T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

MPEG-4 STANDARDINDA KODLANMIŞ VİDEO VERİLERİNİN İÇERİSİNDEN VİDEO NESNELERİNİN ELDE EDİLMESİ

ERHAN TAŞKIN

DOKTORA TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMAN: YRD. DOÇ.DR. NURŞEN TOPÇUBAŞI

EDİRNE, 2013

MPEG-4 STANDARDINDA KODLANMIŞ VİDEO VERİLERİNİN İÇERİSİNDEN VİDEO NESNELERİNİN ELDE EDİLMESİ

ERHAN TAŞKIN

DOKTORA TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

2013

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

T.Ü. Fen Bilimleri Enstitüsü onayı

Prof. Dr. Mustafa ÖZCAN
Fen Bilimleri Enstitüsü Müdürü

Bu tezin Doktora tezi olarak gerekli şartları sağladığını onaylarım.

Yrd. Doç. Dr. Tolga SAKALLI
Anabilim Dalı Başkanı

Bu tez tarafımızca okunmuş, kapsamı ve niteliği açısından bir Doktora tezi olarak kabul edilmiştir.

Yrd. Doç. Dr. Deniz TAŞKIN
İkinci Tez Danışmanı

Yrd. Doç. Dr. Nurşen TOPÇUBAŞI
Tez Danışmanı

Bu tez, tarafımızca okunmuş, kapsam ve niteliği açısından Bilgisayar Mühendisliği Anabilim Dalında bir Doktora tezi olarak oy birliği/oy çokluğu ile kabul edilmiştir.

Jüri Üyeleri :	İmza
Prof. Dr. Mesut RAZBONYALI	……
Prof. Dr. Ahmet KAŞLI	……
Yrd. Doç. Dr. Nurşen TOPÇUBAŞI	……
Yrd. Doç. Dr. Aydın CARUS	……
Yrd. Doç. Dr. Tarık YERLİKAYA	……

Tarih:
31 / 05 / 2013

T.Ü.FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ DOKTORA PROGRAMI
DOĞRULUK BEYANI

İlgili tezin akademik ve etik kurallara uygun olarak yazıldığını ve kullanılan tüm literatür bilgilerinin kaynak gösterilerek ilgili tezde yer aldığını beyan ederim.

31 / 05 / 2013
Erhan TAŞKIN

Doktora Tezi

MPEG-4 Standardında Kodlanmış Video Verilerinin İçerisinden Video Nesnelerinin Elde Edilmesi

T.Ü. Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

ÖZET

Son on beş yılda büyük gelişim gösteren sayısal video artık hayatımızın her alanında kullanılmaktadır. Sayısal videoların çeşitli ortamlarda bu kadar çok yaygınlaşmasındaki faktörlerin başında, video sıkıştırma yöntemleri gelmektedir. Sıkıştırma işlemi, sayısal videoların depolanmasında ve iletiminde önemli bir yere sahiptir. Günümüzde kullanılan en yaygın sıkıştırma yöntemlerinden biri MPEG-4 standardıdır.

Sıkıştırılan sayısal videolar kolay ve daha az maliyetle depolanabilmektedirler. Büyük miktarda depolanan sayısal videolara erişim, bazı zorlukları beraberinde getirmektedir. Video dosyalarının etiketlenmesi temel anlamda arama yapmaya izin vermektedir. Fakat daha detaylı sonuçlara ulaşmak, video verilerinin içerisindeki önemli nesnelerin bulunması ve etiketlenmesi ile mümkün olabilmektedir.

Bu tezin amacı, MPEG-4 standardında kodlanmış video verilerinin içerisinden video nesnelerini elde etmek ve bir veri kümesinde toplamaktır. Kullanıcı tarafından etiketlenen nesnelerin konumları indekslenmektedir. Bu yapı üzerinden videolar üzerinde sorgulama yapılması sağlanmaktadır.

Yıl : 2013

Sayfa Sayısı : 100

Anahtar Kelimeler : MPEG-4, CABAC, CAVLC, Video Nesne Çıkarımı, Hareket Algılama, Video Nesne Bölütleme

Doctorate Thesis
Extraction of Video Objects from MPEG-4 Encoded Video Data
Trakya University Institute of Natural Sciences
Department of Computer Engineering

ABSTRACT

Digital video, which has been developed in a great way over fifteen years, has been used in our daily life. The primary factor causing all of these is video encoding techniques. Encoding process has a great impact on storing and transferring digital video. MPEG-4 standard is commonly used one among the encoding techniques.

Videos which are encoded can cost less and stored easily. Some difficulties have been coming up with accessing digital videos that are stored in a large amount. Labeling video files lets fundamental searching. However, achieving more detailed results is only possible with acquiring significant objects and labeling them.

The purpose of this thesis, extracting video objects from the video data which were encoded as MPEG-4 standards and collecting them into a data set. The positions of the objects which were labeled by the user are indexed. From this structure it is enabled to query the videos.

Year : 2013

Number of Pages : 100

Keywords : MPEG-4, CABAC, CAVLC, Video Object Extraction, Motion Detection, Video Object Segmentation

TEŞEKKÜR

Öncelikle yardımlarını ve desteğini hiçbir zaman esirgemeyen danışman hocam Yrd. Doç. Dr. Nurşen TOPÇUBAŞI'na ve ikinci danışman hocam Yrd. Doç. Dr. Deniz TAŞKIN'a teşekkür ederim.

Ayrıca tez çalışmam sırasında yaptıkları yapıcı eleştirileri ve yönlendirmeleriyle destek olan Prof. Dr. Mesut RAZBONYALI, Prof. Dr. Ahmet KAŞLI, Yrd. Doç. Dr. Aydın CARUS ve Yrd. Doç. Dr. Tarık YERLİKAYA'ya, çalışmalarımda bana sürekli destek olan aileme teşekkürlerimi sunarım.

İÇİNDEKİLER

ÖZET i
ABSTRACT ii
TEŞEKKÜR iii
İÇİNDEKİLER iv
KISALTMALAR DİZİNİ vi
ŞEKİLLER DİZİNİ viii
TABLOLAR DİZİNİ x
BÖLÜM 1 1
1. GİRİŞ 1
BÖLÜM 2 4
2. BU KONUDA YAPILAN ÇALIŞMALAR 4
BÖLÜM 3 7
3. HAREKETLİ GÖRÜNTÜLERİN GELİŞİMİ 7
3.1. Analog Video 9
3.2. Sayısal Video 11
3.2.1. Sayısal Video Biçimleri 12
BÖLÜM 4 16
4. SAYISAL VİDEO SIKIŞTIRMA YÖNTEMLERİ 16
4.1. MPEG Standartları 17
4.2. MPEG-4 AVC (H.264) Standardı 20
4.2.1. ISO Temelli Çoklu Ortam Dosya Yapısı 20
4.2.2. MPEG-4 AVC (H.264) Standardında Kullanılan Sözdizimi Elemanları 32
4.2.2.1. NAL Birimi 32
4.2.2.2. Görüntü Parçası Parametre Kümesi (SPS RBSP) 33
4.2.2.3. Resim Parametre Kümesi (PPS RBSP) 37
4.2.2.4. Tamamlayıcı Geliştirme Bilgileri (SEI RBSP) 40
4.2.2.5. İsteğe Bağlı Ayırıcı Birimler (RBSP) 41
4.2.2.6. Resim Dilimi Katmanı (RBSP) 41

4.2.2.7. Makro Blok Katmanı 47
4.2.3. MPEG-4 AVC (H.264) Standardında Kullanılan Kodlama Yöntemleri ..54
4.2.4. MPEG-4 AVC (H.264) Ters Dönüşüm İşlemleri 66
4.2.5. MPEG-4 AVC (H.264) Standardında Kullanılan Resim Biçimleri 70
4.3. M-JPEG (Hareketli-JPEG) ve Diğer Sıkıştırma Standartları 73
BÖLÜM 5 74
5. MPEG-4 AVC (H.264) STANDARDINDA KODLANMIŞ VİDEOLARIN ÇERÇEVELERİNİN ELDE EDİLMESİ..... 74
5.1. ISO Temelli Çoklu Ortam Dosya Yapısının Çözülmesi 74
5.2. AVC Kod Çözümü 76
5.3. Video Çerçevelerinin Oluşturulması ve Görüntülenmesi 76
BÖLÜM 6 79
6. UYGULAMADA KULLANILAN SAYISAL VİDEO GÖRÜNTÜLERİNDEN NESNE ELDE ETME YÖNTEMLERİ..... 79
6.1. Hareketli Nesnelerin Takibi 79
6.2. Renk Takibi 81
6.3. SURF Algoritması 81
6.4. Elde Edilen Nesnelerin Saklanması ve Aranması 82
BÖLÜM 7 83
7. SONUÇ 83
KAYNAKLAR 84
EKLER 89
ÖZGEÇMİŞ 101

KISALTMALAR DİZİNİ

Kısaltma	İngilizce	Türkçe
AVC	Advanced Video Coding	Gelişmiş Video Kodlama
CABAC	Context-Based Adaptive Binary Arithmetic Coding	Bağlam Temelli Uyarlamalı İkili Aritmetik Kodlama
CAVLC	Context-Based Adaptive Variable Length Coding	Bağlam Temelli Uyarlamalı Değişken Uzunluklu Kodlama
CRT	Cathode Ray Tube	Katot Işın Tüpü
DCT	Discrete Cosine Transform	Ayrık Kosinüs Dönüşümü
DPB	Decoded Picture Buffer	Çözülmüş Resim Ara Belleği
DSS	Digital Satellite System	Sayısal Uydu Sistemi
DTV	Digital Television	Sayısal Televizyon
DV	Digital Video	Sayısal Video
DVB	Digital Video Broadcasting	Sayısal Video Yayıncılığı
DVD	Digital Versatile Disk	Sayısal Çok Amaçlı Disk
GOP	Group Of Pictures	Resim Grubu
HDTV	High Definition Television	Yüksek Tanımlı Televizyon
IDCT	Inverse Discrete Cosine Transform	Test Ayrık Kosinüs Dönüşümü
IDR	Instantaneous Decoding Refresh	Anlık Kod Çözme Yenilemesi
IEC	International Electrotechnical Commission	Uluslararası Elektroteknik Komisyonu
ISO	International Organization for Standardization	Uluslararası Standartlar Teşkilatı
ITU	International Telecommunication Union	Uluslararası İletişim Birliği
JPEG	Joint Photographic Experts Group	Birleşik Fotoğraf Uzmanları Grubu
MPEG	Moving Picture Experts Group	Hareketli Resim Uzmanları Grubu
NAL	Network Abstraction Layer	Ağ Soyutlama Katmanı
NTSC	National Television Standards Comittee	Ulusal Televizyon Standartları Komitesi
PAL	Phase Alternarting Line	Faz Değişimli Hat

PPS	Picture Parameter Set	Resim Parametre Kümesi
RBSP	Raw Byte Sequence Payload	Ham Bayt Dizisi Uzantısı
RGB	Red-Green-Blue	Kırmızı-Yeşil-Mavi
SDL	Syntax Description Language	Sözdizimi Tanımlama Dili
SDTV	Standard Definition Television	Standart Tanımlı Televizyon
SECAM	Sequential Color with Memory	Bellek ile Sıralı Renk
SE	Syntax Element	Sözdizimi Elemanı
SEI	Supplemental Enhancement Information	Tamamlayıcı Geliştirme Bilgileri
SICA	Spatiotemporal Independent Component Analysis	Zaman-Mekan Bağımsız Bileşen Analizi
SPS	Sequence Parameter Set	Görüntü Parametre Kümesi
STB	Set-Top Box	Set Üstü Kutu
SURF	Speeded Up Robust Features	Hızlandırılmış Güçlü Özellikler
UUID	Universal Unique Identifier	Evrensel Benzersiz Tanımlayıcı
VGA	Video Graphics Array	Video Grafikleri Dizisi
VHS	Video Home System	Ev Video Sistemi
VoD	Video on Demand	Talep üzerine Video
VOP	Video Object Plane	Video Nesnesi Düzlemi
VUI	Video Usability Information	Video Kullanılabilirlik Bilgileri
WMV	Windows Media Video	Windows Ortam Videosu
YUV	Luma-Chrominance	Parlaklık-Renk Farkı

ŞEKİLLER DİZİNİ

Şekil 3.1. Etienne Jules Marey ve Eadweard Muybridge'in bir çalışması 7
Şekil 3.2. Zoopraxiscope 8
Şekil 3.3. Cinematographe 8
Şekil 3.4. Vitascope 9
Şekil 3.5 Analog sinyallerden çerçevenin elde edilmesi 10
Şekil 3.6. Video yakalama kartı 11
Şekil 4.1. Video kodlama ve kod-çözme 17
Şekil 4.2 Örnek MPEG çerçeve dizilimleri 20
Şekil 4.3. ISO temelli çoklu-ortam dosya yapısı şematik gösterimi 23
Şekil 4.4. Yığın referansları. 24
Şekil 4.5. Resim bilgilerini gösteren parametreler. 24
Şekil 4.6. NAL birimlerinin ardışık ve ayrı dizilimleri 26
Şekil 4.7. MPEG-4 AVC standardında tanımlanan NAL birimi genel yapısı 27
Şekil 4.8. Bir ISO temelli çoklu ortam dosyasındaki NAL biriminin çözülmesi 28
Şekil 4.9. NAL Birimlerini oluşturan sözdizimi elemanlarının hiyerarşik yapısı 30
Şekil 4.10. NAL Biriminin ana yapısının çözülmesi. 32
Şekil 4.11. Görüntü parçası parametre kümesi elemanlarının elde edilmesi 35
Şekil 4.12. Örnek bir MPEG-4 AVC çoklu ortam dosyasından okunan SPS değerleri .. 36
Şekil 4.13. Resim parametre kümesi elemanlarının elde edilmesi 39
Şekil 4.14. Örnek bir MPEG-4 AVC çoklu ortam dosyasından okunan PPS değerleri .. 40
Şekil 4.15. Örnek SEI mesajları 41
Şekil 4.16. Resim dilimi başlık parametrelerinin ham verilerden elde edilmesi 45
Şekil 4.17. Örnek bir MPEG-4 AVC çoklu ortam dosyasından okunan dilim başlık parametrelerinin değerleri 46
Şekil 4.18. MPEG-4 AVC resim bileşenleri. 47
Şekil 4.19. Makro blok tipleri ve örnek tahmin referansları 48
Şekil 4.20. Dahili tahmin için çeşitli blok boyutlarında kullanılan kaynaklar 50
Şekil 4.21. 4x4 dahili tahmin modları 50
Şekil 4.22. 16x16 dahili tahmin modları 50

Şekil 4.23. Orijinal resim (sol), tahmin resim (orta), orijinalden tahmin çıkarıldıktan sonra kalan resim (sağ).....51
Şekil 4.24. P makro blok tahmin örneği.....52
Şekil 4.25. Fark resmi parlaklık ve renk farkı değerlerinin katsayılarının dizilimi 53
Şekil 4.26. CABAC ile kodlanmış ortam verilerin ayrıştırılmasında kullanılan temel fonksiyonlar 60
Şekil 4.27. CABAC ile kodlanmış ortam verilerinin ayrıştırılması akış şeması.....61
Şekil 4.28. CABAC kodlanmış *mb_type* sözdizimi elemanının çözülmesi.....63
Şekil 4.29. Fark resmi için DC ve AC katsayılarının elde edilmesi 65
Şekil 4.30. Tahmin ve Fark resimlerinden orijinal resmin elde edilmesi 66
Şekil 4.31. İki boyutlu ayrık kosinüs dönüşümü (2-D DCT) denklemi.....66
Şekil 4.32. Yeniden ölçeklendirme ve ters dönüşüm adımları.....67
Şekil 4.33. Parlaklık katsayıları için ters dönüşüm adımları.....68
Şekil 4.34. Renk farklı katsayıları için ters dönüşüm adımları.....68
Şekil 4.35. Yeniden ölçekleme ve ters dönüşüm sürecinin gelişimi.....69
Şekil 4.36. RGB ve YUV Renk Uzayları.....70
Şekil 4.37. Bir resmin YUV bileşenlerine ayrılması. 71
Şekil 4.38. MPEG-4 AVC kodlanmış resim çerçeveleri içerisindeki YCbCr konumları. 72
Şekil 5.1. ISO temelli çoklu ortam dosya yapısını çözen uygulama.....75
Şekil 5.2. YUV – RGB renk dönüşümü.....78
Şekil 5.3. Video çerçevelerini gösteren uygulama.....78
Şekil 6.1. Arka plan çıkarma ve nesne takibi.....80
Şekil 6.2. Renk filtreleme ve nesne elde etme 81
Şekil 6.3. SURF algoritması ile resim içerisinde nesnenin tespit edilmesi.....81

TABLOLAR DİZİNİ

Tablo 3.1 VCD Standartlarının özellikleri 12
Tablo 3.2. SVCD Standardının özellikleri 13
Tablo 3.3.Video biçimlerinin karşılaştırılması 15
Tablo 4.1. MPEG Standartları 19
Tablo 4.2. ISO temelli dosya yapısında bulunan tüm bileşenler 21
Tablo 4.3. MPEG-4 SDL ile ifade edilmiş bir nesne. 22
Tablo 4.4. Örnek Sözdizimi Elemanı (SE) 25
Tablo 4.5. NAL Birimi Türleri 26
Tablo 4.6. NAL birimleri türleri için alt yordamlar. 28
Tablo 4.7. Dahili tahmin türleri 49
Tablo 4.8. Referans resim kaynak listeleri 53
Tablo 4.9. Exp-Golomb kodlama tablosu 55
Tablo 4.10. İşaretli Exp-Golomb kodlama tablosu 55
Tablo 4.11. Haritalanmış Exp-Golomb kodlama tablosu 56
Tablo 4.12 CAVLC Kod Çözümü Örnek Tablosu 57
Tablo 4.13. Temel renklerin RGB ve YUV renk uzaylarındaki değerleri. 70
Tablo 4.14. MPEG4 AVC içerisinde kullanılan YCbCr türleri. 72

BÖLÜM 1

1. GİRİŞ

İnsanoğlunun geçmişi bir şekilde geleceğe taşıma çabalarından bir tanesi de gördüklerini kaydetme isteğidir. Binlerce yıl önce, mağara duvarlarına çizilen resimlerle görülen ilk örnekler, yazının bulunması ile desteklenerek daha da etkili hale gelmiştir. Fakat insanoğlunun gördüklerinin birebir kopyasını çıkarması 19. yüzyıla kadar mümkün olmamıştır. Fotoğrafın keşfi, anı yakalamayı mümkün kılmış, ama zamanla olan değişimi göstermek için yetersiz kalmıştır.

Kökeni Latince videre (görmek)'den gelen video, görüyorum anlamına gelmektedir ve belli bir zaman aralığında ardı ardına gösterilen resim kareleri olarak tanımlanabilmektedir. 20. yüzyılın başlarında, resimler ardışık olarak çeşitli yöntemlerle bir araya getirilerek hareketli görüntüler elde edilmiş ve videonun da temelleri atılmıştır.

İlk zamanlarda analog video ile başlayan süreç, günümüzde yerini sayısal videoya bırakmıştır ve videonun en büyük gelişim ölçütlerinden biri sıkıştırma yöntemleri olmuştur. Video sıkıştırma, görüntüleri bozmadan olabildiğince fazla veriyi atma işlemidir. Video sıkıştırma yöntemleri kayıplı ya da kayıpsız olabilir. Kayıplı sıkıştırma yöntemlerinde kodlama sonrasındaki görüntü ile orijinal görüntü birbiriyle tamamen aynı olmayabilir.

Hareketli Görüntü Uzmanları Birliği (MPEG), video görüntülerini ve ses sinyallerini sıkıştırmak için DCT kullanılan, dünyada yüksek kaliteli sayısal videolarda ortak bir dil sağlama amacıyla, bir dizi standartlar tanımlamıştır. Bu standartlar, her bir çerçeveyi JPEG algoritmasını kullanarak sıkıştırmakta, sonra ardışık çerçevelerde aynı kalan verileri atmaktadır.

İlk çıkan MPEG-1 standardı o günün ihtiyaçlarına uygun bir sıkıştırma olmasına rağmen kısa sürede çok geniş bir kullanım alanı bulması sebebiyle güncellenerek MPEG-2 standartları oluşturulmuştur. MPEG-1'in eksikliklerinin giderildiği ve bazı yeniliklerin eklendiği MPEG-2 standardının kullanımı halen günümüzde de devam etmektedir. Ancak gelişen teknoloji ve internet üzerinden çoklu ortam bilgilerinin aktarımının artması farklı ihtiyaçları da beraberinde getirmektedir. Bunları karşılamak amacıyla MPEG-4 standartları geliştirilmiştir.

MPEG-4, önceki MPEG standartlarına göre daha yüksek sıkıştırma olanakları ve yeni kodlama araçları sunmaktadır. MPEG-4 halen gelişmekte olan bir standart olup yapı olarak bölümlerden oluşmaktadır. Bu standardın önceki standartlara göre en büyük farkı nesne temelli bir yapı kullanmasıdır. Bu yapı sayesinde ortam bileşenlerini farklı nesneler halinde saklamaktadır. Her bir nesne ile farklı etkileşimlerde bulunmaya izin vermektedir.

Video verilerinin sıkıştırılması konusunda oldukça ilerleme kat edilmesi ile beraber, çok miktarda ve büyük hacimli verilerin daha kolay depolanabilmesi ve bir yerden başka bir yere kolaylıkla aktarılabilmesi, video verileri üzerinde farklı çalışma alanlarının ortaya çıkmasına neden olmaktadır. Bu çalışmalardan bir tanesi de video içeriklerinin anlamlandırılması ile ilgilidir.

Akım ya da dosya düzeyinde etiketler ile anlam kazandırılmaya çalışılan videolar içerisinde daha detaylı bir içerik sağlayabilmek için videoda bulunan önemli nesnelerin elde edilmesi ve bunların etiketlenmesi, üzerinde çalışılan güncel ana konulardan biridir. Bu çalışma konusunun amacı, video verilerine sadece genel bir anlam katmak yerine içeriklerinin çıkarılarak çok daha detaylı bir anlamlandırma ve etkileşim sağlamaktır. Bu şekilde işlenmiş ve anlamlandırılmış video verileri içerisinde günümüzde tıpkı metin aramalarda yapılabileceklerin bir benzerinin yapılabilmesi mümkün olacaktır.

Video verilerinin işlenmesi metin verilere nazaran oldukça zordur, yoğun hesaplamalar ve yüksek işlem gücü gerekmektedir. Video içerisinden nesnelerin elde edilebilmesi için farklı birçok yöntem ve algoritmalar bulunmaktadır. Teknolojideki

gelişmelere paralel olarak, geliştirilmiş algoritma ve yöntemlerin performansları artmakta aynı zamanda yeni yaklaşımlar da ortaya çıkmaktadır.

Bu tez çalışmasında, MPEG-4 AVC standardında kodlanmış video verilerinin içerisinden video nesnelerinin elde edilip bir veri kümesinde toplanması amaçlanmaktadır. Video verilerinden elde edilen nesneler kullanıcı tarafından etiketlenmektedir. Nesnelerin konumları indekslenerek MPEG-4 video üzerinden sorgulama yapılması hedeflenmektedir.

BÖLÜM 2

2. BU KONUDA YAPILAN ÇALIŞMALAR

MPEG-4, yüksek performanslı sıkıştırma, sayısal video yayıncılığı, video izleme sistemleri vb. oldukça çeşitli çalışma alanları olup bu alanlarda yapılmış birçok çalışma bulunmaktadır. Video nesnelerinin bulunması ile ilgili daha önceden yapılmış çalışmalardan bazıları bu bölümde listelenmektedir.

A. Müfit Ferman ve arkadaşları MPEG-4'ün nesne temelli kodlama standardı üzerine, video nesnelerine daha kolay ulaşmak ve her bir video nesnesinin anahtar çerçevelerini otomatik elde etmek amacıyla bir zamansal bölütleme algoritması üzerinde çalışmışlardır [1].

MPEG-4, içerik temelli fonksiyonları desteklemektedir. Bunun en büyük avantajlarından biri, önceleri bir arada bulunan video nesnelerinin ayrıştırılarak her bir video nesnesinin tek bir hareketli nesneyi temsil etmesinin sağlanabilmesidir. Thomas Meier ve King N. Ngan çalışmalarında özet olarak bazı önemli hareket bölütleme ve video nesnesi üretme tekniklerini anlatmaktadırlar [2].

Paulo Nunes ve arkadaşları geçerli MPEG-4 Video Doğrulama Modelinin (MPEG-4 Video VM) açıklanması ve video nesnelerinin mobil çoklu ortam sistemleri (MoMuSys) uygulamaları ile kodlanmasının sonucunda elde edilen nesne temelli video sunumuna ait bazı fonksiyonelliklerin gösterimi ile ilgili bir çalışmada bulunmuşlardır [3].

E.P. Ong ve arkadaşları, çalışmalarında video nesne bölütlemesi için iki yeni algoritma önermektedirler [4]. Her iki algoritmada değişim algılama süreci için

uyarlamalı eşik kestirimi metodu kullanmaktadır. Nesne bölütleme maskesini inceltmek için yeni belirlenen hareketli kenarlar ve graf temelli kenar ilişkilendirme metodu kullanımı görülmektedir. Önerilen çalışma video nesne bölütlemesi için, gerçek zamanlı videolar ve MPEG-4 içerik temelli video kodlama üzerinde hızlı ve iyi sonuçlar göstermektedir.

Düşük seviyeli otomatik bölge bölütlemesi ve yüksek seviyeli anlamsal video nesnelerini izleme için geliştirilmiş aktif bir sistem Di Zhong ve Shih-Fu Chang tarafından gerçekleştirilmiştir [5]. Sistem iki aşamadan oluşmaktadır. Başlangıç için nesne bölütleme kısmı anlamsal bir nesne oluşturmak için başlangıç çerçeve girişinin yapıldığı kısım ve bir nesne izleme bölümü, belli başlı nesne için anlamlı bölgelerin takibi ve çerçeveler boyunca gruplanması bölümleridir. Uygulamalar sonucunda farklı video tiplerinde oldukça iyi performans gösterdiği çalışmada belirtilmektedir.

Hua Zhong ve arkadaşları, çalışmalarında otomatik bölütlemenin verimliliği ile elle yapılan bölütlemenin doğruluğunu bir arada kullanarak yarı otomatik bir video nesnelerini bölütleme stratejisi önermektedirler [6].

Arka plandan siluet oluşturulması, istatistiksel özellik analizi yaklaşımı ile nesne tanımlaması, bölge temelli hareket kestirimi ile nesne takibi kullanılarak VOP'lerden otomatik nesne takibi ve bölütlemesi Ferdous Ahmed Sohel ve arkadaşlarının çalışmalarındaki dikkat çekici unsurlardır [7].

Oğuzhan Urhan yüksek lisans tezinde, MPEG-4 video kodlama standardının en temel öğesi olan video nesnelerinin elde edilmesi için yeni bir bölütleme yöntemi önermiş ve gerçekleştirmiştir [8]. Önerilen yöntem blok temelli arka planın kaydedilmesine dayanan zamansal bir bölütleme yöntemidir.

Fatih Porikli çalışmasında, MPEG kodlanmış videolar için gerçek zamanlı nesne bölütlemesi metodu önermektedir [9].

Video düzenleme, sıkıştırma, içerik analizi, MPEG-4 standardının nesneye yönelik video kodlaması kavramı, MPEG-4 video kodlama yaklaşımı ile bölütleme

işlemleri Dirk Sven Farin tarafından hazırlanan doktora tezinin temel konuları içerisindedir [10].

80'lerde duyurulmuş Zaman-Mekan bağımsız bileşen analizi SICA sinir ağı modeli, Xiao-Ping Zhang ve Zhenhe Chen tarafından temel alınarak video nesneleri bölütlemesi gerçekleştirilmiştir [11].

Bölütleme kalite değerlendirmesi için genel bir çalışma Elisa Drelie Gelasca ve Touradj Ebrahimi tarafından yapılmıştır. Seçilen algoritmaların sonuçlarının başarıları değerlendirilmiştir [12].

Değişim tespiti ve arka plan güncelleme temelli verimli gerçek zamanlı bir video nesne bölütleme algoritması Tsong-Yi Chen ve arkadaşları tarafından önerilmektedir [13]. Temelindeki basit fikir, ardışık çerçeveler arasındaki zamansal bilginin analiz edilerek değişimin tespiti ve değişim bölgelerinin elde edilmesi temeline dayanmaktadır.

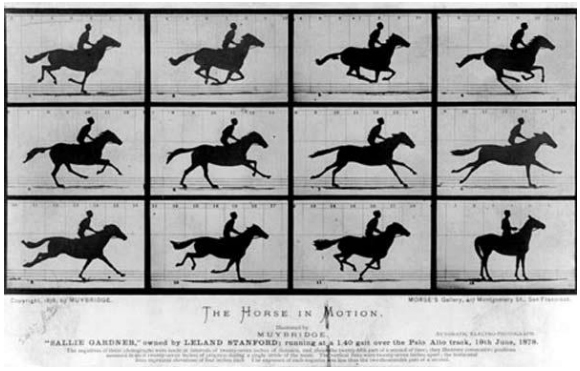
William Brendel ve Sinisa Todorovic bir video içerisinde meydana gelen hareketli ve durgun nesneler için denetimsiz bir bölütleme yaklaşımı sunmaktadırlar [14]. Video bölütlemesi çerçeveler arası bölgelerin izlenmesi ile gerçekleştirilmektedir.

Bu çalışmalardan da anlaşılacağı üzere video bölütlemesi ve nesne elde etme çalışmaları önceki video formatlarında olduğu gibi MPEG-4 üzerinde de devam etmektedir. MPEG-4 yenilikleri ve yapısı ile bu konuda farklı yaklaşımların ortaya çıkmasına imkan sağlamaktadır.

BÖLÜM 3

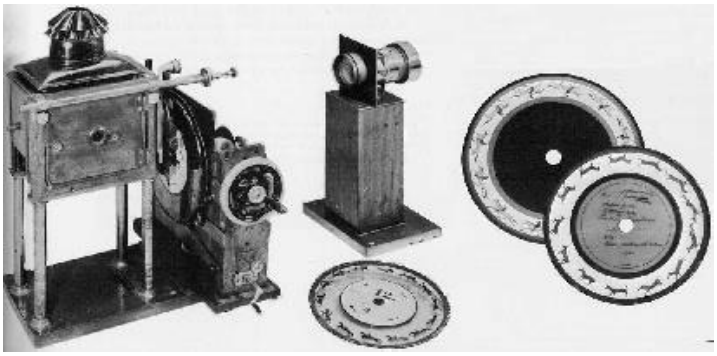
3. HAREKETLİ GÖRÜNTÜLERİN GELİŞİMİ

Onyedinci yüzyılda büyülü fenerlerle başlayan görüntülerin hareketlendirilmesi serüveni 1827 yılında Claude Niepce'nin ilk fotoğrafı çekmesi ile bir başka boyut kazanmıştır. 1800'lerin ikinci çeyreğinde birkaç fotoğrafın belirli mekanizmalarla bir araya gelmesi ile birçok hareketli görüntü elde etme çalışmaları olmuştur. Etienne Jules Marey ve Eadweard Muybridge o yılların başarılı isimlerindendir. Hızlı canlıların hareketlerini dairesel bir plaka üzerine 12 resim şeklinde yerleştirerek (Şekil 3.1) hareketli bir görüntü elde etmişlerdir.



Şekil 3.1. Etienne Jules Marey ve Eadweard Muybridge'in bir çalışması

Amerika'da, animasyon resim ve ya filmleri gösterebilen William Lincoln tarafından 1867 yılında ilk patent alınmış *zoopraxiscope* ya da *wheel of life* (Şekil 3.2) ismindeki makinedir. Hareketli çizimler ve ya fotoğraflar bu makine içindeki dar bir açıklıktan izlenmektedir.



Şekil 3.2. Zoopraxiscope

Hareketli görüntü kamerasının icat edilmesiyle, modern hareketli görüntüler yaygınlaşmaya başlamıştır. Fransız Louis Lumiere, sık sık ilk hareketli görüntü kamerasını icat eden kişi olarak anılmaktadır. Fakat gerçekte, Lumiere ile aynı zamanlarda birkaç farklı kişi daha benzer icatlar yapmışlardır. Lumiere'in icat ettiği, portatif hareketli görüntü kamerası, film işleme ünitesi ve *Cinematographe* isimdeki projektörü dahil üçü bir arada bir makineydi [15].

Cinematographe, hareketli görüntüleri çok popüler hale getirmiştir ve bu yüzden hareketli görüntü çağının başlangıcı Lumiere'nin icadı ile başladı denmesine neden olmuştur. 1895 yılında Lumiere ve erkek kardeşi ilk yansıtılmış, hareketli, fotoğraf görüntüsünü bir ücret karşılığında birden çok insana sunmuşlardır.

Şekil 3.3. Cinematographe

Lumiere kardeşler yansıtılmış filmde ilk değillerdi. 1891 yılında Edison şirketi aynı anda tek kişinin izleyebileceği hareketli görüntü makinesi olan *Kinetoscope*'u

tanıtmıştır. 1896 sonrasında, Edison güçlendirilmiş projektörü *Vitascope*'u (Şekil 3.4) tanıtmış ve bu ticari olarak ilk başarılı projektör olmuştur [15].

Şekil 3.4. Vitascope

1900'lü yılların başında oldukça yaygınlaşan sinema sektörü, hızla gelişimine devam etmiştir. 1906'dan sonra filmler renklenmeye başlamış ve sessiz olan filmler, değişik metotlarla seslendirilmeye çalışılmıştır. Fakat senkronizasyon ve zamanlama sorunları tamamıyla aşılamamıştır. Ses konusundaki en başarılı çözüm 1927 yılında Warner kardeşlerden gelmiştir [16].

1921'de radyo yayınlarının başlamasının ardından 1929'da da ilk televizyon deneme yayınlarının yapılmaya başlanması ile 1930'lu yılların ortalarında kısıtlı olarak hareketli görüntülerin evlere kadar girmesi sağlanmıştır.

3.1. Analog Video

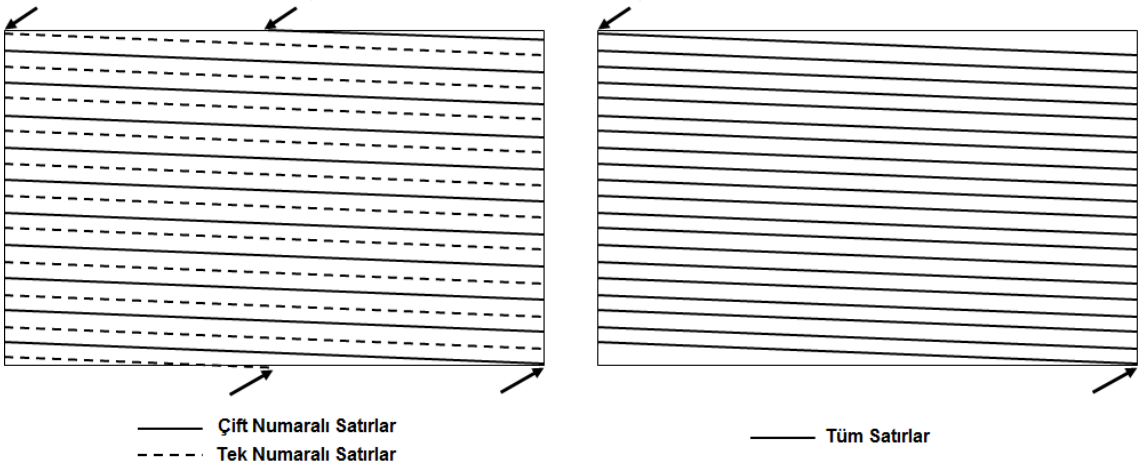
Radyonun keşfi, ses dalgalarının manyetik dalgalara dönüştürülebildiğini ve çok uzak mesafelerdeki radyo alıcılarına iletilebildiğini göstermiştir. Buna benzer olarak bir televizyon kamerası ile elektriksel sinyallere dönüştürülen bir görüntünün renk ve parlaklık bilgisi, hava yolu ile iletilebilmekte ya da bir videoteyp üzerinde kaydedilebilmektedir.

TV Sinyallerinin kodlanmasında 3 farklı biçim vardır [17]:

- Avrupa'nın çoğunun kullandığı PAL
- Fransa, Rusya ve bazı Doğu Avrupa ülkelerinin kullandığı SECAM
- Amerika ve Japonya'nın kullandığı NTSC

PAL şebeke elektrik akımının 50 Hz olmasını temel alır ve saniyede 25 çerçeve resim tarama hızına sahiptir.

Video sinyali, uzaysal içeriği önceden tanımlanmış bir tarama kuralına göre bir dizi resmin temsil edildiği, zaman içinde değişen tek boyutlu sinyallerdir [18]. Bu sinyaller bir video kamerası ya da çoklu ortam oynatıcısı benzeri cihazların çıktısı olarak üretilebilmektedir. Bu sinyallerin tekrar resme dönüştürülmesi televizyon içerisinde bulunan ve resmi göstermeye yarayan televizyon tüpünde (CRT) elektron tabancası ile 625 satırın taranarak birleştirilmesiyle sağlanmaktadır. Bu saniyede 25 kez tekrarlanmaktadır. Resim, elektron tabancasının bazen tek taraması, bazen de tek ve çift satırları ayrı ayrı taraması ile oluşturulmaktadır. Ayrı ayrı taramada çerçeveler arasında daha net bir geçiş sağlanmaktadır. Bant genişliği aynı kalmakla beraber, çözünürlük yarıya düşmektedir. Şekil 3.5'te analog sinyallerin tek ve çift satırların ayrı ayrı taranması ile tek seferde taranması karşılaştırmalı olarak gösterilmektedir [19].

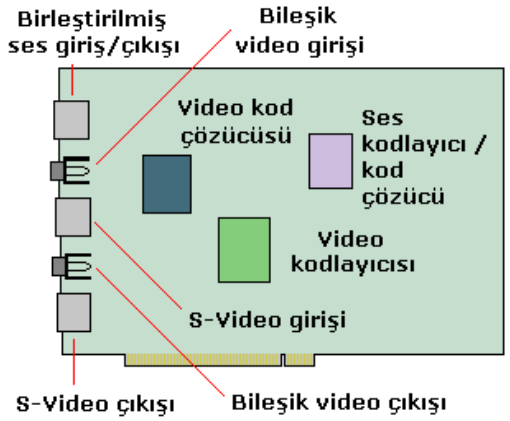


Şekil 3.5 Analog sinyallerden çerçevenin elde edilmesi

3.2. Sayısal Video

Bilgisayarlar, analog bilgilerin tersine daha kesin olan sıfır ve birlerden oluşan ikilik sayıları kullanmaktadırlar. Görsel bir bilginin sayısal olarak depolanması, video sinyallerinin sayısal eşdeğerine çevrilmesi analog-dijital çevirici adını alan özel bilgisayar yongaları sayesinde mümkündür. Dönüştürme işlemi, örnekleme ya da görüntü yakalama olarak bilinmektedir. Sayısal görüntüyü, eski sistem televizyon cihazları gibi cihazlarda izlemek için dijital-analog çeviriciler gerekmektedir [20].

Analog TV sinyallerinin sayısallaştırılması görüntü yakalama kartları tarafından gerçekleştirilmektedir (Şekil 3.6). Bu kartlar her bir çerçeveyi bir dizi bit haritasından oluşan görüntülere çevirmektedirler. Bir yatay çizgi PAL sistem için 768 bölüme ayrılmaktadır. Bu bölümlerin her birinin kırmızı, yeşil ve mavi değerleri hesaplanarak, her bir satır için 768 renkli piksel elde edilmektedir. PAL sinyallerinde 625 satır bulunmaktadır fakat bu satırların 50 tanesi, içerisinde resim bilgisi olmayan teleteks verisi bulunmaktadır. Kalan 575 satır TV için kullanılan en-boy oranı 4:3 değerine uygulandığında tam sayı olmadığı için 576 olarak alınmaktadır.



Şekil 3.6. Video yakalama kartı

Sayısallaştırma sonrası, bir tam çerçeve 768x576 pikselden oluşmaktadır. Her piksel, renk bilgisini barındıran kırmızı, yeşil ve mavi bileşenleri gerektirmektedir (24 bit). Böylece her bir çerçeve 768 x 576 x 3 bayt = 1,3 MB yer kaplamaktadır. 1 saniyelik video için bu değer 32,5 MB (1,3 x 25 fps) olmaktadır.

Bilim adamları insan gözünün renklerden ziyade parlaklığa daha duyarlı olduğunu keşfettikten sonra YUV adı verilen model, televizyon yayıncılığında kullanılmaya başlanmıştır. YUV modelinde keskinlik renkten bağımsız olarak işlenmektedir. Y parlaklık değeridir ve tam çözünürlük için hesaplanmaktadır. U ve V ise renk farklılık sinyalleridir, yarım çözünürlük (YUV 4:2:2) veya çeyrek çözünürlük (YUV 4:1:1) için hesaplanmaktadır. RGB bir sinyal yerine YUV sinyalinin sayısallaştırmada kullanılması 24 bit yerine 16 bit gerektirmektedir. Bu da gerçek renkte 1 saniyelik PAL videosunun kapladığı boyutu 22 MB'ye düşürmektedir.

NTSC sistemi 525 satır ve 30 fps (elektrik şebekesi 60 Hz olduğundan) kullanmaktadır. Fakat NTSC çerçeveler genellikle VGA çözünürlüğüne denk gelen 640x480'de sayısala çevrilmektedirler.

3.2.1. Sayısal Video Biçimleri

Video CD (VCD) biçimi eğlence ve bilgi yayın dünyalarının taleplerini karşılamak amacıyla oluşturulmuştur. 1993 yılında firmalar tarafından kabul gören VCD, oluşturulmuş tek bir biçim yardımıyla, kişisel bilgisayarlar, televizyonlar ve oyun konsolları gibi geniş bir alanda çoğaltılması ucuz çoklu ortam içeriklerinin kullanılmasını sağlamıştır [21]. Tablo 3.1'de VCD standardının PAL, NTSC ve NTSC Film olarak özellikleri özetlenmektedir.

Tablo 3.1 VCD Standartlarının özellikleri

	PAL	**NTSC**	**NTSC Film**
Çözünürlük	352x288	352x240	352x240
Çerçeve Hızı (çerçeve/saniye)	25	29.97	23.976
Video Bit Hızı (Kbit/saniye)	1150 MPEG-1	1150 MPEG-1	1150 MPEG-1
Ses	44.1kHz stereo, MPEG-1 Seviye 2 biçiminde, 224kbits/saniye bit hızında kodlanmış		
Toplam Bit Hızı	1394.40 kbits/saniye		

Bir VCD ortamı 74/80 dakikalık (650/700 MB) MPEG-1 teknolojisi ile sıkıştırılmış videoyu ve buna ait sesi depolayabilecek kapasiteye sahiptir.

1997 yılında Çin'de geliştirilen yeni bir VCD biçimi 1998 yılında Süper VCD (SVCD) adıyla standartlaştırılmıştır [22] ve VCD standardının doğal bir evrimi olarak görülmüştür. Temel farkı video akımları için MPEG-1 yerine daha yüksek bit hızı ve çözünürlüğü destekleyen, altyazılara da izin veren MPEG-2'nin kullanılmasıdır. SVCD aynı zamanda 16:9 (geniş-ekran) görüntü oranını da desteklemektedir. Tablo 3.2'de SVCD standardının özellikleri gösterilmektedir.

Tablo 3.2. SVCD Standardının özellikleri

Özellikler	**SVCD 1.0**
Video	MPEG-2
bit hızı	En fazla 2.6 Mbit/s
NTSC çözünürlüğü	480x480 geçmeli, 29.97Hz
PAL çözünürlüğü	480x576 geçmeli, 25Hz
Durgun resim (fotoğraf)	MPEG-2 I Çerçevesi
NTSC çözünürlüğü	480x480, 704x480
PAL çözünürlüğü	480x576, 704x576
Ses	MPEG-1 Seviye II
Örnekleme	44.1kHz
bit hızı	32'den 384Kbit/s kadar
Ses kanalları	2 stereo ya da 4 monoya kadar
Çepeçevre sarılmış ses	MPEG-2 (5+1) uzantısı

XVCD ve XSVCD biçimleri, var olan standartların resmi olmayan genişletilmiş biçimleridir. Resim kalitesini ve bit aktarım hızını arttırmak amacıyla tasarlanmışlardır.

MiniDVD biçimi, DVD kalitesindeki 15 dakikalık bir videonun 650 MB kapasiteli bir CD-R(W) disk üzerine kodlanmasıdır. Avantajı, 2000 yılının başlarında

daha ucuz olan CD-R disklerinde SVCD'den daha kaliteli olan DVD biçimindeki videonun depolanmasını sağlamasıdır.

DivX biçimi, Microsoft'un MPEG-4 video teknolojisi temel alınarak ve üzerine MP3 ses akımı eklenerek geliştirilmiştir. Tipik olarak 5 GB'lık 80-90 dakikalık 640x480 çözünürlükte bir DVD kalitesindeki film DivX ile sıkıştırıldığında 650/700 MB'lık tek bir CD-ROM'da depolanabilmektedir.

1990'ların sonlarında tamamen yeni bir nesil dijital kamera ve kaydediciler ile yeni bir video biçimi ortaya çıkmıştır: Sayısal Video (DV). En büyük avantajı tüm video işleme döngüsünün sayısal etki alanı içinde kalmasıdır. Bu sistem sayesinde doğrudan sayısal olarak işlenen video, herhangi bir analog-sayısal çevrim yapılmasına gerek kalmadan, gerçek zamanlı ve kayıpsız kalitede kişisel bilgisayarlara basitçe indirilebilmektedir [23].

DCT, insan gözü tarafından kayıpsız olarak kabul gören bir tekniktir ve RGB renk bilgisini YUV renk uzayına çevirerek veri miktarını azaltmaktadır. Sonuç olarak kamera üzerindeki donanımsal sıkıştırma sistemi videoyu M-JPEG'e benzer bir algoritma kullanarak sıkıştırmaktadır.

DV'nin prensipteki problemi MPEG-2'nin aksine ölçeklenebilir değildir. Standart transfer hızlarına (25/50 Mbit/sn) ve sınırlı renk kapasitesine sahiptir.

2000'lerin başlarında yayıncılıkta DV ve MPEG-2 biçimlerinin her ikisi de kullanılmaktaydı. DV ve MPEG, DCT temelli çok benzer teknolojiler olmasından dolayı tasarlanan donanımlar her iki biçimi de destekleyecek şekilde üretilmişlerdir.

DCT, sıkıştırma konusunda gelinen son nokta olmamaktadır, fakat gerçek zamanlı kodlayıcıların geliştirilmesi esnasında dikkatleri üstüne çekmiştir. Geliştirilmekte olan diğer teknolojiler, daha iyi resim kalitesi ve daha düşük veri miktarı için çalışmaktadırlar. Video biçimlerinin karşılaştırılması Tablo 3.3'te detaylı olarak gösterilmektedir.

Tablo 3.3.Video biçimlerinin karşılaştırılması

	VCD	SVCD	X(S)VCD	DivX	DV	DVD
Resmi Standart	Evet	Evet	Hayır	Hayır	Evet	Evet
Çözünürlük NTSC/PAL	352x240 352x288	480x480 480x456	720x480 720x576 veya daha düşük	640x480 veya daha düşük	720x480 720x576	720x480 720x576
Video Sıkıştırma	MPEG-1	MPEG-2	MPEG-1 veya MPEG-2	MPEG-4	DV	MPEG-2
Ses Sıkıştırma	MPEG-1	MPEG-1	MPEG-1	MP3 WMA	DV	MPEG-2 AC-3
MB/dakika	10	10-20	5-20	1-10	216	30-70
DVD Oynatıcı uyumluluğu	Çok İyi	İyi	İyi	-	-	Mükemmel
İşlemci Gereksinimi	Düşük	Yüksek	Yüksek	Çok Yüksek	Yüksek	Çok Yüksek
Kalitesi	İyi	Çok İyi	Çok İyi	Çok İyi	Mükemmel	Mükemmel

BÖLÜM 4

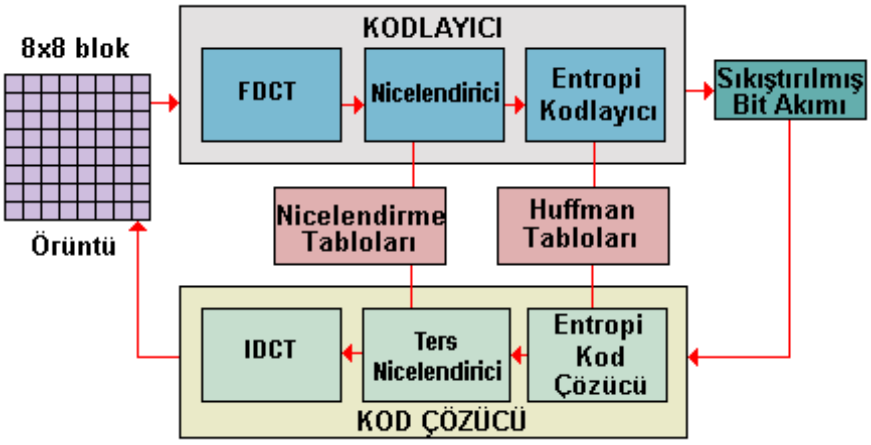
4. SAYISAL VİDEO SIKIŞTIRMA YÖNTEMLERİ

Video sıkıştırması, görüntüleri bozmadan olabildiğince fazla veriyi atma işlemidir. Video sıkıştırma yöntemleri kayıplı ya da kayıpsız olabilmektedir. Kayıplı yöntemler, göz ve beynin genellikle görmezden gelebileceği görsel bilgileri çıkartarak, anlaşılmasını engellemeyen kayıplara neden olarak, veri boyutunu düşürmektedir. Kodlama sonrasındaki görüntü ile orijinal görüntü birbiriyle tamamen aynı olmayabilir. Kayıpsız sıkıştırma ise tersine sadece gereğinden fazla bilgileri göz ardı etmektedir.

Kodlama ve kod çözme işlemleri, yazılım ya da donanım, ya da her ikisinin birleşimi ile gerçekleştirilmektedir. Kodlayıcıların sıkıştırma oranları 2:1 oranından 100:1 oranına kadar değişmektedir. Yüksek sıkıştırma oranlarının kullanımı daha kötü resim kalitesi, renklerin solması, yapay doku ve kirliliklerin belirmesi, nesnelerin kenarlarında bulanıklıklar, hatta hiç anlaşılmayan bir görüntü ile sonuçlanmaktadır.

1990'lı yılların sonlarında en baskın teknik, DCT olarak bilinen üç aşamalı bir algoritmadır [24]. DCT, bir resimdeki fiziksel olarak yakında olan ya da ardışık resimler içinde bulunan aynı değerde olabilen piksellerin komşuluklarını kullanmaktadır. Makro bloğu oluşturan her bir blok üzerinde, bloğu oluşturan piksellerin renk değerlerini frekans uzayında ifade edebilmek için 8x8 matematiksel bir dönüşüm uygulanır. Bu dönüşüm sonrasında elde edilen 8x8 blok artık verileri değil, bu verilere ait dönüşüm katsayılarını temsil etmektedir. Bir niceleme süreci, gerekli sıkıştırma seviyesine bağlı olarak, en az katkısı olan görsel bilgileri çıkarır. Örneğin dönüştürülen verinin %50'sinin kaybı, görsel bilgide sadece %5 kayıp ile sonuçlanmaktadır. Kayıpsız bir

teknik olan entropi kodlama, gerçekten de gereksiz olan tüm bitleri atar. Şekil 4.1'de DCT kodlama ve kod çözme işleminin blok diyagramı gösterilmektedir.



Şekil 4.1. Video kodlama ve kod-çözme

4.1. MPEG Standartları

Hareketli Görüntü Uzmanları Birliği (MPEG), video görüntülerini ve ses sinyallerini sıkıştırmak için DCT kullanılan, dünyada yüksek kaliteli sayısal videolarda ortak bir dil sağlama amacıyla, bir dizi standartlar tanımlamıştır. Bu standartlar, her bir çerçeveyi JPEG algoritmasını kullanarak sıkıştırıp ardışık çerçevelerde aynı kalan verileri atmaktadır. MPEG biçimleri asimetriktir, bunun anlamı bir çerçevenin sıkıştırılmasının, o sıkıştırılmış çerçevenin açılmasından daha uzun sürmesidir. Nedeni dosya boyutunun düşürülmesi için bir dizi güçlü hesaplamalar gerektirmesidir. Bununla birlikte sonuçlar oldukça etkileyicidir:

- MPEG-1, sıradan bir CD'den VHS kalitesindeki videoları sabit bir veri hızında (1,5 Mbit/s) oynatabilmek için tasarlanmıştır. 1993'de çıkartılan standart 1,5 Mbit/s hızına kadar çıkabilen veri aktarım hızı, 192 Kbit/s stereo ses kalitesi, 30 fps'de 352x240 çözünürlük sağlamaktadır. Bir CD-ROM disk üzerinde 70 dakikalık iyi kalitede görüntü ve sesin depolanmasına izin vermektedir [25].

- 1990'larda, ikinci bir ihtiyaç olarak daha yüksek veri hızını destekleyen bir standart olarak geliştirildi ve 1,5 Mbit/s'den 15 Mbit/s veri hızlarına çıkabilen, standart tanımlı televizyonu kodlama yeteneğine sahip MPEG-2 standardı oluşturuldu. MPEG-1'e geriye dönük uyumlu olan MPEG-2, çok kanallı ses kodlamasını da desteklemektedir. MPEG-2, MPEG-1'in 4 katı olan 30 fps'de 704x480 çözünürlüğe sahiptir, Sayısal Uydu Sistemi (DSS) ve DVD-Video gibi yayın ve eğlence uygulamaları için gereken yüksek talepleri karşılamak için optimize edilmiştir [26].

- MPEG-3, MPEG-2 içerisine eklenmiş olarak, HDTV için tasarlanmıştır.

- MPEG-4 üzerindeki çalışmalara 1993 yılında başlanmıştır. MPEG-4, kaydedilmiş video görüntü ve seslerinin bilgisayarda oluşturulan karşılıklarının bir karışımını içerebilen, bir düşük bant genişlikli çoklu ortam biçimidir. Daha önemlisi, MPEG-4 görsel, işitsel ve görsel-işitsel içeriklerin *ortam nesneleri* gibi ayrık olarak ifade edilmesi için standartlaştırılmış yollar sağlamaktadır. Bu nesneler doğal ya da suni temelli olabilmektedir. Bunun anlamı görüntüler bir kamera ile kaydedilmiş olabileceği gibi da bilgisayar ortamında tasarlanmış olabilir. MPEG-4 tarafından yapılan belki de en büyük gelişim, bir sahne içinde izleyicilerin ve dinleyicilerin, ortam nesneleri ile etkileşimde bulunmasına izin vermesidir [27].

- MPEG-7, resmi adı *Çoklu Ortam İçerik Tanımlama Ara yüzü* olan standart, bir bilgisayar kodu ya da bir cihaz tarafından erişilebilecek ya da onlara aktarılabilecek, bilginin anlamının yorumlanmasına belirli bir derece destek verebilecek, çoklu ortamın içeriğinin tanımlanması için bir standart oluşturulmasına yardımcı olmaktadır [28].

MPEG video, M-JPEG'e göre daha düşük bant genişliğine ihtiyaç duymaktadır. M-JPEG dosyalar esasında bir dizi sıkıştırılmış resimdir. Kullanılan dahili çerçeveler ya da uzaysal sıkıştırma, videodaki her bir çerçevenin beraberinde fazlalıkların da

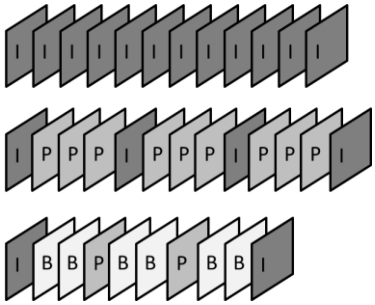
bulunmasına neden olmaktadır. MPEG ise bunların yanında, harici çerçeve veya geçici sıkıştırma gibi farklı yöntemler kullanarak video çerçeveleri arasındaki gereksiz fazlalıkları yok etmektedir. Bir videodan ardışık iki çerçeve incelendiğinde, aralarındaki zamansal fark bir saniyenin 25'te biri olduğundan, aralarında çok küçük farklılıkların bulunduğu görülmektedir. Bu yüzden MPEG, çerçevelerin tamamı yerine değişiklikleri kaydederek veri yoğunluğunu düşürmektedir. MPEG standartları Tablo 4.1'de listelenmektedir.

Tablo 4.1. MPEG Standartları

Kısaltma	Açıklama	Standart Adı	Yayım Tarihi
MPEG-1	Hareketli görüntüleri ve onlara ait sesleri sayısal olarak depolamak amacıyla 1,5 Mbit/s hızına kadar destekleyen kodlama	ISO/IEC 11172	1993
MPEG-2	Bir önceki versiyonun eksikliklerinin giderildiği ve ihtiyaca yönelik yenilendiği genel kodlama	ISO/IEC 13818	1995
MPEG-3	MPEG-2 içersine eklenen ses kodlaması (Standardı MPEG-2 içersinde tanımlanmıştır)		
MPEG-4	Görsel ve duyusal nesnelerin kodlanması	ISO/IEC 14496	1999
MPEG-7	Çoklu ortam içerik tanımlama arayüzü	ISO/IEC 15938	2002
MPEG-21	Çoklu ortam uygulamaları için açık bir çatı tanımlaması (Sayısal haklar/izinler/kısıtlamalar)	ISO/IEC 21000	2001

MPEG video akımları Resim Grubu (GOP) olarak bilinen art arda sıralanmış çerçeveler kümesinden oluşmaktadır. Her bir grup, sadece biri tüm çerçeveyi gösteren ve dahili çerçeve sıkıştırması kullanılarak sıkıştırılmış, 8 ile 25 çerçeve içermektedir. Bir JPEG resmine benzeyen tüm çerçeve, I çerçevesi olarak bilinmekte ve çevresindekiler geçici sıkıştırılmış çerçevelerdir, sadece değişen verileri göstermektedirler. Kodlama anında güçlü hareket tahmin teknikleri, komşu çerçeveleri karşılaştırarak hareket eden alanların yerini belirler, her birinin bir çerçeveden diğerine nasıl hareket ettiğini gösteren vektörleri tanımlamaktadırlar. Bu vektörler kaydedilerek, gerekli olan veri büyük ölçüde azaltılabilir. P (tahmin) çerçeveler sadece bir önceki çerçeveye bakarken, B (iki yönlü tahmin) önceki ve sonraki çerçevelere de bakmaktadır.

Sıkıştırma tekniklerinin bu birleşimi MPEG'i son derece ölçeklenebilir yapmaktadır. Resim gruplarında daha çok B ve P çerçevelerin kullanılması veri miktarını daha da aşağı çekmektedir. Resim gruplarını oluşturan örnek MPEG çerçeve dizilimleri Şekil 4.2'de gösterilmektedir.



Şekil 4.2 Örnek MPEG çerçeve dizilimleri

4.2. MPEG-4 AVC (H.264) Standardı

4.2.1. ISO Temelli Çoklu Ortam Dosya Yapısı

MPEG-4 AVC, ISO temelli çoklu-ortam dosya yapısını kullanmaktadır. Dosyalar, özelliklerine göre kutu adı verilen bir dizi nesneler topluluğu olarak biçimlendirilmişlerdir. Tüm veriler bu kutular içinde depolanmış (Tablo 4.2) ve bu kutular dışında başka veri bulunmamaktadır [29, 30].

MPEG-4 AVC kodlanmış bir dosyadaki verilere ulaşmak için nesne temelli dosya yapısının çözülmesi gerekmektedir.

ISO temelli çoklu ortam dosyası, formatına göre herhangi bir başlangıç imzası içerebilir ve ilave olarak Dosya Tipi Kutusuna da sahip olabilirler.

Nesne, bu terminolojide kutu olarak adlandırılmaktadır. Kutular, boyut ve tip bilgisini veren bir başlık ile başlarlar. Başlıklar kısa (32 bit) veya genişletilmiş (64 bit) türde ve boyutta olabilirler. Standart tüm kutular kısa tiptedirler, birçok kutu da kısa

boyutu kullanırlar. Tipik olarak sadece Ortam Veri Kutuları 64-bit'lik boyuta ihtiyaç duymaktadırlar.

Tablo 4.2. ISO temelli dosya yapısında bulunan tüm bileşenler

ftyp					
pdin					
moov					
	mvhd				
	trak				
		tkhd			
		tref			
		edts			
			elst		
		mdia			
			mdhd		
			hdlr		
			minf		
				vmhd	
				smhd	
				hmhd	
				nmhd	
				dinf	
					dref
				stbl	
					stsd
					stts
					ctts
					stsc
					stsz
					stz2
					stco
					co64
					stss
					stsh
					padb
					stdp
					sdtp
					sbgp
					sgpd
					subs
	mvex				
		mehd			
		trex			
	ipmc				

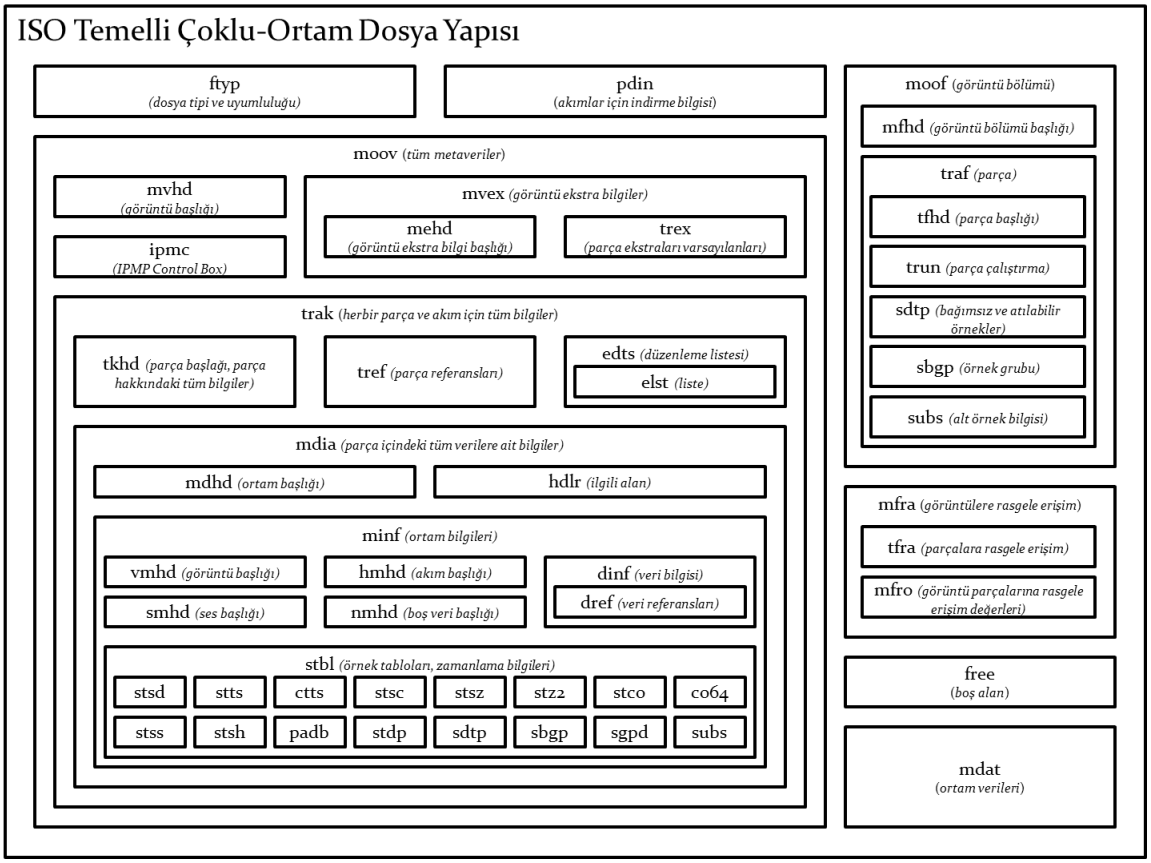
moof					
	mfhd				
	traf				
		tfhd			
		trun			
		sdtp			
		sbgp			
		subs			
mfra					
	tfra				
	mfro				
mdat					
free					
skip					
	udta				
		cprt			
meta					
	hdlr				
	dinf				
		dref			
	ipmc				
	iloc				
	ipro				
		sinf			
			frma		
			imif		
			schm		
			schi		
	iinf				
	xml				
	bxml				
	pitm				
	fiin				
		paen			
			fpar		
			fecr		
		segr			
		gitn			
		tsel			
meco					
	mere				

Boyut bilgisi, kutunun (başlık boyutu, tip boyutu ve içerdiği diğer kutuların boyutları) tamamına aittir. Bu bilgi dosyanın genel ayrıştırılmasına yardımcı olmaktadır. Kutuların tanımlamaları MPEG-4 için oluşturulmuş sözdizimi tanımlama dilinde (SDL) açıklanmıştır. Örnek MPEG-4 SDL ile ifade edilmiş bir nesne Tablo 4.3'de gösterilmektedir. Nesnelerin içindeki alanlar en yüksek değerlikli bayt ilk sırada olacak şekilde depolanmaktadır. Bir bayttan daha küçük veriler tanımlanırken bit dizilimi yüksek değerliklilerden düşük değerliklilere doğru olmaktadır.

Tablo 4.3. MPEG-4 SDL ile ifade edilmiş bir nesne.

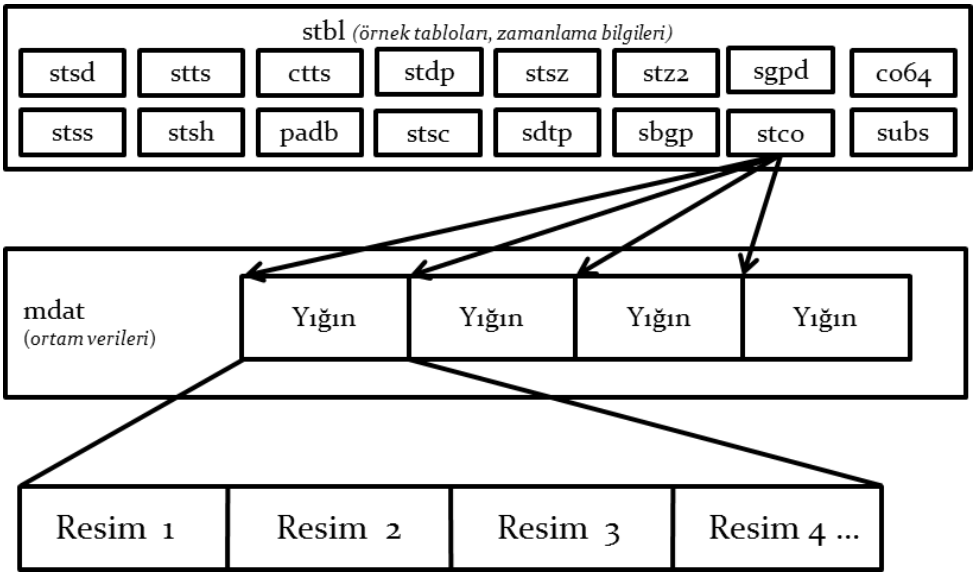
```
aligned(8) class Box (unsigned int(32) boxtype,
optional unsigned int(8)[16] extended_type)
{
     unsigned int(32) size;
     unsigned int(32) type = boxtype;
     if (size==1) {
          unsigned int(64) largesize;
     }
     else if (size==0) {
          // box extends to end of file
     }
     if (boxtype==‘uuid’) {
          unsigned int(8)[16] usertype = extended_type;
     }
}
```

Tablo 4.3’te gösterilen *size* ve *type* alanlarının açıklaması şöyledir; “size” bu kutunun kaç bayttan oluştuğunu gösteren bir tamsayıdır. Eğer “size” değeri “1” ise gerçek boyut bilgisi “largesize” alanında bulunmaktadır. Eğer “size” değeri “0” ise bu kutu dosyanın en sonundaki kutudur, dosya sonuna kadar uzamaktadır. “type” kutunun türünü belirlemektedir, standart kutular kısa türü kullanmaktadırlar. Doğrudan okunabilen kolayca anlaşılması mümkün 4 karakterden oluşmaktadır. Kullanıcı tanımlı uzantılar ise genişletilmiş türü kullanmaktadırlar ve bu tipteki alanlar “uuid” olarak isimlendirilmektedir. Tanınmayan türdeki kutular görmezden gelinebilir ve atlanabilir. Bir çok nesne aynı zamanda bir versiyon numarası ve bayrak bilgileri de içermektedir. Şekil 4.3’te ISO temelli çoklu-ortam dosya yapısında bulunan tüm alanlar gösterilmektedir.



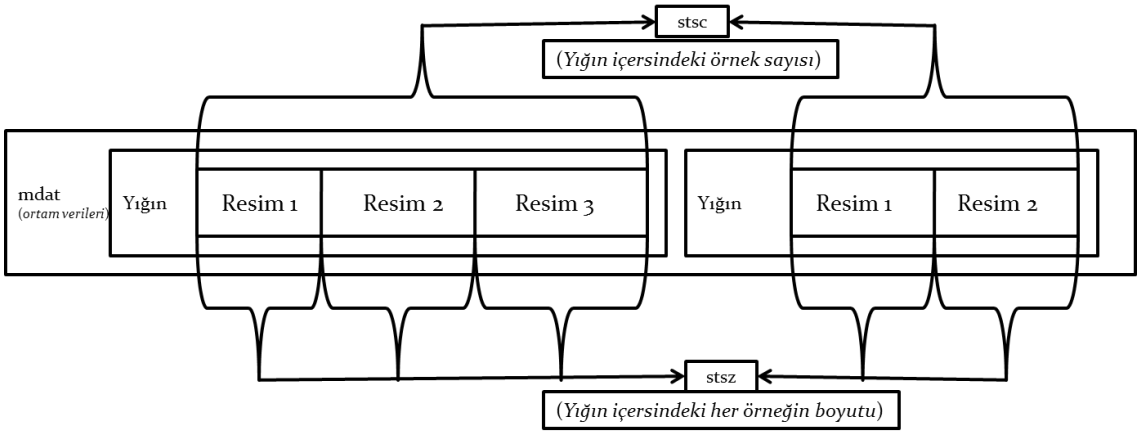
Şekil 4.3. ISO temelli çoklu-ortam dosya yapısı şematik gösterimi

MPEG-4 AVC'de kullanılan ISO temelli dosya yapısı, görüntü kodunun çözülmesi ile ilgili birçok parametreyi ve bilgiyi içermektedir. Ortam veri yığınlarının dosyadaki konumları bu bilgilerden biridir (Şekil 4.4).



Şekil 4.4. Yığın referansları.

Yığınlar içerisinde yer alan resim sayısı ve resimlerin boyutları da bu yapıda belirtilen bilgiler arasındadır (Şekil 4.5)



Şekil 4.5. Resim bilgilerini gösteren parametreler.

Ortam verilerinin kodunun çözülmesi için gelişmiş video kodlama (AVC) standardı kullanılmaktadır [31]. Bu standart ortam verilerini bit düzeyinde ayırarak her

birinin ne anlama geldiğini ve kod çözümünde nasıl kullanılacağını detaylı olarak açıklamaktadır.

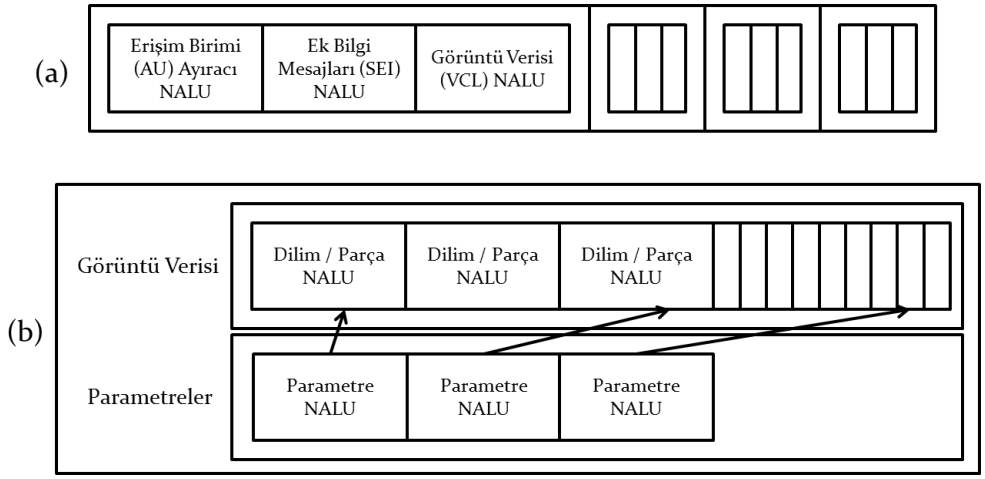
Bit düzeyinde kodlanmış parametrelere sözdizimi elemanı (SE) adı verilmektedir. Sözdizimi elemanlarının tanımlayıcı bir isimleri, ikilik şekilde kodlanmış bir değerleri ve bu değerlerin ne anlama geldiklerini gösteren açıklama tabloları bulunmaktadır. Örnek bir sözdizimi elemanı Tablo 4.4'de gösterilmektedir:

Tablo 4.4. Örnek Sözdizimi Elemanı (SE)

Sözdizimi Elemanı	**İkilik Kodu**	**Değeri**	**Açıklaması**
nal_unit_type	00111	7	Görüntü Bölümü Parametresi
H.264/AVC standartlarında yer alan parametre adı	İkilik olarak kodlanmış değeri	Gerçek değeri	Değere karşılık gelen anlam

Sözdizimi elemanları farklı şekillerde kodlanmaktadırlar. Tüm sözdizimi elemanları ve nasıl kodlandıkları ile ilgili detaylı bilgiler MPEG-4 AVC/H.264 standartlarını tanımlayan dokümanlarda anlatılmaktadır [31].

Ortam verileri, Ağ Soyutlama Katmanı (NAL) birimi adı verilen bir yapı ile paketlenerek saklanmakta ya da akım olarak iletilmektedirler. NAL birimleri kısaca NALU olarak ifade edilmektedir ve içeriklerinde resim, resim parçası (dilimi), kodlanmış resim ile ilgili bilgiler ya da görüntü ile ilgili parametreler bulunmaktadır. NAL birimleri farklı şekillerde bir araya gelerek AVC biçiminde görüntüyü oluşturmaktadırlar. Şekil 4.6'da iki farklı şekilde oluşmuş video akımları görünmektedir.



Şekil 4.6. NAL birimlerinin ardışık ve ayrı dizilimleri

NAL birimlerinin çeşitleri AVC standardındaki şekliyle Tablo 4.5'te gösterilmektedir.

Tablo 4.5. NAL Birimi Türleri

NAL Birimi	**NAL Türü**
1 , 2 , 3 , 4 , 5	Kodlanmış Resim ya da Resim Parçası
6	İlave Geliştirme Bilgisi
7	Görüntü Bölümü Parametreleri
8	Resim Parametreleri
9	Erişim Birimi Ayıracı
10	Görüntü Bölümü Sonu
11	Akım Sonu
12	Doldurma Verisi
13 .. 23	Rezerve
0, 24 .. 31	Tanımsız

NAL birimlerinin kodları çözülürken, bu birimlerin tipi ve içerdikleri parametrelere göre farklı yollar izlenmektedir. MPEG-4 AVC standardında [31] tanımlanan NAL birimi genel yapısı Şekil 4.7'de gösterilmektedir.

nal_unit(NumBytesInNALunit) {	C	Descriptor
forbidden_zero_bit	All	f(1)
nal_ref_idc	All	u(2)
nal_unit_type	All	u(5)
NumBytesInRBSP = 0		
for(i = 1; i < NumBytesInNALunit; i++) {		
if(i + 2 < NumBytesInNALunit && next_bits(24) == 0x000003) {		
rbsp_byte[NumBytesInRBSP++ **]**	All	b(8)
rbsp_byte[NumBytesInRBSP++ **]**	All	b(8)
i += 2		
emulation_prevention_three_byte /* equal to 0x03 */	All	f(8)
} else		
rbsp_byte[NumBytesInRBSP++ **]**	All	b(8)
}		
}		

Şekil 4.7. MPEG-4 AVC standardında tanımlanan NAL birimi genel yapısı

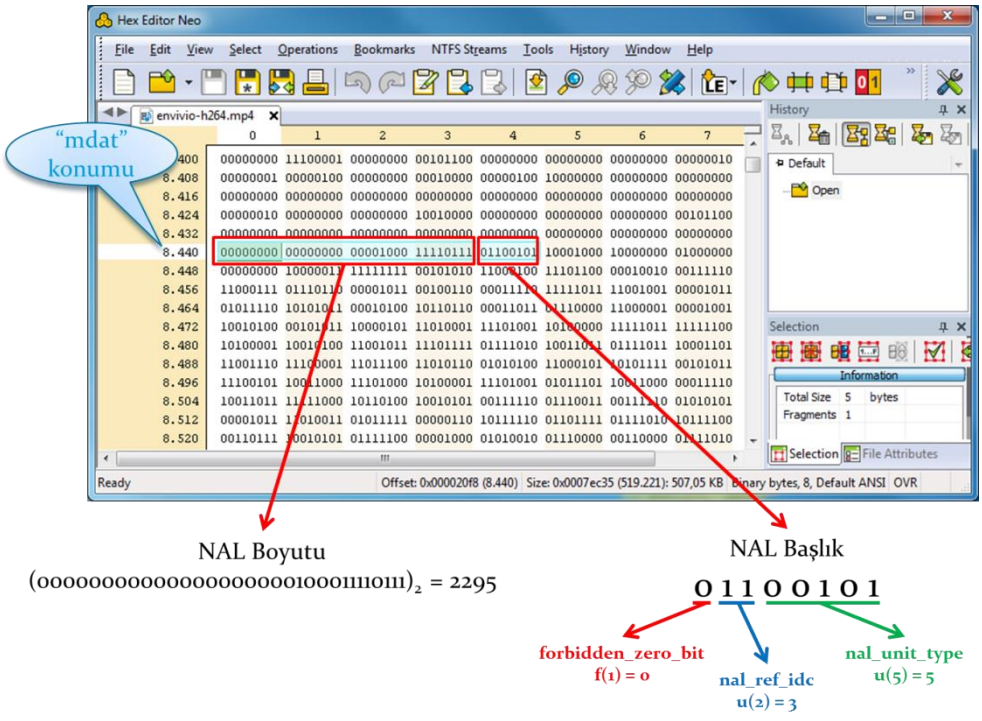
Şekil 4.7'de belirtilen yapıda MPEG-4 AVC/H.264 kodlanmış video verilerinin nasıl çözülecekleri ile ilgili açıklamaları da yer almaktadır. Ham veriler üzerinden bu açıklamalara göre gerekli değerler elde edilmektedir.

NAL birimleri sahip oldukları tür koduna göre, standartlardaki uygun alt çözüm yordamlarına dallanmaktadırlar (Tablo 4.6).

Şekil 4.8'de seçilmiş örnek ISO temelli bir dosya içerisindeki "mdat" bölümünün ilk NAL biriminin ham verilerinin Şekil 4.7'de verilen bilgilere göre yorumlanması gösterilmektedir.

Tablo 4.6. NAL birimleri türleri için alt yordamlar.

NALU Türü	Alt Yordam
1	slice_layer_without_partitioning_rbsp( )
2	slice_data_partition_a_layer_rbsp( )
3	slice_data_partition_b_layer_rbsp( )
4	slice_data_partition_c_layer_rbsp( )
5	slice_layer_without_partitioning_rbsp( )
6	sei_rbsp( )
7	seq_parameter_set_rbsp( )
8	pic_parameter_set_rbsp( )
9	access_unit_delimiter_rbsp( )
10	end_of_seq_rbsp( )
11	end_of_stream_rbsp( )
12	filler_data_rbsp( )



Şekil 4.8. Bir ISO temelli çoklu ortam dosyasındaki NAL biriminin çözülmesi

NAL birimi 4 bayt bir boyut bilgisi ile başlamaktadır. Ardından Şekil 4.8'de belirtilen değerler sıralanmaktadır;

- Zorunlu Sıfır Biti → f(1) sabit uzunlu 1 bitten oluşan parametre, değeri '0'
- NAL Referans Bilgisi → u(2) işaretsiz 2 bit tamsayı, $(11)_2$ değeri '3'
- NAL Birimi Türü → u(5) işaretsiz 5 bit tamsayı $(00101)_2$ değeri ise '5'

şeklinde kodlaması çözülmektedir. NAL Türü değeri 5'in Tablo 4.5'te görüldüğü gibi "Kodlanmış Resim ya da Resim Parçası" olduğu anlaşılmaktadır. NAL biriminin geri kalan bitlerinin kodları çözülmek üzere Tablo 4.6'de NAL türü '5' değerine karşılık gelen "slice_layer_without_partitioning_rbsp()" alt yordamı çalıştırılmaktadır [31].

Tüm NAL birimleri ve sahip oldukları uzantılar (RBSP) yukarıda anlatılan örnekte olduğu gibi çözülmektedir. Resimlerin ham verileri elde edildikten sonra her biri parametreler yardımıyla ilgili standartlarda belirtilen işlemlere tabi tutularak çözülmekte ve görüntüyü oluşturan çerçeveler elde edilmektedir.

Her bir parametrenin ya da elemanın çözülmesi için farklı yöntemler kullanılmaktadır. Özellikle resim ham verilerini sıkıştırmak amacıyla kullanılan yöntemler, karmaşık hesaplamalar ve bunları işleyebilmek için hem kodlayıcılarda hem de kod çözücülerde yüksek işlemci gücü gerektirmektedir. Fakat bunun karşılığında yüksek sıkıştırma başarısı elde edilmektedir.

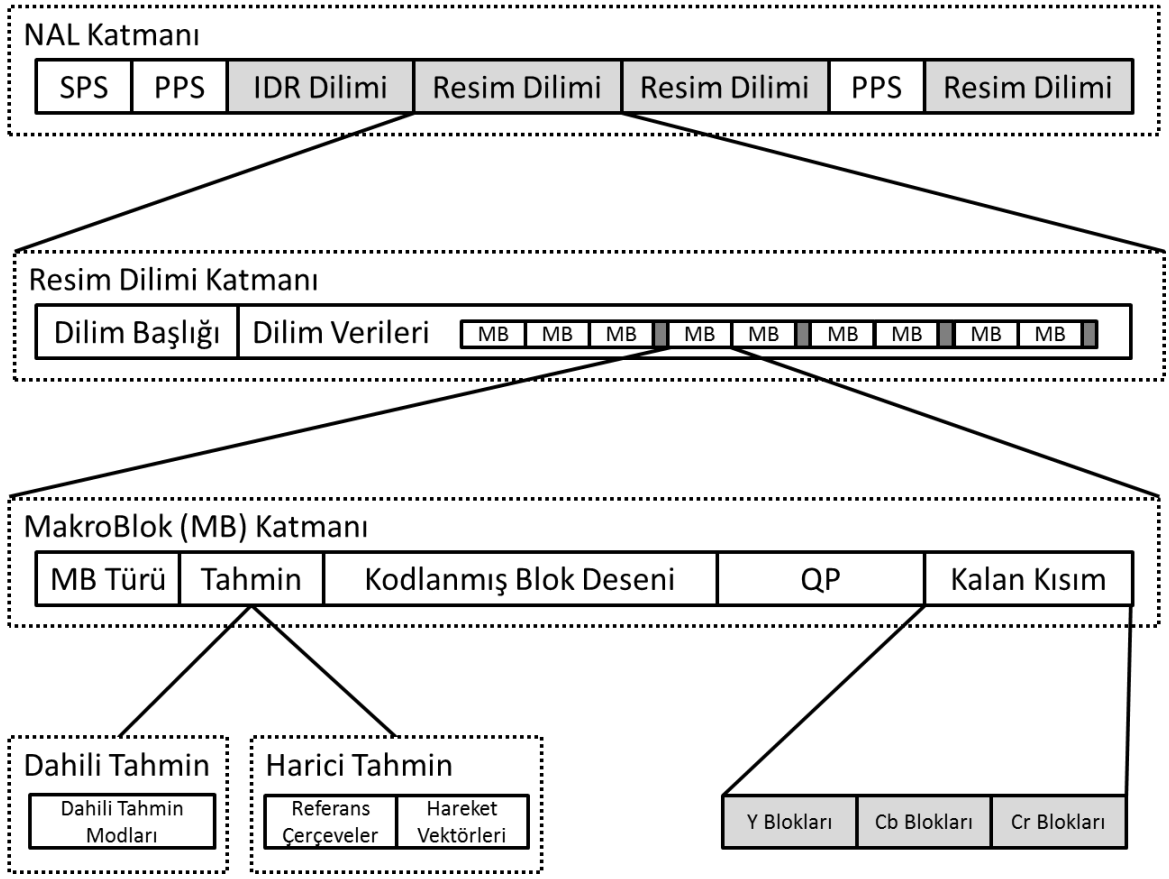
NAL birimleri içerdikleri veriye göre genel olarak iki şekilde sınıflandırılmaktadır;

- Video verisi içeren (Resim Dilimleri, Resim Parçaları ...)
- Video verisi içermeyen (SPS, PPS, ...)

Video verisi içermeyen NAL birimlerinden bazıları video içeren NAL birimleri için parametreleri taşımaktadırlar. Bunlar;

- SPS (Görüntü parçası parametre kümesi, NAL tipi=7)
- PPS (Resim parametre kümesi, NAL tipi=8)

NAL birimlerinin hiyerarşik olarak genel yapısı şekil 4.9'da detaylı olarak belirtilmiştir. NAL katmanı, resim dilimlerinden; resim dilimi katmanı ise makro bloklardan oluşmaktadır. Makro blok katmanında ise resimlere ait kodları çözülecek ham veriler ve parametreler yer almaktadır.



Şekil 4.9. NAL Birimlerini oluşturan sözdizimi elemanlarının hiyerarşik yapısı

SPS ve PPS NAL birimleri ortam verisinin içerisinde olabilecekleri gibi AVC dosya yapısı içerisinde de yer alabilmektedirler.

AVC dosya yapısı içerisinde yer almaları, video akımı ile parametrelerin birbirinden ayrı olmalarını sağlamaktadır. Bu şekilde daha esnek bir yapı elde edilmektedir. Parametreler, video verisi içerisinde tekrar etmemektedir. Akımlarda bir kez gönderilen parametrelerin tekrar gönderilmesi gerekmeyeceğinden bant genişliğinden tasarruf edilebilmektedir.

SPS ve PPS birimlerinin içerisinde resim için çözünürlük, video biçimi, makro blok bilgileri, yerleşim haritası, kod çözücüler için profil ve seviye gibi bilgiler bulunmaktadır. Profil ve seviye parametreleri kod çözücülerin sahip olması gereken özellikleri sınıflandırmaktadır.

Video verisi içeren NAL birimleri ise kodlanmış resim dilimleri ile ilgili sözdizimi elemanlarını barındırmaktadırlar. Dilimlerin veri kısımları, bir dizi kodlanmış makro bloklardan oluşmaktadır. Her bir kodlanmış makro bloğun sahip olduğu temel sözdizimi elemanları şunlardır;

- Makro Blok Türü: I (Dahili kodlanmış), P ya da B (Harici Kodlanmış)
- Tahmin Bilgisi: I türündeki makro bloklar için tahmin modu ya da modları; P veya B makro bloklar için referans çerçeveler ve hareket vektörlerinin bilgileri
- Kodlanmış Blok Deseni: Sıfırdan farklı katsayılar içeren YCbCr bloklarını göstermektedir.
- Niceleme Parametresi (QP)
- Sıfırdan farklı katsayılar içeren bloklar için fark resmi verileri.

Kodlanmış bir görüntü parçası, I resim diliminin özel bir türü olan IDR dilimi ile başlayıp diğer resim dilimleri ile devam etmektedir.

NAL birimleri içerisinde sözdizimi elemanlarının kodlanmış verilerinin bulunduğu bir uzantı (RBSP) bulunmaktadır. Bu uzantı içerisinde sözdizimi elemanları bit düzeyinde çeşitli yöntemlerle kodlanmış olarak ardışık sıradadırlar. Bu yüzden toplam uzunlukları tam bayt olmayabilir. Normal bayt uzunluğunun katlarına tamamlayabilmek için RBSP'nin sonuna sıfır bitleri eklenmektedir [32].

4.2.2. MPEG-4 AVC (H.264) Standardında Kullanılan Sözdizimi Elemanları

4.2.2.1. NAL Birimi

Ağ soyutlama katmanı birimi olarak ifade edilen NAL birimleri, video verilerini ya da parametrelerini barındıran diğer alt elemanları kapsayan bir bölümdür. Kodları çözülecek ham verileri içermektedir.

NAL birimini çözen ve uzantısında hangi sözdizimi elemanı olduğunu tespit ederek ilgili fonksiyona dallanan kod Şekil 4.10'da gösterilmektedir.

```
public nal_unit(byte[] nal_unit_buffer)
    {
        BitReader bitreader = new BitReader(nal_unit_buffer, 0);
        forbidden_zero_bit = bitreader.Read_FixedBits(1);
        nal_ref_idc = bitreader.Read_UnsignedInt(2);
        nal_unit_type = bitreader.Read_UnsignedInt(5);
        IdrPicFlag = (uint)((nal_unit_type == 5) ? 1 : 0);
        switch (nal_unit_type)
        {
            case 1:
            case 5:
              slice_layer_without_partitioning_rbsp(bitreader,sps, pps, this);
                    break;
            case 7:
               seq_parameter_set_rbsp(bitreader);
               sps[SPS_counter] = (C_SPS)rbsp; SPS_counter++; break;
            case 8:
               pic_parameter_set_rbsp(bitreader, sps);
               pps[PPS_counter] = (C_PPS)rbsp; PPS_counter++; break;
            default: break;
        }
   }
```

Şekil 4.10. NAL Biriminin ana yapısının çözülmesi.

4.2.2.2. Görüntü Parçası Parametre Kümesi (SPS RBSP)

Görüntü parçası parametre kümesi, belli bir görüntü bölümü (resimler topluluğu) için bazı ortak parametreleri barındırmaktadır. Bu birim birden fazla resim parametre birimi (PPS) ve tamamlayıcı geliştirme bilgileri (SEI) birimi ile ilişkili olabilmektedir. Bu birimler, içerisinde ilişkili oldukları görüntü parçası parametre kümesinin indisini tutmaktadırlar.

Şekil 4.11'de görünen kod SPS içerisinde yer alan parametreleri ham verilerden elde etmektedir.

```
private void seq_parameter_set_data(BitReader br)
{
   profile_idc = br.Read_UnsignedInt(8);
   constraint_set0_flag = br.Read_UnsignedInt(1);
   constraint_set1_flag = br.Read_UnsignedInt(1);
   constraint_set2_flag = br.Read_UnsignedInt(1);
   constraint_set3_flag = br.Read_UnsignedInt(1);
   constraint_set4_flag = br.Read_UnsignedInt(1);
   constraint_set5_flag = br.Read_UnsignedInt(1);
   reserved_zero_2bits = br.Read_UnsignedInt(2);
   level_idc = br.Read_UnsignedInt(8);
   seq_parameter_set_id = br.Read_ExpGolombUE();

   if (profile_idc == 100 || profile_idc == 110 || profile_idc == 122 ||
       profile_idc == 244 || profile_idc == 44 || profile_idc == 83 ||
       profile_idc == 86 || profile_idc == 118 || profile_idc == 128)
   {
       chroma_format_idc = br.Read_ExpGolombUE();

       if (chroma_format_idc == 3)
       {
          separate_colour_plane_flag = br.Read_UnsignedInt(1);

          if (separate_colour_plane_flag == 0)
             ChromaArrayType = chroma_format_idc;
          else
             ChromaArrayType = 0;
       }
```

```
bit_depth_luma_minus8 = br.Read_ExpGolombUE();
Functions.BitDepthY = (int)(8 + bit_depth_luma_minus8);
bit_depth_chroma_minus8 = br.Read_ExpGolombUE();
Functions.BitDepthC = (int)(8 + bit_depth_chroma_minus8);
qpprime_y_zero_transform_bypass_flag = br.Read_UnsignedInt(1);
seq_scaling_matrix_present_flag = br.Read_UnsignedInt(1);

if (seq_scaling_matrix_present_flag == 1)
{
   int i_max = ((chroma_format_idc != 3) ? 8 : 12);
   seq_scaling_list_present_flag = new uint[i_max];

   for (int i = 0; i < i_max; i++)
   {
      seq_scaling_list_present_flag[i] = br.Read_UnsignedInt(1);

      int[] ScalingList4x4 = new int[i_max];
      uint[] UseDefaultScalingMatrix4x4Flag = new uint[i_max];
      int[] ScalingList8x8 = new int[i_max - 6];
      uint[] UseDefaultScalingMatrix8x8Flag = new uint[i_max - 6];

      if (seq_scaling_list_present_flag[i] == 1)
      {
         if( i < 6 ) C_scaling_list.Apply(br, ScalingList4x4, 16,
                     UseDefaultScalingMatrix4x4Flag[i]);
         else C_scaling_list.Apply(br, ScalingList8x8, 64,
                     UseDefaultScalingMatrix8x8Flag[i]);
      }
    }
  }
}
log2_max_frame_num_minus4 = br.Read_ExpGolombUE();
pic_order_cnt_type = br.Read_ExpGolombUE();

if (pic_order_cnt_type == 0)
{
  log2_max_pic_order_cnt_lsb_minus4 = br.Read_ExpGolombUE();
}
else if (pic_order_cnt_type == 1)
```

```
      {
        delta_pic_order_always_zero_flag = br.Read_UnsignedInt(1);
        offset_for_non_ref_pic = br.Read_ExpGolombSE();
        offset_for_top_to_bottom_field = br.Read_ExpGolombSE();
        num_ref_frames_in_pic_order_cnt_cycle = br.Read_ExpGolombUE();
        offset_for_ref_frame = new int[num_ref_frames_in_pic_order_cnt_cycle];

        for (int i = 0; i < num_ref_frames_in_pic_order_cnt_cycle; i++)
                    offset_for_ref_frame[i] = br.Read_ExpGolombSE();
      }

      max_num_ref_frames = br.Read_ExpGolombUE();
      gaps_in_frame_num_value_allowed_flag = br.Read_UnsignedInt(1);
      pic_width_in_mbs_minus1 = br.Read_ExpGolombUE();
      pic_height_in_map_units_minus1 = br.Read_ExpGolombUE();

      frame_mbs_only_flag = br.Read_UnsignedInt(1);
      if (frame_mbs_only_flag == 0)
      {
         mb_adaptive_frame_field_flag = br.Read_UnsignedInt(1);
      }

      direct_8x8_inference_flag = br.Read_UnsignedInt(1);
      frame_cropping_flag = br.Read_UnsignedInt(1);

      if (frame_cropping_flag == 1)
      {
        frame_crop_left_offset = br.Read_ExpGolombUE();
        frame_crop_right_offset = br.Read_ExpGolombUE();
        frame_crop_top_offset = br.Read_ExpGolombUE();
        frame_crop_bottom_offset = br.Read_ExpGolombUE();
      }
      vui_parameters_present_flag = br.Read_UnsignedInt(1);
 if (vui_parameters_present_flag == 1) vui_parameters = new C_vui_parameters(br);
 initialize_derived_values();
}
```

Şekil 4.11. Görüntü parçası parametre kümesi elemanlarının elde edilmesi

Koddan da görüleceği üzere SPS birimi içerisinde değişik yöntemlerle kodlanmış birçok görüntü parametresi bulunmaktadır. Örnek bir dosya için bu fonksiyon çalıştırıldığında elde edilen değerler Şekil 4.12'de gösterilmektedir.

Name	Value	Type
rbsp	{Mpeg4_Analiz.C_SPS}	object {Mpeg4_Analiz.C_SPS}
bit_depth_chroma_minus8	0	uint
bit_depth_luma_minus8	0	uint
chroma_format_idc	1	uint
ChromaArrayType	0	uint
constraint_set0_flag	0	uint
constraint_set1_flag	0	uint
constraint_set2_flag	0	uint
constraint_set3_flag	0	uint
constraint_set4_flag	0	uint
constraint_set5_flag	0	uint
delta_pic_order_always_zero_flag	0	uint
direct_8x8_inference_flag	1	uint
frame_crop_bottom_offset	0	uint
frame_crop_left_offset	0	uint
frame_crop_right_offset	0	uint
frame_crop_top_offset	0	uint
frame_cropping_flag	0	uint
frame_mbs_only_flag	1	uint
FrameHeightInMbs	45	uint
gaps_in_frame_num_value_allowed_flag	0	uint
level_idc	41	uint
log2_max_frame_num_minus4	5	uint
log2_max_pic_order_cnt_lsb_minus4	6	uint
max_num_ref_frames	4	uint
mb_adaptive_frame_field_flag	0	uint
MbHeightC	8	uint
MbWidthC	8	uint
num_ref_frames_in_pic_order_cnt_cycle	0	uint
offset_for_non_ref_pic	0	int
offset_for_ref_frame	null	int[]
offset_for_top_to_bottom_field	0	int
pic_height_in_map_units_minus1	44	uint
pic_order_cnt_type	0	uint
pic_width_in_mbs_minus1	79	uint
PicHeightInMapUnits	45	uint
PicSizeInMapUnits	3600	uint
PicWidthInMbs	80	uint
PicWidthInSamplesC	0	uint
PicWidthInSamplesL	1280	uint
profile_idc	100	uint
qpprime_y_zero_transform_bypass_flag	0	uint
reserved_zero_2bits	0	uint
separate_colour_plane_flag	0	uint
seq_parameter_set_id	0	uint
seq_scaling_list_present_flag	null	uint[]
seq_scaling_matrix_present_flag	0	uint
SubHeightC	2	uint
SubWidthC	2	uint
vui_parameters	{Mpeg4_Analiz.C_vui_parameters}	Mpeg4_Analiz.C_vui_parameters
vui_parameters_present_flag	1	uint

Şekil 4.12. Örnek bir MPEG-4 AVC çoklu ortam dosyasından okunan SPS değerleri.

Görüntü parçası parametre kümesi içerisinde isteğe bağlı olarak ölçeklendirme listesi ve video kullanılabilirlik bilgisi parametreleri (VUI) de bulunabilir. Ölçeklendirme listesi, kodlayıcı tarafından ters niceleme için belirlenmiş ölçeklendirme matrisidir. Video kullanılabilirlik bilgisi parametreleri ise kod çözümünde gerekli olmayıp video akımı ile ilgili bazı yardımcı bilgiler içermektedir.

4.2.2.3. Resim Parametre Kümesi (PPS RBSP)

Resim parametre kümesi, bir ve ya daha çok kodlanmış resim için ortak parametreleri tutan birimdir. Şekil 4.13'te gösterilen program kod parçası PPS içerisinde yer alan parametreleri ham verilerden elde etmektedir.

```
public C_PPS(BitReader br, C_SPS[] sps)
{
  pic_parameter_set_id = br.Read_ExpGolombUE();
  seq_parameter_set_id = br.Read_ExpGolombUE();
  entropy_coding_mode_flag = br.Read_UnsignedInt(1);
  bottom_field_pic_order_in_frame_present_flag = br.Read_UnsignedInt(1);
  num_slice_groups_minus1 = br.Read_ExpGolombUE();

  if (num_slice_groups_minus1 > 0)
  {
    slice_group_map_type = br.Read_ExpGolombUE();

    if( slice_group_map_type == 0 )
    {
       run_length_minus1 = new uint[num_slice_groups_minus1 + 1];

       for (int iGroup = 0; iGroup <= num_slice_groups_minus1; iGroup++)
              run_length_minus1[iGroup] = br.Read_ExpGolombUE();
    }
    else if( slice_group_map_type == 2 )
    {
       top_left = new uint[num_slice_groups_minus1];
       bottom_right = new uint[num_slice_groups_minus1];

       for(int iGroup = 0; iGroup < num_slice_groups_minus1; iGroup++ )
       {
```

```
        top_left[iGroup] = br.Read_ExpGolombUE();
        bottom_right[iGroup] = br.Read_ExpGolombUE();
      }
    }
    else if (slice_group_map_type == 3 || slice_group_map_type == 4 ||
             slice_group_map_type == 5)
    {
      slice_group_change_direction_flag = br.Read_UnsignedInt(1);
      slice_group_change_rate_minus1 = br.Read_ExpGolombUE();
      SliceGroupChangeRate = slice_group_change_rate_minus1 + 1;
    }
    else if (slice_group_map_type == 6)
    {
      pic_size_in_map_units_minus1 = br.Read_ExpGolombUE();
      slice_group_id = new uint[pic_size_in_map_units_minus1 + 1];

     for (int i = 0; i <= pic_size_in_map_units_minus1; i++)
     slice_group_id[i] = br.Read_UnsignedInt((int)Functions.Ceil(
                         Functions.Log2(num_slice_groups_minus1 + 1)));
   }
  }

  num_ref_idx_l0_default_active_minus1 = br.Read_ExpGolombUE();
  num_ref_idx_l1_default_active_minus1 = br.Read_ExpGolombUE();
  weighted_pred_flag = br.Read_UnsignedInt(1);
  weighted_bipred_idc = br.Read_UnsignedInt(2);
  pic_init_qp_minus26 = br.Read_ExpGolombSE();
  pic_init_qs_minus26 = br.Read_ExpGolombSE();
  chroma_qp_index_offset = br.Read_ExpGolombSE();
  deblocking_filter_control_present_flag = br.Read_UnsignedInt(1);
  constrained_intra_pred_flag = br.Read_UnsignedInt(1);
  redundant_pic_cnt_present_flag = br.Read_UnsignedInt(1);

  if (br.more_rbsp_data())
  {
    transform_8x8_mode_flag = br.Read_UnsignedInt(1);
    pic_scaling_matrix_present_flag = br.Read_UnsignedInt(1);

    if (pic_scaling_matrix_present_flag == 1)
    {
```

```
      int i_max = (int)(6 + ((sps[seq_parameter_set_id].chroma_format_idc != 3) ?
                        2 : 6) * transform_8x8_mode_flag);
      pic_scaling_list_present_flag = new uint[i_max];

      for (int i = 0; i < i_max; i++)
      {
        pic_scaling_list_present_flag[i] = br.Read_UnsignedInt(1);

        int[] ScalingList4x4 = new int[i_max];
        uint[] UseDefaultScalingMatrix4x4Flag = new uint[i_max];

        int[] ScalingList8x8 = new int[i_max - 6];
        uint[] UseDefaultScalingMatrix8x8Flag = new uint[i_max - 6];

        if (pic_scaling_list_present_flag[i] == 1)
        {
          if (i < 6) C_scaling_list.Apply(br, ScalingList4x4, 16,
                                          UseDefaultScalingMatrix4x4Flag[i]);
          else C_scaling_list.Apply(br, ScalingList8x8, 64,
                                    UseDefaultScalingMatrix8x8Flag[i]);
        }
      }

      second_chroma_qp_index_offset = br.Read_ExpGolombSE();
    }
  }
}
```

Şekil 4.13. Resim parametre kümesi elemanlarının elde edilmesi

Örnek bir dosya için bu fonksiyon çalıştırıldığında PPS birimi için elde edilmiş parametreler ve değerleri Şekil 4.14'te gösterilmektedir.

Name	Value	Type
rbsp	{Mpeg4_Analiz.C_PPS}	object {Mpeg4_Analiz.C_PPS}
bottom_field_pic_order_in_frame_present_flag	0	uint
bottom_right	null	uint[]
chroma_qp_index_offset	0	int
constrained_intra_pred_flag	0	uint
deblocking_filter_control_present_flag	1	uint
entropy_coding_mode_flag	1	uint
num_ref_idx_l0_default_active_minus1	0	uint
num_ref_idx_l1_default_active_minus1	0	uint
num_slice_groups_minus1	0	uint
pic_init_qp_minus26	0	int
pic_init_qs_minus26	0	int
pic_parameter_set_id	0	uint
pic_scaling_list_present_flag	null	uint[]
pic_scaling_matrix_present_flag	0	uint
pic_size_in_map_units_minus1	0	uint
redundant_pic_cnt_present_flag	0	uint
run_length_minus1	null	uint[]
second_chroma_qp_index_offset	0	int
seq_parameter_set_id	0	uint
slice_group_change_direction_flag	0	uint
slice_group_change_rate_minus1	0	uint
slice_group_id	null	uint[]
slice_group_map_type	0	uint
SliceGroupChangeRate	0	uint
top_left	null	uint[]
transform_8x8_mode_flag	1	uint
weighted_bipred_idc	2	uint
weighted_pred_flag	0	uint

Şekil 4.14. Örnek bir MPEG-4 AVC çoklu ortam dosyasından okunan PPS değerleri.

Resim parametre kümesi içerisinde isteğe bağlı olarak ölçeklendirme listesi bulunabilir.

SPS ve PPS NAL birimleri diğer birimlerin çözümünde kullanılacakları için önceden elde edilirler ve geçerlilik sürelerince kullanılmaktadırlar. Kodlanmış bir dosya içerisinde birden fazla SPS ve PPS NAL birimi olabilir. Bunlar dizi şeklinde saklanmaktadırlar.

4.2.2.4. Tamamlayıcı Geliştirme Bilgileri (SEI RBSP)

SEI birimi tamamlayıcı bilgi mesajlarını içermektedir. Bu mesajlar kod çözümünde ya da gösteriminde yardımcı olabilmektedirler fakat video çerçevelerinin

kodlarının çözülmesinde gerekli değildirler. Örnek bir MPEG-4 AVC video dosyası içerisindeki SEI mesajları Şekil 4.15'te gösterilmektedir.

"H.264/MPEG-4 AVC codec - options: cabac=1 ref=3 deblock=1:0:0 analyse=0x3:0x13 me=hex subme=6 brdo=1 mixed_ref=1 me_range=16 chroma_me=1 trellis=1 8x8dct=1 cqm=0 deadzone=21,11 chroma_qp_offset=0 threads=1 nr=0 decimate=1 mbaff=0 bframes=3 b_pyramid=0 b_adapt=1 b_bias=0 direct=1 wpredb=1 bime=1 keyint=250 keyint_min=25 scenecut=40 rc=2pass bitrate=4675 ratetol=1.0 rceq='blurCplx^(1-qComp)' qcomp=0.60 qpmin=10 qpmax=51 qpstep=4 cplxblur=20.0 qblur=0.5 ip_ratio=1.40 pb_ratio=1.30"

Şekil 4.15. Örnek SEI mesajları

4.2.2.5. İsteğe Bağlı Ayırıcı Birimler (RBSP)

Ayırıcılar, isteğe bağlı olarak kullanılan, verileri belirli konumlarda sınırlandıran ve sonrasında gelen birim hakkında bilgi veren birimlerdir.

- Erişim Birimi Ayıracı, bir sonraki kodlanmış resmin türünü gösterir.
- Görüntü Bölümü Sonu, bir sonraki resmin anlık kod çözücü yenileme (IDR) türünde olduğunu gösterir.
- Akım Sonu, kodlanmış video akımının sonuna gelindiğini belirtir.

Ayrıca bu sonlandırma birimleri ya da diğer birimleri belirli bir standart boyuta ulaştırmak için bir dizi bayttan oluşan "Doldurma Verisi" de yardımcı bir birim olarak kullanılabilmektedir.

4.2.2.6. Resim Dilimi Katmanı (RBSP)

Resim dilimi katmanları tek katmandan ya da parçalı katmanlardan oluşmaktadır;

- Veri bölümlemenin kullanılmadığı resim dilimi katmanı
- Resim dilimi A veri bölümü katmanı
- Resim dilimi B veri bölümü katmanı
- Resim dilimi C veri bölümü katmanı

Veri bölümlemesi olsun ya da olmasın tüm katmanlar dilim başlığı ve dilim verileri olmak üzere iki ayrı kısımdan oluşmaktadır. Dilim başlığı, kodlanmış dilim verileri için ortak parametreleri barındırmaktadır. Dilim verileri ise kodlanmış makro bloklardan oluşmaktadır.

Dilim başlığında, referans resim listesi yeniden sıralama bölümü, tahmin ağırlık tablosu (isteğe bağlı) ve kodu çözülmüş referans resim işaretleme bölümü (isteğe bağlı) bulunmaktadır. Referans resim listesi yeniden sıralama bölümünde varsayılan referans resim listesi sıralamasını değiştirmek amacıyla komutlar içermektedir. Tahmin ağırlık tablosu, parlaklık ve renk farkı ağırlık değerlerinin ofsetlerini tutar. Kodu çözülmüş referans resim işaretleme bölümü ise resmin uzun vadede referans olarak kullanılabilmesi için bir dizi direktiflerden oluşmaktadır. Resim dilimi başlık parametrelerinin ham verilerden elde edilmesini sağlayan program parçası Şekil 4.16'da gösterilmektedir.

```
public C_slice_header(BitReader br, C_SPS[] sps, C_PPS[] pps, nal_unit nal)
{
  first_mb_in_slice = br.Read_ExpGolombUE();
  slice_type = br.Read_ExpGolombUE();
  pic_parameter_set_id = br.Read_ExpGolombUE();

  active_pps = pps[pic_parameter_set_id];
  active_sps = sps[active_pps.seq_parameter_set_id];

  if (active_sps.separate_colour_plane_flag == 1)
  {
    colour_plane_id = br.Read_UnsignedInt(2);
  }

  frame_num = br.Read_UnsignedInt((int)active_sps.log2_max_frame_num_minus4 + 4);
```

```
if (active_sps.frame_mbs_only_flag == 0)
{
  field_pic_flag = br.Read_UnsignedInt(1);
  MbaffFrameFlag = (uint)((active_sps.mb_adaptive_frame_field_flag == 1 &&
                           field_pic_flag == 0) ? 1 : 0);
  PicHeightInMbs = active_sps.FrameHeightInMbs / (1 + field_pic_flag);
  PicHeightInSamplesL = PicHeightInMbs * 16;
  PicHeightInSamplesC = PicHeightInMbs * active_sps.MbHeightC;
  PicSizeInMbs = active_sps.PicWidthInMbs * PicHeightInMbs;

  if (field_pic_flag == 1) bottom_field_flag = br.Read_UnsignedInt(1);
}

nal.IdrPicFlag = (uint)((nal.nal_unit_type == 5) ? 1 : 0);

if (nal.IdrPicFlag == 1)
{
  idr_pic_id = br.Read_ExpGolombUE();
}

if (active_sps.pic_order_cnt_type == 0)
{
   pic_order_cnt_lsb = br.Read_UnsignedInt((int)
                       active_sps.log2_max_pic_order_cnt_lsb_minus4 + 4);

   if (active_pps.bottom_field_pic_order_in_frame_present_flag == 1 &&
                                                  field_pic_flag == 0)
   {
     delta_pic_order_cnt_bottom = br.Read_ExpGolombSE();
   }
}

if (active_sps.pic_order_cnt_type == 1 &&
     active_sps.delta_pic_order_always_zero_flag == 0)
{
  delta_pic_order_cnt[0] = br.Read_ExpGolombSE();

  if (active_pps.bottom_field_pic_order_in_frame_present_flag == 1 &&
                                                  field_pic_flag == 0)
  {
```

```
      delta_pic_order_cnt[1] = br.Read_ExpGolombSE();
    }
  }

  if (active_pps.redundant_pic_cnt_present_flag == 1)
  {
    redundant_pic_cnt = br.Read_ExpGolombUE();
  }

  if (SLICE_TYPES.Check(slice_type,"B"))
  {
    direct_spatial_mv_pred_flag = br.Read_UnsignedInt(1);
  }

  if (SLICE_TYPES.Check(slice_type, "P") || SLICE_TYPES.Check(slice_type, "SP")
                                         || SLICE_TYPES.Check(slice_type, "B"))
  {
     num_ref_idx_active_override_flag = br.Read_UnsignedInt(1);

     if (num_ref_idx_active_override_flag == 1)
     {
        num_ref_idx_l0_active_minus1 = br.Read_ExpGolombUE();

        if (SLICE_TYPES.Check(slice_type, "B"))
                        num_ref_idx_l1_active_minus1 = br.Read_ExpGolombUE();
     }
  }

  if (nal.nal_unit_type == 20)
      ;//ref_pic_list_mvc_modification(); /* Annex H */
  else
     ref_pic_list_modification = new C_ref_pic_list_modification(br, this);

  if((active_pps.weighted_pred_flag == 1 && ( SLICE_TYPES.Check(slice_type, "P")
    || SLICE_TYPES.Check(slice_type,"SP")))||(active_pps.weighted_bipred_idc == 1
    && SLICE_TYPES.Check(slice_type, "B") ) )
   pred_weight_table = new C_pred_weight_table(br, active_sps, active_pps, this);

if (nal.nal_ref_idc != 0)
                dec_ref_pic_marking = new C_dec_ref_pic_marking(br, this, nal);
```

```
  if (active_pps.entropy_coding_mode_flag == 1 &&
     !SLICE_TYPES.Check(slice_type, "I") && !SLICE_TYPES.Check(slice_type, "SI"))
  {
    cabac_init_idc = br.Read_ExpGolombUE();
  }

  slice_qp_delta = br.Read_ExpGolombSE();
  SliceQPY = 26 + active_pps.pic_init_qp_minus26 + slice_qp_delta;

  if( SLICE_TYPES.Check(slice_type, "SP") || SLICE_TYPES.Check(slice_type, "SI"))
  {
    if (SLICE_TYPES.Check(slice_type, "SP"))
            sp_for_switch_flag = br.Read_UnsignedInt(1);
    slice_qs_delta = br.Read_ExpGolombSE();
  }

  if (active_pps.deblocking_filter_control_present_flag == 1)
  {
     disable_deblocking_filter_idc = br.Read_ExpGolombUE();

     if (disable_deblocking_filter_idc != 1)
     {
       slice_alpha_c0_offset_div2 = br.Read_ExpGolombSE();
       slice_beta_offset_div2 = br.Read_ExpGolombSE();
     }
  }

  if (active_pps.num_slice_groups_minus1 > 0 &&
    active_pps.slice_group_map_type >= 3 && active_pps.slice_group_map_type <= 5)
  {
    slice_group_change_cycle = br.Read_UnsignedInt((int)(Functions.Ceil(
                               Functions.Log2( active_sps.PicSizeInMapUnits /
                               active_pps.SliceGroupChangeRate + 1))));
  }
}
```

Şekil 4.16. Resim dilimi başlık parametrelerinin ham verilerden elde edilmesi

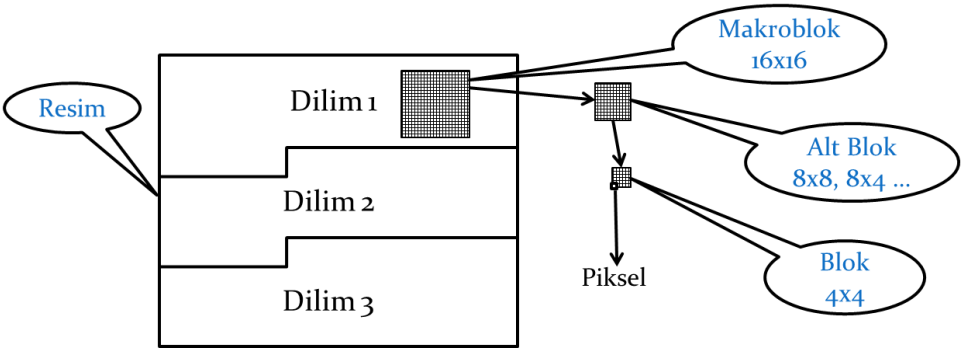
Örnek bir dosya için bu fonksiyon çalıştırıldığında resim dilimi başlığı için elde edilmiş parametreler ve değerleri Şekil 4.17'de gösterilmektedir.

Name	Value	Type
slice_header	{Mpeg4_Analiz.C_slice_header}	Mpeg4_Analiz.C_slice_header
active_pps	{Mpeg4_Analiz.C_PPS}	Mpeg4_Analiz.C_PPS
active_sps	{Mpeg4_Analiz.C_SPS}	Mpeg4_Analiz.C_SPS
bottom_field_flag	0	uint
cabac_init_idc	0	uint
colour_plane_id	0	uint
dec_ref_pic_marking	{Mpeg4_Analiz.C_dec_ref_pic_marking}	Mpeg4_Analiz.C_dec_ref_pic_marking
adaptive_ref_pic_marking_mode_flag	0	uint
difference_of_pic_nums_minus1	0	uint
long_term_frame_idx	0	uint
long_term_pic_num	0	uint
long_term_reference_flag	0	uint
max_long_term_frame_idx_plus1	0	uint
memory_management_control_operation	0	uint
no_output_of_prior_pics_flag	0	uint
delta_pic_order_cnt	{int[2]}	int[]
[0]	0	int
[1]	0	int
delta_pic_order_cnt_bottom	0	int
direct_spatial_mv_pred_flag	0	uint
disable_deblocking_filter_idc	1	uint
field_pic_flag	0	uint
first_mb_in_slice	0	uint
frame_num	0	uint
idr_pic_id	0	uint
mapUnitToSliceGroupMap	null	uint[]
MbaffFrameFlag	0	uint
MbToSliceGroupMap	null	uint[]
num_ref_idx_active_override_flag	0	uint
num_ref_idx_l0_active_minus1	0	uint
num_ref_idx_l1_active_minus1	0	uint
pic_order_cnt_lsb	0	uint
pic_parameter_set_id	0	uint
PicHeightInMbs	0	uint
PicHeightInSamplesC	0	uint
PicHeightInSamplesL	0	uint
PicSizeInMbs	0	uint
pred_weight_table	null	Mpeg4_Analiz.C_pred_weight_table
redundant_pic_cnt	0	uint
ref_pic_list_modification	{Mpeg4_Analiz.C_ref_pic_list_modificatio	Mpeg4_Analiz.C_ref_pic_list_modification
abs_diff_pic_num_minus1	0	uint
long_term_pic_num	0	uint
modification_of_pic_nums_idc	0	uint
ref_pic_list_modification_flag_l0	0	uint
ref_pic_list_modification_flag_l1	0	uint
slice_alpha_c0_offset_div2	0	int
slice_beta_offset_div2	0	int
slice_group_change_cycle	0	uint
slice_qp_delta	-13	int
slice_qs_delta	0	int
slice_type	7	uint
SliceQPY	13	int
sp_for_switch_flag	0	uint

Şekil 4.17. Örnek bir MPEG-4 AVC çoklu ortam dosyasından okunan dilim başlık parametrelerinin değerleri

Dilim verilerini oluşturan makro blok katmanı; PCM kodlanmış örnekleri ya da makro blok başlıkları ile tahmin ve dönüşüm katsayılarını içermektedir. İsteğe bağlı olarak makro blok tahmini, alt makro blok tahmini ve fark resim verileri de bulunabilmektedir. Makro blok tahmini, dahili tahmin modlarından ya da referans indisleri ve hareket vektörlerinden oluşurken alt makro blok tahmini ise sadece referans indisleri ve hareket vektörlerinden oluşmaktadır. Fark resim verileri, kodlanmış blok deseni olarak gösterilen bir dizi artık bloklar içermektedir. Bu bloklar dönüşüm katsayılarının kodlanma yöntemine göre CAVLC ve CABAC olmak üzere ikiye ayrılırlar.

Bir resim bir ya da birden çok dilimden oluşabilir. Dilimler makro bloklardan, makro bloklar da alt bloklardan ve bloklardan meydana gelmektedir (Şekil 4.18).



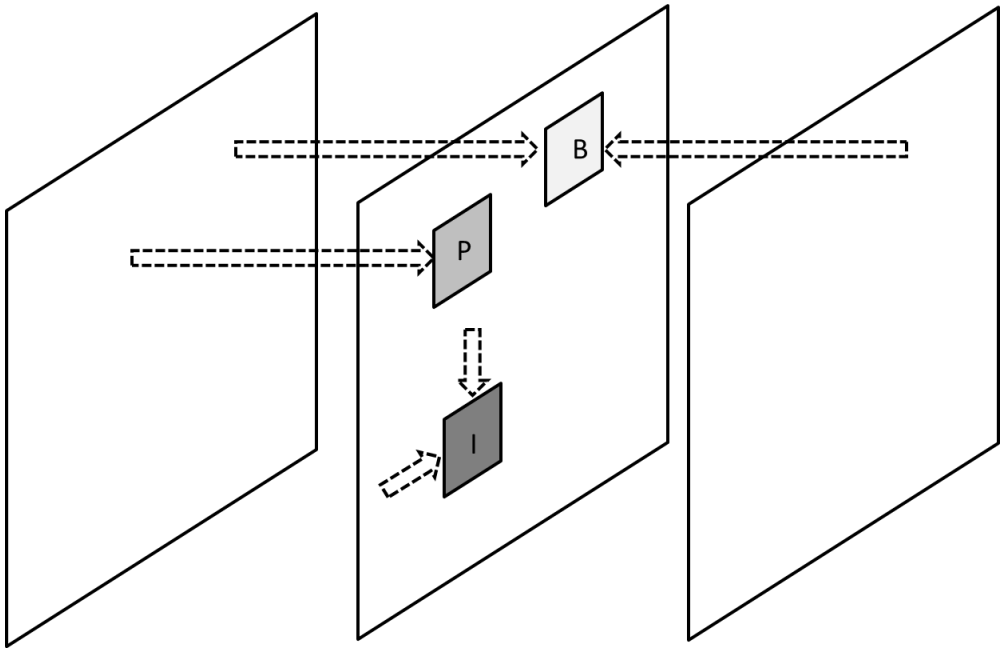
Şekil 4.18. MPEG-4 AVC resim bileşenleri.

CAVLC veya CABAC yöntemleri ile kodlanmış katsayıların değerleri elde edildikten sonra ters dönüşüm işlemleri yapılarak gerçek resim oluşturulmaktadır.

4.2.2.7. Makro Blok Katmanı

Makro blok katmanı tek bir makro bloğun kodunun çözülmesi için gereken tüm sözdizimi elemanlarını içermektedir. Makro blok tipi, makro bloğun ne şekilde kodlandığını (I, SI, P, B) belirtmektedir ve bu kodlama ile ilgili daha fazla bilgi vermektedir. Bir I makro blok tüm dilim türlerinde görülebilmektedir ve diğer hiçbir resim dilimlerinden referans olmadan kodlanmıştır. Bir P makro blok, P resim

dilimlerinde görülmektedir ve bir harici kodlanmış tahmin referansına sahiptir. Son olarak B makro blok ise B türündeki resim dilimlerinde görülmektedir ve harici kodlanmış bir ya da iki tahmin referansına sahiptir. Şekil 4.19'da makro blok türleri ve referans alabilecekleri kaynakları göstermektedir.



Şekil 4.19. Makro blok tipleri ve örnek tahmin referansları

I makro bloklar dahili tahmin yolu ile ya da alternatif olarak doğrudan da kodlanabilmektedirler. I_PCM adı verilen doğrudan kodlamada, makro blok parlaklık ve renk farkı değerleri doğrudan veriler içerisinde bulunmaktadır. Bu I_PCM biçiminin herhangi bir sıkıştırma sağlamadığı ve renk verileri MPEG-4 AVC çoklu ortam dosyasında ya da akımında ham haliyle tutulduğu anlamına gelmektedir.

Makro blok tahmini; sözdizimi elemanları, dahili ya da harici tahminin mevcut makro blok için nasıl gerçekleştirileceğini göstermektedir.

Dahili tahmin, bir I makro bloğun bulunduğu resim dilimi verisi dışında hiçbir veriyi kullanmadan kodlanması esasına dayanmaktadır. Dahili tahmin ile makro bloğun

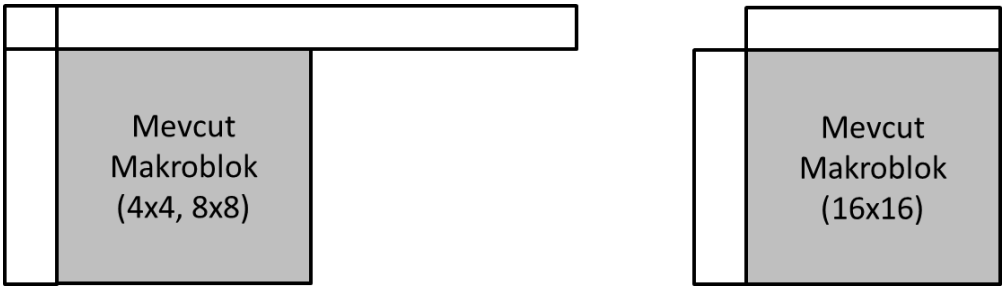
kodlanması için aynı resim dilimi içerisindeki önceki kodlanmış veriler kullanılmaktadır.

Bir I makro blok içinde parlaklık bileşeni Y için 16x16, 8x8, 4x4 olmak üzere üç farklı tahmin blok boyutu seçeneği bulunmaktadır. Renk farkı bileşenleri Cb ve Cr için ise tek bir tahmin bloğu üretilmektedir. Her bir tahmin bloğu Tablo 4.7'de belirtilen belli sayıdaki olası tahmin modlarından bir tanesi ile oluşturulmaktadırlar.

Tablo 4.7. Dahili tahmin türleri

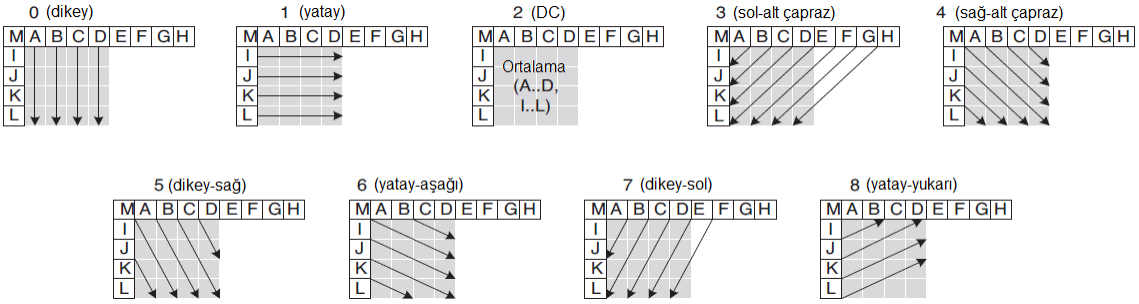
Dahili Tahmin Blok Boyutları	**Açıklama**
16x16 (Parlaklık)	Tek bir 16x16 P tahmin bloğu üretilir ve 4 olası tahmin modu bulunur.
8x8 (Parlaklık)	Her bir 8x8 parlaklık bloğu için bir 8x8 P tahmin bloğu üretilir ve 9 olası tahmin modu bulunur.
4x4 (Parlaklık)	Her bir 4x4 parlaklık bloğu için bir 4x4 P tahmin bloğu üretilir ve 9 olası tahmin modu bulunur.
Renk Farkları	Her bir renk farkı bileşeni için bir P tahmin bloğu üretilir. 4 olası tahmin modu bulunur. Aynı tahmin modu her iki renk farklı bileşeni için de kullanılır.

Parlaklık bileşeni blok boyutu 4x4 ve ya 8x8 seçildiğinde dahili tahmin, mevcut bloğun üst, sol, sağ üst örneklerden ya da bunların bir kombinasyonundan oluşturulur. 16x16 parlaklık blokları ya da renk farkı blokları için ise sol ve ya üst örnekler yada bunların bir kombinasyonu kullanılarak tahmin oluşturulur (Şekil 4.20).

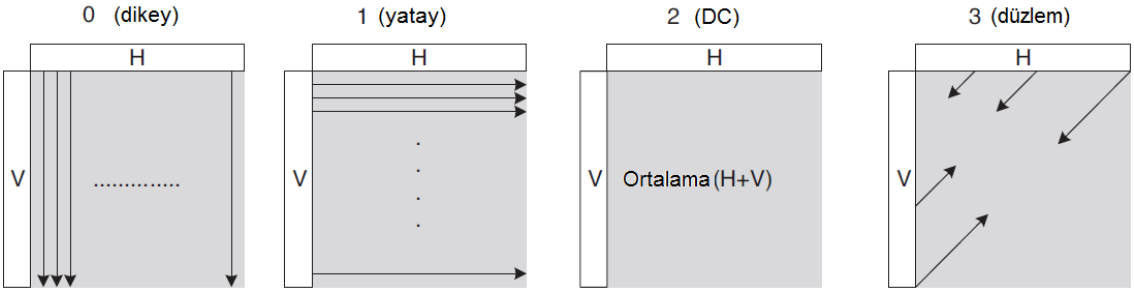


Şekil 4.20. Dahili tahmin için çeşitli blok boyutlarında kullanılan kaynaklar

4x4 ve 16x16 dahili tahmin modları Şekil 4.21 ve Şekil 4.22'de şematik olarak gösterilmektedir. 8x8 dahili tahmin için de 4x4 dahili tahmin modlarının benzerleri kullanılmaktadır.



Şekil 4.21. 4x4 dahili tahmin modları



Şekil 4.22. 16x16 dahili tahmin modları

Kodlayıcılar genellikle tahmin ve kalan resim verisi bit sayısını en aza indirmek için en uygun dahili tahmin modunu seçmektedirler. Tahmin modları kullanılarak kodlanan bir resim, orijinaline kaba bir yaklaşım sağlamaktadır. Fakat tahmin, orijinalden çıkarıldığında geriye kalan resim (residual) orijinalinden oldukça az bilgiye sahiptir ve bu nedenle sıkıştırması daha kolaydır. Şekil 4.23'de örnek bir resimden elde edilen tahmin ve orijinalden çıkarıldıktan sonra kalan resim görülmektedir.

Şekil 4.23. Orijinal resim (sol), tahmin resim (orta), orijinalden tahmin çıkarıldıktan sonra kalan resim (sağ).

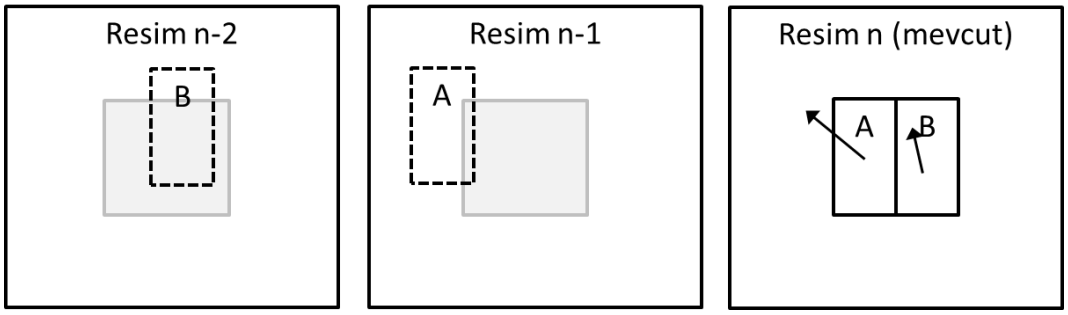
Harici tahmin, bir bloğun parlaklık ve renk farkı bileşenlerinin daha önceden kodu çözülmüş bir referans resimden tahmin edilmesi süreci olarak adlandırılmaktadır. Bir tahmin bölgesinin seçilmesi, tahmin bloğunun üretilmesi ve orijinal bloktan çıkarılarak elde edilen kalan resmin kodlanması işlemlerini içermektedir. Blok boyutları tam bir 16x16 makro bloktan, makro blok bölümleri ya da alt makro bloklara hatta 4x4 bir bloğa kadar farklılık gösterebilmektedir.

Referans resimler daha önce kodları çözülmüş resimlerin bulunduğu bir listeden seçilmektedirler. Mevcut bölge ile referans resim içindeki tahmin bölgesi arasındaki konum farkı bir hareket vektörü olarak isimlendirilmektedir. Hareket vektörü tam sayı, yarım ya da çeyrek değerler ile referans resimdeki bileşenlerin pozisyonunu gösterebilmektedir. Yarım ve çeyrek değerler referans resimdeki örneklerin ara değerlerini elde etmek amacıyla üretilmektedirler.

Tahmin bloğu, P ve B makro bloklar için bir referans resim içindeki tek bir tahmin bölgesinden üretilebileceği gibi, B makro bloklar için referans resimlerin

içindeki iki tahmin bölgesinden de oluşturulabilmektedir. İsteğe bağlı olarak, tahmin bloğu mevcut ve referans resimler arasındaki geçici mesafeye göre ağırlıklı olabilmektedir. B makro bloklarında, bir blok, kalan resim bilgisi ve hareket vektörleri olmadan doğrudan tahmin edilebilmektedir. Kod çözücü bunun için önceden hesaplanmış vektörlerden elde ettiği hareket vektörlerini kullanmaktadır.

Şekil 4.24'te mevcut resim içerisindeki bir makro blok her biri 8x16 boyutunda iki bölüme ayrılmıştır. A bölümü bir önceki resimdeki bir bölgeden, B bölümü ise iki önceki resimdeki bir bölgeden tahmin edilmektedir. A bölümünün (-7.5 , -6.75) değerlerine sahip bir hareket vektörü vardır. Bu vektörden referans bölgenin, x eksenine göre -7.5 birim solda ve y eksenine göre de -6.75 birim yukarıda olduğu anlamına gelmektedir. Aynı şekilde, B bölümünün de (-2.25 , -5) değerlerine sahip bir vektöre sahip olduğu görülmektedir.



Şekil 4.24. P makro blok tahmin örneği.

Harici tahmin, video dosyası ya da akımlarında bulunan önceden kodlanmış resimlerin kod çözücü tarafından erişilebilir olmasını gerektirmektedir. Bu yüzden resim dilimleri alınıp, kodları çözülerek resimler üretilip görüntülenirken aynı zamanda harici tahminlerde kullanılmak için çözülmüş resim ara belleğinde (DPB) depolanmaktadırlar. Bu bellekte resimler indekslenerek ve mevcut makro bloğun P ya da B resim dilimlerinde olmasına bağlı olarak belirli bir sırada listelenmektedir.

- Liste0 (P dilimi): Tüm referans resimler için tek bir listedir ve listedeki ilk resim en son kodu çözülmüş resimdir.

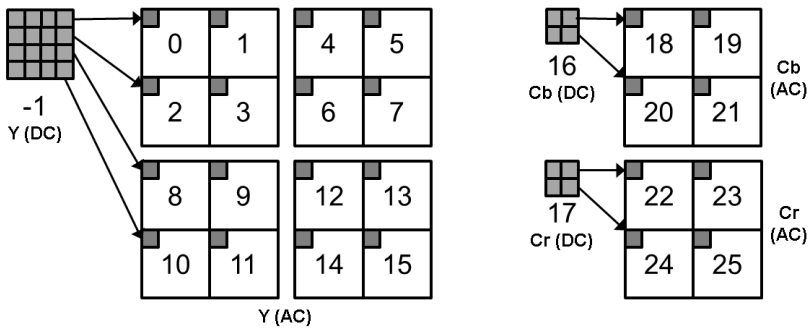
- Liste0 (B dilimi): Tüm referans resimler için bir listedir ve listedeki ilk resim gösterim sırasına göre mevcut resimden önceki resimdir.
- Liste1 (B dilimi): Tüm referans resimler için bir listedir ve listedeki ilk resim gösterim sırasına göre mevcut resimden sonraki resimdir.

Liste0'ın yapısı makro blok ve dilim türlerine bağlıdır. Tablo 4.8'de türlere göre kullanılabilen listeler belirtilmektedir.

Tablo 4.8. Referans resim kaynak listeleri

Resim Dilim Türü	Makro Blok Türü	Referans Resim Kaynakları
P	P	Liste0 (P Dilimi)
B	P	Liste0 (B Dilimi)
B	B	Liste0 (B Dilimi) ve Liste1 (B Dilimi)

Fark resmi, orijinal resim ile tahmin edilen resim arasındaki fark değerlerinden oluşmaktadır ve bu değerler orijinal resim verisine göre daha çok sıkıştırılma potansiyeline sahiptir. Sıkıştırılmış fark resmi, kalan veri adı verilen bir kısımda kodlanmış katsayılar şeklinde, makro blok katmanı içerisinde yer almaktadır. Blok boyutlarına göre fark blokları farklı şekillerde kodlanmaktadırlar. Şekil 4.25'te kodlanmış bir fark resmi bloğunun DC ve AC katsayı değerlerinin konumları ve blok sıraları 4:2:0 örnekleme için gösterilmektedir [32].



Şekil 4.25. Fark resmi parlaklık ve renk farkı değerlerinin katsayılarının dizilimi

Her bir fark resmi bloğu CAVLC ya da CABAC kullanılarak kodlanmıştır. Yine bu yöntemler kullanılarak çözüldüğünde DC ve AC katsayıları yeniden elde edilmektedir.

4.2.3. MPEG-4 AVC (H.264) Standardında Kullanılan Kodlama Yöntemleri

Video verisi içeren NAL birimlerinin kodlanmasında birçok aritmetiksel yöntem kullanılmaktadır. Bu yöntemler veriyi en iyi şekilde sıkıştırmayı amaçlamaktadırlar. Başlıca sıkıştırma yöntemleri şu şekilde sıralanabilir:

- Exp-Golomb değişken uzunluklu kodlama
- Bağlam temelli uyarlamalı değişken uzunluklu kodlama (CAVLC)
- Bağlam temelli uyarlamalı ikili aritmetik kodlama (CABAC)

Exp-Golomb kodlaması, pek çok ortam veri elemanı için ortak, basit ve iyi yapılandırılmış bir değişken uzunluklu kodlama şeklidir. Birçok veri bu kodlama ile kodlanmaktadır [31].

Exp-Golomb kodlaması farklı veri tiplerini kodlamak için alt türlere ayrılmaktadır ve temel olarak dört farklı yöntem bulunmaktadır.

- İşaretsiz tamsayı (unsigned) Exp-Golomb kodlama (ue(v))
- İşaretli tamsayı (signed) Exp-Golomb kodlama (se(v))
- Haritalanmış (mapped) Exp-Golomb kodlama (me(v))
- Kesilmiş (truncated) Exp-Golomb kodlama (te(v))

Standart Exp-Golomb kodlaması (Tablo 4.9) tam sayıları işaretsiz kodlamaktadır.

Tablo 4.9. Exp-Golomb kodlama tablosu

Bit Dizisi	Kod Değeri
1	0
0 1 0	1
0 1 1	2
0 0 1 0 0	3
0 0 1 0 1	4
0 0 1 1 0	5
0 0 1 1 1	6
0 0 0 1 0 0 0	7
0 0 0 1 0 0 1	8
...	...

İşaretli tamsayı Exp-Golomb kodlaması negatif değer de alabilen sayıları kodlamak amacıyla kullanılmaktadır. Tablo 4.10'da kodlamaya ait tablo ve genel formülü gösterilmektedir.

Tablo 4.10. İşaretli Exp-Golomb kodlama tablosu

Bit Dizisi	Kod Değeri	Sözdizimi Eleman Değeri
1	0	0
0 1 0	1	1
0 1 1	2	-1
0 0 1 0 0	3	2
0 0 1 0 1	4	-2
0 0 1 1 0	5	3
0 0 1 1 1	6	-3
0 0 0 1 0 0 0	7	4
...	...	$(-1)^{k+1}$ YukarıYuvarla(k/2)

Haritalanmış Exp-Golomb kodlamada, çeşitli parametrelere göre hazırlanmış bir tablo yardımı ile okunan bit dizisi değeri yorumlanmaktadır (Tablo 4.11).

Tablo 4.11. Haritalanmış Exp-Golomb kodlama tablosu

‘ChromaArrayType’ Parametresinin Değeri ‘1’ ve ya ‘2’ için			
Bit Dizisi	Kod Değeri	Sözdizimi Eleman Değeri	
		Dahili_4x4 ve Dahili_8x8	Harici
1	0	47	0
0 1 0	1	31	16
0 1 1	2	15	1
0 0 1 0 0	3	0	2
0 0 1 0 1	4	23	4
0 0 1 1 0	5	27	8
...	...	...	...

Kesilmiş Exp-Golomb kodlama 0 .. x (x > 0) arasında değer alabilen bir kodlamadır. x > 1 ise sözdizimi elemanı standart Exp-Golomb (ue(v)) ile hesaplanır. x = 1 ise sözdizimi elemanı aşağıdaki şekilde hesaplanmaktadır:

- b = read_bits(1)
- KodDeğeri = !b

Bağlam uyarlamalı değişken uzunluklu kodlama (CAVLC) dönüşüm katsayısı değerlerinin kodlandığı değişken uzunluklu bir kodlama çeşididir. Katsayı dönüşüm tablolarından dizi içerisinde kaç tane “0”, kaç tane “1” ve kaç tane bunlardan farklı katsayı olduğu bulunmaktadır. Belirli kurallar çerçevesinde bu tablo değerleri bir araya getirilerek bit dizisi oluşturulmaktadır [33].

Kod çözülürken de bu katsayı dönüşüm tablolarından faydalanılarak okunan bitler yorumlanarak değerler elde edilmektedir.

“000010001110010111101101” şeklinde (CAVLC) kodlanmış bir bit dizisi için katsayı dönüşüm tablolarından dizi içerisinde kaç tane “0”, kaç tane “1” ve kaç tane bunlardan farklı katsayı olduğu bulunmaktadır. Tablo 4.12’de gösterildiği gibi işlem sürdürülmektedir.

Tablo 4.12 CAVLC Kod Çözümü Örnek Tablosu

Kod	**Eleman**	**Değeri**	**Çıktı Dizisi**						
0000100	Katsayı Bilgisi	Katsayı=5, Birler=3	Boş						
0	T1	+	1						
1	T1	-	-1	1					
1	T1	-	-1	-1	1				
1	Seviye	1	1	-1	-1	1			
0010	Seviye	3	3	1	-1	-1	1		
111	Toplam Sıfırlar	3	3	1	-1	-1	1		
10	Sıfır Konumu	1	3	1	-1	-1	0	1	
1	Sıfır Konumu	0	3	1	-1	-1	0	1	
1	Sıfır Konumu	0	3	1	-1	-1	0	1	
01	Sıfır Konumu	1	3	0	1	-1	-1	0	1
	Son Sıfır		**0 , 3 , 0 , 1 , -1 , -1 , 0 , 1**						

Bloğun verileri:

0, 3, 0, 1, -1, -1, 0, 1

Tam Blok verisi:

0, 3, 0, 1, -1, -1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0

4x4 Blok:

0	3	-1	0
0	-1	1	0
1	0	0	0
0	0	0	0

Bağlam temelli uyarlamalı ikili aritmetik kodlama (CABAC) ortam verilerinin olasılıklarını hesaplayarak görüntü içindeki sözdizimi elemanlarını kayıpsız olarak kodlamak için geliştirilmiş bir yöntemdir [34]. Şu ana kadarki anlatılan kodlamalardan en verimlisi olup, diğerlerinden çok fazla işlemci gücü gerektirmektedir.

Görüntü dilimindeki ilk sözdizimi elemanı çözülmeden önce bağlam değişkenlerinin ilk değer atamaları yapılarak kod çözücü başlatılmaktadır. Ardından sözdizimi elemanının ikilik karşılığı bulunmaktadır. Bir bağlam modeli seçilerek bit bazında kod çözümü döngüsü başlatılmaktadır. Çözülen kod, söz dizimi elemanının ikilik karşılıkları arasında olup olmadığı kontrol edilmektedir [35].

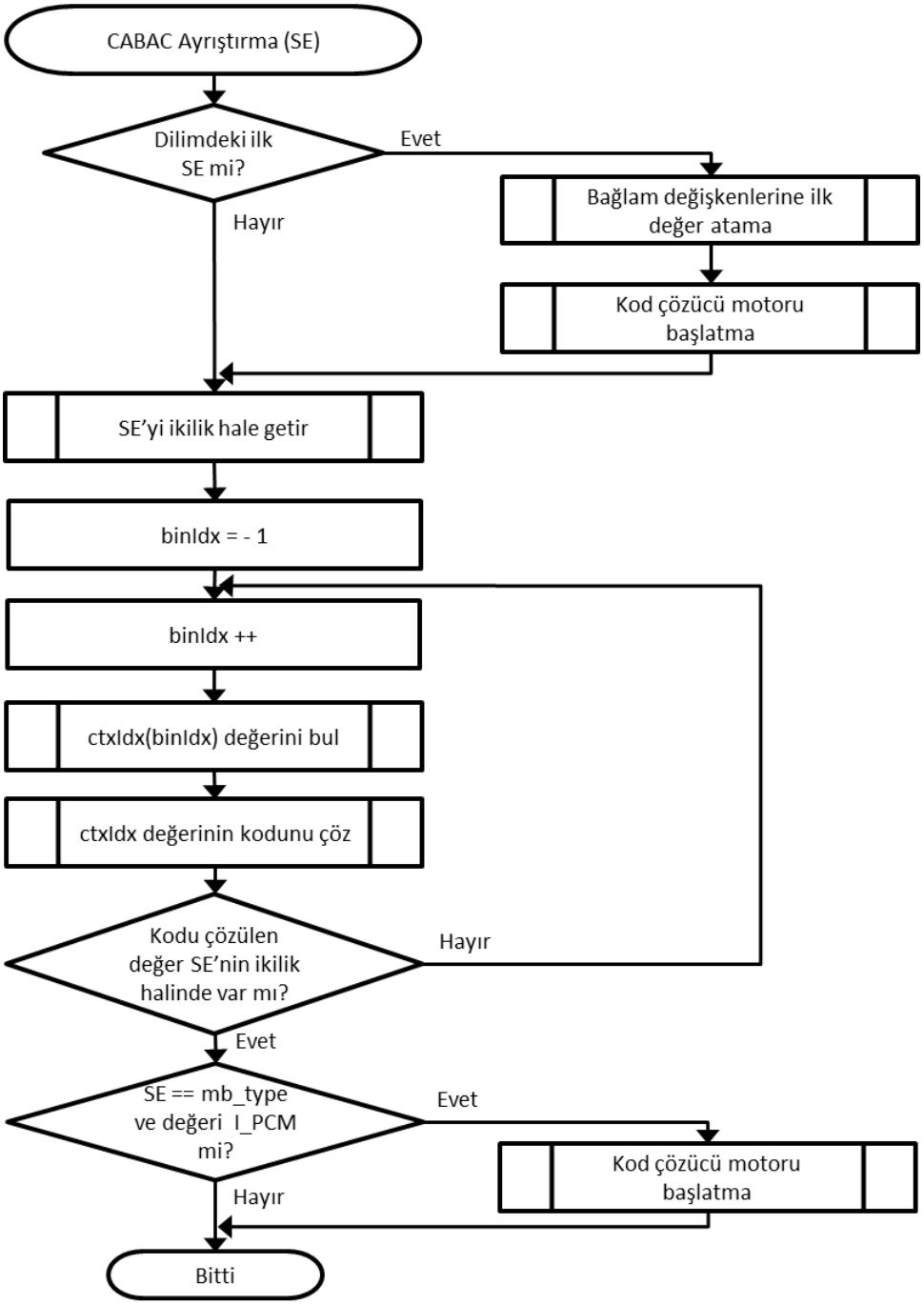
Eğer varsa bu tabloda ikilik değere karşılık gelen değer bu sözdizimi elemanının değeri olmaktadır. Aksi durumda bir bit daha alınarak yeni bir bağlam modeli seçilerek döngüye devam edilmektedir. Değer bulunduktan sonra belli koşullara göre kod çözücü motora başlangıç değerleri yüklenmektedir. CABAC kod çözümü için kullanılan temel fonksiyonlar Şekil 4.26'da, kod çözüm işlemlerinin akış şeması ise Şekil 4.27'de gösterilmektedir.

```
private void DecodeBin(int ctxIdx)
{
    if (bypassFlag == 1)
    {
        DecodeBypass();
    }
    else if (ctxIdx == 276)
    {
        DecodeTerminate();
    }
    else
    {
        DecodeDecision(ctxIdx);
    }
}
```

```
        private void DecodeBypass()
        {
            codIOffset = codIOffset << 1;
            codIOffset = codIOffset | (int)br.read_bits(1);

            if (codIOffset >= codIRange)
            {
                binVal = 1;
                codIOffset = codIOffset - codIRange;
            }
            else
            {
                binVal = 0;
            }
        }

        private void DecodeTerminate()
        {
            codIRange = codIRange - 2;

            if (codIOffset >= codIRange)
            {
                binVal = 1;
            }
            else
            {
                binVal = 0;
                RenormD();
            }
        }

        private void DecodeDecision(int ctxIdx)
        {
            qCodIRangeIdx = (codIRange >> 6) & 3;
            codIRangeLPS = rangeTabLPS[pStateIdx[ctxIdx], qCodIRangeIdx];
            codIRange = codIRange - codIRangeLPS;

            if (codIOffset >= codIRange)
            {
                binVal = 1 - valMPS[ctxIdx];
                codIOffset = codIOffset - codIRange;
                codIRange = codIRangeLPS;
```

```
            if (pStateIdx[ctxIdx] == 0) valMPS[ctxIdx] = 1 - valMPS[ctxIdx];
            pStateIdx[ctxIdx] = transIdxLPS[pStateIdx[ctxIdx]];
        }
        else
        {
            binVal = valMPS[ctxIdx];
            pStateIdx[ctxIdx] = transIdxMPS[pStateIdx[ctxIdx]];
        }

        RenormD();
    }

    private void RenormD()
    {
        while (codIRange < 256)
        {
            codIRange = codIRange << 1;
            codIOffset = codIOffset << 1;
            codIOffset = codIOffset | (int)br.read_bits(1);
        }
    }
```

Şekil 4.26. CABAC ile kodlanmış ortam verilerin ayrıştırılmasında kullanılan temel fonksiyonlar



Şekil 4.27. CABAC ile kodlanmış ortam verilerinin ayrıştırılması akış şeması

Bu temel fonksiyonlar, tüm CABAC kodlanmış sözdizimi elemanlarının çözümünde ortak olarak kullanılmaktadır. Şekil 4.28'de gösterilen kod CABAC kod çözümünde hem temel fonksiyonları hem de sözdizimi elemanına ait MPEG-4 AVC standardındaki tanımlamaları kullanmaktadır [31].

```
        private uint Binarize_mb_type(SyntaxElements SE)
        {
            uint value = MB_TYPES.MB_ERROR;

            maxBinIdxCtx = 6;
            ctxIdxOffset = 3;
            bypassFlag = 0;

            binIdx = -1;

            do
            {
                binIdx++;

                ctxIdxInc = FindctxIdxInc(ctxIdxOffset, binIdx);
                if (ctxIdxInc != -1) ctxIdx = ctxIdxOffset + ctxIdxInc;
                DecodeBin(ctxIdx);
                binString += binVal.ToString();

                if (SLICE_TYPES.Check(slice_header.slice_type, "I") && (SE ==
                     SyntaxElements.mb_type))
                {
                    switch (binString)
                    {
                        case "0": value = MB_TYPES.I_NxN; break;
                        case "100000": value = MB_TYPES.I_16x16_0_0_0; break;
                        case "100001": value = MB_TYPES.I_16x16_1_0_0; break;
                        case "100010": value = MB_TYPES.I_16x16_2_0_0; break;
                        case "100011": value = MB_TYPES.I_16x16_3_0_0; break;
                        case "1001000": value = MB_TYPES.I_16x16_0_1_0; break;
                        case "1001001": value = MB_TYPES.I_16x16_1_1_0; break;
                        case "1001010": value = MB_TYPES.I_16x16_2_1_0; break;
                        case "1001011": value = MB_TYPES.I_16x16_3_1_0; break;
                        case "1001100": value = MB_TYPES.I_16x16_0_2_0; break;
                        case "1001101": value = MB_TYPES.I_16x16_1_2_0; break;
                        case "1001110": value = MB_TYPES.I_16x16_2_2_0; break;
                        case "1001111": value = MB_TYPES.I_16x16_3_2_0; break;
                        case "101000": value = MB_TYPES.I_16x16_0_0_1; break;
                        case "101001": value = MB_TYPES.I_16x16_1_0_1; break;
                        case "101010": value = MB_TYPES.I_16x16_2_0_1; break;
                        case "101011": value = MB_TYPES.I_16x16_3_0_1; break;
                        case "1011000": value = MB_TYPES.I_16x16_0_1_1; break;
```

```
                        case "1011001": value = MB_TYPES.I_16x16_1_1_1; break;
                        case "1011010": value = MB_TYPES.I_16x16_2_1_1; break;
                        case "1011011": value = MB_TYPES.I_16x16_3_1_1; break;
                        case "1011100": value = MB_TYPES.I_16x16_0_2_1; break;
                        case "1011101": value = MB_TYPES.I_16x16_1_2_1; break;
                        case "1011110": value = MB_TYPES.I_16x16_2_2_1; break;
                        case "1011111": value = MB_TYPES.I_16x16_3_2_1; break;
                        case "11": value = MB_TYPES.I_PCM; break;
                    }
                }

            }
            while (binIdx <= maxBinIdxCtx && value == MB_TYPES.MB_ERROR);

            if (value == MB_TYPES.I_PCM)
            {
                codIRange = 510;
                codIOffset = br.Read_Int(9);
            }

            return value;
        }
```

Şekil 4.28. CABAC kodlanmış *mb_type* sözdizimi elemanının çözülmesi

CABAC ve CAVLC yöntemleri, fark resminin çözülmesi ve ters dönüşüm işlemleri için kullanılacak olan DC ve AC katsayılarının elde edilmesini sağlamaktadır. Şekil 4.29'da CABAC ya da CAVLC ile kodlanmış blokların parlaklık ve renk farkı değerlerine ait DC ve AC katsayılarının elde edildiği kod bölümü görünmektedir.

```
public C_residual(BitReader br, C_macroblock_layer macroblock_layer,
                  int startIdx, int endIdx)
{
  C_residual_block residual_block;

  this.macroblock_layer = macroblock_layer;
  this.br = br;
```

```
if (macroblock_layer.slice_header.active_pps.entropy_coding_mode_flag != 1)
   residual_block = new C_residual_block_cavlc(br, macroblock_layer);
else
   residual_block = new C_residual_block_cabac(br, macroblock_layer);

residual_luma(ref i16x16DClevel, ref i16x16AClevel, ref level, ref level8x8,
              startIdx, endIdx);

Intra16x16DCLevel = (int[])i16x16DClevel.Clone();
Intra16x16ACLevel = (int[][])i16x16AClevel.Clone();
LumaLevel = (int[][])level.Clone();
LumaLevel8x8 = (int[][])level8x8.Clone();

if (macroblock_layer.slice_header.active_sps.ChromaArrayType == 1 ||
    macroblock_layer.slice_header.active_sps.ChromaArrayType == 2)
{
  NumC8x8 = 4 / (int)(macroblock_layer.slice_header.active_sps.SubWidthC *
                      macroblock_layer.slice_header.active_sps.SubHeightC);
  ChromaDCLevel = new int[2][];

  for (int iCbCr = 0; iCbCr < 2; iCbCr++)
    if (((MB_TYPES.CodedBlockPatternChroma(macroblock_layer.mb_type,
          macroblock_layer) & 3) == 1) && (startIdx == 0))
    {
     ChromaDCLevel[iCbCr] = new int[4 * NumC8x8];
     residual_block.ReadBlock(BlockCat.ChromaDCLevel, ref ChromaDCLevel[iCbCr],
                              0, 4 * NumC8x8 - 1, 4 * NumC8x8);
    }
    else
    {
      ChromaDCLevel[iCbCr] = new int[4 * NumC8x8];
      for (int i = 0; i < 4 * NumC8x8; i++)
           ChromaDCLevel[iCbCr][i] = 0;
    }
  ChromaACLevel = new int[2][][];

  for (int iCbCr = 0; iCbCr < 2; iCbCr++)
  {
    ChromaACLevel[iCbCr] = new int[NumC8x8 * 4 + 4][];
```

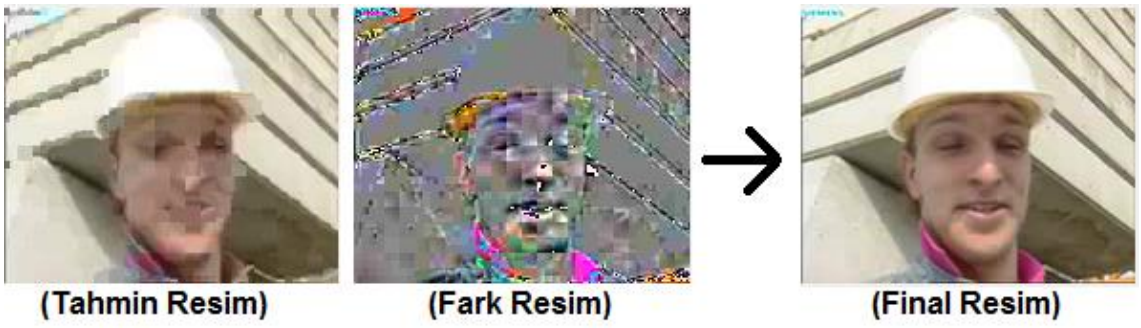
```
      for (int i8x8 = 0; i8x8 < NumC8x8; i8x8++)
       for (int i4x4 = 0; i4x4 < 4; i4x4++)
        if ((MB_TYPES.CodedBlockPatternChroma(macroblock_layer.mb_type,
             macroblock_layer) & 2) == 1)
        {
          residual_block.ReadBlock(BlockCat.ChromaACLevel,
                        ref ChromaACLevel[iCbCr][i8x8 * 4 + i4x4],
                        (int)Functions.Max(0, startIdx - 1), endIdx - 1, 15);
        }
        else
        {
          ChromaACLevel[iCbCr][i8x8 * 4 + i4x4] = new int[15];
          for (int i = 0; i < 15; i++)
             ChromaACLevel[iCbCr][i8x8 * 4 + i4x4][i] = 0;
        }
      }
    }
    else if (macroblock_layer.slice_header.active_sps.ChromaArrayType == 3)
    {
      residual_luma(ref i16x16DClevel, ref i16x16AClevel, ref level,
                    ref level8x8, startIdx, endIdx);

      CbIntra16x16DCLevel = (int[])i16x16DClevel.Clone();
      CbIntra16x16ACLevel = (int[][])i16x16AClevel.Clone();
      CbLevel = (int[][])level.Clone();
      CbLevel8x8 = (int[][])level8x8.Clone();
      residual_luma(ref i16x16DClevel, ref i16x16AClevel, ref level,
                    ref level8x8, startIdx, endIdx);
      CrIntra16x16DCLevel = (int[])i16x16DClevel.Clone();
      CrIntra16x16ACLevel = (int[][])i16x16AClevel.Clone();
      CrLevel = (int[][])level.Clone();
      CrLevel8x8 = (int[][])level8x8.Clone();
    }
}
```

Şekil 4.29. Fark resmi için DC ve AC katsayılarının elde edilmesi

4.2.4. MPEG-4 AVC (H.264) Ters Dönüşüm İşlemleri

MPEG-4 AVC kodlanmış bir ortam verisinden; dosya yapısı, NAL birimleri, resim dilimleri ve makro blok verilerine kadar uzanan hiyerarşik süreç sonunda resmi oluşturmak için gerekli parametreler ve katsayılar elde edilmiş olmaktadır.

En iyi sıkıştırmayı elde edebilmek için tahmin yöntemleri ile kabaca kodlanan resim verisinin orijinal veriden farkı kalan resim verisi olarak sıkıştırılarak saklanmaktaydı. Resmin elde edilmesi için de bu izlenen sürecin tersi yürütülmektedir. Ham verilerden tahmin resmi ile fark resmi elde edilerek birleştirilir ve orijinal resim elde edilmektedir (Şekil 4.30).



Şekil 4.30. Tahmin ve Fark resimlerinden orijinal resmin elde edilmesi

MPEG-4 AVC (H.264) standardından önceki resim ve video sıkıştırma standartları genellikle iki boyutlu *Ayrık Kosinüs Dönüşümü* (2-D DCT) adı verilen yöntemi kullanmaktadırlar (Şekil 4.31). Bu yöntemde dönüşüm bir denklem olarak tanımlanmaktadır [24].

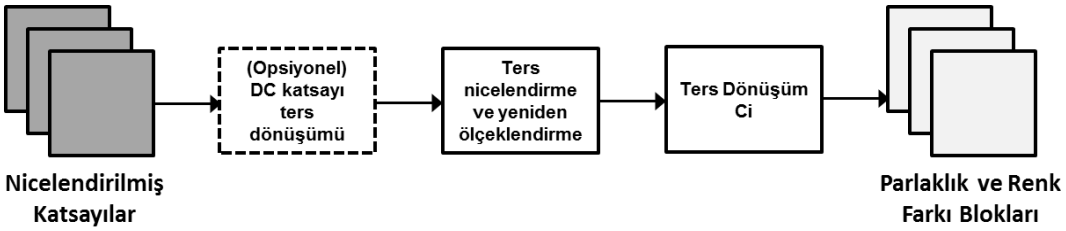
$$X_{ij} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} C_x C_y Y_{xy} \cos \frac{(2j+1)y\pi}{2N} \cos \frac{(2i+1)x\pi}{2N}$$

Şekil 4.31. İki boyutlu ayrık kosinüs dönüşümü (2-D DCT) denklemi

Bu denklemde NxN boyutlu bloklardan oluşan Y_{xy} girdi katsayıları ve X_{ij} ise çıktı ya da fark resmidir.

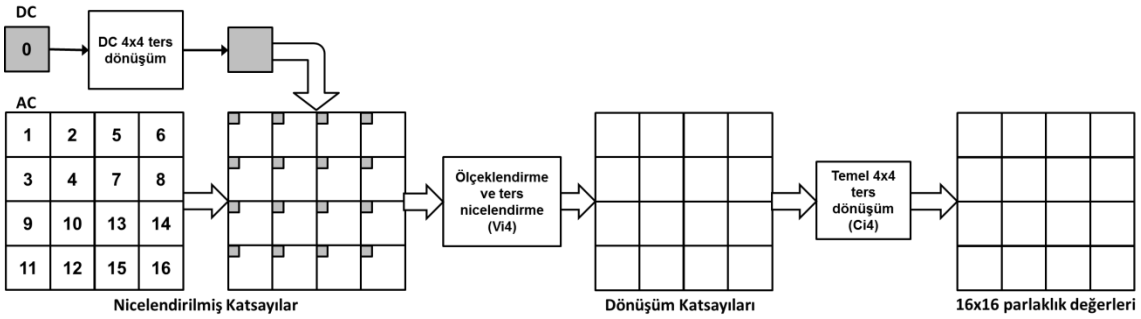
MPEG-4 AVC (H.264) standardında ise dönüşüm ve nicelendirme işlemleri hesaplama karmaşasını azaltmak ve kodlayıcı/kod çözücü uyumsuzluğunu engellemek için sınırlı hassasiyetli tamsayı aritmetiği kullanmaktadır. Bunu gerçekleştirirken, içeriği tamsayı dönüşümü olan, bir temel dönüşüm kullanılmaktadır ve resim fark verisini işlemek için gerekli çarpımları azaltmak için adımlar birleştirilmektedir.

MPEG-4 AVC (H.264) standardındaki ters dönüşüm ve yeniden ölçeklendirme ya da ters nicelendirme işlemleri Şekil 4.32'de gösterilmektedir. Bu adımlar tüm MPEG-4 AVC (H.264) uyumlu kod çözücülerde uygulanmalıdır. DC katsayılarının ters dönüşümü, varsa yeniden ölçeklendirmeden önce gerçekleştirilmektedir. Yeniden ölçeklendirilmiş katsayılara, 4x4 ya da 8x8 bir tamsayı dönüşümü ile ters dönüşüm işlemi yapılmaktadır.

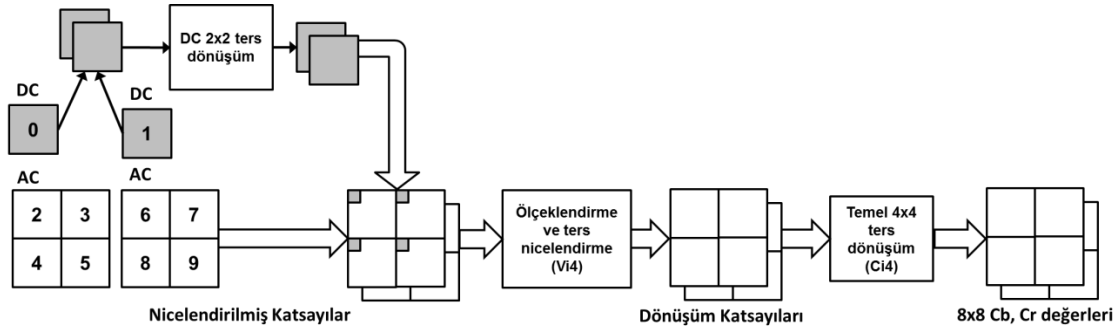


Şekil 4.32. Yeniden ölçeklendirme ve ters dönüşüm adımları

Şekil 4.33'de 16x16 boyutlu parlaklık değerleri ile Şekil 4.34'de 4:2:0 makro blok biçimindeki renk farkı değerlerinin dönüşüm adımları gösterilmektedir.

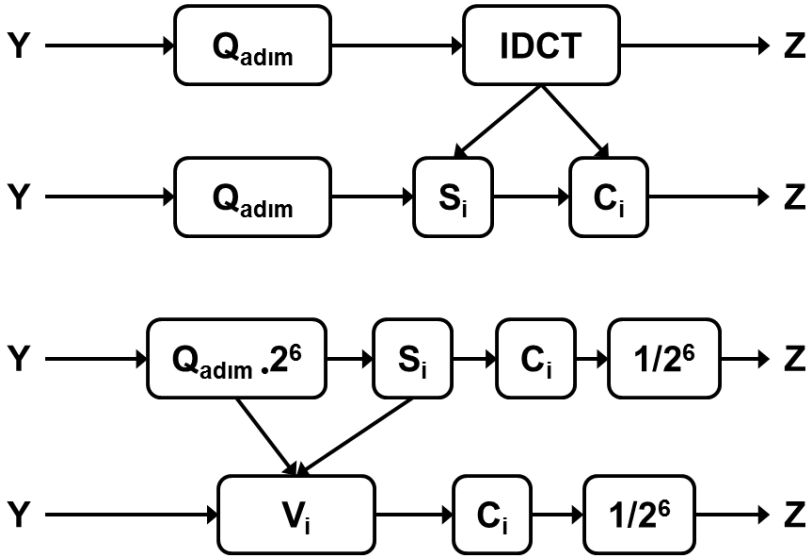


Şekil 4.33. Parlaklık katsayıları için ters dönüşüm adımları



Şekil 4.34. Renk farklı katsayıları için ters dönüşüm adımları

Yeniden ölçeklendirme ve ters nicelendirme işlemlerini ($Q_{adım}$) iki boyutlu ters ayrık kosinüs dönüşümü (IDCT) işlemi takip etmektedir. IDCT süreci bir temel dönüşüm (Ci) ve bir ölçeklendirme matrisi (Si) olmak üzere ikiye ayrılmaktadır. Ölçeklendirme adımı, sabit bir değer (2^6 gibi) aracılığı ile yeniden ölçeklendirilmekte ve nihai sonuç bölünerek ve yuvarlanarak eşitlenmektedir. Vi içerisinde Si ve yeniden ölçeklendirme işlemi birleştirilmektedir (Şekil 4.35).



Şekil 4.35. Yeniden ölçekleme ve ters dönüşüm sürecinin gelişimi

Bu adımlar sonucunda V_i değerinin yaklaşık karşılığı aşağıdaki gibi olmaktadır:

$$V_i \approx S_i \cdot 2^6 \cdot Q_{adım}$$

Bir 4x4 iki boyutlu IDCT Y bloğu için, Z değerlerinin hesaplanması:

$$Z = A^T \cdot Y \cdot A$$

Formülü ile gerçekleştirilmektedir.

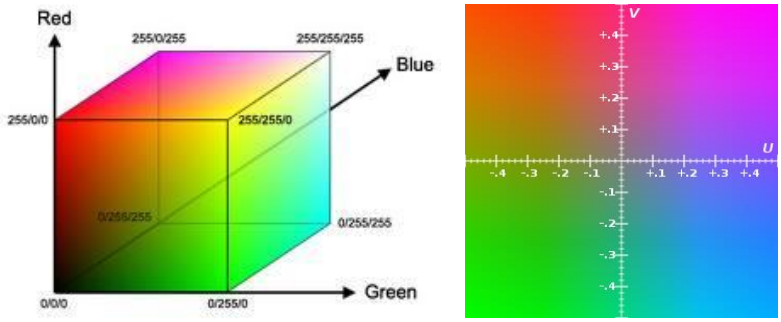
MPEG-4 AVC (H.264) standardındaki belirlenmiş parametre tablolarındaki değerler niceleme parametresi ile (QP) birlikle kullanılarak, 4x4 bir blok için ters dönüşüm formülü aşağıdaki şekle dönüşmektedir:

$$Z = \text{yuvarla}([C_{i4}^T] \cdot [Y \bullet v(QP\%6, n) \cdot 2^{\text{yukarıyuvarla}(QP/6)}] \cdot [C_{i4}] \cdot 1/2^6)$$

Formüldeki v(QP%6, n) değeri V_{i4} = v(QP, n) ifadesinden gelir ve QP değerinin 0…5 arasındaki değerlerine karşılık gelen bir satır ve sütun bilgisidir [32].

4.2.5. MPEG-4 AVC (H.264) Standardında Kullanılan Resim Biçimleri

MPEG-4 AVC standardında YUV renk uzayı kullanılmaktadır (Şekil 4.36). YUV, genellikle video görüntülerini işlemek için kullanılan bir renk sistemidir. Y parlaklık (Luminance), U mavi renk farkı (Cb, Chrominance1), V kırmızı renk farkı (Cr, Chrominance2) sözcüklerinin baş harflerinden oluşan kısaltmadır. İnsan gözünün algıladığı şekilde renkleri gruplayarak yüksek sıkıştırma sağlamakta ve kullanılacak bant genişliğini de düşüren bir renk kodlaması oluşturmaktadır. YUV genellikle analog sinyaller için kullanılmaktadır. Sayısal görüntü işlemede ise YCbCr olarak ifade edilmektedir.



Şekil 4.36. RGB ve YUV Renk Uzayları.

Tablo 4.13. Temel renklerin RGB ve YUV renk uzaylarındaki değerleri.

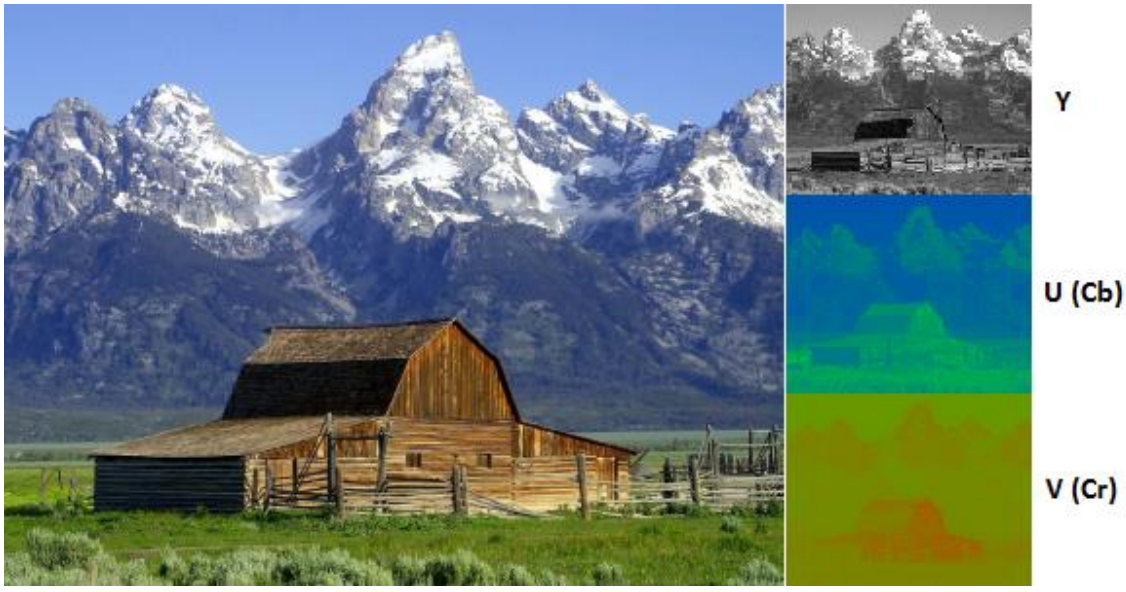
Renk	R	G	B	Y'	Cb	Cr
Siyah	0	0	0	16	128	128
Kırmızı	255	0	0	81	90	240
Yeşil	0	255	0	145	54	34
Mavi	0	0	255	41	240	110
Camgöbeği	0	255	255	170	166	16
Eflatun	255	0	255	106	202	222
Sarı	255	255	0	210	16	146
Beyaz	255	255	255	235	128	128

YUV ve RGB renk uzaylarının birbirine dönüştürülmesinde SDTV için ITU-R BT.601 ve HDTV için ITU-R BT.709 gibi standartlar kullanılmaktadır:

$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.2126 & 0.7152 & 0.0722 \\ -0.09991 & -0.33609 & 0.436 \\ 0.615 & -0.55861 & -0.05639 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.28033 \\ 1 & -0.21482 & -0.38059 \\ 1 & 2.12798 & 0 \end{bmatrix} \begin{bmatrix} Y' \\ U \\ V \end{bmatrix}$$

Kodlanmış bir resim YUV bileşenlerine ayrıldığında Şekil 4.37'de gösterildiği gibi görünmektedir.



Şekil 4.37. Bir resmin YUV bileşenlerine ayrılması.

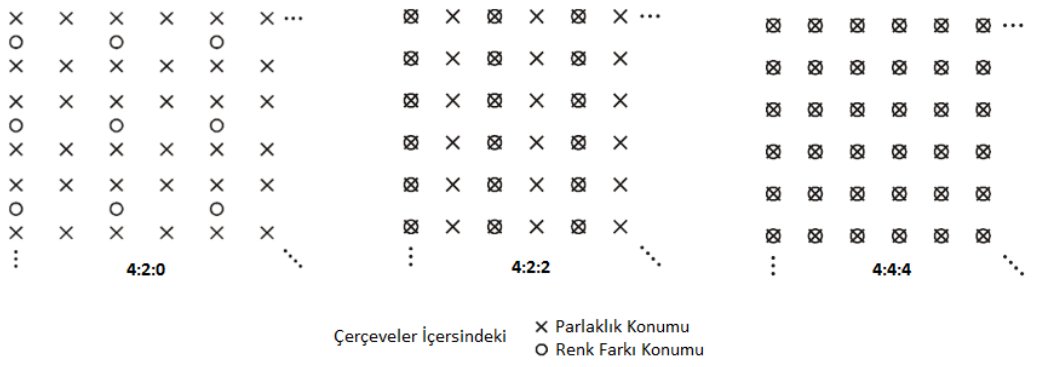
Şekilde de görüldüğü gibi Y bileşeni siyah beyaz parlaklık bilgisini, U bileşeni mavi renk farkını, V bileşeni de kırmızı renk farkını göstermektedir.

MPEG-4 AVC kodlama standardında kullanılan YCbCr biçimleri temelde üç tanedir. Bunlar belli parametre değerleri ile kodlanmış video veri içerisinde verilmektedir. Tablo 4.14'te parametrelere göre seçilen biçimler gösterilmektedir.

Tablo 4.14. MPEG4 AVC içerisinde kullanılan YCbCr türleri.

chroma_format_idc	separate_colour_plane_flag	Chroma Format	SubWidthC	SubHeightC
0	0	monochrome	-	-
1	0	4:2:0	2	2
2	0	4:2:2	2	1
3	0	4:4:4	1	1
3	1	4:4:4	-	-

Tabloda 4.14'te verilen değerler doğrultusunda örnekleme dizileri belirlenmektedir. Siyah-beyaz örneklemede parlaklık değerlerini içeren tek bir örnek dizisi bulunmaktadır. 4:2:0 örneklemede, parlaklık dizisinin yarısı yüksekliğinde ve genişliğinde iki tane renk farkı dizisi içermektedir. 4:2:2 örneklemede, parlaklık dizisiyle aynı yükseklikte ve yarısı genişlikte iki renk farkı dizisi bulunmaktadır. 4:4:4 örneklemede ise ayrı renk düzlemi işareti parametresi değeri "0" ise parlaklık dizisi ile aynı boyutta iki renk farkı dizisi, değeri "1" ise ayrı ayrı işlenecek üç siyah-beyaz örneklenmiş resim dizisi olarak anlaşılmaktadır. Örneklemelere göre parlaklık ve renk farkı konumları Şekil 4.38'de gösterilmektedir [31].



Şekil 4.38. MPEG-4 AVC kodlanmış resim çerçeveleri içerisindeki YCbCr konumları.

4.3. M-JPEG (Hareketli-JPEG) ve Diğer Sıkıştırma Standartları

Birleşik Fotoğraf Uzmanları Grubu (JPEG), en çok bilinen sıkıştırma standartlarından biridir. MPEG'in aksine, M-JPEG, bir çerçevenin diğer çerçevelere göre farkları yerine tüm çerçeveleri sıkıştırarak saklamaktadır. Bu yüzden MPEG'den daha çok alan gerektirmektedir, fakat hızlı sahne geçişlerinde daha verimli ve düzenlenmesi daha kolay olmaktadır. Çok çeşitli sıkıştırma oranlarına sahip olan standart, 2:1 ve 12:1 oranları arasında bir performans göstermektedir. 5:1 veya daha düşük oranlar televizyon yayın kalitesi olarak varsayılmaktadır. Bu değerlerden 12:1'e kadar olan yüksek değerler, daha çok yarı profesyonel ya da tüketici amaçlı kullanımı kabul görmektedir [36].

M-JPEG kodlama, video yakalama yongaları üzerinde mikro kodlar ile yapıldığında en iyi performansla çalışmaktadır. Bu şekilde donanım ile gerçekleştirildiğinde bilgisayarın ana işlemcisi örneğin hard diske veri transferi yapma gibi diğer görevlerle ilgilenmesi için serbest kalmaktadır. Bu algoritma bir kodlayıcı-kod çözücüsü yazılımı ile de çalışabilmektedir.

Sayısal video dünyasında lokomotif rolü üstlenmesine rağmen, M-JPEG için gelecekte yaygın kullanılacak sıkıştırma yöntemlerinden biri olarak görülmemektedir. Yeni sayısal video (DV) biçimleri profesyonel ve yarı profesyonel video pazarında geniş bir rekabet içinde bulunmaktadırlar. Bu biçimlerden beklenen, Analog-sayısal çevrimden daha çok, tamamen sayısal olarak daha iyi resim kalitesinin sunulmasıdır.

MPEG ve M-JPEG video görüntülerinin sıkıştırılmasının öncülerinden olmasına rağmen, o dönemlerde farklı firmalar tarafından geliştirilmiş ürünler de bulunmaktadır. Apple firması tarafından geliştirilmiş CINEPAK, Intel firması tarafından geliştirilmiş IVI, Microsoft tarafından geliştirilmiş Windows için Video (VfW) örnekler arasında bulunmaktadır.

BÖLÜM 5

5. MPEG-4 AVC (H.264) STANDARDINDA KODLANMIŞ VİDEOLARIN ÇERÇEVELERİNİN ELDE EDİLMESİ

MPEG-4 AVC (H.264) standardında kodlanmış video verilerinden çerçevelerin elde edilmesi için aşağıdaki adımlar uygulama yazılımında gerçekleştirilmiştir:

- ISO temelli çoklu ortam dosya yapısının çözülmesi
- AVC kod çözümü ile video parametrelerin ve video çerçevelerinin ham verilerinin elde edilmesi
- Video çerçevelerinin oluşturulması ve görüntülenmesi

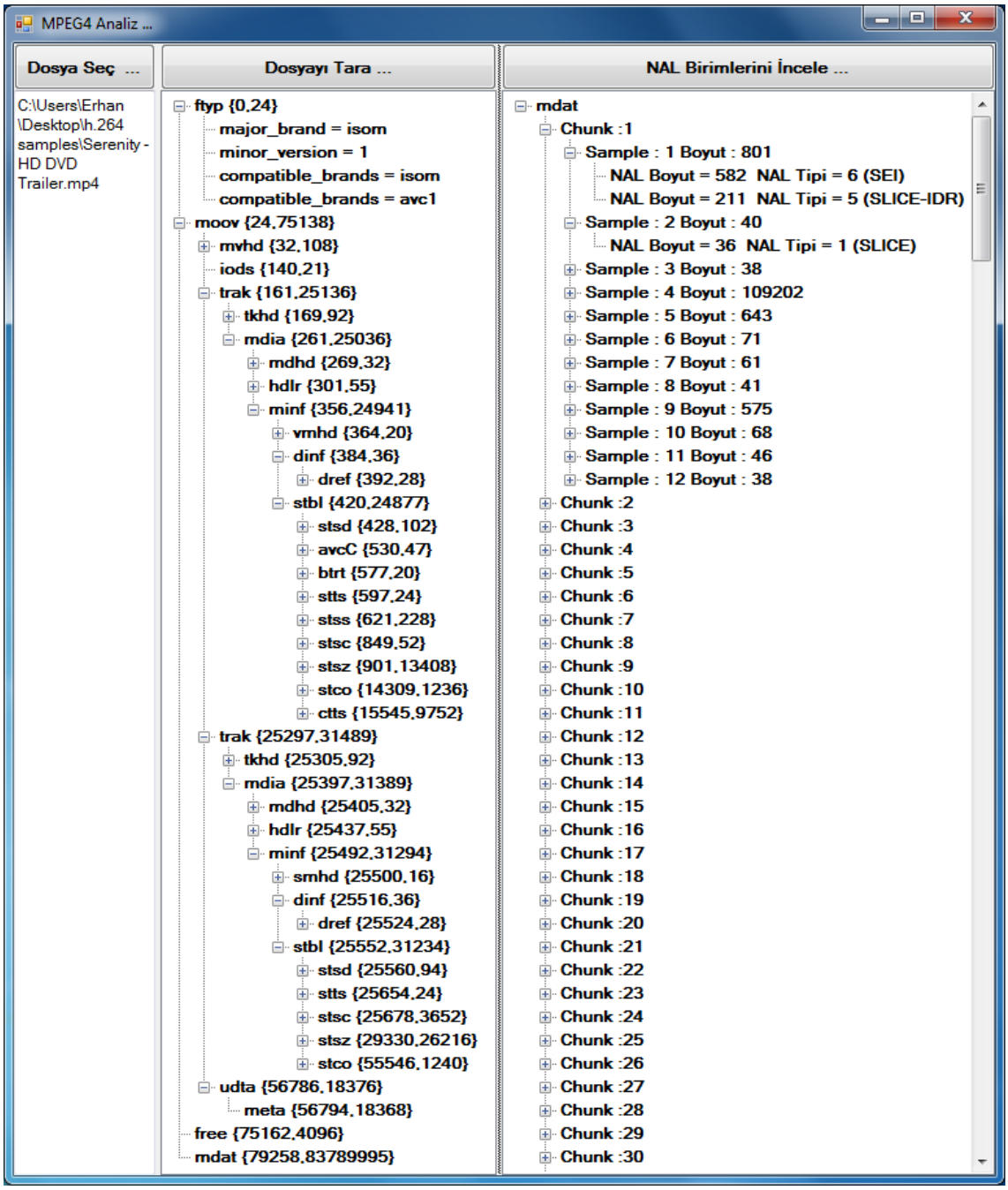
5.1. ISO Temelli Çoklu Ortam Dosya Yapısının Çözülmesi

ISO temelli dosya yapısının çözülmesi ve dosya içerisinde nesnelerin bulunması için Tablo 4.2'de verilen yapının tamamen elde edilmesi gerekmektedir. Detayları bölüm 4.2.1'de anlatılan bu yapıyı elde etmek için geliştirilen uygulama Şekil 5.1'de gösterilmektedir.

Uygulama iki ana kısımdan oluşmaktadır. İlk kısım çoklu ortam dosyasını hiyerarşik olarak tarayarak, Tablo 4.2'de listelenen bileşenlerden hangilerine sahip olduğunu göstermektedir. Bu bileşenler aynı zamanda yapılarında bazı bilgiler ya da parametreler de barındırmaktadırlar. Uygulama bunları da olarak okuyarak ilgili bileşenin altında listelemektedir. İkinci kısım ise bileşenlerin verileri kullanılarak ortam verisi (mdat) adı verilen ve video verilerinin sıkıştırılmış olarak saklandığı bölümü taramaktadır. Burada yığınlar halinde gruplanmış NAL birimleri bulunmaktadır.

Uygulama NAL birimlerinin her birinin konumunu, boyutunu ve türünü tarama esnasında tespit etmektedir.

Bu uygulama dosya yapısı içerisinden AVC kod çözümü için gerekli bileşenleri elde ederek, kod çözümüne uygun alt yapıyı sağlamaktadır.



Şekil 5.1. ISO temelli çoklu ortam dosya yapısını çözen uygulama

5.2. AVC Kod Çözümü

Dosya yapısından elde edilen NAL birimleri kodu çözülecek ham video verilerini ve parametrelerini taşımaktadırlar. Bölüm 4.3.2'de anlatılan kodlama yöntemleri ile bu birimler içerisindeki verilere anlam kazandırılmaktadır.

Diğer birimlerin çözümünde kullanılacakları için ilk önce SPS ve PPS NAL birimleri elde edilmektedir ve geçerlilik sürelerince kullanılmaktadır. Kodlanmış bir dosya içerisinde birden fazla SPS ve PPS NAL birimi olabilmekte ve bunlar dizi şeklinde saklanmaktadırlar.

İçinde resim barındıran NAL birimleri ise kodlama tipine (CABAC ya da CAVLC) göre farklı şekillerde çözülmektedirler. Kod çözümü sonrasında resmi oluşturacak katsayılar elde edilmiş olmaktadır. Bölüm 4.2.4'de anlatılan ters niceleme ve dönüşüm işlemlerinin ardından YCbCr verileri elde edilerek resim oluşturulur ve görüntülenmeye hazır hale gelmiştir [41].

5.3. Video Çerçevelerinin Oluşturulması ve Görüntülenmesi

Resim çerçevesinin YUV (YCbCr) formatındaki makro blok matrisi elde edildikten sonra, çoğu zaman farklı uygulamalarda kullanılması ya da resmin görüntülenebilmesi için renk uzayının RGB'ye dönüştürülmesi gerekmektedir.

Uygulamada resimlerin görüntülenmesi için Bölüm 4.2.5'de anlatılan dönüştürme yapılmaktadır. Bu sayede elde edilen resim çerçevelerinde nesne elde algoritmaları da uygulanabilmektedir. Çerçeveleri elde eden ve gösteren uygulama Şekil 5.3'de, YUV resim formatının RGB'ye çevrilmesinde kullanılan fonksiyon ise Şekil 5.2'de gösterilmektedir.

```
public C_YUV_Viewer(string FileName, System.Windows.Forms.PictureBox PBox,
                    int YUV_Format, int PixX, int PixY, int PicNumber)
{
      int Y, U, V;
      int R, G, B;
      int SubWidthC, SubHeightC;
```

```
switch (YUV_Format)
{
    case 420: SubWidthC = 2; SubHeightC = 2; break;
    case 422: SubWidthC = 2; SubHeightC = 1; break;
    case 444: SubWidthC = 1; SubHeightC = 1; break;
    default: SubWidthC = 0; SubHeightC = 0; break;
}

FileStream Dosya = new FileStream(FileName,FileMode.Open,  FileAccess.Read);

if (PicNumber <= 0)
{
    Dosya.Seek(0, SeekOrigin.Begin);
}
else
{
    Dosya.Seek((PixX * PixY + (PixX * PixY / (SubWidthC * SubHeightC /2)))
                * PicNumber , SeekOrigin.Begin);
}

byte[] buffer = new byte[PixX * PixY + (PixX * PixY / (SubWidthC *
                        SubHeightC / 2))];

while (Dosya.Position < Dosya.Length)
{
    Dosya.Read(buffer, 0, buffer.Length);
    Bitmap bmap = new Bitmap(PixX, PixY);

    for (int y = 0; y < PixY; y++)
        for (int x = 0; x < PixX; x++)
        {
           Y = (int)buffer[PixX * y + x];
           U = (int)buffer[PixX * PixY + (PixX / SubWidthC * (y /
                        SubHeightC) + (x / SubWidthC))];
           V = (int)buffer[(int)(PixX * PixY) + (PixX * PixY /
                        (SubWidthC * SubHeightC)) + (PixX / SubWidthC * (y /
                        SubHeightC) + (x / SubWidthC))];

           B = (int)(1.164 * (Y - 16) + 2.018 * (U - 128));
           G = (int)(1.164 * (Y - 16) - 0.813 * (V - 128) - 0.391 * (U - 128));
           R = (int)(1.164 * (Y - 16) + 1.596 * (V - 128));
```

```
                if (R > 255) R = 255;
                if (G > 255) G = 255;
                if (B > 255) B = 255;

                if (R < 0) R = 0;
                if (G < 0) G = 0;
                if (B < 0) B = 0;

                bmap.SetPixel(x, y, Color.FromArgb(R, G, B));
            }

        PBox.Image = bmap;
        PBox.Refresh();
    }
    Dosya.Close();
}
```

Şekil 5.2. YUV – RGB renk dönüşümü

Şekil 5.3. Video çerçevelerini gösteren uygulama

BÖLÜM 6

6. UYGULAMADA KULLANILAN SAYISAL VİDEO GÖRÜNTÜLERİNDEN NESNE ELDE ETME YÖNTEMLERİ

Video görüntülerinin işlenmesindeki temel konulardan bir tanesi de, video akımında ya da dosyasında ön plandaki nesnelerin tespit edilmesidir. Nesneleri çıkarmak için yaygın olarak kullanılan yaklaşım arka plan çıkarma ve türevleridir. Bu teknikler gerçek zamanlı video işlemede geniş ölçüde kullanılmaktadırlar. Ancak arka plandaki gölgeler, hareket eden nesneler ve ışık değişimleri işlemleri daha zor hale getirmektedir [37, 38].

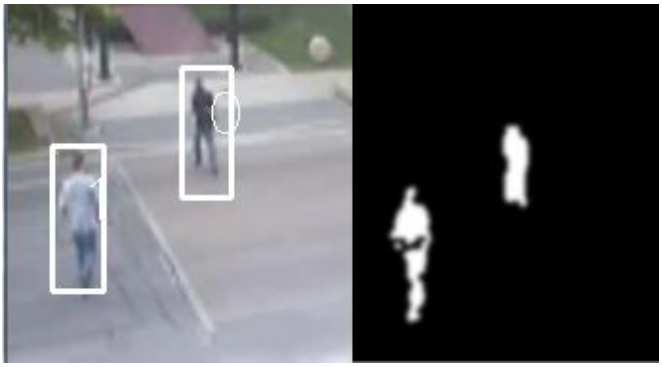
Sayısal video görüntüleri üzerinden nesne elde edilmesi için birçok farklı yaklaşım bulunmaktadır. Bu yaklaşımların birçoğu, içerisinde kullanılan algoritmaların farklılık gösterdiği benzer yöntemleri kullanmaktadır [39, 40].

Uygulamada kullanılan yaklaşımlarda öncelikle arka plan ve ya ön plan çıkarımı yapılmaktadır. Bu sayede görüntü üzerindeki kullanılmayacak alanlar filtrelenerek geri kalan alanlar üzerinde nesne elde etme algoritmaları işletilebilmektedir.

6.1. Hareketli Nesnelerin Takibi

Hareketli nesnelerin takibinde, göreceli durgun bir arka plana karşı, ön plandaki hareket eden nesneler izlenmektedir. Uygulamada video görüntüleri içerisinde nesne elde etme ve bu nesnelerin takibi üç temel adımda gerçekleştirilmektedir:

- Bir ön plan / arka plan ayırıcısı görüntünün her bir pikselini arka plana ya da ön plana ait olacak şekilde etiketlemektedir.
- Bir nesne tespit edici ön plana ait pikselleri gruplayarak belli parametrelere göre birleştirmektedir.
- Bir nesne takipçisi gruplanmış ve çeşitli filtreler uygulanmış bu piksel topluluklarına kimlik numaraları atayarak sonraki görüntülerde hareketlerini takip etmektedir.



Şekil 6.1. Arka plan çıkarma ve nesne takibi

Bir çok metot gerçek zamanlı nesne tespiti imkanı sağlamaktadır. Ancak, bunların birçoğu, arka planın, renk veya yoğunluğu zaman içinde yavaş değişen durağan nesnelerden oluştuğunu varsaymaktadır. Arka plan renklerini düzeltmek için en basit yol bir Sonsuz Etki Tepki (IIR) ya da Kalman filtresi uygulamaktır. Arka plan değişimlerini görmezden gelmek için bir Gauss fonksiyonu çalıştırılabilir [42].

En başarılı nesne tespit yöntemlerinden biri de Gaussların Karışımıdır (MoG). Bu yöntemde arka plandaki renkler çoklu Gauss dağılımları ile tanımlanmaktadır. İyi sonuç vermesine rağmen nesne tespitinde performansı düşürmektedir.

Karmaşık videolardan nesne çıkarmak için önerilen bir başka başarılı yöntem de Bayes karar çatısı kullanan yöntemdir. Bir Bayes karar verme kuralı, formüle edilmiş genel bir özellik vektöründen, arka planı ve ön planı sınıflamaktadır.

Uygulama yazılımında anlatılan son iki yöntem de kullanılabilmektedir [43,44].

6.2. Renk Takibi

Renk Takibi, iki renk eşik değeri dışında kalan değerlerin filtrelenip, ardından kenar detektörü [45] ve dönüşüm [46] uygulanarak, nesnelerin elde edilmesi ve izlenmesi yöntemidir.

Şekil 6.2. Renk filtreleme ve nesne elde etme

6.3. SURF Algoritması

Uygulamada elde edilmiş nesnelerin tekrar bulunması için SURF algoritması [47] kullanılmaktadır. Bu yöntem nesnenin özelliklerini kullanarak belli bir resim içerisinde tespit edilmesini sağlamaktadır.

Şekil 6.3. SURF algoritması ile resim içerisinde nesnenin tespit edilmesi

SURF algoritması, yerel değişmeyen benzerliklerin gösterimi ve karşılaştırılması için hızlı ve güçlü bir algoritmadır. Bu algoritma ardışık üç adımdan oluşmaktadır.

- İlgi noktası tespiti
- Her bir ilgi noktası ile alakalı tanımlayıcı oluşturmak
- Tanımlayıcı eşleştirme

SURF algoritması, hızlı sayılmasına rağmen gerçek zamanlı video görüntülerinde kullanılmak istendiğinde yavaş kalabilmektedir.

6.4. Elde Edilen Nesnelerin Saklanması ve Aranması

Nesne adayları tespit edildikten sonra bir dizide depolanmaktadır. Kullanıcı, nesnelerin tanımlarını yaparak etiketlemesini sağlamaktadır. Bu bilgiler bir dosyada ya da veri tabanında saklanmaktadır. Etiketleme yapılmayan nesneler göz ardı edilebilmektedir.

Yapılan etiketlemeler içerisinde arama yapılabilmektedir. İlgili kimlik numarasına ait nesne, SURF algoritması aracılığı ile görüntü üstünde tekrar seçilebilmektedir.

BÖLÜM 7

7. SONUÇ

Bu tez çalışmasında, MPEG-4 AVC ISO temelli çoklu ortam dosya yapısı çözüldükten sonra NAL birimleri tespit edilerek, parametre bilgileri ve video ham verileri elde edilmektedir. Elde edilen parametreler doğrultusunda görüntü dilimlerinin içerisinde yer alan makro blokların çözümü ve video görüntülerinin elde edilmesi işlemi tamamlanmaktadır.

Görüntüden nesneleri elde etmek için ön plan çıkarma, nesne takibi ve renk takibi yöntemleri kullanılmaktadır. Nesne elde etme yöntemleri ile elde edilen verilerin etiketlenmesi işlemi, uygulama ara yüzünden gerçekleştirilebilmekte ve sağlanan ikinci bir ara yüz ile bu etiketler üzerinde arama yapılabilmektedir.

Uygulama içerisinde kullanılan nesne elde etme yöntemlerinde karşılaşılan en büyük zorluk, yüksek yoğunluklu hesaplama gerektirmelerinden dolayı yavaş çalışmalarıdır. Bu yüzden bazı algoritmalar gerçek zamanlı çalıştırılamamaktadır. Bu yöntemlerin yazılımsal olarak hızlandırılması ya da gelişen donanım teknolojilerine aktarılması ile daha verimli hale getirilebileceği düşünülmektedir.

Bu tez konusunun devamı olarak; nesne elde etme yöntemlerinin başarımının arttırılması, paralel programlama teknikleri ile var olan yöntemlerin gerçek zamanlı videolarda daha verimli olarak çalıştırılmalarının sağlanması ve MPEG-4 AVC kodlanmış video verilerinin kısmen kod çözümü yapılarak nesne elde etme yöntemleri ile entegre edilmesi konuları üzerine çalışmalar yapılabilir.

KAYNAKLAR

[1] A. Müfit Ferman, Bilge Günsel, A. Murat Tekalp, ***Object-Based Indexing of MPEG-4 Compressed Video,*** SPIE, Cilt 3024, Sayfa 953-963, 1997

[2] Thomas Meier, King N. Ngan, ***Automatic Segmentation of Moving Objects for Video Object Plane Generation***, IEEE Trans. Circuits and Systems for Video Technology, Cilt 8, Sayı 5, Sayfa 525-538, 1998

[3] Paulo Nunes, Paulo Correia, Fernando Pereira, ***Coding Video Objects with the Emerging MPEG-4 Standard***, Proc Conf. on Telecommunications - ConfTele, Cilt 1, Sayfa 425-428, 1997

[4] E.P. Ong, B.J. Tye, W.S. Lin, M. Etoh, ***A Fast Video Object Segmentation Scheme for MPEG-4 Video Coding***, Third International Conference on Information and Communications Security, 2001

[5] Di Zhong, Shih-Fu Chang, ***AMOS: An Active System For MPEG-4 Video Object Segmentation***, ICIP'98, Cilt 1, Sayfa 647, 1998

[6] Hua Zhong, Liu Wenyin, Shipeng Li, ***Interactive Tracker – A Semi-Automatic Video Object Tracking And Segmentation System***, IEEE International Conference on Multimedia and Expo (ICME'01), Sayfa 1167-1170, 2001

[7] Ferdous Ahmed Sohel, Chowdhury Mofizur Rahman, Gour C. Karmakar, ***Automatic Video Object Segmentation from VOP***, ICEECE, 2003

[8] Oğuzhan Urhan, ***Video İçin Nesne Bölütlemesi***, 2003

[9] Fatih Porikli, ***Real-Time Video Object Segmentation for MPEG Encoded Video Sequences***, SPIE Conference on Real-Time Imagining VIII, Cilt 5297, Sayfa 195-203, 2004

[10] Dirk Sven Farin, ***Automatic Video Segmentation Employing Object/Camera Modeling Techniques***, 2005

[11] Xiao-Ping Zhang, Zhenhe Chen, ***An Automated Video Object Extraction System Based on Spatiotemporal Independent Component Analysis and Multiscale Segmentation***, EURASIP Journal on Applied Signal Processing, Cilt 2006, Sayfa 184, 2006

[12] Elisa Drelie Gelasca, Touradj Ebrahimi, ***On Evaluating Video Object Segmentation Quality: A Perceptually Driven Objective Metric***, IEEE Journal Of Selected Topics In Signal Processing, Cilt 3, Sayı 2, Sayfa 319-335, 2009

[13] Tsong-Yi Chen, Thou-Ho (Chao-Ho) Chen, Da-Jinn Wang, Yung-Chuen Chiou, ***Real-Time Video Object Segmentation Algorithm Based On Change Detection And Background Updating***, International Journal of Innovative Computing, Information and Control, Cilt 5, Sayı 7, Sayfa 1979-1810, 2009

[14] William Brendel, Sinisa Todorovic, ***Video Object Segmentation by Tracking Regions***, ICCV2009, ISSN: 1550-5499, Sayfa 833-840, 2009

[15] Raymond Fielding, ***A Technological History of Motion Pictures and Television***, University of California Press, 1983

[16] M. Hilmes, J. Jacobs, ***The Television History Book***, BFI, 2008

[17] R. Bruin, ***Digital Video Broadcasting: Technology, Standards, and Regulations***, Artech House Publishers, 1999

[18] Oge Marques, ***Practical Image and Video Processing Using MATLAB***, Wiley-IEEE Press, 2011

[19] Ji-Won Lee, Min-Jeong Lee, Hae-Yeoun Lee, Heung-Kyu Lee, ***Screenshot identification by analysis of directional inequality of interlaced video***, EURASIP Journal on Image and Video Processing, 2012

[20] C. A. Poynton, ***Digital Video and HDTV: Algorithms and Interfaces***, Morgan Kaufmann, 2002

[21] ISO/IEC 11172, ***Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s***, 1993

[22] IEC 62107, ***Super Video Compact Disc – Disc-interchange system-specification,*** 2000

[23] IEC 61834-2, ***Recording – Helical-scan digital video cassette recording system using 6,35 mm magnetic tape for consumer use***, 1998

[24] N. Ahmed, T. Natarajan, ***Discrete Cosine Transform***, IEEE Transactions on Computers, Cilt C-3, Sayı 1, Sayfa 90-93, 1974

[25] ISO/IEC 11172-2, ***Information technology -- Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s -- Part 2: Video***, 1993

[26] ISO/IEC 13818-2, ***Information technology -- Generic coding of moving pictures and associated audio information: Video***, 2000

[27] ISO/IEC 14496-2, ***Information technology -- Coding of audio-visual objects -- Part 2: Visual***, 2004

[28] ISO/IEC 15938-3, ***Information technology -- Multimedia content description interface -- Part 3: Visual***, 2002

[29] ISO/IEC 14496-12, ***Information technology -- Coding of audio-visual objects -- Part 12: ISO base media file format***, 2012

[30] ISO/IEC 14496-15, ***Information technology -- Coding of audio-visual objects -- Part 15: Advanced Video Coding (AVC) file format***, 2010

[31] ISO/IEC 14496-10, ***Information technology -- Coding of audio-visual objects -- Part 10: Advanced Video Coding***, 2012

[32] I. E. G. Richardson, ***H.264 and MPEG-4 Video Compression Video Coding for Next-generation Multimedia***, John Wiley & Sons Ltd, 2003

[33] I. E. G. Richardson, ***H.264 / MPEG-4 Part 10: Variable Length Coding***, 2002

[34] I. E. G. Richardson, ***H.264 / MPEG-4 Part 10: Introduction to CABAC***, 2002

[35] Eric Bodden, Malte Clasen, Joachim Kneis, ***Arithmetic Coding revealed***, 2007

[36] ISO/IEC 15444-3, ***Information technology -- JPEG 2000 image coding system -- Part 3: Motion JPEG 2000,*** 2002

[37] Wei Zenga, Jun Dub, Wen Gaoc, Qingming Huangb, ***Robust moving object segmentation on H.264/AVC compressed video using the block-based MRF model,*** Real-Time Imagining 11, Cilt 11, Sayı 4, Sayfa 290-299, 2005

[38] Jae-Kyun Ahn, Dae-Yeon Lee, Chul Lee, Chang-Su Kim, ***Automatic Moving Object Segmentation from Video Sequences Using Alternate Flashing System***, EURASIP Journal on Advances in Signal Processing 2010, Cilt 2010, Sayı 6, 2010

[39] Yuchi Huang, Qingshan Liu, Dimitris Metaxas, ***Video Object Segmentation by Hypergraph Cut,*** CVPR2009, ISSN: 1063-6919, Sayfa 1738-1745, 2009

[40] C. Rother, V. Kolmogorov, A. Blake, ***GrabCut: Interactive Foreground Extraction using Iterated Graph Cuts***, ACM SIGGRAPH 2004, Cilt 23, Sayı 3, Sayfa 309-314, 2004

[41] JM Software, H***.264 / AVC Software Coordination***, http://iphome.hhi.de/ suehring/tml/, Erişim: 05.03.2012

[42] Liyuan Li, Weimin Huang, Irene Y.H. Gu, and Qi Tian, ***Foreground Object Detection from Videos Containing Complex Background***, MULTIMEDIA '03 Proceedings of the eleventh ACM international conference on Multimedia, Sayfa 2-10, 2003

[43] OpenCV (Open Source Computer Vision), ***A Library of Programming Functions for Realtime Computer Vision***, http://code.opencv.org/projects/opencv/wiki, Erişim: 01.11.2012

[44] Emgu CV, ***A cross platform .Net wrapper to the OpenCV image processing library***, http://www.emgu.com/wiki/index.php/Main_Page, Erişim: 01.12.2012

[45] J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, Sayfa 679–698, 1986

[46] R. O Duda, P. E. Hart, ***Use of the Hough Transformation to Detect Lines and Curves in Pictures***, Comm. ACM, Cilt 15, Sayfa 11–15, 1972

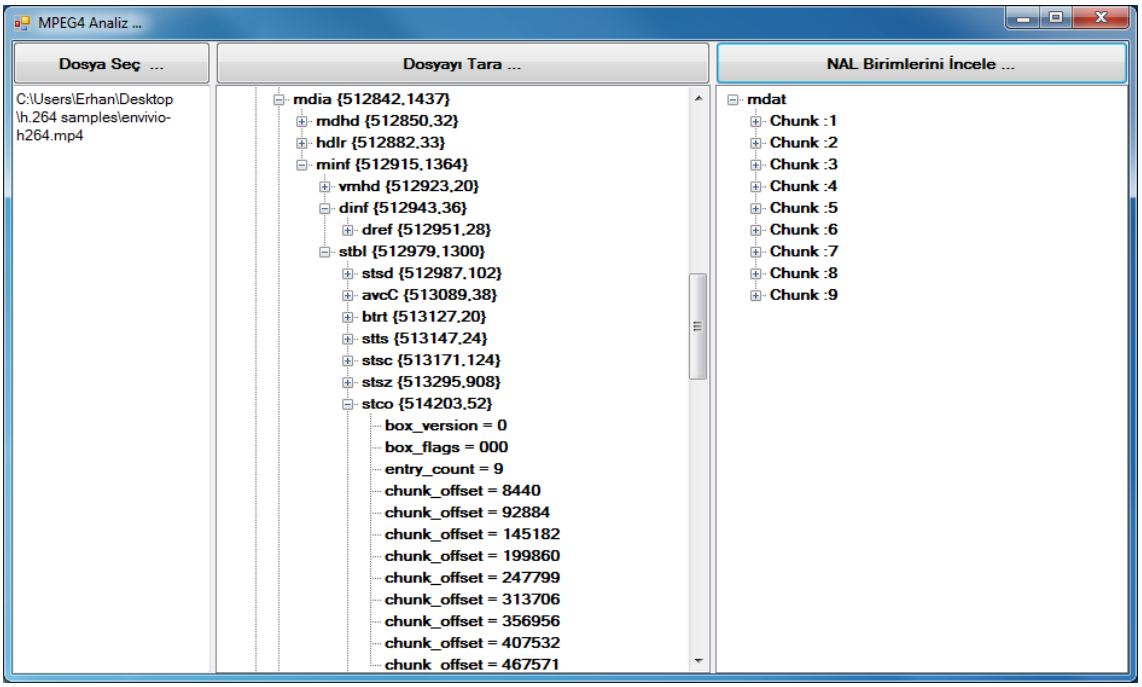
[47] Herbert Bay, Tinne Tuytelaars, Luc Van Gool, ***SURF: Speeded Up Robust Features***, Computer Vision and Image Understanding, Cilt 110, Sayı 3, Sayfa 346-359, 2008

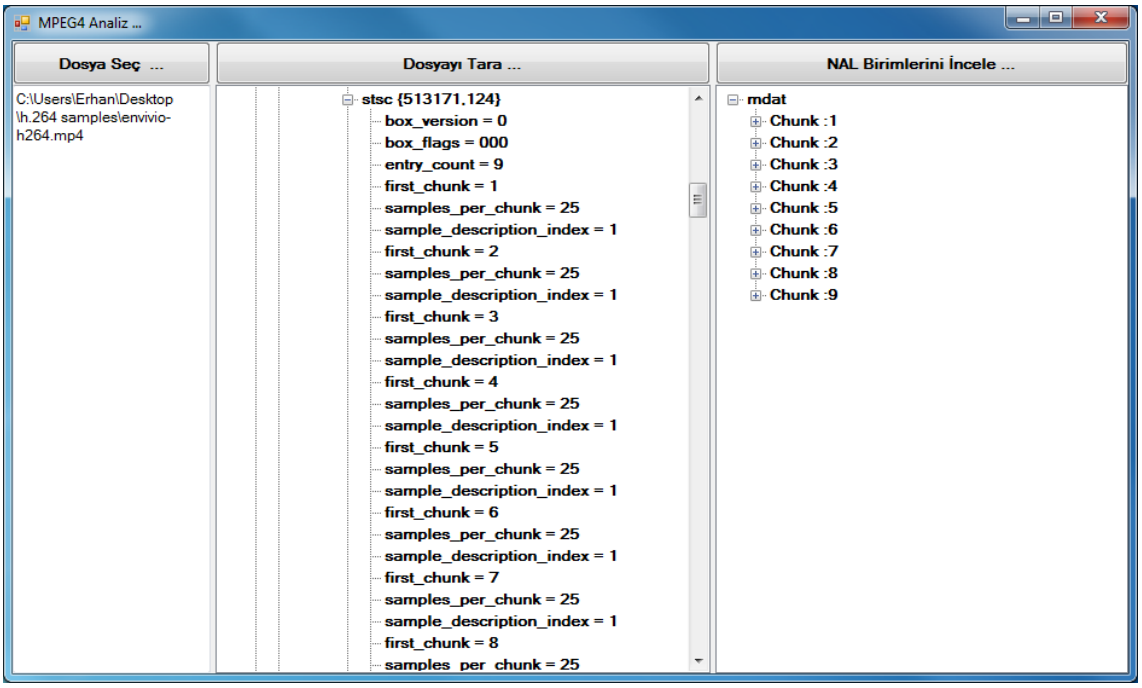
EKLER

Ek-A.Video Standartlarının Detaylı Karşılaştırılması

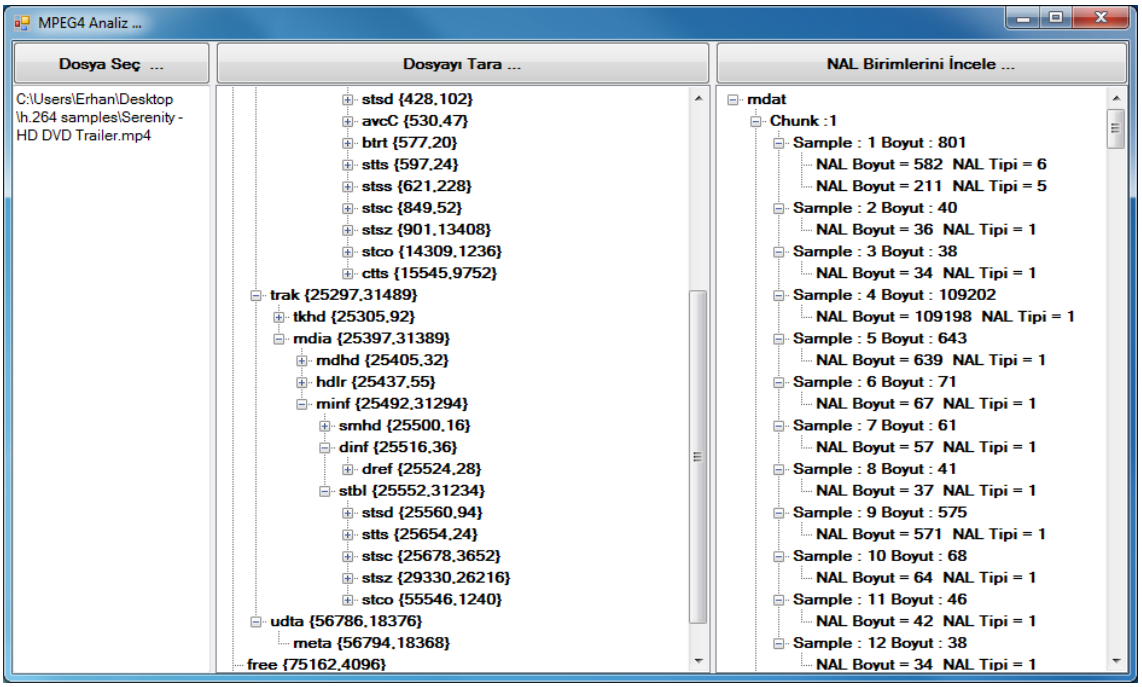
Format	VCD	SVCD	AVI DV	DVD	HDDVD HDTV (WMVHD)	AVI DivX XviD WMV	MOV Quick-Time	RM Real-Media
Çözünürlük NTSC/PAL	352x240 352x288	480x480 480x576	720x480 720x576	720x480 720x576	1920x1080 1280x720	640x480	640x480	320x240
Video Sıkıştırma	MPEG1	MPEG2	DV	MPEG2, MPEG1	MPEG2 (WMV-MPEG4)	MPEG4	Sorenson, Cinepak, MPEG4 ...	RM
Video Bit Hızı	1150kbps	~2000kbps	25Mbps	~5000kbps	~20Mbps (~8Mbps)	~1000kbps	~1000kbps	~350kbps
Ses Sıkıştırması	MP1	MP1	DV	MP1, MP2, AC3, DTS, PCM	MP1, MP2, AC3, DTS, PCM	MP3, WMA, OGG, AAC, AC3	QDesign Music, MP3 ...	RM
Ses Bit Hızı	224kbps	~224kbps	~1500kbps	~448kbps	~448kbps	~128kbps	~128kbps	~64kbps
Boyut/ Dakika	10 MB/dk	10-20 MB/dk	216MB/dk	30-70 MB/dk	~150MB/dk (~60MB/dk)	4-10 MB/dk	4-20 MB/dk	2-5 MB/dk
DVD Oynatıcı Uyumluluğu	Çok İyi	İyi	Yok	Mükemmel	Yok	Az	Yok	Yok
Bilgisayar İşlemci Kullanımı	Düşük	Yüksek	Yüksel	Çok Yüksek	Aşırı Yüksek	Çok Yüksek	Yüksek	Düşük
Kalite	İyi	Çok İyi	Mükemmel	Mükemmel	Muhteşem	Çok İyi	Çok İyi	Orta

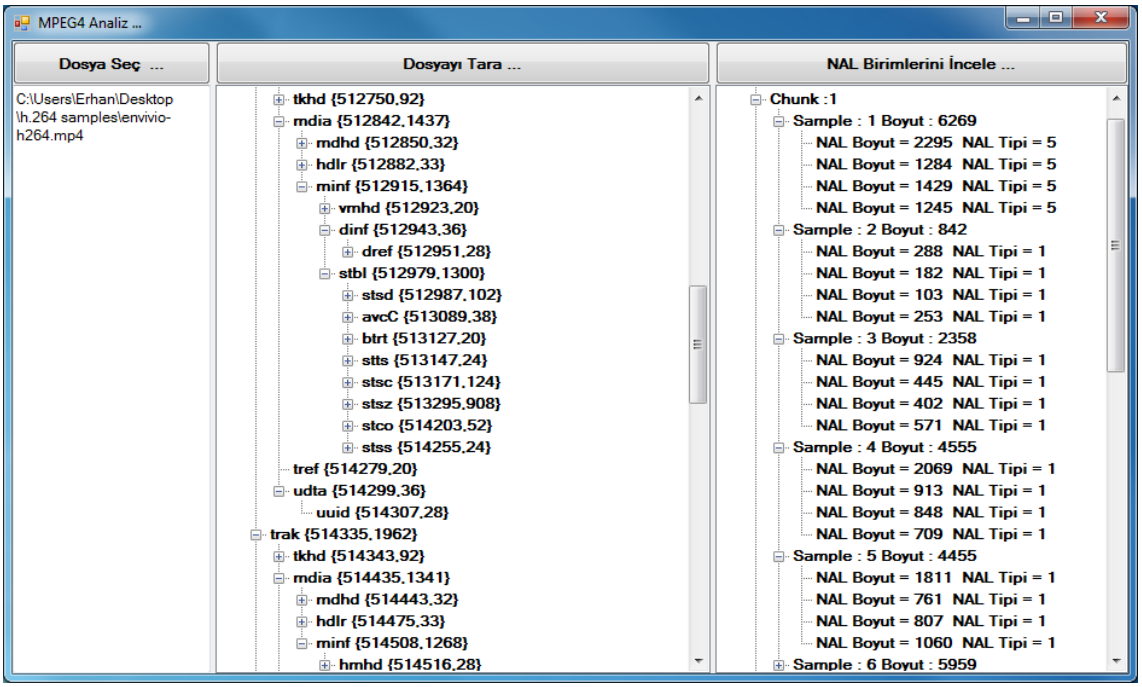
Ek-B. ISO Temelli Dosya Yapısının Uygulama Programında Gösterimi



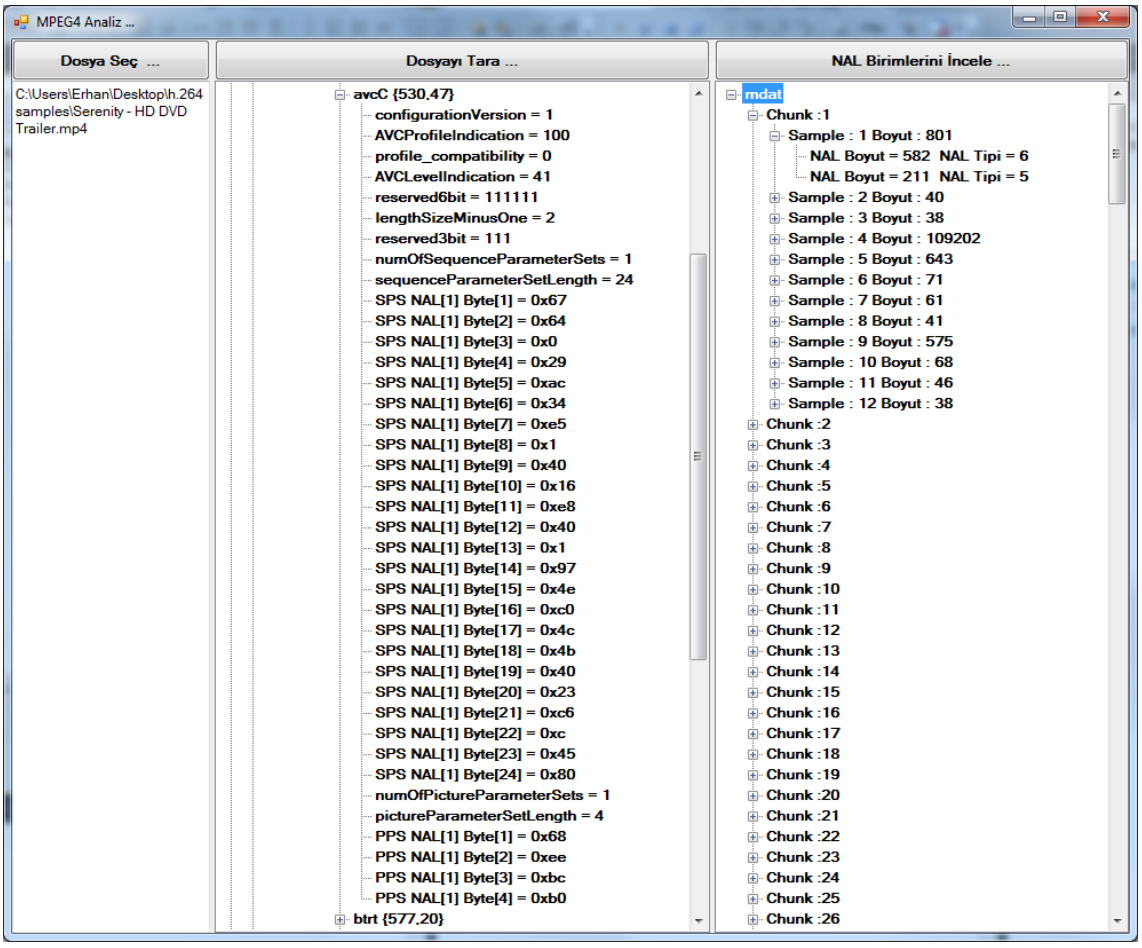


Ek-C. Farklı Tipteki NAL Birimlerinin, Farklı Şekillerde Sıralanması





Ek-D. SPS ve PPS NAL Birimlerini AVC Dosya Yapısında Tutan “**avcC**” Nesnesi



Ek-E. CAVLC Tabloları

"0" ların Sayısı	Katsayı Adedi						
	1	2	3	4	5	6	7
0	1	111	0101	00011	0101	000001	000001
1	11	110	111	111	0100	00001	00001
2	10	101	110	0101	0011	111	101
3	11	100	101	0100	111	110	100
4	10	011	0100	110	110	101	011
...	...	...	...	...	...	...	...

"1" lerin Sayısı	Katsayı Adedi	Bit Dizisi
0	0	1
0	1	000101
1	1	01
0	2	00000111
1	2	000100
2	2	001011
0	3	000000111
1	3	00000110
2	3	0000101
3	3	00011
0	4	0000000111
1	4	000000110
2	4	00000101
3	4	000011
0	5	00000000111
1	5	0000000110
2	5	000000101
3	5	0000100
...	...	...

Seviye Öneki	Bit Dizisi
0	1
1	01
2	001
3	0001
4	0000 1
5	0000 01
6	0000 001
7	0000 0001
...	...

Ek-F. CABAC Sözdizimi elemanları için başlatılması gereken model aralıkları tablosu.

	Sözdizimi Elemanı	Resim Dilimi Türü			
		SI	I	P,SP	B
Resim Dilimi	mb_skip_flag			11-13	24-26
	mb_field_decoding_flag	70-72	70-72	70-72	70-72
Makro Blok Katmanı	mb_type	0-10	3-10	14-20	27-35
	transform_size_8x8_flag		399-401	399-401	399-401
	coded_block_pattern (Y)	73-76	73-76	73-76	73-76
	coded_block_pattern (CbCr)	77-84	77-84	77-84	77-84
	mb_qp_delta	60-63	60-63	60-63	60-63
Makro Blok Tahmini	prev_intra4x4_pred_mode_flag	68	68	68	68
	rem_intra4x4_pred_mode	69	69	69	69
	prev_intra8x8_pred_mode_flag		68	68	68
	rem_intra8x8_pred_mode		69	69	69
	intra_chroma_pred_mode	64-67	64-67	64-67	64-67
Makro Blok ve Alt Makro Blok Tahmini	ref_idx_l0			54-59	54-59
	ref_idx_l1				54-59
	mvd_l0[ ][ ][0]			40-46	40-46
	mvd_l1[ ][ ][0]				40-46
	mvd_l0[ ][ ][1]			47-53	47-53
	mvd_l1[ ][ ][1]				47-53
Alt Makro Blok Tahmini	sub_mb_type			21-23	36-39
CABAC Resim Farkı Bloğu	coded_block_flag	85-104 460-483	85-104 460-483 1012-1023	85-104 460-483 1012-1023	85-104 460-483 1012-1023
	significant_coeff_flag[ ]	105-165 277-337	105-165 277-337 402-416 436-450 484-571 776-863 660-689 718-747	105-165 277-337 402-416 436-450 484-571 776-863 660-689 718-747	105-165 277-337 402-416 436-450 484-571 776-863 660-689 718-747
	last_significant_coeff_flag[ ]	166-226 338-398	166-226 338-398 417-425 451-459 572-659 864-951 690-707 748-765	166-226 338-398 417-425 451-459 572-659 864-951 690-707 748-765	166-226 338-398 417-425 451-459 572-659 864-951 690-707 748-765
	coeff_abs_level_minus1[ ]	227-275	227-275 426-435 952-1011 708-717 766-775	227-275 426-435 952-1011 708-717 766-775	227-275 426-435 952-1011 708-717 766-775

Ek-G. CABAC Bağlam Model Tablosu.

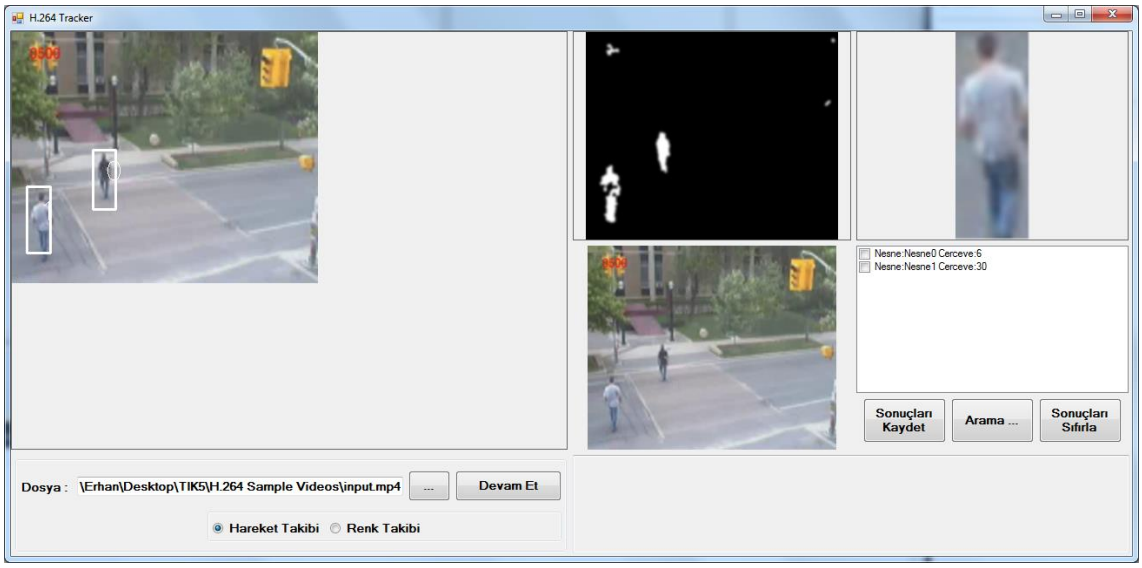
Sözdizimi Elemanı	Kodlama Türü	maxBinIdxCtx	ctxIdxOffset
mb_type (Sadece SI dilimi için)	Önek ve Sonek	Önek: 0 Sonek: 6	Önek: 0 Sonek: 3
mb_type (Sadece I dilimi için)		6	3
mb_skip_flag (Sadece P, SP dilimleri için)	Sabit Uzunluklu, cMax=1	0	11
mb_type (Sadece P, SP dilimleri için)	Önek ve Sonek	Önek: 2 Sonek: 5	Önek: 14 Sonek: 17
sub_mb_type (Sadece P, SP dilimleri için)		2	21
mb_skip_flag (Sadece B dilimi için)	Sabit Uzunluklu, cMax=1	0	24
mb_type (Sadece B dilimi için)	Önek ve Sonek	Önek: 3 Sonek: 5	Önek: 27 Sonek: 32
sub_mb_type (Sadece B dilimi için)		3	36
mvd_l0[ ][ ][0], mvd_l1[ ][ ][0]	Önek ve Sonek UEG3 signedValFlag=1, uCoff=9	Önek: 4	Önek: 40 Sonek: DecodeBypass
mvd_l0[ ][ ][1], mvd_l1[ ][ ][1]		Önek: 4	Önek: 47 Sonek: DecodeBypass
ref_idx_l0, ref_idx_l1	Tek Bileşenli	2	54
mb_qp_delta		2	60
intra_chroma_pred_mode	Kesilmiş Tek Bileşenli, cMax=3	1	64
prev_intra4x4_pred_mode_flag, prev_intra8x8_pred_mode_flag	Sabit Uzunluklu, cMax=1	0	68
rem_intra4x4_pred_mode, rem_intra8x8_pred_mode	Sabit Uzunluklu, cMax=7	0	69
mb_field_decoding_flag	Sabit Uzunluklu, cMax=1	0	70
coded_block_pattern	Önek ve Sonek	Önek: 3 Sonek: 1	Önek: 73 Sonek: 77
coded_block_flag	Sabit Uzunluklu, cMax=1	0	85
significant_coeff_flag (blok kategorisi<5 olan çerçeve blokları)	Sabit Uzunluklu, cMax=1	0	105
last_significant_coeff_flag (blok kategorisi<5 olan çerçeve blokları)	Sabit Uzunluklu, cMax=1	0	166
coeff_abs_level_minus1 (blok kategorisi<5 olan bloklar)	Önek ve Sonek UEG0 signedValFlag=0, uCoff=14	Önek: 1	Önek: 227 Sonek: DecodeBypass
coeff_sign_flag	Sabit Uzunluklu, cMax=1	0	DecodeBypass
end_of_slice_flag	Sabit Uzunluklu, cMax=1	0	276

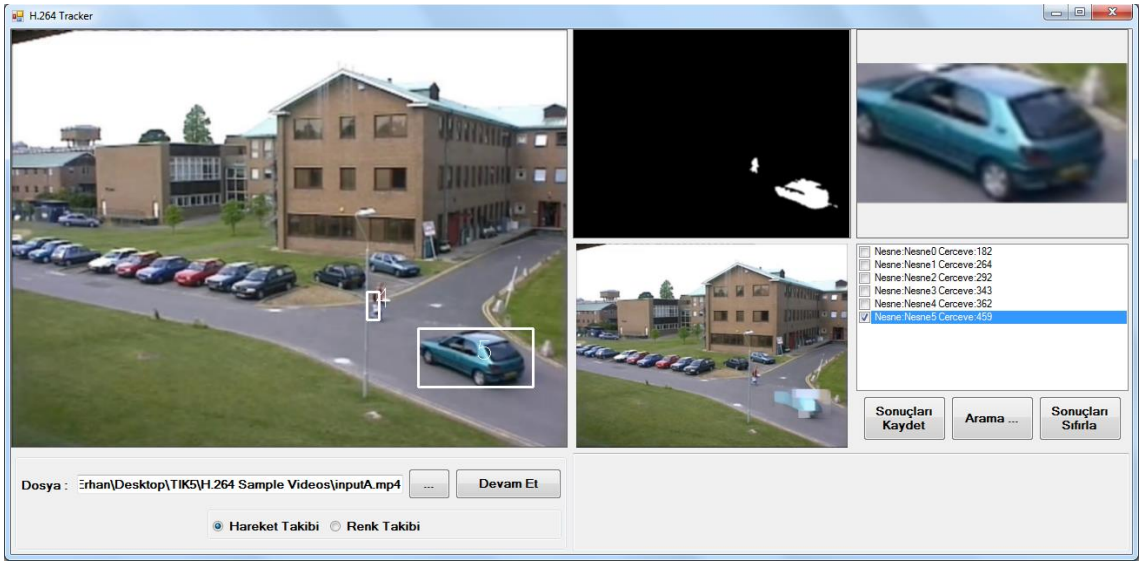
significant_coeff_flag (blok kategorisi<5 olan alan blokları)	Sabit Uzunluklu, cMax=1	0	277
last_significant_coeff_flag (blok kategorisi<5 olan alan blokları)	Sabit Uzunluklu, cMax=1	0	338
transform_size_8x8_flag	Sabit Uzunluklu, cMax=1	0	399
significant_coeff_flag (blok kategorisi=5 olan çerçeve blokları)	Sabit Uzunluklu, cMax=1	0	402
last_significant_coeff_flag (blok kategorisi=5 olan çerçeve blokları)	Sabit Uzunluklu, cMax=1	0	417
coeff_abs_level_minus1 (blok kategorisi=5 olan bloklar)	Önek ve Sonek UEG0 signedValFlag=0, uCoff=14	Önek: 1	Önek: 426 Sonek: DecodeBypass
significant_coeff_flag (blok kategorisi=5 olan alan blokları)	Sabit Uzunluklu, cMax=1	0	436
last_significant_coeff_flag (blok kategorisi=5 olan alan blokları)	Sabit Uzunluklu, cMax=1	0	451
coded_block_flag (5 < blok kategorisi < 9)	Sabit Uzunluklu, cMax=1	0	460
coded_block_flag (9 < blok kategorisi < 13)	Sabit Uzunluklu, cMax=1	0	472
coded_block_flag (blok kategorisi=5, 9, 13)	Sabit Uzunluklu, cMax=1	0	1012
significant_coeff_flag (5 < blok kategorisi < 9 olan çerçeve blokları)	Sabit Uzunluklu, cMax=1	0	484
significant_coeff_flag (9 < blok kategorisi < 13 olan çerçeve blokları)	Sabit Uzunluklu, cMax=1	0	528
last_significant_coeff_flag (5 < blok kategorisi < 9 olan çerçeve blokları)	Sabit Uzunluklu, cMax=1	0	572
last_significant_coeff_flag (9 < blok kategorisi < 13 olan çerçeve blokları)	Sabit Uzunluklu, cMax=1	0	616
coeff_abs_level_minus1 (5 < blok kategorisi < 9 olan bloklar)	Önek ve Sonek UEG0 signedValFlag=0, uCoff=14	Önek: 1	Önek: 952 Sonek: DecodeBypass
coeff_abs_level_minus1 (9 < blok kategorisi < 13 olan bloklar)	Önek ve Sonek UEG0 signedValFlag=0, uCoff=14	Önek: 1	Önek: 982 Sonek: DecodeBypass

significant_coeff_flag (5 < blok kategorisi < 9 olan alan blokları)	Sabit Uzunluklu, cMax=1	0	776
significant_coeff_flag (9 < blok kategorisi < 13 olan alan blokları)	Sabit Uzunluklu, cMax=1	0	820
last_significant_coeff_flag (5 < blok kategorisi < 9 olan alan blokları)	Sabit Uzunluklu, cMax=1	0	864
last_significant_coeff_flag (9 < blok kategorisi < 13 olan alan blokları)	Sabit Uzunluklu, cMax=1	0	908
significant_coeff_flag (blok kategorisi=9 olan çerçeve blokları)	Sabit Uzunluklu, cMax=1	0	660
significant_coeff_flag (blok kategorisi=13 olan çerçeve blokları)	Sabit Uzunluklu, cMax=1	0	718
last_significant_coeff_flag (blok kategorisi=9 olan çerçeve blokları)	Sabit Uzunluklu, cMax=1	0	690
last_significant_coeff_flag (blok kategorisi=13 olan çerçeve blokları)	Sabit Uzunluklu, cMax=1	0	748
coeff_abs_level_minus1 (blok kategorisi=9 olan bloklar)	Önek ve Sonek UEG0 signedValFlag=0, uCoff=14	Önek: 1	Önek: 708 Sonek: DecodeBypass
coeff_abs_level_minus1 (blok kategorisi=13 olan bloklar)	Önek ve Sonek UEG0 signedValFlag=0, uCoff=14	Önek: 1	Önek: 766 Sonek: DecodeBypass
significant_coeff_flag (blok kategorisi=9 olan alan blokları)	Sabit Uzunluklu, cMax=1	0	675
significant_coeff_flag (blok kategorisi=13 olan alan blokları)	Sabit Uzunluklu, cMax=1	0	733
last_significant_coeff_flag (blok kategorisi=9 olan alan blokları)	Sabit Uzunluklu, cMax=1	0	699
last_significant_coeff_flag (blok kategorisi=13 olan alan blokları)	Sabit Uzunluklu, cMax=1	0	757

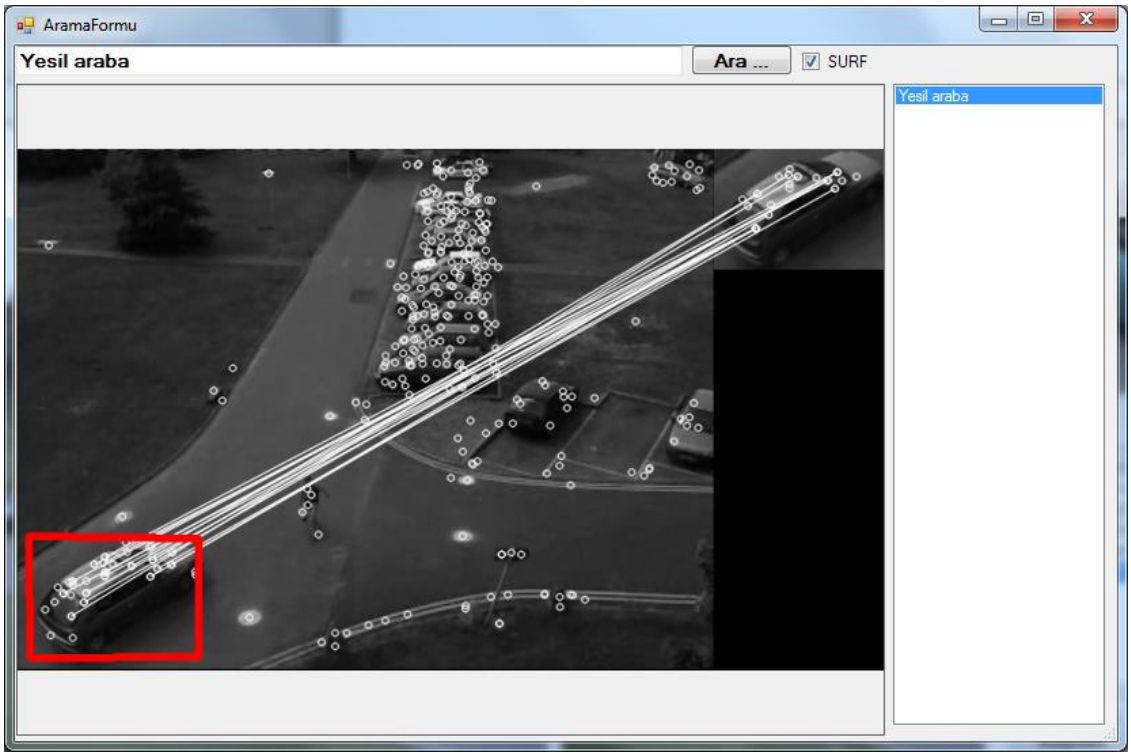
EK-H. Video kod çözücüsü çıktısı

Ek-I. Nesne bulma ve arama arayüzleri





Ek-J. Nesne bulma arayüzünde SURF algoritmasının kullanımı



ÖZGEÇMİŞ

1979 yılında Edirne'de doğdum. Çanakkale Onsekiz Mart Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü'nden 2002 yılında mezun olduktan sonra, 2005 yılında aynı üniversitenin Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda yüksek lisansımı tamamladım. 2009 yılından bu yana Trakya Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı'nda doktora öğrenimi görmekteyim. 2002-2007 yılları arasında Çanakkale Onsekiz Mart Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü'nde araştırma görevlisi ve öğretim görevlisi olarak çalıştım. 2007 yılından bu yana da özel bir telekomünikasyon şirketinde uzman bilgisayar mühendisi olarak çalışmaya devam etmekteyim.