USB flash bellek kullanarak devam ettirilmiştir.

3.3.5. Robot Kamerası

Robot kamerası ve buna ait linux sürücüsü bu tez çalışmasının en çok zaman alan kısımlarından biridir. Tez çalışmasında uygun olarak seçilen en son kamera, basit, ucuz, hızlı çalışan ve oldukça net yüksek çözünürlüklü görüntü alabilen bir kameradır. Yeni kamera arayışına gidilmesinin sebebi, K-Team tarafından sağlanan kameraların görüntüsünün çok kötü olması, ortam ışığına göre çok değişken görüntü vermeleri ve dar açıya sahip olmalarıdır. Üç kamera arasındaki fark Şekil 3.14.'de görülmektedir.



(a) Logitech Quickcam Chat (b) Logitech E2500 (c) Logitech C600

Şekil 3.14.: (a) K-Team'in eski kamerası, (b) K-Team'in yeni kamerası, (c) Sonradan entegre edilen C600 kamerası

Kullanılan Logitech C600 kamerası denenen 9 kameranın en sonuncusudur. 2 Megapiksel CMOS algılayıcısı olan bu kamera Şekil 3.15.'de görülmektedir.

Şekil 3.15.: Logitech C600 Kamerası

Çizelge 3.3. : Kamera Sürücüsüne Yeni Kamera Bilgilerinin Eklenmesi

```
/* Logitech Webcam C500 HD */
{ .match_flags   = USB_DEVICE_ID_MATCH_DEVICE
    | USB_DEVICE_ID_MATCH_INT_INFO,
  .idVendor       = 0x046d,
  .idProduct      = 0x0807,
  .bInterfaceClass = USB_CLASS_VIDEO,
  .bInterfaceSubClass = 1,
  .bInterfaceProtocol = 0,
},
```

3.4. Kamera Linux Sürücüsü

Linux üzerinde kamera sürücüleri genelde 2 başlık altında toplanabilir, *gspca* ve *uvcvideo*. Khepera 3 üzerindeki K-Team tarafından üretilen hem eski hem de yeni kameranın sürücüleri, K-Team tarafından kullanıcının kullanımına hazır hale getirilmiş *gspca* tipi linux video sürücülerdir. *Gspca* tipi linux video sürücüleri genellikle eski kameraların sürücüleridir. Son yıllarda üretilen ve Tak-Çalıştır türünde olan kameralar ise *uvcvideo* tipi linux video sürücüsü kullanırlar. *Uvcvideo* linux video sürücüsünün en yeni versiyonları çapraz derlemeye müsait değillerdir. Bu tez çalışmasında da çapraz derlemeye müsait olan en son versiyon olan uvcvideo r263 kullanılmıştır. Logitech C600 kamerası ve daha öncesinde denenen C500 kamerası yeni kameralar oldukları için uvcvideo r263 video sürücüsü çapraz derlendiğinde robot üzerinde çalışmamışlar, robot tarafından kamera olarak algılanmamışlardır. Bu sorunun üstesinden gelebilmek için uvcvideo r263 içindeki uvc_driver.c dosyasına kamera hakkında bilgiler eklenmelidir. Ekleme Çizelge 3.3. 'deki gibi yapılmıştır.

Daha önceden denenen Logitech C500 sürücüsü ile de çalışabildiği için Logitech C600 için ayrıca bir ekleme yapılmamıştır. Uvcvideo tipi sürücü kullanan bir kamera seçilmesindeki sebeplerden biri de uvcvideo sürücüsü kullanan kameraların gspca sürücüsü kullanan kameralara göre daha hızlı açılabilmeleri ve çalışabilmeledir. Khepera 3, gspca sürücülü bir kamerayı 10 saniyede açabilirken, uvcvideo sürücülü bir kamerayı 1 saniyede açabilmektedir. Çapraz derlenen uvcvideo sürücüsü ile alakalı çözülemeyen bir sorun mevcuttur. Muhtemelen Khepera 3'ün USB bağlantılarını denetleyen KoreBot üzerinde bulunan OTG[3] çipinin sürücüsünden kaynaklanan bir sorun yüzünden, robot üzerinden kamera bir sefer çalıştırıldıktan sonra aynı kod çalışmamakta ancak robotun işletim sistemi kapatıp açıldığında çalışabilmektedir.

[3] On-The-Go

Bir sonraki kısımda robot üzerinde görüntü işlememize imkan veren OpenCV ve de görüntü işleme teknikleri üzerinde durulacaktır.

3.5. OpenCV ve Görüntü İşleme

OpenCV açık kaynak kodlu bir bilgisayarla görme kütüphanesidir. C ve C++ dilleri kullanılarak yazılan bu kütüphane Linux, Windows ve MAC OS X işletim sistemleri üzerinde çalışabilmektedir. İşlem gücünde verimlilik için tasarlanan OpenCV, gerçek zamanlı uygulamalar üzerine odaklanılarak hazırlanmıştır. Bilgisayarla görme alanında 500 fonksiyondan daha fazla fonksiyona sahip olan OpenCV, fabrika ürün denetimi, tıbbi görüntüleme, güvenlik, kullanıcı arayüzü, kamera kalibrasyonu, stereo görme ve robotik gibi bir çok alanda kullanılabilmektedir [20].

OpenCV ile yapılan çalışmalar öncelikle bilgisayar üzerinde Ubuntu Linux işletim sistemi üzerinde denenmiş, daha sonra robotun üzerindeki armlinux işletim sistemi için farklı bir bilgisayarda çapraz derleme yapılmıştır. Uzaklık kestirimi için kullanılan nesneyi bulabilmek için birkaç farklı yöntem denenmiştir. Bunların en önemlileri aşağıda anlatılmıştır.

3.5.1. RGB Renk Uzayında Filtreleme

Birçok renk uzayından biri olan RGB renk uzayı doğrusal bir renk uzayıdır. Bu renk uzayında mevcut renkler eksenleri R, G ve B yani sırasıyla kırmızı, yeşil ve mavi olacak şekilde birim küp üzerine oturtulmuştur [15].

RGB renk uzayında yapılan filtreleme çalışmalarında bulmak istenilen nesnenin R, G ve B renk aralıkları belirlenmiş ve kameradan alınan görüntü üzerinde bu aralığa uyan pikseller beyaz olarak işaretlenmiştir. RGB renk uzayında daha sonra bahsedilecek olan HSV renk uzayında kullanılan yöntemlerle hemen hemen aynı yöntemler kullanılmasına rağmen filtreleme yapıldığında ortamın ışık yoğunluğuna, ışığın geliş açısına ve çevredeki nesnelerin filtrelemede kullanılan renk aralığına çok fazla dahil olmasından dolayı başarılı olunamamıştır. Bahsedilen fazladan piksellerin de filtrelenmiş görüntüde gözükmesi Şekil 3.16.'de görülmektedir.

Şekil 3.16.'da görüldüğü gibi görüntüde eli oluşturan RGB değerleri filtre aralığında bulunduğu için nesne ile beraber filtreden çıkmışlardır. Bu durum sürekli hareket halinde ve daha karmaşık arka planı gören çalışmadaki robotlar için daha kötü sonuçlar doğurabilmektedir.

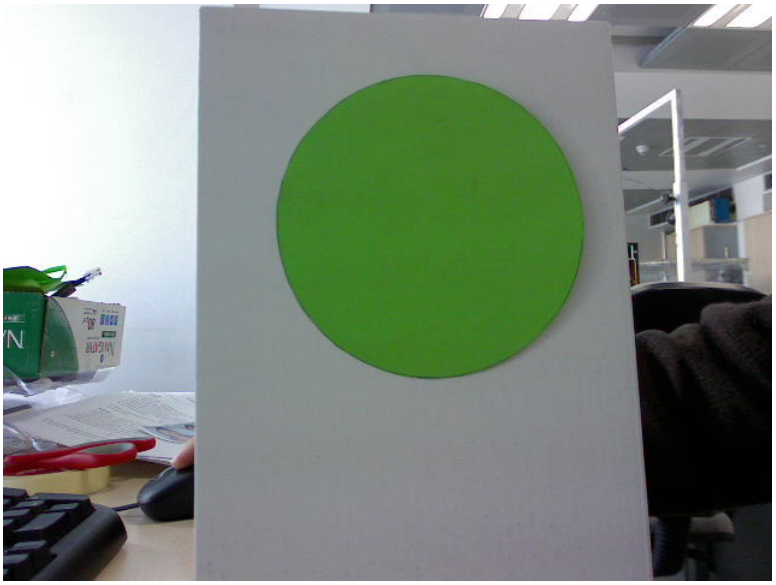
Şekil 3.16.: RGB Renk Uzayında Filtrelenmiş Nesne Örneği

3.5.2. Dairesel Hough Dönüşümü

Ortamdaki ışığa ve nesnelere karşı çok duyarlı olan renk tabanlı nesne bulunmasındaki sorunlardan dolayı şekil tabanlı nesne bulunması yöntemleri çalışılmıştır. Herhangi bir nesnenin bulunması için kullanılabilen Genel Hough Dönüşümü işlem gücü olarak elimizdeki robotları yavaşlatabileceğinden daha özel bir versiyonu olan Dairesel Hough Dönüşümü [21] denenmiştir.

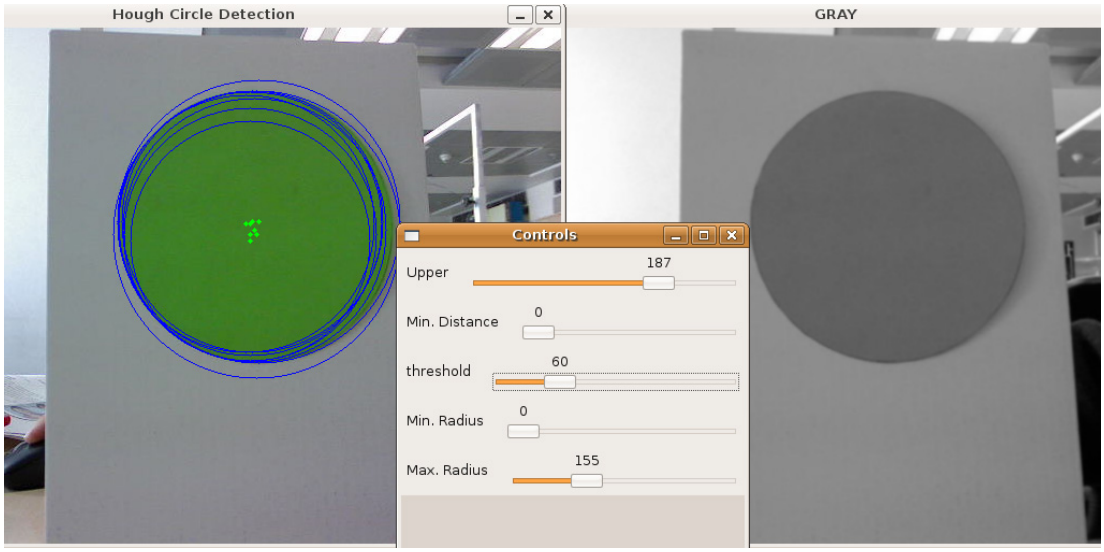
OpenCV içinde bulunan Dairesel Hough Dönüşümü fonksiyonu için algoritmayı biraz daha hızlandıran Hough Eğim Yöntemi kullanılmıştır [20]. Bu yöntemde öncelikle alınan görüntünün Canny kenar bulma yöntemi kullanılarak kenarları çıkarılır. Kenarları çıkarılmış yeni görüntüde sıfırdan farklı piksel değerine sahip bütün pikseller, Sobel x ve y türevleri alınarak yerel eğimleri bulunur. Daha sonra bu eğim noktalarından aday çember merkezi noktaları bir toplayıcı düzlemi ki burada Hough dönüşümü olarak elde edeceğimiz görüntü, oluşturulur. Aday çember merkezi noktalarından daha önceden belirlenen bir yarıçap aralığında yine daha önceden belirlenen bir eşik değeri sınır değer olarak belirlenerek kontrol edilir. Bu aday çember merkezi noktalarının gerçekten çember merkezi olup olmadığı merkez noktasının değeri, bakılan yarıçapta sıfırdan farklı piksel değerleri içeriyorsa yükseltilerek yapılır. Böylece aday çember merkezi noktaları ve bu merkezlerin çevresindeki daireler bulunmuş olur.

Dairesel Hough Dönüşümü için kullanılan Şekil 3.17.'deki daire, çevresine göre kontrastı fazla olan net gözüken bir dairedir. Şekil 3.18.'de OpenCV içindeki dairesel hough dönüşümü fonksiyonu ile bulunan daireler ve hazırlanan arayüz görülmektedir. Belirli ve net bir daire için başarılı olan bu fonksiyon, dairenin



Şekil 3.17.: Dairesel Hough Dönüşümü İçin Kullanılan Görüntü Örneği

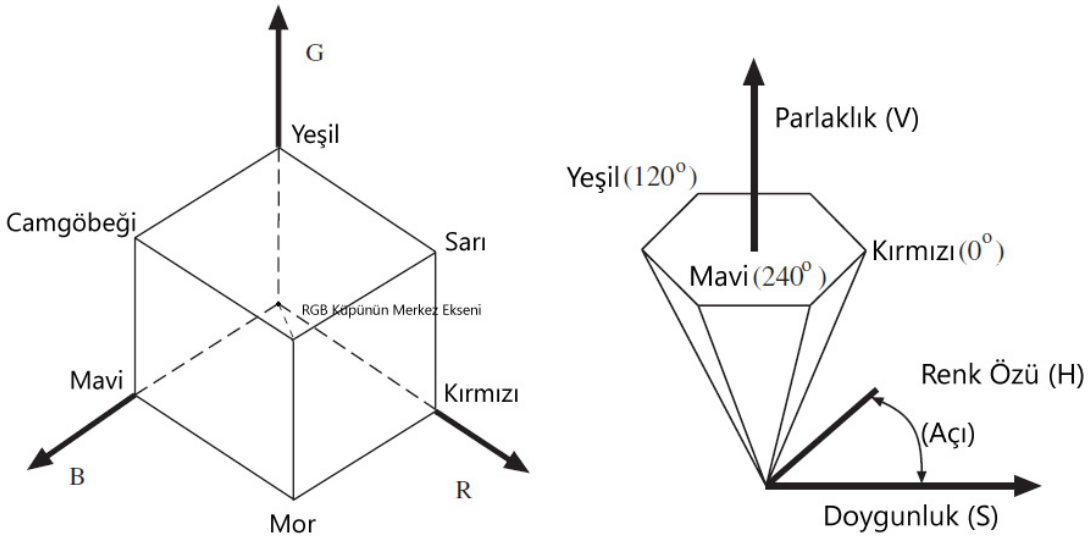
görüntü içerisinde sağa sola ya da yukarı aşağı gibi yer değiştirmesi durumunda mevcut yarıçap aralığı, eşik değeri ve daire merkezlerinin birbirinden minimum uzaklığı gibi parametrelerin güncellenmesi gerekmektedir. Parametrelere göre bir sürü yanlış daire de bulunabilmektedir. Şekil 3.18.'de bu daireler açıkça görülmektedir. Çalışma süresince genelde kullanılan 320x240 çözünürlüğünde birkaç pikselin bile önemli olduğu tek bakış açısı derinlik kestirimi yönteminde, böyle bir durum kabul edilemez bir durumdur. O yüzden en son olarak HSV renk uzayında filtreleme denenmiştir.



Şekil 3.18.: Dairesel Hough Dönüşümü İle Bulunan Daireler ve Kullanıcı Arayüzü

3.5.3. HSV Renk Uzayında Filtreleme

HSV renk uzayı, RGB renk uzayının doğrusal olmayan bir dönüşüme tabi tutulmuş halidir. RGB küpünün merkez ekseninden bakarak, HSV renk uzayı elde edilir [15]. Merkez ekseni, $(0, 0, 0)$ koordinatını birim küp üzerindeki $(1, 1, 1)$ koordinatına birleştiren doğrudur. RGB ve HSV renk uzayları Şekil 3.19.'da gösterilmiştir.



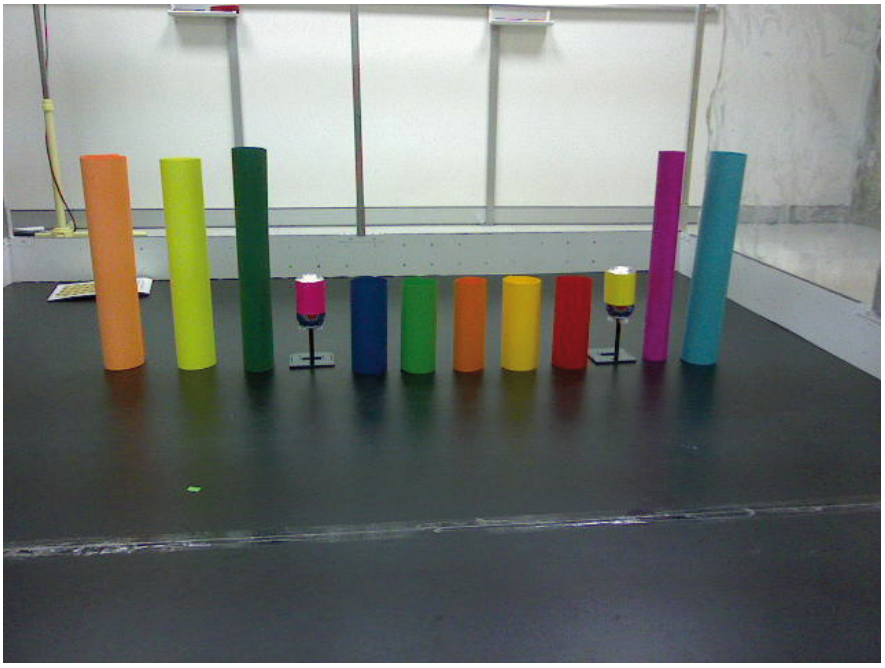
Şekil 3.19.: RGB ve HSV Renk Uzaylarının Gösterimi (Şekil [15] kaynağından alıntıdır.)

HSV renk uzayında çalışırken istenilen nesne için gündüz belirlenen renk aralığı ile gün boyunca rahat çalışılabildiği görülmüştür. Ortam ışığından daha az etkilenen HSV, karmaşık arka planlarla çalışırken de gürbüz sonuçlar vermiştir. Nesnenin filtrelenmesinde yapılan işlemler aşağıda anlatılmıştır.

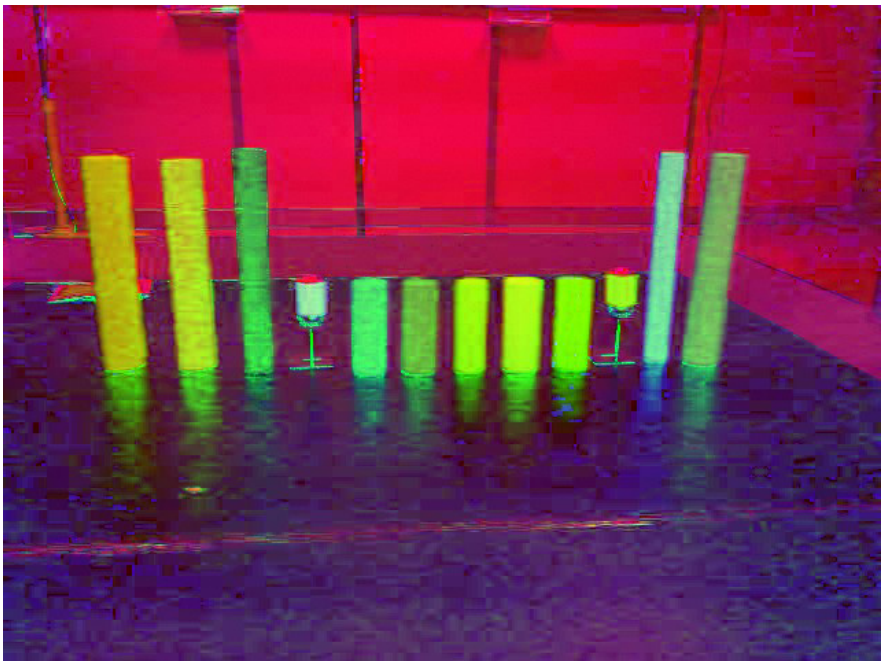
Şekil 3.20.'de çalışmada kullanılabilecek renkler toplu halde tek bir görüntüde görülmektedir. Kullanılan yöntemi anlatmak için örnek olarak kırmızı nesne seçilmiştir.

Şekil 3.21.'de görüntü HSV renk uzayına çevrilmiştir. OpenCV yazılan bir kod ile kırmızı nesne seçilmiş ve H, S ve V değerleri için alt ve üst sınırlar belirlenmiştir.

Şekil 3.22.'de HSV renk aralığı ile filtrelenmiş görüntü görülmektedir. Asıl nesnenin dışında görüntüde bulunan küçük piksel öbeklerinden oluşan gürültüler, anlatım sırasında işlemleri kolay gösterebilmek için HSV renk aralığının ideal olandan daha geniş tutulması sonucu oluşmuşlardır.

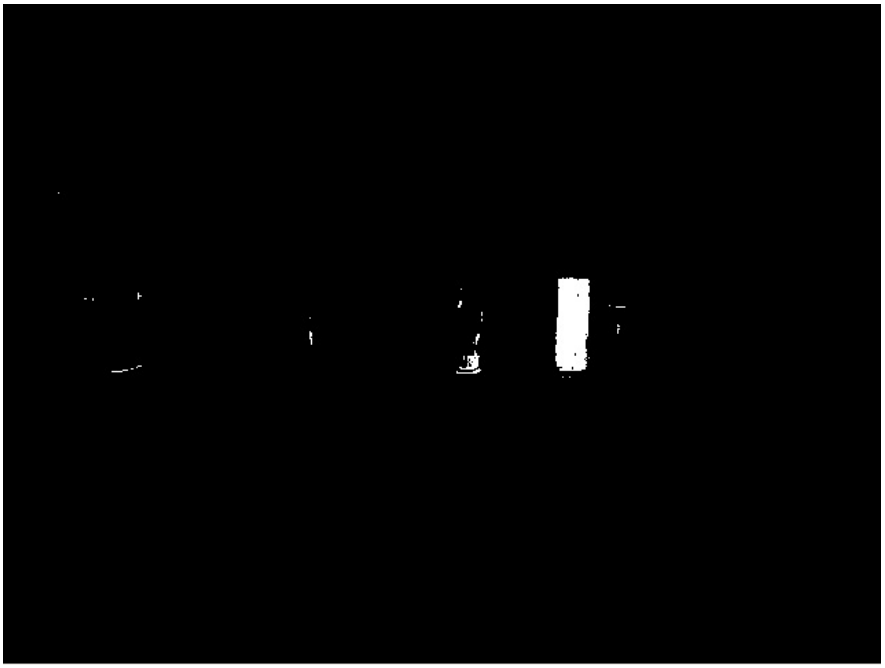


Şekil 3.20.: HSV Renk Uzayında Filtrelenecek Görüntü

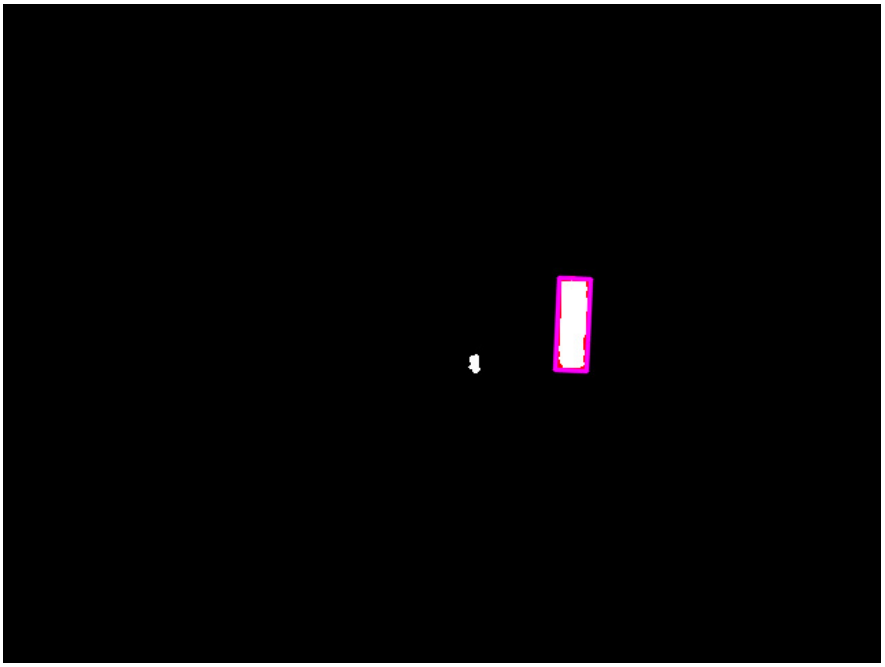


Şekil 3.21.: RGB'den HSV'ye Çevrilmiş Görüntü

Şekil 3.22.'de meydana gelen gürültülerden kurtulmak için medyan filtresi kullanılmıştır. Medyan filtresi burada tuz biber gürültüsü olarak bilinen piksel gürültülerinden kurtulmak için kullanılan bir filtredir. Medyan filtresi uygulanmasına



Şekil 3.22.: Filtrelenmiş Görüntü



Şekil 3.23.: Medyan Filtreden Geçirilmiş Görüntünün Son Hali

rağmen Şekil 3.23.'de asıl nesne ile beraber küçük bir öbek de kalmıştır. OpenCV içindeki fonksiyonları kullanarak görüntü içindeki bütün piksel öbeklerinin dış çizgilerini ortaya çıkarak piksellerin koordinatları bulunabilmektedir. Şekil 3.23.'de

kırmızı çizgi ile görülen bu dış çizgidir. Ancak dış çizgiler bulunduktan sonra bu çizgilerin sınırladıkları alanı içine alacak en küçük dikdörtgeni yerleştirecek bir OpenCV fonksiyonu en son aşamada kullanılmıştır. Bu didörtgenin uzun kenarı istenilen nesnenin piksel cinsinden yüksekliğini bize vermektedir. Şekil 3.23.'de istenilen nesne dışındaki diğer küçük öbek ise bu öbeğe ait dikdörtgenin alanı belirli bir değerden küçük olduğu için elenmiştir. Yükseklik olarak bulunan piksel cinsinden uzun kenar (2.13) eşitliğindeki ($\bar{h} + \tilde{h}$) toplamına denk gelmektedir. Ayrıca (2.17) eşitliğindeki ($\tilde{x} + \bar{x} - x_0$) kısmı ise görüntü işleme sonucunda yeri belirlenen nesnenin piksel cinsinden orta noktası ile kameranın kalibrasyonundan çıkan orta noktasının farkına denk gelmektedir. Sonraki bölümde, tek bakış açısı derinlik kestirimi yönteminin robot denetiminde nasıl kullanıldığı anlatılmıştır.

BÖLÜM 4

4. TEK BAKIŞ AÇISI DERİNLİK KESTİRİMİ YÖNTEMİ İLE DİZİLİM DENETİMİ

Önceki kısımlarda tek bakış açısı derinlik kestirimi yaklaşımının bir robot erkini için daha önceden büyüklüğü bilinen bir cisme olan göreceli mesafesinin bulunmasında nasıl kullanılacağı anlatılmıştı. Bu kısımda ise bu yöntem ile 2 boyutlu bir düzlem üzerinde bu robot erkinlerinden oluşan bir robot sürüsünün [22] çalışmasındaki yapıyı kullanarak önceden belirlenen bir ilk dizilim ile hareket sırasında dizilimi bozmadan, bir son konuma nasıl hareket edeceği anlatılmıştır. Fakat gerçek robotlar üzerine uygulama olmamasının yanı sıra (ki [8] çalışmasında sadece Matlab benzetimlerine yer verilmiştir) bu çalışmada [8] çalışmasından başka önemli farklar da vardır. [8] çalışmasında erkin modeli (aşağıda anlatılacağı gibi) tüm yönlü iken bu çalışmada kullanılan Khepera 3 robotları hız kısıtlı robotlardır. Bu durum önemli zorluklar getirmektedir. Robot erkinlerinin denetimine geçmeden önce erkinlerin modellenmesi ve dizilim kavramlarından bahsedilecektir. Burada [8]'daki çalışma takip edilecektir.

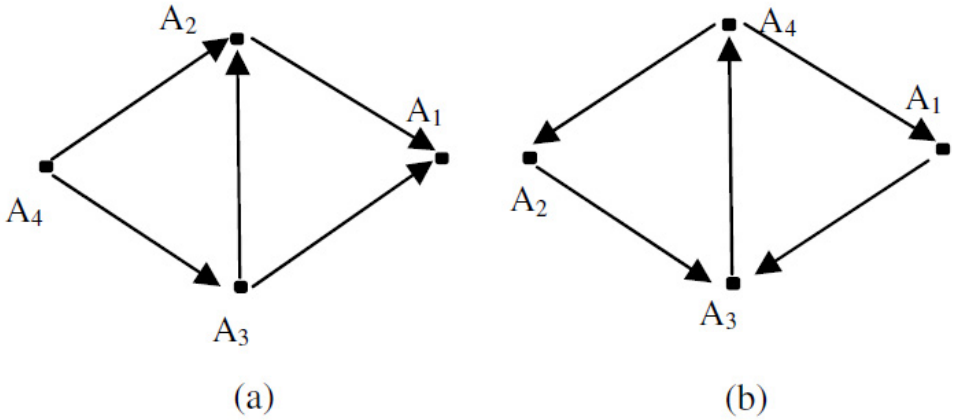
4.1. Erkin ve Sürü Modellenmesi

$M \geq 3$ olmak üzere $A_1, A_2, ..., A_M$ erkinlerinden oluşan bir robot sürüsünü ele alalım. Bu erkinleri de tüm yönlü nokta erkinler olarak yani büyüklüğü (kapladığı alan) sıfır olan ve rastgele her yönde hareket edebilen erkinler olarak düşünelim. Her A_i erkininin hız kinematiğini şu şekilde ifade edebiliriz :

$$\dot{p}_i(t) = v_i(t) \tag{4.1}$$

Burada $p_i(t) = (x_i(t), y_i(t)) \in \mathbb{R}^2$, A_i erkininin konumunu ifade ederken, $v_i(t) = (v_{xi}(t), v_{yi}(t)) \in \mathbb{R}^2$, A_i erkininin hızını ifade etmektedir. Erkinlerden her birinin v_i hızı sınırlandırılmıştır, yani maksimum bir $\bar{v} > 0$ için $\|v_i(t)\| < \bar{v}, \forall t$ olarak tanımlanmıştır.

Belirli bir dizilimi oluşturmak için çok erkinli bir robot sürüsü için sürünün yönlü çizgesi denen $G_F = (V_F, E_F)$ kavramı kullanılmıştır. $G_F = (V_F, E_F)$'ye ayrıca sürünün yada dizilimin temel teşkil eden yönlü çizgesi de denmektedir.



Şekil 4.1.: Süreğen bir dizilimin temel teşkil eden yönlü çizgesi, (a) Lider takipçi yapı, (b) Üç eş lider yapısı (Şekil [6] kaynağından alıntıdır.)

Şekil 4.1.'de bahsedilen temel teşkil eden yönlü çizgeye örnekler gösterilmiştir. Yönlü temel teşkil eden çizge kendi içinde bir köşe kümesi olan V_F ile bir kenar kümesi olan E_F'den oluşmaktadır. Köşeler kümesindeki her köşeyi F dizilimindeki $A_i, i \in V_F$, erkinleri oluşturmaktadır. Kenar kümesindeki kenarları ise (A_i, A_j) erkinleri arasındaki d_{ij} mesafesini göstermek için kullanılan $(\overrightarrow{i,j})$ vektörleri ifade etmektedir. d_{ij} ise A_i erkininin A_j erkininden göreceli olarak bulunması gereken mesafe koşuludur. d_{ij} ifadesini biraz daha açarsak, bu ifade A_i, A_j erkinini takip eder yada A_i, A_j erkininin takipçisidir olarak yorumlanmıştır. Sürünün erkinlerinden oluşan küme, $S = \{A_1, A_2, ..., A_m\}$ ve mesafe kümesi de $D_F = \{d_{ij} | (\overrightarrow{i,j}) \in E_F\}$ olarak ifade edilmiştir. Bu üç kümenin birleşimi de $F = \{S, G_F, D_F\}$ dizilimini ifade etmek için kullanılmıştır.

4.2. Süreğen Dizilimler ve Yapışık Hareket

$F = \{S, G_F, D_F\}$ diziliminde her (A_i, A_j) çifti arasındaki mesafe korunuyorsa bu yapıya *esnemez* denir. Esnemez yapıdaki bir G_F içinde her $(\overrightarrow{i,j})$ kenarı sabittir. Eğer F dizilimindeki her A_i erkini, F dizilimindeki diğer bütün erkinler gerekli kısıtlamaları sağlamaya çalışırken, kendi kısıtlamalarını sağlıyorsa bu yapıya da *kısıtlı süreğen* yapı denir. Eğer bir dizilim hem esnemez, hem de kısıtlı süreğen ise buna da *süreğen* yapı denir [22], [23]. Süreğen yapıdaki bir dizilimde tek bir kenarın dahi çıkarılması süreğen yapıyı bozuyorsa, bozulmamış en son haline *minimum süreğen* denir [23].

Bu çalışmada lider takipçi dizilim kullanılmıştır. Alternatif olarak kullanılabilecek olan yapılardan biri olan üç eş lider diziliminin kullanılmamasının sebeplerinden biri

robotlarda kullanılan kameranın görüş açısının göreceli olarak düşük olmasıdır. Şekil 4.1.'e bakıldığında üç eş lider yapısında A_4'ün diğer 2 robotu aynı anda görebilmesi için görüş açısının en az 120 derece olması gerektiği görülmektedir. Robotlarda kullanılan basit kameranın yatayda görüş açısı 60.5 derecedir. Diğer bir sebep de bütün eş liderlerin bazı zamanlarda aynı anda hem kendi hedeflerini hem de takip ettikleri robotları aynı anda görememeleridir. Örneğin A_1 erkini ileriye doğru hareket ederken hem hareket halinde bulunan hedefi hem de arkasında bulunan A_3 erkinini görmek zorundadır. Benzer sıkıntılar lider-takipçi yapısında yoktur. Bu yüzden de bu yapının mevcut duruma uygulanması daha kolaydır.

4.3. Problem Tanımı

$S = \{A_1, A_2, ..., A_m\}$,$M \geq 3$ robot erkin sürüsünü ve istenen 2 boyutlu minimum süreğen bir $F = \{S, G_F, D_F\}$ dizilimini ele alalım. Lider takipçi yapıdaki F diziliminin lideri Şekil 4.1.'deki A_1, birinci takipçi de Şekil 4.1.'deki A_2 olsun. Her robot (4.1) eşitliğindeki kinematiğe uygun şekilde hareket eden, üzerine tek bir kamera takılmış, ne kendi ne de diğer robotların konum bilgilerine dışardan herhangi bir şekilde erişemeyen ve birbirleri arasında haberleşme olmayan birer robot olsun. Her bir erkinin denetleyicisini tasarlarken amaç, her bir A_i erkininin önceden belirlenen bir $p_{i0}, i \in \{1, 2, ..., M\}$ ilk konumundan $p_{if}, i \in \{1, 2, ..., M\}$ son konumuna gidecek şekilde hazırlanan $F = \{S, G_F, D_F\}$ diziliminde kendi v_i hız sinyallerini üreterek yapışık hareket ettirebilmek olsun. Yapışık hareketten kasıt, D_F mesafe kısıtlarına uymak ve hareket ederken dizilimin şeklini bozmadan ilerleyebilmektir. Her bir $A_i, \{1, 2, ..., M\}$ erkini için üretilen hız sinyali olan v_i sürekli bir sinyal olmalı ve üst sınır $\bar{v} > 0$ için $\|v_i(t)\| \leq \bar{v}, \forall t$ koşulunu sağlıyor olmalıdır.

4.4. Dağıtık Denetim Şeması

Her bir robot erkini için tasarlanan denetleyici bu kısımda anlatılmıştır. Denetleyici tasarlanırken göz önüne alınan öncelikler sırasıyla şunlardır :

1. Dizilimin mevcut şeklini korumak.
2. İstenilen son konuma doğru hareket etmek.

Robotların denetleyicileri dağıtık yapıda olmalı ve her erkin için kendi referans koordinat sistemine göre göreceli olmalıdır. Bir başka deyişle her robot sadece kendi ölçtüğü yerel bilgilere dayanarak ve kendi iç denetleyicisi ile hareket etmelidir. Ayrıca

bütünsel yapı da korunmalıdır. Denetleyicilerin formüllerinde şu şekilde bir gösterim biçimi kullanılmıştır :

- $p_k^{(i)}(t)$: t anında A_i erkininin koordinat sistemine göre A_k erkininin koordinatları.
- $p_{if}^{(i)}(t)$: t anında A_i erkininin koordinat sistemine göre A_i erkininin ulaşması gereken hedef noktasının koordinatları.
- $p_i^{(i)}(t) = (0,0)$: Her t anı için her bir erkinin kendi koordinat sistemine göre koordinatları orijini gösterir.

Bu tez çalışmasında yukarıda da belirtildiği gibi Şekil 4.1.(a)'da görülen lider-takipçi yapısı kullanılacaktır. Bu yapıda bir lider, bir birinci takipçi ve diğerleri de sıradan takipçi olmak üzere üç farklı tipte erkin vardır. Aşağıda bunların denetleyicileri anlatılacaktır.

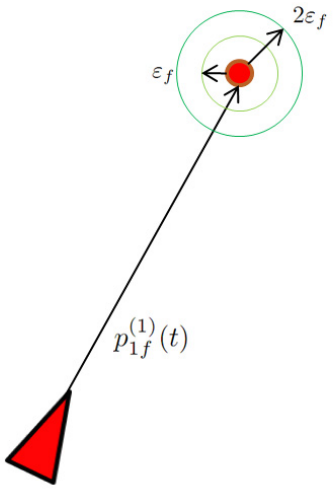
4.4.1. Liderin Denetimi

Diğer erkinelere göre denetleyici daha basit olan lider A_1, bütün hızını hedefi olan p_{1f} noktasına ulaşmak için kullanır. Kendi referans koordinat sistemine göre konumu her zaman $(0,0)$ olan lider A_1, her t anında $p_{1f}^{(1)}(t)$ yönünde tek parça bir hız bileşeni içerir ve şu şekilde ifade edilir :

$$v_1(t) = \sigma_l \bar{v} \bar{\beta}_1(t) p_{1f}^{(1)}(t) / \|p_{1f}^{(1)}(t)\| \tag{4.2}$$

$$\bar{\beta}_1(t) = \begin{cases} 0, & \|p_{1f}^{(1)}(t)\| < \varepsilon_f \\ \frac{\|p_{1f}^{(1)}(t)\| - \varepsilon_f}{\varepsilon_f}, & \varepsilon_f \leq \|p_{1f}^{(1)}(t)\| < 2\varepsilon_f \\ 1, & \|p_{1f}^{(1)}(t)\| \geq 2\varepsilon_f \end{cases} \tag{4.3}$$

Şekil 4.2.'de görüldüğü üzere $\bar{\beta}_1(t)$ katsayısı, liderin hedef noktasına geldiğinde olabilecek çatırdamaları engellemek için konulmuştur. Hedef noktasına $2\varepsilon_f$ mesafesine kadar maksimum hızla yaklaşan lider, $\varepsilon_f \leq \|p_{1f}^{(1)}(t)\| < 2\varepsilon_f$ aralığında yavaşça hızını kaybeder ve hedefe ε_f yarıçapında dairesel bir alana girdiği anda da hızı sıfırlanır ve hedef sabit durduğu sürece de sabit durur. $\sigma_l < 1$ ise arkadan gelen



Şekil 4.2.: Lider gösterimi

robotların yetişebilmeleri ve liderin çok hızlı gitmesini engellemek için konulmuş bir katsayıdır.

4.4.2. Birinci Takipçinin Denetimi

Lider A_1 robotunu takip eden birinci takipçi A_2, denetimi en zor olan robottur. Üzerindeki kameranın görüş açısının sınırlı olması ve de liderin hedef ile birinci takipçi arasına girdiği sorunlu olabilecek noktalardan A_1 ve kendi hedefi olan p_{2f} noktasını bir arada görene kadar kaçınmak üzere tasarlanan denetleyicisi 3 hız parçasından oluşmaktadır. [22] ve [23]'dekine benzer şekilde fakat göreceli referans koordinat sistemine göre tasarlanan birinci takipçi A_2'nin denetleyicisi şu şekildedir :

$$v_2^{(2)}(t) = \beta_2(t)v_{21}(t) + \sqrt{1-\beta_2^2(t)}v_{22}(t) + v_1^{(2)}(t) \tag{4.4}$$

(4.4) eşitliğinde görüldüğü gibi birinci takipçinin hız vektörü 3 hız vektörünün toplamıdır. Bu hız parçalarından ilki (4.7) eşitliğinde gösterilmiştir. Bu hız parçası birinci takipçinin, lideri takip etmesini sağlayan ve lider yönünde olan hız parçasıdır. Burada,

$$\beta_2(t) = \begin{cases} 0, & |\bar{\delta}_{12}(t)| < \varepsilon_k \\ \frac{|\bar{\delta}_{12}(t)|-\varepsilon_k}{\varepsilon_k}, & \varepsilon_k \leq |\bar{\delta}_{12}(t)| < 2\varepsilon_k \\ 1, & |\bar{\delta}_{12}(t)| \geq 2\varepsilon_k \end{cases} \tag{4.5}$$

olarak hesaplanır ve,

$$\bar{\delta}_{12}(t) = \|p_1^{(2)}(t)\|^2 - d_{21}^2 \tag{4.6}$$

$$v_{21}(t) = \bar{v} sgn(\bar{\delta}_{12}(t)) p_1^{(2)}(t) / \|p_1^{(2)}(t)\| \tag{4.7}$$

olarak tanımlanmıştır.

Burada $\bar{\delta}_{12}(t)$ birinci takipçinin lidere yaklaşması gereken mesafeye göre büyüklük ve işaret olarak (4.7)'e katılır. $\bar{\delta}_{12}(t)$ değeri pozitifse yaklaştıracak şekilde, negatifse uzaklaştıracak şekilde etki yapar. (4.5) eşitliğindeki $\beta_2(t)$ ise hız bileşenlerini paylaştırır. Eğer birinci takipçi, lidere çok uzaksa $\beta_2(t)$'in değeri 1 olur ve $v_{21}(t)$ hız vektörü olarak katılır, $v_{22}(t)$ ise katılmaz. Eğer birinci takipçi, lidere olması gereken mesafe kadar yaklaşmıssa $\beta_2(t)$'in değeri 0 olur ve $v_{22}(t)$ hız vektörü olarak katılır, $v_{21}(t)$ ise katılmaz. İkinci hız parçası olan $v_{22}(t)$, birinci takipçi lidere yeterince yaklaştığında, birinci takipçi-lider yönüne dik olaraktan, lideri ve kendi hedefini aynı anda görecek şekilde döndürmekle yükümlüdür. Bu hız parçasının hesaplanması ve bileşenleri aşağıda verilmiştir:

$$v_{22}(t) = \begin{cases} \sigma_f \bar{v} \delta_{12}^{\perp}(t), & |\varphi_{2f}^1| > FOV \\ \sigma_f \bar{v} \bar{\beta}_2(t) sgn\left(p_{2f}^{(2)T}(t) \delta_{12}^{\perp}(t)\right) \delta_{12}^{\perp}(t), & |\varphi_{2f}^1| \leq FOV \end{cases} \tag{4.8}$$

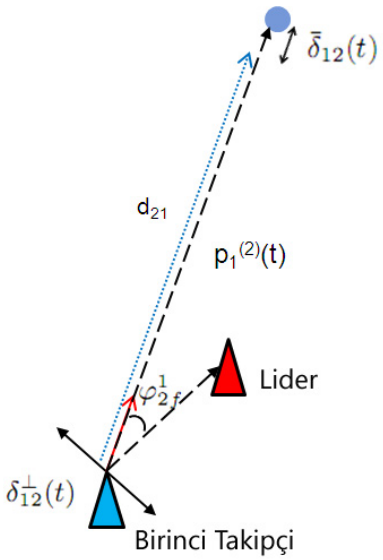
$$sgn(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases} \tag{4.9}$$

$$\delta_{12}^{\perp}(t) = \left(-p_{1y}^{(2)}(t), p_{1x}^{(2)}(t)\right) / \|p_1^{(2)}(t)\| \tag{4.10}$$

$$\bar{\beta}_2 = \begin{cases} 0, & \|p_{2f}^{(2)}(t)\| < \varepsilon_f \\ \frac{\|p_{2f}^{(2)}(t)\| - \varepsilon_f}{\varepsilon_f}, & \varepsilon_f \leq \|p_{2f}^{(2)}(t)\| < 2\varepsilon_f \\ 1, & \|p_{2f}^{(2)}(t)\| \geq 2\varepsilon_f \end{cases} \tag{4.11}$$

(4.8) eşitliğinde geçen φ_{2f}^1 ise birinci takipçi-lider yönü ile birinci takipçi-hedef yönü arasındaki açıyı gösterir. Eğer bu açı görüş alanı açısından fazla ise sürekli birinci takipçi-lider yönüne dik olarak robotu sola doğru döndürür. Eğer φ_{2f}^1 açısı görüş alanı açısından küçük ise $v_{22}(t)$ robotu hedefine gerektiği kadar yaklaşması için döndürür ve hedefe belirlenen mesafeye gelindiğinde çatırdamaları engelleyecek şekilde durur. $0 < \sigma_f < 1$ katsayısı önceden belirlenen bir sabit olup, A_2 erkininin hedefinden hızla uzaklaşmasını engellemek ve de takipçilerin kendisini takip edebilmesini sağlamak

için konmuştur.



Şekil 4.3.: Birinci Takipçinin Gösterimi

Şekil 4.3.'de birinci takipçinin lidere göre konumu ve açılar gösterilmiştir. Üçüncü hız parçası olan $v_1^{(2)}(t)$ ise liderin hızını taklit etmek için eklenmiştir. Yani, lider bir yönde aniden belirli bir hızla harekete geçerse birinci takipçi de liderin hızını görüp bunu kendi hız vektörüne eklerse ani hız değişimleri için iki robota da aynı hız vektörü eklendiğinden lider, birinci takipçiye göre durağan gibi olur.

Fakat $v_1^{(2)}(t)$ hız parçasını pratikte robot üzerine kodlamak kolay değildir ve sorun yaratabilmektedir. Bu amaçla ;

- Servo motorların hassasiyeti tekrar kontrol edilmiştir.
- Kameranın kalibrasyon matrisi tekrar kontrol edilmiştir.
- Daha hassas ölçüm alabilmek için maksimum hız azaltılmıştır.
- Hassasiyeti artırabilmek için normalde çalışılan 320x240 çözünürlüğünden 640x480 çözünürlüğüne çıkılmıştır.

Yine de sorun tatmin edici biçimde çözülememiştir çünkü hata bakılan kısımlardan kaynaklanmamıştır. Durağan halde bile lideri hareket halinde gören birinci takipçi hareket etmemesi gerekirken anormal hızlarda lidere yaklaşmıştır. Bunun sebebi daha sonradan kameranın uzaklığını ölçtüğü nesnenin büyüklüğünü bulurken sürekli

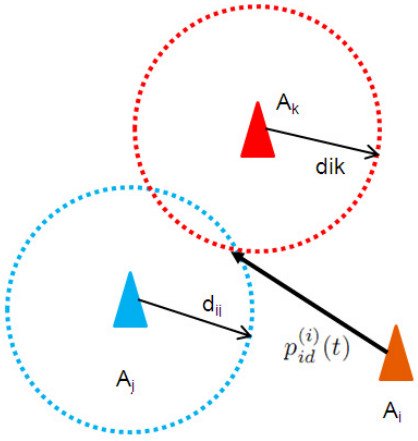
aynı olarak görmemesi olarak anlaşılmıştır. Diğer bir deyişle 50 cm uzaklığındaki nesneyi ideal olarak mesela 30 piksel olarak görmesi gerekirken ortamın ışığındaki küçük dalgalanmalardan dolayı 29-31 piksel aralığında gördüğü tespit edilmiştir. 1 piksellik hatanın, robotun mesafe ölçümünde yaklaşık 1 cm lik bir değişime sebep olduğu ve duran cismi 1 cm/s hızla ileri yada geri gittiğini gördüğü anlaşılmıştır. 1-2 cm/s maksimum hızlarda yapılan bu dizilim denetimi çalışmasında bu hata payı çok büyük olduğu ve robotun diğer hız vektörlerine baskın gelmesi sebebiyle bu hız parçası robotun denetim koduna eklenmemiştir.

4.4.3. Sıradan Takipçilerin Denetimi

Sıradan takipçilerin denetiminde ise sıradan takipçi-1 olan A_3, lider A_1 ve birinci takipçi A_2'ye olan mesafesini aynı tutmaya, sıradan takipçi-2 olan A_4 ise birinci takipçi A_2 ve sıradan takipçi-1 A_3 ile olan mesafesini aynı tutmaya çalışmaktadır. Bu durum Şekil 4.1.(a)'daki oklarla ifade edilmiştir. Sürüde daha fazla robot olması durumunda her sıradan takipçi robot önünde bulunan iki robota mesafelerini koruyacak şekilde ve süreğenlik özellikleri korunacak şekilde ilişkiler belirlenebilir. Sıradan takipçiler için denetim kuralı ise şöyledir:

$$v_i^{(i)}(t) = \bar{v}\beta_i(t)p_{id}^{(i)}(t)/\|p_{id}^{(i)}(t)\| + \dot{p}_{id}^{(i)}(t) \tag{4.12}$$

$$\beta_i(t) = \begin{cases} 0, & \|p_{id}^{(i)}(t)\| < \varepsilon_k \\ \frac{\|p_{id}^{(i)}(t)\| - \varepsilon_k}{\varepsilon_k}, & \varepsilon_k \leq \|p_{id}^{(i)}(t)\| < 2\varepsilon_k \\ 1, & \|p_{id}^{(i)}(t)\| \geq 2\varepsilon_k \end{cases} \tag{4.13}$$



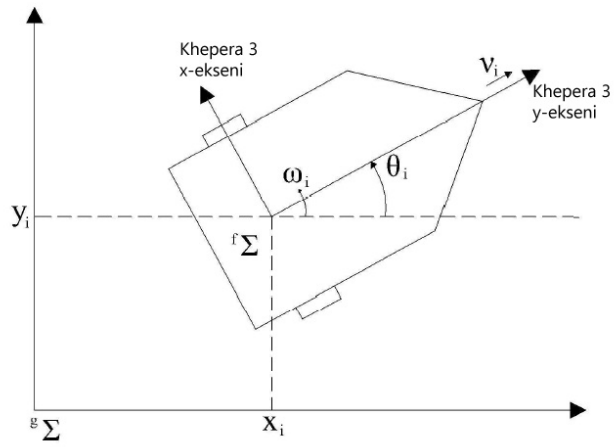
Şekil 4.4.: Takipçilerin Gösterimi

Burada $p_{id}^{(i)}(t)$ Şekil 4.4. dairelerin kesişim noktasına takipçiden uzanan vektörü temsil etmektedir. $\beta_i(t)$ katsayısı çatırdamaları engelleyerek durmak için burada da konulmuştur. Daha önceki lider ve birinci takipçide görülen hız kısıtlayıcı σ katsayısına burada rastlanmamaktadır. Birinci takipçide olduğu gibi burada da kesişim noktasının hızını taklit eden $\dot{p}_{id}^{(i)}(t)$ diğer hız bileşenine baskın geldiği ve doğru okunamadığı için pratikte kullanılmamıştır.

4.5. Robotla Denetimi ile İlgili Pratik Konular

4.5.1. Erkin Modellemesi

Bu tez çalışmasında kullanılan Khepera 3 robotlarda (4.1) eşitliğindeki hız bileşeni direkt olarak kullanılamaz. (4.1) eşitliğine göre noktasal olarak tasarlanan hız bileşenine göre robot herhangi bir t anında istediği yöne hareket edebilmektedir. Fakat tüm yönlü olmayan Khepera 3 gibi robotlarda, robot aniden dingil boyunca harekete geçemez. Bu yüzden girdi olarak verilecek hız bileşeni Khepera 3 robotuna uygun olmalıdır. Tüm yönlü olmayan bir erkinin modeli ve bu model üzerinde Khepera 3'ün eksenleri Şekil 4.5.'de görülmektedir.



Şekil 4.5.: Erkin Modeli ve Khepera 3 Eksenlerinin Gösterimi (Şekil [8] kaynağından alıntıdır.)

Khepera 3 robotlarının hareketlerini modelleyen tüm yönlü olmayan erkin modeli için

denetim kuralları aşağıdaki gibidir:

$$\dot{x}_i(t) = \bar{v}_i(t)cos(\theta_i(t)) \tag{4.14}$$

$$\dot{y}_i(t) = \bar{v}_i(t)sin(\theta_i(t)) \tag{4.15}$$

$$\dot{\theta}_i(t) = w_i(t) \tag{4.16}$$

Burada $x_i(t)$ ve $y_i(t)$ robotun t anındaki kartezyen koordinatları, $\theta_i(t)$ de robotun koordinat sistemine göre doğrultusunun x-ekseni ile yaptığı açıdır. $v_i(t)$, doğrusal hız, $w_i(t)$ de açısal hızdır. Khepera 3 ile gelen araç kitinde hız denetimi, robotun sağ ve sol tekerinin hızına girdi verilerek yapılmaktadır. Şekil 4.5.'de modeldeki gibi robota doğrusal ve açısal hız girdisinin verilmesi, Sürü Sistemleri Labı'nda yüksek lisansını tamamlamış Yunus Ataş tarafından Khepera araç kitinde değişiklik yapılması ile sağlanmıştır.

$\bar{v}_i(t) = \|v_i(t)\|$ olarak ifade edilebilir. Burada $v_i(t)$ her erkin için önceki bölümde anlatılan yöntemi kullanarak hesaplanmaktadır. Açısal hız,

$$w_i(t) = -\alpha(\theta_i(t) - \theta_{id}(t)) \tag{4.17}$$

olarak ifade edilir. Burada α oransal denetleyici katsayısıdır. Oransal denetim kullanılan açısal hız da,

$$\theta_{id}(t) = tan^{-1}\left(\frac{v_{yi}(t)}{v_{xi}(t)}\right) \tag{4.18}$$

istenilen robot açısını temsil etmektedir. Burada yine $v_{xi}(t)$ ve $v_{yi}(t)$ önceki bölümdeki (4.2),(4.4) ve (4.12) denklemleri ile hesaplanan $v_i(t)$ hız vektörünün x ve y eksenleri boyunca bileşenleridir. Bu şekilde önceki bölümde tüm yönlü robotlar için geliştirilmiş yöntemi kullanarak hesaplanan denetim girdilerinden Khepera 3 robotlarına uygun girdilere dönüştürülmektedir.

4.5.2. Robot Kamerasının Açı Denetimi

Kameranın hareketli olmasının temel sebebi robotun hareketinden yönünden bağımsız olarak robotun istediği açıdan görüntü alabilmesidir. Böylece mesela robot kamerası sol tarafa dönük veri alırken robot ileri doğru hareketini sürdürebilmektedir. Robotun kamerası kontrol edilirken şöyle bir yöntem izlenmiştir. $A_i(i = 1, 2, ..., N)$ erkininin

kamerasının açısal kontrolü;

$$\dot{\theta}_{ci} = -w_i - \alpha_c \varphi^i_{jk} \tag{4.19}$$

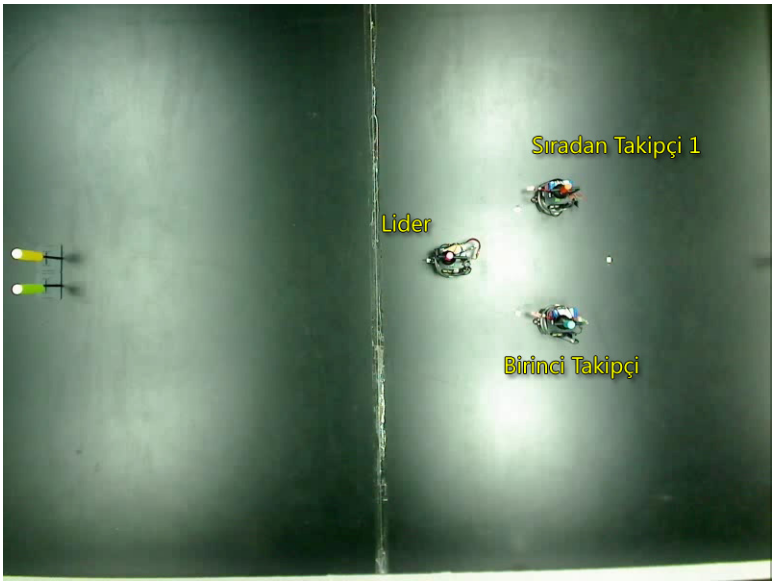
Burada w_i (4.17) denkleminde hesaplanan erkinin açısal hızıdır.(4.19) eşitliğinde $-w_i$ denmesinin sebebi ise robot belli bir yönde w_i açısal hızı ile dönerken kameranın da $-w_i$ açısal hızı ile bunu dengelemesidir. $\alpha_c > 0$ olan orantısal bir katsayıdır. Sıradan takipçiler için $\varphi^i_{jk} = \left(\frac{\phi^i_j + \phi^i_k}{2}\right)$ olarak ifade edilebilir. ϕ^i_j ve ϕ^i_k sırasıyla A_j ve A_k erkininin A_i erkinine göre yaptıkları açılardır. Lider için $\varphi^1_{jk} = \phi^1_f$'dir yani liderin doğrultusu ile hedefi arasındaki açıdır. Birinci takipçi içinse $\varphi^2_{jk} = \varphi^2_{1f}$'dir yani birinci takipçinin hedefi ve liderin birinci takipçiye göre olan açısıdır.

Teorik olarak kamera kontrolcüsü sorunsuz olsa da pratikte özellikle sıradan takipçilerde sorun çıkmaktadır. Bunun sebebleri, sıradan takipçilerin takip ettikleri hedeflerin hareket halindeki robotlar olması ve kamera açısının 60 derece olmasıdır. Yani sıradan takipçiler önlerindeki robotları görüntü alanlarının sağ ve sol sınırlarında görmektedirler. Herhangi bir robot görüntü alanından çıkarsa kamera mevcut açısına göre sağa ve sola 30-40 derecelik bir dönüş yapar. Görüntü alanından çıkan robotlar böylece sıradan takipçiler tarafından bulunurlar. Sıradan takipçi bulduğu robotların açılarına göre yine (4.19) eşitliğindeki kontrolü kullanırlar.

4.6. Robot Üzerindeki Uygulamalar ve Sonuçları

Robotlar üzerinde yapılan deneyler, Sürü Sistemleri Labı'nın 240 cm x 320 cm boyutlarındaki büyük arenasında yapılmıştır. Deneyin kaydı ise tavana takılan bir Logitech Quickcam 9000 Pro web kamerası ile yapılmıştır. Şekil 4.6.'da kameranın görebildiği kısım arenanın yaklaşık olarak tamamıdır.

Şekil 4.6.'da 3 robot ve 2 hedef görülmektedir. Robotlardan en soldaki lider, alttaki birinci takipçi ve üstteki de sıradan takipçi-1'dir. Görüntünün solunda sarı ve yeşil olarak görülen hedefler 13 cm yüksekliğinde, 3 cm çapındadır. Hedeflerin boyutu arenanın büyüklüğüne göre en uygun olacak şekilde seçilmiştir. Bu seçim yapılırken birinci takipçi kendi hedefine daha uzakta olduğu için bu mesafeden 320x240 çözünürlükte rahatlıkla görüntü işleme yapılabilecek kadar büyük fakat perspektif etkisinden dolayı hedefe yakınlaştıkça yanlış ölçümler almayacak kadar da küçük seçilmiştir. Lider, sarı (üstteki) hedefe, birinci takipçi de yeşil (alttaki) hedefe gidecek şekilde ayarlanmıştır. Robotların kendi üzerlerinde bulunan renkli silindirler ise 6 cm yükseliğinde 2 cm çapındadır. Robotlar için uygun ölçülerde



Şekil 4.6.: 3'lü Robot Dizilimi Tepe Kamerası

yapılan bu renk belirleyicileri 15-50 cm aralığında birbirlerini rahatlıkla görmelerine olanak sağlamıştır.

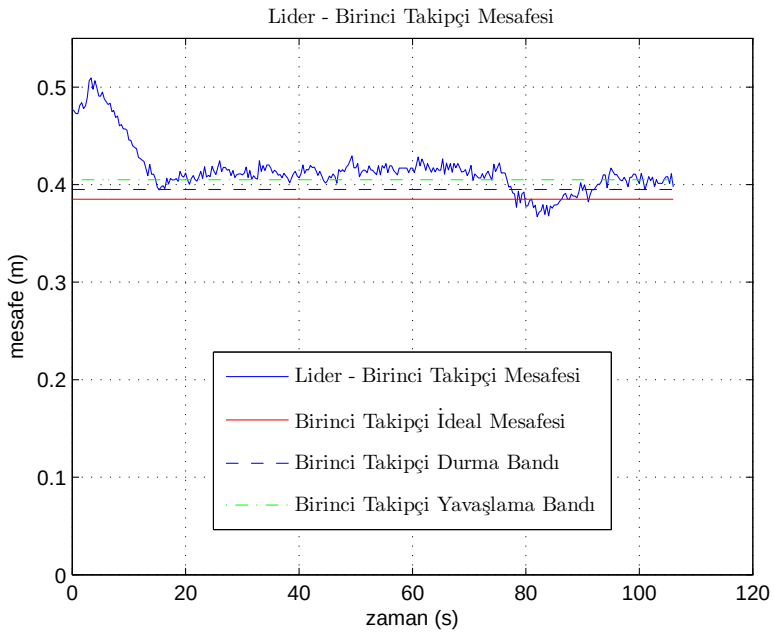
Deney sırasında lider için hız sınırlayıcı $\sigma_l = 0.75$, maksimum hız $\bar{v} = 0.015m/s$, durma payı $\varepsilon_f = 0.01$, açısal hız katsayısı $\sigma_{acisal} = 0.5$ ve liderin durma mesafesi $d_{lider-hedef} = 0.2m$ olarak seçilmiştir. Birinci takipçi içinse $\sigma_f = 0.6$, $\bar{v} = 0.0165m/s$, $\varepsilon_f = \varepsilon_k = 0.01$, $\sigma_{acisal} = 0.07$, $d_{B.Takipci-Lider} = 0.36m$ ve $d_{B.Takipci-Hedef} = 0.30m$ olarak seçilmiştir. Sıradan takipçinin katsayıları ise $\bar{v} = 0.017m/s$, $\varepsilon_k = 0.03$, $\sigma_{acisal} = 0.1$ ve $d_{S.Takipci1-B.Takipci} = d_{S.Takipci1-Lider} = 0.38m$ olarak girilmiştir.

Robotların yol boyunca hareketlerinde x, y ve θ bilgilerini robotun ilk konum ve açısına göre ölçen bir kör konumlandırma[1] sistemi vardır. Fakat, robotlar üzerindeki kör konumlandırma uzun süreli hareketlerde toplanarak büyüyen hataları da içinde barındırır. Kör konumlandırma güvenilir olmadığı için tepe kamerasından görüntü işleme yoluyla robotların çizdikleri yollar belirlenmiş ve birbirlerine olan mesafeleri hesaplanmıştır. Görüntü işleme için videodan 5 karede bir kare alınmış, daha sonra her robot için sabit ve en yüksek nokta olan renk belirleyici nesnelerin tepe noktalarının koordinatları Matlab ile alınmış ve kaydedilmiştir. Bu tepe noktalarının yerden yüksekliği 30 cm olup görüntü merkezinden uzaklaştıkça perspektif etkisi yüzünden görüntü merkezinden uzaklaşmalarının tersi yönünde kayma olmaktadır. x

[1] ing: odometry

ve y koordinatlarında 1 m mesafede 30 cm yüksekliğinde ne kadar kayma olduğuna bakılmış ve bu kayma miktarı hesaba katılarak robotlar arası mesafelerin grafikleri çıkarılmıştır.

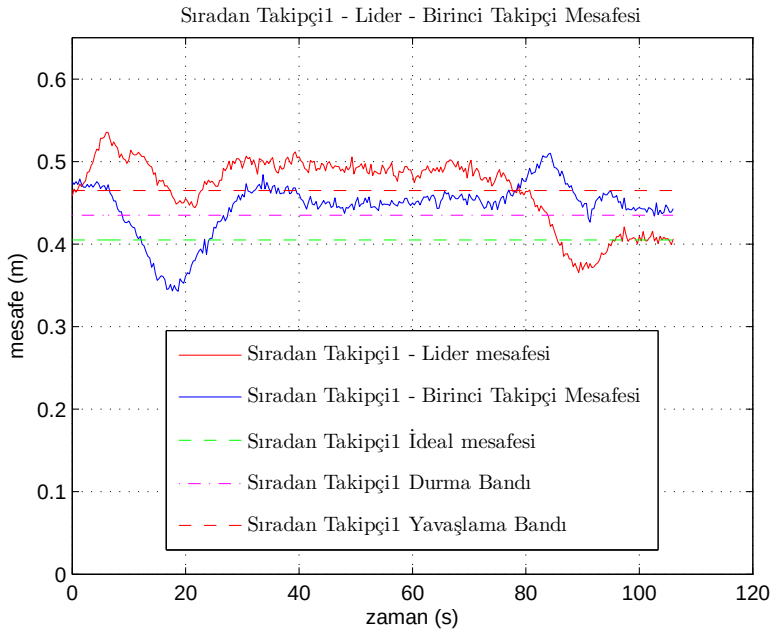
Şekil 4.7.'de lider ile birinci takipçi arasındaki mesafe ilk başta yaklaşık 45 cm'dir. Fakat robot kodlarınının ana bilgisayardaki terminalden sırayla çalıştırılması yüzünden birinci takipçi çalışmaya liderden birkaç saniye sonra başlamaktadır. Şekil 4.7.'deki ilk saniyelerde lider-birinci takipçi arasındaki mesafenin artması ve daha sonra birinci takipçinin liderden daha hızlı olması sebebiyle aradaki mesafeyi kapatıp istenilen mesafeye yakınsaması görülmektedir. Şekil 4.7.'de 80. saniye civarında lider



Şekil 4.7.: Birinci Takipçi - Lider Mesafesi

hedefine ulaşmış ve durmuş olduğundan birinci takipçi yavaşlamış bu yüzden istenen mesafeden birkaç cm içeri girmiştir. Daha sonra birinçi takipçi liderin etrafında kendi hedefini de görerek daire çizmiş ve hem kendi hedefine hem de lidere istenen mesafede durmuştur.

Şekil 4.8.'de de yine en son kodu çalıştırılan sıradan takipçi-1 liderden nispeten çok, birinci takipçiden az geride kalıyor ve 20. saniye civarında yaklaşarak istenen mesafeye yakınsıyor. Fakat 20. saniyeden sonra hem lidere, hem de birinci takipçiye göre sabit bir şekilde istenilen mesafelerden daha uzak mesafelerde önündeki iki robotu takip ediyor. Bunun sebebi sıradan takipçinin, liderden ve birinci takipçiden

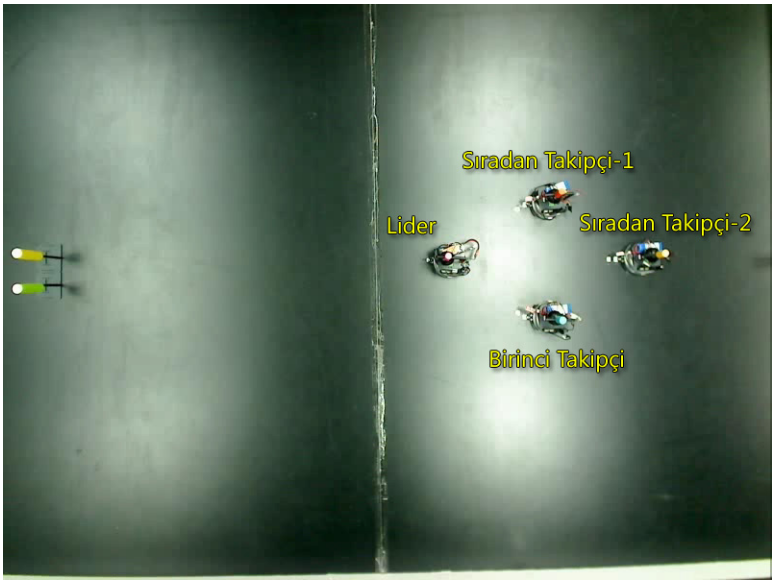


Şekil 4.8.: Sıradan Takipçi1 - Lider - Birinci Takipçi Mesafesi

farklı bir özelliğinin olmasıdır. Lider ve birinci takipçi, kendi hedeflerini rahatlıkla görebilmektedirler. Yani, lider kendi hedefini, birinci takipçi de hem lideri hem de kendi hedefini kameranın görüş açısı olan 60 derece içinde rahatlıkla görebilmektedir. Fakat sıradan takipçi-1 lider ve birinci takipçi ile eşkenar üçgen oluşturmak zorunda olduğundan sıradan takipçi-1'e göre lider ile birinci takipçi arasındaki açı yaklaşık 60 derece olmalıdır. Sıradan takipçi-1 önündeki robotları kendi görüş alanının sınırları içinde gördüğünden, hem kendisi hem de diğer robotlar hareket halinde olduğundan, robotlardan birini görüntüden kaybetmesi normal bir durumdur. Görüntüden çıkan robotun yerini bulabilmek için sıradan takipçi-1'in kamerası sağa ve sola 30-40 derece dönerek kaybolan robotu arar. Bu sırada zaman geçtiğinden sıradan takipçi-1 geride kalmakta ve diğer robotlarla arası bir miktar açılmaktadır. Şekil 4.8.'de 80. saniye civarında sıradan takipçi-1 ile lider arasındaki mesafe hızla düşerken, sıradan takipçi-1 ile birinci takipçi arasındaki mesafenin hızlıca arttığı görülmektedir. Bu durum, lider kendi hedefine ulaşıp durduğundan sıradan takipçi-1 lidere en yakın olacak şekilde durur ancak bu sırada birinci takipçi çoktan dönmeye başlamıştır ve sıradan takipçi-1 ile birinci takipçinin arası açılır. Khepera 3 tüm yönlü bir robot olmadığından birinci takipçi lider etrafında dönerken sıradan takipçi-1 hızlı bi şekilde yönünü değiştirmesi gerekmektedir. Yani, hem hızlıca sola dönmeye çalışmakta hem de öne doğru ilerlemeye çalışmaktadır. Bu işlem bir süre devam eder. Lider ile birinci takipçi hedeflerine ulaştığında sıradan takipçi-1 de önündeki robotlara istenilen

mesafeye yakınsar ve durur.

4 robotlu dizilim denetiminde ise deney ortamı ve ilk üç robot için kullanılan katsayılar aynıdır. Tek değişiklik 4. robot olan sıradan takipçi-2'dir. Sıradan takipçi-2 için kullanılan katsayılar ise $\bar{v} = 0.018m/s$, $\varepsilon_k = 0.035$, $\sigma_{acisal} = 0.13$ ve $d_{S.Takipi2-S.Takipi1} = d_{S.Takipi2-B.Takipi} = 0.43m$'dir. 4 robotlu dizilim Şekil 4.9.'da görülmektedir.

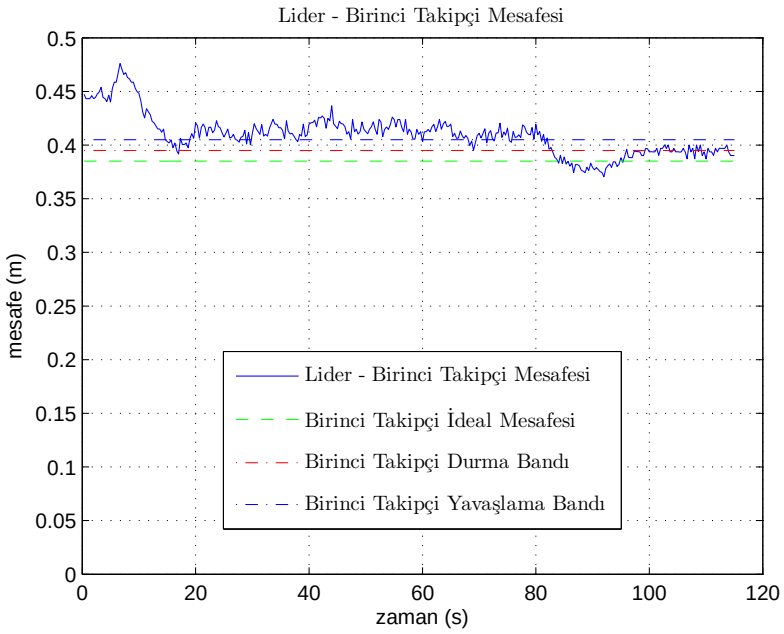


Şekil 4.9.: 4'lü Robot Dizilimi Tepe Kamerası

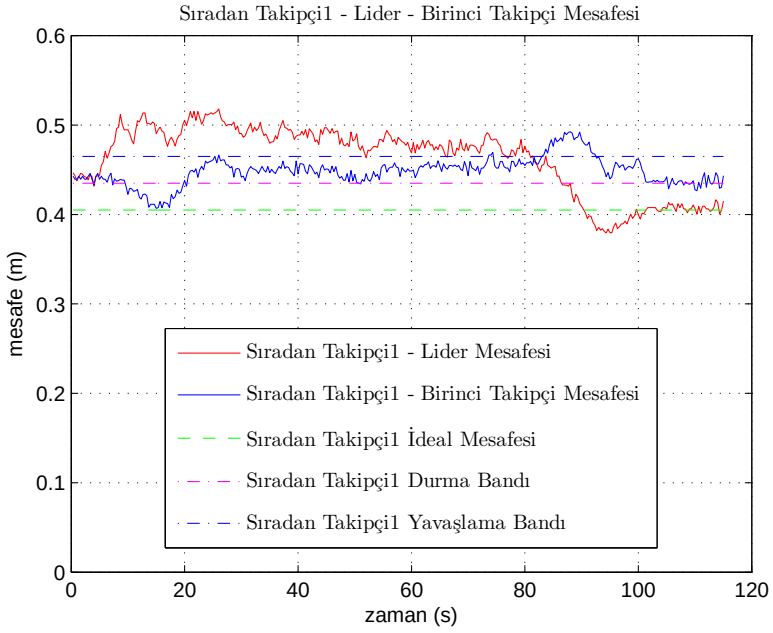
3 robotlu dizilimde olduğu gibi 4 robotlu dizilimde de birinci takipçi liderden yaklaşık 45 cm geride harekete başlıyor ve birinci takipçi ilklendirilene kadar geçen sürede lider ile birinci takipçi arasındaki mesafe açılıyor. Şekil 4.10.'da da görüldüğü gibi 20. saniye civarında lider birinci takipçiyi yakalamakta ve 80. saniye civarında lider durduğunda çevresinde daire çizerek kendi hedefine ulaşıp uygun mesafede duruyor.

Şekil 4.11.'de ise sıradan takipçi-1 ilk 20 saniye içerisinde birinci takipçiye yaklaştığı için bir miktar liderden uzaklaştığı görülmektedir. 20. saniyeden sonra ise 80. saniyeye kadar yakınsaması gereken mesafelere yaklaştığı görülmektedir. 80. saniyeden sonra sabit duran lidere ve dönmekte olan birinci takipçiye uygun mesafelerde yakınsamıştır. Sıradan takipçi-1'in birinci takipçi ve lidere durma bandı içinde yakınsamış olduğu Şekil 4.11.'de görülmektedir.

4 robotlu dizilimdeki en arkadaki robot olan sıradan takipçi-2 için $\varepsilon_k = 0.035$



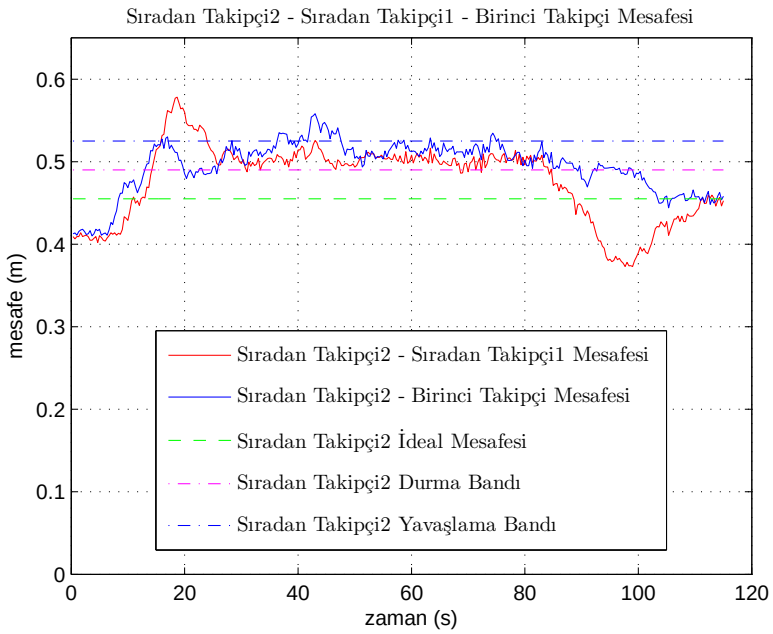
Şekil 4.10.: Lider - Birinci Takipçi Mesafesi



Şekil 4.11.: Sıradan Takipçi 1 - Lider - Birinci Takipçi Mesafesi

olduğundan yavaşlama, durma ve ideal mesafe bandları arasındaki mesafe diğer robotlar arası mesafelerden daha fazla olarak 3.5 cm'dir. Ayrıca ideal mesafesi, $d_{S.Takipci2-S.Takipci1} = d_{S.Takipci2-B.Takipci} = 0.43m$, de fazla olan sıradan takipçi-2

Şekil 4.12.'de görülmektedir. İlk 20 saniye içerisinde ilklendirmesi en son olan sıradan takipçi-2'nin birinci takipçi ve sıradan takipçi ile mesafesinin daha fazla açıldığı görülmektedir. Buna rağmen 20. saniyeden sonra Şekil 4.12.'de sıradan takipçi-2'nin yoluna yavaşlama ve durma bandı arasında devam ettiği görülmektedir. 80. saniyeden sonra sıradan takipçi-1 dönmeye başladığında sıradan takipçi-2 merkezi lider olan bir dairenin çevresinde dönmeye başladığından doğrusal ve açısal hızının deneyde kullanılan değerlerden daha fazla olması gerekmektedir. Fakat, Khepera 3 robotları, görüntü işlemedeki yavaşlıktan ve kamera bakış açısının gerekli minimum değerde olmasından dolayı yüksek hızlarda deney yapmaya uygun değildir. Bunlardan dolayı da 80. saniyeden sonra sıradan takipçi-2, sıradan takipçi-1 kadar hızlı tepki verememekte ve sıradan takipçi-2 sıradan takipçi-1'e oldukça yaklaşmaktadır. Fakat Şekil 4.12.'de de görüldüğü üzere deneyin sonunda sıradan takipçi-2 ideal mesafesine yakınsamıştır.



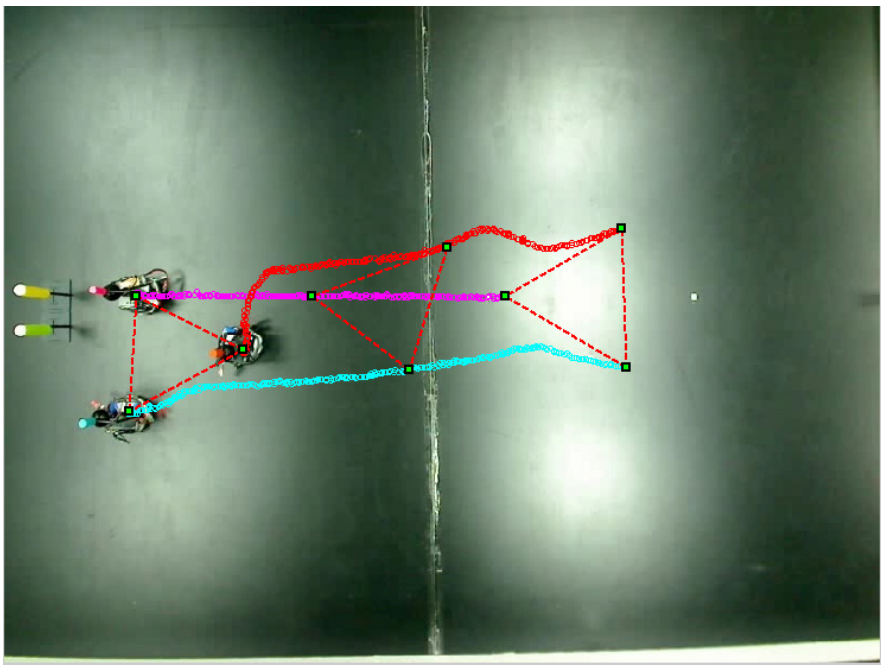
Şekil 4.12.: Sıradan Takipçi 2 - Sıradan Takipçi 1 - Birinci Takipçi Mesafesi

Dikkat edilirse Şekil 4.7., Şekil 4.8., Şekil 4.10., Şekil 4.11. ve Şekil 4.12.'de robotların ideal mesafeleri, katsayılar arasında belirtilen d mesafelerinden daha büyüktür. Bunun sebebi ise katsayılar arasında belirtilen mesafeler, robotların üzerindeki kameraların lensinden yani ön kısmından hedeflere olan mesafelerdir. Fakat robotların ideal mesafeleri ise renk belirleyici nesnelerden, aynı zamanda kameraların dönüş eksenlerinden hedeflere olan mesafelerdir. Kodlar içinde kullanılan mesafeler

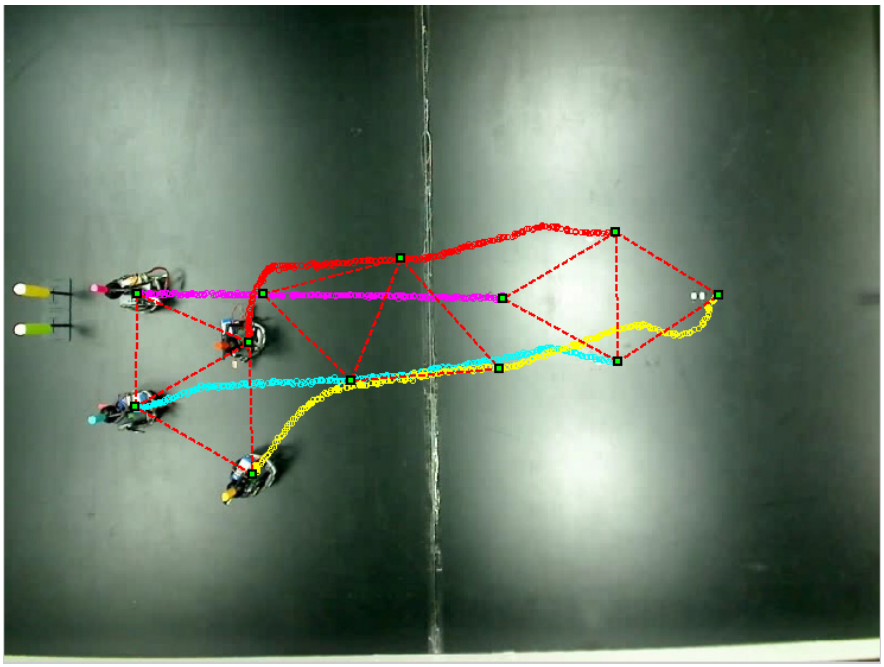
ile grafiklerdeki mesafeler arasındaki fark 2.5 cm'dir ve kameraların ön kısmı ile dönüş eksenleri arasındaki farktan ileri gelmektedir.

Bu 2.5 cm'lik farkın deneylerde önemli bir etkisi daha vardır. Farzedelim ki birinci takipçi lidere istenen mesafede, 40 cm'de, olsun. Sıradan takipçi-1 ise lider ve birinci takipçi arasındaki mesafeyi, 2.5 cm farktan dolayı yaklaşık 42.5 cm olarak görür. Sıradan takipçi-2 ise aynı sebepten birinci takipçi ve sıradan takipçi-1 arasındaki mesafeyi 45 cm olarak görür. Bu yüzden de dizilimde robotlar aslında kendi aralarında eşkenar üçgen değil de ikizkenar üçgen oluşturmaktadırlar.

3 ve 4 robotlu dizilim deneylerinde robotların izlediği yollar ve deneyin başında, ortasında ve sonunda dizilimin durumu Şekil 4.13. ve Şekil 4.14.'de görülmektedir.



Şekil 4.13.: 3 Robotlu Dizilim Deneyinde Robotların İzlediği Yollar



Şekil 4.14.: 4 Robotlu Dizilim Deneyinde Robotların İzlediği Yollar

BÖLÜM 5

5. GERİ BESLEME DOĞRUSALLAŞTIRMASI YÖNTEMİ İLE HAREKETLİ ROBOTLARDA HEDEF ETRAFINDA ÇEMBER TAKİBİ

Bu bölümde çoklu robotlar için düşünülen fakat şu an için tek bir robot üzerinde uygulanmış olan bir yöntem anlatılacaktır. Mesafe ölçümü daha önceki bölümlerde anlatılan ve kullanılan tek bakış açısı derinlik kestirimi yöntemidir. Burada amaç, hedef olarak belirlenen nesnenin etrafında robotun denetleyicinin kendisi tarafından parametrelerinin hesaplanması ve istenilen çember yörüngesine yakınsamasının sağlanmasıdır. Denetleyiciler iki farklı yöntemle tasarlanmışlardır. Bunlardan ilki olan TKTÇ (Tek Girdi Tek Çıktı içeren sistem) modeli kullanarak tasarlanmıştır. Bu denetleyicide doğrusal hız sabit açısal hız güncellenmiştir [24]. Aynı problemi çözmek için ÇGÇÇ (Çok Girdi Çok Çıktı içeren sistem) de mevcuttur. Bu yöntemde hem doğrusal hız hem de açısal hız güncellenmektedir. Fakat bu tez çalışmasında ÇGÇÇ yöntemi ele alınmamıştır. Anlatılan yöntemin özelliği çok az parametre ile istenilen davranışların elde edilmesidir. Sürü uygulamalarına geçildiğinde bu durum önemli avantajlar sağlayacağı düşünülmektedir. Bu bölümde [24]'deki çalışma takip edilecektir.

5.1. Matematiksel Model

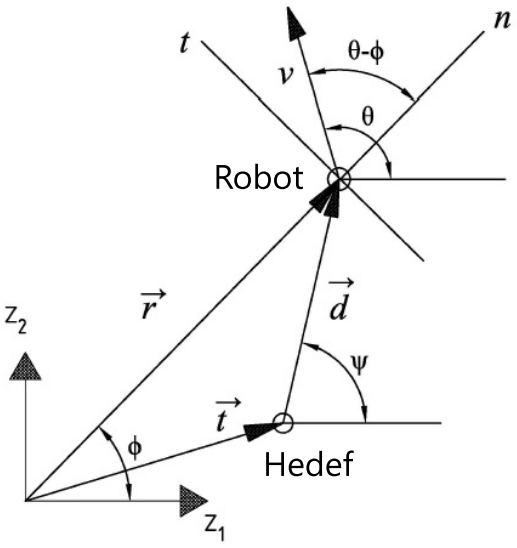
Bu çalışmadaki denetleme problemi tek tekerli[1] araçların birinci dereceden kinematik modeli göz önünde bulundurarak tasarlanmıştır. Laboratuvarımızda bulunan ve diferansiyel sürüşe sahip Khepera 3 robotlarının hareketi de bu model ile ifade edilebilir. Hız ve konum gibi girdileri verebildiğimiz hareketli bir robotun tek tekerli dinamikleri şu şekilde modellenebilir :

$$\dot{z}_1 = v\cos(\theta) \tag{5.1}$$

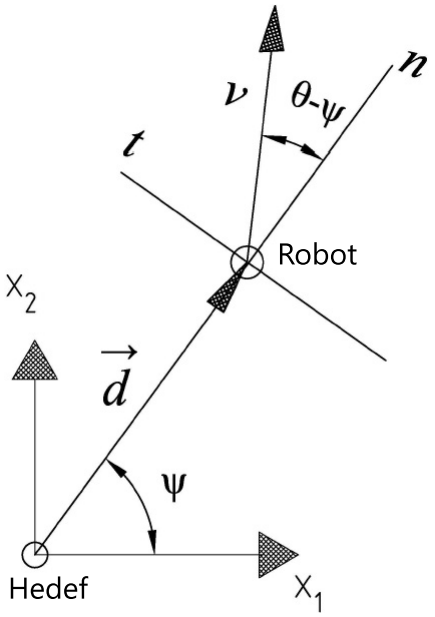
$$\dot{z}_2 = v\sin(\theta) \tag{5.2}$$

$$\dot{\theta} = u \tag{5.3}$$

[1] İng: unicycle



Şekil 5.1.: Kutupsal Koordinatlarda Sistem Modeli (Şekil [24] kaynağından alıntıdır.)



Şekil 5.2.: Sistem modelinin hedefe göreceli koordinat sistemi (Şekil [24] kaynağından alıntıdır.)

Bu model ayrıca daha önceki bölümde 4.5.1. kısmında bahsi geçen erkin modelidir. Buradaki v robotun doğrusal hızı olup Şekil 5.1.'de gösterilmiştir. z_1 ve z_2 ise (z_1, z_2) koordinat sistemine göre robotun konumunun sırasıyla z_1 ve z_2 eksenleri boyunca bileşenleridir. θ ise (z_1, z_2) koordinat sistemine göre robotun doğrultusunun

z_1 ekseni ile yaptığı açıdır. Şekil 5.2.'de ise koordinat eksenleri, merkezi hedef olacak şekilde kaydırılmıştır. Hedef, hareketsiz ve de sürekli konumu ölçülebilir olarak kabul edilmiştir. (z_1, z_2) koordinat sisteminden (x_1, x_2) koordinat sistemine geçildiğinde dinamiklerin yeni modeli şu şekilde değişmiştir [24]:

$$\dot{x}_1 = vcos(\theta) \tag{5.4}$$
$$\dot{x}_2 = vsin(\theta) \tag{5.5}$$
$$\dot{\theta} = u(\theta, \gamma) \tag{5.6}$$

Şekil 5.2.'deki bilgilerden $d = \sqrt{x_1^2 + x_2^2}$ ve $\phi = atan2(x_2, x_1)$ çıkarabiliriz. d, ϕ ve hedefin hareketsiz olduğu varsayımını kullanarak sistem dinamiklerini şu şekilde güncelleyebiliriz [24]:

$$\dot{d} = vcos(\theta - \psi) \tag{5.7}$$
$$\dot{\psi} = \frac{1}{d}vsin(\theta - \psi) \tag{5.8}$$
$$\dot{\theta} = u(\theta, \gamma) \tag{5.9}$$

Burada θ, robotun doğrusal hızının yani anlık doğrultusunun (x_1, x_2) koordinat sistemine göre açısı; ψ, hedeften robota olan vektörün (x_1, x_2) koordinat sistemine göre açısı ve γ da bu iki açının farkıdır, $\gamma = \theta - \psi$.

Yapılan çalışmalarda kullanılan TGTÇ[2] sisteminde, doğrusal hız sabit tutulmuş ve girdi olarak açısal hız kullanılmıştır, $\dot{\theta} = u$. Sistem dinamiklerini matris şeklinde ifade etmek istersek [24]:

$$\begin{bmatrix} \dot{d} \\ \dot{\psi} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} vcos(\theta - \psi) \\ \frac{1}{d}vsin(\theta - \psi) \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u \tag{5.10}$$

θ ve ψ değişkenlerini $\gamma = \theta - \psi$ şeklinde ifade edip birleştirildiğinde ise matris formu şu hale dönüşür [24]:

$$\begin{bmatrix} \dot{d} \\ \dot{\gamma} \end{bmatrix} = \begin{bmatrix} vcos(\gamma) \\ -\frac{1}{d}vsin(\gamma) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{5.11}$$

[2]Tek Girdi Tek Çıktı

Robotun dairesel davranışı için denge noktasında ise ;

$$\dot{d} = vcos(\gamma) = 0 \Rightarrow \gamma = \frac{\pi}{2}(2k+1) \tag{5.12}$$

Robotun dengeye ulaştığında hedef olan mesafesine $d = d_0$ dersek, denge durumunda diğer durum değişkeni de şu şekilde ifade edilir:

$$\dot{\gamma} = u - \frac{1}{d}vsin(\gamma) = 0 \Rightarrow u = \pm\frac{v}{d_0} \tag{5.13}$$

Denge durumunu elde edebilmek için sistem geri besleme doğrusallaştırılması ile doğrusallaştırıldığında doğrusallaştırmayı sağlayan denetleyici [24]:

$$u = -\frac{w}{vsin(\gamma)} + \frac{1}{d}vsin(\gamma) \tag{5.14}$$

biçiminde bulunabilir. Bu denetleyicinin elde edilmesi için detaylı çıkarım için [24] kaynağına bakılabilir. (5.14) eşitliğinde paydada görülen $vsin(\gamma)$'dan dolayı, $\gamma = \theta - \psi = k\pi, k = 1, 2, ...$ durumundan kaçınılmalıdır çünkü bu değerlerde payda sıfır olacağından denetleyiciyinin değerini sonsuza götürecektir ve denetleyici tanımsız olacaktır. Yani, robotun yönünün hiçbir zaman hedefe doğru yaklaşıyor şekilde ya da arkası hedefe bakacak şekilde hedeften uzaklaşıyor olmaması gerekmektedir. Robot denge noktasına ulaştığında beklenen, hedefin etrafında dönüş yönüne bağlı olarak $\gamma = \pi/2$ yada $\gamma = -\pi/2$ değerlerine yakınsamasıdır.

Doğrusallaşmayı sağlayan dönüşüm [24]:

$$\xi_1 = d \tag{5.15}$$
$$\xi_2 = vcos(\gamma) \tag{5.16}$$

biçiminde ifade edilebilir. Burada (5.11) eşitliğinden ;

$$\dot{\xi_1} = \dot{d} = vcos(\gamma) \tag{5.17}$$

(5.11) ve (5.14) eşitliklerinden;

$$\begin{aligned}
\dot{\xi_2} &= -v\dot{\gamma}sin(\gamma) \\
&= -vsin(\gamma)[u - \frac{1}{d}vsin(\gamma)] \\
&= -vsin(\gamma)\left[-\frac{w}{vsin(\gamma)} + \frac{1}{\cancel{d}}\cancel{vsin(\gamma)} - \frac{1}{\cancel{d}}\cancel{vsin(\gamma)}\right] \\
&= w
\end{aligned} \tag{5.18}$$

elde edilir. $\dot{\xi_1}$ ve $\dot{\xi_2}$ değişkenlerini tekrar yazarsak;

$$\dot{\xi_1} = \xi_2 \tag{5.19}$$
$$\dot{\xi_2} = w \tag{5.20}$$

Robotun dairesel davranışı sırasında hedefe olan mesafe $d = d_0$ olduğundan $\xi_1 = d - d_0$ olarak kaydırılmıştır. Burada denetleyiciyi;

$$w = -K_1\xi_1 - K_2\xi_2, K_1 > 0, K_2 > 0 \tag{5.21}$$

biçiminde doğrusal durum geri beslemesi olarak seçersek sistem;

$$\dot{\xi_1} = \xi_2 \tag{5.22}$$
$$\dot{\xi_2} = -K_1\xi_1 - K_2\xi_2 \tag{5.23}$$

olur. Bu sistemin sistem matrisi;

$$J = \begin{bmatrix} 0 & 1 \\ -K_1 & -K_2 \end{bmatrix} \tag{5.24}$$

biçimindedir ve karakteristik polinomu;

$$s^2 + K_2 s + K_1 = 0 \tag{5.25}$$

biçiminde olur. Bu polinomun kökleri;

$$s_{1,2} = \frac{-K_2 \pm \sqrt{K_2^2 - 4K_1}}{2} \tag{5.26}$$

biçimindedir. Burada az-sönümlenmeli cevap için;

$$K_2^2 - 4K_1 < 0 \tag{5.27}$$

ve buradan da;

$$K_1 > \frac{K_2^2}{4} \tag{5.28}$$

sağlanmalıdır. Benzer şekilde çok-sönümlenmeli cevap için ise;

$$K_2^2 - 4K_1 > 0 \tag{5.29}$$

ve buradan da

$$K_1 < \frac{K_2^2}{4} \tag{5.30}$$

sağlanmalıdır. Doğrusal durum geribeslemesi, w ise;

$$w = -K_1\xi_1 - K_2\xi_2 \tag{5.31}$$

olur. w değişkenini (5.14) eşitliğinde yerine yazarsak ;

$$u = \frac{K_1\xi_1 + K_2\xi_2}{vsin(\gamma)} + \frac{1}{d}vsin(\gamma) \tag{5.32}$$

şeklinde ifade edilebilir. Daha da sadeleştirilirse;

$$u = \frac{K_1(d - d_0) + K_2vcos(\gamma)}{vsin(\gamma)} + \frac{1}{d}vsin(\gamma) \tag{5.33}$$

Robotun kamerası ile d mesafesini sürekli ölçebildiğinden ve kamera açısından γ değişkenini bulunabileceğinden, uygun K_1 ve K_2 katsayıları kullanarak u girdisi yani robotun açısal hızı hesaplanabilir.

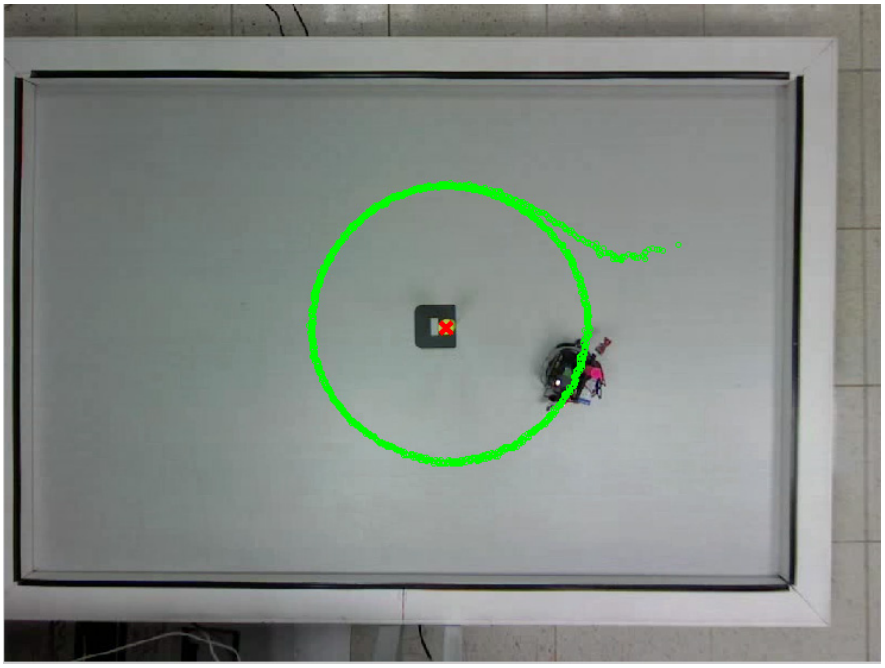
5.2. Robot Üzerinde Uygulamalar ve Sonuçları

Çember takibi deneyleri Şekil 5.3.'de de görülen Sürü Sistemleri Labı'nın 120 cm x 180 cm boyutlarında olan küçük arenasında yapılmıştır. Arenanın sağ kenarı robot için x-ekseni, alt kenarı da y-ekseni olarak kabul edilmiştir. Dizilim deneylerinde kullanılan robot ve hedef düzeneği değiştirilmemiştir. Koordinat eksenleri bu şekilde kabul edilen arenaya robot hedefe yaklaşık 50 cm mesafede ve x-eksenine 60 derece yapacak şekilde konulmuş ve deney başlatılmıştır.

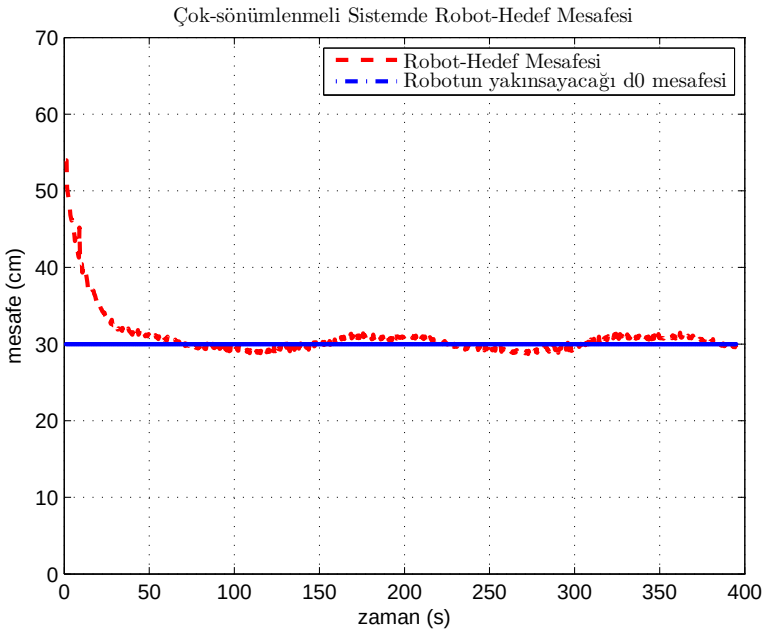
Deneyde çok-sönümlenmeli sistem için $K_1 = 0.002, K_2 = 0.15$, az-sönümlenmeli sistem içinse $K_1 = 1, K_2 = 1$ olarak seçilmiştir. Hedefin yüksekliği 8 cm olarak hazırlanmış, kameranın hedefe uzaklığı da 27.5 cm olarak atanmıştır. Böylece daha önceki bölümde anlatılan kameranın merceği ile dönüş ekseni arasındaki 2.5 cm farktan dolayı robotun 30 cm yarıçapında bir daire çizmesi sağlanmıştır. Hız deneylerde sabit tutulmuş ve $v = 2.5cm/s$ olarak verilmiştir. Dizilim deneylerini de esas alarak belirlenen bu hız, çember takibi için de uygun bir hızdır. Az-sönümlenmeli ve çok-sönümlenmeli sistemler için farklı katsayılar denenmiş bazı durumlarda γ açısı 88-89 derece civarında kalmıştır. (5.33) eşitliğinde sistem dengeye oturduğunda $K_1(d - d_0)$ kısmı sıfır olmaktadır. Dengeleyici kısım olarak her zaman pozitif kalan $\frac{1}{d}vsin(\gamma)$ kısmına karşılık $K_2vcos(\gamma)$ kalmaktadır. γ açısı 90 dereceye ulaşamayıp 88-89 civarında kaldığında ise $\frac{1}{d}vsin(\gamma)$ kısmını sürekli pozitif olarak artırmaktadır. Durum böyle olunca da katsayılara bağlı olarak robot sürekli içeri doğru dönmekte ve spiral çizip hedefe çarpmaktadır. Servo motorların açı hassasiyetinden şüphelenip tekrar bakılmış ama buna sebep olacak bir şey bulunamamıştır. Sorunun kaynağı katsayılar olarak görülmüş ve değiştirilmiştir. Böylece γ açısı 88-92 derece aralığında kalarak $cos(\gamma)$ ile dengeleyici etki oluşturmuştur.

Çok-sönümlenmeli sistem deneyi Şekil 5.3.'de götülmektedir. Beklenildiği gibi yakınsaması gereken mesafeye yaklaşmış ve salınım yapmadan istenilen yarıçapta bir daireye oturmuştur.

Şekil 5.4.'de ise Şekil 5.3.'deki robotun hedefe olan mesafesi çizdirilmiştir. Görüntü işlemede her 5 karede hareket eden pikseller hesaplanmış, arka plan görüntüden çıkarılarak robot ortaya çıkarılmış ve bulunan robotun ağırlık merkezi hesap edilerek robotun hedefe mesafesi bulunmuştur. Yaklaşık 50 cm mesafeden çalışmaya başlayan robot Şekil 5.3.'dekine uygun şekilde istenilen $d_0 = 30cm$ mesafesine yakınsamıştır. Şekil 5.4.'deki hafif dalgalanma robotun 1 saniyede 1 görüntü işlemesinden kaynaklı olarak uygulanan açısal hıza bağlı olarak oluşmaktadır.

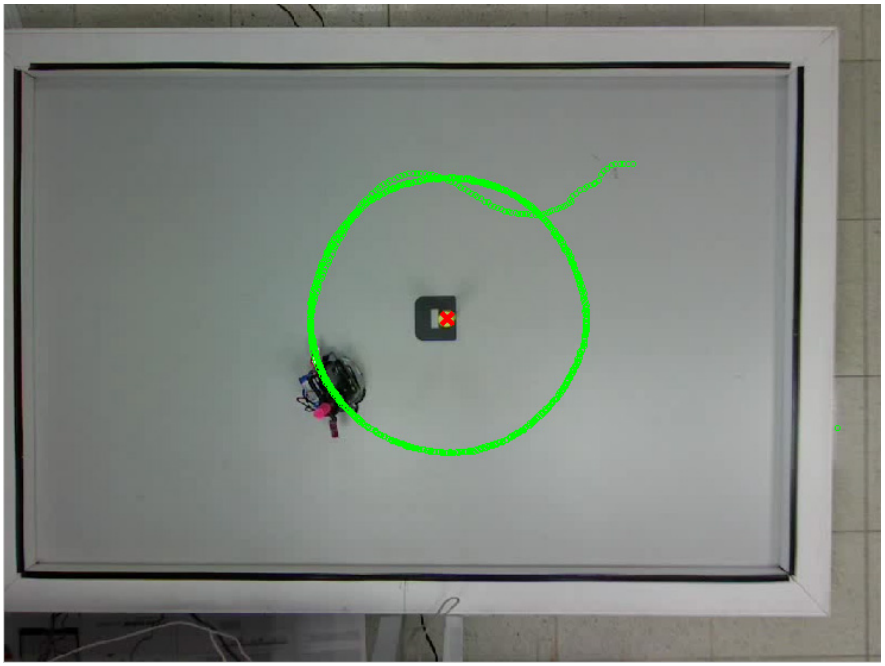


Şekil 5.3.: Çok-sönümlenmeli sistemde robot güzergahı



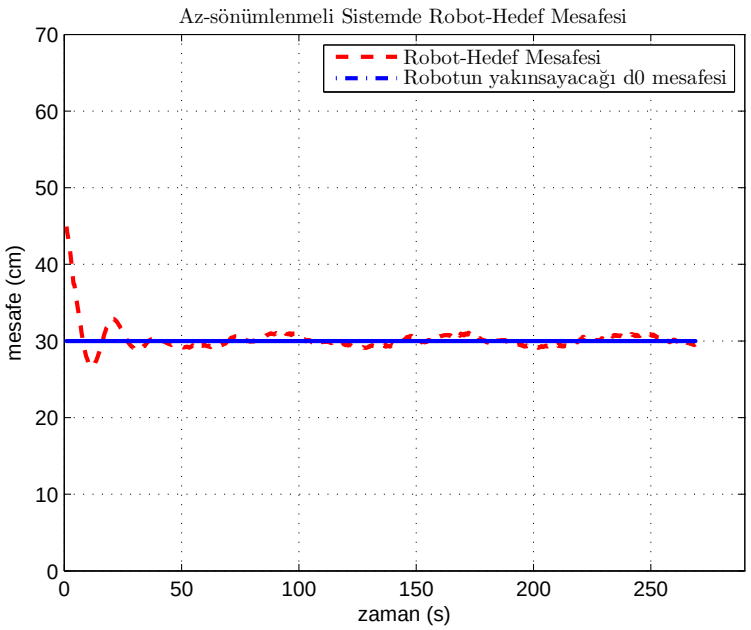
Şekil 5.4.: Çok-sönümlenmeli sistemde robot-hedef mesafesi

Az-sönümlenmeli sistem deneyi ise Şekil 5.5.'da görülmektedir. Burada da az-sönümlenmeli bir sistemden beklenildiği gibi istenilen yarıçapa oturmadan önce robot hafif bir salınım yapmış ve bir süre sonra istenilen yarıçapa yakınsamıştır.



Şekil 5.5.: Az-sönümlenmeli sistemde robot güzergahı

Çok-sönümlenmeli sistem deneyine benzer şekilde görüntü işlemeden sonra çizdirilen robot-hedef mesafesi Şekil 5.6.'da görülmektedir. Şekil 5.5.'de de görülen robotun hedef etrafında yaptığı salınım Şekil 5.6.'da görülmektedir.



Şekil 5.6.: Az-sönümlenmeli sistemde robot-hedef mesafesi

BÖLÜM 6

6. SONUÇ

6.1. Yorumlar

Bu tez çalışmasında çok erkinli robot sistemleri üzerinde görüntü tabanlı robot denetimi yöntemleri gerçeklenmiştir. Robotların kendileri için tasarlanan denetleyicileri ile uygun katsayı değerleri kullanıldığında istenilen amaçlara ulaştıkları gözlenmiştir. Dizilim denetiminde robotların donanımsal yavaşlıkları yanı sıra görüntü işlemede kullanılan OpenCV'den kaynaklanan yavaşlıklar yüzünden dizilimin mecburi olarak az bozulmuş halde hedefine yol aldığı görülmüştür. Bunda robotların kendi içindeki kodların işlem döngülerinin uzun sürmesi, kamera görüş açısının deneylerin sürdürülebileceği minimum değerde olması ve kullanılan Khepera 3 robotlarının tüm yönlü olmamasından dolayı her yöne aniden hareket edememesi etkili olmuştur. Yine benzer problemlerden dolayı çember takibinde de simulasyonlarda çalışmasına rağmen robotun işlem frekansının düşük olmasından kaynaklı sorunlar yaşanmıştır. Fakat, uygun katsayılar kullanılarak bu sorunların üstesinden gelinmiştir.

Bütün bu deneylerin yapılabilmesinde kilit rol oynayan kameralar ve bunlara ait linux sürücülerinin çalıştırılabilmesi uzun bir süre almıştır. Basit bir kamera olmasına karşın deneylerde kullanılan kamera görüntü işleme açısından net görüntü verebildiği için oldukça iyi bir kameradır. Kamera seçiminden sonra robot ile denetlenebilecek en fazla açıya sahip servo motor için bir dizi servo motor denenmiştir. Ancak kamera ile servo motorun robot ile uyumu sağlandığı vakit deneylere başlanabilmiştir. Lider ile başlanan deneylerde her bir robot ile yüzlerce defa deney yapılması ile çalışmanın amacına ulaşması sağlanabilmiştir. 4 robot ile deney yaparken aynı anda bütün robotlar çalıştırıldığında bir tanesinin kilitlenmesi bile bütün deneyi bozduğundan, her robot kendinden önceki robotlar kendi amaçlarını gerçekleştirmiş gibi davranarak farklı kombinasyonlarda defalarca deney yapılmıştır. Muhtemel sorunlar, robotun alt pilinin bitmiş olması, robotun birbirine eklenen KoreBot KoreIOLE gibi kartları arasındaki I2C haberleşmesinde meydana gelen iletişimsizlik, Li-Po pil ile robot arasında herhangi bir temassızlık olması durumunda robotun dışarıdan güç alamaması ve flash belleğin dosya sisteminde meydana gelen sorun şeklinde sıralanabilir. Bunları önlemek için her robotun kodu çalıştırılmadan önce bir çalıştırılan küçük bir program hazırlanmıştır. Bu küçük program önceki deneye ait verileri silerek flash bellekte

herhangi bir sorun olup olmadığının görülmesini sağlamaktadır. Ayrıca battery adında K-Team firması tarafından yazılan bir programı çalıştırarak alt pilin durumunun görülmesine imkan sağlamıştır. Böylece deney ortasında meydana gelen kilitlenmeler minimuma çekilmiştir.

Geri besleme doğrusallaştırılması deneyinde ise önceden belirlenen bir çember şekli için denetleyici tasarlanmış ve uygulamada da istenilen şekilde çember takibi yapılabildiği gözlenmiştir. Bu çalışma sayesinde çember yerine başka bir şekil için de denetleyici uyarlandığında robotun herhangi bir hareketi yapabilmesi ve istenilen rotaya yakınsamasının sağlanabileceği görülmektedir.

6.2. Gelecek Çalışmalar

Sonradan entegre edilen kamera, servo motorlar ile 300 derece çevresini görebilen Khepera 3, görüntü tabanlı hemen her türlü uygulamaya hazır hale getirilmiştir. Bu tez çalışmasında hazırlanan kısımlara bazı eklemeler yapılabilir. Örnek vermek gerekirse dizilim denetlenmesinde arenada hareket eden robotları engelleyebilecek herhangi bir engelin olmadığı görülecektir. Engellerden kaçınılarak yapılacak bir dizilim denetimi çalışmasında lider ve birinci takipçinin engeli geçip hedeflerini görebilecekleri şekilde hız parçaları hazırlanabilir böylece çok erkinli robot sisteminin hedeflerine ulaşması sağlanabilir. Burada sıradan takipçiler için herhangi bir değişiklik yapmaya gerek yoktur çünkü sıradan takipçiler lider ve birinci takipçiye bağımlı olarak onlar hedefi aştıklarında kendileri de aşacaklardır.

Geri besleme doğrusallaştırması ile çember takibinde ise denetleyici TGTÇ modeli kullanarak hazırlanmıştı. Diğer bir yöntem olan ÇGÇÇ modeli de robot üzerinde denenebilir. TGTÇ modelinde doğrusal hız sabit açısal hız girdi olarak verilirken, ÇGÇÇ modelinde hem doğrusal hem de açısal hız değişken ve girdi olarak verilmektedirler.

Daha önce de bahsedildiği gibi robot üzerindeki kamera basit bir kamera olmasına karşın Logitech firmasının en son çıkardığı kameralar içinde en iyilerinden biridir. Her ne kadar Khepera 3 bu kameradan en fazla 800x600 çözünürlükte görüntü çekebilmekte ise de C600 kamerası donanım olarak en fazla 1600x1200 görüntü sağlayabilmektedir. Ortam ışığı sabitken ani görüntü değişimleri vermeyen bu gürbüz kamera ile alınan görüntüden görüntü işleme teknikleri ile rahatlıkla özellik çıkarımı yapılabilir. Bu tez çalışmasında yer almamasına karşın görüntü tabanlı SLAM diğer

bir deyişle VSLAM[1] uygulamaları yapılabilir. Kameradan alınan görüntüden çıkarılan özellikler ile robotun daha önceden bilmediği bir mekanın haritasını çıkarması ve bu haritaya göre de kendi konumunu bulabilmesi zor bir işlem olsa da bu konu üzerinde son yıllarda daha da artan çalışmalar mevcuttur.

Khepera 3 robotunun içindeki KoreBot bir gömülü sistem örneğidir. Üzerinde kendi işlemcisi, RAM'i, belleği gibi standart bir bilgisayarda bulunması gereken temel özelliklere sahiptir. Fakat, deneylerde kullanılan arm-linux işletim sistemi nispeten eski sürüm bir işletim sistemidir ve özellikle USB portunu kontrol eden OTG çipinin linux sürücüsünde hatalar olduğu tahmin edilmektedir. Kamera sürücüsü hazırlanırken de çıkan sorunların çoğunun bu ve benzer sürücülere ait hatalardan kaynaklandığı düşünülmektedir. Eğer işletim sistemi yenilenirse Khepera 3 üzerine takılan kamera veya diğer parçalarla çok daha verimli bir şekilde çalışabilir. Ayrıca robotlar üzerindeki deneylerde OpenCV'nin 1.1 versiyonu kullanılmıştır. Şu an en güncel sürümü 2.1 olan OpenCV kütüphanelerinin yenileri çapraz derlenip robota atılırsa görüntü işleme daha hızlı gerçekleşebilir. Mevcut hali ile bile, yani OpenCV 1.1 ile bile, robotların üzerindeki kodlar 3-4 kat daha hızlı çalıştırılabilir. Fakat bunu yapabilmek için kameranın arabelleğinin anlık görüntüyü alabilmesi için temizlenmesi gerekmektedir. Kamera belirli bir konumda iken gerçek konumundan görüntü alabilmek için arabellekteki 5-8 kare görüntü devamlı okunmakta ve bu şekilde arabellek temizlenebilmektedir. Yeni OpenCV versiyonu ile deneylerde 1-1.2 saniyede bir görüntü işlenebilirken, arabelleği temizleyerek bir eniyileme yapıldığında bu işlem yaklaşık 0.2 saniyeye düşebilir. Böylece görüntü işlemede 5-6 kat hız artışı olmuş olur.

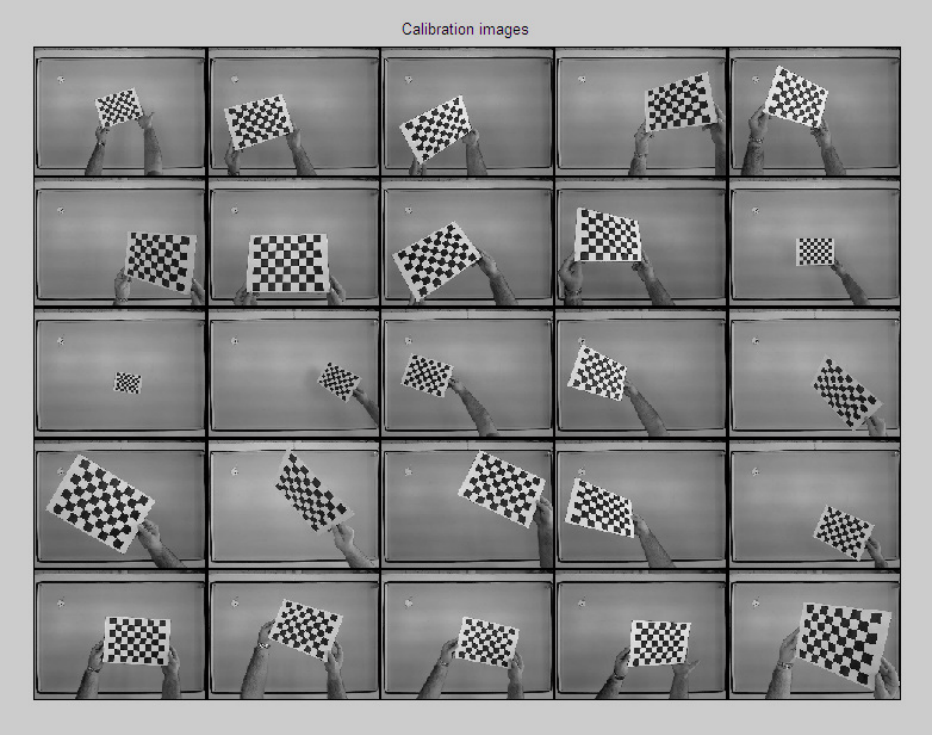
[1] Visual Simultaneous Localization and Mapping

EK 1

A KAMERA KALİBRASYON ARAÇ KİTİ

Jean-Yves Bouguet tarafından hazırlanan diğer bir adı da Caltech Kalibrasyon Araç Kiti olan bu araç kiti, parametreleri bilinmeyen, firma tarafından son kullanıcıya bildirmeye gerek duyulmayan bu tez çalışmasındaki kamera gibi kameraların parametrelerini bulmak için faydalı bir kaynaktır [11]. Bu tez çalışmasında basit bir kameranın parametrelerini bulmak için kullanılan bu araç kiti ayrıca stereo görüş için iki kameralı sistemlerin ve balık gözü kameraların kalibrasyonu için de kullanılabilir. Her yönlü kamera kalibrasyonu için de [25] kaynağına bakılabilir.

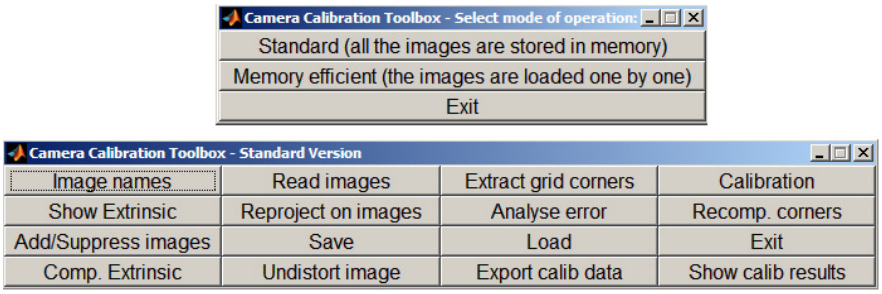
Parametreleri bulunacak kamera için öncelikle yaklaşık 20-30 kare, çalışılacak çözünürlükte daha önceden hazırlanan bir dama tahtası çıktısının görüntüsü çekilir. Şekil A1.'deki görüntüler için çözünürlük, 960x720'dir.



Şekil A1.: Kamera Kalibrasyonu İçin Çekilen Kareler

Şekil A1.'deki görüntüler Sürü Sistemleri Labı'ndaki 120 cm x 180 cm en ve boyundaki küçük arena üzerinde bir tepe kamerası ile çekilmiştir. Kalibrasyon için

kullanılan dama tahtası görüntüsünde siyah beyaz karelerin sayısı çok az olmamakla beraber daha fazla da olabilirdi. Kalibrasyon için siyah beyaz 4 karenin birbirleri ile kesişim noktaları bulunduğu için en dış hat karelerin kesişim noktaları dahil edilmez. Kamera kalibrasyon araç kitinin menüsü Şekil A2.'de görüldüğü gibidir.



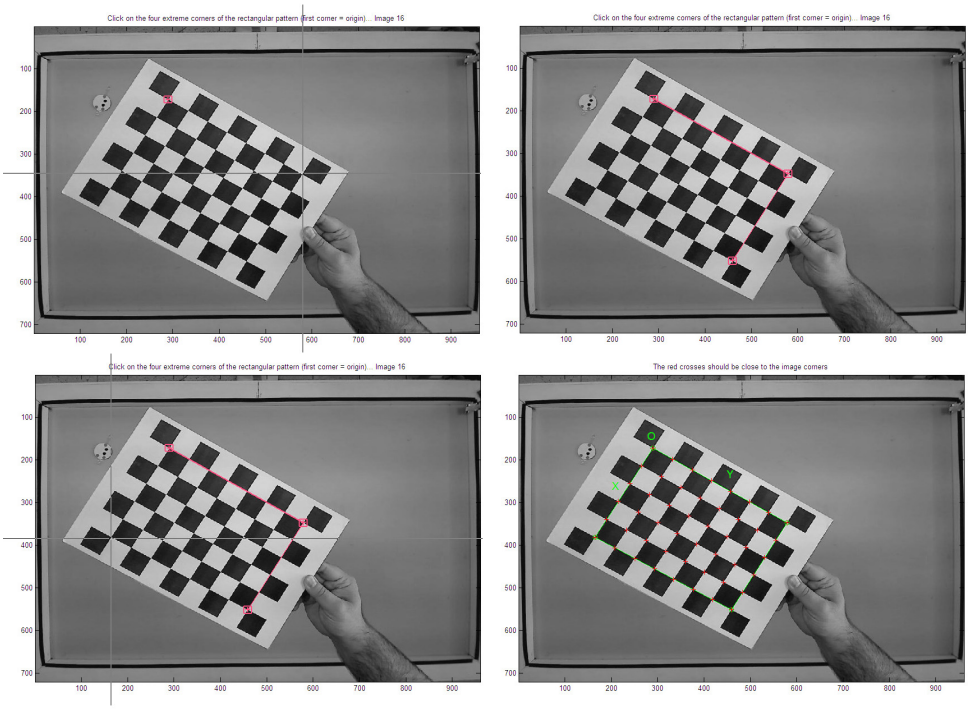
Şekil A2.: Kamera Kalibrasyon Araç Kitinin Menüleri

Şekil A1., Şekil A2.'deki "Image Names" kısmı seçildikten sonra bütün karelerin okunması sonrasında oluşturulur. Daha sonra "Extract Grid Corners" seçilerek her bir görüntü içindeki dama tahtası için yapılan işlem şu şekildedir. Öncelikle sol üst köşedeki 4 siyah beyaz karenin kesişim noktaları fare yardımı ile seçilir. Daha sonra saat yönünde benzer sağ üst, sağ alt ve sol alt üç kesişim noktası için de işlem tekrar edilir. Bu işlem Şekil A3.'de görülmektedir.

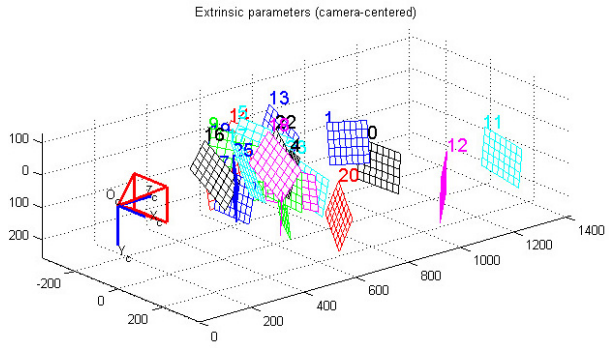
4 iç kesişim noktası seçildiğinde araç kiti bütün kesişim noktalarını otomatik olarak kendisi bulur. Kullanıcıdan karelerin bir kenar uzunluğunu "mm" cinsinden alan araç kiti ile bütün görüntüler için 4 iç kesişim noktası seçimi tekrarlanır. Bütün görüntüler için işlem tamamlandığında menüden "Calibration" seçilir ve yaklaşık olarak bütün kalibrasyon parametreleri bulunur. "Show Extrinsic" kısmı ile kameraya göre çekilen görüntülerin konumları yada görüntülerin kendilerine göre kamera konumları görülebilir. Şekil A4. ve Şekil A5.'de bu konumlar görülmektedir.

Menüden "Reproject on images" kısmı seçildiğinde bütün görüntülerdeki bütün kesişim noktaları iyileştirme yapılarak tekrar hesaplanır. Doğru bulunamayan kesişim noktaları elle kullanıcı tarafından doğru kesişim noktasına getirilerek de düzeltilebilir. İyileştirme yapıldıktan sonra piksel bazındaki hata miktarları bütün noktaları içerecek şekilde menüden "Analyse error" seçilerek çıkan sonuç, Şekil A6.'da görülebilir.

Menüden "Recomp. corners" seçilerek kalibrasyon öncesi işlemler tekrarlanır ve son olarak "Calibration" seçilerek iyileştirilmiş olan kalibrasyon parametreleri elde edilir. "Calibration" seçildikten sonra çıkan kalibrasyon sonuçları Şekil A7.'deki gibidir [11]:



Şekil A3.: Kalibrasyon İçin 4 Noktanın Seçimi

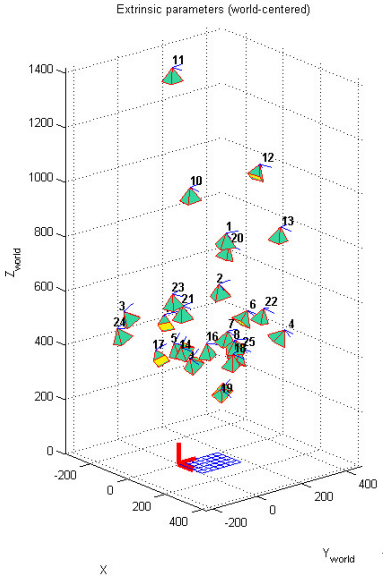


Şekil A4.: Kameraya Göreceli Olarak Görüntü Konumları

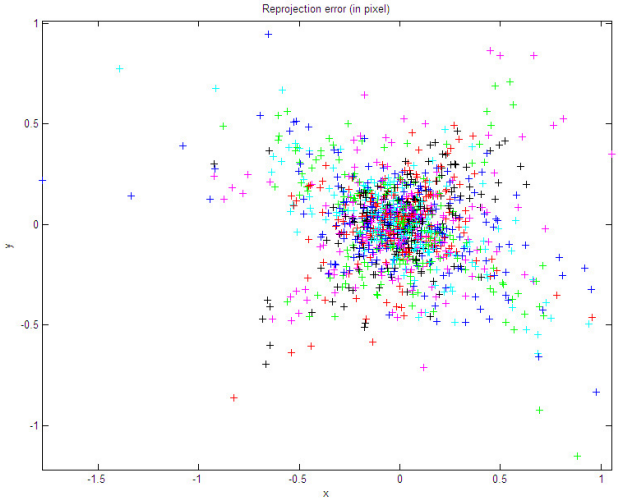
Odak Uzaklığı[1]**:** Daha önceki kısımlarda da anlatılan odak uzaklığı, araç kitinin sonuçlarında x ve y eksenlerindeki odak uzaklığıdır. 2x1 boyutunda bir vektörle ifade edilir ve tez çalışmasında f olarak gösterilmiştir.

Asıl Nokta[2]**:** Kamera görüntüsünün merkezine denk gelen bu nokta, kamera lensinin

[1]ing: Focal Point
[2]ing: Principal Point



Şekil A5.: Görüntülere Göreceli Olarak Kamera Konumları



Şekil A6.: Piksel Bazında Kesişim Noktalarındaki Hata Payları

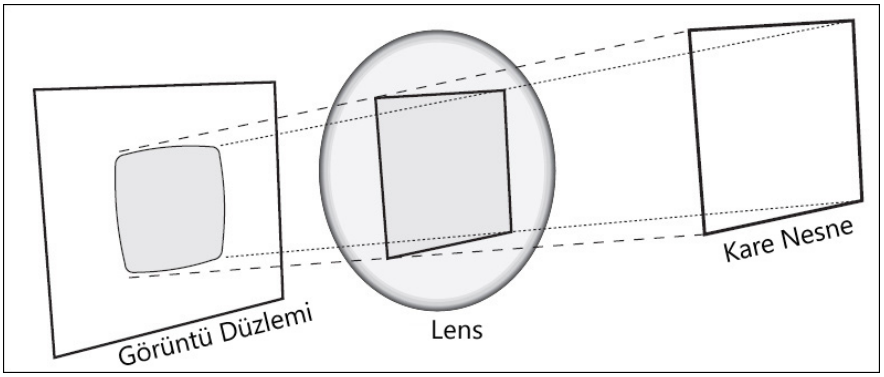
```
Calibration results (with uncertainties):

Focal Length:          fc = [ 804.34599   805.75290 ] ± [ 5.49573   5.45690 ]
Principal point:       cc = [ 490.80793   322.37273 ] ± [ 6.21260   6.20166 ]
Skew:             alpha_c = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:            kc = [ 0.08190   -0.20809   -0.00612   -0.00049  0.00000 ] ± [ 0.01609   0.05287   0.00282
Pixel error:          err = [ 0.28758   0.22459 ]

Note: The numerical errors are approximately three times the standard deviations (for reference).
```

Şekil A7.: Kalibrasyon Sonuçları

ideal olmamasından kaynaklı olarak çalışılan çözünürlükte ideal orta noktadan biraz daha kayık olabilir. 960x720 yapılan çekimlere göre ideal orta nokta *(480,360)* olmalıydı. Halbuki kalibrasyon sonuçlarında da görüldüğü gibi *(490.8,322.3)* olarak gözükmektedir. Bu asıl nokta 2x1 boyutunda bir vektörle ifade edilir ve tez çalışmasında x_0 ve y_0 olarak gösterilmiştir.

Eğiklik[3] **parametresi:** x ve y piksel eksenleri arasındaki açıyı ifade etmek için kullanılan bu parametre skalar bir değer olarak ifade edilir ve tez çalışmasında s olarak gösterilmiştir.

Bozulma[4]**:** 5x1 boyutunda bir vektörle ifade edilen bozulma, içinde 3 tanesi yarıçapsal bozulmaya, 2 tanesi de teğetsel bozulmaya ait toplam 5 katsayı ile ifade edilir. Işık ışınları, basit bir lenste merkezden kenarlara doğru daha fazla kırılıma uğrarlar. Bu yüzden görüntü düzleminde karenin kenarları daha fazla bükülmüş olarak gözükür. Bahsedilen bükülme Şekil A8.'de gösterilmiştir.

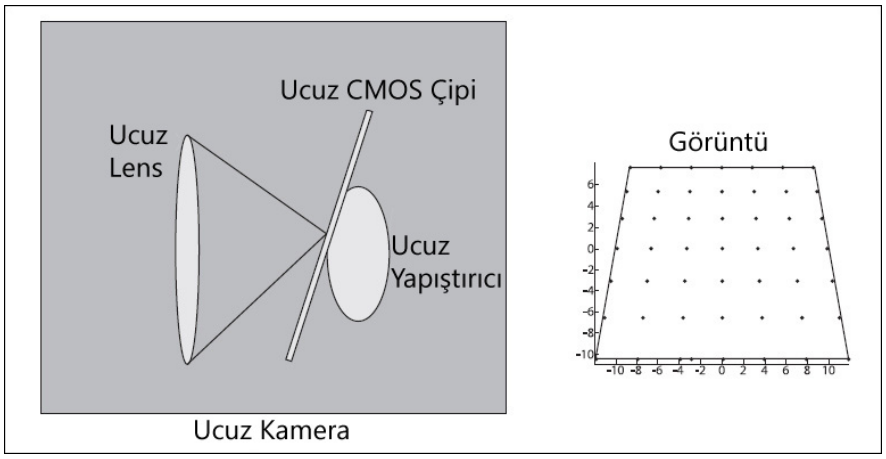


Şekil A8.: Yarıçapsal Bozulma (Şekil [20] kaynağından alıntıdır.)

Kameranın lensi kullanılan yapıştırıcı yüzünden her zaman görüntü düzlemine paralel olmaz. Bu durumda Şekil A9.'daki gibi bir bozulma görülür.

Teğetsel ve yarıçapsal bozulmalara ait katsayılar bu tez çalışmasında kullanılmamıştır. Detaylı anlatım için [11] kaynağına başvurulabilir.

[3]ing: Skew
[4]ing: Distortion



Şekil A9.: Teğetsel Bozulma (Şekil [20] kaynağından alıntıdır.)

EK 2

B DİZİLİM DENETİMİ İÇİN BİRİNCİ TAKİPÇİ KODU

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <math.h>
5  #include <time.h>
6  #include <string.h>
7  #include "khepera3.h"
8  #include "measurement.h"
9  #include "commandline.h"
10 #include "interactiveinput.h"
11 #include "odometry_track.h"
12 #include "odometry_goto.h"
13 #include "cv.h"
14 #include "highgui.h"
15 #include <signal.h>
16 #include <korebot/korebot.h>
17 #include <sys/time.h>
18 #include <sys/types.h>
19
20 #define STDIN 0
21
22 #define LOG_SIZE 1024
23 #define FOV 60
24
25 void servo_control(double);
26 void mesafe_hesapla( IplImage* , double* ,int ,int ,int ,int ,int ,int ,double );
27 double gettime();
28
29
30 double fx=266.98, fy=263.18;
31 double ppx=152.22, ppy=120.45;
32 double lider_yukseklik=0.06;
33 double hedef_yukseklik=0.130;
34
35 int h1_min=159,h1_max=172,s1_min=146,s1_max=256,v1_min=94,v1_max=230;
36 int h2_min=39,h2_max=55,s2_min=102,s2_max=231,v2_min=105,v2_max=240;
37 int frame_temizleme = 5;
38 int en=320;
```

```
39 int boy=240;
40
41 static int quitReq = 0;
42 static knet_dev_t * koreio;
43
44 static void ctrlc_handler( int sig )
45 {
46   quitReq = 1;
47 }
48
49 float max(float a,float b)
50 {
51  if (a>b) return (a);
52  else return(b);
53 }
54
55 // Buffers
56 struct sMeasurement {
57  int sample_number;
58  struct sKhepera3SensorsInfrared infrared_proximity;
59 };
60
61 struct sOdometryGoto og;
62 struct sOdometryTrack s_odometry;
63 struct sMeasurement buf[LOG_SIZE];
64 unsigned int sample_number = 0;
65
66 // Takes one measurement
67 void measurement_take() {
68  khepera3_infrared_proximity_p(&(buf[0].infrared_proximity));
69  buf[0].sample_number = sample_number;
70  sample_number++;
71 }
72
73 void motor_initiliaze()
74 {
75
76  // Initialize left motor
77  if (khepera3_motor_initialize(&(khepera3.motor_left)))
78  {
79   printf("Left␣motor␣initialized.\r\n");
80  }
81  else
82  {
```

```
83    printf("Left motor: initialization failed.\r\n");
84   }
85
86   // Initialize right motor
87   if (khepera3_motor_initialize(&(khepera3.motor_right)))
88   {
89    printf("Right motor initialized.\r\n");
90   }
91   else
92   {
93    printf("Right motor: initialization failed.\r\n");
94   }
95   // Put the wheels in normal (control) mode
96   khepera3_drive_start();
97  }
98
99  void irobstacleavoid();
100 void potential_function(double,double,int);
101 void goto_position(double,double);
102 void goto_heading(double);
103
104 void LookForKey();
105
106 // Important program constants
107 double maxspeed = 0.0165; // Max speed of the robot
108 double maxspeed_koruma = 0.025; // Max speed of the robot
109 double maxturnrate = 25; // Maximum angular speed of the robot...
110 in degrees
111 double maxturnrate_koruma = 40; // Maximum angular speed of the...
112 robot in degrees
113 double epsilon_f=0.01;
114 double epsilon_k=0.01;
115 double sigma_f=0.6;
116 double alfa_turn=0.07;
117 double beta=0.3;
118 double alpha=0.005;
119 double maxdist=0.05;
120 double F[2][2]={{0},{0}};
121 double sample[LOG_SIZE]={0};
122 int i;
123 // Main program.
124 int main(int argc , char * argv[] )
125 {
126  int rc;
```

```
127  int alt_motor_pin=0;
128  int ust_motor_pin=4;
129  int alt_motor_hiza=145;
130  int ust_motor_hiza=145;
131
132  kb_set_debug_level(2);
133
134  if((rc = kb_init( argc , argv )) < 0 )
135   return 1;
136
137  signal( SIGINT , ctrlc_handler );
138
139  koreio = knet_open( "KoreIOLE:Board", KNET_BUS_ANY, 0 , NULL );
140
141  if(!koreio)
142  {
143   printf("Cannot␣open␣KoreIO␣device␣trying␣alternate␣address\r\n");
144   koreio = knet_open( "KoreIOLE:AltBoard", KNET_BUS_ANY, 0 , NULL );
145   if(!koreio)
146    {
147    printf("Cannot␣open␣KoreIO␣device\r\n");
148    return 1;
149    }
150  }
151
152  kio_ConfigIO(koreio,alt_motor_pin,2);
153  kio_ConfigIO(koreio,ust_motor_pin,2);
154  kio_ChangePWM_freq(koreio, 0);
155  kio_ChangePWM_ratio(koreio, alt_motor_pin, alt_motor_hiza);
156  kio_ChangePWM_ratio(koreio, ust_motor_pin, ust_motor_hiza);
157   //servo_control(45);
158
159  // Initialization
160  khepera3_init();
161  motor_initiliaze();
162  commandline_init();
163  measurement_init();
164
165  odometry_track_init();
166  interactiveinput_init();
167
168  // Take continuous measures
169  measurement_configuration.log_size = LOG_SIZE;
170  measurement_configuration.hook_measure = &measurement_take;
```

```
171
172   og.configuration.speed_max=floor((maxspeed_koruma+...
173  (dtor(maxturnrate_koruma)*0.08841/2.0))*140*1000);
174   og.configuration.linearspeed_max=maxspeed_koruma;
175   og.configuration.angularspeed_max=dtor(maxturnrate_koruma);
176
177   if(odometry_track_start(&s_odometry))
178   {
179    printf("Odometry␣initiliazed.\r\n");
180   }
181   else
182   {
183    printf("Odometry␣initiliazation␣failed.\r\n");
184    return -1;
185   }
186
187   odometry_goto_start(&og, &s_odometry);
188
189   s_odometry.initial.flag = 1;
190   s_odometry.initial.x = 0;
191   s_odometry.initial.y = 0;
192   s_odometry.initial.theta = 0;
193
194   coleader_control();
195   return 0;
196  }
197
198  void coleader_control(){
199
200  // FILE *fp;
201  // fp=fopen("Eslider.txt","w");
202   FILE *od;
203   od=fopen("Eslider_odometry.txt","w");
204
205   int amac=0;
206   int cnt2=1;
207   double phi_12f;
208   double robot_yakinlik,hedef_yakinlik;
209
210   robot_yakinlik=0.36;
211   hedef_yakinlik=0.30;
212
213   IplImage* frame;
214   CvCapture* capture;
```

```
215   capture = cvCreateCameraCapture( 0 );
216   assert( capture != NULL );
217   cvSetCaptureProperty (capture, CV_CAP_PROP_FRAME_WIDTH, en);
218   cvSetCaptureProperty (capture, CV_CAP_PROP_FRAME_HEIGHT, boy);
219
220   double turnrate=0, speed=0;
221   char ilkkare[30];
222   int zaman_cnt=0;
223   int v12_flag=0;
224
225   int obje1_flag,obje2_flag;
226   double zaman[10000];
227   double motor_acisi=0;
228
229   double mesafe1[15],mesafe2[15],obje1_tarama[5],obje2_tarama[5];
230   int tarama_cntr;
231   double gentar_acisi,obje1_aci,obje2_aci,phi_coleader,...
232  kolider_hedef_mesafe,kolider_robot_mesafe;
233   double x_eski,x_simdi,z_eski,z_simdi;
234   double v12_x,v12_z,v21_x,v21_z,v22_x=0,v22_z=0,v2_x=0,...
235  v2_z=0,v2_mag,v2_teta;
236   double deltabar12,deltaperp12_x,deltaperp12_z,betabar_2,beta2;
237   double ox_simdi,ox_eski,oz_simdi,oz_eski;
238  // double ox_int,oz_int,ox=0,oz=0,otheta=0;
239   double zaman_simdi,zaman_eski,theta_simdi,theta_eski;
240
241   zaman[zaman_cnt]=gettime();
242   zaman_cnt++;
243   zaman[zaman_cnt]=gettime();
244
245  while( amac == 0) {
246
247   printf("\n\n----------------------------------------------------------------------\n");
248   printf("Dongu numarasi = %d\n\n",cnt2);
249
250   obje1_flag=0;
251   obje2_flag=0;
252
253    for(i = 0; i<frame_temizleme; i++)
254    {
255     frame = cvQueryFrame( capture );
256     if( !frame ) break;
257    }
258
```

```
259   mesafe_hesapla( frame, mesafe1, h1_min, h1_max, s1_min,...
260  s1_max, v1_min, v1_max, lider_yukseklik);
261
262   if(mesafe1[0] > 0)
263    obje1_flag=1;
264
265   mesafe_hesapla( frame, mesafe2, h2_min, h2_max, s2_min,...
266   s2_max, v2_min, v2_max, hedef_yukseklik);
267
268   if(mesafe2[0] > 0)
269    obje2_flag=1;
270
271   tarama_cntr=0;
272   gentar_acisi=0;
273
274   if(mesafe1[0] > 0)
275    obje1_aci=mesafe1[4]+dtor(motor_acisi);
276   if(mesafe2[0] > 0)
277    obje2_aci=mesafe2[4]+dtor(motor_acisi);
278
279   if((obje1_flag == 0) || (obje2_flag == 0))
280    for(tarama_cntr=1;tarama_cntr <=2;tarama_cntr++)
281     {
282     gentar_acisi = motor_acisi + power(-1,tarama_cntr+1)*45;
283     servo_control(max(-120,min(gentar_acisi,120)));
284
285     for(i=0;i<frame_temizleme;i++)
286      {
287       frame = cvQueryFrame( capture );
288       if( !frame ) break;
289      }
290
291     if(obje1_flag==0)
292      mesafe_hesapla( frame, obje1_tarama, h1_min,...
293   h1_max, s1_min, s1_max, v1_min, v1_max, lider_yukseklik);
294
295     if(obje2_flag==0)
296      mesafe_hesapla( frame, obje2_tarama, h2_min,...
297   h2_max, s2_min, s2_max, v2_min, v2_max, hedef_yukseklik);
298
299     if((obje1_tarama[0] > 0) && (obje1_flag == 0))
300      {
301       obje1_flag=1;
302       obje1_aci=dtor(gentar_acisi) + obje1_tarama[4];
```

```
303         mesafe1[3]=obje1_tarama[3];
304        }
305
306       if((obje2_tarama[0] > 0) && (obje2_flag == 0))
307        {
308         obje2_flag=1;
309         obje2_aci=dtor(gentar_acisi) + obje2_tarama[4];
310         mesafe2[3]=obje2_tarama[3];
311        }
312
313     }
314
315    phi_coleader = rtod(obje1_aci + obje2_aci)/2.0;
316
317    kolider_hedef_mesafe = mesafe2[3]-hedef_yakinlik;
318    kolider_robot_mesafe = mesafe1[3]-robot_yakinlik;
319
320    zaman_simdi=gettime();
321
322    odometry_track_step(&s_odometry);
323    fprintf(od,"%f\t%f\t%f\t%f\n",og.track->result.x,og.track->result.y,...
324   rtod(og.track->result.theta),gettime());
325
326    theta_simdi =s_odometry.result.theta;
327    x_simdi = mesafe1[3]*sin(dtor(motor_acisi)+mesafe1[4]+...
328   s_odometry.result.theta);
329    z_simdi = mesafe1[3]*cos(dtor(motor_acisi)+mesafe1[4]+...
330   s_odometry.result.theta);
331    ox_simdi = s_odometry.result.y;
332    oz_simdi = s_odometry.result.x;
333
334    if( v12_flag == 0 )
335    {
336     x_eski = x_simdi;
337     ox_eski = ox_simdi;
338     z_eski = z_simdi;
339     oz_eski = oz_simdi;
340     v12_flag = 1;
341    }
342
343    v12_x = (x_simdi-x_eski+ox_simdi-ox_eski)...
344    / (zaman_simdi-zaman_eski) ;
345    v12_z = (z_simdi-z_eski+oz_simdi-oz_eski)...
346    / (zaman_simdi-zaman_eski) ;
```

```
347
348   deltabar12 = mesafe1[3]*mesafe1[3] - robot_yakinlik*robot_yakinlik;
349
350   v21_x = maxspeed*sign(deltabar12)*sin(obje1_aci);
351   v21_z = maxspeed*sign(deltabar12)*cos(obje1_aci);
352
353   deltaperp12_x = -cos(obje1_aci);
354   deltaperp12_z =  sin(obje1_aci);
355
356   odometry_track_step(&s_odometry);
357   fprintf(od,"%f\t%f\t%f\t%f\n",og.track->result.x,og.track->result.y,...
358  rtod(og.track->result.theta),gettime());
359
360   if(fabs(kolider_hedef_mesafe) < epsilon_f)
361    betabar_2 = 0;
362   else if( (fabs(kolider_hedef_mesafe) >= epsilon_f) && ...
363  (fabs(kolider_hedef_mesafe) < 2*epsilon_f) )
364    betabar_2 = (fabs(kolider_hedef_mesafe) - epsilon_f)/epsilon_f;
365   else if(kolider_hedef_mesafe >= 2*epsilon_f)
366    betabar_2 = 1;
367
368   phi_12f = fabs(rtod(obje1_aci-obje2_aci));
369
370   odometry_track_step(&s_odometry);
371   fprintf(od,"%f\t%f\t%f\t%f\n",og.track->result.x,og.track->result.y,...
372  rtod(og.track->result.theta),gettime());
373
374   if( fabs(phi_12f) > FOV )
375   {
376    v22_x = sigma_f * maxspeed * deltaperp12_x;
377    v22_z = sigma_f * maxspeed * deltaperp12_z;
378   }
379   else
380   {
381    v22_x = sigma_f * maxspeed * betabar_2 ...
382  * sign(cos(fabs(obje1_aci-obje2_aci))) * deltaperp12_x ;
383    v22_z = sigma_f * maxspeed * betabar_2 ...
384  * sign(cos(fabs(obje1_aci-obje2_aci))) * deltaperp12_z ;
385   }
386
387   if(fabs(deltabar12) < epsilon_k)
388    beta2=0;
389   else if( (fabs(deltabar12) < 2*epsilon_k) &&...
390   (fabs(deltabar12) >= epsilon_k))
```

```
391     beta2=(fabs(deltabar12)-epsilon_k)/epsilon_k;
392    else
393     beta2=1;
394
395    odometry_track_step(&s_odometry);
396
397   // v2_x = beta2 * v21_x + sqrt(1-beta2*beta2) * v22_x + v12_x;
398   // v2_z = beta2 * v21_z + sqrt(1-beta2*beta2) * v22_z + v12_z;
399
400    v2_x = beta2 * v21_x + sqrt(1-beta2*beta2) * v22_x;
401    v2_z = beta2 * v21_z + sqrt(1-beta2*beta2) * v22_z;
402
403    v2_teta=atan2(v2_x,v2_z);
404    v2_mag=sqrt(v2_x*v2_x+v2_z*v2_z);
405
406    odometry_track_step(&s_odometry);
407    motor_acisi = 0 -rtod(theta_simdi-theta_eski) + phi_coleader;
408    servo_control(max(-120,min(motor_acisi,120)));
409
410    odometry_track_step(&s_odometry);
411    fprintf(od,"%f\t%f\t%f\t%f\n",og.track->result.x,og.track->result.y,...
412   rtod(og.track->result.theta),gettime());
413
414    speed=v2_mag;
415    turnrate=alfa_turn*v2_teta;
416
417    odometry_track_step(&s_odometry);
418    fprintf(od,"%f\t%f\t%f\t%f\n",og.track->result.x,og.track->result.y,...
419   rtod(og.track->result.theta),gettime());
420    convert_khepera3_drive_set_speed(speed, turnrate);
421
422    x_eski = x_simdi;
423    z_eski = z_simdi;
424    ox_eski = ox_simdi;
425    oz_eski = oz_simdi;
426    zaman_eski = zaman_simdi;
427    theta_eski = theta_simdi;
428
429    cnt2++;
430    }
431   }
432
433   void LookForKey()
434   {
```

```
435   int key, wait_us=50;
436   // Wait for the next key
437   key = interactiveinput_waitkey(wait_us);
438
439   // Quit on ENTER, SPACE or 0
440   if ((key == 10) || (key == 32) || (key == '0')) {
441    convert_khepera3_drive_set_speed(0,0);
442    exit(EXIT_SUCCESS);
443   }
444  }
445
446  void servo_control(double motor_donus_acisi)
447  {
448   double fazlalik_aci,fazlalik_ust_motor_deger;
449   int fazlalik_ust_motor_pwm,alt_motor_pwm;
450   int alt_motor_pin=0;
451   int ust_motor_pin=4;
452   int alt_motor_hiza=145;
453   int ust_motor_hiza=145;
454   double cozunurluk=23.0/35;
455
456   if(motor_donus_acisi > 0){
457
458    if(motor_donus_acisi>(255-alt_motor_hiza)*cozunurluk)
459     {
460      fazlalik_aci=motor_donus_acisi-(255-alt_motor_hiza)*cozunurluk;
461      fazlalik_ust_motor_deger=round(fazlalik_aci/cozunurluk);
462      fazlalik_ust_motor_pwm=ust_motor_hiza + fazlalik_ust_motor_deger;
463      kio_ChangePWM_ratio(koreio, alt_motor_pin, 255);
464      kio_ChangePWM_ratio(koreio, ust_motor_pin, fazlalik_ust_motor_pwm);
465      printf("Alt␣motor=255␣Ust␣motor=%d\n",fazlalik_ust_motor_pwm);
466     }
467    else
468     {
469      alt_motor_pwm=round(motor_donus_acisi/cozunurluk);
470      printf("Alt␣motor=%d␣Ust␣motor=%d\n",...
471  (alt_motor_hiza+alt_motor_pwm),ust_motor_hiza);
472      kio_ChangePWM_ratio(koreio, ust_motor_pin, ust_motor_hiza);
473      kio_ChangePWM_ratio(koreio, alt_motor_pin, alt_motor_hiza+alt_motor_pwm);
474     }
475  }
476   else if(motor_donus_acisi < 0){
477
478    if(fabs(motor_donus_acisi) > (alt_motor_hiza-45)*cozunurluk)
```

```
479    {
480      fazlalik_aci=fabs(motor_donus_acisi)-(alt_motor_hiza-45)*cozunurluk;
481      fazlalik_ust_motor_deger=round(fazlalik_aci/cozunurluk);
482      fazlalik_ust_motor_pwm=ust_motor_hiza - fazlalik_ust_motor_deger;
483      printf("Alt motor=%d Ust motor=%d\n",45,fazlalik_ust_motor_pwm);
484      kio_ChangePWM_ratio(koreio, alt_motor_pin, 45);
485      kio_ChangePWM_ratio(koreio, ust_motor_pin, fazlalik_ust_motor_pwm);
486    }
487   else
488    {
489      alt_motor_pwm=round(fabs(motor_donus_acisi)/cozunurluk);
490      printf("Alt motor=%d Ust motor=%d\n" ,...
491  (alt_motor_hiza-alt_motor_pwm),ust_motor_hiza);
492      kio_ChangePWM_ratio(koreio, ust_motor_pin, ust_motor_hiza);
493      kio_ChangePWM_ratio(koreio, alt_motor_pin, alt_motor_hiza-alt_motor_pwm);
494    }
495    }
496   else
497    {
498      kio_ChangePWM_ratio(koreio, alt_motor_pin, alt_motor_hiza);
499      kio_ChangePWM_ratio(koreio, ust_motor_pin, ust_motor_hiza);
500    }
501  }
502
503  void mesafe_hesapla( IplImage* image, double* msf,int hmin,int hmax,...
504  int smin,int smax,int vmin,int vmax, double real_height)
505  {
506   int i;
507   msf[0]=0;
508   msf[1]=0;
509   msf[2]=0;
510   msf[3]=0;
511   msf[4]=0;
512
513   IplImage* hsv_frame = cvCreateImage(cvGetSize(image), IPL_DEPTH_8U, 3);
514   IplImage* thresholded = cvCreateImage(cvGetSize(image), IPL_DEPTH_8U, 1);
515
516   CvScalar hsv_min = cvScalar(hmin, smin, vmin, 0);
517   CvScalar hsv_max = cvScalar(hmax, smax, vmax, 0);
518
519   cvCvtColor(image, hsv_frame, CV_BGR2HSV);
520   cvInRangeS(hsv_frame, hsv_min, hsv_max, thresholded);
521
522  // cvSaveImage("thresholded.jpg",thresholded);
```

```
523
524   cvSmooth(thresholded , thresholded , CV_MEDIAN, 3,3,1,1);
525
526   int n=0;
527   CvSeq* c;
528   CvPoint* p;
529   int Nc;
530
531   IplImage* img_8uc1 = cvCloneImage( thresholded );
532   IplImage* img_edge = cvCreateImage( cvGetSize(img_8uc1), 8, 1 );
533   IplImage* img_8uc3 = cvCreateImage( cvGetSize(img_8uc1), 8, 3 );
534   cvThreshold( img_8uc1, img_edge, 128, 255, CV_THRESH_BINARY );
535
536   CvMemStorage* storage = cvCreateMemStorage(0);
537   CvSeq* first_contour = 0;
538
539   Nc = cvFindContours( img_edge, storage, &first_contour, ...
540  sizeof(CvContour), CV_RETR_LIST, CV_CHAIN_APPROX_SIMPLE, cvPoint(0,0) );
541
542   for( c=first_contour; c!=NULL; c=c->h_next )
543   {
544
545    cvCvtColor( img_8uc1, img_8uc3, CV_GRAY2BGR );
546
547    for( i=0; i<c->total; ++i )
548     p = CV_GET_SEQ_ELEM( CvPoint, c, i );
549
550   int kose=4;
551   CvPoint2D32f pt_32f[4];
552   CvPoint box_point[4];
553   CvBox2D rect = cvMinAreaRect2(c,storage);
554   cvBoxPoints(rect,pt_32f);
555
556    for (i=0; i<kose ; i++)
557     box_point[i] = cvPoint(cvRound(pt_32f[i].x),cvRound(pt_32f[i].y));
558
559   IplImage* temp = cvCloneImage( img_8uc3 );
560   IplImage* goruntu = cvCloneImage( image );
561
562   if(rect.size.height*rect.size.width >= 40)
563   {
564
565   double objeyukseklik = ((rect.size.height < ...
566  rect.size.width) ? rect.size.width : rect.size.height) + 1;
```

```
567   double orta_x= (box_point[0].x + box_point[1].x + ...
568  box_point[2].x + box_point[3].x)/4;
569   double z_mesafe = fy * real_height / objeyukseklik ;
570   double x_mesafe = fy * real_height * (orta_x - ppx)/fx/ objeyukseklik;
571   double d_mesafe = sqrt(z_mesafe*z_mesafe+x_mesafe*x_mesafe);
572   double teta = atan2(x_mesafe,z_mesafe);
573
574   msf[0]=objeyukseklik;
575   msf[1]=z_mesafe;
576   msf[2]=x_mesafe;
577   msf[3]=d_mesafe;
578   msf[4]=teta;
579   msf[5]=box_point[0].x;
580   msf[6]=box_point[0].y;
581   msf[7]=box_point[1].x;
582   msf[8]=box_point[1].y;
583   msf[9]=box_point[2].x;
584   msf[10]=box_point[2].y;
585   msf[11]=box_point[3].x;
586   msf[12]=box_point[3].y;
587
588   }
589
590   n++;
591
592   cvReleaseImage( &temp);     // Bu iki satir yapilmazsa kod rami sisiriyor.
593   cvReleaseImage( &goruntu);
594   }
595
596   cvReleaseImage( &thresholded );
597   cvReleaseImage( &hsv_frame );
598   cvReleaseImage( &img_8uc1);
599   cvReleaseImage( &img_edge);
600   cvReleaseImage( &img_8uc3);
601
602   cvReleaseMemStorage( &storage );
603  }
604
605  double gettime(){
606
607   char hour[20],minute[20],second[20];
608   struct timeval tv;
609   int H,M,S;
610   double exact_second;
```

```
611
612     time_t curtime;
613
614     gettimeofday(&tv, NULL);
615     curtime=tv.tv_sec;
616
617     strftime(hour,20,"%H",localtime(&curtime));
618     strftime(minute,20,"%M",localtime(&curtime));
619     strftime(second,20,"%S",localtime(&curtime));
620
621     H=atoi(hour);
622     M=atoi(minute);
623     S=atoi(second);
624     exact_second=60*M+S+tv.tv_usec/1000000.0;
625
626     return exact_second;
627
628   }
```

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, Adı	: Karataş, Engin
Uyruğu	: Türkiye Cumhuriyeti
Doğum tarihi ve yeri	: 01.09.1980 Ordu
Medeni hali	: Bekar
Telefon	: 0 (312) 292 42 91
Faks	: 0 (312) 292 40 91
e-mail	: mayantd@gmail.com

Eğitim

Derece	**Eğitim Birimi**	**Mezuniyet Tarihi**
Lisans	Orta Doğu Teknik Üniversitesi Elektrik ve Elektronik Mühendisliği	2007

İş Deneyimi

Yıl	**Yer**	**Görev**
2007-2010	TOBB ETÜ	Araştırma Görevlisi

Yabancı Dil

İngilizce

KAYNAKLAR

[1] R. Murphy, *An Introduction to AI Robotics*. The MIT Press, 2000.

[2] M. J. Woolridge, *An Introduction to Multiagent Systems*. John Wiley & Sons, Ltd. (UK), 2002.

[3] G. Weiss, *Multiagent Systems: A Modern Approach to Distributed Approach to Artificial Intelligence*. The MIT Press, 1999.

[4] G. Tel, "Distributed infimum approximation," Dept of Computer Science, Utrecht Univ., Tech. Rep., 1986.

[5] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*. Addison-Wesley Publishing Company, 1992.

[6] S. Zhai and B. Fidan, "Single view depth estimation based formation control of robotic swarms: Fundamental design and analysis," *Proc. 16th Mediterranean Conf. Control and Automation*, 2008.

[7] V. Gazi, B. Fidan, and S. Zhai, "Single view depth estimation based formation control of robotic swarms: Implementation using realistic robot simulator," *CLAWAR*, 2008.

[8] S. Zhai, B. Fidan, Şadi Çağatay Öztürk, and V. Gazi, "Single view depth estimation based formation control of robotic swarms: Obstacle avoidance, simulation, and practical issues," *16th Mediterranean Conference on Control and Automation*, 2008.

[9] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2. Baskı. Cambridge University Press, 2003.

[10] http://www.the-digital-picture.com/Reviews/Canon-EF-15mm-f-2.8-Fisheye-Lens-Review.aspx. , Erişim Tarihi: 3 Temmuz 2010

[11] http://www.vision.caltech.edu/bouguetj/calib_doc/. , Erişim Tarihi : 12 Haziran 2009

[12] D. Brown, *Photogrammetric Engineering*. The American Society of Photogrammetry, 1971, blm. Close-Range Camera Calibration, sf. 855–866.

[13] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," *ICCV*, 1999.

[14] J. Heikkilä and O. Silvén, "A four-step camera calibration procedure with implicit image correction," *CVPR*, 1997.

[15] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

[16] "Khepera 3," http://www.k-team.com/mobile-robotics-products/khepera-iii. , Erişim Tarihi: 18 Ağustos 2009

[17] "Khepera 3 manual," http://ftp.k-team.com/KheperaIII/Kh3.Robot.UserManual.2.2.pdf. , Erişim Tarihi: 18 Ağustos 2009

[18] "Koreiole manual," http://www.roadnarrows.com/robotics/store/aitdownloadablefiles/download/aitfile/aitfile_id/18/. , Erişim Tarihi: 23 Aralık 2009

[19] "Hekstronik hxt900 servo motor," http://www.servodatabase.com/servo/hextronik/hxt900. , Erişim Tarihi: 8 Temmuz 2010

[20] G. Bradski and A. Kaehler, *OPENCV:Computer Vision with the OPENCV Library*. O'Reilly, 2008.

[21] C. Kimme, D. Ballard, and J. Sklansky, "Finding circles by an array of accumulators," *Short Communications Graphics and Image Processing*, 1975.

[22] B. Fidan, B. D. Anderson, C. Yu, and J. M. Hendrickx, *Modeling and Control of Complex Systems*. Taylor and Francis, 2007, blm. Persistent autonomous formations and cohesive motion control.

[23] S. Sandeep, B. Fidan, and C. Yu, "Decentralized cohesive motion control of multi-agent formations," *14th Meditteranean Conferance on Control and Automation*, June 2006.

[24] A. T. Şamiloğlu, V. Gazi, and A. B. Koku, "Design of circling around a target controllers for mobile robots by feedback linearization," *ALCOSP*, 2010.

[25] http://asl.epfl.ch/~scaramuz/research/Davide_Scaramuzza_files/Research/OcamCalib_Tutorial.htm/. , Erişim Tarihi: 10 Ağustos 2010