T.C.
ONDOKUZ MAYIS ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

HAREKETE DUYARLI İZLEME DÜZENEĞİ

ARİF EMRE ÖZDEMİR

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

SAMSUN
2018

TEZ ONAYI

Arif Emre Özdemir tarafından hazırlanan "Harekete Duyarlı İzleme Düzeneği" adlı tez çalışması 09/05/2018 tarihinde aşağıdaki jüri tarafından Ondokuz Mayıs Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalı'nda **Yüksek Lisans Tezi** olarak kabul edilmiştir.

Danışman Prof. Dr. Güven ÖNBİLGİN
Elektrik-Elektronik Mühendisliği Anabilim Dalı

Jüri Üyeleri

Başkan Prof. Dr. Güven ÖNBİLGİN
Ondokuz Mayıs Üniversitesi
Elektrik-Elektronik Muhendisliği Anabilim Dalı

Üye Dr. Öğretim Üyesi İlyas EMİNOĞLU
Ondokuz Mayıs Üniversitesi
Elektrik-Elektronik Muhendisliği Anabilim Dalı

Üye Dr. Öğretim Üyesi Ali Ekber ÖZDEMİR
Ordu Üniversitesi
Deniz Bilimleri ve Teknolojisi Mühendisliği Anabilim Dalı

Yukarıdaki sonucu onaylarım. / / 2018

Prof. Dr. Bahtiyar ÖZTÜRK
Enstitü Müdürü

ETİK BEYAN

Ondokuz Mayıs Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım bu tez içindeki bütün bilgilerin doğru ve tam olduğunu, bilgilerin üretilmesi aşamasında bilimsel etiğe uygun davrandığımı, yararlandığım bütün kaynakları atıf yaparak belirttiğimi beyan ederim.

25/12/2018

Arif Emre ÖZDEMİR

ÖZET

Yüksek Lisans Tezi

HAREKETE DUYARLI İZLEME DÜZENEĞİ

Arif Emre ÖZDEMİR

Ondokuz Mayıs Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman : Prof. Dr. Güven ÖNBİLGİN

Bu çalışmada, web kamerasından görüntüsü alınan haraketli bir nesnenin Microsoft Visual Studio derleyicisinde C++ programlama dili kullanılarak HSV değerleri tanımlandı ve hareketli nesnenin ağırlık merkezinin konum bilgisinin hesaplaması yapıldı. Görüntülerin daha anlamlı hale gelebilmesi için C++ ve C# programlama dillerinde geliştirilmiş OPENCV kütüphanesi kullanıldı. Görsel program arayüzü ve seri haberleşme için de Qt uygulama geliştirme kütüphanesi kullanıldı.

Hareketli nesnelerin görüntü alanından çıkmaması için web kamerası ile birlikte iki adet RC-Servo motor kullanılarak iki eksenli hareket düzeneği oluşturuldu. RC-Servo motorların konumlandırılması için ARM Cortex-M serisi mbed NXP LPC1768 mikrodenetleyicisi kullanıldı. Bu mikrodenetleyicinin kullanımı için Proteus-ISIS baskı devre tasarım programında tasarlamış olduğum devre kartı kullanıldı. Mbed NXP LPC1768'in üzerinde bulunan USB Serial arayüzü sayesinde Microsoft Visual Studio'dan alınan konum değişim bilgisi bilgisayardan Mbed NXP LPC1768'e iletildi. İnternet üzerinden C/C++ programlama tabanlı derlenebilen mbed NXP LPC1768 ile hareketli nesnenin ekran merkezinden çıkmaması için kod yazılarak aynı devre kartı üzerinde bulunan iki eksenki RC-Servo Motor düzeneğine yön verilerek hareketli nesnenin takibi sağlandı.

Mayıs 2018, 91 Sayfa

Anahtar Kelimeler: Microsoft Visual Studio; HSV; C/C++/C# ; OPENCV; Qt; RC-Servo motor; mbed NXP LPC1768; Proteus-ISIS.

ABSTRACT

Master's Thesis

MOTION SENSITIVE TRACKING SYSTEM

Arif Emre OZDEMIR

Ondokuz Mayis University
Graduate School of Sciences
Department of Electrical and Electronics Engineering

Supervisor : Prof. Dr. Guven ONBILGIN

In this study, the HSV values were defined by using the C++ programming language in the Microsoft Visual Studio compiler of a moving object that taken from webcam image and the position information of the center of gravity of the moving object was calculated. The OPENCV library developed in C++ and C# programming languages was used to make the images more meaningful. The Qt application development library was used for visual program interface and serial communication.

In order to prevent moving objects get out from the image area, a biaxial motion system was formed by using two RC-Servo motors together with webcam. The ARM Cortex-M series mbed NXP LPC1768 microcontroller was used for positioning of RC-Servo motors. In the Proteus-ISIS printed circuit design program for the use of this microcontroller the a circuit board which I designed was used. Through the USB Serial interface on Mbed NXP LPC1768 the position change information from the Microsoft Visual Studio was transmitted from the computer to the Mbed NXP LPC1768. With the NXP LPC1768 which can be compiled with C/C++ programming based over the internet by writting code to prevent the moving object from coming out of the screen center and by giving direction of the biaxial RC-Servo Motor on the same circuit board the moving object was provided.

May 2018, 91 Pages

Key Words: Microsoft Visual Studio; HSV; C/C++/C# ; OPENCV; Qt; RC-Servo motor; mbed NXP LPC1768; Proteus-ISIS.

ÖNSÖZ VE TEŞEKKÜR

Yüksek lisans tezi olarak sunduğum bu çalışmanın planlanması ve yürütülmesinde tecrübelerini ve kıymetli bilgilerini esirgemeyen çok değerli danışmanım Sayın Prof. Dr. Güven ÖNBİLGİN'e ve çalışma konumun belirlenmesinde yardımlarını esirgemeyen, görüntü işleme ve kontrol alanlarında kendimi geliştirmeme olanak sağlayan çok değerli hocam Sayın Yrd. Doç. Dr. Abdullah SEZGİN' e en içten şükranlarımı arz ederim.

Tez çalışmam esnasında teknik desteklerini benden esirgemeyen Yasin AKAN ve Şaban AKDERE' ye teşekkür ederim.

Eğitim ve öğretim hayatım boyunca hiçbir zaman desteklerini benden esirgemeyen annem Penbe ÖZDEMİR'e, babam Şenel ÖZDEMİR'e, ablam Nurgül CAVUNT'a , enişte m Murat CAVUNT'a, çok sevdiğim yeğenlerim Berra Nur CAVUNT ve Yusuf Talha CAVUNT'a teşekkür ederim.

Mayıs 2018, Samsun

Arif Emre Özdemir

İÇİNDEKİLER DİZİNİ

KISALTMALAR

HSV **:** Hue, Saturation, Value
ARM **:** Advanced RISC Machine
USB **:** Universal Serial Bus
RC **:** Radio Control
IIR **:** Infinite Impulse Response
PWM **:** Pulse Width Modulation
COTS **:** Commercial Off-The-Shelf
UAV **:** Unmanned Aerial Vehicles

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

Sayfa

1. GİRİŞ

Görüntü işleme, verilerin toplanıp ölçme ve değerlendirme işleminden sonra başka bir aygıta okunabilir bir biçimde dönüştürülmesi ya da elektronik ortama aktarılmasına yönelik bir çalışma olan "sinyal işlemeden" farklı bir işlemdir (Gonzalez & Woods, 2007). Görüntü işleme yöntemi genellikle bulanıklık, kötü görüntü vb. etkileri filtreler sayesinde azaltmak için kullanılmaktadır. Bunun yanında görüntüdeki nesnelerin tanımlanması ve sınıflandırılması görüntü işleme teknikleri tarafından yapılmaktadır. Teknolojinin gelişmesiyle birlikte tıp, güvenlik, makine vb. birçok alanda görüntü işleme teknikleri kullanılmaktadır. Görüntü işleme, uygulama alanlarına göre zamandan tasarruf sağlar.

Güvenlik sistemleri konusunda teknoloji günden güne artmaktadır. İnsan ile bilgisayar etkileşimi düşünüldüğünde, sağlıklı bir güvenlik sisteminin en temel bileşeninin gözcü ve gözetleme sistemleri olduğu söylenebilir. Bu sistemler bilgisayar tabanlı sistemler olduğundan anlık görüntünün insanlar tarafından takip edilemediği durumlarda, görüntü işleme teknikleri kullanılarak, gözetleme yapılması istenen bölgede tespit ve takipte bulunulmasına imkân verir. Geçmişte analog olarak kullanılan görüntü işleme işlemleri günümüzde dijital olarak yapılmaktadır. İçerik tabanlı video bilgisi elde etme (İTVBE ,Content-based video retrieval) sistemleri için sayısal video işleme bir temel teşkil etmektedir. İçerik tabanlı video bilgi elde etme sistemleri, kullanılan videonun belirlenmesinden sonra inceleme, ayrıştırma ve sınıflandırma işlemlerinin yapılabilir hale gelmesine gereksinim duymaktadırlar. Bu methot hareketli nesnenin içinde bulunmuş olduğu çevrenin özellikleri ve durumları hakkında bilgi elde edinilmesine olanak sağlamaktadır. Bilgisayarlı görü kapsamında hareketli olan nesne veya nesnelerin tespiti ve sonrasında takibi büyük bir problemdir. Bu tür problemlerin yorumlanmasında ise bir kesinlik söz konusu değildir. Bundan dolayı videolarda hareketli olan nesnenin tesbiti ve takibi için gerekli olan sürecin belirli aşamalarının alt problemlere ayrıştırılması daha etkin bir yolla çözüm üretmeyi sağlamaktadır. Bu tür alt problemler değerlendirilerek esas problemin çözümü gerçekleştirilebilmektedir. Videolarda hareketli nesnelerin ekran içerisinde özelliklerine ayrılması için iki yöntem kullanılmaktadır. Bunlar zamansal içerik (temporal segmentation) ve uzamsal içerik (spatial seğmentation) yöntemleridir. Genel olarak nesne tespitinin yapılabilmesi için uzamsal içerik, nesne takibinin yapılabilmesi

için ise zamansal içerik kullanılmaktadır. Kullanılan zamansal ve uzamsal içerik nesnelerin tespit ve takip sürecinde büyük bir problem oluşturabilir.

Video işleme, belirli bir video görüntüsünde var olan sahne içerisindeki değişimleri incelemede kullanılabilecek çeşitli yöntemleri içermektedir. Sahne, kendi içerisinde arka planda veya ön planda var olan çeşitli durağan veya hareketli nesneleri içerebilmektedir. Nesneler ile sahne içerisindeki arka plan ve diğer nesneler arası etkileşimler ise video işlemenin temel olarak ilgilendiği konulardır (Karasulu B, 2010). Bir video dizisi bazı yapısal birimler içermektedir. Hiyerarşik yapıda en küçük birim video çerçevesidir (video frame). Video uygulamalarının, bir analiz işlemine başlamadan önce, ilk işlem olarak bir videoyu video çekimlerine (video shot) bölümlemeleri gerekir (Camara-Chavez vd, 2008).

Günümüzde görüntü işleme bilgisayar bilimlerinin en fazla araştırma yapılan bölümü haline gelmiştir. Literatürdeki birçok çalışmada nesne tespiti ve takibi için, iki-boyutlu videolar, çoklu ortam içerik-tabanlı endekslemede, bilgi elde etmede, dağıtık çapraz-kamera ile gözetleme ve görsel gözetleme sistemlerinde, insan (yaya) takibi, trafik izleme ve üç-boyutlu televizyon (3DTV) uygulamalarında karşılaşılan çeşitli içerik yöntemleri kullanılmaktadır (Remagnino ,2002).

1.1. Görüntü İşlemenin Kullanıldığı Başlıca Alanlar ve Uygulamaları

I. Tasarım ve İmalat Uygulamaları
- Üretim Süreci Ürün Tespiti
- Üç Boyutlu Analiz
- Nesne Tespiti
- Ürün Analizi
- Ürün Hasar Kontrolü
- Robotik-Otomasyon

II. Güvenlik Uygulamaları
- Hedef İzleme
- Nesne Tespiti
- Yüz Tespiti
- Görüntü İyileştirme
- Parmak İzi Tespit

III. Tıp Alanı Uygulamaları
- Mikroskobik
- Kardiografi
- Sintigrafi
- Röntgen ve Ultrason Görüntüleme
- Ortapedi

IV. Mimari Uygulamalarda
- Tarihsel Yapılara Doku Giydirme
- Mimari Yapıların Yeniden Modellenmesi

V. Harita ve Jeodezi Uygulamaları
- Uzaktan Algılama

VI. Gıda Uygulamaları
- Gıda Sınıflandırma
- Besin Alan Tespiti

1.2. Tezin Amacı

Bu çalışmada önce, web kamerası sayesinde hareketli olan nesnenin belirlenmesi daha sonra görüntü işleme teknikleri kullanılarak hareketli nesnenin ekran üzerindeki konumunun belirlenmesi amaçlanmıştır. Oluşturulan iki eksenli RC-Servo motor, web kamerası ve Mikrodenetleyici düzeneği sayesinde hareketli nesnenin bilgisayarın ekran merkezinde bulunmasının sağlanması amaçlanmıştır. Nesnenin hareket sonucundaki yeni konumunun görüntü işleme teknikleri kullanılarak tespit edilmesi ve hareketli nesnenin son konum bilgilerine göre RC-Servo motor, web kamerası ve Mikrodenetleyiciden oluşan düzenek kullanılarak görüntünün bilgisayar ekranının merkezinde bulunması amaçlanmıştır. Sonuç olarak oluşturulan kontrol mekanizması sayesinde web kamerası ile görüntüsü alınan hareketli nesnenin takibinin yapılması amaçlanmıştır.

1.3. Literatür Araştırması

Nesne algılama ve izlemeyi uygulamak için hazırlanan görsel bir projeyle video/resim verileri sürekli kamerayla yakalanır ve Raspberry Pi isimli yazılım ile bilgisayar arayüzü oluşturulur. Hareketli nesnenin şekli ve rengi algılandığında servo motor nesnenin hareket ettiği yere doğru hareket ettirilir. Servo motorlar bir mikrodenetleyici olan Ardunio kartı sayesinde kontrol edilir. Arduino Mikrodenetleyici kartının PWM pinlerine bağlanan servo motorların döndürme açısı kontrol edilir (Vijayalaxmi vd, 2014)

Video seri görüntülerdeki hareketli bölümlemenin asıl amacı görüntülerden temsili gövdeyi parçalamak ve çıkarmaktır. Hareket halindeki nesnenin konumunu daha iyi saptamak için mevcut olan çerçeve ve arka plan arasında fark yaratma yöntemi bu çalışmada sunulmaktadır. Bu algoritmanın temel problemi sabit arka planın

çıkarılmasıdır. Bu çalışmada ise, stabil arka planın çıkarılması için yeni bir yöntem sunulmuştur. Video görüntülerinin gri karakterlerini kullanarak eski N kareden iyi bir arka plan çekilebilmektedir. Daha sonra şimdiki kareye göre arka planı güncellemek için Surendra algoritması kullanılmaktadır (Cheng-liang vd, 2001). Bunun sonucunda da stabil bir arka plan elde edilebilmektedir. Arka plan çıkarıldığında şimdiki çerçeve ve arka plan arasındaki fark bulunabilmektedir. Fark görüntüsü üzerinde bazı işlemler yapıldıktan sonra hareketli nesne doğru bir şekilde çıkarılabilmektedir (Chengzhong vd, 2005).

Video logolarının algılanıp kaldırılması için bu çalışmada video çerçevelerinin geçici ilişkisi kullanılmaktadır. Videoda logo algılama bölümünde başlangıç basamağı olarak, logo sınır kutusu video karelerine önce bir mesafe eşiği kullanılarak yerleştirilir ve ayrıca kenar uzunluklarının bir karşılaştırması kullanılarak arıtılır. İkincisi, önerilen Bayesian sınıflandırıcı yapılmasıdır. Logo-lets olarak adlandırılan logolar parçalarını bulmaktadırlar. Bu çerçevede, güçlü bir algılama sonucunu elde etmek için video logolarının konumu ve iç yerel özellikleri hakkında önceden bilgi birikimi entegre edilmiştir. Logo kaldırma bölümümüzde logo bölgesi işaretlendikten sonra video çekimi sırasında işaretli bölge için en iyi değiştirme yamasını bulmak için eşleşen bir teknik kullanılmıştır. Bu teknik, küçük logoları bulmak için oldukça elverişlidir. Ayrıca görüntü video düzeltme tekniğiyle genişletilmiştir. Görüntü düzeltme tekniğinde 2 boyutlu düşümlerin kullanılmasının aksine, video içindeki geçici bağlantılardan istifade eden 3 boyutlu düşümleri kullanarak video karelerinin logo alanı düzeltilir. Bu algoritmanın avantajı, düzeltilmiş bölgelerin çevreleyen yapı ile uyumlu olması ve sonuçta algısal olarak memnun edici olmasıdır. Uygulamanın sonuçları sunulmuş ve logonun kaldırılması için yöntemlerin faydaları anlatılmıştır (Wq vd, 2005).

Spora özellikle futbola, dünyada birçok insan ilgi duymaktadır. Hızla gelişen iletişim teknolojileri ile insanlar yayınlanan içeriğin çoğunu seyretmek için yeterli zamana sahip olmayabilir. Video içeriği özetleme, bu tür sorunları ortadan kaldıracak yeterlilikte umut verici bir çözüm olarak kendini göstermektedir. Otomatik video içeriği analizi sayesinde kullanıcıya goller, ataklar ve diğer pozisyonlar gibi en önemli olayları içeren bir özet sunulabileceği düşünülmektedir. Dolayısıyla özetleme, video içerikli birçok uygulamada alanında önemli bir rol oynar (Eldib vd, 2009).

Trafik işareti tanımlama; trafik işaretlerini düzenlemek, sürücüyü uyarmak ve belirli eylemleri yönetmek ya da engellemek için kullanılmaktadır. Hızlı gerçek zamanlı ve güvenilir, otomatik trafik işareti algılama ve tanıma özelliği sürücüyü desteklemenin yanısıra sürüş güvenliği ve konforunu da önemli ölçüde artırmaktadır. Otomatik akıllı sürüş aracı veya sürücü yardım sistemleri için trafik işaretlerinin otomatik tanınması oldukça önemlidir. Bu çalışmada sinir ağları tekniğini kullanarak trafik işareti modellerini tanıyan bir çalışma sunulmaktadır. Görüntüler önce eşik teknikleri, Gauss filtresi, Canny kenar algılama, Kontur ve Uyum elipsi gibi çeşitli görüntü işleme teknikleriyle işlenilmektedir. Sistem en iyi ağ yapısını bulmak için geliştirilmiş ve onaylanmıştır. Yapılan çalışmalar, karmaşık arka plan görüntüleri ve önerilen yöntemin hesaplama maliyetiyle trafik işareti modellerinin doğru sınıflandırmalarını göstermektedir (Lorsakul & Suthakorn, 2007).

Bilgisayarlı görme kütüphaneleri ve COTS donanımı kullanılarak gerçek zamanlı görme ve önleme sistemi üzerine araştırmalar yapılmıştır. Öncelikli olarak görme-önleme sisteminde veri izleme ve veri ilişkilendirilmesi yapılır. Bu çalışmada esasen insansız hava araçlarının (UAV) görme ve önleme sorunları üzerine odaklanılmıştır. Çeşitli izleme algoritmaları ve Kalman filtresi kullanılarak geliştirilen bir sistemdir. Bununla birlikte sınırlı çözünürlükteki video kameralar sivil havacılık sahası için kabul edilebilir. COTS donanımı kullanılan sistemler teknolojinin gelişmesiyle ve yüksek çözünürlüklü kameraların kullanılmasıyla daha fazla çalışma alanı sağlayabilir (Driessen, 2004)

Statik videodaki hareketli nesnelerin belirlenmesi ve izlenmesinde farklı yaklaşımlar kullanılmaktadır. Temel olarak statik kamerayla arka plan çıkarımı kullanılmaktadır. Arka plan çıkarma işleminde videonun arka planı öncelikli olarak hesaplanır ve videonun her karesinden çıkartılır. İki ayrı algoritma kullanılarak yabancı nesneler izole edilir. Her iki algoritma da nesnenin ağırlık merkezi için uygulanır ancak ilk algoritma nesnenin hareket edeceği yolu izler. Bu uygulamanın gürültülere karşı çok esnek olmaması nedeniyle hareketli görüntünün etrafına yeşil bir dikdörtgen yerleştirilerek kişiyi takip eden araç geliştirilmiştir. Bu sayede başarılı bir şekilde kişi algılanır ve takip edilir (Grosvenor, 2009).

2. MATERYAL VE YÖNTEM

2.1. Görüntü İşleme Teknikleri

Görüntü işleme yapılırken farklı ihtiyaçlara göre geliştirilen farklı teknikler kullanılır. Geliştirilen yöntemlerle hareketli nesnelerin takibi yapılmaktadır. İlk olarak yapılması gereken hareket eden nesnelere karşılık gelen bölgenin tespit edilmesidir. Çünkü dikkat odağı sağlanmış olur ve daha sonra yapılacak işlemler basit hale gelir. Tekrar eden hareketler (ağaç yapraklarının rüzgardan dolayı hareketleri), ani aydınlatma gibi durumlardan dolayı görüntü işlemek sorun haline gelebilir. Hareketli nesneleri algılamak için sık kullanılan yöntemler arka plan çıkartma, istatiksel yöntemler, zamansal farklar ve optik akış yöntemleridir. Görüntü işleme algoritması Şekil 2.1' de gösterilmiştir.

Şekil 2.1. Görüntü İşleme Algoritması

2.1.1. Arka plan çıkarımı

Genellikle sabit bir zemin üzerindeki hareketli nesneleri (insan, araç, ürün vb.) yakalamak ve takip etmek için görüntü işleme uygulamalarında kullanılan arka plan çıkarımı kullanılmaktadır. Görüntü işleme algoritması sabit kalan Arka Planı ve üzerindeki değişiklikleri tespit ederek hareketleri yakalayabilir.

Arka Plan Çıkarımında kullanılan farklı yaklaşımlar vardır. Bunların en basit versiyonu Heikkila and Silven tarafından kullanılmıştır. Bu yöntemde ilk tanımlanan eşiğe □, ön plan koordinat bilgisi It(x, y) belirtilmiştir (Heikkila & Silven, 1999).

$$|It(x, y) - Bt(x, y)| > \square \qquad (2.1)$$

Arka plan resmi BT bir IIR Filtresi kullanılarak aşağıdaki gibi güncellenmiştir.

$$B_{t+1} = \alpha I_t + (1 - \alpha)B_t \quad (2.2)$$

Arka Plan çıkarma teknikleri çoğu ayıklamada iyi performans gösterse de hareketli bölgelerdeki ilgili piksellerin durmasına rağmen bu teknik, örneğin sabit nesneler arka planı açığa çıkardığında (örneğin, park edilmiş bir araba park yerinden çıkarsa) veya ani aydınlatma değişiklikler oluştuğunda anlık değişimlerde hassastır.

2.1.2. İstatistiksel yöntemler

Takip edilecek nesnenin istatistiksel özellikleri kullanılarak geliştirilen bu yöntem arka plan eksikliklerin düzeltilmesi için geliştirilmiştir. Bu istatistiksel yöntemler temel olarak arka plan çıkarma yöntemleri ve arka planda olan görüntü piksellerinin dinamik olarak istatiksel güncellenmesi sonucunda oluşturulmuştur. Arka plan modelinin her pikselin istatistikleriyle karşılaştırmasıyla oluşan bu yaklaşım gürültü, aydınlatma değişiklikleri ve gölgeler içeren ekranda güvenilirliği sayesinde daha popüler olmuştur (Wang vd, 2003).

İnsanları tespit etmek ve izlemek için gerçek zamanlı bir görsel gözetim sistemi W4, hareketli nesnenin olmadığı ekranda her bir piksenin minimum (M) ve maksimum (N) yoğunluk değerleri, temsil edildiği istatistiksel bir arka plan modeli ve maksimum olduğunda izlenen ardışık çerçeveler arasındaki yoğunluk farkı (D) kullanılır. It anlık görüntü içerisindeki bir piksel olarak sınıflandırılırsa aşağıdaki denklem yazılabilir (Haritaoglu vd, 1998).

$$|M(x, y) - I_t(x, y)| > D(x, y) \text{ veya } |N(x, y) - I_t(x, y)| > D(x, y) \quad (2.3)$$

Eşik işleminden sonra ön planda tespit edilen gürültüyü gidermek için yineleme işlemi uygulanır. Orijinal boyutlarında bozulmuş bölgeyi büyütmek için bozulmuş ve seyreltilmiş ön plan piksel haritası gerçekleştirilir. Ayrıca birbirine bağlı bileşenler bulunduktan sonra küçük boyutlu bölgeler elenir. Böylece resmin hareket etmeyen bölgesine ait arka plan piksellerinin yeni görüntü verileri güncellenmiş olur.

İstatistiksel yöntemlerin bir başka örneği gerçek zamanlı izleme için uyarlanabilir bir arka plan karışımı modelidir (Stauffer & Grimson, 1999). Bu yöntemde her bir piksel anlık gelen bilgilere göre ayrı ayrı Gauss dağılımı ile

modellenir. Bir pikselin ön plan veya arka plan işlemine uygun olup olmadığını belirlemek için Gauss dağılımlarıyla bir model gerçekleştirilir.

2.1.3. Zamansal farklılaşma

Zamansal fark, video dizisinde ardışık karelerin (iki veya üç) piksel piksel farkından faydalanarak hareketli bölgelerin algılanmasında kullanılır. Bu yöntem dinamik sahne değişikliklerinde kullanılsa da bazı haraketli nesnelerin tüm piksellerinin tespit edilmesinde başarısız olur. Örneğin yanlış algılama nesnesi Şekil 2.2' de gösterilmektedir (Dedeoğlu,2004).

(a) (b)

Şekil 2.2. Zamansal Farklılaşma Örneği. (a) İki hareketli nesneyle basit bir ekran görüntüsü. (b) Zamansal Farklılaşma aynı renkteki sol taraftaki nesnenin tüm hareket piksellerinin algılamada başarısız olduğu görüntü

Aşağıda verilen denklem ön plan olarak işaretlenen piksellerin olduğu iki çerçeveli farklılık şemasını göstermektedir (Lipton vd,1998).

$$|I_t(x, y) - I_{t-1}(x, y)| > \square \qquad (2.4)$$

Bazı durumlarda iki çerçeveli farklılığın eksikliklerini gidermek için üç çerçeveli farklılık kullanılabilir (Wang vd, 2003). Örneğin Collins et al. VSAM projesi için arka plan çıkarım modelini uyarlayarak üç çerçeveli farklılığı birleştiren bir hibrid yöntemi geliştirmişlerdir (Collins vd, 2000). Bu hibrid algoritma, arka plan yaklaşımı ve zamansal farklılaşmanın eksiklikleri olmadan videodaki hareketli bölgeleri parçalara ayırır.

2.1.4. Optik akış

Optik akış yöntemlerinde bir görüntüdeki hareketli nesnelerin akış yönleri kullanılır. Bu yöntemde görüntü hareketli bir kameradan sağlansa bile hareketli nesne algılanabilir ancak optik akış yöntemleri çoğu hesaplama açısından karışıktır ve özelleştirilmiş donanımlar olmadan gerçek zamanlı kullanılamazlar (Wang vd, 2003).

2.1.5. Gölge ve ışık değişim tespiti

Hareket algılama için geliştirilen algoritmalar iç ve dış ortamlarda iyi performans göstererek gerçek zamanlı gözetim için kullanılmıştır. Buna rağmen gölge ve ışık değişimlerine bu algoritmaların çoğu hassastır. Gölgeler yüksek seviyede nesne sınıflandırmasında başarısızlığa neden olur. Ani ışık değişimleriyle başa çıkmak için bazı yöntemler geliştirilmiştir. Horprasert ve ark. yeni bir arka plan çıkarma ve gölge algılama yöntemi oluşturmuşlardır (Horprasert vd, 1999). Bu yöntemde her piksel parlaklığı bileşenlerinden ayrılmış renk modeli olarak temsil edilir. Dört ayrı kategoriye (arka plan, gölgeli arka plan veya gölge, vurgulanan arka plan ve hareketli ön plan nesnesi) göre arka plan ve mevcut resim arasındaki parlaklık ve renklilik piksellere göre sınıflandırılır. McKenna ve ark. tarafından geliştirilen bir sistemde meyil bilgileri kullanılır (McKenna vd, 2000). Hareketli bölge içerisindeki güvenliği sağlamak için kullanılan bu yöntemde renk değişiklikleri olmaksızın gölgenin yoğunlukta olduğu alanda meyil bilgileri kullanılır. Diğer bir yöntem arka plan görüntüsü ile karşılaştırıldığında gölge bölgelerindeki piksel yoğunluğunun değerinin azalma eğiliminde olduğu durumlarda geliştirilmiştir (Chen, 2001).

Gölgelerle başa çıkmak için kullanılan en etkili yöntem W4S sistemidir (Haritaoğlu, 1998). Bu yöntemle iki veya daha fazla görüntü kullanan SVM cihazı sayesinde gerçek zamanlı aralık görüntüsü hesaplanır. SVM tarafından sağlanan aralık bilgisi yardımıyla W4S gölgeler ve aydınlatma değişiklikleri ile başa çıkılır.

2.2. Visual Studio

Visual Studio, C, C++, C# gibi programlama dillerinde yazılan kodları oluşturmak için gereken tüm işlevselliği içeren zengin araçlara sahip bir programlama arayüzüdür. Farklı programlama dillerinde oluşturulan modülleri tek bir program gibi birleştiren projeler de yapılabilir. Visual Studio 1979 yılında Bjarne STroustrup tarafından

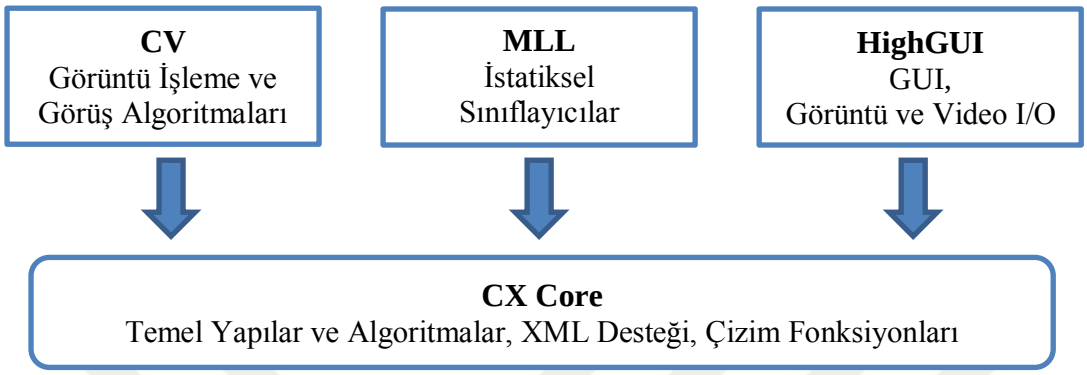
geliştirilmeye başlanmıştır. Visual Studio, C programlama dillerini kapsayan bir arayüzdür. Başlangıçta C sınıfları olarak adlandırılırken 1983 yılından sonra C++ kullanılmaktadır.

C programları aynı zamanda bi C++ programıdır ancak C++ programlama dili C programlama dili gibi kullanılmaz. C++ programlama dilinin avantajı nesne tabanlı programlamada kullanılmasıdır. Nesne tabanlı programlarda kullanılan sınıflarla yeni veri türleri oluşturulabilir. Sınıfların biçimi sayesinde yazılan sınıf kodları aynı türden oluşturulan yeni sınıflarda da çalışır (Yılmaz, 2006).

2.3. OpenCv

OpenCV (Open Source Computer Vision) açık kaynak kodlu görüntü işleme kütüphanesidir. İlk olarak Intel'in Rusya'daki labaratuvarında geliştirilmeye başlanmıştır. İlk olarak C programlama dili ile kodlanmaya başlanmış olmasına rağmen, 2.0 versiyonundan sonra C++ programlama dilli bir yapıya kavuşmuştur. C/C++ dilleri dışında pek çok dille (Python, Java, Matlab,/Octave, C#) beraber kullanılabilir. Özellikle gerçek zamanlı uygulamalar hedef alınarak geliştirilmiş olması, ticari kullanımı dahil ücretsiz olması ve Windows, Linux, MacOS X gibi farklı platformlarda kullanılabilmesi bu kütüphaneyi diğer görüntü işleme araçlarından bir adım öne çıkarmaktadır. OpenCV kütüphanesi, beş temel bileşenden oluşmaktadır. Bu bileşenlerin dört tanesi Şekil 2.3' te görülmektedir. Computer Vision (Bilgisayarla Görü/Görme) kelimesinin baş harfleri kullanılarak isimlendirilen CV bileşeni, temel resim işleme fonksiyonları ve Bilgisayarla Görü/Görme için kullanılan yüksek seviyeli algoritmaları bünyesinde barındıran beş temel kütüphaneden biridir. Machine Learning Library kelimesinin baş harfleri kullanılarak isimlendirilen MLL bileşeni, adından da anlaşılacağı üzere Makina Öğrenmesi dalı için gerekli istatistiksel verilere ulaşmak, mevcut verileri sınıflandırmak için kullanılan fonksiyonları/araçları içeren diğer bir kütüphanedir. HighGUI bileşeni, slider, form gibi OpenCV kütüphanesi içerisinde tanımlanmış pek çok nesneyi yaratabilmemizi sağlayan bir grafik arabirimi olmakla beraber, resim ve videoları kaydetmek, yüklemek, hafızadan silmek için gerekli giriş/çıkış (I/O) fonksiyonlarını da içerir . CXCore bileşeni, OpenCV'ye ait IplImage, cvPoint, cvSize, cvMat, cvHistogram vs gibi veri yapılarını bünyesinde barındıran, XML desteği de sağlayan bir kütüphanedir. Son olarak CvAux bileşeni, şablon eşleştirme (template-matching), şekil eşleştirme (shape matching), bir objenin

ana hatlarını bulma (finding skeletons), yüz tanıma (face-recognition), ağız hareketleri izleme (mouth-tracking), vücut hareketlerini tanıma (gesture recognition) ve kamera kalibrasyonu gibi daha pek çok deneysel algoritmaları bünyesinde barındıran kütüphanedir (Bradski vd, 2008).



Şekil 2.3. OpenCV Bileşenleri

2.3.1. OpenCV bileşenleri

Core: OpenCV'nin temel fonksiyonları ve matris, point, size gibi veri yapılarını bulundurur. Ayrıca görüntü üzerine çizim yapabilmek için kullanılabilecek metotları ve XML işlemleri için gerekli bileşenleri barındırır.

HighGui: Resim görüntüleme, pencereleri yönetme ve grafiksel kullanıcı arabirimleri için gerekli olabilecek metotları barındırır.

Imgproc: Filtreleme operatörleri, kenar bulma, nesne belirleme, renk uzayı yönetimi, renk yönetimi ve eşikleme gibi neredeyse tüm fonksiyonları içine alan bir pakettir.

Imgcodecs: Dosya sistemi üzerinden resim ve video okuma/yazma işlemlerini yerine getiren metotları barındırmaktadır.

Videoio: Kameralara ve video cihazlarına erişmek ve görüntü almak ve görüntü yazmak için gerekli metotları barındırır.

2.4. Qt

Qt, birden çok platformu destekleyen bir grafiksel kullanıcı arayüzü geliştirme programıdır. Genellikle GUI programını geliştirmek için kullanılır. Qt, belli bir platforma bağımlı kalmadan uygulamalar yapmak amacıyla oluşturulmuş bir geliştirme ortamı, aynı zamanda da bir geliştirme kütüphanesidir. 1995 yılında

Trolltech adlı Norveç' li bir firma tarafından geliştirilmiştir. Daha sonra ise Haziran 2008'de Nokia tarafından satın alınmıştır. Genellikle GUI içeren programlar geliştirmede kullanılır. Qt 4 sürümü ile konsol araçları ya da sunucular gibi grafik arayüz içermeyen programlar geliştirmede de kullanılır olmuştur. Qt bir C++ sınıf kütüphanesi ve geliştirme ve uluslararasılaştırma için araçlar içerir. Qt standart C++ kullanır. Ama diğer dillerde kod yazımına da olanak sağlayan bağlayıcıları içerir (binding) : Python (PyQt), Ruby (QtRuby), PHP, C, Perl, Pascal, C#, Java (Jambi).

2.5. C++ Yazılımları

2.5.1. Ayarların dosyaya yazılması ve dosyadan okunması

Ayarların dosyaya yazılmasını ve dosyadan okunmasını sağlayan C++ kodları aşağıda verilmiştir.

```
#pragma once
#include <QFile>
#include <QColor>
#include <QObject>
#include <QVariant>
#include <QSettings>
#include <QApplication>
#include "parameters.h"
class QSettings;
class Config : public QObject
{
    Q_OBJECT
public:
    explicit Config(QObject *parent = 0);
    void LoadConfiguration(QString group = "Default");
    void CreateConfigurationFile(QString group = "Default");
    void SyncConfiguration();
    int GetBaudRate();
    QString GetSoundFile();
```

```
    int GetMusicPlaySecond();

    parameters m_appSet;

    int m_FullScreenWith;

    int m_FullScreenHeight;

private:

    QSettings *m_Settings;

    QString m_Version;

};

#include "config.h"
#include <QFile>
#include <QApplication>
#include <QSettings>
Config::Config( QObject *parent /*= 0*/ ): QObject(parent)
{
    m_Settings = new QSettings("parameters.ini",QSettings::IniFormat,parent);
    QSettings::Status st =  m_Settings->status();
    if (  st ==  QSettings::NoError && QFile::exists("parameters.ini") )
    {
        LoadConfiguration();
    }
    else
    {
        CreateConfigurationFile();
        LoadConfiguration();
    }
};
void Config::LoadConfiguration(QString group /*= "Default"*/)
{
    m_Settings->beginGroup(group); //
    m_appSet.iLowH = m_Settings->value("iLowH",m_appSet.iLowH).toInt();
    m_appSet.iHighH = m_Settings->value("iHighH",m_appSet.iHighH).toInt();
    m_appSet.iLowS = m_Settings->value("iLowS",m_appSet.iLowS).toInt();
    m_appSet.iHighS = m_Settings->value("iHighS",m_appSet.iHighS).toInt();
    m_appSet.iLowV = m_Settings->value("iLowV",m_appSet.iLowV).toInt();
    m_appSet.iHighV = m_Settings->value("iHighV",m_appSet.iHighV).toInt();
    m_appSet.m_minRadius = m_Settings-
>value("m_minRadius",m_appSet.m_minRadius).toInt();
    m_appSet.m_maxRadius = m_Settings-
>value("m_maxRadius",m_appSet.m_maxRadius).toInt();
    m_appSet.m_minFillness = m_Settings-
>value("m_minFillness",m_appSet.m_minFillness).toDouble();
    m_appSet.m_method = m_Settings-
>value("m_method",m_appSet.m_method).toInt();
    m_appSet.m_input = m_Settings-
>value("m_inputType",m_appSet.m_input).toInt();
    m_appSet.m_fileName = m_Settings-
>value("m_fileName",m_appSet.m_fileName).toString();
    m_Settings->endGroup();
}

void Config::CreateConfigurationFile(QString group)
{
    m_Settings->beginGroup(group);
    m_Settings->setValue("iLowH",m_appSet.iLowH);
```

```
    m_Settings->setValue("iHighH",m_appSet.iHighH);
    m_Settings->setValue("iLowS",m_appSet.iLowS);
    m_Settings->setValue("iHighS",m_appSet.iHighS);
    m_Settings->setValue("iLowV",m_appSet.iLowV);
    m_Settings->setValue("iHighV",m_appSet.iHighV);
    m_Settings->setValue("m_minRadius",m_appSet.m_minRadius);
    m_Settings->setValue("m_maxRadius",m_appSet.m_maxRadius);
    m_Settings->setValue("m_minFillness",m_appSet.m_minFillness);
    m_Settings->setValue("m_method",m_appSet.m_method);
    m_Settings->setValue("m_inputType",m_appSet.m_input);
    m_Settings->setValue("m_fileName",m_appSet.m_fileName);
    m_Settings->endGroup();
    m_Settings->sync();
}

void Config::SyncConfiguration()
{
    CreateConfigurationFile();
}
```

2.5.2. Görüntü işleme algoritmaları

OpenCV kütüphanesini kullanarak webcamden alınan resimlerde görüntü işleme algoritmalarını çalıştırılan ve kullanıcı arayüzünde gösterilecek şekilde ileten temel işlemlerin yapıldığı bölüm burasıdır. Opencv sayesinde matematiksel bilgileri Qt arayüzünde kullanılan QImage nesnesine çevrilir. Web kamerasından görüntü alınmasını sağlayan OpenCV sınıfı burada yazılır. Web kamerası tarafından alınan bilgilerdeki görüntü ayarlarının tutulduğu nesne burada tanımlanmıştır. Görüntü işleme algoritmalarının kullanıldığı sınıfları içeren C++ kodları aşağıda verilmiştir.

```
#pragma once

#include "opencv2/highgui/highgui.hpp"

#include <iostream>

#include <QImage>

#include "result.h"

#include "parameters.h"

#include "opencv/highgui.h"

#include "Serial.h"

#include <QMutex>

static int max_thresh = 255;

static int thresh = 100;

using namespace cv;
```

```
using namespace std;

class ObjectTracker

{

public:

    ObjectTracker(parameters pram);

    ~ObjectTracker();

       /*

       Temel işleminin yapıldığı fonksiyondur.

       */

    result* process(QImage &img1,QImage &img2);

       /*

       Opencv mat nesnesini Qt arayüzünde kullanılan QImage nesnesine ceviren
fonksiyondur.

       */

    QImage MatToQImage(const Mat& mat);

       /*

       Web kamerasından görüntü alınmasından sorumlu Opencv sınıfıdır.

       */

    VideoCapture *m_cap;

       /*

       Web kamerasından alınan görüntünü tutuldugu nesnedir.

       */

    Mat m_frame;

       /*

       Ekran ayarlamasının yapılan bölümdür.

       */

    void setSettings(parameters param);

       /*

       Görüntünün eşik değer işlemini yapar.

       */

    Mat GetThresholdedImage(Mat img);
```

```
        /*
        Sayıcıyı hesaplar.
        */
    void getCounters(Mat& imgYellowThresh, Mat &imgScribble);
    parameters m_params;
    bool getPOsitions(float& l,float& u);
    void setPOsitions( float l,float u );
    float m_l;
    float m_u;
    int c;
    int lastC;
    QMutex m_mtex;
};

#include "objecttrack.h"
#include <QPixmap>
#include <opencv2/opencv.hpp>
#include "opencv/cv.h"
#include "opencv/highgui.h"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <exception>
#include "PositionCalculator.h"
using namespace cv;
using namespace std;
using namespace cv;
static RNG rng(12345);
ObjectTracker::ObjectTracker(parameters pram) :m_params(pram)
{
    m_l =0.5;
    m_u =0.5;
    int ii = 0;
    c = 0;
    lastC =0;
    m_cap = new VideoCapture(m_params.m_input);
    if (!m_cap->isOpened())  // Başarılı olunmadığında programdan çıkar
    {
        cout << "Cannot open the video cam" << endl;
        return;
    }
    double dWidth = m_cap->get(CV_CAP_PROP_FRAME_WIDTH); //Alınan video
görüntünün genişliğinin tutulduğu yer
```

```
    double dHeight = m_cap->get(CV_CAP_PROP_FRAME_HEIGHT); // Alınan video
görüntünün yüksekliğinin tutulduğu yer
}
ObjectTracker::~ObjectTracker()
{
    delete m_cap;
}
QImage ObjectTracker::MatToQImage(const Mat& mat)
{
    if (mat.type() == CV_8UC1)
    {
        // Renk tablosunun ayarlandığı bölüm( Renk değerlerinin qRGB'ye
çevrildiği bölüm))
        QVector<QRgb> colorTable;
        for (int i = 0; i < 256; i++)
            colorTable.push_back(qRgb(i, i, i));
        // Giriş değerlerinin kopyalandığı bölüm
        const uchar *qImageBuffer = (const uchar*)mat.data;
        // Giriş değerleriyle aynı boyutta resim dosyasının oluşturulduğu
bölüm
        QImage img(qImageBuffer, mat.cols, mat.rows, mat.step,
QImage::Format_Indexed8);
        img.setColorTable(colorTable);
        return img;
    }
        else if (mat.type() == CV_8UC3)
    {
        // Giriş değerlerinin kopyalandığı bölüm
        const uchar *qImageBuffer = (const uchar*)mat.data;
        // Giriş değerleriyle aynı boyutta resim dosyasının oluşturulduğu
bölüm
        QImage img(qImageBuffer, mat.cols, mat.rows, mat.step,
QImage::Format_RGB888);
        return img.rgbSwapped();
    }
    else
    {
        //Resim dosyasına dönüştürülemediğinde oluşan hata
        return QImage();
    }
}
Mat equalizeIntensity(const Mat& inputImage)
{
    if(inputImage.channels() >= 3)
    {
        Mat ycrcb;
        cvtColor(inputImage,ycrcb,CV_BGR2YCrCb);
        vector<Mat> channels;
        split(ycrcb,channels);
        equalizeHist(channels[0], channels[0]);
        Mat result;
        merge(channels,ycrcb);
        cvtColor(ycrcb,result,CV_YCrCb2BGR);
        return result;
    }
    return Mat();
}
result* ObjectTracker::process(QImage &img1,QImage &img2) {
    result *retVal = new result;
    bool bSuccess = m_cap->read(m_frame); // Videodan yeni resim çerçevesi oku
    if (!bSuccess) //Başarısız olunduğunda döngüden çıkma
    {
```

```
            cout << "Cannot read a frame from video stream" << endl;
        }
        else
        {
            Mat imgYellowThresh = GetThresholdedImage(m_frame);
            getCounters(imgYellowThresh, m_frame);
            img1 = MatToQImage(m_frame);
            img2 = MatToQImage(imgYellowThresh);
        }
        return retVal;
}
Mat ObjectTracker::GetThresholdedImage(Mat img)
{
    Mat imgHSV;
    cvtColor(img, imgHSV, COLOR_BGR2HSV); // Yakalanan görüntünün BGR den
HSV'ye dönüştürme
    Mat imgThresholded;
    inRange(imgHSV, Scalar(m_params.iLowH, m_params.iLowS, m_params.iLowV),
Scalar(m_params.iHighH, m_params.iHighS, m_params.iHighV), imgThresholded);
    // Küçük nesneleri ön alandan kaldırma
    erode(imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE,
Size(5, 5)) );
    dilate( imgThresholded, imgThresholded,
getStructuringElement(MORPH_ELLIPSE, Size(5, 5)) );
    //Ön plandaki küçük delikleri doldurma
    dilate( imgThresholded, imgThresholded,
getStructuringElement(MORPH_ELLIPSE, Size(5, 5)) );
    erode(imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE,
Size(5, 5)) );
    return imgThresholded;
}
void ObjectTracker::getCounters(Mat &imgYellowThresh, Mat &frame) {
    vector<vector<Point> > contours;
    vector<Vec4i> hierarchy;
    Point2i center;
    float radius = 0.0;
    int naxx=0;
    Mat counterImg = imgYellowThresh.clone();
    /// Yeni çevre oluşturma
    findContours(counterImg, contours, hierarchy, CV_RETR_TREE,
CV_CHAIN_APPROX_SIMPLE, Point(0, 0));
    float tempradius=0;
    Point2f tempcenter;
    // Görüntünün çevresinin çizildiği bölüm
        bool tt = false;
    for (int i = 0; i < contours.size(); i++)
    {
        minEnclosingCircle(contours[i], tempcenter, tempradius);
        if (  tempradius >= radius  &&
                tempradius <= m_params.m_maxRadius &&
                tempradius >= m_params.m_minRadius
           )
        {
            radius = tempradius;
            center = tempcenter;
            tt = true;
        }
         }
    if (contours.size() && tt)
    {
        Scalar color = Scalar(255, 0, 0);
```

```
        Rect region_of_interest = Rect(center.x - radius , center.y - radius,
radius*2, radius*2);
        region_of_interest.x = region_of_interest.x > 0 ? region_of_interest.x
: 0;
        region_of_interest.y = region_of_interest.y > 0 ? region_of_interest.y
: 0;
        region_of_interest.width -= (region_of_interest.x +
region_of_interest.width) > counterImg.cols ?
                                    (region_of_interest.x +
region_of_interest.width) - counterImg.cols :
                                    0;
        region_of_interest.height -= (region_of_interest.y +
region_of_interest.height) > counterImg.rows ?
                                     (region_of_interest.y +
region_of_interest.height) - counterImg.rows :
                                     0;
        if (region_of_interest.x <0 || region_of_interest.y < 0)
        {
            int a = 67;
        }
        Mat image_roi = counterImg(region_of_interest);
        int c = countNonZero(image_roi);
        double fillnes = (c / (radius * radius * 4) * 400 / 314 );
        if (fillnes >= m_params.m_minFillness)
        {
            rectangle(m_frame, region_of_interest, color, 2);
            circle(m_frame, center, radius, color, 5);
            setPOsitions(center.x,center.y);
        }
    }
}
void ObjectTracker::setSettings(parameters param)
{
    m_params = param;
}
bool ObjectTracker::getPOsitions( float& l,float& u )
{
    bool retVal = false;
    m_mtex.lock();
    l = m_l;
    u = m_u;
    retVal = lastC != c;
    if (retVal)
    {
        lastC =c;
    }
    m_mtex.unlock();
    return retVal;
}
void ObjectTracker::setPOsitions( float l,float u )
{
    m_mtex.lock();
    m_l =l;
    m_u = u;
    c++;
    c = c >1000 ? 0 : c;
    m_mtex.unlock();
}
```

Görüntü işleme algoritmaları çalışırken çok fazla işlem yükü gerektireceğinden dolayı ayrı bir işlem parçacığında program çalıştırılır. Bu sayede kullanıcı arayüzü ve seriport haberleşme kontrolleri aksamadan yapılmış olur. Görüntü işleme algoritmalarının çalıştırıldığı Thread'in C++ kodları aşağıda verilmiştir.

```
#pragma once
#include <QThread>
#include <QImage>
#include "result.h"
#include <QMetaType>
#include "objecttrack.h"
#include <QMutex>
Q_DECLARE_METATYPE(result);
class ProcessThread : public QThread
{
    Q_OBJECT
public:
    ProcessThread(ObjectTracker *);
    void run();
    void setExit();
    QImage m_img1;
    QImage m_img2;
    void getImages(QImage &img1, QImage &img2);
    void setImages(QImage img1, QImage img2);
    ObjectTracker *m_processoer;
private:
    bool m_exit;
    QMutex m_mutex;
signals:
    void resultReady(void *img);
    void ready();
};
```

```
#include "opencvThread.h"
#include "objecttrack.h"
#include <QImage>
#include <ctime>
void ProcessThread::run()
{
    while (m_exit)
    {
        if (m_processoer != NULL)
        {
            using namespace std;
            clock_t begin = clock();
            QImage img1,img2;
            result *r = m_processoer->process(img1,img2);
            setImages(img1,img2);
            clock_t end = clock();
            double elapsed_Msecs = double(end - begin) ;
            if (elapsed_Msecs <40 )
            {
                msleep(40-elapsed_Msecs);
            }
            else
            {
                msleep(4);
            }
            delete r;
        }
    }
}

ProcessThread::ProcessThread(ObjectTracker *worker)
{
    m_exit = true;
    m_processoer = worker;
}
void ProcessThread::setExit()
{
    m_exit = false;
}
void ProcessThread::getImages( QImage &img1, QImage &img2 )
{
    m_mutex.lock();
    img1  =m_img1;
    img2 = m_img2;
    m_mutex.unlock();
}
void ProcessThread::setImages( QImage img1, QImage img2 )
{
    m_mutex.lock();
    m_img1 = img1;
    m_img2 = img2;
    m_mutex.unlock();
}
```

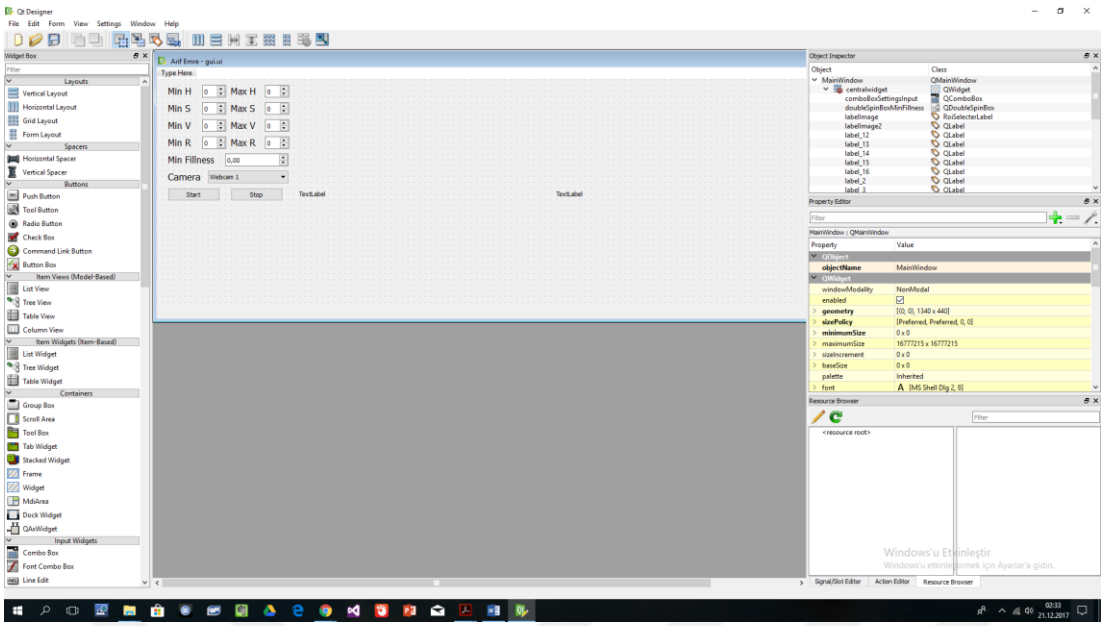
2.5.3. Kullanıcı arayüzünün uygulanması

Görüntü işleme algoritması bölümünde hesaplanan parametrelerin ve üretilen resimlerin aktarılmasınında kullanılan C++ programı aşağıda verilmiştir.

```
#pragma once

#include <QImage>

#include <vector>

class result

{

public:

    result();

    ~result();

    std::vector<QImage> m_imglist;

    void pushImage(QImage img);

    QImage getImage(int i);

};
```

```
#include "result.h"
result::result()
{
}
result::~result()
{
}
void result::pushImage(QImage img)
{
    m_imglist.push_back(img);
}
QImage result::getImage(int i)
{
    if (m_imglist.size()>i)
        return m_imglist[i];
    QImage img;
    return img;
}
```

Visual Studio geliştirme programında C++ programlama dili kullanılarak yazılan programda Qt Designer arayüzü sayesinde kullanıcı ekranında kısayol araçları oluşturulur. Oluşturulan arayüz Şekil 2.4' te gösterilmiştir.



Şekil 2.4. Qt Designer Arayüzü

Kullanılan kısayol araçları oluşturulduğu C++ kodları aşağıda verilmiştir.

```
#ifndef GUI_H // Görsel alet çantasının tanımlanması

#define GUI_H // Görsel alet çantasının atanması

#include <QtGlobal>

#if QT_VERSION >= 0x050000 //Qt Versiyon testi

#include <QtWidgets/QMainWindow> // 5'den büyükse kullanılacak kütüphane

#else

#include <QtGui/QMainWindow> // 5'den büyükse kullanılacak kütüphane

#endif

#include "ui_gui.h" // Grafiksel kullanıcı arayüzü kütüphanesini ekler

#include <QImage>

#include "opencv/highgui.h"

#include "result.h"

#include "objecttrack.h"

#include "opencvThread.h"

#include "serial.h"

#include "config.h"

#include "serialportthread.h"
```

```
#include <QTimer>
class Gui : public QMainWindow
{
    Q_OBJECT
public:
    Gui(QWidget *parent = 0, Qt::WindowFlags flags = 0);
    ~Gui();
       void readParamsFromGui(parameters &params);
       void InitVideoObject(QString fileName );
    void startWorkInAThread();
       ObjectTracker *m_tracker;
    Config m_settings;
    MySerial m_serial;
    ProcessThread *m_precess;
    SerialPortThread *m_serialThread;
private:
    Ui::MainWindow ui;
    QTimer *m_timer;
public slots:
    void setLabelImages();
    void handleResults(void*);
    void resetparams();
    void setGuiParams(QString name);
    void currentInputChangedg(const QString & text);
    void startSerialPort();
    void stopSerialPort();
    void imageReady();
signals:
       void start();
};
#endif // GUI2_H
```

Qt Arayüzü kullanılarak hazırlanan görsel ekranda kullanılan özelliklerin ayarlarının tutulduğu C++ kodları aşağıda verilmiştir.

```
#ifndef UI_GUI_H
#define UI_GUI_H
#include <QtCore/QVariant>
#include <QtGui/QAction>
#include <QtGui/QApplication>
#include <QtGui/QButtonGroup>
#include <QtGui/QComboBox>
#include <QtGui/QDoubleSpinBox>
#include <QtGui/QHeaderView>
#include <QtGui/QLabel>
#include <QtGui/QMainWindow>
#include <QtGui/QMenuBar>
#include <QtGui/QPushButton>
#include <QtGui/QSpinBox>
#include <QtGui/QStatusBar>
#include <QtGui/QWidget>
#include "roiselecterlabel.h"
QT_BEGIN_NAMESPACE
class Ui_MainWindow
{
public:
    QAction *actionOpenFile;
    QAction *actionExit;
    QWidget *centralwidget;
    RoiSelecterLabel *labelImage;
    QSpinBox *spinBoxMinH;
    QLabel *label_2;
```

```
    QSpinBox *spinBoxMaxH;
    QLabel *label_3;
    QSpinBox *spinBoxMinS;
    QLabel *label_8;
    QLabel *label_9;
    QSpinBox *spinBoxMaxS;
    QLabel *label_12;
    QLabel *label_13;
    QSpinBox *spinBoxMinV;
    QSpinBox *spinBoxMaxV;
    QLabel *labelImage2;
    QLabel *label_14;
    QLabel *label_15;
    QSpinBox *spinBoxMaxR;
    QSpinBox *spinBoxMinR;
    QLabel *label_16;
    QDoubleSpinBox *doubleSpinBoxMinFillness;
    QLabel *label_5;
    QComboBox *comboBoxSettingsInput;
    QPushButton *pushButtonStart;
    QPushButton *pushButtonStop;
    QMenuBar *menubar;
    QStatusBar *statusbar;
    void setupUi(QMainWindow *MainWindow)
    {
        if (MainWindow->objectName().isEmpty())
            MainWindow->setObjectName(QString::fromUtf8("MainWindow"));
        MainWindow->resize(1340, 440);
        QSizePolicy sizePolicy(QSizePolicy::Preferred,
QSizePolicy::Preferred);
        sizePolicy.setHorizontalStretch(0);
```

```
        sizePolicy.setVerticalStretch(0);
        sizePolicy.setHeightForWidth(MainWindow-
>sizePolicy().hasHeightForWidth());
        MainWindow->setSizePolicy(sizePolicy);
        MainWindow->setMinimumSize(QSize(0, 0));
        actionOpenFile = new QAction(MainWindow);
        actionOpenFile->setObjectName(QString::fromUtf8("actionOpenFile"));
        actionExit = new QAction(MainWindow);
        actionExit->setObjectName(QString::fromUtf8("actionExit"));
        centralwidget = new QWidget(MainWindow);
        centralwidget->setObjectName(QString::fromUtf8("centralwidget"));
        labelImage = new RoiSelecterLabel(centralwidget);
        labelImage->setObjectName(QString::fromUtf8("labelImage"));
        labelImage->setGeometry(QRect(250, 10, 431, 381));
        spinBoxMinH = new QSpinBox(centralwidget);
        spinBoxMinH->setObjectName(QString::fromUtf8("spinBoxMinH"));
        spinBoxMinH->setGeometry(QRect(80, 10, 42, 22));
        spinBoxMinH->setMaximum(255);
        label_2 = new QLabel(centralwidget);
        label_2->setObjectName(QString::fromUtf8("label_2"));
        label_2->setGeometry(QRect(20, 10, 41, 21));
        QFont font;
        font.setPointSize(12);
        label_2->setFont(font);
        spinBoxMaxH = new QSpinBox(centralwidget);
        spinBoxMaxH->setObjectName(QString::fromUtf8("spinBoxMaxH"));
        spinBoxMaxH->setGeometry(QRect(190, 10, 42, 22));
        spinBoxMaxH->setMaximum(255);
        label_3 = new QLabel(centralwidget);
        label_3->setObjectName(QString::fromUtf8("label_3"));
        label_3->setGeometry(QRect(130, 10, 51, 21));
```

```
label_3->setFont(font);
spinBoxMinS = new QSpinBox(centralwidget);
spinBoxMinS->setObjectName(QString::fromUtf8("spinBoxMinS"));
spinBoxMinS->setGeometry(QRect(80, 40, 42, 22));
spinBoxMinS->setMaximum(255);
label_8 = new QLabel(centralwidget);
label_8->setObjectName(QString::fromUtf8("label_8"));
label_8->setGeometry(QRect(130, 40, 51, 21));
label_8->setFont(font);
label_9 = new QLabel(centralwidget);
label_9->setObjectName(QString::fromUtf8("label_9"));
label_9->setGeometry(QRect(20, 40, 41, 21));
label_9->setFont(font);
spinBoxMaxS = new QSpinBox(centralwidget);
spinBoxMaxS->setObjectName(QString::fromUtf8("spinBoxMaxS"));
spinBoxMaxS->setGeometry(QRect(190, 40, 42, 22));
spinBoxMaxS->setMaximum(255);
label_12 = new QLabel(centralwidget);
label_12->setObjectName(QString::fromUtf8("label_12"));
label_12->setGeometry(QRect(20, 70, 41, 21));
label_12->setFont(font);
label_13 = new QLabel(centralwidget);
label_13->setObjectName(QString::fromUtf8("label_13"));
label_13->setGeometry(QRect(130, 70, 51, 21));
label_13->setFont(font);
spinBoxMinV = new QSpinBox(centralwidget);
spinBoxMinV->setObjectName(QString::fromUtf8("spinBoxMinV"));
spinBoxMinV->setGeometry(QRect(80, 70, 42, 22));
spinBoxMinV->setMaximum(255);
spinBoxMaxV = new QSpinBox(centralwidget);
spinBoxMaxV->setObjectName(QString::fromUtf8("spinBoxMaxV"));
```

```
        spinBoxMaxV->setGeometry(QRect(190, 70, 42, 22));
        spinBoxMaxV->setMaximum(255);
        labelImage2 = new QLabel(centralwidget);
        labelImage2->setObjectName(QString::fromUtf8("labelImage2"));
        labelImage2->setGeometry(QRect(700, 10, 431, 381));
        label_14 = new QLabel(centralwidget);
        label_14->setObjectName(QString::fromUtf8("label_14"));
        label_14->setGeometry(QRect(130, 100, 51, 21));
        label_14->setFont(font);
        label_15 = new QLabel(centralwidget);
        label_15->setObjectName(QString::fromUtf8("label_15"));
        label_15->setGeometry(QRect(20, 100, 41, 21));
        label_15->setFont(font);
        spinBoxMaxR = new QSpinBox(centralwidget);
        spinBoxMaxR->setObjectName(QString::fromUtf8("spinBoxMaxR"));
        spinBoxMaxR->setGeometry(QRect(190, 100, 42, 22));
        spinBoxMaxR->setMaximum(255);
        spinBoxMinR = new QSpinBox(centralwidget);
        spinBoxMinR->setObjectName(QString::fromUtf8("spinBoxMinR"));
        spinBoxMinR->setGeometry(QRect(80, 100, 42, 22));
        spinBoxMinR->setMaximum(255);
        label_16 = new QLabel(centralwidget);
        label_16->setObjectName(QString::fromUtf8("label_16"));
        label_16->setGeometry(QRect(20, 130, 91, 21));
        label_16->setFont(font);
        doubleSpinBoxMinFillness = new QDoubleSpinBox(centralwidget);
        doubleSpinBoxMinFillness-
>setObjectName(QString::fromUtf8("doubleSpinBoxMinFillness"));
        doubleSpinBoxMinFillness->setGeometry(QRect(120, 130, 111, 22));
        label_5 = new QLabel(centralwidget);
        label_5->setObjectName(QString::fromUtf8("label_5"));
```

```
        label_5->setGeometry(QRect(20, 160, 211, 21));
        QFont font1;
        font1.setPointSize(13);
        label_5->setFont(font1);
        comboBoxSettingsInput = new QComboBox(centralwidget);
        comboBoxSettingsInput->setObjectName(QString::fromUtf8("comboBoxSettingsInput"));
        comboBoxSettingsInput->setGeometry(QRect(90, 160, 141, 22));
        pushButtonStart = new QPushButton(centralwidget);
        pushButtonStart->setObjectName(QString::fromUtf8("pushButtonStart"));
        pushButtonStart->setGeometry(QRect(20, 190, 91, 23));
        pushButtonStop = new QPushButton(centralwidget);
        pushButtonStop->setObjectName(QString::fromUtf8("pushButtonStop"));
        pushButtonStop->setGeometry(QRect(130, 190, 91, 23));
        MainWindow->setCentralWidget(centralwidget);
        menubar = new QMenuBar(MainWindow);
        menubar->setObjectName(QString::fromUtf8("menubar"));
        menubar->setGeometry(QRect(0, 0, 1340, 21));
        MainWindow->setMenuBar(menubar);
        statusbar = new QStatusBar(MainWindow);
        statusbar->setObjectName(QString::fromUtf8("statusbar"));
        MainWindow->setStatusBar(statusbar);
        retranslateUi(MainWindow);
        QMetaObject::connectSlotsByName(MainWindow);
    } // setupUi
    void retranslateUi(QMainWindow *MainWindow)
    {
        MainWindow->setWindowTitle(QApplication::translate("MainWindow", "Arif Emre", 0, QApplication::UnicodeUTF8));
        actionOpenFile->setText(QApplication::translate("MainWindow", "Dosya A\303\247", 0, QApplication::UnicodeUTF8));
        actionExit->setText(QApplication::translate("MainWindow",
```

```
"\303\207\304\261k\304\261\305\237", 0, QApplication::UnicodeUTF8));

        labelImage->setText(QApplication::translate("MainWindow", "TextLabel",
0, QApplication::UnicodeUTF8));

        label_2->setText(QApplication::translate("MainWindow", "Min H", 0,
QApplication::UnicodeUTF8));

        label_3->setText(QApplication::translate("MainWindow", "Max H", 0,
QApplication::UnicodeUTF8));

        label_8->setText(QApplication::translate("MainWindow", "Max S", 0,
QApplication::UnicodeUTF8));

        label_9->setText(QApplication::translate("MainWindow", "Min S", 0,
QApplication::UnicodeUTF8));

        label_12->setText(QApplication::translate("MainWindow", "Min V", 0,
QApplication::UnicodeUTF8));

        label_13->setText(QApplication::translate("MainWindow", "Max V", 0,
QApplication::UnicodeUTF8));

        labelImage2->setText(QApplication::translate("MainWindow",
"TextLabel", 0, QApplication::UnicodeUTF8));

        label_14->setText(QApplication::translate("MainWindow", "Max R", 0,
QApplication::UnicodeUTF8));

        label_15->setText(QApplication::translate("MainWindow", "Min R", 0,
QApplication::UnicodeUTF8));

        label_16->setText(QApplication::translate("MainWindow", "Min
Fillness", 0, QApplication::UnicodeUTF8));

        label_5->setText(QApplication::translate("MainWindow", "Camera", 0,
QApplication::UnicodeUTF8));

        comboBoxSettingsInput->clear();

        comboBoxSettingsInput->insertItems(0, QStringList()

         << QApplication::translate("MainWindow", "Webcam 1", 0,
QApplication::UnicodeUTF8)

         << QApplication::translate("MainWindow", "Webcam 2", 0,
QApplication::UnicodeUTF8)

         << QApplication::translate("MainWindow", "Webcam 3", 0,
QApplication::UnicodeUTF8)

        );

        pushButtonStart->setText(QApplication::translate("MainWindow",
"Start", 0, QApplication::UnicodeUTF8));

        pushButtonStop->setText(QApplication::translate("MainWindow", "Stop",
0, QApplication::UnicodeUTF8));
```

```
    } // retranslateUi

};

namespace Ui {

    class MainWindow: public Ui_MainWindow {};

} // namespace Ui

QT_END_NAMESPACE

#endif // UI_GUI_H
```

```
#include "gui.h"
#include <QTimer>
#include <QFileDialog>
#include "opencvThread.h"
#include <iostream>
#include "objecttrack.h"
#include "parameters.h"
#include <QString>
#include <QThread>
#include "TrackParameters.h"

Gui::Gui(QWidget *parent, Qt::WindowFlags flags)
    : QMainWindow(parent, flags)
{
    m_precess = NULL;
    m_tracker = NULL;
    qRegisterMetaType<result>("result");
    ui.setupUi(this);
    m_settings.LoadConfiguration();
    setGuiParams("Default");
    connect(ui.spinBoxMinH, SIGNAL(valueChanged(int)), this,
SLOT(resetparams()));
    connect(ui.spinBoxMaxH, SIGNAL(valueChanged(int)), this,
SLOT(resetparams()));
    connect(ui.spinBoxMinS, SIGNAL(valueChanged(int)), this,
SLOT(resetparams()));
    connect(ui.spinBoxMaxS, SIGNAL(valueChanged(int)), this,
SLOT(resetparams()));
    connect(ui.spinBoxMinV, SIGNAL(valueChanged(int)), this,
SLOT(resetparams()));
    connect(ui.spinBoxMaxV, SIGNAL(valueChanged(int)), this,
SLOT(resetparams()));
    connect(ui.spinBoxMaxR, SIGNAL(valueChanged(int)), this,
SLOT(resetparams()));
    connect(ui.spinBoxMinR, SIGNAL(valueChanged(int)), this,
SLOT(resetparams()));
    connect(ui.doubleSpinBoxMinFillness, SIGNAL(valueChanged(double)), this,
SLOT(resetparams()));
    connect(ui.comboBoxSettingsInput, SIGNAL(currentIndexChanged(QString)),
this, SLOT(currentInputChangedg(QString)));
    connect(ui.pushButtonStart, SIGNAL(clicked()), this,
SLOT(startSerialPort()));
    connect(ui.pushButtonStop, SIGNAL(clicked()), this,
SLOT(stopSerialPort()));
```

```
    readParamsFromGui(m_settings.m_appSet);
    m_tracker = new ObjectTracker(m_settings.m_appSet);
    m_precess = new ProcessThread(m_tracker);
    m_serialThread = new SerialPortThread(m_tracker);
    connect(m_precess, SIGNAL(resultReady(void*)), this,
SLOT(handleResults(void*)));
    m_timer = new QTimer(this);
    connect(m_timer, SIGNAL(timeout()), this, SLOT(setLabelImages()));
    m_timer->start(25);
    m_precess->start();
    m_serialThread->start();
    connect(ui.labelImage, SIGNAL(ImageReady()), this, SLOT(imageReady()));
}
void Gui::startWorkInAThread()
{
}
Gui::~Gui()
{
}
void Gui::imageReady()
{
    TrackParameters paramCalculator;
    paramCalculator.Calculate();
    m_settings.m_appSet.iLowH           =           paramCalculator.iLowH
       ;
    m_settings.m_appSet.iHighH    =     paramCalculator.iHighH     ;
    m_settings.m_appSet.iLowS           =     paramCalculator.iLowS      ;
    m_settings.m_appSet.iHighS    =     paramCalculator.iHighS     ;
    m_settings.m_appSet.iLowV           =     paramCalculator.iLowV      ;
    m_settings.m_appSet.iHighV    =     paramCalculator.iHighV     ;
    m_settings.SyncConfiguration();
    setGuiParams("Default");
    ui.spinBoxMinH->setValue(paramCalculator.iLowH);
    ui.spinBoxMaxH->setValue(paramCalculator.iHighH);
    ui.spinBoxMinS->setValue(paramCalculator.iLowS);
    ui.spinBoxMaxS->setValue(paramCalculator.iHighS);
    ui.spinBoxMinV->setValue(paramCalculator.iLowV);
    ui.spinBoxMaxV->setValue(paramCalculator.iHighV);
    resetparams();
}
void Gui::setLabelImages()
{
    QImage temp;
    QImage temp2;
    m_precess->getImages(temp,temp2);
    QImage scaled = temp.scaled(ui.labelImage->size());
    QImage scaled2 = temp2.scaled(ui.labelImage2->size());
    ui.labelImage->setPixmap(QPixmap::fromImage(scaled));
    ui.labelImage2->setPixmap(QPixmap::fromImage(scaled2));
    repaint();
}
void Gui::handleResults(void *img1)
{
    result *img((result*)img1);
    QImage temp = img->getImage(0);
    QImage scaled = temp.scaled(ui.labelImage->size());
    QImage temp2 = img->getImage(1);
    QImage scaled2 = temp2.scaled(ui.labelImage2->size());
    ui.labelImage->setPixmap(QPixmap::fromImage(scaled));
    ui.labelImage2->setPixmap(QPixmap::fromImage(scaled2));
    repaint();
}
```

```
void Gui::resetparams()
{
    readParamsFromGui(m_settings.m_appSet);
    m_tracker->setSettings(m_settings.m_appSet);
    m_settings.SyncConfiguration();
}
void Gui::setGuiParams(QString name)
{
    m_settings.LoadConfiguration(name);
    ui.spinBoxMinH->setValue(m_settings.m_appSet.iLowH);
    ui.spinBoxMaxH->setValue(m_settings.m_appSet.iHighH);
    ui.spinBoxMinS->setValue(m_settings.m_appSet.iLowS);
    ui.spinBoxMaxS->setValue(m_settings.m_appSet.iHighS);
    ui.spinBoxMinV->setValue(m_settings.m_appSet.iLowV);
    ui.spinBoxMaxV->setValue(m_settings.m_appSet.iHighV);
    ui.spinBoxMinR->setValue(m_settings.m_appSet.m_minRadius);
    ui.spinBoxMaxR->setValue(m_settings.m_appSet.m_maxRadius);
    ui.doubleSpinBoxMinFillness->setValue(m_settings.m_appSet.m_minFillness);
    ui.comboBoxSettingsInput->setCurrentIndex(m_settings.m_appSet.m_input);
    m_settings.SyncConfiguration();
}
void Gui::currentInputChangedg(const QString & text)
{
    QString t = text;
    QString i = t.replace("Webcam ","");
    m_settings.m_appSet.m_input  =i.toInt()-1;
    readParamsFromGui(m_settings.m_appSet);
    m_settings.SyncConfiguration();
    m_precess->setExit();
    m_serialThread->setExit();
    m_timer->stop();
    Sleep(100);
    delete m_timer;
    m_timer = new QTimer(this);
    connect(m_timer, SIGNAL(timeout()), this, SLOT(setLabelImages()));
    m_timer->start(25);
    delete m_serialThread;
    delete m_tracker;
    m_tracker = new ObjectTracker(m_settings.m_appSet);
    m_precess = new ProcessThread(m_tracker);
    m_serialThread = new SerialPortThread(m_tracker);
    m_precess->start();
    m_serialThread->start();
}
void Gui::readParamsFromGui(parameters &params)
{
    params.iLowH = ui.spinBoxMinH->value();
    params.iHighH = ui.spinBoxMaxH->value();
    params.iLowS = ui.spinBoxMinS->value();
    params.iHighS = ui.spinBoxMaxS->value();
    params.iLowV = ui.spinBoxMinV->value();
    params.iHighV = ui.spinBoxMaxV->value();
    params.m_minRadius =ui.spinBoxMinR->value();
    params.m_maxRadius = ui.spinBoxMaxR->value();
    params.m_minFillness = ui.doubleSpinBoxMinFillness->value();
       params.m_input = ui.comboBoxSettingsInput->currentIndex();
}
void Gui::stopSerialPort()
{
    m_serialThread->setPause(true);
}
```

```
void Gui::startSerialPort()
{
    m_serialThread->setPause(false);
}
```

2.5.4. Parametrelerin tutulması

Takip edilecek görüntünün Hue (Renk Özü), Saturation (Doygunluk) ve Value (Parlaklık) değerlerine tutulduğu bölümdür. Burada HSV değerlerinin en yüksek ve en düşük değerlerinin ayarları yapılır. Sistemin renk gürültüsünden daha az etkilenmesi için ve daha kararlı çalışabilmesi için takip edilen nesnenin doluluk oranı ayarlaması yapılabilmektedir. Ayrıca takip edilecek nesnenin bulunduğu bölgedeki durumuna göre en yüksek ve en düşük yarıçap değerleri de ayarlanabilmektedir. Nesnenin fiziksel bilgilerinin ayarlanılmasını sağlayan C++ kodları aşağıda verilmiştir.

```
#pragma once

#include <iostream>

#include <QString>

class parameters

{

public:

    parameters();

    ~parameters();

    int iLowH;

    int iHighH;

    int iLowS;

    int iHighS;

    int iLowV;

    int iHighV;

    int m_minRadius;

    int m_maxRadius;

    double m_minFillness;

    int m_method;

    int m_input;

    QString m_fileName;
```

```
    void SetDefault();

};

#include "parameters.h"

parameters::parameters()

{
    SetDefault();
}

parameters::~parameters()
{
}

void parameters::SetDefault()
{
    iLowH = 38;
    iHighH=75;
    iLowS =100;
    iHighS =255;
    iLowV =100;
    iHighV =255;
    m_minRadius =0;
    m_maxRadius =500;
    m_minFillness =0.0;
    m_method =0;
    m_input =0;
    m_fileName =QString("./a.mp4");
}
```

Takip edilecek nesnenin parametlerinin tutultuğu sınıfı oluşturan C++ kodları aşağıda verilmiştir.

```
#pragma once

class TrackParameters

{

public:

    TrackParameters(void);

    ~TrackParameters(void);

    void Calculate();

    int iLowH;

    int iHighH;

    int iLowS;

    int iHighS;

    int iLowV;

    int iHighV;
```

```
};

#include "TrackParameters.h"
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <iostream>
#include "opencv2/imgproc/imgproc.hpp"
using namespace cv;

TrackParameters::TrackParameters(void)
{
}

TrackParameters::~TrackParameters(void)
{
}
void TrackParameters::Calculate()
{
    Mat image;
    image = imread("track.jpg", CV_LOAD_IMAGE_COLOR);
    if(! image.data )                              // Check for invalid input
    {
        std::cout <<  "Could not open or find the image" << std::endl ;
    }
    Mat imgHSV;
    cvtColor(image, imgHSV, COLOR_BGR2HSV); //Convert the captured frame from
BGR to HSV
    Scalar tempVal = mean( imgHSV );
    int meanV = tempVal.val[2];
    int meanS = tempVal.val[1];
    int meanH = tempVal.val[0];
    iLowH  = meanH-10;
    iHighH = meanH+15;
    iLowS   =meanS-30;
    iHighS  =meanS+30;
    iLowV = meanV-30;
    iHighV=   meanV+30;
}
```

2.5.5. Açı değerlerinin hesaplanması

RC Servo motorlar açı değerlerine göre çalıştıkları için takip edilecek nesnenin değişen koordinat bilgilerinin açısal değerlere dönüştürülmesi gerekmektedir. RC Servo motora USB Port üzerinden gönderilecek açı değerlerini hesaplayan C++ kodları aşağıda verilmiştir

```
#pragma once
class PositionCalculator
{
public:
    PositionCalculator(void);
    ~PositionCalculator(void);
    static bool calculate(int centerX,int centerY,int Width,int height,float
&leftPosition,float &upPosition);
    static bool calculate2(int centerX,int centerY,int Width,int height,float
&leftPosition,float &upPosition);
```

```
    static bool calculate3(int centerX,int centerY,int Width,int height,float
&leftPosition,float &upPosition);
};
#include "PositionCalculator.h"
#include <iostream>

static  float m_leftPosition =0.5;
static  float m_upPosition =0.5;
static int c =0;
static int lastX  =320;

PositionCalculator::PositionCalculator(void)
{
}

PositionCalculator::~PositionCalculator(void)
{
}

bool PositionCalculator::calculate( int centerX,int centerY,int Width,int
height,float &leftPosition,float &upPosition )
{
    upPosition = 0.5;
    if ( /*(c % 10) == 0*/1)
    {
        if (centerX > 340  )
        {
            //leftPosition = m_leftPosition +0.01*(abs(centerX-320)/60);
            leftPosition = m_leftPosition +(0.01 +( abs(centerX-
320)/1280.0)/5.0 );
            leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
            leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
            m_leftPosition =leftPosition;
            std::cout <<leftPosition <<"\n";
        }
        else if (centerX < 300  )
        {
            //leftPosition = m_leftPosition -0.01*(abs(centerX-320)/60);
            leftPosition = m_leftPosition -(0.01 + abs(centerX-320)/1280.0/5.0
);
            leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
            leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
            m_leftPosition =leftPosition;
            std::cout <<leftPosition <<"\n";
        } else {
            std::cout <<"ortada"<<"\n";
            leftPosition = m_leftPosition;
        }
        lastX = centerX;
        return true;
    }
    c= c++ > 100 ? 0 :c;
    return false;
}



bool PositionCalculator::calculate2( int centerX,int centerY,int Width,int
height,float &leftPosition,float &upPosition )
{
    upPosition = 0.5;
    if (     centerX > 520  && centerX <640  )
```

```
    {
        leftPosition = m_leftPosition + 0.02;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else      if (     centerX >= 430  && centerX <520 )
    {
        leftPosition = m_leftPosition + 0.01;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else      if (     centerX >= 370  && centerX <430 )
    {
        leftPosition = m_leftPosition + 0.005;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else      if (     centerX >= 340  && centerX <370 )
    {
        leftPosition = m_leftPosition + 0.0025;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else if (    centerX >= 270  && centerX <300  )
    {
        leftPosition = m_leftPosition - 0.0025;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else      if (     centerX > 210  && centerX <270 )
    {
        leftPosition = m_leftPosition - 0.005;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else if (     centerX >= 120  && centerX <210 )
    {
        leftPosition = m_leftPosition - 0.01;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else if (     centerX >= 0  && centerX <120 )
    {
        leftPosition = m_leftPosition - 0.02;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else
    {
        //orta
        leftPosition = m_leftPosition;
    }
    m_leftPosition = leftPosition;
    return true;
}

bool left( int centerX,int centerY,int Width,int height,float
&leftPosition,float &upPosition )
{
    if (     centerX > 520  && centerX <640  )
    {
        leftPosition = m_leftPosition + 0.05;
```

```
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else      if (     centerX >= 430  && centerX <520 )
    {
        leftPosition = m_leftPosition + 0.03;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else      if (     centerX >= 370  && centerX <430 )
    {
        leftPosition = m_leftPosition + 0.02;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else      if (     centerX >= 340  && centerX <370 )
    {
        leftPosition = m_leftPosition + 0.01;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else if (    centerX >= 270  && centerX <300  )
    {
        leftPosition = m_leftPosition - 0.01;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else      if (     centerX > 210  && centerX <270 )
    {
        leftPosition = m_leftPosition - 0.02;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else if (     centerX >= 120  && centerX <210 )
    {
        leftPosition = m_leftPosition - 0.03;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else if (     centerX >= 0  && centerX <120 )
    {
        leftPosition = m_leftPosition - 0.05;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else
    {
        //orta
        leftPosition = m_leftPosition;
    }
    m_leftPosition = leftPosition;
    return true;
}
bool up( int centerX,int centerY,int Width,int height,float
&leftPosition,float &upPosition )
{
    if (      centerY >= 384  && centerY <480    )
    {
        upPosition = m_upPosition -0.07;
        upPosition= upPosition > 0.9 ? 0.9 :upPosition;
        upPosition= upPosition < 0.001 ? 0.0 :upPosition;
    }
```

```
    else      if (   centerY >=312  && centerY < 384 )
    {
        upPosition = m_upPosition  -0.05;
        upPosition= upPosition > 0.9 ? 0.9 :upPosition;
        upPosition= upPosition < 0.001 ? 0.0 :upPosition;
    }
    else      if (   centerY >=264  && centerY < 312 )
    {
        upPosition = m_upPosition  -0.01;
        upPosition= upPosition > 0.9 ? 0.9 :upPosition;
        upPosition= upPosition < 0.001 ? 0.0 :upPosition;
    }
    else      if (   centerY >=240  && centerY < 264 )
    {
        upPosition = m_upPosition ;
        upPosition= upPosition > 0.9 ? 0.9 :upPosition;
        upPosition= upPosition < 0.001 ? 0.0 :upPosition;
    }
    //
    else if (      centerY >= 216  && centerY <240    )
    {
        upPosition = m_upPosition ;
        upPosition= upPosition > 0.9? 0.9 :upPosition;
        upPosition= upPosition < 0.001 ? 0.0 :upPosition;
    }
    else      if (   centerY >=168  && centerY < 216 )
    {
        upPosition = m_upPosition  +0.01;
        upPosition= upPosition > 0.9 ? 0.9 :upPosition;
        upPosition= upPosition < 0.001 ? 0.0  :upPosition;
    }
    else      if (   centerY >=96  && centerY < 168 )
    {
        upPosition = m_upPosition  +0.05;
        upPosition= upPosition > 0.9 ? 0.9 :upPosition;
        upPosition= upPosition < 0.001 ? 0.0 :upPosition;
    }
    else      if (   centerY >=0  && centerY < 96 )
    {
        upPosition = m_upPosition  +0.07;
        upPosition= upPosition > 0.9 ? 0.9 :upPosition;
        upPosition= upPosition < 0.001 ? 0.0 :upPosition;
    }
    else
    {
        //orta
        upPosition = m_upPosition;
    }
    //std::cout <<m_upPosition<<"\n";
    m_upPosition = upPosition;
    return true;
}

bool PositionCalculator::calculate3( int centerX,int centerY,int Width,int
height,float &leftPosition,float &upPosition )
{
    upPosition = 0.5;
    bool l = left(centerX,centerY,Width,height,leftPosition,upPosition);
    bool u = up(centerX,centerY,Width,height,leftPosition,upPosition);
    return l && u;
    ////////////////////////////////////////////////////////////////////////
    if (     centerX > 520  && centerX <640  )
```

```
    {
        leftPosition = m_leftPosition + 0.02;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else      if (     centerX >= 430  && centerX <520 )
    {
        leftPosition = m_leftPosition + 0.01;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else      if (     centerX >= 370  && centerX <430 )
    {
        leftPosition = m_leftPosition + 0.005;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else      if (     centerX >= 340  && centerX <370 )
    {
        leftPosition = m_leftPosition + 0.0025;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else if (    centerX >= 270  && centerX <300  )
    {
        leftPosition = m_leftPosition - 0.0025;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else      if (     centerX > 210  && centerX <270 )
    {
        leftPosition = m_leftPosition - 0.005;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else if (     centerX >= 120  && centerX <210 )
    {
        leftPosition = m_leftPosition - 0.01;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else if (     centerX >= 0  && centerX <120 )
    {
        leftPosition = m_leftPosition - 0.02;
        leftPosition= leftPosition > 1.0 ? 1.0 :leftPosition;
        leftPosition= leftPosition < 0.01 ? 0.0 :leftPosition;
    }
    else
    {
        //orta
        leftPosition = m_leftPosition;
    }
    m_leftPosition = leftPosition;
    return true;
}
```

2.5.6. Takip edilecek nesnenin seçilmesi

Takip edilecek nesnelerin renk bilgileri elle girilerek ayarlanılabildiği gibi nesnenin renk bilgileri seçim yapılarak da ayarlanılabilir. Bunun için fare ile takip edilecek nesnenin üzerine gelinir. Daha sonra sol tuşa basılı tutularak nesne seçilir. Son olarak sağ tuşa tıklayarak ayarla sekmesi seçilir. Böylece takip edilecek nesnenin renk bilgileri ayarlanmış olur. Seçilen nesnenin renk bilgilerini hesaplayan C++ kodları aşağıda verilmiştir.

```
#ifndef WIDGET_H
#define WIDGET_H
#include <QLabel>
#include <QMenu>
class RoiSelecterLabel : public QLabel
{
    Q_OBJECT
public:
    RoiSelecterLabel(QWidget *parent = 0);
    ~RoiSelecterLabel();
protected:
    void paintEvent(QPaintEvent *e);
    void mousePressEvent(QMouseEvent *e);
    void mouseMoveEvent(QMouseEvent *e);
    void mouseReleaseEvent(QMouseEvent *e);
private:
    bool m_selectionStarted;
    QRect m_selectionRect;
    QMenu m_contextMenu;
private slots:
    void GetValuesSlot();
signals:
    void ImageReady();
```

```
};

#endif // WIDGET_H
```

```
#include "roiselecterlabel.h"
#include <QPainter>
#include <QMouseEvent>
#include <QDebug>
#include <QAction>
#include <QFileDialog>

RoiSelecterLabel::RoiSelecterLabel(QWidget *parent)
    : QLabel(parent)
{
    m_selectionStarted=false;
    QAction *saveAction=m_contextMenu.addAction("Ayarla");
    connect(saveAction,SIGNAL(triggered()),this,SLOT(GetValuesSlot()));
}

RoiSelecterLabel::~RoiSelecterLabel()
{
}

void RoiSelecterLabel::paintEvent(QPaintEvent *e)
{
    QLabel::paintEvent(e);
    QPainter painter(this);
    painter.setPen(QPen(QBrush(QColor(0,0,0,180)),1,Qt::DashLine));
    painter.setBrush(QBrush(QColor(255,255,255,120)));
//painter.drawRect(selectionRect);
    painter.drawEllipse(m_selectionRect);
}

void RoiSelecterLabel::mousePressEvent(QMouseEvent *e)
{
    if (e->button()==Qt::RightButton)
    {
        if (m_selectionRect.contains(e->pos())) m_contextMenu.exec(this-
>mapToGlobal(e->pos()));
    }
    else
    {
        m_selectionStarted=true;
        m_selectionRect.setTopLeft(e->pos());
        m_selectionRect.setBottomRight(e->pos());
    }
}

void RoiSelecterLabel::mouseMoveEvent(QMouseEvent *e)
{
    if (m_selectionStarted)
    {
        m_selectionRect.setBottomRight(e->pos());
        repaint();
    }
}

void RoiSelecterLabel::mouseReleaseEvent(QMouseEvent *e)
```

```
{
    m_selectionStarted=false;
}

void RoiSelecterLabel::GetValuesSlot()
{
    QString fileName("track.jpg");
    this->pixmap()->copy(m_selectionRect).save(fileName);
    emit ImageReady();
}
```

2.5.7. Konum bilgisinin seri portla iletilmesi

Windows işletim sistemi üzerinden seri haberleşme işlemlerinin yapıldığı C++ kodları aşağıda verilmiştir.

```
// Serial.h

#ifndef __SERIAL_H__

#define __SERIAL_H__

#ifdef WIN32

#define FC_DTRDSR       0x01

#define FC_RTSCTS       0x02

#define FC_XONXOFF      0x04

#define ASCII_BEL       0x07

#define ASCII_BS        0x08

#define ASCII_LF        0x0A

#define ASCII_CR        0x0D

#define ASCII_XON       0x11

#define ASCII_XOFF      0x13

#define NOMINMAX

#include "windows.h"

class CSerial

{

public:

    CSerial();

    ~CSerial();

    BOOL Open( int nPort = 2, int nBaud = 9600 );
```

```
    BOOL Close( void );


    int ReadData( void *, int );
    int SendData( const char *, int );
    int ReadDataWaiting( void );
    BOOL IsOpened( void ) {
        return( m_bOpened );
    }
protected:
    BOOL WriteCommByte( unsigned char );
    HANDLE m_hIDComDev;
    OVERLAPPED m_OverlappedRead, m_OverlappedWrite;
    BOOL m_bOpened;
};
#endif
#ifdef WIN32
typedef CSerial MySerial;
#else
#include "serialinterface.h"
typedef SerialInterface MySerial;
#endif
#endif
```

Linux işletim sisteminde seri haberleşme işlemlerinin yapıldığı C++ kodları aşağıda verilmiştir.

```
#ifndef SERIALINTERFACE_H
#define SERIALINTERFACE_H
class SerialInterface
{
public:
    SerialInterface() {};
```

```
    ~SerialInterface() {};
    bool Open( int nPort = 2, int nBaud = 9600 ) {
        return false;
    };
    bool Close( void ) {
        return true;
    };
    int ReadData( void *, int ) {
        return 0;
    };
    int SendData( const char *, int ) {
        return 0;
    };
    int ReadDataWaiting( void ) {
        return 0;
    };
    bool IsOpened( void ) {
        return( true );
    }
};
#endif // SERIALINTERFACE_H
```

Seri port işlemleri için açılan bir sınıf sayesinde düzenli bir şekilde seri haberleşme sağlanmaktadır. Bu sayede Mbed NXP LPC1768 Mikrodenetleyicisine gönderilecek değerler anında seri portlara iletilmiştir. Bu nesneyi oluşturmayı sağlayan C++ kodları aşağıda verilmiştir.

```
#pragma once
#include <QThread>
#include <QImage>
#include "result.h"
#include <QMetaType>
#include "objecttrack.h"
```

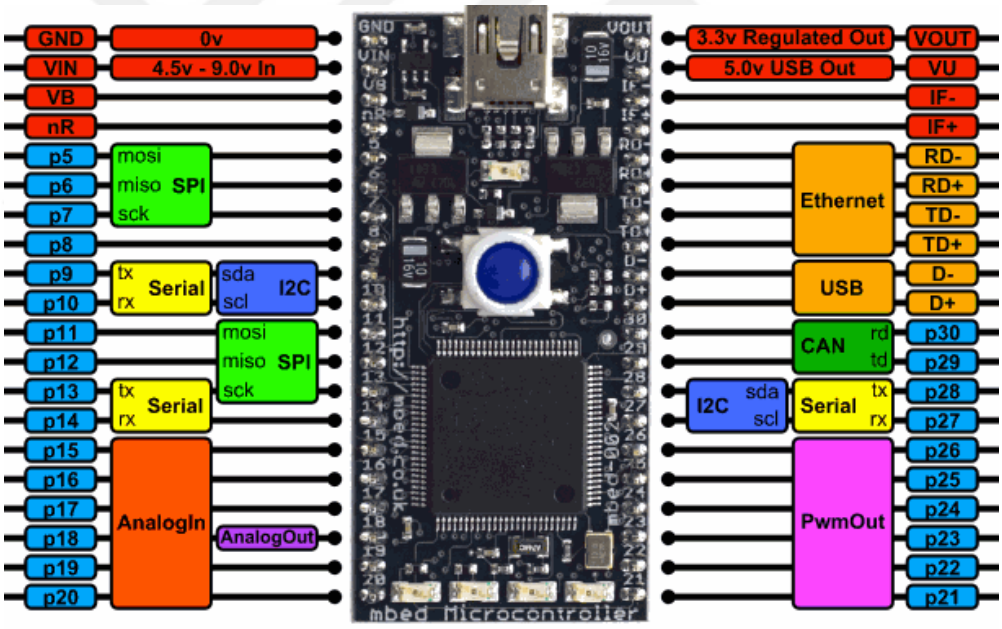
```
class SerialPortThread : public QThread

{

    Q_OBJECT

public:

    SerialPortThread(ObjectTracker *);

    void run();

    void setExit();

    void setPause(bool p);

    ObjectTracker *m_processoer;

    float m_l;

    float m_u;

private:

    bool m_exit;

    bool m_pause;

signals:

};

#include "serialPortThread.h"
#include "objecttrack.h"
#include <QImage>
#include <ctime>
#include "PositionCalculator.h"
void SerialPortThread::run()
{
    m_pause = false;
    MySerial serial;
    serial.Open(3, 19200);
    float leftPosition=0.0;
    float upPosition=0.0;
    char tx_line[50];
    float l=0.0;
    float u=0.0;

    while (m_exit)
    {
        if (m_processoer != NULL && serial.IsOpened() )
        {
            try
            {
                bool c = m_processoer->getPOsitions(l,u);
                if ( !m_pause && c &&
PositionCalculator::calculate3(l,u,640,480,leftPosition,upPosition))
                {
                    sprintf(tx_line,"%f %f %d
%d\r\n",leftPosition,upPosition,(int)l,(int)u);
                    std::cout<<" X = "<<(int)l <<" y = "<<(int)u<<"\n";
                    serial.SendData(tx_line,50);
```

```
                    msleep(50);
                }
            }
            catch (std::exception* e)
            {
                int as = 8;
            }
        }
    }
}

SerialPortThread::SerialPortThread(ObjectTracker *worker)
{
    m_exit = true;
    m_processoer = worker;
    m_l =0.5;
    m_u =0.5;
}

void SerialPortThread::setExit()
{
    m_exit = false;
}

void SerialPortThread::setPause(bool p)
{
    m_pause = p;
}
```

3. HAREKETLİ NESNENİN TAKİBİ

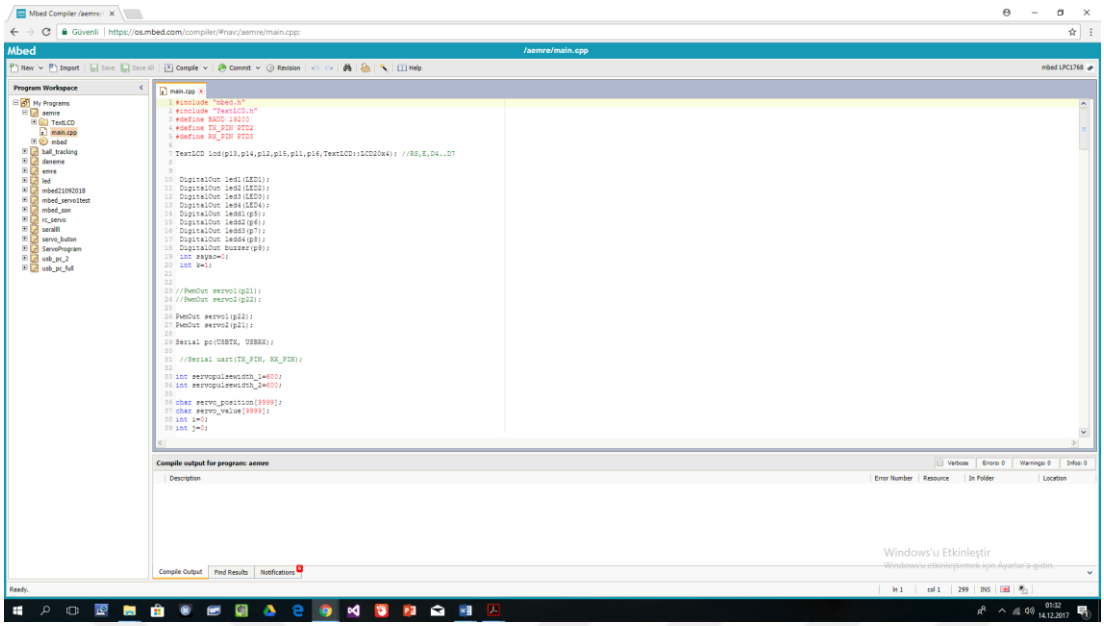
3.1. ARM Cortex-M Serisi Mbed NXP LPC1768 Mikrodenetleyici

Mbed Mikrodenetleyiciler hızlı uygulamalar için tasarlanmış ARM mikroişlemci geliştirme kartlarıdır. Kartın yapısı Şekil-3.1' de görülmektedir. Gömülü sistem özelliğine sahip olan bu mikrodenetleyiciyle çok sayıda uygulama yapılmaktadır. Gömülü sistem herhangi bir sistemin içinde yer alan ve o sisteme "akıllılık" özelliğini veren elektronik donanım ve yazılımdan oluşan bütünü ifade etmektedir. Burada sözü edilen yazılımlar, genel amaçlı yazılımlardan farklı olarak, kullanıcıyla direk değil dolaylı etkileşimde bulunan ve genellikle tek bir görevi yerine getiren yazılımlardır.



Şekil 3.1. ARM Cortex-M Serisi Mbed NXP LPC1768 Mikrodenetleyici

Mbed 32-bit ARM Cortex-M3 çekirdekli 96MHz işlemciden oluşur. İnternet üzerinden kullanılan bir mikrodenetleyici olduğundan derlemesi kolaydır. Programlama arayüzü Şekil 3.2' de gösterilmiştir.



Şekil 3.2. Mbed NXP LPC1768 Program yazma arayüzü

Mbed C ve C++ tabanlı kodlar tarafından yazıldığı için çok çeşitli uygulamalarda kolaylık sağlar. Üzerinde 40 adet pin bulunur. Flash bellek olarak da kullanılabilir. Analog giriş özelliğinden dolayı programda kullanılan basit kodlarla analog sinyal algılanabilir. Doğrudan Ethernet pinleri sayesinde internetin gerektiği uygulamalar yapılabilir. PWM çıkışlarını kullanarak R/C Servo motorun konumlandırılması yapılmaktadır. Ayrıca USB pinleri sayesinde hiçbir seri haberleşme devresi kullanılmadan bilgisayar ile haberleşme yapılmaktadır. Haberleşme esnasında dikkat edilmesi gereken konu makine dili ile kullanılan dilin farklı olmasıdır. Bu nedenle ASCII kod kullanımı gerekir. Örneğin klavyeden girilen “0” makine için ASCII kod olan 48’e eşittir. ASCII Kod tablosu Çizelge-1’ de gösterilmiştir. Servo motor kontrolünde istenilen açı değerinde motoru konumlandırabilmek için ASCII karşılıklarına göre kod yazmak gerekir. Bilgisayardan gönderilen bilgilerin doğruluğunu denetlemek için çeşitli hazır programlar bulunur. Bu programlar sayesinde örneğin klavyeden girilen bir değerin etkisini devre üzerinde görülebilir.

Çizelge 1. ASCII Kod Çizelge

İkilik sistem	Sekizlik sistem	Onluk sistem	On altılık sistem	Karakter
010 0000	040	32	20	sp
010 0001	041	33	21	!
010 0010	042	34	22	"
010 0011	043	35	23	#
010 0100	044	36	24	$
010 0101	045	37	25	%
010 0110	046	38	26	&
010 0111	047	39	27	'
010 1000	050	40	28	(
010 1001	051	41	29	)
010 1010	052	42	2A	*
010 1011	053	43	2B	+
010 1100	054	44	2C	,
010 1101	055	45	2D	-
010 1110	056	46	2E	.
010 1111	057	47	2F	/
011 0000	060	48	30	0
011 0001	061	49	31	1
011 0010	062	50	32	2
011 0011	063	51	33	3
011 0100	064	52	34	4
011 0101	065	53	35	5
011 0110	066	54	36	6
011 0111	067	55	37	7
011 1000	070	56	38	8
011 1001	071	57	39	9
011 1010	072	58	3A	[[:]]
011 1011	073	59	3B	;
011 1100	074	60	3C	[[<]]
011 1101	075	61	3D	=
011 1110	076	62	3E	[[>]]
011 1111	077	63	3F	?
100 0000	100	64	40	@
100 0001	101	65	41	A
100 0010	102	66	42	B
100 0011	103	67	43	C
100 0100	104	68	44	D
100 0101	105	69	45	E
100 0110	106	70	46	F
100 0111	107	71	47	G
100 1000	110	72	48	H
100 1001	111	73	49	I
100 1010	112	74	4A	J
100 1011	113	75	4B	K
100 1100	114	76	4C	L
100 1101	115	77	4D	M
100 1110	116	78	4E	N
100 1111	117	79	4F	O
101 0000	120	80	50	P
101 0001	121	81	51	Q
101 0010	122	82	52	R
101 0011	123	83	53	S
101 0100	124	84	54	T
101 0101	125	85	55	U
101 0110	126	86	56	V
101 0111	127	87	57	W
101 1000	130	88	58	X
101 1001	131	89	59	Y
101 1010	132	90	5A	Z
101 1011	133	91	5B	[
101 1100	134	92	5C	\
101 1101	135	93	5D	]
101 1110	136	94	5E	^
101 1111	137	95	5F	_
110 0000	140	96	60	`
110 0001	141	97	61	a
110 0010	142	98	62	b
110 0011	143	99	63	c
110 0100	144	100	64	d
110 0101	145	101	65	e
110 0110	146	102	66	f
110 0111	147	103	67	g
110 1000	150	104	68	h
110 1001	151	105	69	i
110 1010	152	106	6A	j
110 1011	153	107	6B	k
110 1100	154	108	6C	l
110 1101	155	109	6D	m
110 1110	156	110	6E	n
110 1111	157	111	6F	o
111 0000	160	112	70	p
111 0001	161	113	71	q
111 0010	162	114	72	r
111 0011	163	115	73	s
111 0100	164	116	74	t
111 0101	165	117	75	u
111 0110	166	118	76	v
111 0111	167	119	77	w
111 1000	170	120	78	x
111 1001	171	121	79	y
111 1010	172	122	7A	z
111 1011	173	123	7B	{
111 1100	174	124	7C	|
111 1101	175	125	7D	}
111 1110	176	126	7E	~

3.2. RC Servo Motor

Servo motorlarda DC motorların temeli kullanılır. Fakat buna ek olarak başka bileşenleri de vardır. Servo motorun bileşenleri aşağıda verilmiştir. R/C Servo motorun iç yapısının görüntüsü Şekil 3.3' te gösterilmektedir.

Şekil 3.3. RC-Servo Motor

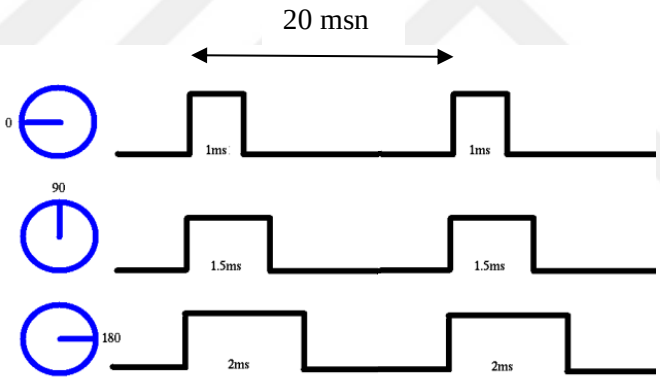
- DC motor
- Torku arttırmak için dişli sistemi
- Elektronik şaft (mil) pozisyon ve kontrol devresi

Servo motorlar şaftın kaç derece döndüğünü ve hangi hızda döndüğünü algılar ve girişine geri besleme verir. Böylece girişte istenen durum ile çıkış durumundan oluşan bir hata sinyali oluşur. Bu hata sinyaline göre servo motor şaftı döndürerek istenen pozisyon ve hızı sağlar. Bu algılama işlemi için bu servo motorlarda rotora takılı bir encoder ve pozisyon algılayıcı potansiyometre bulunur. Bu tipteki servo motorlarda encoder belirli sayıda boşluklardan oluşmaktadır. Mil döndükçe deliklerin sayısı pozisyon devresi tarafından sayılmaktadır. Servo motorun kaç tur atacağı dijital bilgi olarak servo motora bildirilir. Böylece pozisyon kontrol devresi kaç tur atacağını bilir ve o kadar tur atar.

R/C tipi servo motorlarda encoder yoktur. Bunun yerine şafta bağlanmış ve dönüşü algılayan potansiyometre vardır. Bu tip servo motorlar PWM (pulse width modulation) tekniğiyle çalışırlar. Kontrol ucuna gelen PWM sinyalinin görev çevrimine (duty cycle) göre belirli açılarda dönme yaparlar. Bu tip servo motorlara

R/C tip servo motor denmesinin nedeni, genelde radio frekansı ile uzaktan kontrol edilerek hobi amaçlı araba, uçak, helikopter v.b. yapılarında kullanılmasındandır. R/C ifadesi Radio Controlled kelimesinin baş harflerinden meydana gelmiştir. R/C Servo motorlar derece esasına göre dönerler. PWM sinyallerini yorumlayarak gerekli dönme derecelerini hesaplar ve o değer ölçüsünde dönüş yaparlar. Kontrol sinyali aynı kaldığı müddetçe konumlarını korurlar. R/C Servo motorlar genellikle 0° ile 180° arasında dönme gerçekleştirirler. Bu dönme dereceleri PWM sinyalinin görev çevriminin genişliğine göre değişir.

Çoğu ticari servo motorlarda dönmenin gerçekleşmesi için PWM sinyalinin periyodu 20ms olmalıdır. Yani R/C servo motorların kontrol sinyalinin frekansı 50 Hz olmalıdır. 0° ile 180° arası dönüş dereceleri ise PWM sinyalinin görev çevriminin (duty cycle) yaklaşık 1ms ile 2ms arasında değiştirilmesi ile elde edilir. R/C servo motorların görev çevrimine göre konumlandırılması Şekil 3.4 ' te gösterilmiştir.

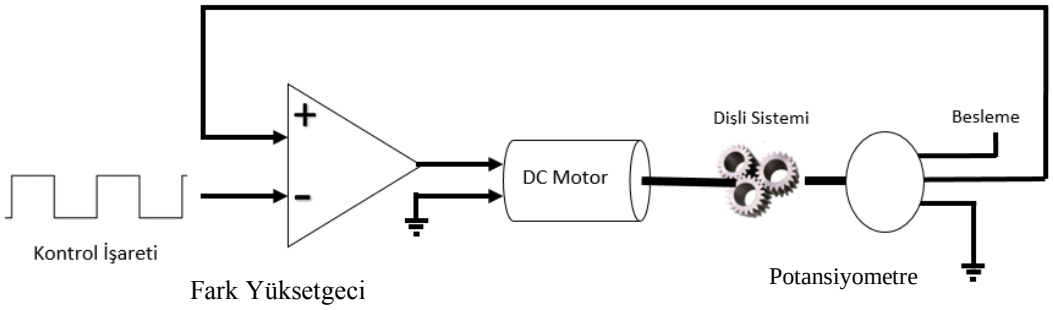


Şekil 3.4. RC-Servo Motor Görev Çevrimi

PWM sinyalindeki 1,5 ms'lik görev çevrimi motoru merkez konuma (0°), 1 ms'lik görev çevrimi motoru tam sol (90°sol) konuma, 2ms'lik görev çevrimi motoru tam sağ (90°sağ) konuma getirir. R/C Servo motorların besleme gerilimleri ise 4,8V ile 6V arasında olmalıdır. R/C Servo motorların üzerinde Tork güçleri de verilmektedir. Bu Tork gücü kg.cm cinsinden verilmektedir. Örneğin 6V besleme altında 3.7 kg.cm çıkış torkuna sahip bir R/C Servo motor, motor milini 1cm uzaklıkta bağlanan 3.7kg ağırlıktaki bir yükü kaldırabilir.(Kuvvet= Yük Ağırlığı x Kol Uzunluğu=3.7kg x 1 cm)

R/C Servo motorun iç yapısı bir DC motor, bir potansiyometre, bir yükselteç devresidir. Dişli çarklardan oluşmaktadır. Servo motora bir kontrol sinyali geldiğinde

motor istenilen pozisyon ile mevcut pozisyonu karşılaştırır. Eğer istenen pozisyon mevcut pozisyondan küçük ise servo motor mili sola döner, büyük ise servo motor mili sağa döner. RC Servo motorun blok diyagramı Şekil 3.5' te gösterilmiştir.



Şekil 3.5 R/C Servo Motor Blok Diyagramı

R/C Servo motorlar genellikle uzaktan kumandalı model uçaklar, helikopterler, gemi maketlerinde ve bazı robotik uygulamalarda kullanılmaktadır. Bu motorlar genelde merkezde, tam sağda ve tam solda olacak şekilde yönlendirilirler. Ara açı değerlerinin bu motorlarda çok hassas olarak elde edilmesi zordur. R/C Servoların dijital olanları da vardır. Dijital R/C Servo motorlar, normal servo motorlara göre daha küçük açı değerlerinde çalıştırılabilir. Bazı güç kaynaklarına fazla sayıda R/C servo motor bağlandığında bu motorların çalışması esnasında güç kaynağının geriliminde değişiklikler olmaktadır. Bu nedenle R/C servo motorun besleme gerilimi ile denetimini yapacak mikrokontrol veya mikroişlemcinin besleme gerilimi ayrı tutulmalıdır. R/C Servo motorun istenilen açı değerine getirilebilmesi için gerekli pozitif darbe süresi aşağıdaki denklemle hesaplanır.

$$t = 1 + \frac{\text{Açı}}{180}\ (msn) \tag{3.1}$$

3.3. Takip Mekanizması

Visual Studio program derleyicisinde OpenCV kütüphanesinden yararlanarak yazılan görüntü işleme algoritması sayesinde takip edilecek nesnenin 480x640 çözünürlükte ekran üzerindeki koordinat bilgileri USB Haberleşme portuyla ARM Cortex-M Serisi Mbed NXP LPC1768 Mikrodenetleyici' ne iletilmektedir. Takip edilecek nesnenin görüntü ekranından çıkmaması için nesnenin ağırlık merkezini 480x640 olarak ayarlanan ekranın merkezinde olmasını sağlayan bir algoritma geliştirilmiştir. İnternet

üzerinden Mbed Compiler arayüzünde yazılan USB Haberleşme ve RC-Servo motor kontrolüne ait kod aşağıda verilmiştir.

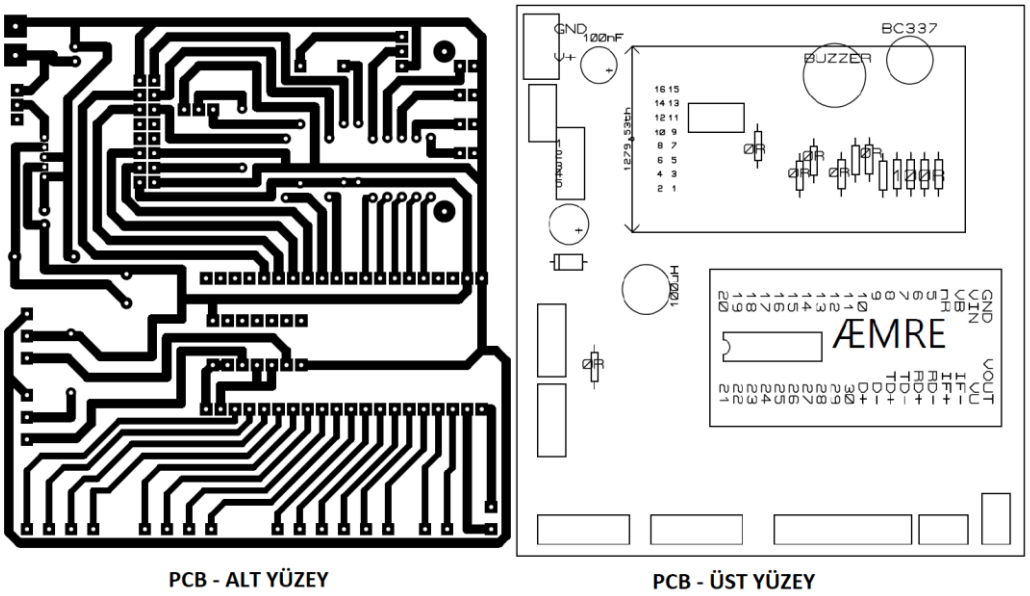
```
#include "mbed.h"
#include "Servo.h"
#include "TextLCD.h"
#define BAUD 19200
#define TX_PIN PTD2
#define RX_PIN PTD3
// Seri haberleşme için gerekli olan tanımlamalar
const int buffer_size = 255;
char tx_buffer[buffer_size+1];
char rx_buffer[buffer_size+1];
// Okuma-yazma işleminin yapıldığı bölüm
volatile int tx_in=0;
volatile int tx_out=0;
volatile int rx_in=0;
volatile int rx_out=0;
// Yazma ve okuma işlemleri için gerekli satırlar
char tx_line[80];
char rx_line[80];
void Rx_interrupt();
void read_line();
DigitalOut led1(LED1);
DigitalOut buzzer(p9);
TextLCD lcd(p13,p14,p12,p15,p11,p16,TextLCD::LCD20x4); //RS,E,D4..D7
Servo myservoUp(p21);
Servo myservoLeft(p22);//sag sol   aralık = 0.00087;
Serial pc(USBTX, USBRX);
int main() {
   buzzer=0;
   pc.baud(BAUD);
// Veri almak için seri bir kesme işlevi yapıldı.
   pc.attach(&Rx_interrupt);
```

```
    float rangeLeft = 0.00086;
    float rangeUp = 0.0005;
    myservoLeft.calibrate(rangeLeft, 45.0);
    myservoUp.calibrate(rangeUp, 45.0);
     lcd.cls();
     lcd.locate(0,0);
     lcd.printf("Starting");
     wait(1);
    float servo1=0;
    float servo2=0;
       while(1)
       {
          tx_in=0;
          tx_out=0;
          rx_in=0;
          rx_out=0;
          int x;
          int y;
          read_line();
          sscanf(rx_line,"%f %f %d %d",&servo1,&servo2,&x,&y);
          lcd.locate(0,0);
          lcd.cls();
          lcd.printf("x = %d\ny = %d",x,y);
          myservoLeft = 1-servo1;
          wait(0.01);
          myservoUp = servo2;
         wait(0.01);
    }
}
void read_line() {
    int i;
    i = 0;
    NVIC_DisableIRQ(UART1_IRQn);
    // Satır sonuna kadar okumayı sağlar.
```

```
   while ((i==0) || (rx_line[i-1] != '\r'))
{
      // Arabellekte bilgi yoksa bekler.
      if (rx_in == rx_out) {
         // Yeni  bilgi alınmasını sağlayan kesmedir.
         NVIC_EnableIRQ(UART1_IRQn);
         while (rx_in == rx_out) {
         }
         NVIC_DisableIRQ(UART1_IRQn);
      }
      rx_line[i] = rx_buffer[rx_out];
      i++;
      rx_out = (rx_out + 1) % buffer_size;
   }
   NVIC_EnableIRQ(UART1_IRQn);
   rx_line[i-1] = 0;
   return;
}
//Seri porttan veri okumak için gerekli olan kesmedir.
void Rx_interrupt()
{
   led1=1;
   // Birden fazla karakter okunmasını sağlayan döngü
   while ((pc.readable()) && (((rx_in + 1) % buffer_size) != rx_out))
{
      rx_buffer[rx_in] = pc.getc();
      rx_in = (rx_in + 1) % buffer_size;
   }
   led1=0;
   return;
}
```

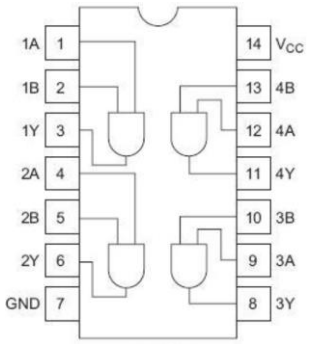
3.4. Devre Tasarımı

Görsel olarak elektronik devrelerin benzetimini yapabilen Proteus programı sayesinde hazırlanan devrenin PCB baskı devre çizimi Şekil 3.6' da verilmiştir. Devrede ARM tabanlı mikroişlemci Mbed – LPC1768 Geliştirme Kartı kullanılmıştır. Mbed – LPC1768 denetleyicisinin PWM(Pulse-Width Modulation) Port çıkışlarından p21 ve p22 ye iki adet RC-Servo motor bağlantısı yapılmıştır. PWM portlarının kullanılmasının nedeni RC-Servo motorların dönüş açılarının analog sinyallere göre değişkenlik göstermesidir.



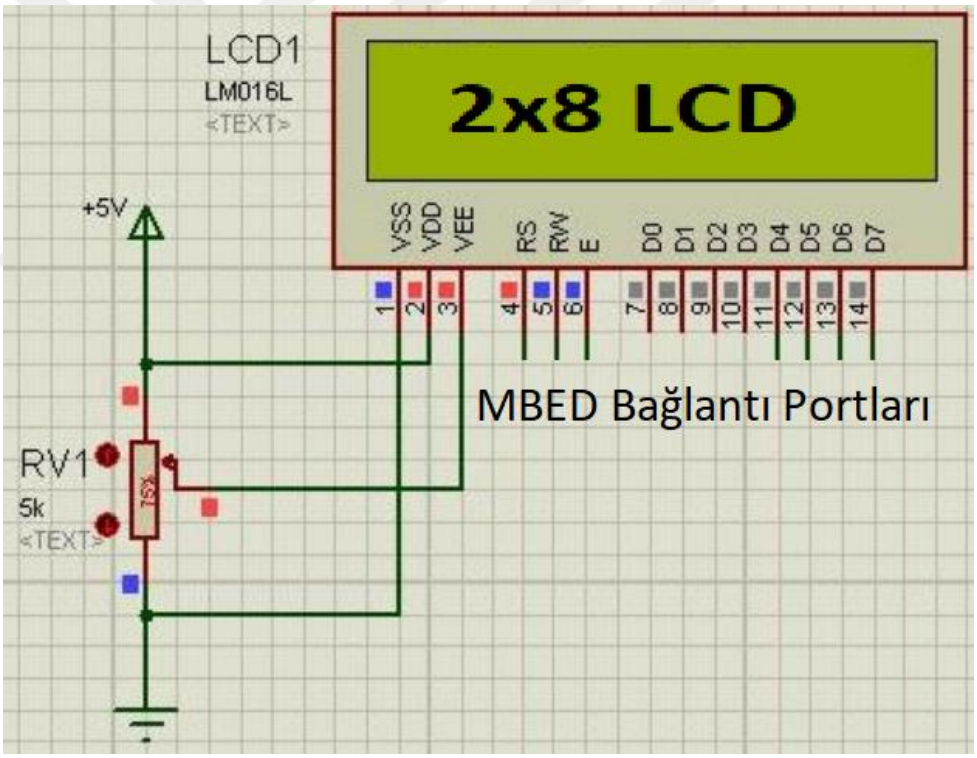
Şekil 3.6. PCB Baskı Devre

PWM sinyalinin daha kararlı olabilmesi için HD74HC08 entegresi kullanılmışır. Mantıksal AND (ve kapısı) olan bu entegre Mbed – LPC1768' in çıkış portlarındaki geriliminin istenilen düzeyde(5V) olmasını sağlamaktadır. HD74HC08 entegresinin bağlantısı Şekil 3.7 ' de gösterilmiştir.



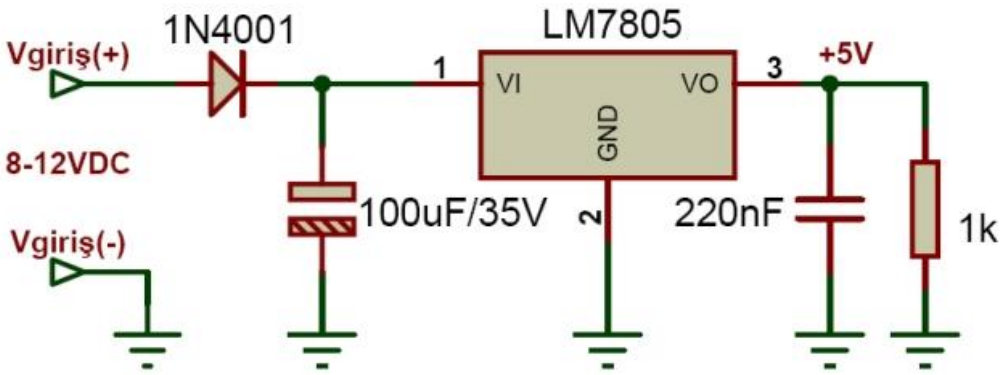
Şekil 3.7. HD74HC08 Entegre Bağlantısı

Devre kartında Mbed – LPC1768' e bilgisayar tarafından gönderilen koordinat bilgilerinin anlık gösterilmesi için 2x8 LCD Ekran kullanılmıştır. LCD ekranın devre bağlantısı Şekil 3.8 ' de gösterilmiştir.



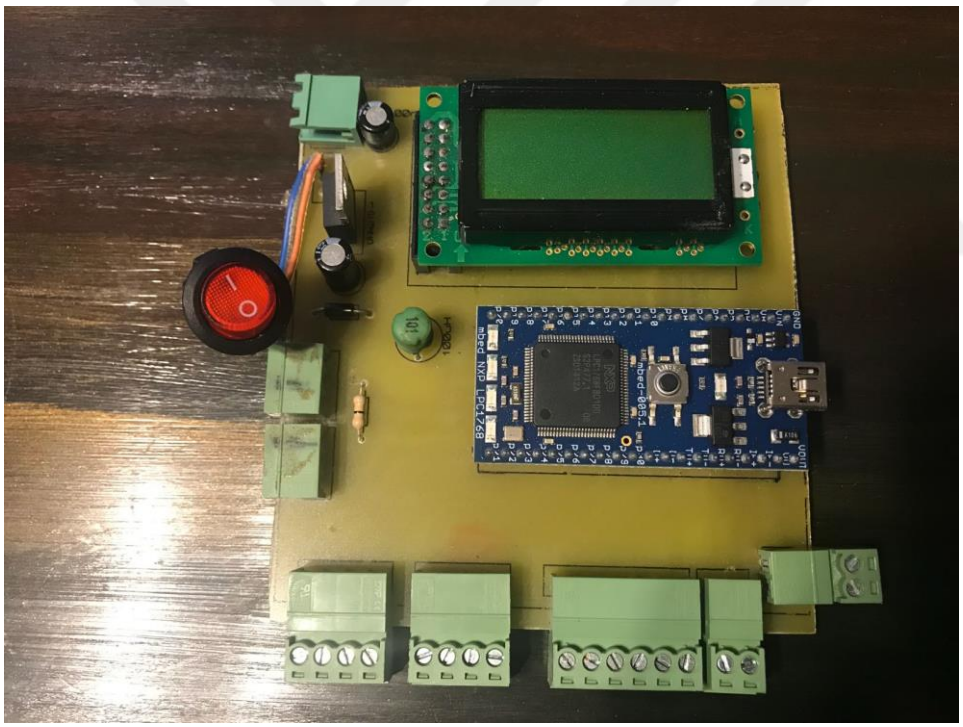
Şekil 3.8. LCD Bağlantı Şeması

Ekranın çalıştırılabilmesi ve regülatör devresi kullanılarak LCD' nin çalıştırılabilmesi için gerekli olan 5V sağlanmıştır. Regülatör devresi Şekil 3.9 ' da gösterilmiştir. Ayrıca devrede Mbed – LPC1768 işlemcisindeki programın çalışmaya başladığının farkedilebilmesi için Buzzer kullanılmıştır.



Şekil 3.9. Regülatör Devresi

Devrenin farklı çalışmalarda da kullanılabilmesi için bazı port çıkışlarının PCB kartı üzerindeki yerlerine Klemens bağlantıları yapılmıştır. Devre kartı Şekil 3.10'da gösterilmektedir.



Şekil 3.10. Kontrol Kartı

4. SONUÇ VE ÖNERİLER

Görüntü işleme teknikleri genellikle Askeri Endüstri (sualtı Görüntüleme), Tıp (tümör tespiti, damar gibi yapıların belirginleştirilmesi, tomografi, ultrason), Robotik, Trafik, Anstronomi, Radar ve Güvenlik Sistemleri gibi birçok alanda kullanılmaktadır. Görüntü işleme algoritmaları genellikle C++ programlama dili için OpenCV, Java için JavaCV, C# için EmguCV veya Python ve MATLAB kullanılarak oluşturulur. Haraketli nesnenin takibi için Mbed, PIC, Raspberry Pi, Arduino gibi fiziksel programlama platformları kullanılır.

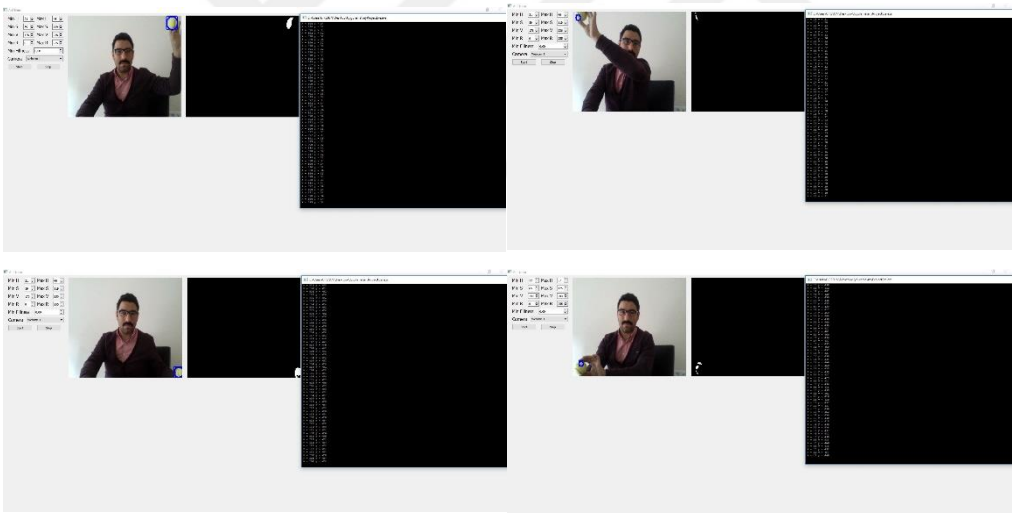
OpenCv kütüphanesi açık kaynaklı görüntü işleme kütüphanesi olduğu için nesne tanıma, yüz algılama, hareket algılama, geometrik şekil algılama, kamera kalibrasyonu vb. birçok uygulamalarda tercih edilmiştir. Bu sayede daha anlaşılır bir yapı kullanılarak nesne takibi yapılmıştır. Ayrıca OpenCV bağımsız bir kütüphane olduğu için Windows, Linux, Android, İOS, FreeBSD, Mac OS platformlarında da çalışabilmektedir.

ARM Cortex M serisi Mbed mikrodenetleyiciler üzerinde hızlı protoipleme ve ürün geliştirme amaçlı kullanılmaktadır. Gelişen teknolojiyle birlikte birçok mikroişlemci modelini desteklemektedir. ARM firması tarafından desteklenen geliştirme ortamı sayesinde internet üzerinden programların yazılması, derlenmesi ve oluşturulması yapılmaktadır. Bütün işletim sistemlerinde program yazılabilmesi ve internet ortamında programın kalıcı hale gelmesi avantaj oluşturmaktadır. Ayrıca Mbed USB, SPI, I2C, CAN, Ethernet ve Uart bağlantı arayüzlerine sahip olması ve direk olarak derlenen programın Mbed içerisine atılarak kullanımı sağlanmaktadır.

Bu çalışmada öncelikle hareketli nesnelerin Visual Studio Programı arayüzünde HSV degerlerinin OpenCV kütüphanesi sayesinde bulunması sağlanmıştır. Takip edilecek nesnenin görüntüsünün HSV değerleri bizim tarafımızdan tanımlanabileceği gibi, OpenCV kütüphanesi kullanılarak yazılan program sayesinde mause ile seçim yapılarakta tanımlanabilmektedir. Tanımlama işlemi Şekil 4.1 ’de gösterilmiştir.

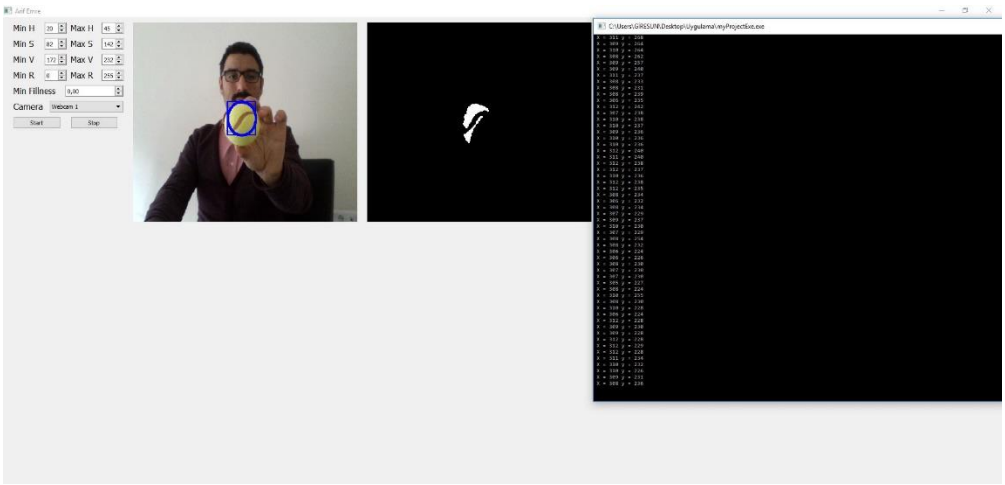
Şekil 4.1 Takip Edilecek Nesnenin Renk Bilgisinin Ayarlanması

Renk değerlerine göre filtreleme işlemleri yapılarak görüntünün ağırlık merkezi bulunmuştur. Bu bilgiden yararlanılarak görüntünün ekran üzerinde hangi koordinatlar üzerinde olduğu saptanmıştır. Görüntü için kullanılan ekran 640 x 480 çözünürlükte ayarlanmıştır. Ekranın farklı bölümlerinde bulunan nesnenin koordinat bilgilerinin çıkarımı Şekil 4.2'de gösterilmiştir.



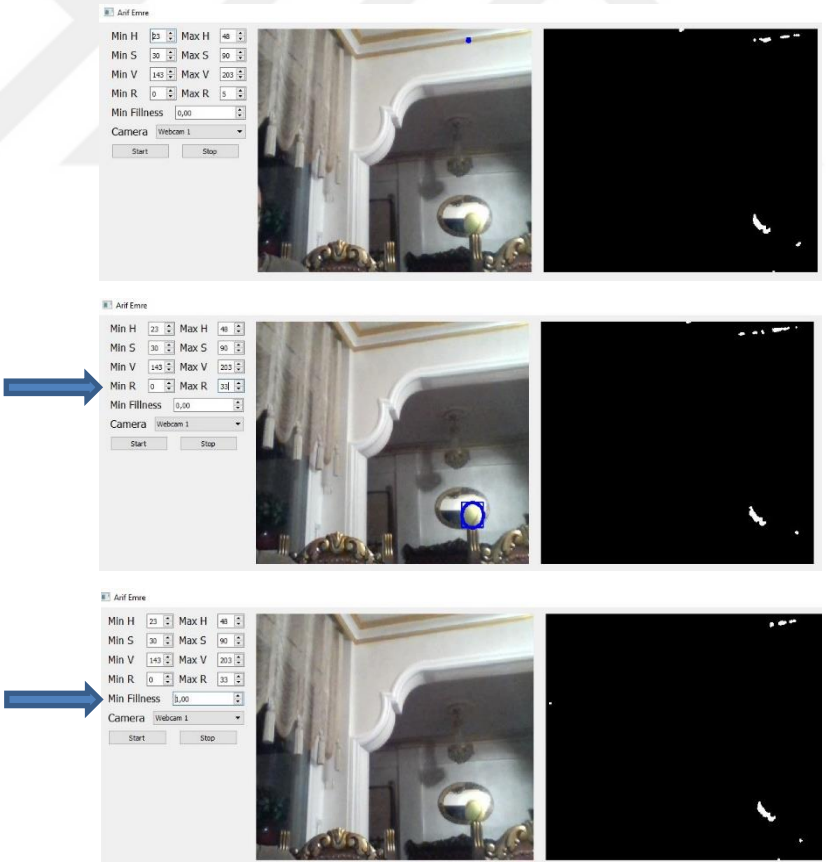
Şekil 4.2 Nesnenin koordinat bilgisinin çıkartılması

Görüntünün ekran içerisinden çıkmaması için ekranın orta noktası ile nesnenin ağırlık merkezi arasındaki fark hesaplanmıştır. Konum değişikliğinin oranına bakılarak geliştirilen program sayesinde kameranın görüntünün ağırlık merkezine odaklanması sağlanmıştır. Nesnenin ağırlık merkezini 640 x 480 çözünürlükte olan bir ekranda 320 x 240 noktasında bulundurmak amaçlanmıştır. Hareketli nesnenin ekranın ağırlık merkezi ve çevresinde bulundurulması Şekil 4.3' te gösterilmiştir.



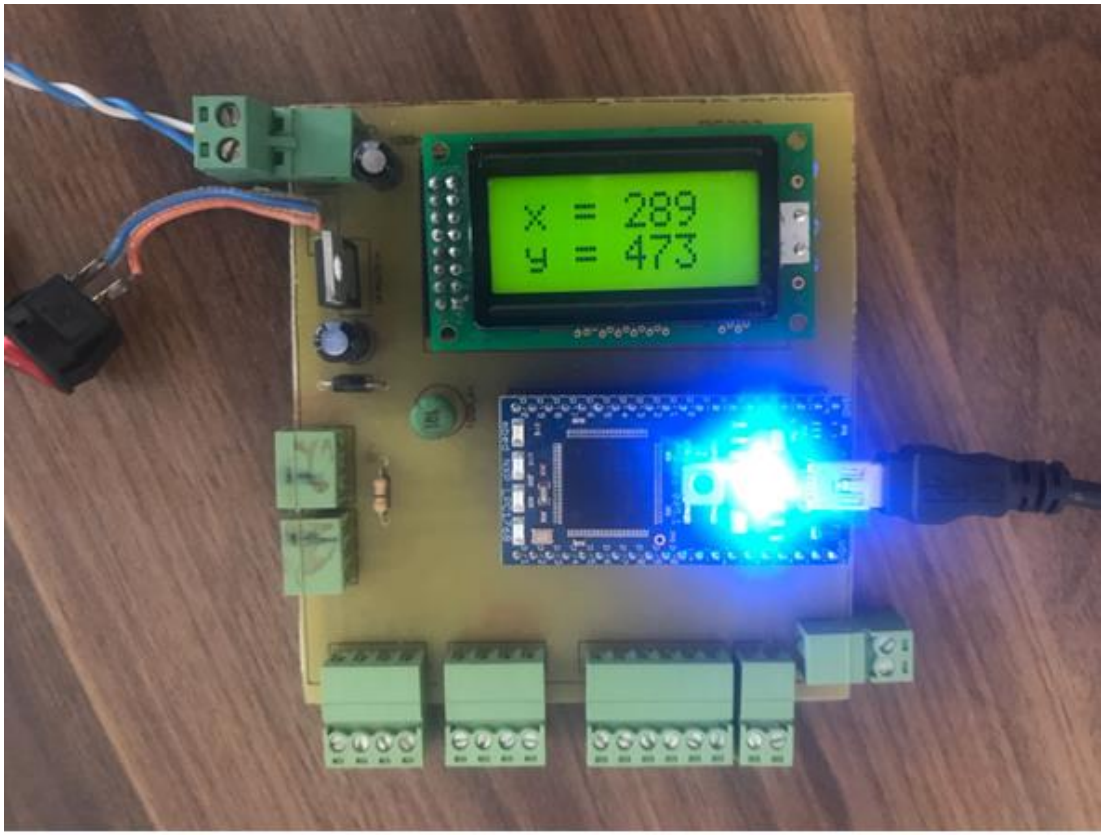
Şekil 4.3 Hareketli nesnenin ekranın ağırlık merkezinde bulundurulması

Aynı renkte olup ekranın farklı yerlerinde görünen görüntü kirliliklerini önlemek için ise algılanan nesnenin doluluk oranı veya ölçü kriterlerinin ayarlaması yapılmaktadır. Şekil 4.4' te doluluk oranı ve ölçü kriterinin ayarlanmasının takip edilecek nesne üzerindeki etkisi görülmektedir.



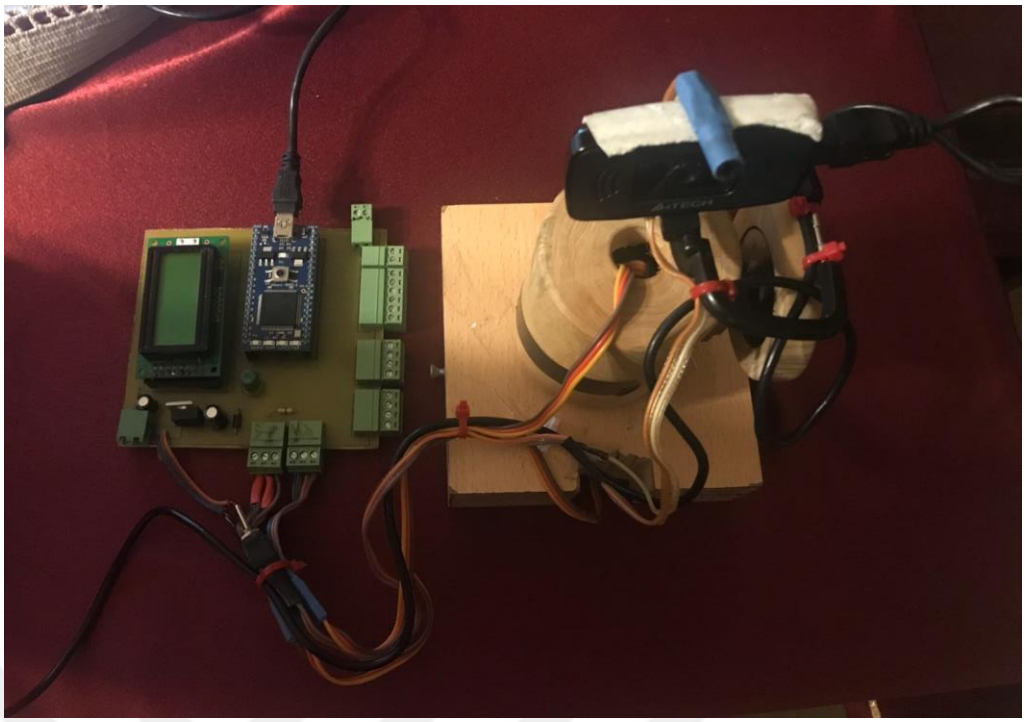
Şekil 4.4 Doluluk oranı ve ölçü kriterinin etkisi

USB seri haberleşme protokolü sayesinde takip edilecek nesnenin ekran üzerindeki anlık bilgisi Mbed' e gönderilmektedir. Mbed içerisine yazılan program sayesinde RC-Servo motor düzeneği sürekli bir şekilde takip edilecek nesneyi ağırlık merkezinde konumlandırmak için hareket edecektir. Mded bulunan elektronik devrede koordinat bilgilerininde anlık görüntülenmesi için LCD kullanılmıştır. Mbed tarafından koordinatların LCD'de görüntülenmesi Şekil 4,5'te gösterilmiştir.



Şekil 4.5 Koordinat bilgisinin Lcd'de görüntülenmesi

Burada dikkat edilmesi gereken konulardan bir tanesi nesnenin hareket ettirilecek konumdaki bilgisinin değil anlık bilgisine göre hareket mekanizmasının çalıştırılması sağlanmalıdır. Takip düzeneği Şekil 4.6' da gösterilmiştir. Nesnenin konumlandırılması yapılırken anlık koordinat bilgisi alınmaktadır. Bu sayede hareketli nesnenin anlık değişimlerinde sistemin en az zarar görmesi sağlanmış olur.



Şekil 4.6 Takip Düzeneği

Görüntü işlemede kullanılan farklı filtreleme yöntemleri ve algoritmalar sistemin kararlılığını arttırabilir. Teknolojinin gelişmesiyle birlikte çözünürlüğü ve anlık görüntü alma sayısı daha yüksek kameralar kullanılarak hareketli nesne takibi mükemmel bir şekilde yapılabilir.

KAYNAKLAR

Bradski G & Kaehler A (2008). Learning OpenCV: Computer Vision with the OpenCV Library. *O'Reilly Media,* 16-17, USA.

Camara-Chavez G, Precioso F, Cord M, Phillip-Foliguet S and Araujo A D A (2008). An interactive video content-based retrieval system and Image Processing. *Signals and Image Processing,* 133-136. doi:10.1109/IWSSIP.2008.4604385.

Chen H T, Lin H H & Liu T L (2001). Multi-object tracking using dynamical graph matching. *In Processing of IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 210–217.

Chengzhong Yu, Jun Zhu and Xiaohui Yuan (2005). Video object detection based on background subtraction, *Journal of Southeast University (Natural Science Edition),* 2.

Collins R T, Lipton A J, Kanade T, Fujiyoshi H, Duggins D, Tsin Y, Tolliver D, Enomoto N, Hasegawa O, Burt1 P & Wixson1 L (2000). A system for video surveillance and monitoring: VSAM final report. Technical report CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University.

Dedeoglu Y (2004). Akıllı video gözetimi için hareketli nesne bulma, takip etme ve sınıflandırma, Yüksek Lisans Tezi, Bilkent Üniversitesi Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, Ankara.

Driessen J (2004). Object Tracking in a Computer Vision Based Autonomous See-and-Avoid System for Unmanned Aerial Vehicles. KTH Numerical Analysis and Computer Science, Master's Thesis in Computer Science, Royal Institute of Technology , Sweden.

Gonzalez R C & Woods R E (2007). *Digital image processing* (Third edition). Prentice Hall, 1-7, New Jersey.

Grosvenor D (2009). Detection and Tracking of Human Subjects. Master's Thesis in Computer Science, Bachelor of Science in Computer Engineering,402 , USA.

Haritaoglu I (1998). A Real Time System for Detection and Tracking of People and Recognizing Their Activities Doctoral Dissertation, University of Maryland at College Park, USA.

Haritaoglu I, Harwood D & Davis L S (1998). W4: A real time system for detecting and tracking people. *In Computer Vision and Pattern Recognition,* 962–967.

Heikkila J & Silven O (1999). A real-time system for monitoring of cyclists and pedestrians. In Proc. of Second IEEE Workshop on Visual Surveillance, 74–81, Fort Collins, Colorado.

Horprasert T, Harwood D & Davis L S (1999). A statistical approach for realtime robust background subtraction and shadow detection. *In Processing of IEEE Frame Rate Workshop,* 1–19, Kerkyra, Greece.

Karasulu B (2013). Videolardaki Hareketli Nesnelerin Tespit ve Takibi İçin Uyarlanabilir Arkaplan Çıkarımı Yaklaşımı. Doktora Tezi, Ege Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı, İzmir.

Lipton A J, Fujiyoshi H & Patil R S (1998). Moving target classification and tracking from real-time video. *In Processing of Workshop Applications of Computer Vision,* 129–136.

Lorsakul A & Suthakorn J (2007). Traffic Sign Recognition for Intelligent Vehicle/Driver Assistance System Using Neural Network on OpenCV, The 4th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI) , 22-24 November, Book of Abstracts,1-6, South Korea.

McKenna S J, Jabri S, Duric Z & Wechsler H (2000). Tracking interacting people. *In Processing of International Conference on Automatic Face and Gesture Recognition,* 348–353.

Mohamed Y E, Bassam S A Z & Hossam M Z (2009). Soccer video summarization using enhanced logo detection, 16th IEEE International Conference on Image Processing (ICIP), 7-12 November, Book of Abstracts,4345-4348, Cairo, Egypt.

Remagnino P, Jones G A, Paragios N and Regazzoni C S (2002). *Video-Based Surveillance Systems: Computer Vision and Distributed Processing*. Kluwer, 279, Boston.

Stauffer C & Grimson W (1999). Adaptive background mixture models for realtime tracking. *In Processing of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 246-252.

Vijayalaxmi, Anjali K, Srujana B and Rohith Kumar P (2014). Object Detection and Tracking Using Image Processing. *Global Journal of Advanced Engineering Technologies*, 2277-6370.

Wang Cheng-liang, Jia Liang-liang, Chen Juan-Juan (2001). Moving Object Detection and Tracking Based on Improved Surendra Background Updating Algorithm, Third International Conference on Digital Image Processing. *South J, Blass B,* 1-6, Blackwell, London.

Wang L ,Hu W & Tan T (2003). Recent developments in human motion analysis. Pattern Recognition, 36 (3): 585–601.

Yan Wq , Wang J & Kankanhalli M S (2005). Automatic video logo detection and removal, Multimedia Systems , 10(5): 379-391.

Yilmaz A, Javed O & Shah M (2006). Object Tracking: A Survey. *ACM Computing Surveys,* 38(4):13.

ÖZGEÇMİŞ

Adı Soyadı	**:** Arif Emre ÖZDEMİR
DoğumYeri veTarihi	**:** Giresun 19/08/1988
Adres	**:** Gedikkaya Mahallesi Romancı Güzide Sabri Sokak Özlem Site B/1 Blok No:49 Daire:13 Giresun
E-Posta	**:** arifemre2m@gmail.com
Lisans	**:** Kırıkkale Üniversitesi, Mühendislik Fakültesi, Elektrik ve Elektronik Mühendisliği, 2011.
Yüksek Lisans	**:** Ondokuzmayıs Üniversitesi., Fen Bilimleri Enstitüsü, Elektrik ve Elektronik Mühendisliği ABD, Samsun, 2018.
Mesleki Deneyimi	**:** Özel Sektör (Elektrik ve Elektronik Mühendisi)