KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

İNSANSIZ ARAÇLAR İLE GÜVENLİ HABERLEŞME VE NESNE TAKİBİ

MÜHENDİSLİK TASARIMI

Hakan Murat AKSÜT
Yavuz ÜNVER
Onur ÇEVİK

2018-2019 GÜZ/BAHAR DÖNEMİ
KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

İNSANSIZ ARAÇLAR İLE GÜVENLİ HABERLEŞME VE NESNE TAKİBİ

MÜHENDİSLİK TASARIMI

Hakan Murat AKSÜT
Yavuz ÜNVER
Onur ÇEVİK

2018-2019 GÜZ/BAHAR DÖNEMİ

IEEE Etik Kuralları
IEEE Code of Ethics

Mesleğime karşı şahsi sorumluluğumu kabul ederek, hizmet ettiğim toplumlara ve üyelerine en yüksek etik ve mesleki davranışta bulunmaya söz verdiğimi ve aşağıdaki etik kurallarını kabul ettiğimi ifade ederim:

1. Kamu güvenliği, sağlığı ve refahı ile uyumlu kararlar vermenin sorumluluğunu kabul etmek ve kamu veya çevreyi tehdit edebilecek faktörleri derhal açıklamak;
2. Mümkün olabilecek çıkar çatışması, ister gerçekten var olması isterse sadece algı olması, durumlarından kaçınmak. Çıkar çatışması olması durumunda, etkilenen taraflara durumu bildirmek;
3. Mevcut verilere dayalı tahminlerde ve fikir beyan etmelerde gerçekçi ve dürüst olmak;
4. Her türlü rüşveti reddetmek;
5. Mütenasip uygulamalarını ve muhtemel sonuçlarını gözeterek teknoloji anlayışını geliştirmek;
6. Teknik yeterliliklerimizi sürdürmek ve geliştirmek, yeterli eğitim veya tecrübe olması veya işin zorluk sınırları ifade edilmesi durumunda ancak başkaları için teknolojik sorumlulukları üstlenmek;
7. Teknik bir çalışma hakkında yansız bir eleştiri için uğraşmak, eleştiriyi kabul etmek ve eleştiriyi yapmak; hatları kabul etmek ve düzeltmek; diğer katkı sunanların emeklerini ifade etmek;
8. Bütün kişilere adilane davranmak; ırk, din, cinsiyet, yaş, milliyet, cinsi tercih, cinsiyet kimliği, veya cinsiyet ifadesi üzerinden ayırımcılık yapma durumuna girişmemek;
9. Yanlış veya kötü amaçlı eylemler sonucu kimsenin yaralanması, mülklerinin zarar görmesi, itibarlarının veya istihdamlarının zedelenmesi durumlarının oluşmasından kaçınmak;
10. Meslektaşlara ve yardımcı personele mesleki gelişimlerinde yardımcı olmak ve onları desteklemek.

IEEE Yönetim Kurulu tarafından Ağustos 1990'da onaylanmıştır.

ÖNSÖZ

"İnsansız Araçlar ile Güvenli Haberleşme ve Nesne Takibi" adlı bu çalışma Karadeniz Teknik Üniversitesi Bilgisayar Mühendisliği Bölümü Lisans Tasarım Projesi olarak hazırlanmıştır. Bu proje ile kameradan alınan görüntü şifrelenerek belli bir istasyona gönderilmiş ve istasyon üzerindeki bilgisayarda nesne takip algoritmaları uygulanmıştır.

Bu projenin yapımında bize yardımcı olan Öğr.Gör.Dr. Zafer YAVUZ hocamıza saygılarımızı sunup teşekkür ediyoruz. Ayrıca her zaman yanımızda olan ve desteklerini, sevgilerini bizden hiçbir zaman esirgemeyen ailelerimize her şey için çok teşekkür ediyoruz.

Hakan Murat AKSÜT
Yavuz ÜNVER
Onur ÇEVİK
Trabzon 2018

İÇİNDEKİLER

ÖZET

Projede, güvenli iletim ortamı üzerinden kullanıcılara gerçek zamanlı nesne takip hizmetinin sunulması amaçlanmaktadır. İnsansız araçtan alınan görüntünün saldırganlar tarafından ele geçirilmemesi için AES algoritması ile şifrelenerek TCP protokolü ile istasyona gönderilmesi, istasyon başındaki görevlinin ise arayüze yansıtılan görüntüden mouse yardımı ile herhangi bir nesneyi seçmesinin ardından seçilen nesnenin takibinin başlaması sağlanacaktır. Takip edilmek istenen nesne insan, araç gibi hareket eden herhangi bir nesne olabilir. Proje tamamlandığında takip algoritmalarının karmaşık ortamlarda nesneleri karıştırmaması ve nesne belli bir süre kaybolsa bile geri geldiğinde nesnenin tekrar bulunarak takibe devam edilmesini hedeflemektedir.

1. GENEL BİLGİLER

1.1. Giriş

Günümüzde görüntü işleme teknolojisi tespit, takip etme, öğrenme gibi işlemlerin kolaylıkla yapılabilmesini sağlamaktadır. Görüntü işleme teknolojisi tıbbi, askeri sosyal medya uygulamaları ve bilgisayar bilimleri gibi birçok alanda kullanılmaktadır.Bu ve bunlar gibi bilimsel projelerin bazılarında bir veya birden fazla cihazın birbirleriyle internet teknolojisini kullanarak iletişim halinde olması gerekebilir. Fakat internetin gelişmesiyle birlikte çeşitli siber saldırı yöntemleri de gelişmiştir. Bunlardan bazıları sadece bilişim sistemlerini devre dışı bırakmayı hedeflerken, bazıları ise veri hırsızlığı yapmaya yöneliktir. Bu nedenle verinin önemli olduğu alanlarda veriyi korumak amacıyla şifreleme uygulanmaktadır. Benzer sebeplerden dolayı bu projede insansız araçtan alınan veri AES Şifreleme Algoritması ile şifrelenecek ve istasyon makineye şifreli bir biçimde yollanacaktır. İstasyon makine tarafından alınan veri çözülecek ve kullanıcı arabirimi üzerinde gösterilecektir.

Projenin nesne takibi safhasında ise kullanıcı arabirimi üzerinden seçilen nesnenin takibi yapılarak takip edilen nesne yine kullanıcı arabirimi üzerinden gerçek zamanlı olarak gösterilecektir.

1.2. Qt ile Kullanıcı Arabirimi ve Veri Transferi

Qt birden çok platformu destekleyen bir grafiksel kullanıcı arayüzü geliştirme araç takımıdır. Ağırlıklı olarak C++ ve Python dilleri ile arayüz geliştirmeye olanak sağlayan bu framework ilk olarak 1990 yılında bir ultrason projesi üzerinde çalışırken aynı anda Unix, Macintosh, and Windows işletim sistemlerinde de çalışabilen bir GUI frameworküne ihtiyaç duyan Haavard Nord ve Eirik Chambe-Eng tarafından fikir olarak ortaya atılıp 1995 yılında open source(açık kaynaklı) olarak piyasaya sürülmüştür.

Qt, günümüzde

1) Cross-Platform olması
2) Bellek yönetiminin iyi olması
3) Qt Creator isminde çok güçlü bir IDE'ye sahip olması
4) Geniş ve güçlü kütüphanelere sahip olması
5) Geniş dökümanlara sahip olması
6) C++ üzerine kurulmuş bir framework olduğundan hızlı ve nesneye dayalı bir yapıya sahip olması yönlerinden dolayı günümüzde bir çok kullanıcının tercih ettiği bir frameworktür.

Qt ile geliştirilen bazı projeler

1) Linux işletim sisteminin en popüler masaüstü ortamlarından biri olan KDE, baştan sona Qt ile geliştirilmiştir.
2) Tesla marka araçların kadran ve diğer bazı parçaları Qt kullanarak tasarlanmıştır.
3) Parallels Desktop for Mac
4) Wireshark
5) WPS Office
6) Google Earth
7) Gnu Octave
8) BlackBerry 10

Qt içerisinde gui kütüphaneleri dışında thread(QThread), ağ kütüphaneleri(QUdp, QTcp vb), pdf(QPdfWriter), çeşitli veri yapıları(QVector, QMap, QByteArray vb) gibi birçok paketi barındırarak üçüncü parti paket kullanımını azaltmakta ve projeleri mümkün olduğunca qt ekosistemi dışına çıkarmadan programlamaya imkan vermektedir. Ayrıca tamamen nesne yönelimli programlama kullanılarak tasarlandığı için ihtiyaç haline istenen nesnelerin kalıtım yoluyla özelleştirilmesini kolaylaştırmaktadır.

1.2.1. Kullanıcı Arabirimi

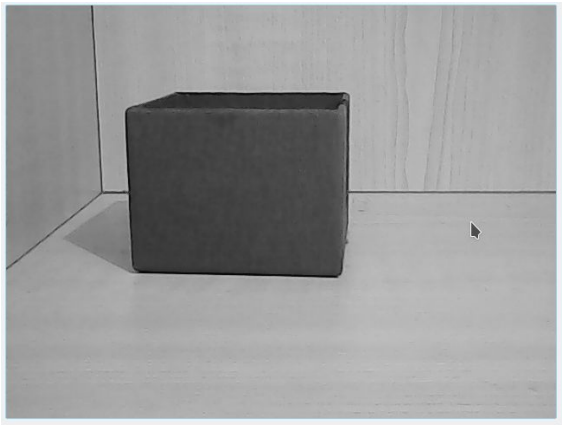
İletim ve görüntü işleme kısımlarını hızlı ve verilere erişim kolaylığı yönünden C++ ile programlamaya karar verildiğinden ve yukarıdaki bölümde bahsedilen Qt'nin kütüphane çeşitliliği ve nesneye dayalı bir framework olması sebebiyle projenin arayüz kısmının Qt ile yapılmasına karar verildi.

Projede sürekli olarak görüntü şifreleme-deşifreleme olacağından ve görüntü işleme aşamalarında akan videonun her bir frame(kare)ine erişim ihtiyacı duyulduğundan kameradan video çekmek yerine sürekli resim çekilip alıcı tarafta video olarak birleştirilmesine karar verildi. Kolaylık açısından ve insan gözünün saniyede 25 kareden fazlasını algılayamadığından dolayı saniyede 24 kare gönderilmesine karar verildi.

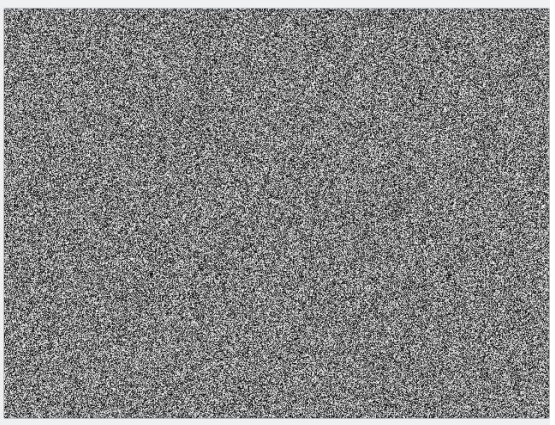
Proje gönderici(client) ve alıcı(server) olarak 2 uygulamanın iletişimi şeklinde çalışmaktadır. Gönderici program kameradan aldığı görüntüyü şifreleyerek alıcıya göndermekte ve alıcı program ise şifreyi çözerek arayüze yansıtıp obje seçimi için kullanıcıdan komut beklemektedir. Alıcı program kullanıcının seçtiği nesnenin verisini sonraki bölümlerde gösterilen görüntü işleme algoritmaları ile işlemektedir.

Qt'de resimler `QImage` sınıf nesnelerinde buffer değeri verilerek oluşturulabilir. Bu resim nesneleri sonrasında `QGraphicsView` denilen kanvas bileşeninde `QGraphicsScene` setlenerek `QGraphicsPixmap` ve `QPixmap` kütüphaneleri yardımıyla gösterilebilir.

Bu projede de kameradan alınan YUV görüntü RGB formatına dönüştürülüp gerekli işlemlerden geçikten sonra, boyutun küçültülmesi açısından grayscale yani siyah-beyaz görüntüye dönüştürülerek, bufferlar yardımıyla karşı tarafa iletilip burada görüntü tekrar oluşturulduktan sonra, arayüze gösterilmiştir. Bu aşamadan sonra arayüzde gösterilen görüntüde, kullanıcının takip edilmesini istediği nesneyi seçmesini sağlamak amacıyla fare ile objeyi seçme özelliği de eklenmiştir. Bu seçme işlemi `QRubberBand` ve mouse eventler ile gerçekleştirilmiştir. Fare ile nesnenin bulunduğu karesel alanın seçimi sonrasında görüntü işleme süreci başlamaktadır. Bunun dışında görüntü frame frame geldiğinden, bir adet radio button yardımıyla görüntünün şifreli ve şifresiz hali kullanıcının seçimine göre arayüzde real-time gösterilmektedir.



Şekil 1.2.1.1 Şifresiz Görüntü



Şekil 1.2.1.2 Şifrelenmiş Görüntü

1.2.2. Veri Transferi

İletim kısmında her ne kadar gerçek zamanlı canlı yayın uygulamalarında UDP, RTP, RTSP gibi iletim protokolleri tercih edilse de bu projede görüntü iletiminde TCP protokolü kullanılmıştır. Bunun öncelikli sebebi işlenecek görüntüde doğruluğun gecikmeye göre daha öncelikli olmasıdır. Bu yüzden TCP kullanmamızın avantajı bilginin karşı tarafa doğru olarak ulaştığından emin olurken , dezavantajı bu protokolün gecikmelere ve takılmalara sebep olabilmesidir.

Yukarıda bahsedilen sürekli olarak ve saniyede 24 kez kere gerçekleşen foto gönderimi Qt'nin timer zamanlayıcısı ve signal-slot sistemi sayesinde gerçekleşmektedir. İletim protokolü olarak TCP kullanıldığından iletim bağlantı tabanlı(connection-oriented)dır. Alıcı(server) uygulama çalıştırıldığında belli bir port için dinlemeye başlar. Bu yüzden iki uygulama (client-server) arası iletişim kullanıcının client yani gönderici kısmının arayüzündeki butona basmasıyla başlar. Her bir kare için gönderici, alıcı tarafı için yeni bir connection oluşturup veriyi karşı tarafa işlemesi için iletir.

Sonuç olarak sağlıklı ve güvenli bir şekilde alınan veriler güvenli iletişim üzerinden iletimlerinin ardından işlenilip bahsi geçen arayüzde yayınlanır.

1.3. AES ile Veri Şifreleme

1.3.1. AES Algoritmasının Tarihçesi

1997 yılına kadar veri şifreleme standardı olarak DES Algoritması kullanılıyordu. Gelişen teknoloji ile birlikte DES Algoritmasının 64 bitlik anahtar uzayı güvenilirliğini yitirmeye başladı. Bu sebeple NIST (Ulusal Standartlar ve Teknoloji Enstitüsü) 1997 yılında yeni bir yarışma düzenlemiştir. 1998 yılında ilk turu gerçekleştirilen yarışmaya 15 farklı proje başvurmuştur, 1999 yılındaki ikinci tura ise sadece 5 proje kalmıştır. Toplamda 4 yıl süren değerlendirmeler sonucunda 2000 yılında Joan Daemen ve Vincent Rijmen tarafından tasarlanan Rijndael Algoritması Advanced Encryption Standard (AES) ismiyle NIST tarafından kabul edilmiştir.

1.3.2. AES Algoritmasının Genel Yapısı

AES simetrik bir şifreleme algoritmasıdır yani hem şifreleme hem şifre çözme işlemleri için aynı anahtar kullanılır. AES için girdi ve çıktı matrisleri her zaman 128 bit olmak zorundadır ancak anahtar uzunluğu 128, 192 veya 256 bit olabilir.

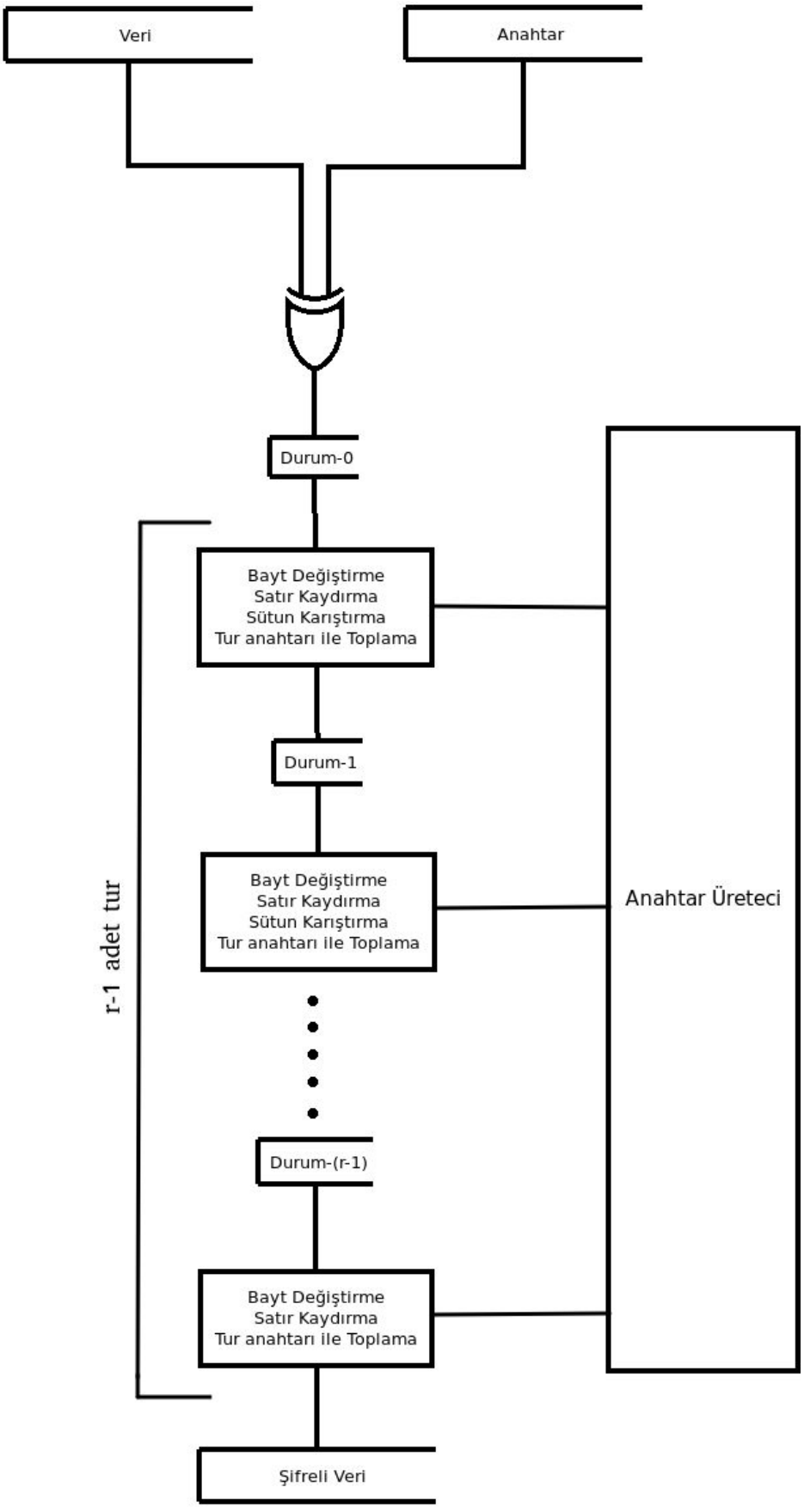
S0	S4	S8	S12
S1	S5	S9	S13
S2	S6	S10	S14
S3	S7	S11	S15

Tablo 1.3.2.1. Durum Matrisi

Tablo 1.3.2.1 ile gösterilen durum matrisinde her bir hücre 8 bit yer kaplamaktadır, 16 hücre bulunduğu için toplam 128 bitlik bir veriye karşılık düşer. Şifrelenecek mesaj ve anahtar durum matrisleri şeklinde düşünülerek üzerlerinde gerekli işlemler yapılır.

AES algoritması Bayt Değiştirme, Satır Kaydırma, Sütun Karıştırma ve Tur Anahtarı ile Toplama gibi adımların tekrar etmesi şeklinde düşünülebilir.

128 bit AES şifrelemesi için 10, 192 bit için 12, 256 için ise 14 tur sonunda şifrelenmiş mesaja ulaşılır.



Şekil 1.3.2.1. AES Algoritması Genel Akış

1.3.3. Bayt Değiştirme

Bayt değiştirme işlemi durum matrisindeki baytların farklı baytlara dönüştürülmesi işlemidir. Bu dönüşüm daha önceden belli bir lookup table üzerinden yapılır. Örnek bir tablo aşağıda verilmiştir. Örneğin bayt değiştirme yapılacak olan değer 0x37 olsun, bayt değiştirme sonrası bu değer 0x9A (sbox'ın 0x37. değeri) değerine dönüşür.

```
BYTE sbox[256] = {
0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe,
0xd7, 0xab, 0x76,
0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c,
0xa4, 0x72, 0xc0,
0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71,
0xd8, 0x31, 0x15,
0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb,
0x27, 0xb2, 0x75,
0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29,
0xe3, 0x2f, 0x84,
0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a,
0x4c, 0x58, 0xcf,
0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50,
0x3c, 0x9f, 0xa8,
0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10,
0xff, 0xf3, 0xd2,
0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64,
0x5d, 0x19, 0x73,
0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde,
0x5e, 0x0b, 0xdb,
0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91,
0x95, 0xe4, 0x79,
0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65,
0x7a, 0xae, 0x08,
0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b,
0xbd, 0x8b, 0x8a,
0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86,
0xc1, 0x1d, 0x9e,
0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce,
0x55, 0x28, 0xdf,
0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0,
0x54, 0xbb, 0x16};
```

Tablo 1.3.3.1. Bayt değiştirme matrisi

0x48	**0xf6**	**0x24**	**0x5b**		**0x52**	**0x42**	**0x36**	**0x39**
0xcc	**0x1a**	**0xb7**	**0x34**		**0x4b**	**0xa2**	**0xa9**	**0x18**
0x37	**0xc9**	**0xd4**	**0x71**		**0x9a**	**0xdd**	**0x48**	**0xa3**
0x1a	**0x67**	**0x33**	**0x01**		**0xa2**	**0x85**	**0xc3**	**0x7c**

Tablo 1.3.3.2. Bayt değiştirme öncesi ve sonrası durum matrisi

1.3.4. Satır Kaydırma

Satır kaydırma işlemi bayt değiştirme işleminden sonra oluşan yeni durum üzerine uygulanır. Durum matrisindeki satırların belli değerde sola kaydırılması anlamına gelir.

				Kaydır				
0x48	**0xf6**	**0x24**	**0x5b**	**0 bayt**	**0x48**	**0xf6**	**0x24**	**0x5b**
0xcc	**0x1a**	**0xb7**	**0x34**	**1 bayt**	**0x1a**	**0xb7**	**0x34**	**0xcc**
0x37	**0xc9**	**0xd4**	**0x71**	**2 bayt**	**0xd4**	**0x71**	**0x37**	**0xc9**
0x1a	**0x67**	**0x33**	**0x01**	**3 bayt**	**0x01**	**0x1a**	**0x67**	**0x33**

Tablo 1.3.4.1 Satır kaydırma işlemi öncesi ve sonrası durum matrisi

1.3.5. Sütun Karıştırma

Sütun karıştırma işlemi satır karıştırma adımında oluşan durum matrisinin her sütunun ayrı ayrı belli bir matris ile çarpılması ve ortaya çıkan matrisin yeni sütun olarak kullanılmasıdır.

0x02	0x03	0x01	0x01	·	S0	=	S0’
0x01	0x02	0x03	0x01		S1		S1’
0x01	0x01	0x02	0x03		S2		S2’
0x03	0x01	0x01	0x02		S3		S3’

Tablo 1.3.5.1 Sütun Karıştırma

Bu çarpma işlemlerinden çıkan sonuç bazen 8 bitten büyük olabilir, böyle durumlarda çıkan sonuç indirgenmesi gerekir. Örneğin;

$$0xd4 * 0x02 = 11010100 * 00000010 = (x^7 + x^6 + x^4 + x^2) * (x) = x^8 + x^7 + x^5 + x^3$$

Görüldüğü gibi burada bulunan değer en az 9 bit ile ifade edilebilir. Bu nedenle sonucun 8. dereceden indirgenemez bir polinom(m(x)) ile modu alınmalıdır. Bu polinomun böleni tektir ve sadece kendisidir.

$$m(x) = x^8 + x^4 + x^3 + x + 1 = 100011011$$

Mod işlemi recursive olarak m(x) fonksiyonunun modu alınacak sayının en yüksek dereceli basamağına kadar kaydırılması ve exorlanması ile gerçekleştirilir. Recursive işlemlerin son bulacağı şart ise çıkan sonucun 8 bit ile ifade edilebildiği noktadır.

$$0xd4 * 0x02 = x^8 + x^7 + x^5 + x^3 = 110101000$$

Şekil 1.3.5.2. Mod işlemi

Bu örnekte mod işlemi tek iterasyonda tamamlandı ancak bazı örneklerde bu durum birkaç iterasyon sürebilir, ayrıca yeni sütundaki her bir bayt için ayrı ayrı 4 çarpma ve toplama işlemi olduğu için en kötü durumda her çarpmadan sonra mod işlemi uygulamak maliyeti artırmaktadır.

Maliyeti düşürmek için mümkün tüm durumlar daha önceden hesaplanıp bir lookup table üzerinde tutulur. [12] üzerinde tüm lookup table'lar görülebilir.

1.3.6. Tur Anahtarı ile Toplama

Her turun sonunda bulunan mesaj o anki tur anahtarı ile toplanır. Her turda farklı bir anahtar kullanıldığı için tur sayısı kadar yeni anahtar gereklidir. Aşağıda 10 turluk bir AES 128 algoritmasında gerekli anahtar tablosunun bir kısmı gösterilmiştir.

0	1	2	3	4	5	6	7		42	43
0x61	0x74	0x6c	0x6b	0xf2	0x86	0xea	0x81	……..	0xa3	0x6e
0x79	0x61	0x65	0x74	0x80	0xe1	0x84	0xf0	……..	0x39	0x1b
0x73	0x74	0x63	0x69	0x8a	0xfe	0x9d	0xf4	……..	0x4c	0x99
0x65	0x69	0x69	0x69	0x1a	0x73	0x1a	0x73	……..	0x67	0xa8

Tablo 1.3.6.1 Örnek genişletilmiş anahtar tablosu

4. sütun 2. anahtarın başlangıç noktasını ifade eder. 128 bitlik AES Algoritması için 11 adet farklı anahtara ihtiyaç vardır. Bunlardan ilki(belirlenen anahtar) en başta mesaj ile toplanır. Geriye kalan anahtarlar ilk anahtar yardımıyla üretilir ve her tur sonunda oluşan mesaj üzerine eklenir.

1.3.7. Tur Anahtarının Üretilmesi

Tur anahtarlarının üretilip sıralı bir şekilde bir bellek alanına yazılmasıyla oluşan diziye genişletilmiş anahtar dizisi denir. Genişletilmiş anahtar dizisi aşağıdaki algoritma ile bulunur.

N = Anahtardaki Kelime Sayısı
R = Tur Sayısı
K_i = İlk Anahtardaki Kelimeler, i=0…(N-1)
W_i = Genişletilmiş Anahtardaki Kelimeler, i=0...4(R+1)-1
$rcon^t$ = [0x00, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a]

$W_i = K_i$, Eğer i<N
$W_i = W_{i-N} \oplus$ Kaydır(SBox(W_{i-1})) $\oplus rcon_i$, Eğer i>=N ve $i \equiv 0 (\text{mod } N)$
$W_i = W_{i-N} \oplus$ SBox(W_{i-1}) , Eğer i>=N, N>6 ve $i \equiv 0 (\text{mod } N)$
$W_i = W_{i-N} \oplus W_{i-1}$, diğer durumlar

1.3.8. Deşifreleme

Deşifreleme algoritmasının akışı Şekil 1.3.2.1. deki akış adımlarının terslerini içermektedir. Ters Bayt Değiştirme, Ters Satır Kaydırma, Ters Sütun Karıştırma ve Tur Anahtarıyla Toplama adımlarını içerir ancak Tur Anahtarıyla Toplama adımı genişletilmiş anahtarın içerisindeki son anahtardan başlayarak geriye doğru ilerler. Yani şifreleme için kullandığımız son anahtar deşifreleme için kullandığımız ilk anahtar olur.

Ters bayt değiştirme işlemi için bayt değiştirme işleminde olduğu gibi bir lookup table'dan faydalanılır. Bu tablodaki veriler bayt değiştirme tablosundaki verilerin tersleridir. Örneğin 0x00 değerinin bayt değiştirme tablosundaki karşılığı 0x63 iken, 0x63 değerinin ters bayt değiştirme adımındaki değeri 0x00'dır. Aşağıda örnek bir ters bayt değiştirme tablosu görülmektedir.

```
BYTE invSbox[256] = {
0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3, 0x9e, 0x81,
0xf3, 0xd7, 0xfb,
0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e, 0x43, 0x44, 0xc4,
0xde, 0xe9, 0xcb,
0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b, 0x42,
0xfa, 0xc3, 0x4e,
0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d,
0x8b, 0xd1, 0x25,
0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d,
0x65, 0xb6, 0x92,
0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15, 0x46, 0x57, 0xa7,
0x8d, 0x9d, 0x84,
0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8,
0xb3, 0x45, 0x06,
0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf, 0xbd, 0x03, 0x01,
0x13, 0x8a, 0x6b,
0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0,
0xb4, 0xe6, 0x73,
0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37, 0xe8, 0x1c,
0x75, 0xdf, 0x6e,
0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa,
0x18, 0xbe, 0x1b,
0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0, 0xfe, 0x78,
0xcd, 0x5a, 0xf4,
0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27,
0x80, 0xec, 0x5f,
0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f, 0x93,
0xc9, 0x9c, 0xef,
0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83,
0x53, 0x99, 0x61,
0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63, 0x55,
0x21, 0x0c, 0x7d};
```

Tablo 1.3.8.1. Deşifreleme matrisi

Ters satır kaydırma adımında, satır kaydırma adımında yapılan işlemler sağa kaydırılarak tekrarlanır.

				Kaydır				
0x48	**0xf6**	**0x24**	**0x5b**	**0 bayt**	**0x48**	**0xf6**	**0x24**	**0x5b**
0xcc	**0x1a**	**0xb7**	**0x34**	**1 bayt**	**0x34**	**0xcc**	**0x1a**	**0xb7**
0x37	**0xc9**	**0xd4**	**0x71**	**2 bayt**	**0xd4**	**0x71**	**0x37**	**0xc9**
0x1a	**0x67**	**0x33**	**0x01**	**3 bayt**	**0x67**	**0x33**	**0x01**	**0x1a**

Tablo 1.3.8.2 Ters Satır Kaydırma İşlemi

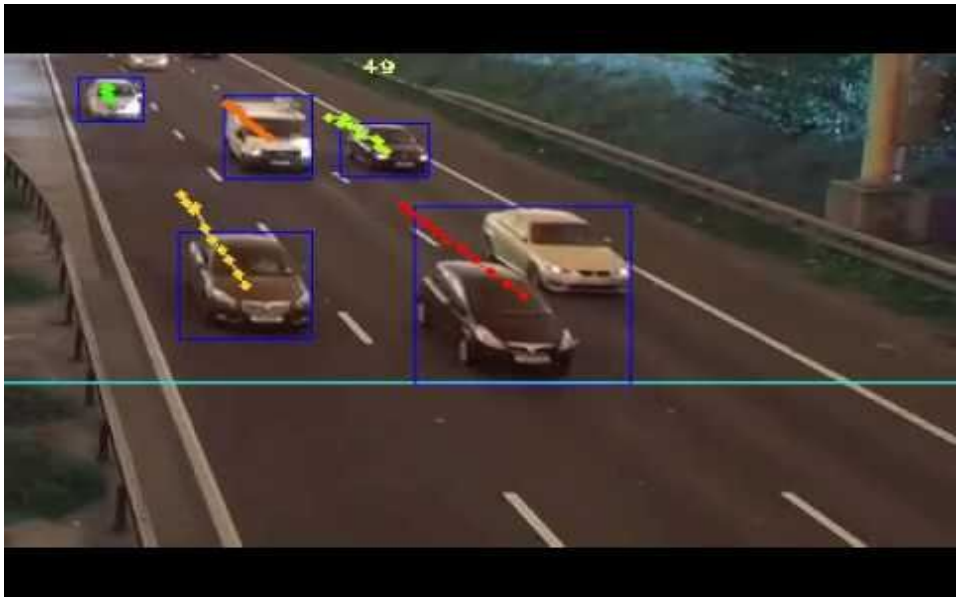
Ters sütun karıştırma adımında ise yine sütun karıştırma adımına benzer işlemler yapılır ancak bu kez aşağıdaki matris kullanılır.

0x0e	0x0b	0x0d	0x09
0x09	0x0e	0x0b	0x0d
0x0d	0x09	0x0e	0x0b
0x0b	0x0d	0x09	0x0e

Tablo 1.3.8.3 Ters Sütun Karıştırma

1.4. Görüntüde Nesne Takibi

Görüntü işleme, görüntü içerisindeki bilgilerin okunması, çıkarılması ve bu bilgilerin ihtiyaca göre işlenmesi için kullanılan bir yöntemdir. Görüntü içerisinde bulunan nesnelerin tespiti, tanımlanması, sınıflandırılması ve takibi gibi ihtiyaçları karşılayacak birçok yöntem geliştirilmiştir. Özellikle görüntülerdeki hedef nesnenin bulunması ve ileriki zaman dilimlerinde bu nesnenin kaybedilmemesi hatta daha sonra bu nesnenin çeşitli öğrenme yöntemleriyle tanınması ve belirlenen nesnenin ne olduğunun tahmin edilmesi birçok alandaki uygulamalarda sıklıkla kullanılmaktadır.

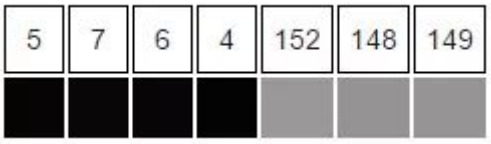


Şekil 1.4.1.

Takip edilecek nesnenin değişken bir ortam içinde bulunması nesne takibi ve analizini zorlaştıran temel problemdir. Bu problemleri çözmek ve nesnenin başarılı bir şekilde takip edilmesi için birçok farklı yöntem geliştirilmiştir. *Bu çalışmada nesne takibi için kenar çıkarma algoritmalarından yararlanarak* nesnenin ileriki zaman diliminde kenar benzerliklerine dayalı bir puanlama algoritması ile nesne takibinin sağlanması çalışması yapıldı.

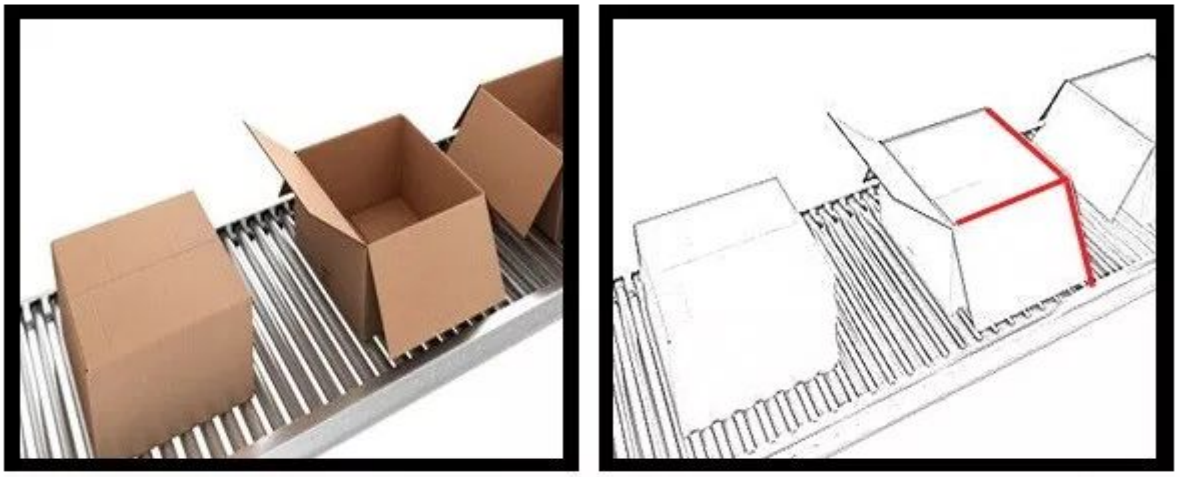
1.4.1 Görüntüdeki Kenarların Çıkarılması

Görüntüde kenar tespiti yapan algoritmalar en temel anlatımıyla, görüntü üzerindeki piksellerin renk değerlerinin birbirinden farklılaşması ile belirlenmesidir.



Şekil 1.4.1.1.

Şekil 1.4.1.1'e baktığınızda farklılaşmanın nereden başladığını tahmin edebilir misiniz? Gördüğünüz üzere 4 ve 152 numaralı matris elemanları arasında keskin bir renk geçişi olmuş, bu renk geçişi iki farklı nesnenin başlangıç ve bitiş çizgilerini ifade edebilir. İşte bu geçişler bizim için kenar çizgilerini ifade etmektedir. Örnek matrisimiz görüldüğü üzere gri (Gray) renk uzayına sahip, bu renk uzayında matris elemanları yani pikseller 0-255 arasında renk değerlerine sahiptir bu renk değerlerinde çalışıp görüntümüzü ikili(binary) yani 0-1 değerlerine sahip bir matrise çevirmek işimizi çok daha kolaylaştıracaktır.

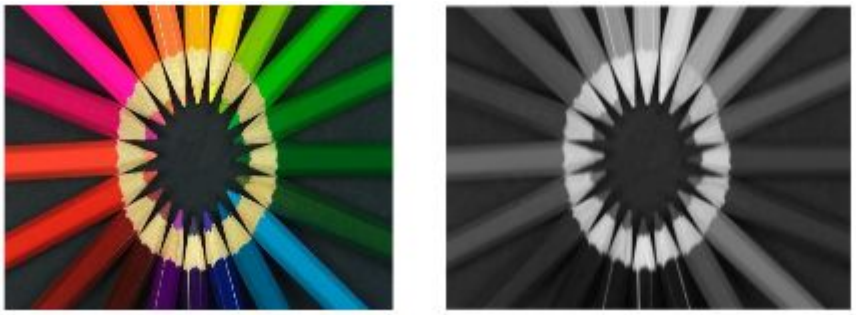


Şekil 1.4.1.2.

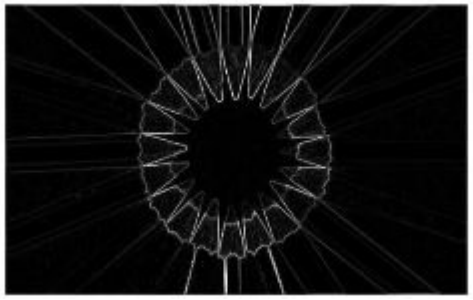
1.4.1.1 Canny Algoritması

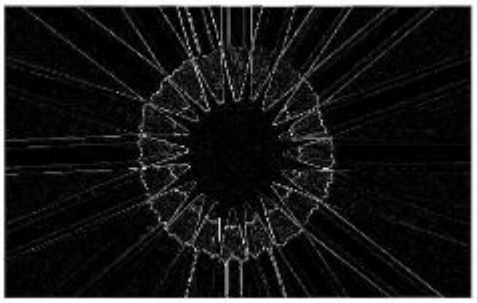
Bir çok kenar belirleme algoritması vardır ancak Canny algoritması verdiği başarılı sonuçlar neticesinde en sık kullanılandır. Canny kenar belirleme algoritması; görüntüde keskin olarak belirlenmiş kenarları bulmak için John F.Canny tarafından geliştirilmiş ve aşamaları olan bir algoritmadır. Kenar bulmada son derece etkin olarak kullanılır.

Algoritmanın aşamalarını sıralarsak; Öncelikle görüntünün gürültülerini azaltmak amacıyla Gaussian filtre kullanılır. Gaussian filtre dışında Mean ya da Medyan filtre de kullanılabilir. Daha sonra gradyan operatörü (kenar yönüne dik olan doğru) uygulanır. Bu şekilde görüntünün Gradyan büyüklüğü ve yönü hesaplanır. Kenarlar Non maxima baskılama kullanılarak incelemeye alınır ve ikili eşikleme uygulanır bu şekilde istenmeyen ayrıntılardan arındırma işlemi gerçekleştirilir. Güçlü-zayıf ayrımı yapıldıktan sonra baskılama uygulanır ve asıl kenarlarla görüntüye son hali verilir.

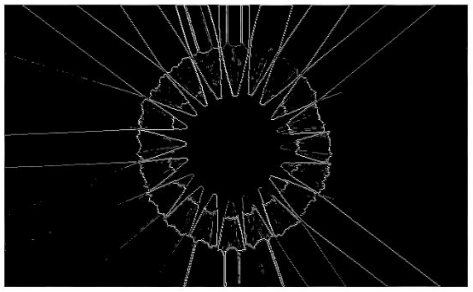


Şekil 1.4.1.1.1. Mean filtresi uygulanmış hali





Şekil 1.4.1.1.2. a-Gradyan görüntü b-Non maxima görüntü



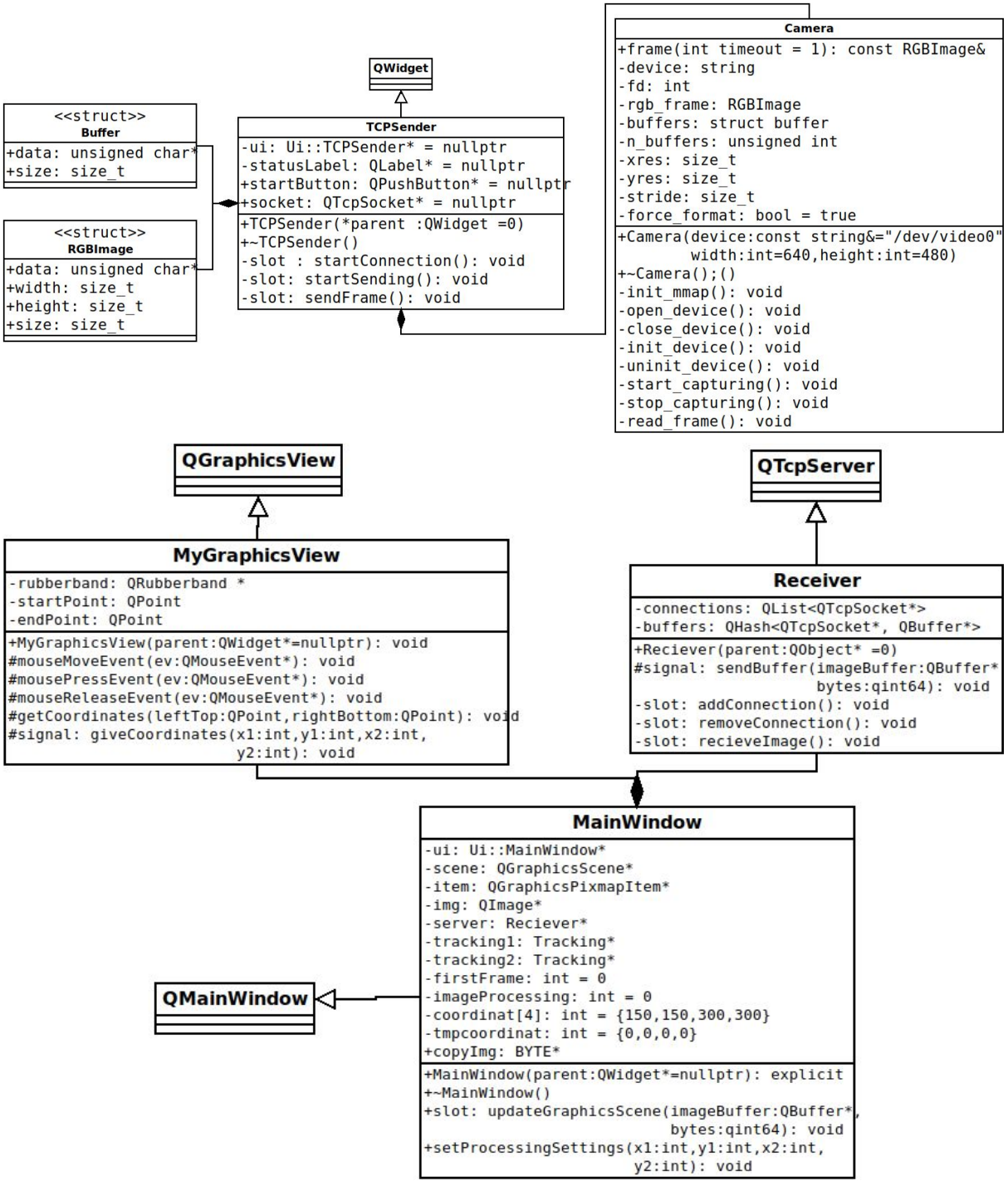
Şekil 1.4.1.1.3. İkili eşikleme ve kenar ayrımı

2. PROJE TASARIMI

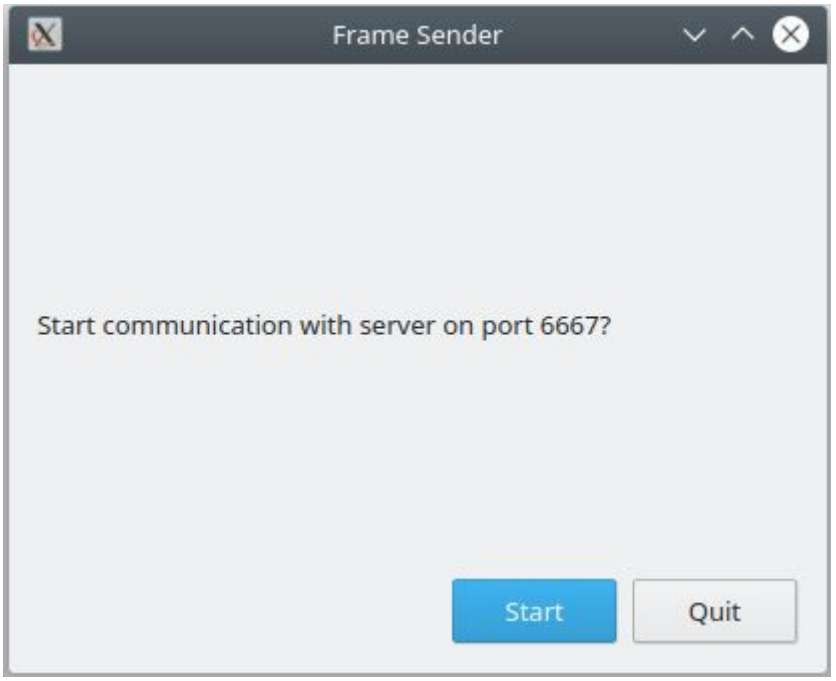
2.2. Mimari Tasarım

Projenin tasarımıyla alakalı uml diyagramları ve bazı kritik yapılar aşağıdaki bölümlerde detaylı bir biçimde açıklanmıştır.

2.2.1. Kullanıcı Arabirimi ve İletim



Gönderici ve Alıcı uygulamalarında, gönderici yani client tarafının arayüzü mümkün olduğunca basit tutuldu. Bu sebeple arayüzde iletimi başlatan ve pencereyi kapatmak üzere 2 adet buton ve 1 adet bilgilendirme label'ı bulunmaktadır.



Şekil 2.2.1.1. Gönderici arayüzü

Programda butonlar için QPushButton sınıfı kullanıldı. Buttonların tıklandığında yapacakları eylemleri belirlemek için yine Qt'nin callback fonksiyonlara bir alternatif olan Signal ve Slot sistemi kullanıldı. Bu sistem Qt bileşenlerinin veya fonksiyonların birbirlerini tetiklemesini sağlıyor.

Öncelikle aşağıdaki kod parçasında bu buttonların nasıl oluşturulduğunu ve arayüze eklendiğini inceleyelim.

```
startButton = new QPushButton(tr("&Start"));
auto quitButton = new QPushButton(tr("&Quit"));
auto buttonBox = new QDialogButtonBox;
buttonBox->addButton(startButton, QDialogButtonBox::ActionRole);
buttonBox->addButton(quitButton, QDialogButtonBox::RejectRole);
auto mainLayout = new QVBoxLayout;
mainLayout->addWidget(statusLabel);
mainLayout->addWidget(buttonBox);
setLayout(mainLayout);
```

Burada görüldüğü üzere Button'lar oluşturulup daha sonra buttonbox bileşenine eklendikten sonra layout'a eklenip ekranda gösteriliyor.

Button'ların tıklandığında gereken işlemleri yapabilmesi için *connect* yardımıyla bir slota bağlanması gerekir. Aşağıda bu işlemi görmekteyiz.

```
connect(startButton, &QPushButton::clicked, this, &TCPSender::startConnection);
connect(quitButton, &QPushButton::clicked, this, &TCPSender::close);
```

Burada;

1) *startButton* signal'in geleceği öğe
2) *&QPushButton::clicked* gönderilecek signal
3) *this* slotun gönderileceği öğe olan *TCPSender* classı
4) *&TCPSender::startConnection* ise *TCPSender* class'ı içerisinde bulunan ve server-client bağlantısını başlatan slot fonksiyon olmaktadır.

Böylece basit bir işlem ile bir signal ile slot birleştirilmiş oldu. Bu sayede *quitButton* tıklandığında *startConnection()* fonksiyonu çalışacak ve server ile bağlantı kurulacaktır.

```
void TCPSender::startConnection()
{
    if (socket->state() == QAbstractSocket::UnconnectedState)
    {
        socket->connectToHost("127.0.0.1", 6667, QIODevice::WriteOnly);
    }
    else
    {
        socket->disconnectFromHost();
    }
}
```

Yukarıdaki slot fonksiyon tetiklendiğinde *QTcpSocket* sınıfının *socket* nesnesi ile *connectToHost* fonksiyonuna erişerek "127.0.0.1" IP adresinin 6667 portuna bağlanılıp server tarafı bu bağlantıyı kabul ettiğinde veri gönderilir. Burada *socket* nesnesi server'a bağlandığında bir *connected()* signali üretilmiş olur. Bu sinyali *startSending()* fonksiyonuna bağlayarak iletişim başladığında frame gönderimi başlatılır. Bu sayede server ile iletişime geçtiğimizi anlayıp verileri gönderilir. Verileri saniyede 24 kere göndermek için *QTimer* kullanılmaktadır.

```
connect(socket, SIGNAL(connected()), SLOT(startSending()));
connect(&timer, &QTimer::timeout, this, &TCPSender::sendFrame);
```

```
void TCPSender::startSending()
{
    startButton->setEnabled(false);
    timer.start(1000/24);
}
```

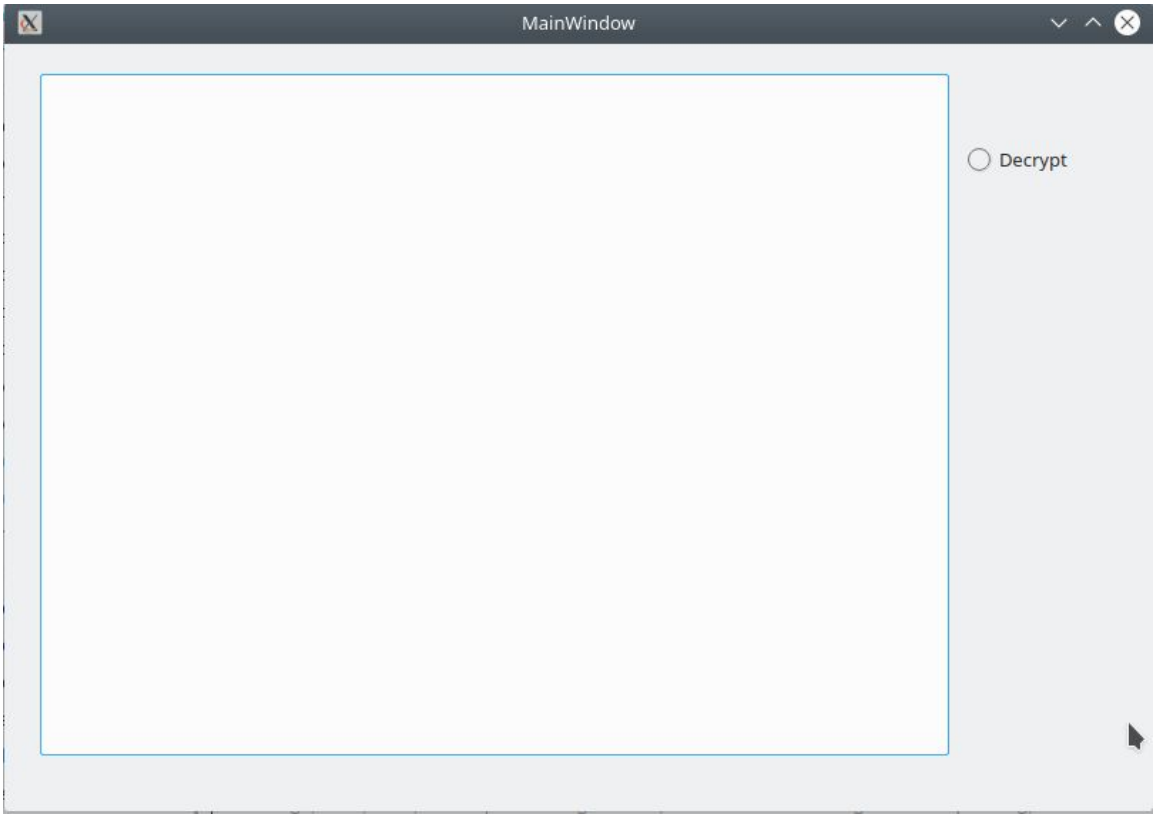
```
void TCPSender::sendFrame()
{
    if(socket->state()==QAbstractSocket::ConnectedState){
        auto frame = camera->frame();
        image = new QImage(frame.data,XRES,YRES,QImage::Format_RGB888);
        QImage im = image->convertToFormat(QImage::Format_Grayscale8);
        QByteArray ba;
        QBuffer buffer(&ba);
        im.save(&buffer,"BMP");
        uint8_t key[16] = {'a', 'y', 's', 'e', 't', 'a', 't', 'i', 'l', 'e',
'c', 'i', 'k', 's','i', 'n'};
        uint8_t* dataPointer = (uint8_t*)ba.data();
```

```
        dataPointer+=1078;
        Encryption en(dataPointer,key,128);
        for(int i=0;i<640*480;i+=16)
        {
            en.fastEncrypt();
            dataPointer+=16;
            en.setMessage(dataPointer);
        }
        socket->write(ba);
        delete image;
        qDebug()<<"image sent. Buffer:"<<buffer.size();
    }
     startConnection();
}
```

Timer'ın çalışmasıyla birlikte yukarıdaki fonksiyon saniyede 24 kez çalışıp görüntüyü karşı tarafa göndermiş olur. Burada önce *camera*'dan aldığımız frame'i *QImage* tipine dönüştürüp, grayscale görüntüye çevirip, boyutunu azalttıktan sonra *QByteArray'a* yazılan görüntü *QBuffer* nesnesiyle birlikte kaydedildikten sonra şifrelenip karşı tarafa gönderilir. Burada projeyi ubuntu ortamında geliştirdiğimizden dolayı görüntüyü kameradan *video4linux* kütüphanesi ile almaktayız. Dolayısıyla frame'i aldığımız camera değişkeni de aslında v4l kütüphanesini kullanıp yazılan fonksiyonların bulunduğu *Camera* sınıfından türetilmiş bir nesnedir. Bu nesne linuxta

```
camera = new Camera("/dev/video0", XRES, YRES);
```

şeklinde önce aygıtı sonra da sabit genişlik ve yükseklik değerleri verilerek tanımlanabilir.



Şekil 2.2.1.2 Alıcı arabirimi

Alıcı arabiriminin arkaplan fonksiyonları için *QTcpServer* sınıfı kalıtımlandı. Burada List ve Hash veri yapıları kullanılarak gelen her bir frame için aşağıdaki şekilde *QList ve QHash* sınıflarından faydalanılarak yeni bir connection ve buffer oluşturuldu.

```
QList<QTcpSocket*> connections;
QHash<QTcpSocket*, QBuffer*> buffers;
```

Gelen her bir connection'ı *connections* listeye eklemek için constructor içerisinde *newConnection* signali ile *addConnection* fonksiyon slotu birbirine bağlandı.

```
Reciever::Reciever(QObject* parent): QTcpServer(parent)
{
    connect(this, SIGNAL(newConnection()), this, SLOT(addConnection()));
}
void Reciever::addConnection()
{
    QTcpSocket* connection = nextPendingConnection();
    connections.append(connection);
    QBuffer* buffer = new QBuffer(this);
    buffer->open(QIODevice::ReadWrite);
    buffers.insert(connection, buffer);
    connect(connection, SIGNAL(disconnected()), SLOT(removeConnection()));
}
```

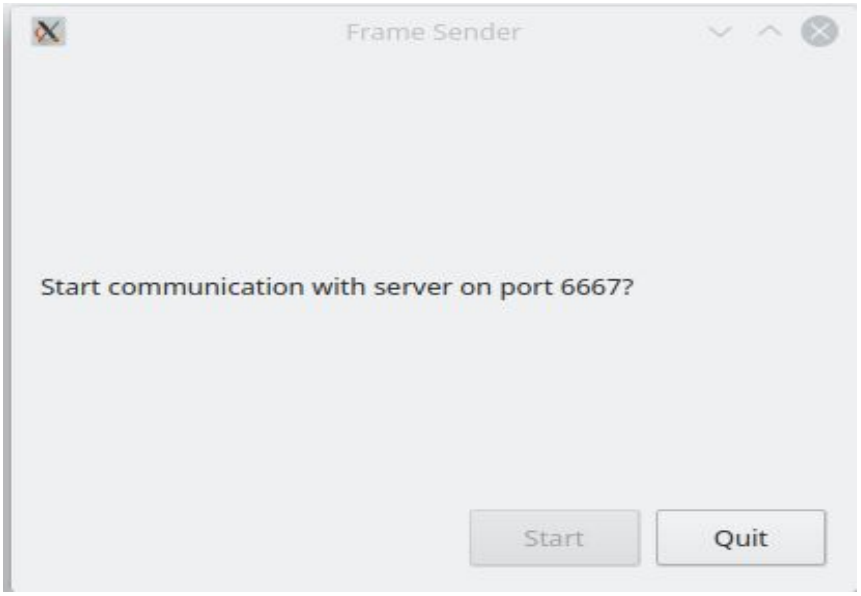
```
void Reciever::removeConnection()
{
    receiveImage();
    QTcpSocket *socket = static_cast<QTcpSocket*>(sender());
    QBuffer* buffer = buffers.take(socket);
    buffer->close();
    buffer->deleteLater();
    connections.removeAll(socket);
    socket->deleteLater();
}
void Reciever::receiveImage()
{
    QTcpSocket* socket = static_cast<QTcpSocket*>(sender());
    QBuffer* buffer = buffers.value(socket);
    qint64 bytes = buffer->write(socket->readAll());
    emit sendBuffer(buffer,bytes);
}
```

addConnection ile *removeConnection* birbirine bağlandıktan sonra önceki bölümde buffer yardımı ile gönderdiğimiz görüntü yine buffera okunarak *emit sendBuffer(buffer,bytes)* sinyali ile *MainWindow* sınıfına gönderilir. Böylece sınıflar arası iletişimi de sağlamış oluruz. Ana arayüz sınıfına gelen frame burada önce deşifre edilip ardından Qt Creator'in designer'ından eklediğimiz *QGraphicsView'a* promote edilmiş *myGraphicsView* classında gösterilir. Kullanıcı mouse ile seçtiği alan ile diğer bölümlerde anlatılan görüntü işleme işlemleri başlar.

Mouse ile seçme ve seçilen alanı signal ile gönderme işlemi de aşağıdaki şekilde halledilir.

```
void MyGraphicsView::mousePressEvent(QMouseEvent *ev)
{
    startPoint = ev->pos();
      qDebug()<<"START POINT!: "<<startPoint;
     if(!rubberBand)
             rubberBand = new QRubberBand(QRubberBand::Rectangle, this);
        rubberBand->setGeometry(QRect(startPoint,QSize()));
        rubberBand->show();
        QGraphicsView::mousePressEvent(ev);
}
void MyGraphicsView::mouseMoveEvent(QMouseEvent *ev)
{
     rubberBand->setGeometry(QRect(startPoint,ev->pos()).normalized());
     QGraphicsView::mouseMoveEvent(ev);
}
void MyGraphicsView::mouseReleaseEvent(QMouseEvent *ev)
{
    endPoint = ev->pos();
    qDebug()<<"END POINT!: "<<endPoint;
    rubberBand->hide();
    update();
    getCoordinates(startPoint,endPoint);
    QGraphicsView::mouseReleaseEvent(ev);}
void MyGraphicsView::getCoordinates(QPoint leftTop, QPoint rightBottom)
{
    leftTop=startPoint;
    rightBottom=endPoint;
if(((leftTop.x()<rightBottom.x())&&(leftTop.y()>rightBottom.y()))||((leftTop.x()
>rightBottom.x())&&(leftTop.y()<rightBottom.y())))
    {
        int tempY=leftTop.y();
        leftTop.setY(rightBottom.y());
        rightBottom.setY(tempY);
    }
    emit
giveCoordinates(leftTop.x(),leftTop.y(),rightBottom.x(),rightBottom.y());
}
```

Arayüzlerin son hali aşağıdaki şekildedir.



Şekil 2.2.1.3 Gönderici arayüz

Şekil 2.2.1.4 Alıcı arayüz

2.2.2. AES ile Görüntü Şifreleme

Bölüm 1.3. üzerinde AES Algoritması detaylı bir biçimde anlatılmıştır. Algoritma incelendikten sonra üç farklı alt parçaya ayrılabileceği açıkca görülmektedir. Bunlar şifreleme, deşifreleme, ve anahtar üretecidir. Ayrıca algoritmayı hızlandırmak için daha önceden hesaplanıp tablo haline getirilmiş verileri de bir sınıf ile birlikte sunduğumuzda toplam dört sınıf ile tüm algoritmayı gerçekleştirmiş oluyoruz.

Aşağıda AES Algoritmasını mümkün tüm anahtar uzunluklarıyla birlikte gerçekleştirmiş olduğumuz projenin Uml Diyagramı bulunmaktadır. Dikkat edilirse Encryption sınıfında byte değiştirme, satır kaydırma ve sütun karıştırma gibi alt işlevler bulunmamakta, tüm bu işlemler *fastEncrypt()* metodu altında toplu bir şekilde gerçekleştirilmekte böylece algoritmik karmaşıklık ciddi derecede azalmaktadır.

[9] numaralı bağlantıda AES Algoritmasının performansını artırmak bir dizi çalışmalar yapılmıştır. Bunlardan en önemlisi sütun karıştırma işlemi için kullandığımız tabloların bayt değiştirme işlemi için kullandığımız tablolarla birleştirilmesidir. Aşağıda matematiksel ifadesi verilen bu işlem tüm şifreleme işlemilerini tek döngüde gerçekleştirmeye imkan sağlayarak *fastEncrypt()* metodunun en önemli yapı taşı olmuştur.

subbyte(), *bayt değiştirme fonksiyonu*
*mul*3(), *mul*2(), *sütun karıştırma için çarpım fonksiyonları*
$f2(x) = mul2(subbyte())$
$f3(x) = mul3(subbyte())$

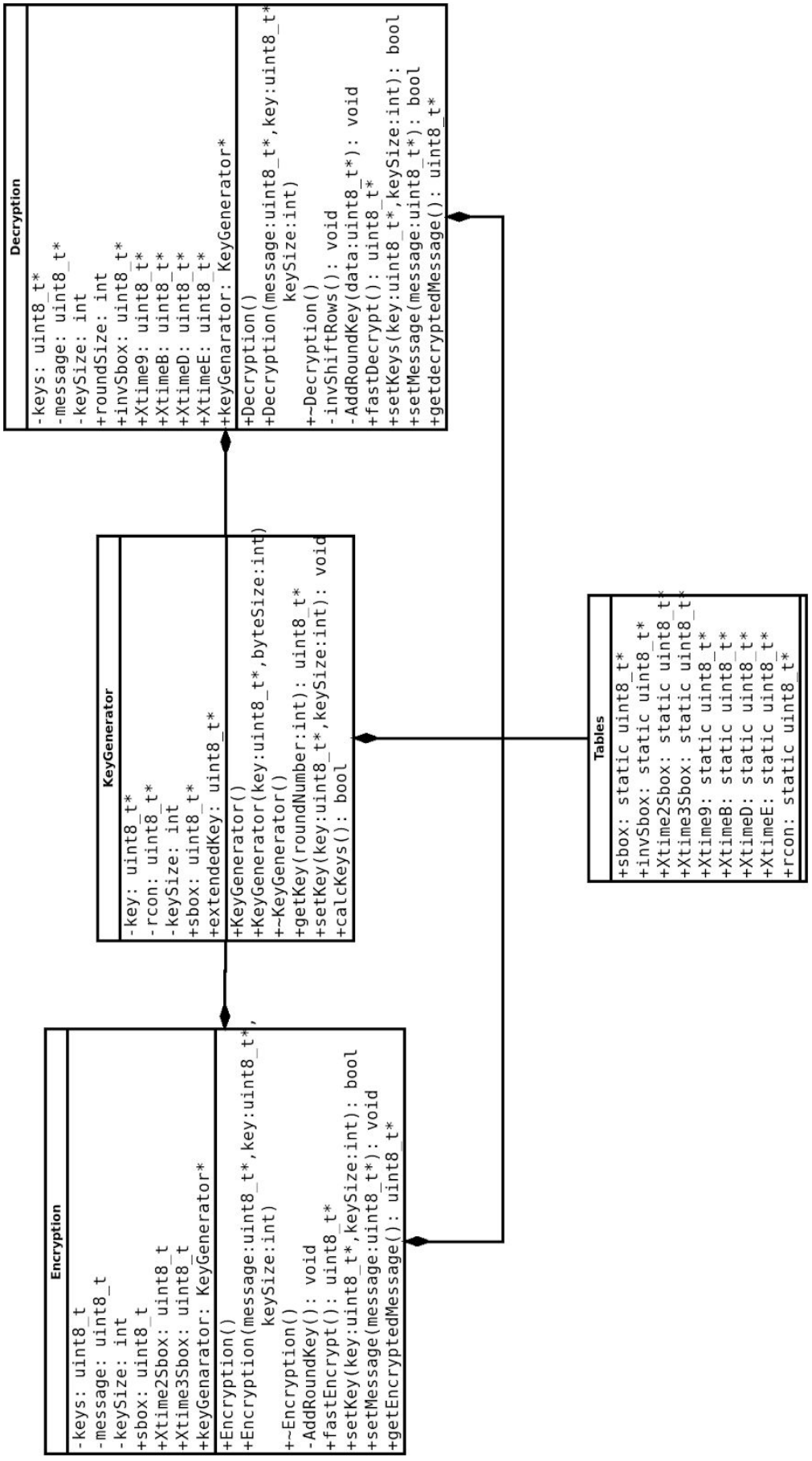
Bir sütun için bayt değiştirme, satır kaydırma ve sütun karıştırma adımlarının nasıl tek seferde yapıldığını aşağıdaki kod parçası üzerinde inceleyelim.

```
// Xtime2Sbox = mul2(subbyte())
// Xtime3Sbox = mul3(subbyte())
temp[0] = Xtime2Sbox[message[0]] ^ Xtime3Sbox[message[5]] ^
          sbox[message[10]] ^ sbox[message[15]];
temp[1] = sbox[message[0]] ^ Xtime2Sbox[message[5]] ^
          Xtime3Sbox[message[10]] ^ sbox[message[15]];
temp[2] = sbox[message[0]] ^ sbox[message[5]] ^ Xtime2Sbox[message[10]] ^
          Xtime3Sbox[message[15]];
temp[3] = Xtime3Sbox[message[0]] ^ sbox[message[5]] ^ sbox[message[10]] ^
          Xtime2Sbox[message[15]];
```

Tablo 2.2.2.1. Hızlı şifreleme

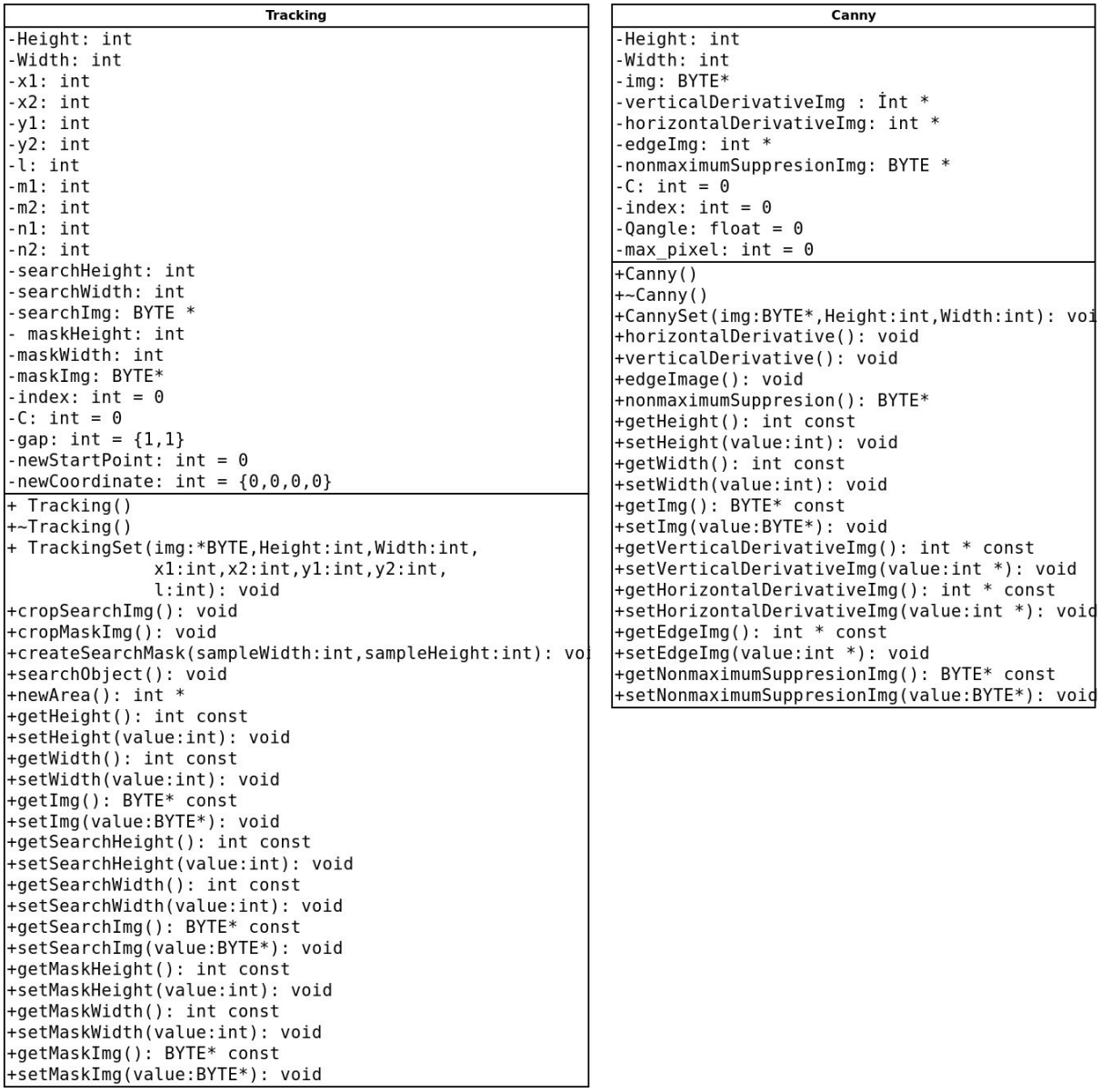
Standart algoritma takip edilse algoritmik karmaşıklığı O(3n) olması gereken adımlar bu yöntem sayesinde O(n) değerine düşmüştür.

Uml diyagramı üzerindeki ilişkileri yakından incelemek gerekirse Encryption ve Decryption sınıfları KeyGenerator ve Tables sınıfları ile kompozisyon ilişkisine sahiptir. KeyGenerator sınıfı ise Tables sınıfı ile ayrıca kompozisyon ilişkisine sahiptir.



Şekil 2.2.2.1. AES şifreleme modülü uml diyagramı

2.2.3. Nesne Takibi



Şekil 2.2.3.1

Yukarıdaki şekilde projede nesne takibi için kullanılan algoritmaların UML diyagramını görüyorsunuz.

2.2.3.1 Görüntüde kenar bulma ve binary görüntü elde etme

```
void Canny::horizontalDerivative()
{
   index=0;
   for (int i = 1; i < Height - 1; i++)
       {
           for (int j = 1; j < Width - 1; j++)
           {
               C = (i*Width + j);
               // 1 0 -1 -  2 0 -2 -  1 0 -1  maskesini gezidiriyoruz
// kenar yönü dikeyde
```

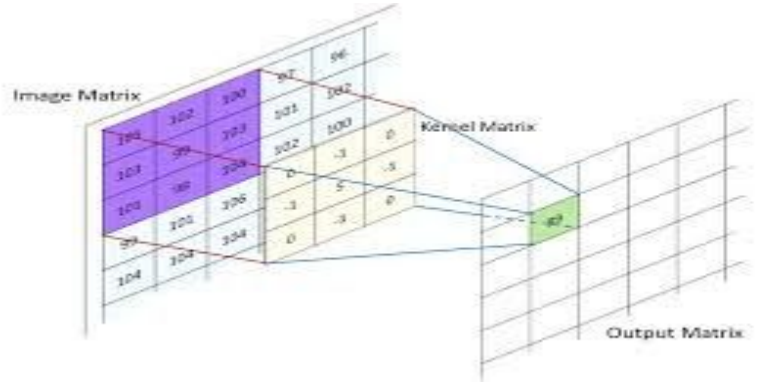
```
      horizontalDerivativeImg[index] =
(1 * img[(C - Width - 1)] + 0 * img[(C - Width)] + (-1) * img[(C - Width + 1)]
+ 2* img[(C - 1)] + 0 * img[C] + (-2) * img[(C + 1)]
+ 1* img[(C + Width - 1)] + 0 * img[(C + Width)] + (-1) * img[(C + Width + 1)]);
       index++;
           }
       }
}

void Canny::verticalDerivative()
{
   index=0;
   for (int i = 1; i < Height - 1; i++)
       {
           for (int j = 1; j < Width - 1; j++)
           {
               C = (i*Width + j);
               // 1 2 1  -  0 0 0 -  -1 -2 -1  maskesini gezidiriyoruz
//kenar yönü yatayda
      verticalDerivativeImg[index] =
(1 * img[(C - Width - 1)] + 2 * img[(C - Width)] + 1 * img[(C - Width + 1)]
+ 0 * img[(C - 1)] + 0 * img[C] + 0 * img[(C + 1)]
+(-1)* img[(C + Width - 1)] + (-2) * img[(C + Width)] + (-1)* img[(C+Width+1)]);
       index++;
           }
       }

}
```

Tablo 2.2.3.1.1.

Tablo 2.2.3.1.1'de bu projede kullanılan, gri görüntünün yatayda ve dikeyde türev alma işlemini yapan kod parçası görülmetekdir. İlk fonksiyon(horizontalDerivative) görüntüde 1,0,1,2,0,-2,1,0,-1 değerlerini içeren 3x3'lük matrisi gezdirir bu işleme maske gezdirmede denir. Bu maske gezdirme işleminin amacı maskenin ortasına denk gelen pikselin sağ ve solundaki piksellerin arasındaki değer farkına göre değerlendirmektir eğer sağındaki ve solundaki değerlerin farkı maksimum ise (yani biri 0(siyah) diğeri 255(beyaz)) ;

```
(1*255 + 2*255 + 1*255) - (-1*0 + (-2*0) + (-1*0)) =1020
```

Merkezdeki piksele karşılık 1020 değeri verilir bunun anlamı bu piksel'in en yüksek ihtimalle kenar olduğu anlamına gelir aynı zamanda bu değeler negatifde çıkabilir ama önemli olan mutlak büyüklükleridir. Benzer işlem ikinci fonksiyonda da(verticalDerivative) yapılır aralarındaki fark görüntüde gezdirdikleri maske değerleridir bu fonksiyon ise 1,2,1,0,0,0,-1,-2,-1 değerlerine sahip 3x3'lük matris gezdirir. Bu matrislerin farklı olmasının sebebine gelirsek, ilk fonksiyonda daha önceden de söylendiği gibi hedef pikselin sağ ve sol komşuluklarının farkı değerlendiriliyor, yani yatayda türev alınarak dikey kenarlar tespit ediliyordu. Bu fonksiyon da ise hedef pikselin üst ve alt komşuluklarının farkı değerlendirilerek dikeyde türev alınıyor ve yataydaki kenarlar tespit ediliyor.



Şekil 2.2.3.1.1

```
void Canny::edgeImage()
{
   index=0;

   for (int i = 0; i < Height - 2; i++)
       {
           for (int j = 0; j < Width -2; j++)
           {
               index = (i*(Width-2) + j);
               // 2 maske sonucunu topluyoruz  not: 255den daha buyuk degeler
olabilir max 1020 olabilir
            edgeImg[index]  =abs(verticalDerivativeImg[index])
                             +abs(horizontalDerivativeImg[index]);
           }
       }
}
```

Tablo 2.2.3.1.2

Yatay ve dikey doğrultuda bulunan kenarlar Tablo 2.2.3.1.2'deki fonksiyondaki gibi toplanır ve birleştirilir ama tabiki birleştirilirken mutlak değerleri alınır, çünkü iki maskenin sonuçlarında da negatif değerler olabilir ama kenar özellikleri mutlak büyüklüklerine bakılarak değerlendirilir. Edgeimage'de görüntüdeki kenar her pikselin yatay ve dikeydeki komşuluklarının farkına göre kenar özelliklerini belirten pozitif sayılar tutulur.

```
  Qangle = atan2( verticalDerivativeImg[C] , horizontalDerivativeImg[C])*180/PI;
```

Tablo 2.2.3.1.3.

Bundan sonra edgeimage matrisinde her pikselin gradient doğrusu(kenar doğrultusuna dik doğru) tablo 2.2.3.1.3'deki kod parçasındaki gibi bulunur. Bulunan bu doğrultuya bakılarak 0, 45, 90, 135, 180, 225, 270, 315, 360 derecelik aralıklarla hedef pikselin gradient doğrusu bulunur ve bu doğrultudaki pikseller ile hedef piksel, büyüklüklerine göre kıyaslanır ve eğer hedef piksel gradient doğrultusundaki komşu değerlerinden büyük ise aynen kalır ama eğer daha büyük bir değer var ise hedef piksel 0'a setlenir. Bunun amacı görüntüdeki gürültüyü azaltmak ve daha belirgin kenarları ortaya çıkarmaktır.

Bunun sonucunda non-maximum görüntü elde edilir. Bu projede non-maximum görüntü bulunurken edgeimage her aşamada güncellenerek, yani hedef piksel kontrol edilip

güncellendikten sonra sıradaki hedef pikselin komşulukları kıyaslanmıştır bu şekilde gürültülerin daha fazla azaldığı görülmüştür.

```
//goruntuyu binary cevirme
   for (int i = 0; i < Height - 2; i++)
       {
           for (int j = 0; j < Width - 2; j++)
           {
              C = i * (Width - 2) + j;
               if (edgeImg[C] <= max_pixel/4)
               {
                   nonmaximumSuppresionImg[C] = 0;
               }
               else
               {
                   nonmaximumSuppresionImg[C] =255;
               }
           }
       }
```

Tablo 2.2.3.1.4.

Son olarak non-maximum görüntüdeki maksimum değerli piksel belirlenir ve Tablo 2.2.3.1.4'deki kod gibi maksimum değerli pikselin dörtte birine eşit ve küçük değerli pikseller sıfıra setlenir geri kalan piksel değerleri ise 255'e setlenerek binary görüntü elde edilir. Burada daha verimli bir şekilde Tlow ve Thigh değerleri belirlenip daha sonra kenar yönündeki komşuluklara bakılarak binary görüntü elde edilebilirdi ancak bu projede gerçek zamanlı şifreleme ve nesne takibi yapılacağı için bu işlemlere gerek duyulmamıştır. Bu işlemlerden sonra verilen gri görüntünün kenarlarının bulunması gerçekleştirilmiştir artık kenar ilişkilerine göre nesne takibi için gerekli olan giriş değerleri alınabilecektir.

2.2.3.1. Görüntüdeki Kenarlara Göre Nesne Takibi

Gelen görüntülerin kenarlarını bulup binary görüntüyü elde ettikten sonra görüntüde seçilen nesnenin takibini yapacak algoritma çalışmaya başlıyor bu algoritmanın çalışma prensibi şu şekilde;

```
void cropSearchImg();
void cropMaskImg();
void createSearchMask(int sampleWidth, int sampleHeight);
void searchObject();
int *newArea();
```

Tablo 2.2.3.1.1.

Tablo 2.2.3.1.1'de görüldüğü gibi beş temel fonksiyon var bunlar sırası ile gelen görüntü ve koordinatlardan görüntüde arama yapılacak alan, görüntüde aranacak nesne, görüntüde aranacak olan nesneden ne sıklıkla örnek alınacağı ve son olarak arama işleminin sonucu ile elde edilen koordinatlar. Bu yeni koordinatlar verilen görüntüdeki aranmasını istenen nesnenin yine aynı görüntüdeki koordinatlardır.

```
void Tracking::TrackingSet(BYTE *img, int Height, int Width,int x1,int y1,int
x2,int y2,int l)
```

Tablo 2.2.3.1.2.

Tablo 2.2.3.1.2'de arama algoritmasının çalışması için gereken değerleri 'Tracking' nesnesine veren 'TrackingSet' fonksiyonu görülmekte burada 'img' arama yapılacak görüntü ve sonraki iki değişken ise yine bu görüntünün boyutlarıdır. Sonraki dört parametre ise aranmasını istediğimiz nesnenin koordinatlarıdır(bu koordinatlar bir önceki frame'den gelir) ve son olarak 'l' parametresi ise verilen nesnenin görüntüde kaçar piksel aralıkta aranacağını belirler. Yani verilen koordinatlar ve 'l' değişkeni ile nesnenin aranacağı görüntü aralığı belirlenir.

Bütün bu parametreler verildikten sonra gerekli fonksiyonlar çağrılarak görüntüden, arama yapılacak yeni görüntü alanı ve bu görüntüde aranacak nesne çıkarılır.

```
void Tracking::createSearchMask(int sampleWidth, int sampleHeight)
{
   if(sampleHeight>=maskHeight)
       sampleHeight=maskHeight;
   if(sampleWidth>=maskWidth)
       sampleWidth=maskWidth;
   gap[0] = round(maskHeight/sampleHeight);
   gap[1] = round(maskWidth/sampleWidth);
}
```

Tablo 2.2.3.1.3.

Tüm bu işlemlerden sonra algoritmada sıra görüntüden çıkarılan bu yeni iki alanda arama işleminin yapılmasına gelir. Arama işlemi başlamadan önce kullanıcı tarafından aranacak nesnenin görüntüntüsünden ne sıklıkla örnek alınacağı belirlenir bu işlem tablo 2.2.3.2.3'de görüldüğü gibi yapılır. Bu fonksiyona verilen parametreler sırasıyla genişlikten kaçar örnek ve yükseklikten kaçar örnek alınacağı belirlenerek verilir.

```
void Tracking::searchObject()
{
   int score=0;
   int maxscore=0;
   int StartPoint=0;
   newStartPoint=0;
   for (int i = 0; i < (searchHeight-maskHeight)+1; i++)
           for (int j  = 0; j < (searchWidth-maskWidth)+1; j++)
           {
              score=0;
              for (int k = 0; k< maskHeight; k+=gap[0])
                  for (int h = 0; h< maskWidth; h+=gap[1])
                  {
                      if(h==0 && k==0)
                          StartPoint=(i* searchWidth + j)+(k * maskWidth + h);

                 score+=
                        (searchImg[(i* searchWidth + j)+(k * searchWidth + h)]
                          == maskImg [k * maskWidth + h]);
                  }
```

```
                if(score>maxscore)
                {
                    newStartPoint=StartPoint;
                    maxscore=score;
                 }
             }
}
```

Tablo 2.2.3.1.4.

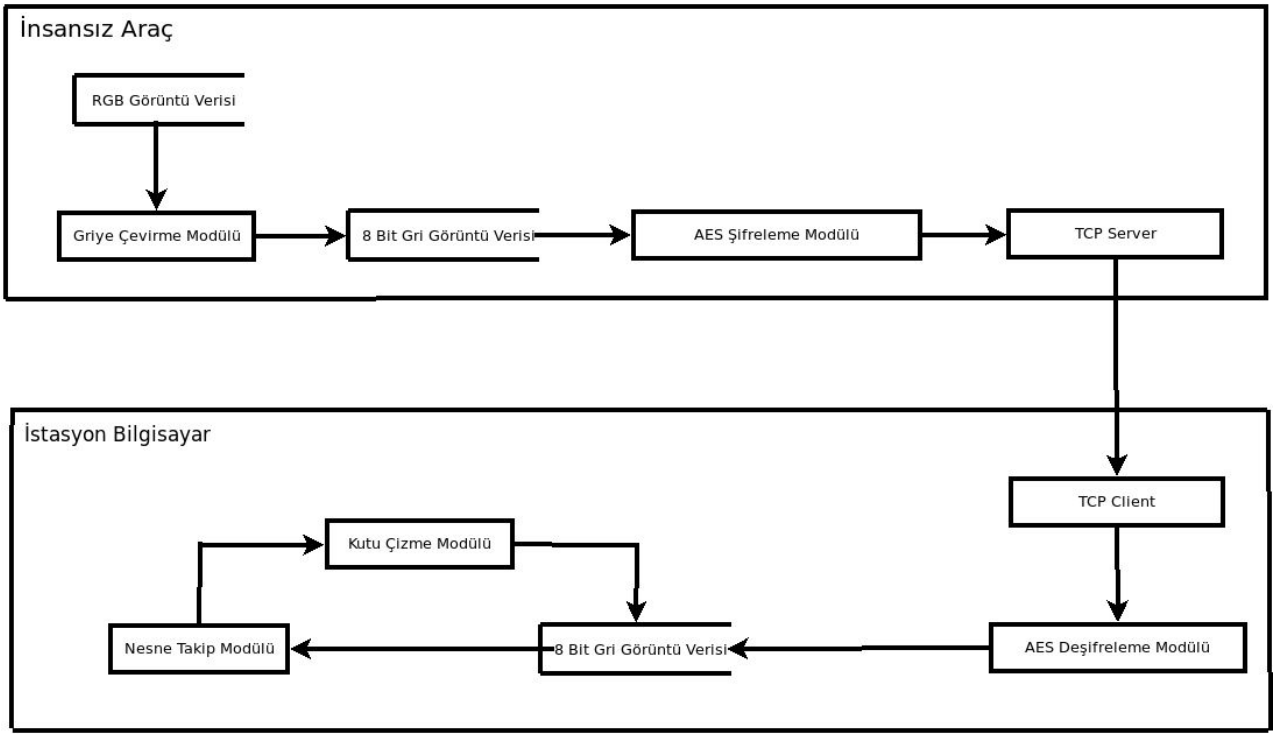
Arama işlemini yapan fonksiyon tablo 2.2.3.1.4'de gösterilmiştir. Aranacak nesne, aranacak görüntü aralığı ve son olarak hangi sıklıkla benzerliklerine bakılacağı belirlendikten sonra, aranan nesne, aranacak görüntü alanında bir maske gezdirme işlemi yaparak gezer ama bu gezme işlemini yaparken her bir değerin kıyaslanması yapılmaz onun yerine kullanıcı tarafından belirlenen aralıklarda piksel kıyaslamaları yapılır. Yapılan bu kıyaslamalar sonucunda her bir bölgedeki eşleşen pikseller bir puan anlamına gelir ve arama sonucunda aranan nesnenin, aranan bölgede en çok nerede eşleşme sağlamışsa orada maksimum puana sahip olur ve bu puanın maksimum olduğu yer aranan nesnenin yeni başlangıç noktasıdır.

```
int *Tracking::newArea()
{
   int searchY1= newStartPoint/searchWidth;
   int searchX1=newStartPoint-(searchY1*searchWidth);

   newCoordinate[0]=(m1)+searchX1;
   newCoordinate[1]=(n1)+searchY1;
   newCoordinate[2]=newCoordinate[0]+(x2-x1);
   newCoordinate[3]=newCoordinate[1]+(y2-y1);
   return newCoordinate;
}
```

Tablo 2.2.3.1.4.

Son olarak algoritma, arama işlemi sonucunda nesnenin yeri başlangıç noktasından faydalanarak, tablo 2.2.3.1.4'deki fonksiyon ile nesnenin algoritmaya ilk verilen görüntüdeki yeni koordinatlarını belirler.

2.3. Yapılan Çalışmalar



Şekil 2.3.1. Sistem akışı

Şekil 2.3.1 üzerinde görülen modüller gerçeklenmiş ve bir sistem ortaya konmuştur. Ortaya konan bu sistemde bazı eksikler olmakla birlikte projenin ilerleyen süreçlerinde eksiksiz bir görüntü takibi hedeflenmektedir. Bu modülleri açıklamak gerekirse v4l(video for linux) kütüphanesi yardımıyla kameradan RGB görüntü okunmuştur. Okunan görüntü QImage kütüphanesi ile 8 bitlik gri formata çevrilmiştir. Bu işlemin amacı hem iletilecek görüntü boyutunu düşürmek hem de görüntü işlemele algoritmalarının gri görüntüler üzerinde çalışması ve böylece daha hızlı sonuçlar üretmesidir. Alınan 8 bitlik görüntü AES 128 bit ile şifrelenmekte ve şifreli görüntü TCP ile istasyon bilgisayara gönderilmektedir. İstasyon tarafında ki istemci uygulamada yazılımın test edilmesi amaçlı “Decrypt” adlı bir checkbox konmuştur. Eğer checkbox seçili değilse istemci üzerinde şifreli görüntü gösterilmektedir. İstemci tarafından alınan görüntü şifresi çözüldükten sonra önce canny ile kenarları çıkarılır ardından kenarları bulunan görüntüler üzerinden bir sonraki frame ile arama yapılır. Bir sonraki framede bulunan en doğru alana yeni bir kutu çizilir.
Projede insansız araç ve istasyon bilgisayar, kişisel bilgisayarlar üzerinde simule edilmiştir. Şekil 2.2.1.3 ve Şekil 2.2.1.4 üzerinde yazılımların arayüzleri(sunucu,istemci) ile alakalı görüntüler görülebilir.

3. KAYNAKLAR

1) **K. Hanbay, H. Üzen, Nesne tespit ve takip metotları:Kapsamlı bir derleme, Türk Doğa ve Fen Dergisi, 6(2), pp. 40-49, 2017.**
2) ***Scott E. Umbaugh, 2005; Computer Imaging: Digital Image Analysis and Processing, A CRC Press Book, Taylor and Francis Group***
3) **M. Abomhara, O.O. Khalifa, O. Zakaria, A.A. Zaidan, B.B. Zaidan and A. Rame, 2010. Video Compression Techniques: An Overview. *Journal of Applied Sciences, 10: 1834-1840.***
4) ***M. Abomhara, Omar Zakaria, Othman O. Khalifa, "An Overview of Video Encryption Techniques," International Journal of Computer Theory and Engineering vol. 2, no. 1, pp. 103-110, 2010.***
5) ***Vaibhav Narawade Deepali P. Chaudhari, "Secure Multimedia Data using H.264/AVC and AES Algorithm" International Journal of Computer Science and Technology vol. 3, no. 1, pp.179-182, 2012.***
6) ***K. Hanbay, H. Üzen, Nesne tespit ve takip metotları:Kapsamlı bir derleme, Türk Doğa ve Fen Dergisi, 6(2), pp. 40-49, 2017.***
7) ***Zdenek Kalal, Krystian Mikolajczyk, Jiri Matas: Tracking-Learning-Detection. IEEE Trans. Pattern Anal. Mach. Intell. 34(7): 1409-1422 (2012)***
8) ***CHOKKALINGAM, B. (2013). Evaluation of TLD/Predator algorithm : Evaluation of TLD(aka Predator) algorithm used for real-time tracking of unknownobjects in a video stream from eye tracking perspective. (Dissertation). Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-142017***
9) ***https://code.google.com/archive/p/byte-oriented-aes/***
10) ***https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf***
11) **http://mesutpiskin.com/blog/opencv-ile-kenar-belirleme-algoritmalari.html**
12) **http://embedded.kocaeli.edu.tr/canny/**
13) **http://www.wikizeroo.net/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvQ2FubnlfZWRnZV9kZXRlY3Rvcg**
14) **https://wiki.qt.io/Qt_History**
15) **https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Rijndael_mix_columns.html**

STANDARTLAR ve KISITLAR FORMU

Projenin hazırlanmasında uyulan standart ve kısıtlarla ilgili olarak, aşağıdaki soruları cevaplayınız.

1. Projenizin tasarım boyutu nedir? (Yeni bir proje midir? Var olan bir projenin tekrarı mıdır? Bir projenin parçası mıdır? Sizin tasarımınız proje toplamının yüzde olarak ne kadarını oluşturmaktadır?)

Proje askeri alanda birçok farklı şekilde kendine yer buluyordur.

2. Projenizde bir mühendislik problemini kendiniz formüle edip, çözdünüz mü? Açıklayınız.

Projede görüntü takibi; bir önceki görüntü karesinden, o anki görüntü karesinde arama yaparak sağlanıyor. Bu doğrultuda ham görüntü yerine kenarları çıkarılmış görüntüde çalışmanın daha iyi sonuçlar vereceği düşünülmüştür. Ayrıca projede gerçek zamanlılık önemli olduğu için aranacak nesnenin görüntüsünden kullanıcının belirleyeceği aralıklar ile örnek alınması sağlanmıştır.

3. Önceki derslerde edindiğiniz hangi bilgi ve becerileri kullandınız?

Öncelikle programlama bilgileri olmak üzere, görüntü işleme ve bilgisayar ağları gibi derslerde edindiğimiz bilgileri de kullanma fırsatı yakaladık.

4. Kullandığınız veya dikkate aldığınız mühendislik standartları nelerdir? (Proje konunuzla ilgili olarak kullandığınız ve kullanılması gereken standartları burada kod ve isimleri ile sıralayınız).

Öncelikle fonksiyon ve değişken isimlendirmeleri için camel case sınıf isimlendirmeleri için ise pascal case kullanılıyor.

5. Kullandığınız veya dikkate aldığınız gerçekçi kısıtlar nelerdir? Lütfen boşlukları uygun yanıtlarla doldurunuz.

a) Ekonomi

Bu tip başarılı projeler dışarıya ihraç edilebilir, böylece ekonomiye olumlu katkıda bulunabilir.

b) Çevre sorunları:

Proje bu haliyle herhangi bir çevre sorununa çözüm üretmeyi hedeflememektedir.

c) Sürdürülebilirlik:

Proje askeri alanda sürekli geliştirilebilir bir projedir.

d) Üretilebilirlik:

Proje günümüz teknolojisiyle, desteklendiği müddetçe rahat bir biçimde üretilebilir.

e) Etik:

Askeri projelerin etik açılardan değerlendirilmesi oldukça zordur bu nedenle bu proje içinde net bir yargıda bulunamıyoruz.

f) Sağlık:

Projenin askeri amaçla kullanılıp, saldırıları önlemesi olası ölüm ve yaralanmaların önüne geçebileceği için bu proje sağlık ile bir miktar ilişkilidir.

g) Güvenlik:

Proje tam anlamıyla güvenlik ile ilişkilidir. Daha güvenli ve huzurlu ülkeler için proje katkı sağlayabilir.

h) Sosyal ve politik sorunlar:

Politik olarak ülkeler birbirlerine karşı çeşitli yaptırımlar uyguluyabiliyorlar. Günümüzde bunlardan en önemlileri arasında savunma sanayi gelmektedir. Bu nedenle bu tip projelerin çıkarılması ülkelerin dışa bağımlılığını azaltacaktır.