İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

H.264/AVC İLE DÜŞÜK GECİKMELİ SIKIŞTIRMA

YÜKSEK LİSANS TEZİ

Altay Hüseyin ERŞAHİN

Anabilim Dalı : Elektronik ve Haberleşme Mühendisliği

Programı : Telekomünikasyon Mühendisliği

HAZİRAN 2009

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

H.264/AVC İLE DÜŞÜK GECİKMELİ SIKIŞTIRMA

YÜKSEK LİSANS TEZİ
Altay Hüseyin ERŞAHİN
(504061304)

Tezin Enstitüye Verildiği Tarih : 04 Mayıs 2009
Tezin Savunulduğu Tarih : 03 Haziran 2009

Tez Danışmanı : Prof. Dr. Melih PAZARCI (İTÜ)
Diğer Jüri Üyeleri : Prof. Dr. Bilge GÜNSEL (İTÜ)
Doç. Dr. Uluğ BAYAZIT (İTÜ)

HAZİRAN 2009

ÖNSÖZ

Tez çalışmam için beraber çalışmaya başladığımızdan bu yana bana yardımcı olan ve her konuda yapıcı yaklaşımını esirgemeyen tez danışmanım Prof. Dr. Melih PAZARCI'ya ve çalışmalarım süresince gösterdikleri anlayış ve destek için sevgili aileme teşekkür ederim. Yüksek lisans eğitimim süresince verdiği destekten dolayı TÜBİTAK – Bilim İnsanı Destekleme Daire Başkanlığı'na (BİDEB) da teşekkürlerimi sunarım.

Haziran 2009

Altay Hüseyin ERŞAHİN

Elektronik ve Haberleşme Mühendisi

İÇİNDEKİLER

Sayfa

KISALTMALAR ..... **vii**
TERİMLER ..... **ix**
ÇİZELGE LİSTESİ ..... **xi**
ŞEKİL LİSTESİ ..... **xiii**
ÖZET ..... **xv**
SUMMARY ..... **xvii**
1. GİRİŞ **1**
2. H.264/AVC ve ÇERÇEVE İÇİ KODLAMA **3**
2.1 H.264/AVC'nin Diğer Standartlardan Farklı Özellikleri 3
2.2 H.264/AVC Profilleri 6
2.3 H.264/AVC Video Kodlayıcı Katmanı 7
2.3.1 Dilimler, makrobloklar ve bloklar 9
2.3.2 Çerçeve içi kestirim 11
2.3.2.1 4x4 parlaklık (luma) kestirim modları 11
2.3.2.2 8x8 parlaklık kestirim modları 13
2.3.2.3 16x16 parlaklık kestirim modları 14
2.3.2.4 8x8 Renk (chroma) kestirim modları 15
2.3.3 Dönüşüm ve kuantalama 15
2.3.3.1 4x4 artık bloklar için dönüşüm ve kuantalama 18
2.3.3.2 4x4 parlaklık (luma) DC katsayıları için dönüşüm ve kuantalama 19
2.3.3.3 2x2 Renk (chroma) DC katsayıları için dönüşüm ve kuantalama 20
2.3.3.4 8x8 parlaklık dönüşümü 20
2.3.4 Entropi kodlama ve CABAC 21
2.3.4.1 Kontekst Uyumlu Ikili Aritmek Kodlama (CABAC) 22
2.3.5 Bit oranı – bozulma optimizasyonu 24
2.4 Çerçeve İçi Kodlama için Kodlayıcı Adımları 26
3. ŞABLON EŞLEME İLE ÇERÇEVE İÇİ KODLAMA **29**
3.1 Doku Sentezleme 29
3.2 Şablon Eşleme ile Doku Sentezleme 31
3.3 Şablon Eşleme ile Çerçeve İçi Kestirim 33
3.3.1 H.264/AVC kodlayıcı / kod çözücü üzerinde şablon eşleme 34
3.3.1.1 Kodlayıcı – kod çözücü senkronizasyonu 34
3.3.1.2 Çerçeve içi kestirim modu bilgisinin gönderilmesi 35
3.3.2 Şablon eşleme parametreleri 36
3.3.2.1 Alt blok büyüklüğü 37
3.3.2.2 Şablonun şekli ve büyüklüğü 37
3.3.2.3 Arama bölgesi büyüklüğü 38
3.3.2.4 Kaynak blok şablonuna karar verme kriteri 38
3.3.2.5 Arama bölgesi piksel hassasiyeti 39
3.3.2.6 Kullanılan kestirim alt bloğu sayısı 43

4. SONUÇLAR 45
4.1 Test Kriterleri ve Test Ortamı 45
4.2 Referans Uygulama 47
4.3 Şablon Eşleme Parametrelerinin Kodlama Verimliliğine Etkisi 51
4.3.1 8x8 bloklar için 4x4 alt blok kullanılması 51
4.3.2 Genişletilmiş şablon kullanılması 52
4.3.3 Arama bölgesinin büyütülmesi 52
4.3.4 Kaynak – hedef şablon eşleştirmesinde karesel farkların kullanımı 54
4.3.5 Şablon eşlemenin yarım piksel hassasiyetiyle yapılması 55
4.3.6 Şablon eşlemenin çeyrek piksel hassasiyetiyle yapılması 57
4.3.7 Birden fazla kestirim alt bloğunun kullanılması 58
4.3.8 Yarım ve çeyrek piksel hassasiyetle birden fazla kestirim alt bloğunun kullanılması 60
YORUM VE ÖNERİLER 63
KAYNAKLAR 65
EKLER 67

KISALTMALAR

ASO : Arbitrary Slice Ordering
AVC : Advanced Video Coding
BD-RATE : Bjontegaard Delta Rate
CABAC : Context Adaptive Binary Arithmetic Coding
CAVLC : Context Adaptive Variable Length Coding
FMO : Flexible Macroblock Ordering
IEC : International Engineering Consortium
ISO : International Organization for Standardization
ITU-T : International Telecommunications Union - Telecommunication
LPS : Least Probable Symbol
MB : Macroblock
MPEG : Moving Picture Experts Group
MPS : Most Probable Symbol
NAL : Network Abstraction Layer
QP : Quantization Parameter
RDO : Rate Distortion Optimization
SAD : Sum of Absoute Differences
SSD : Sum of Squared Differences
VCEG : Video Coding Experts Group
VCL : Video Coding Layer
VLC : Variable Length Coding

TERİMLER

Binarization : İkilileme
Binay Arithmetic Coder : İkili Aritmetik Kodlayıcı
Bit stream : Bit Akışı
Bitrate : Bit iletim oranı
Chroma component : Renk bileşeni
Context adaptive : Konteks uyumlu
Decode : Kod çözme
Encode : Kodlama
Feature matching : Özellik eşleme
Interlaced : Geçmeli
Intra frame coding : Çerçeve içi kodlama
Inverse transorm : Ters dönüşüm
Luma component : Parlaklık bileşeni
Markov random fields : Markov Rastlantısal Alanları
Most probable mode : En olası mod
Motion compensation : Hareket dengeleme
Prediction : Kestirim
Quantization : Kuantalama
Rate-distortion optimization: Bit oranı-bozulma optimizasyonu
Reoderding : Sıra düzenleme
Sampling : Örnekleme
Scaler matrix : Ölçekleyici matris
Scaling : Ölçekleme
Slice : Dilim
Syntax : Sözdizimi
Texture : Doku
Template matching : Şablon eşleme
Transform kernel : Dönüşüm çekirdeği

ÇİZELGE LİSTESİ

Sayfa

Çizelge 2.1 : H.264/AVC dilim tipleri 10
Çizelge 2.2 : H.264/AVC'de tanımlanan kuantalama adımı değerleri.....19
Çizelge 2.3 : Kodlanması gereken parametre örnekleri 21
Çizelge 3.1 : Kestirim modu sözdizimi elemanlarının kodlanması 36
Çizelge 4.1 : Test için kullanılan H.264/AVC kodlayıcı parametreleri 46
Çizelge 4.2 : Test için kullanılan videolar.....46
Çizelge 4.3 : Kestirim modu sözdizimi elemanlarının yeni modla beraber kodlanması 48
Çizelge 4.4 : Referans uygulama şablon eşleme parametreleri 48
Çizelge 4.5 : Önceki çalışmalarda elde edilen kazanç değerleri.....49
Çizelge 4.6 : Referans uygulama ile Standart kodlamanın verimlilik farkı 50
Çizelge 4.7 : JM yazılımı üzerinde şablon eşlemenin video kodlama süresine etkisi 50
Çizelge 4.8 : 8x8 lik bloklar için farklı alt blok büyüklüklerinin oluşturduğu % BD-Rate değişimi (JM14.1'e göre).....51
Çizelge 4.9 : Genişletilmiş şablon kullanımın oluşturduğu BD-Rate değişimleri...52
Çizelge 4.10 : Şablon eşleştirmede SSD kullanımın oluşturduğu BD-Rate değişimleri 55
Çizelge 4.11 : Arama bölgesi hassasiyetinin artırılmasının oluşturduğu % BD-Rate değişimleri.....56
Çizelge 4.12 : Arama bölgesi hassasiyetinin artırılmasının oluşturduğu ortalama çerçeve kodlama süresi artışları (milisaniye).....56
Çizelge 4.13 : Seçilen piksel atma konfigürasyonuna göre elde edilen % BD-Rate kazancı 57
Çizelge 4.14 : Arama bölgesinde çeyrek piksel kullanımının oluşturduğu % BD-Rate değişimleri.....58
Çizelge 4.15 : Birden fazla kestirim alt bloğu kullanımına ilişkin önceki çalışmalarda elde edilen değerler.....59
Çizelge 4.16 : Birden fazla kestirim alt bloğu kullanımıyla elde edilen kazanç değerleri 59
Çizelge A.1 : Arama bölgesi büyüklüklerine karşı gelen %BD-Rate kazançları 70
Çizelge A.2 : Birden fazla alt bloğun ortalaması alınarak yapılan kestirimlerin BD-Rate kazançları (tam piksel).....71
Çizelge A.3 : Yarım piksel hassasiyeti ile birden fazla alt blok kullanarak şablon eşlemede elde edilen ortalama kazanç değerleri 72
Çizelge A.4 : Çeyrek piksel hassasiyeti ile birden fazla alt blok kullanarak şablon eşlemede elde edilen ortalama kazanç değerleri 72

ŞEKİL LİSTESİ

Sayfa

Şekil 2.1 : H.264/AVC standardında tanımlı profiller 7
Şekil 2.2 : H.264/AVC kodlama katmanları.....7
Şekil 2.3 : H.264/AVC kodlayıcı yapısı.....8
Şekil 2.4 : H. 264/AVC kod çözücü yapısı.....9
Şekil 2.5 : 4x4 lük kestirim bloğu ve komşu pikseller 12
Şekil 2.6 : 4x4 parlaklık kestirim modları 13
Şekil 2.7 : 8x8 parlaklık kestiriminde kullanılan pikseller.....14
Şekil 2.8 : 16x16 kestirim modlarından Mod 0 ve Mod 1.....15
Şekil 2.9 : Makroblok içerisindeki artık blokların taranma sıraları 16
Şekil 2.10 : Dönüşüm, kuantalama, geri kuantalama ve ters dönüşüm blok diyagramı 18
Şekil 2.11 : 8x8 dönüşümde kullanılan normalize edilmemiş dönüşüm matrisi 20
Şekil 2.12 : CABAC kodlama blok diyagramı 22
Şekil 2.13 : LPS olasılık değerleri ve olasılık kestiriminin güncellenmesi için geçiş kuralları.....24
Şekil 3.1 : Şablon eşleme ile doku sentezleme.. 32
Şekil 3.2 : Şablon eşleme ile çerçeve içi kestirimde kullanılan bloklar.....34
Şekil 3.3 : Farklı blok büyüklükleri için şablon eşleme alt blok büyüklükleri.....37
Şekil 3.4 : Genişletilmiş şablonun şekli.....38
Şekil 3.5 : Yarım piksel hassasiyetine geçişte oluşturulan pikseller.....40
Şekil 3.6 : Şablon için tam piksel - yarım piksel geçişi.....41
Şekil 3.7 : Hedef alt bloğun oluşturulması için seçilebilecek piksel atma konfigürasyonları.....41
Şekil 3.8 : Çeyrek piksellerin oluşturulması.....42
Şekil 4.1 : Arama bölgesi büyüklüklerinin BD-Rate kazancına ortalama etkisi 53
Şekil 4.2 : Arama bölgesi büyüklüklerinin video kodlama süresine etkisi 54
Şekil 4.3 : Hedef alt bloğun oluşturulması için seçilen piksel atma konfigürasyon 58
Şekil 4.4 : Birden fazla alt bloğun ortalaması alınarak yapılan kestirimlerin BD-Rate kazançları 60
Şekil 4.5 : Tam, yarım ve çeyrek piksel hassasiyetleri ile birden fazla alt blok kullanarak şablon eşlemede elde edilen ortalama kazanç değerleri.....62

H.264/AVC İLE DÜŞÜK GECİKMELİ SIKIŞTIRMA

ÖZET

Video kodlama uygulamalarında bazen kodlama ile kod çözme arasında geçen zaman, kritik önem taşımaktadır. Kodlayıcı tarafından kodlanan video ile kod çözücüden elde edilen video arasında mümkün olduğunca az gecikme olması gereken bu uygulamalarda, kodlayıcının kodlamayı geciktirecek yöntemler kullanmaması gerekmektedir. Ayrıca kodlanmış video çerçevelerine rasgele erişimin hızlı olması gerektiği takdirde, kodlayıcının çerçeveler arasındaki korelasyondan faydalanmadan, çerçeveleri bağımsız kodlaması gerekmektedir. Bu durumda kodlama yalnızca çerçeve içi korelasyonlar kullanılarak yapılır. Bu çalışmada çerçevelerin düşük gecikmeli ve birbirinden bağımsız kodlandığı, H.264/AVC standardını temel alan kodlayıcı – kod çözücü ikilisinin bulunduğu kapalı bir sistemde, video kodlama verimliliğini artırabilecek bir çerçeve içi kestirim yöntemi incelenmiştir.

Şablon eşleme adı verilen bu yöntemde, kestirimi yapılacak çerçeve bölgesi doku sentezlenecek bir alan gibi düşünülmekte ve çevresindeki belirli büyüklükteki bir alan kullanılarak bu bölgenin sentezlemesi yapılmaktadır. H.264/AVC standardında tanımlanan çerçeve içi kestirim modlarının ortak özelliği, yalnızca kestirimi yapılacak bölgeye komşu olan önceden kodlanmış piksellerin kullanılmasıdır. Bu yöntemde ise, sadece komşu pikseller değil, kodlanmakta olan bölgenin geniş bir komşuluğu kestirim için kullanılmaktadır. Bu yöntemin bir diğer farkı da, kestirim için hangi çerçeve bölgelerinin kullanılacağına hem kodlayıcıda hem de kod çözücüde karar veriliyor olmasıdır.

Bu çalışmada kodlama verimliliğini %5'e kadar artıran bu çerçeve içi kestirim yönteminin parametrelerine dair önceki çalışmalarda yapılan önermeler incelenmiştir. Ayrıca şablon eşlemenin yarım ve çeyrek piksel hassasiyetle yapılması ve şablon eşlemede karesel farkların kullanımı yöntemleri de denenmiş ve sağladıkları ortalama kodlama kazancı ile yöntemlerin başarımları değerlendirilmiştir.

LOW DELAY CODING WITH H.264/AVC

SUMMARY

In some video coding applications, time interval between encoding and decoding of the video is time critical. Since the encoder has to encode the sampled video frames as soon as possible, it should avoid using encoding methods that delay the encoding of a video frame. Also if random access to encoded frames is also time critical, correlations between frames can not be exploited (inter frame coding) and frames should be coded independently. In this case, only correlations within a frame can be used (intra frame coding). In this work, in a system where frames are coded independently and with low delay and a coder / encoder pair exists which are based on H.264/AVC standard, an intra frame prediction method that can increase the coding efficiency is studied.

In this method which is referred as "Template Matching", a part of the frame that will be predicted is texture synthesized by using a certain size of neighbour region to find the most resembling texture. The intra frame prediction modes defined in H.264/AVC standard, only use the already encoded pixels which are neighbour to the region to be predicted. However, in this method, not only neighbour pixels but also pixels that are in a large neighbourhood region of the frame part to be predicted are used. Another property of this method is, both encoder and decoder have to decide which parts of the frame compose the prediction block.

This method increases the coding efficiency by 5 % and in this work the previously proposed parameters about this intra prediction mode are investigated. Also matching templates with half and quarter pixel accuracy and using sum of squared errors for choosing the source template are tested and their contributions to the coding efficiency are investigated.

1. GİRİŞ

Hareketli görüntülerin iletimi ve saklanmasında başlıca ihtiyaçlardan biri olan görüntü (video) verisinin sıkıştırılması, sayısal ortamların kullanılmaya başlanmasıyla birlikte önemli bir konu olarak ortaya çıkmıştır. Video sıkıştırmada görüntü kalitesini mümkün olduğunca bozmadan, iletim için gerekli olan band genişliğini ve saklama için gerekli olan sıkıştırılmış dosya boyutunu azaltmak başlıca hedeflerdendir. Son yıllarda geliştirilen video kodlama standartlarından en yenisi ve diğerlerine göre en verimli olanı H.264/AVC, ITU-T VCEG ve ISO/IEC MPEG tarafından geliştirilmiştir ve günümüzde yayın olarak kullanılmaktadır. H.264/AVC gibi standartlar video kodlarken, çerçeveler arası korelasyondan faydalanmakla birlikte, bir çerçevenin kendi içerisindeki korelasyonunu da kodlama verimliliğini artırmak için kullanabilmektedirler.

Kodlama ve kod çözme işlemlerinin zaman gecikmesiz yapılması gereken uygulamalarda, çerçeveler arası korelasyonu kullanabilmek için gerekli olan, kodlamanın birkaç çerçeve örnekleme periyodu kadar geciktirilmesine müsamaha gösterilemez. Bu nedenle böyle bir uygulamada sadece çerçeve içi korelasyonu kullanmaya izin verilebilir. H.264/AVC sadece çerçeve içi kodlamayı da destekleyen bir standart olduğundan, gecikmesiz kodlama / kod çözme uygulamalarında kullanılabilir.

Sadece çerçeve içi korelasyon kullanılarak sıkıştırma yapıldığında, kodlamada çerçeve içi kestirim kullanılması gerekmektedir. H.264/AVC standardında çerçeve içi kestirim metodları tanımlanmıştır ve bit iletim oranıyla beraber kodlanmış videoda oluşacak bozulmayı dengeleyen bir sistem en uygun çerçeve içi kestirim modunu seçer.

H.264/AVC'nin çerçeve içi kodlamanın (sıkıştırma) performansını iyileştirmek için, çerçeve içi kestirim metodlarının iyileştirilmesi ya da yenilerinin eklenmesi gerekmektedir. Bu çalışmada H.264/AVC çerçeve içi kestirim modlarına bir yenisi eklenerek, kodlama verimliliğinin iyileştirilmesi amaçlanmaktadır. Bu iyileştirme

yapılırken standartta yer alan yöntem referans alınacak, ancak standardın tanımladığı kodlama / kod çözme adımlarının dışına çıkılacaktır.

Doku sentezleme, bozulmuş resimlerin onarılması, iletiminde kayıp yaşanmış video veya fotoğrafların tamamlanması, istenmeyen bir objenin resimden çıkartılması ve video sıkıştırma gibi alanlarda uygulaması olan bir görüntü işleme yöntemidir. Bir görüntü (video çerçevesi de olabilir) üzerindeki istenilen bölgenin diğer bölgeleri kullanarak elde edilebilmesi doku sentezlemenin video sıkıştırma alanında kullanılabilmesini sağlamaktadır. Çerçevenin kodlanmakta olan bölgesinin bir doku sentezleyici ile oluşturulması (ya da kestirilmesi), çerçeve içi kestirimin daha başarılı bir şekilde yapılmasını sağlayabilir.

H.264/AVC kodlayıcısı hibrit bir kodlayıcıdır yani, kodlama yapılırken aynı zamanda kod çözme işlemi de yapılarak kod çözücünün ne elde edeceği tam olarak bilinmektedir. Standartta tanımlı çerçeve içi kestirim metodunda kodlayıcı, kod çözücünün de sahip olduğu bilgileri kullanarak kestirim yapar. Bu nedenle doku sentezleme gibi bir yöntemin çerçeve içi kestirim modlarına eklenmesinde, hem kodlayıcı hem de kod çözücüdeki ortak bilgileri kullanan bir doku sentezleyici kullanmak gerekmektedir. Ayrıca bit iletim oranını artırmamak için, kodlayıcının tercih ettiği çerçeve içi kestirim modunu kod çözücüye iletirken kullandığı bilgilere, vektör ya da indis gibi yeni veriler eklenmesi gerekmemelidir.

Bu çalışmada şablon eşleme adı verilen doku sentezleme modunun H.264/AVC çerçeve içi kestirim metoduna dahil edilirken seçilen sentezleme parametrelerinin verimliği araştırılmaktadır. Kodlama ve kod çözme adımlarında gerekli değişiklikler ve eklemeler yapılarak şablon eşleme metodu çerçeve içi kestirim modlarına eklenecektir. Şablon eşleme ile ilgili literatürde yer alan geçmiş çalışmalarda önerilen yöntemlerle, şablon eşlemenin kullanılabileceği uygulamalar için en uygun şablon eşleme parametreleri seçilecektir.

Bölüm 2'de H.264/AVC standardı ve çerçeve içi kodlama hakkında detaylı bilgi verilecektir. Bölüm 3'de şablon eşlemenin çerçeve içi kodlamada kullanımı ve şablon eşleme parametrelerinin neler olduğu anlatılacaktır. Bölüm 4'de ise bu parametrelerin kullanımına ilişkin sonuçlar verilecek ve hangi durumlarda kullanımlarının uygun olduğu belirlenmeye çalışılacaktır.

2. H.264/AVC ve ÇERÇEVE İÇİ KODLAMA

H.264/AVC, ITU-T ve ISO/IEC tarafından ortaklaşa geliştirilen en son video kodlama standardıdır. Bu standardın geliştirilmesindeki başlıca hedefler, video sıkıştırma performansının iyileştirilmesi ve sıkıştırılmış video verisinin ağ üzerinden iletime daha uygun hale getirilmesidir.

Bu standart, kodlayıcı / kod çözücü ikilisinden sadece kod çözücüye ilişkin tanımlamalar yapmaktadır. Kod çözücüye girdi olarak verilebilecek bit akışı ve bu bit akışının içerisinde yer alan sözdizimi (syntax) ile sözdizimi bileşenlerinin çözülme işlemine dair tanımlamalar standart tarafından yapılmaktadır. Böylece standarda uygun bir bit akışı girdi olarak alan her standarda uyumlu kod çözücü, aynı çıktıyı verecektir. Bu nedenle videonun kodlanması sırasında hangi yöntemlerin izlenebileceği kesin bir şekilde tanımlı değildir ve hedef uygulamaya göre kodlayıcı optimize edilebilmektedir.

2.1 H.264/AVC'nin Diğer Standartlardan Farklı Özellikleri

Kendisinden daha önce oluşturulmuş standartlardan farklı olarak H.264/AVC'in, kodlanacak bir çerçevenin içerisindeki değerlerin tahmin edilebilirliğini artırarak kodlama verimliliğinin yükselmesini sağlayan özellikleri şunlardır.

- Değişken blok büyüklüklü hareket dengeleme: Hareket dengeleme için kullanılacak blok büyüklüğü ve şekli için birden fazla seçenek sağlanmıştır. Parlaklık bileşeni için en küçük hareket dengeleme blok büyüklüğü 4x4'dür [1].
- Dörtte bir piksel doğruluklu hareket dengeleme: Daha önce MPEG-4 (2. kısım) standardının gelişmiş bir profiline eklenmiş olan bu özellik daha da geliştirilmiş ve gerekli olan enterpolasyon işleminin karmaşıklığı azaltılmıştır [1].
- Birden fazla referans çerçeve kullanarak hareket dengeleme: İlk defa H.263++ standardında kullanılan referans çerçeve seçme tekniği geliştirilerek kodlayıcının daha fazla çerçeve içerisinden (15) seçim yapabilmesi sağlanmıştır. Bu

çerçeveler önceden kodlanmış ve kod çözücüden geçirilmiş çerçevelerden oluşurlar [1].

- Ağırlıklandırılmış kestirim: H.264/AVC ile birlikte gelen bu özellik ile hareket dengelenmiş kestirim işareti kodlayıcı tarafından belirtilen miktarlarla ağırlıklandırılabilir ve ötelenebilir [1].
- Çerçeve içi kodlama için yönlü uzamsal kestirim: Kodlanmakta olan çerçevenin önceden kodlanmış bölümlerinin kenarlarını ekstrapolasyonla bulmak için eklenmiş bir özelliktir. Kestirim işaretinin kalitesini artırmakla birlikte, çerçeve içi modu ile kodlanmamış komşu bölgelerden kestirim yapılabilmesini sağlar [1].

Geliştirilmiş bu kestirim yöntemlerine ilaveten, kodlama verimini artıran diğer özellikler arasında aşağıdakiler yer almaktadır:

- Küçük blok büyüklüklü dönüşüm: Daha önceki standartlarda dönüşüm blok büyüklüğü olarak genelde 8x8 kullanılırken, H.264/AVC 4x4 büyüklüklü dönüşüme ağırlık vermektedir. Bu sayede kodlayıcı işaretleri yerel bilgilere daha uyarlamalı olarak ifade ederek, çınlama etkisi olarak bilinen hatların azalmasını sağlar [1].
- Tam eşleşen ters dönüşüm: Önceki video kodlama standartlarında video verisi üzerinde kullanılan dönüşümler genelde belirli bir hata toleransı içinde tanımlanabiliyordu. Nedeni ise belirtilen ideal ters dönüşümlerle bire bir eşleşen bir dönüşüm yöntemi elde etmenin imkansız olmasıydı. Bu nedenle her kod çözücü tasarımı az da olsa farklı bir çözülmüş video üretiyordu ve video kalitesinde düşüşe neden oluyordu. H.264/AVC kodlayıcı ve kod çözücüde aynı videonun elde edilebilmesini sağlayan ilk standarttır [1].
- Aritmetik entropi kodlayıcı: İleri bir entropi kodlama yöntemi olarak bilinen aritmetik kodlama H.264/AVC'ye dahil edilmiştir. Daha önce H.263 standardında bir seçenek olarak bulunan bu metodun daha efektif bir kullanım tekniği geliştirilmiş ve Kontekst Uyumlu İkili Aritmetik Kodlama (Context Adaptive Binary Arithmetic Coding - CABAC) adı altında standarda eklenmiştir [1].
- Kontekst uyumlu entropi kodlama: H.264/AVC'de kullanılan her iki entropi kodlama yöntemi de daha iyi performans sağlamak için kontekst tabanlı

uyumluluk kullanırlar. Bunlar CAVLC (Kontekst Uyumlu Değişken Uzunluklu kodlama) ve CABAC'dır [1].

Veri hataları ve kayıplarına dayanıklılık ve çeşitli ağ ortamlarında çalışabilecek esneklik H.264/AVC'de aşağıda belirtilen özelliklerle sağlanmaktadır:

- Parametre seti yapısı: Parametre seti tasarımı, başlık bilgisinin daha gürbüz ve etkin bir şekilde taşınmasını sağlamaktadır. Verinin (sıra başlığı veya çerçeve başlık bilgisi gibi) bazı önemli bitlerinin kayıp edilmesi halinde kod çözme işleminde oluşacak olumsuzlukların giderilmesi için önemli bilgiler ayrık olarak ve esnetilebilen bir yöntemle kod çözücüye gönderilirler [1].
- NAL (ağ soyutlama katmanı) birimi sözdizimi yapısı: Her bir sözdizimi yapısı, NAL birimi adı verilen mantıksal veri paketlerine yerleştirilirler. NAL birimi sözdizimi yapısı, video verisinin ağın özelliklerine uygun bir şekilde taşınabilmesi için özelleştirilebilir bir yapıdadır [1].
- Değişken dilim büyüklüğü: H.264/AVC'de bir çerçevenin bölündüğü dilimlerin büyüklüğü sabit değildir ve MPEG-1'de olduğu gibi değiştirilebilir [1].
- Değişken makroblok sıralaması (FMO): Çerçeveler, her biri bağımsız olarak çözülebilen dilimlere ayrılabilirler. Bu dilimler içerisinde makroblokların sıraları değiştirilerek (FMO kullanılarak), veri kayıplarına karşın dayanıklılık artırılabilir [1].
- Değişken dilim sıralaması (ASO): Bir çerçevenin dilimleri birbirinden bağımsız olarak çözülebildiğinden dolayı, her biri istenilen sırada gönderilip alınabilir. Bu sayede gerçek zamanlı uygulamalarda yaşanan gecikmelere rağmen videonun önemli kısımlarındaki gecikmeler daha makul seviyelere çekilebilir [1].
- Veri bölümleme: Video verisinin içerisindeki değişik bölgelere ait bazı bilgiler (örneğin hareket vektörleri) diğer bilgilere göre videonun gösterilebilmesi açısından daha önemli ve kritik olduğundan, H.264/AVC her bir dilimin sözdiziminin, iletim için en fazla üç parça olacak şekilde ayrılabilmesine izin vermektedir [1].

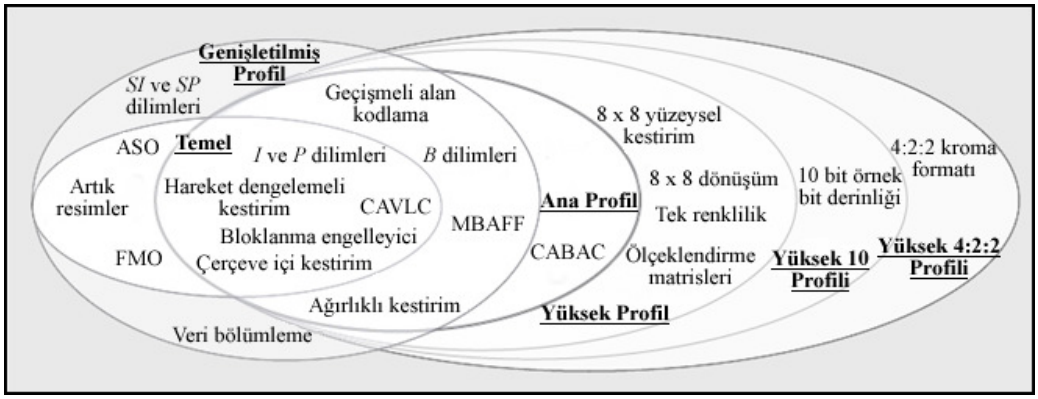
Bu özellikler gecikmenin önemsenmediği tek yönlü video akışının olduğu uygulamalarda H.264/AVC ana profiline, MPEG 2'ye göre ortalama %63 ve H.263 yüksek gecikme profiline göre ortalama %47 kodlama kazancı sağlamaktadır [2].

Gecikmenin önemli olduğu uygulamalarda ise (örneğin video konferans), H.264/AVC temel profili, H.263 temel profiline göre ortalama %41 ve H.263 iki yönlü yüksek sıkıştırma profiline göre ortalama %27 kodlama kazancı sağlamaktadır [2].

2.2 H.264/AVC Profilleri

H.264/AVC standardında, farklı özelliklerin desteğini içeren ve hedef uygulamaya göre kodlayıcı tarafından seçilen birden fazla profil tanımlanmıştır. Bu profillere zaman içerisinde yenileri eklenmektedir. Bu profillerden en yaygın kullanılanları aşağıda belirtilmiştir ve Şekil 2.1'de görülmektedir.

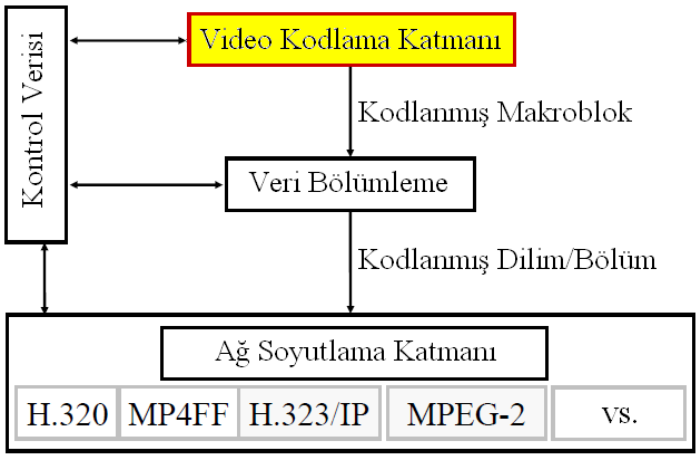
- Temel Profil (Baseline Profile – BP): Sınırlı işlem gücü olan ve düşük maliyetli uygulamalarda (örneğin video konferans) kullanılır.
- Ana Profil (Main Profile – MP): Genelde kullanılacak profil olarak tasarlanmıştır ve TV yayınları ile depolama uygulamaları için kullanılmaktadır. Yüksek Profil'in standarda eklenmesi ile kullanım alanı azalmıştır.
- Genişletilmiş Profil (Extended Profile – XP): Yüksek sıkıştırma ve veri kayıplarına dayanıklılık desteği içermektedir.
- Yüksek Profil (High Profile – HiP): TV yayınları ve disk depolama uygulamaları için kullanılmaktadır. Yüksek çözünürlüklü televizyon uygulamalarında (Blueray gibi) da tercih edilmektedir.
- Yüksek 10 Profili (High 10 Profile – Hi10P): Yüksek Profil'e örnek başına 8 bit yerine 9 veya 10 bit kullanılması desteğini eklemektedir. Bu sayede örneklerin hassasiyet düzeyleri artırılmaktadır.
- Yüksek 4:2:2 Profili (High 4:2:2 Profile – Hi422P): Geçmeli (interlaced) video desteği gerektiren uygulamalarda kullanılan bu profil, Yüksek 10 Profili'ne 4:2:2 renk alt örnekleme formatı desteğini eklemektedir.



Şekil 2.1 : H.264/AVC standardında tanımlı profiller

2.3 H.264/AVC Video Kodlayıcı Katmanı

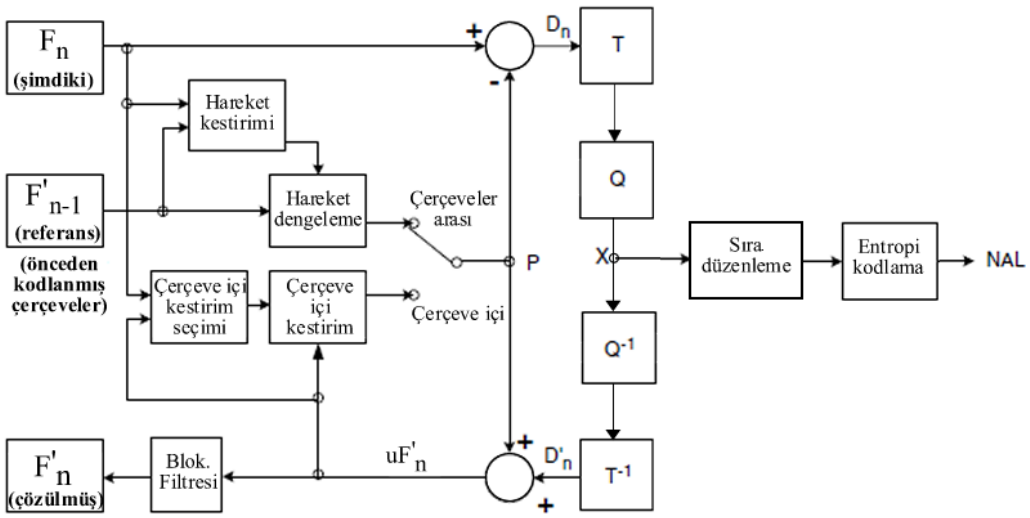
Değişik türdeki uygulamalarda gerekli olan ihtiyaçları karşılayabilmek için H.264/AVC standardı, video verisinin katmanlar halinde kodlanmasını ve paketlenmesini sağlamaktadır. Video verisin verimli olarak kodlanabilmesi için video kodlama katmanı (VCL) ve farklı iletim yöntemlerine ya da depolama ortamlarına uyumluluk sağlanabilmesi için, videonun VCL gösterimini gerekli şekilde biçimlendiren ve gerekli başlık bilgilerini ekleyen ağ soyutlama katmanı (NAL) standart tarafından tanımlanmıştır. Bu katmanlar Şekil 2.2'de görülmektedir.



Şekil 2.2 : H.264/AVC kodlama katmanları

H.264/AVC standardının kesin bir kodlayıcı/çözücü çifti tanımlamamasına rağmen, kodlanmış veriye ilişkin sözdizimini ve çözme yöntemlerini tanımlıyor olması,

standarda uygun kodlayıcı ve çözücülerde ortak fonksiyonel elemanlar bulunmasını sağlamaktadır. Bu elemanların bir kısmı Şekil 2.3 ve Şekil 2.4'de görülmektedir.



Şekil 2.3 : H.264/AVC kodlayıcı yapısı [3]

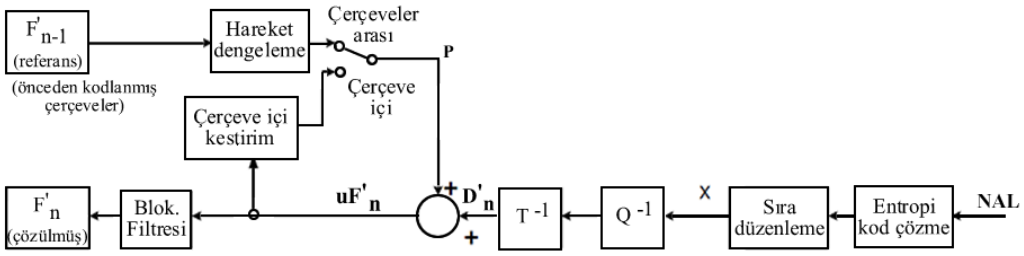
Şekil 2.3'de görülen kodlayıcıda iki akış yolu vardır ve bunlar ileri yol (soldan sağa) ve yeniden oluşturma yolu (sağdan sola) olarak adlandırılırlar. Buna karşın Şekil 2.4'de görülen akış yolu sağdan sola doğrudur ve kodlayıcıdaki yeniden oluşturma yoluna benzemektedir.

Kodlayıcıdaki ileri yönlü akış şu şekilde işlemektedir: Kodlanacak olan F_n çerçevesi, parlaklık bileşenleri için 16x16 piksel, renk bileşenleri için 8x8 büyüklüğündeki makrobloklara bölünür. Her makroblok için ya çerçeve içi ya da çerçeveler arası kestirim kullanılır ve halihazırda kodlanmış resim elemanları kullanılarak yapılan kestirim sonucu bir kestirim bloğu oluşturulur (Şekilde P ile gösterilmiştir). Çerçeve içi kodlama durumunda, kestirim bloğu, kodlanan resmin o an kodlanmakta olan makroblokla aynı dilim içerisinde bulunan ve halihazırda kodlanmış, geri çözülmüş ve yeniden oluşturulmuş resim bölgeleri (uF'_n) kullanılarak oluşturulur. Çerçeveler arası kodlama durumunda ise kestirim bloğu, referans çerçeveler arasındaki bir ya da iki referans çerçeveden hareket dengelemeli kestirim yöntemi ile oluşturulur. Şekillerde referans çerçeve F'_{n-1} olarak gösterilmiştir ancak referans çerçeve, kodlanmakta olan resmin örneklenme zamanına göre önceden ya da sonradan örneklenmiş bir resmin kodlanmış çerçevesi olabilir.

Tahmin bloğu P, işlenmekte olan bloktan çıkartılarak bir artık blok (şekilde D_n ile gösterilmiştir) oluşturulur. Bu artık blok, blok bazlı dönüşüme sokulur ve sonrasında

kuantalanarak şekilde X ile gösterilen kuantalanmış dönüşüm katsayıları oluşturulur. Tekrar sıralanan bu katsayılar entropi kodlayıcıya gönderilirler. Entropi kodlanmış katsayılar, her bir bloğu çözmek için gerekli diğer ek bilgilerle (kestirim modları, kuantalama parametresi, hareket vektörü bilgisi, vs.) birlikte ağ soyutlama katmanına aktarılacak olan sıkıştırılmış bit dizisini oluştururlar.

Kodlayıcıdaki geri yönlü akış ile yeniden oluşturma ise şu şekilde olmaktadır: X ile gösterilmiş olan katsayılar ölçeklendirilir (Q^{-1}) ve ters dönüşüm (T^{-1}) uygulanarak fark bloğu (D'_n) elde edilir. Kestirim bloğu P ile fark bloğu toplanarak, yeniden oluşturulmuş uF'_n (orijinal bloğun çözülmüş ancak filtrelenmemiş hali) elde edilir. Daha sonra bloklanma etkilerini gidermek üzere bir filtreden geçirilir. Yeniden elde edilmiş çerçeveleri oluşturmak için bu bloklardan gerekli miktarı bir araya getirilir.



Şekil 2.4 : H. 264/AVC kod çözücü yapısı [3]

Kod çözücü ise sıkıştırılmış bit akışını NAL'dan alarak entropi kod çözücüyü kullanarak akış içerisindeki veri bileşenlerinden kuantalanmış X katsayılarını elde eder. Bu katsayılar ölçeklendirilir ve ters dönüşüm uygulanarak, kodlayıcıdaki ile tamamen aynı olan D'_n elde edilmiş olur. Bit akışının içerisinde yer alan ek bilgileri de kullanarak kod çözücü, kodlayıcıdaki ile birebir aynı olan kestirim bloğu P'yi oluşturur. Bu P bloğu D'_n'e eklenerek uF'_n oluşturulur. En son uF'_n filtrelenerek çözülmüş F'_n bloğu elde edilir.

2.3.1 Dilimler, makrobloklar ve bloklar

Kodlanacak çerçeve sabit büyüklükteki, kare şeklindeki makrobloklara bölünür. Bu makroblokların büyüklükleri parlaklık bileşenleri için 16x16 piksel iken, renk bileşenleri (chroma) için 8x8 piksel büyüklüğündedir.

Her bir çerçeve bir ya da daha fazla dilime bölünerek kodlanır. Dilimlerin içerisinde ise makrobloklar bulunur ve bir dilim içerisinde bulunan makroblok sayısı en az 1 ve

en fazla çerçeve içerisindeki toplam makroblok sayısı kadardır. Çerçeve içerisindeki her dilim başına düşen makroblok sayısı aynı olmak zorunda değildir. Dilimler arasında birbirine bağımlılık en alt düzeydedir (her biri ayrı ayrı çözülebilir). Bu şekilde oluşabilecek muhtemel veri hataların yayılmasının önüne geçilmiş olur. Bloklar arasındaki keskin geçişleri yumuşatmak için kullanılan bloklanma önleyici filtre için diğer komşu dilimlere ihtiyaç duyulabilir. Beş farklı dilim tipi vardır (Çizelge 2.1'de görülmektedir) ve bir çerçeve içerisinde farklı tipte dilimler yer alabilir.

Çizelge 2.1 : H.264/AVC dilim tipleri [3]

Dilim Tipi	Tanımı	Profiller
I (çerçeve içi)	Sadece I makrobloklardan oluşur ve her bir makroblok aynı dilim içerisindeki daha önceden kodlanmış piksellerden kestirim yapılarak oluşturur.	Hepsinde var
P (çerçeveler arası)	P tipi makrobloklar içerir ve makrobloklar ya da makroblok parçaları önceden kodlanmış çerçevelere ya da I makrobloklara referans verilerek kestirilir.	Hepsinde var
B (iki yönlü kestirim)	B tipi makrobloklar içerir ve makrobloklar ya da makroblok parçaları önceden veya sonrasında kodlanmış çerçevelere ya da I makrobloklara referans verilerek kestirilir.	Genişletilmiş ve Ana Profil
SP (anahtarlamalı P)	Kodlanmış bit akışları arasında geçiş yapılabilmesini sağlar. Sadece P ve/veya I makrobloklar içerir.	Genişletilmiş Profil
SI (anahtarlamalı I)	Kodlanmış bit akışları arasında geçiş yapılabilmesini sağlar. Sadece SI makrobloklar (çerçeve içi makroblokların özel bir tipi) içerir.	Genişletilmiş Profil

Kodlanan her makroblok daha önceden kodlanmış bir veriden kestirilir. Çerçeve içi (I) makrobloklarındaki her bir örnek, aynı dilim içerisindeki önceden kodlanmış, çözülmüş ve yeniden oluşturulmuş örnek değerlerinden kestirilir. Bu kestirim, kodlanmakta olan makrobloktan çıkartılır ve geri kalan artık değerler sıkıştırılarak kod çözücüye gönderilir.

Ayrıca makrobloğun hangi kestirim yöntemi kullanılarak kodlandığı, gerekiyorsa hareket vektörleri ve diğer bilgiler de makrobloğun sıkıştırılmış artık örnek

değerleriyle birlikte gönderilerek, kod çözücünün makrobloğu nasıl çözmesi gerektiği belirtilmiş olur. Böylece kod çözücü, kodlayıcı tarafından belirtilen parametrelere göre kod çözme ve kestirim işlemini gerçekleştirerek kodlayıcıda bulunan kodlanmış/çözülmüş makroblok bilgisinin aynısına ulaşır. Bunu gerçekleştirebilmek için kodlayıcı kestirimlerinde, çerçevede örneklerin kodlanmış ve geri kod çözülmüş halini kullanır. Bu şekilde her iki tarafın da aynı kestirimi yapabilmesi sağlanmış olur.

2.3.2 Çerçeve içi kestirim

Her makroblok için bir kodlama tipi seçilir. Makrobloğun içinde bulunduğu dilimin tipine göre seçilebilecek kodlama tipleri değişiklik gösterir. Tüm dilimlerde kullanılabilecek kodlama tipleri ise şunlardır: "çerçeve içi 4x4", "çerçeve içi 8x8", "çerçeve içi 16x16" ve IPCM. IPCM modunda kestirim yapılmaz ve blok içerisindeki piksel değerleri doğrudan kodlanarak kod çözücüye gönderilir.

Kodlanmakta olan makrobloğun hangi kodlama tipinde olduğu, içerisindeki blokların tamamını etkiler. 16x16 pikselden oluşan bir makroblok eğer "çerçeve içi 4x4" tipinde ise, 16'ya bölünür ve her blok 4x4 piksel içerir. Sonrasında 4x4 lük her blok sırayla kodlanır.

H.264/AVC'de çerçeve içi kestirim uzaysal domende yapılır. Kestirim, kodlanmakta olan bloğun sol ve üst komşuluklarında bulunan, daha önceden kodlanmış, kod çözülmüş ve yeniden elde edilmiş blokların pikselleri kullanılarak yapılır.

2.3.2.1 4x4 parlaklık (luma) kestirim modları

4x4 lük kestirim modları, resmin parlaklık bileşenleri kodlanırken kullanılabilir ve resimlerin yüksek detay içeren kısımlarını kodlamak için uygundur. Her 4x4 lük blok, komşu bloklardaki pikseller kullanılarak kestirilir. Şekil 2.5'de görüldüğü gibi a-p ile belirtilen 16 piksel değeri önceden kodlanmış/kod çözülmüş komşu A-Q pikselleri kullanılarak kestirilir. Her bir 4x4 blok için kullanılabilecek dokuz kestirim modu mevcuttur.

Q	A	B	C	D	E	F	G	H
I	a	b	c	d				
J	e	f	g	h				
K	i	j	k	l				
L	m	n	o	p				

Şekil 2.5 : 4x4 lük kestirim bloğu ve komşu pikseller

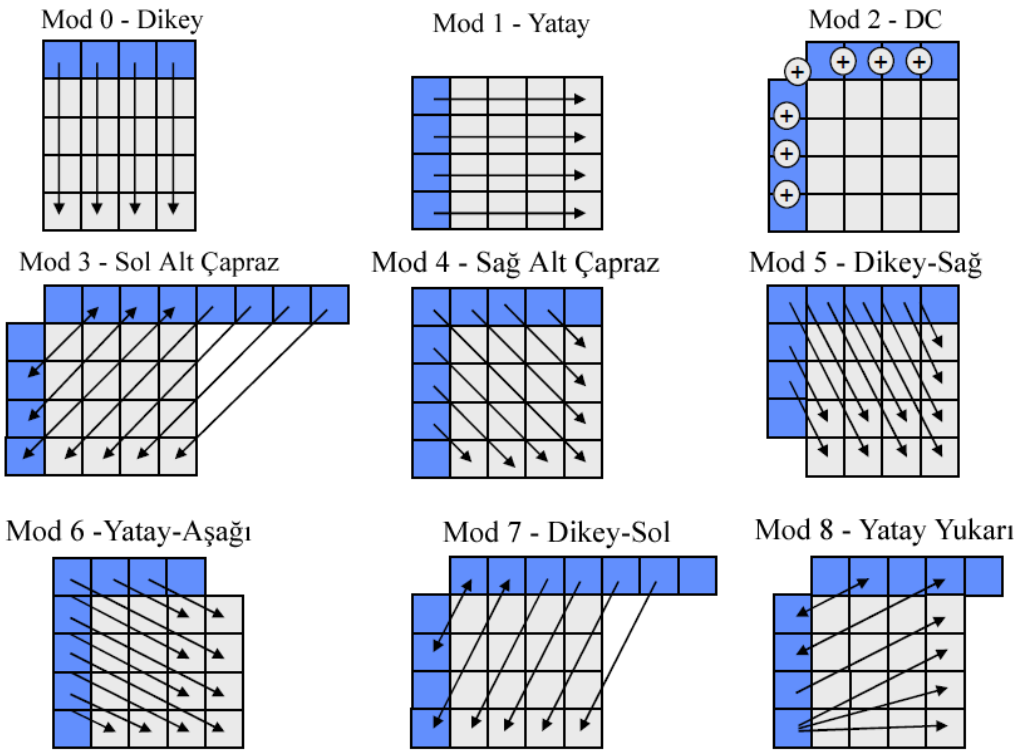
Şekil 2.6'de verilmiş olan dokuz kestirim modu şu şekilde çalışır:

- Mod 0'da 4x4 lük bloğun üzerinde yer alan piksellerin değerleri şekildeki gibi düşey olarak bloğun içerisine kopyalanır.
- Mod 1'de bloğun solunda yer alan piksellerin değerleri şekilde gibi yatay olarak bloğun içerisine kopyalanır.
- Mod 2'de bloğun hemen üstünde ve solunda yer alan piksel değerlerinin ortalaması alınır ve bloğun her pikseline bu ortalama, kestirim değeri olarak verilir.
- Geri kalan modlarda ise çapraz kestirim yapılır. Bu modların isimlerinde belirtildiği yönlerde kestirim yapılır ve resmin ilgili bloğunda belirtilen yönde bir doku mevcut ise tercih edilme olasılıkları yüksektir.

Eğer Sol Alt Çapraz modu için gerekli olan E – H pikselleri hazır değilse (henüz kodlanmamış ya da dilim dışında olabilir), sadece kestirimde kullanılabilmeleri için yerlerine D pikselinin değeri kopyalanır.

Tüm modlar bütün bloklar için kullanılamayabilir. Bir kestirim modu denenmeden önce, ihtiyaç duyduğu piksellerin bulunduğu blokların hazır olup olmadığı (kodlanmış olmalıdır) kontrol edilir. Eğer yukarıdaki ve soldaki bloklardan hiçbiri hazır değilse, bu durumda sadece DC modu kullanılabilir ve kestirim değeri olarak önceden tanımlanmış bir değer atanır [4].

Bu kestirim modlarının sırası standartta belirtilmiştir. İstatistiksel olarak en çok tercih edilen kestirim moduna en düşük numara verilmiştir. Bunun nedeni ileride açıklanacaktır.



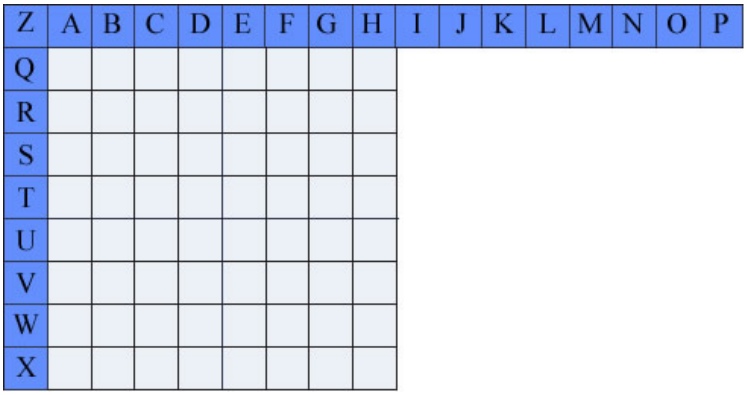
Şekil 2.6 : 4x4 parlaklık kestirim modları

2.3.2.2 8x8 parlaklık kestirim modları

Resmin orta seviyede detay bulunan kısımlarında başarılı kestirim yapabilen bu kestirim modları, 4x4 lük kestirim modlarına göre daha az maliyetlidir, çünkü seçilen kestirim modu daha fazla pikseli temsil etmektedir.

8x8 tipinde kodlanmakta olan bir makroblokta, kodlanan 8x8 lik blokların kestirimi için kullanılabilecek yine dokuz adet mod vardır. Bu modlar 4x4 parlaklık kestiriminde kullanılan modlarla aynı isimdelerdir ve kestirim yönleri aynıdır. Farkları ise en fazla on üç komşu piksel yerine 25 komşu piksel kullanması ve bir kestirimde 16 piksel yerine 64 pikselin değerinin hesaplanmasıdır. Şekil 2.7'de 8x8 lik bir blok için komşu piksellerin yerleşimi görülmektedir. A – X pikselleri komşu blokların daha önceden kodlanmış/kod çözülmüş piksellerini temsil etmektedir.

8x8 lik blokların kestiriminin 4x4 lük blokların kestiriminden önemli bir farkı da, komşu piksellerin kestirim öncesi filtrelenmesidir. Şekil 2.7'de A – X ve Z ile gösterilen, kodlanmış ve geri çözülmüş piksel değerlerine, ikinci dereceden bir binom filtresi uygulanır. Ardından kestirim işlemi gerçekleştirilir [6].



Şekil 2.7 : 8x8 parlaklık kestiriminde kullanılan pikseller

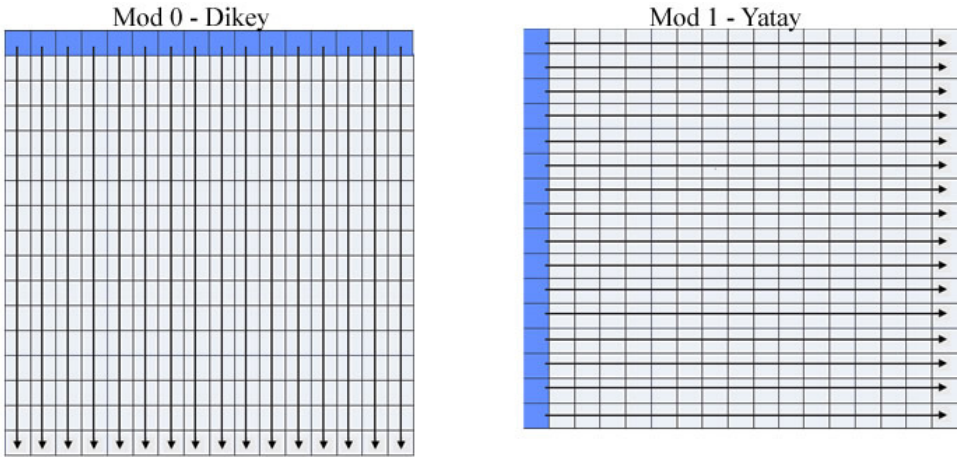
2.3.2.3 16x16 parlaklık kestirim modları

Resmin parlaklık bileşenin fazla değişiklik göstermediği alanlarında başarılı kestirim yapabilen bu modlar, en az maliyeti üretmektedir, çünkü tek bir seferde en fazla piksel buradaki kestirim modlarında kestirilmektedir.

Bir parlaklık bileşenine ait makroblok 16x16 tipinde kodlanıyorsa, kestirim yapabilmek için dört mod bulunmaktadır. Bu modlardan ilk ikisi Şekil 2.8'de gösterilmiştir.

Mod 3, DC kestirim modudur ve kestirim yapılmadan önce hangi komşu blokların kullanılabilir olduğuna bakılır. Eğer üstteki ve soldaki blokların tamamı kullanılabilir durumda ise yukarıdaki ve soldaki komşu piksellerin tamamının ortalaması alınarak kestirim oluşturulur. Kullanılabilir durumda olmayan bir komşu piksel var ise, ilgili pikselin bulunduğu yatay ya da düşey tüm piksellerden vazgeçilir ve sadece diğer taraftaki piksellerin ortalaması alınarak DC kestirim oluşturulur.

Mod 4, yüzeysel kestirim modu olarak isimlendirilmiştir ve standartta tanımlı bir fonksiyonun sol ve üst komşuluklarda bulunan piksellere oturtulmasıyla çalışır. Bu kestirim modunun kullanılabilmesi için üst ve soldaki tüm komşu piksellerin kullanılabilir durumda olması gerekmektedir. Resimlerin parlaklık seviyesi az miktarda değişen kısımlarında tercih edilmektedir.



Şekil 2.8 : 16x16 kestirim modlarından Mod 0 ve Mod 1

2.3.2.4 8x8 Renk (chroma) kestirim modları

Kodlanmakta olan bir resmin renk bileşenleri 8x8 lik makrobloklara bölünür. Resimde iki renk bileşeni olduğundan, bu iki renk bileşeni için ortak bir kestirim modu belirlenir. 8x8 lik renk makroblokları için de dört kestirim modu bulunmaktadır. Bu modlar 16x16 lık parlaklık kestirim modları ile aynı şekilde çalışmaktadır. Ancak kestirim yapılan alan 8x8 lik bir bölgedir. Dolayısıyla gerek duyulan komşu piksel sayısı daha azdır.

16x16 lık parlaklık kestirim modlarının sahip oldukları mod indeksleri, 8x8 lik renk kestirim modları için aynı değildir. DC mod ve Düşey modun indeksleri yer değiştirmiş durumdadır. Yani Mod 0 DC mod olarak kullanılırken, Mod 2 Düşey mod olarak kullanılmaktadır. Modların seçim olasılıklarıyla ilgili olarak belirlenmiş olan bu mod indekslerinin seçim detayları ilerde anlatılacaktır.

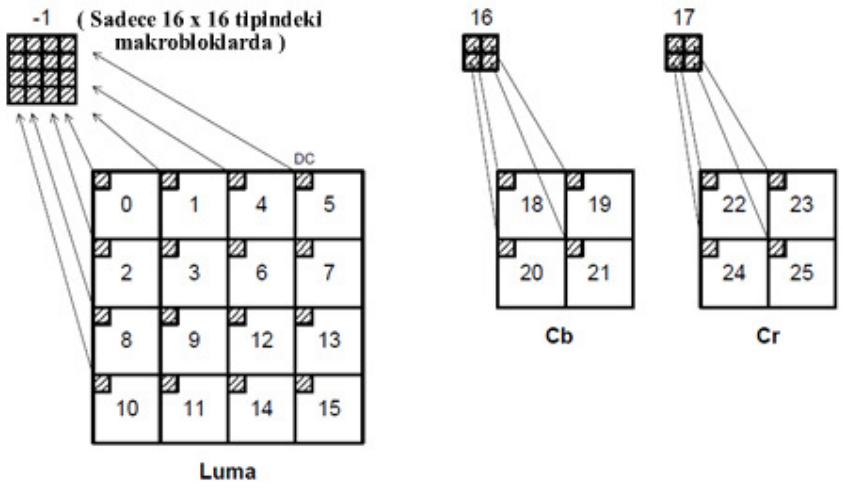
2.3.3 Dönüşüm ve kuantalama

Bir makroblok içerisindeki tüm pikseller için kestirim yapılmasının ardından, makrobloğun orijinal resimde karşılık gelen alanından kestirim verisi piksel piksel çıkartılır. Geriye kalan artık veriler, kodlanmadan önce blok tabanlı dönüşüme tabi tutulur, ardından dönüşüm katsayıları ölçeklenerek kuantalanır. H.264/AVC standardında birden fazla dönüşüm tanımlanmıştır ve tanımlanan bu adımlarda ölçekleme ve kuantalama birlikte gerçekleştirilir.

Kullanılacak dönüşüm, dönüşümü yapılacak artık bloğun tipine bağlıdır. Temel profilde üç tip dönüşüm tanımlanmıştır. Kodlanmakta olan blok tipine göre dönüşüm şu şekilde belirlenir:

- 16x16 tipinde kodlanmış makroblokların 4x4 lük DC blokları için Hadamard dönüşümü kullanılır.
- Renk makrobloklarının 2x2 lik DC blokları için ayrı bir dönüşüm kullanılır
- Geri kalan tüm 4x4 lük artık veri blokları için DCT benzeri özel bir dönüşüm kullanılır

Bir makroblok içerisinde yer alan tüm artık veriler Şekil 2.9'da görülen sırada kod çözücüye gönderilirler. Eğer makroblok 16x16 parlaklık tipinde ise, ilk önce 4x4 lük blokların DC bileşenlerini içeren -1 ile gösterilmiş blok gönderilir. Ardından 0 – 15 parlaklık artık blokları gönderilir. Blok 16 ve 17, Cb ve Cr renk bileşenlerinin DC değerlerinden oluşan 2x2 lik bloklardır. Son olarak 18 – 25 arasındaki renk artık blokları DC değerleri sıfırlanmış olarak gönderilirler.



Şekil 2.9 : Makroblok içerisindeki artık blokların taranma sıraları

Bir artık bloğun dönüşümü ve kuantalanması sürecinde gerçekleştirilenler şu şekilde özetlenebilir (Şekil 2.10):

Kodlayıcıda

1) Girdi olarak alınan 4x4 lük X artık bloğu alınır
2) İleri dönüşüm çekirdeğine (transform kernel) tabi tutulur :

$$W = C_f \; X \; C_f{}^T \tag{2.1}$$

(Renk DC ve 16x16 tipindeki parlaklık DC bileşenleri için sonrasında ileri dönüşüm gerçekleştirilir)

3) Dönüşüm sonrası ölçekleme ve kuantalama geçekleştirilir (PF ölçekleme katsayısı olarak tanımlanır):

$$Z = W \cdot \frac{PF}{Q_{adım} \cdot 2^{qbit}} \tag{2.2}$$

(Renk DC ve 16x16 parlaklık DC bileşenleri için farklılık içerir)

Kod çözücüde:

4) Renk DC ve 16x16 parlaklık DC bileşenleri için ters dönüşüm gerçekleştirilir
5) Geri ölçekleme ve dönüşüm öncesi ölçeklemesi yapılır:

$$W^{'} = Z \cdot Q_{adım} \cdot PF \cdot 64 \tag{2.3}$$

(Renk DC ve 16x16 parlaklık DC bileşenleri için farklılık içerir)

6) Ters dönüşüm çekirdeği:

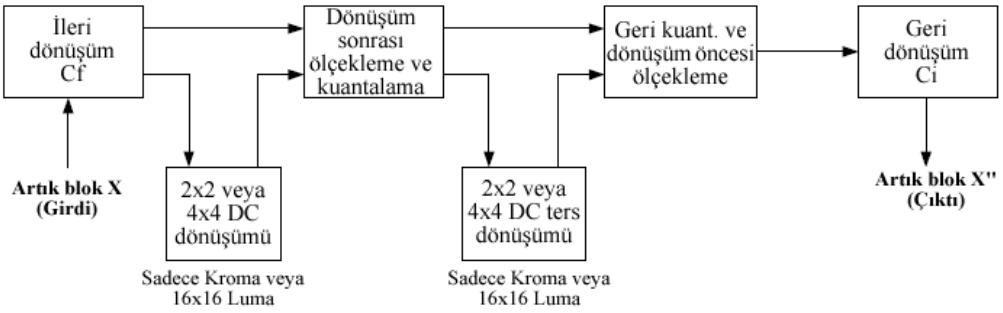
$$X' = C_i{}^T \; W' \; C_i \tag{2.4}$$

7) Ters dönüşüm sonrası ölçekleme:

$$X'' = \text{yuvarla}\left(\frac{X'}{64}\right) \tag{2.5}$$

8) Çıktı olarak 4x4 artık bloğu üretilir:

$$X''$$



Şekil 2.10 : Dönüşüm, kuantalama, geri kuantalama ve ters dönüşüm blok diyagramı [3]

2.3.3.1 4x4 artık bloklar için dönüşüm ve kuantalama

Şekil 2.9'da 0 – 15 ile belirtilen bloklar, çerçeve içi kestirimin ardından 4x4 lük dönüşüme tabi tutulurlar. Bu dönüşüm DCT tabanlıdır ancak bazı temel farklılıklar içerir:

1) Tam sayı tabanlı bir dönüşümdür, hassasiyet kaybı olmadan tamsayı aritmetiği ile gerçekleştirilebilir
2) Kodlayıcı ve kod çözücüde gerçekleştirilen dönüşüm ve ters dönüşümlerde kayıp yoktur.
3) Dönüşümün çekirdek kısmı sadece artırma ve kaydırma işlemleri ile gerçekleştirilebilir.
4) Kuantalayıcının içine entegre edilmiş "ölçekleme çarpımı" ile toplam çarpma işlemi sayısı azaltılmıştır.

Dönüşüm formülü olarak (2.6)'da belirtilen ifade kullanılır. DCT'den farklı olarak, trigonometrik fonksiyon cinsinden yazılmış ifadeler kesirli sayılara yakınsatılmaktadır ve bu sayılar ayrı bir ölçekleyici çarpım matrisine (E_f) aktarılmaktadır. Burada $\otimes$ sembolü, ($C_f \; X \; C_f{}^T$)'nin her elemanının ölçekleme matrisinde aynı pozisyonda bulunan, ölçekleme katsayısıyla çarpıldığını ifade etmektedir.

$$W = C_f \; X \; C_f{}^T \otimes Ef \qquad \textbf{(2.6)}$$

(2.1)'de verilen ifade, dönüşümün çekirdek kısmıdır. E_f matrisi kuantalama işleminde ele alınır.

Kuantalamada (2.7) kullanılır. *PF* burada dönüşüm sonrası ölçekleme katsayısı olarak tanımlanmaktadır. Aldığı değer i ve j nin değerine göre $a^2, \frac{b^2}{4}$ yada $\frac{ab}{2}$ olur.

$$Z_{ij} = \text{yuvarla}(W_{ij} \cdot \frac{PF}{Q_{adım}}) \quad \textbf{(2.7)}$$

Qadım parametresinin alabileceği 52 farklı değer vardır (Çizelge 2.2). Kuantalama parametresi olarak tanımlanan QP değeri, *Qadım* değerini doğrudan etkiler. QP = 0 için *Qadım* değer 0.625'dir ve QP'nin 1 artması, *Qadım* değerinin de %12,5 artmasına neden olur. QP değerinde 6 artış, *Qadım*'ın iki katına çıkmasına neden olacaktır [5].

Çizelge 2.2 : H.264/AVC'de tanımlanan kuantalama adımı değerleri [3]

QP	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**...**	**10**	**...**	**18**
Qadım	0.625	0.6875	0.8125	0.875	1	1.125	1.25		2		5
QP	**24**	**...**	**30**	**...**	**36**	**...**	**42**	**...**	**48**	**...**	**51**
Qadım	10		20		40		80		160		224

Kodlayıcıda, QP değerlerine karşı gelen bir *Qadım* tablosuna ihtiyaç yoktur çünkü, (2.8)'de görülen ifadede, kuantalama adımı başka bir parametrenin (MF: Multiplication Factor) içerisine gömülmüştür.

$$Z_{ij} = \text{yuvarla}(W_{ij} \cdot \frac{MF}{2^{qbit}}) \quad \textbf{(2.8)}$$

(2.9)'da verilmiş ifade (2.8)'deki bölmede kaç kez sağa kaydırma işlemi yapılacağını belirtir.

$$qbit = 15 + taban(\frac{QP}{6}) \quad \textbf{(2.9)}$$

2.3.3.2 4x4 parlaklık (luma) DC katsayıları için dönüşüm ve kuantalama

Makroblok 16x16 tipinde kodlanmış ise, 4x4 lük 16 bloğa bölünür ve 4x4 lük dönüşüm çekirdeği uygulanır. Dönüşüm sonrasında her bir blok için elde edilen DC değerleri bir araya getirilerek yeni bir 4x4 lük blok oluşturulur ve bu blok 4x4 lük Hadamard dönüşümüne tabi tutulur. Hadamard dönüşümünün detayları için [3]'e bakınız.

Çerçeve içi kodlanmış bloklarda enerjinin çoğu DC bileşenlerde toplanmıştır ve bu ek dönüşüm 4x4 lük parlaklık DC bileşenlerinin arasındaki korelasyondan faydalanmayı amaçlar.

2.3.3.3 2x2 Renk (chroma) DC katsayıları için dönüşüm ve kuantalama

Bir makroblok içerisindeki renk bileşenleri 4x4 lük 4 artık bloktan oluşur. Her bir 4x4 artık blok daha önce bahsedildiği gibi dönüşüme tabi tutulur. Bu 4x4 lük blokların DC bileşenleri tekrar 2x2 lik alt gruplara ayrılır ve diğerlerine benzer bir dönüşüm ve kuantalama işlemi gerçekleştirilir.

2.3.3.4 8x8 parlaklık dönüşümü

H.264/AVC standardına daha sonradan eklenmiş olan bu dönüşüm tipi, yüksek çözünürlüklü videolarda dokuların temsil edilebilmesi için gerekli olan daha büyük taban fonksiyonları kullanarak dönüşüm yapılabilmesini sağlar [6].

İki boyutlu bu dönüşüm, önce satırlarda yapılacak bir tek boyutlu dönüşüm ve ardından sütunlarda yapılacak bir tek boyutlu dönüşüm olarak ayrılabilecek yapıdadır. Tek boyutlu bu dönüşümler için kullanılabilecek dönüşüm matrisi T_{8x8}, Şekil 2.11'de verilmektedir.

8x8 lik dönüşümün seçilmesi, kodlayıcı tarafından makroblok bilgi başlığı içerisinde kullanılan bir belirteç ile kod çözücüye bildirilir. Çerçeve içi kodlanan makrobloklarda 8x8 lik dönüşümün seçilebilmesi için, makrobloğun 8x8 tipinde olması, dolayısıyla kestirimlerin de 8x8 modlarında yapılmış olması gerekmektedir.

$$T_{8x8} = \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & 10 & 6 & 3 & -3 & -6 & -10 & -12 \\ 8 & 4 & -4 & -8 & -8 & -4 & 4 & 8 \\ 10 & -3 & -12 & -6 & 6 & 12 & 3 & -10 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -12 & 3 & 10 & -10 & -3 & 12 & -6 \\ 4 & -8 & 8 & -4 & -4 & 8 & -8 & 4 \\ 3 & -6 & 10 & -12 & 12 & -10 & 6 & -3 \end{bmatrix}$$

Şekil 2.11 : 8x8 dönüşümde kullanılan normalize edilmemiş dönüşüm matrisi

2.3.4 Entropi kodlama ve CABAC

Tüm makroblokları kodlanmış bir dilimin, kod çözücüye iletilmesi için kestirilmiş, dönüşüm yapılmış ve gerekli sırada taranmış video verisinin ve birtakım sözdizimi elemanlarının kodlanması gerekmektedir. Dilime ait bu verilerin kodlanmasında ya değişken uzunluklu kodlar (variable length codes – VLCs), yada kontekst uyumlu ikili aritmetik kodlama (CABAC) kullanılır.

Entropi kodlama modu VLC olarak seçildiğimde, artık blok verileri CAVLC kullanılarak kodlanır. Diğer değişken uzunluklu sözdizimi elemanları ise Exp-Golomb kodları kullanılarak kodlanır. VLC, Huffman kodlama tabanlıdır.

Entropi kodlama modunun CABAC seçildiği durumda ise, artık blok verileri ve sözdizimi elemanları bir ikili aritmetik kodlama motoru kullanılarak kodlanır. CAVLC'ye oranla daha fazla kompleks olmasının yanı sıra CABAC, standart ve yüksek çözünürlük görüntülerde çerçeve başına gönderilen bit sayısını %10 ila %15 oranında azaltmaktadır [7].

Kodlanıp kod çözücüye gönderilmesi gereken parametrelere örnek Çizelge 2.3'te verilmiştir.

Çizelge 2.3 : Kodlanması gereken parametre örnekleri

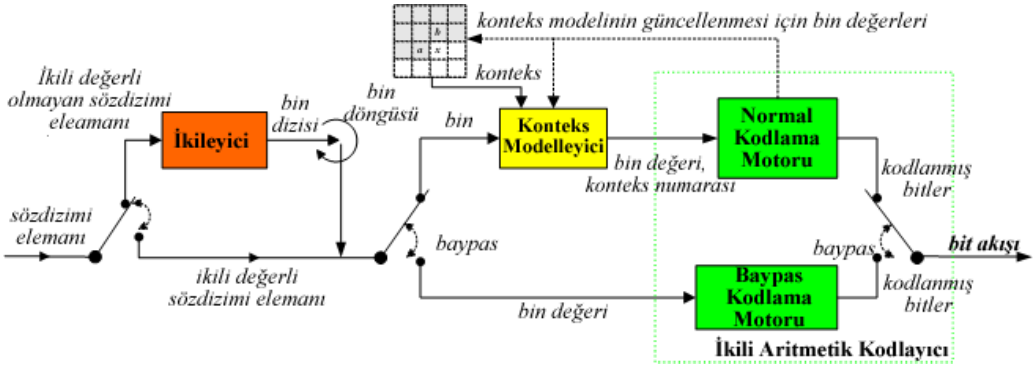
Parametre	Açıklaması
Sıra-, çerçeve- ve dilim-katmanı sözdizimi elemanları	Başlıklar ve parametreler
Makroblok tipi	Her bir makroblok için gönderilir. Kestirim tipini belirtir. (4x4, 8x8 yada 16x16)
Kodlanmış blok bilgisi	Bir makroblok içerisindeki hangi blokların kodlanmış katsayılar içerdiğini belirtir.
Kuantalama parametresi	Bir önceki kuantalama parametresine göre fark değeri gönderilir
Artık blok verisi	8x8, 4x4 ya da 2x2 lik dönüşüm sonucu elde edilen artık veri katsayılarıdır.
Çerçeve içi kestirim modu	Makroblok içerisindeki her blok için gönderilir. Bloğun kestirim modunu belirtir.

2.3.4.1 Kontekst Uyumlu Ikili Aritmek Kodlama (CABAC)

CABAC'ın iyi bir sıkıştırma performansı sağlamasının üç temel nedeni vardır:

1) Her bir sözdizimi elemanı için, elemanın konteksine göre olasılık modelini belirler
2) Olasılık dağılımlarını yerel istatistiklere dayandırır
3) Değişken uzunluklu kodlama yerine aritmetik kodlama kullanır

CABAC kodlamasının temel elemanları ikilileme, kontekst modelleme, ve ikili aritmetik kodlamadır. Bu fonksiyonel elemanlar Şekil 2.12'deki CABAC kodlama blok diyagramında gösterilmiştir.



Şekil 2.12 : CABAC kodlama blok diyagramı

Bir veri sembolünün kodlanmasında şu adımlar gerçekleştirilir:

1) **İkilileme:** CABAC ikili aritmetik kodlayıcı kullandığından dolayı, sadece 0 ve 1 olarak ifade edilen simgelere göre karar verilmektedir. İkili değere sahip olmayan bir sembol, (örneğin dönüşüm katsayıları) aritmetik kodlama öncesinde ikili bir koda dönüştürülür. 2, 3 ve 4. adımlar her bir bit (yada "bin") için tekrarlanır
2) **Kodlama modu ve kontekst modelinin seçimi:** Her sembolün ikili hale dönüştürülmesinden sonra yapılacak işlem, seçilecek kodlama moduna bağlıdır. Baypas kodlama modu, düzgün dağılımlı olduğu varsayılan az önemli binler içindir ve seçilmesi durumunda normal aritmetik kodlama işlemi atlanır. Normal kodlama modunda ise kodlama için kullanılacak kontekst belirlenir.

   Kontekst modeli, ikililenen bir sembolün bitlerinin kodlanmasında

kullanılan, önceden kodlanmış veri sembollerinin istatistiki durumuna dayanarak seçilen olasılık modelidir. Kontekst modeli her bir bitin 0 yada 1 olması ihtimalini tutar.

Sık kodlanan semboller için, birden fazla kontekst mevcuttur ve böylece semboller arası tekrarlamalardan yararlanılır.

3) **Aritmetik kodlama:** Aritmetik kodlayıcı her bir biti seçilen olasılık modeline göre kodlar. Her bir bit için yalnızca iki ihtimal vardır (0 yada 1).

4) **Olasılık güncellemesi:** Seçilen aritmetik model, kodlanan değere göre güncellenir. Eğer kodlanan değer 1 ise, 1'in frekans değeri artırılır.

CABAC'da dört farklı kontekst modeli tasarımı kullanılmıştır. İlki, kodlanmakta olan sözdizimi elemanının daha önce kodlanmış, en fazla iki komşusundaki karşılığından yararlanılmasıdır [7]. Komşuluğun tanımı sözdizimi elemanına göre değişmektedir. Örnek olarak, bir bloğun kodlanması sırasında, bloğun bir sembolünün kontekstinin belirlenmesi için üst ve sol bloklardaki aynı sembolün aldığı değerlere bakılması verilebilir.

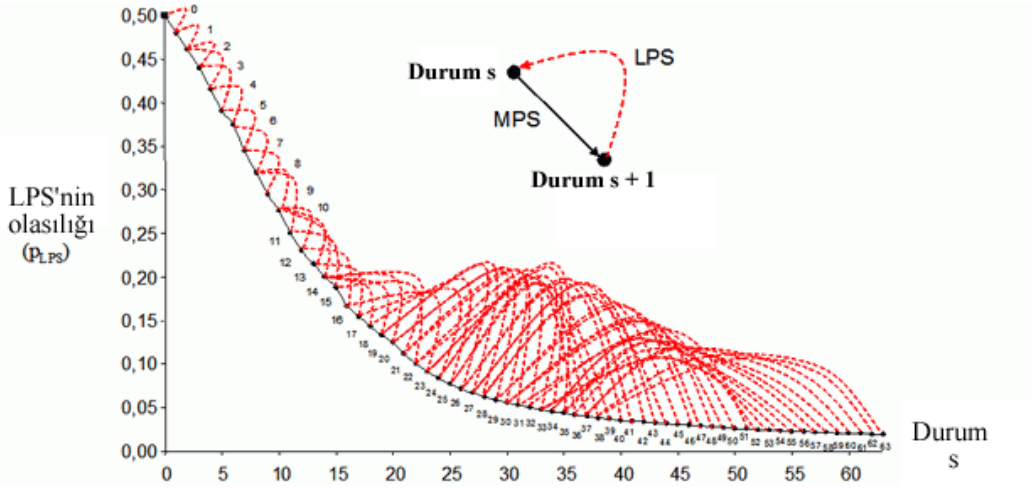
İkinci kontekst modeli tipi, sadece makroblok ve alt makroblok tiplerini tanımlayan sözdizimi elemanları için kullanılmaktadır. Bu kontekst modelinde, önceden kodlanmış bitlerin değerleri, bu bitlerden sonra gelen, kodlanacak olan bitin kontekst modelini belirlemekte kullanılmaktadır [7].

Üçüncü ve dördüncü kontekst modelleri sadece artık blok verileri için kullanılır. Bu tipteki kontekstler kodlanan bloğun tipine dayanarak belirlenirler. Ayrıca üçüncü tipteki kontekst modelleri, geçmişte kodlanan verilere değil, verinin (artık pikselin) taranma pozisyonuna dayanarak kontekstin seçilmesini sağlarlar [7].

Her bir bin, aritmetik kodlama motoruna ya normal ya da baypas kodlama modunda girer. Baypas kodlama modunda, motor gelen bitin değerine göre dallanarak durumunu belirler. Normal kodlama durumunda ise, bin değerinin kodlanmasında, seçilmiş kontekstin durumu bağlayıcıdır.

CABAC'da olasılık kestirimi, sonlu durum makinesi kullanan bir tablo tabanlı kestiriciyle yapılır. Her bir olasılık modeli 128 farklı durumdan birinde olur ve bu

durumlara karşılık gelen olasılıklar [0.01875 , 0.98125] arasındadır. En düşük olasılıklı sembol (least probable symbol – LPS) ve en yüksek olasılıklı sembol (most probable symbol - MPS) değerleri aritmetik kodlayıcıda atılacak adımın uzunluğunun belirlenmesini sağlar.



Şekil 2.13 : LPS olasılık değerleri ve olasılık kestiriminin güncellenmesi için geçiş kuralları [7]

Durum makinesi geçişleri ve karşılık gelen LPS olasılık değerleri Şekil 2.13'de görülmektedir. Her bir olasılık modelinin ilk durumunu belirlemek için kuantalama parametresine bağlı bir ilklendirme fonksiyonu kullanılmaktadır. Kuantalama parametresi (QP) her dilim başında gönderildiğinden, her dilimde kontekst modelleri tekrar ilklendirilir [7].

2.3.5 Bit oranı – bozulma optimizasyonu

Bir H.264/AVC kodlayıcısı, kodlanmakta olan bir dilimdeki her makroblok için, kullanılabilecek en iyi makroblok tipini bulmaya çalışır. Bunun için de her bir makroblok tipinde, makroblok içerisindeki her bloğun en iyi kestirim modunun belirlenmesi gerekmektedir.

Bir bloğun en iyi kestirim modunu belirlemek için, bloğun her bir moddadaki kestirim maliyeti bulunur. Kestirim maliyeti, bloğun bozulma miktarı ile bloğun kodlanması için gerekli olan bit sayısının bir birleşimidir.

Maliyet hesabı için Lagrange tipi fonksiyonların kullanımı video kodlamada yaygın olarak tercih edilmektedir. H.264/AVC standardı da blokların kestiriminde maliyet hesabı için, Lagrange optimizasyon metodunun kullanılmasını işaret etmektedir (şart

koşmamaktadır, çünkü kullanılacak yöntem kodlayıcının inisiyatifindedir). Bunun iki temel nedeni vardır. Birinci neden Lagrange metotlarının verimli olmasıdır. İkinci nedeni ise basit bir yöntem olmasıdır [2].

Bir blokun kodlanması aşamasında bloğun bozulumunu ve bloğun iletimi için gerekli bit sayısını (kodlayıcı çıktısındaki bit oranı) etkileyen en önemli parametre kuantalayıcı parametresi QP olduğundan dolayı, bu parametreye dayanan bir Lagrange fonksiyonu kullanılmalıdır.

Bozulmayı D, bit oranını R , Lagrange parametresini λ , maliyeti de J ile gösterecek olursak, söz konusu Lagrange fonksiyonu (2.10)'da görüldüğü gibi kullanılmaktadır.

$$J_{BLOK} = D_{BLOK} + \lambda_{MOD} \cdot R_{BLOK} \quad \textbf{(2.10)}$$

λ_{MOD}'un değerinin QP değeri ile nasıl değiştiği gözlemlenerek, her bir kestirim modu için (çerçeve içi ya da çerçeveler arası) QP'ye bağlı λ_{MOD} değeri oluşturulmuştur [8]. H.264/AVC için de ilgili ölçümler yapılmış ve λ_{MOD} değerinin QP'ye bağlı olarak (2.11)'deki gibi kullanılmasına karar verilmiştir [4].

$$\lambda_{MOD} = 0.85 \cdot 2^{(QP-12)/3} \quad \textbf{(2.11)}$$

D_{BLOK} ve R_{BLOK}'un hesaplanmasında iki alternatif vardır. Birincisi bloğun tamamen kodlanması ve gerekli bit oranı ile bozulmanın hesaplanmasıdır. RDO_{ON} adı verilen [9] bu durumda D_{BLOK}, orijinal kodlanmamış çerçevedeki ilgili blokla, kodlanmış ve kod çözülerek yeniden oluşturulmuş blok arasındaki karesel farkların toplamı (SSE) olarak hesaplanmaktadır. R_{BLOK} ise, kestirimin ardından dönüşüm yapılmış artık bloğun entropi kodlaması sonucu oluşan çıkış bit oranı ile blok için gönderilen başlık bilgilerinin bit oranının toplamından oluşmaktadır.

RDO_{OFF} olarak adlandırılan [9] ikinci alternatifte ise, D_{BLOK} orijinal kodlanmamış çerçevedeki ilgili blokla, kestirim bloğu arasındaki mutlak farkların toplamı (SAD) olarak hesaplanır. R_{BLOK} da, bloğun kestirim modu bilgisi, varsa hareket vektörleri ve diğer ek bilgilerin toplam bit sayısı olarak kullanılır. Bu alternatif durumda, entropi kodlama gerekmeden en iyi blok kestirim modu seçilebilmektedir,

dolayısıyla diğer duruma göre karmaşıklık çok daha düşüktür. Ancak kodlama performansı da aynı ölçüde daha kötüdür.

2.4 Çerçeve İçi Kodlama için Kodlayıcı Adımları

Bir H.264/AVC kodlayıcısını sadece çerçeve içi kodlama yapacak şekilde çalıştırmak, kodlayıcıdaki çerçeveler arası kestirim için harcanan hesaplama gücünü ve bit oranı-bozulma optimizasyonundaki adımları büyük ölçüde azaltmaktadır. Sadece çerçeveler arası dilimlerde kullanılabilecek makroblok tipleri kullanılmayacak ve her makroblok için sadece çerçeve içi kestirim modları denenecektir.

Çerçeve başında tek bir dilim kullanan, entropi kodlayıcı olarak CABAC ve RDO_{ON} kullanmak üzere ayarlanmış bir kodlayıcıda, tüm bir çerçevenin kodlanması için aşağıdaki adımlar izlenmektedir:

1) Çerçeve'nin parlaklık bileşeni için tüm çerçeve 16x16 lık makrobloklara bölünür. Renk bileşenleri ise 8x8 lik makrobloklara bölünür. Bu bloklar bir nevi satır taramayla (makroblok yüksekliğiyle aynı miktardaki bir satır genişliğiyle) sıradan kodlamaya tabi tutulur.

2) Parlaklık makroblokları için, üç farklı makroblok tipinden hangisinin maliyetinin daha düşük olduğunu hesaplamak üzere, her bir makroblok tipinde kestirimi yapılacak alt bloklar oluşturulur

3) Parlaklık alt blokları ve renk makroblokları için tüm kestirim modları denenir. Her bir kestirim modunda komşu piksellerden kestirim yapılmasının yanı sıra, üst ve sol komşulukta bulunan blokların kestirim modları da kullanılarak kodlanmakta olan blok için en en olası kestirim modu bulunur. En olası kestirim modunun belirlenmesinin amacı, bu modun seçilmesi durumunda daha az bit harcanarak bloğun kestirim modunun kod çözücüye bildirilmesidir. Bu uygulama için, kod çözücünün de her bir bloğun kodunu çözmeden önce komşu blokların kestirim moduna bakarak en olası kestirim modunu belirlemesi gerekir.

   Örneğin 9 kestirim modu bulunan 4x4 tipi bir bloğun kestirim modunu belirtmek için 1+3 bit kullanılır. En olası modun seçilmesi durumunda, tek bir bitle mod belirtilmiş olurken, seçilmemesi durumunda 9 kestirim

modundan bu mod çıkartılarak geri kalan 8 moddan hangisi seçilmişse 3 bitle kodlanarak kod çözcüye gönderilir.

Her blok için bit oranı – bozulma optimizasyonu yapılırken, en olası modun, mod bilgisi sözdizimi elemanının kodlanmasından dolayı yaklaşık 3 bit avantajı olur. Yaklaşıklığın sebebi, CABAC kodlama motorunun o anki durumu nedeniyle bit maliyetlerinin değişmiş durumda olabilmesidir.

4) Bir makroblok için tüm makroblok tipleri denendikten sonra en düşük maliyetli olan makroblok tipi seçilir ve bit oranı – bozulma optimizasyonu yapılırken entropi kodlaması yapılmış alt bloklar için tekrar bir kodlamaya ihtiyaç kalmadan, sadece makroblok başlığı, blok içerisindeki bloklara göre düzenlenerek makrobloğun kodlanması tamamlanmış olur. Ardından yine bit oranı – bozulma optimizasyonu yapılırken elde edilmiş geri oluşturulmuş piksel değerleri, sonraki makroblokların kestiriminde kullanılmak üzere tamponlanır.

5) Çerçevedeki tüm makroblokların kodlanmasının ardından, entropi kodlanmış makroblok verileri NAL katmanına aktarılarak taşıyıcı dosya ya da hedef uygulamaya uygun bit akışı formatında paketlenir.

3. ŞABLON EŞLEME İLE ÇERÇEVE İÇİ KODLAMA

Çerçeve içi kodlama, çerçeve içerisindeki birbirine yakın olan piksellerin, yüksek korelasyona sahip olması özelliğinden faydalanan ve çerçeve içerisindeki artıklığı azaltarak kodlama maliyetini düşüren bir yöntemdir. Blok kodlayıcılar için bu yöntemin kullanılması ilk defa [10]'da önerilmiştir. Kodlanacak olan bloğun çevresindeki piksellerden çeşitli yönlerde kestirim yapılarak, blok içerisindeki örüntünün yönüne en çok uyan kestirim tipi seçilmekte ve kodlanması gereken verinin miktarı azaltılabilmektedir. Bunun yanı sıra düzgün dağılımlı bloklar için de DC kestirim modu kullanılabilmektedir. Bu yöntemin maliyeti oldukça düşüktür.

Ancak yönsel şekillerin ya da düzgün dağılımlı bölgelerin az olduğu, karmaşık doku içeren blokların kodlanmasında mevcut yönlü kestirim modları yeterli olmamaktadır. Birbirine benzeyen blokların ya da dokuların olduğu çerçevelerin kodlanmasında, doku kestirimi yapabilecek yöntemlerin kullanılması kodlama performansını artırabilir.

Bu çalışmada, çerçeve içi kodlama etkinliğini artırabilecek yöntemler araştırılmaktadır. H.264/AVC'de çerçeve içi kestirim modlarına ilave edilebilecek doku kestirim yöntemleri araştırmanın temel parçasını oluşturmaktadır. Bu yöntemlerin etkin kullanımını sağlamak için de H.264/AVC kodlama / kod çözme bloklarında gerekli değişiklikler ve optimizasyonlar belirlenerek kodlamaya katkısı gözlemlenecektir.

3.1 Doku Sentezleme

Doku sentezleme, verilen bir doku örneğinden, insan gözüyle bakıldığında aynı stokastik süreç tarafından üretilmiş gibi algılanan yeni bir görüntü oluşturmaktır [11]. Doku sentezlemenin kullanım alanları arasında belirli bir bölgesi bozulmuş resimlerin onarılması, iletiminde kayıp yaşanmış video veya fotoğrafların tamamlanması, istenmeyen bir objenin resimden çıkartılması gibi uygulamalar

vardır. Doku analizi ve sentezlemenin görüntü ve video sıkıştırma alanında kullanımı da üzerinde çalışılan konular arasındadır.

Doku sentezleme işleminde ilk olarak kullanılacak doku alanının analizi yapılarak dokunun oluşumunda yer almış olan stokastik süreç saptanmaya çalışır. Elde edilen stokastik model doku sentezleme işlemine girdi olarak alınır ve sentezleyici gerekli olan resmi ya da videoyu sentezler.

Bir doku sentezleyici tasarlanacağı zaman nasıl çalışacağına karar verilmesi gereken iki fonksiyon vardır:

1) Modelleme: Verilen bir sonlu doku örneğinin stokastik modelinin nasıl saptanacağı belirlenmelidir.
2) Örnekleme: Verilen modelden yeni dokuların üretilmesini sağlayacak isabetli bir örnekleme yöntemi seçilmelidir.

Modellemenin ne kadar doğru seçildiği sentezlenen dokunun görsel kalitesini belirlerken, örneklemenin etkinliği doku sentezlemenin hesaplama maliyetinde etkili olmaktadır [11].

Doku analizi ve sentezlemesi için önerilmiş pek çok yöntem mevcuttur. Bunlardan bazıları şu şekilde özetlenebilir:

Fiziksel simülasyon ile: Bazı yüzey dokularını simülasyonla sentezlemek mümkündür. Deniz, hava, toprak yada insan derisi gibi önceden tanımlanmış ortak fiziksel özellikler içeren yüzeylerin sentezinde, önce sentezlenecek bölgenin fiziksel özelliği tespit edilir sonrasına bu özelliklere dayanarak doldurulacak alan için doku sentezi gerçekleştirilir. Farklı dokular genelde farklı fiziksel özelliklere sahip olduklarından dolayı, bu yöntem sınırlı sayıda doku tipinde kullanılabilmektedir [11].

Markov Rastlantısal Alanları ve Gibbs örneklemesi ile: Dokuların modellenmesinde Markov Rastsal Alanları kullanılarak, dokular olasılık örneklemesi ile oluşturulabilmektedir. Pek çok doku sınıfı için Markov Rastlantısal Alanları (Markov Random Fields – MRF) iyi bir yakınsama sağlamaktadır ve alınan sonuç çoğunlukla yeterli düzeydedir. Ancak bu yöntemin çalışması küçük doku sentezleme işlemleri için bile saatler mertebesinde uzun sürmektedir [11].

Özellik eşleme ile: Verilen bir dokuyu bir özellikler seti olarak tanımlayıp, sentezlenecek yeni doku için bu özelliklerin örnek bir doku üzerinde eşlenmesine

dayanan bir yöntemdir. Bu yöntemi kullanan algoritmalar, MRF yöntemini kullanan yöntemlerden daha verimlidir. Ancak bu yöntemi kullanan uygulamalarda, karmaşık yapılar içeren dokularda, kendini tekrar edemeyen dokularda ve dokuların küçük ölçekli olduğu durumlarda çeşitli sorunlarla karşılaşılmaktadır [11].

Doku sentezleme yönteminin eksik resim parçalarını tamamlayabilmesi özelliği, yöntemin çerçeve içi kodlamada kullanılması fikrini beraberinde getirmektedir. Kodlanmakta olan bir bloğun kestirim modu olarak doku sentezleme yöntemi seçilerek, daha doğru kestirim sonuçları elde etmek mümkün olacaktır. Kestirimi daha doğrulukla yapılmış bir blok için gönderilecek artık blok verisi de azalacağından, doku içeren çerçevelerde kodlayıcı daha yüksek sıkıştırma oranı elde edecektir.

Ancak video verisi işlenirken her bir çerçeve için, çözünürlükle orantılı olarak blok sayısı, dolayısıyla yapılan kestirim işlemi yüksek miktarlara ulaştığı için, kullanılacak doku sentezleyicinin maliyetinin düşük olması gerekmektedir.

3.2 Şablon Eşleme ile Doku Sentezleme

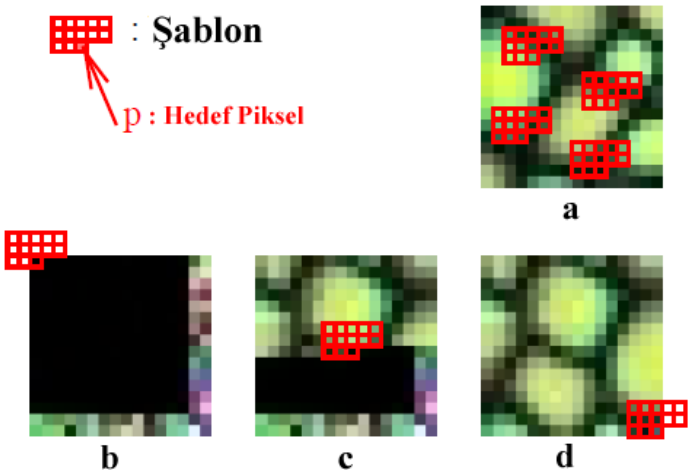
Wei ve Levoy, verilen bir resimden şablon eşleme ile başka bir resim sentezlemek için bir metot önermişlerdir [11]. Bu doku sentezleme yönteminde Markov Rastlantısal Alanları kullanılmıştır. MRF yöntemleri bir dokuyu yerel ve durağan rastlantısal süreçlerin gerçeklenmesi olarak modeller. Doku resmindeki her bir piksel, komşu piksellerin bir kümesi ile karakterize edilir. Yerel benzerlik, her bir pikselin komşu piksellerin bir kümesinden kestirilebildiği ve resmin geri kalanından bağımsızlık olarak tanımlanmıştır. Bu yöntemde, verilen bir örnek doku parçasına yerel benzerlik taşıyan yeni bir doku sentezlenmektedir.

MRF lerin hesaplama maliyetini düşürmek için ise olasılık dağılımı hesaplamaları ve örnekleme adımları yönteme dahil edilmemiştir. Bu nedenle yöntem diğer doku sentezleme yöntemlerine oranla daha hızlı çalışabilmektedir.

Yöntem, girdi olarak bir doku örneği ve de üzerinde sentezleme gerçekleştirilecek bölgeyi alır. Amaç, sentezleme gerçekleştirilecek bölgenin verilen doku örneğine benzer bir dokuya sahip hale getirilmesidir.

Üzerinde işlem yapılan bölgedeki bir (hedef) pikselin yeni değerini belirlemek için, tanımlanmış bir şekle sahip komşuluk bölgesi (şablon), doku örneğindeki tüm olası

komşuluklarla karşılaştırılır. Doku örneğindeki aranan şablona en çok benzeyen komşuluk bölgesine sahip pikselin değeri, sentezlenmekte olan (hedef) pikselin değeri olarak tayin edilmiş olur. Şekil 3.1'de şablon ve hedef pikselin yeri görülmektedir.



Şekil 3.1 : Şablon eşleme ile doku sentezleme. a sentezlemede kullanılan dokuyu göstermektedir. b, ilk pikselin, c orta pikselin, d son pikselin sentezlenmesini göstermektedir.

Doku üzerindeki en benzer komşuluk aranırken, şablon ile komşuluğun benzerliği karesel farkların toplamı yöntemi ile hesaplanır. Amaç, yeni atanmış p değerinin doku ile sentezlenen bölge arasındaki yerel benzerliği en çok artıran değere sahip olmasıdır. Bu süreç, hedef bölgedeki tüm piksellerin sentezlenmesi tamamlanana kadar devam eder.

Bu yöntemde doku sentezlemenin başarımı, kullanılan şablonla doğrudan ilişkilidir. Eğer sentezlenmekte olan bölgede önceden bir değer varsa bile (çizik bir resmin onarılması ya da bir gürültü işareti yerine doku sentezlemek gibi) bu değerler sınır bölgeler haricinde kullanılmamakta ve hedef pikselin değeri, komşuluğunda bulunan pikseller şablon kabul edilerek belirlenmektedir. Şablondaki pikseller sınırlar dışında ise (Şekil 3.1.b gibi), ya sınıra yakın pikseller aynalanır yada sınır pikselleri sınır dışında doğru tekrar edilir. Şablonun, doku içerisindeki düzgün şekillerden en büyüğünün boyunda olması tercih edilmelidir [11].

Şablon şekli sentezleme kalitesini belirleyen bir diğer etkendir. Şablon şeklinin nedensel olması gerekmektedir. Pikselin şablon olarak kullanılacak komşuluklarındaki pikseller, sentezleme sonucu oluşturulmuş olmalıdır. Doku

üzerinde şablona en çok benzeyen piksel grubu aranırken, dokunun parçası olmayan piksellerin de karşılaştırmaya dahil edilmesi, doku sentezlemenin başarımını düşürür [11].

Bu yöntemle doku sentezlerken şu akış izlenir:

1 Doku sentezlenecek bölge belirlenir.

2 Bölgedeki her bir piksel (hedef) için:

3 Hedef pikselin şablon bölgesindeki pikseller belirlenir

4 Kullanılan dokudaki her bir muhtemel şablon için:

5 Kaynak şablonu oluşturulur

6 Kaynak şablon ile hedef piksel şablonu arasındaki fark ölçülür

7 Fark o ana kadarki en küçük fark ise kaydedilir

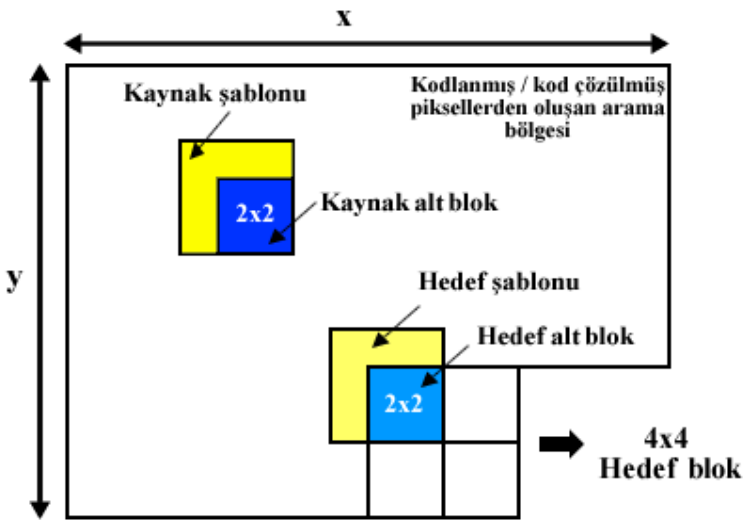
8 En küçük farka sahip kaynak şablona sahip olan doku pikseli hedef piksele atanır

3.3 Şablon Eşleme ile Çerçeve İçi Kestirim

Şablon eşleme ile doku sentezleme yöntemi, bir resmin eksik parçalarını tamamlamak için de kullanılabilir. Ancak bir resim üzerinde referans bir doku resmi olmadan doku sentezleyebilmek için, resim üzerindeki bir bölgeyi doku referansı olarak seçmek gerekmektedir.

Çerçeve içi kodlamada ise, şablon eşleme ile çerçeve içi kestirim yapmak, kodlanmakta olan bloğu hedef bölge seçip, etrafındaki belirli büyüklükteki bir arama bölgesini de doku bölgesi seçerek mümkün olmaktadır. Bu yöntemin ilk uygulaması Tan, Boon ve Suzuki [12] tarafından yapılmıştır.

Bu uygulamada kestirimi yapılacak 4x4 lük bir blok 2x2 lik 4 parçaya bölünmekte ve kestirim 2x2 lik alt bloklar üzerinde yapılmaktadır (Şekil 3.2). Alt blokların çevresindeki pikseller şablon olarak seçilmektedir. Her bir 4x4 lük bloğun çevresindeki önceden kodlanmış pikseller de arama bölgesi olarak kullanılmaktadır. Alt bloğa ait hedef şablonu, arama bölgesi içerisinde aranarak, en çok benzeyen kaynak şablonu bulunur. Kaynak şablonuna sahip kaynak alt blok, hedef alt bloğun kestirimini oluşturur. Tüm alt bloklar için bu işlem tekrarlanır.



Şekil 3.2 : Şablon eşleme ile çerçeve içi kestirimde kullanılan bloklar

3.3.1 H.264/AVC kodlayıcı / kod çözücü üzerinde şablon eşleme

Bir çerçeve içi kestirim modu olarak şablon eşleme kullanmak için, H.264/AVC standardında tanımlanmış olan çerçeve içi kodlama mekanizmasında bazı değişiklikler yapılması gerekir. 4x4 lük bloklar için tanımlanmış 9 kestirim modu bulunmaktadır. Bir blok için, bit oranı – bozulma optimizasyonu çerçevesinde en düşük maliyetli kestirim modu seçilir. Kestirim modlarının seçilme oranları, modun kodlanmakta olan video karakteristiğine ne kadar uygun kestirim yapabildiğiyle orantılıdır. Dolayısıyla yeni bir kestirim modu eklendiğinde bu modun kodlama verimliliğine katkısı, diğer modlara oranla orijinal bloğa ne kadar yakın kestirim yapabildiği (bozulma) ve artık blokta ne kadar az katsayı kalmasını sağladığıyla (bit oranını azaltmasıyla) orantılıdır.

Bu çalışmada şablon eşleme metodu, yeni bir çerçeve içi kestirim modu olarak mevcut kestirim modlarına ilave edilmiştir. Bu nedenle şablon eşlemenin kodlayıcı verimliliğine etkisi, ilave bir mod olarak incelenecektir.

3.3.1.1 Kodlayıcı – kod çözücü senkronizasyonu

Standartta tanımlı çerçeve içi kestirim yönteminde, kodlayıcı kod çözücüye her bir blok için hangi kestirim modunu tercih ettiğini belirtir. Kod çözücü de ilgili bloğun kod çözülme sırası geldiğinde belirtilen kestirim moduna göre kodlayıcıda yapılan kestirim işleminin aynısını tekrar ederek kestirim bloğunu elde eder. Kullanılan

kestirim moduna göre yapılacak kestirim işlemi standartta tanımlı olduğundan, kod çözücü diğer alternatif modlarla ilgilenmeden bloğun kestirimini yapmaktadır.

Şablon eşleme ile kestirim yapılırken, bir arama bölgesi içerisinde en uygun kaynak şablonu aranmaktadır. Seçilen şablonun alt bloğu kestirimde kullanıldığından, seçilen alt bloğun koordinatlarının kod çözücüye bildirilmesiyle, diğer kestirim modlarında olduğu gibi kod çözücünün alt blok kestirimlerini hemen oluşturabilmesi mümkündür. Ancak böyle bir durumda her bir alt blok için kaynak alt bloğu işaret eden bir vektör ya da pozisyon belirteci kullanılması gerekir. Bu durumda blok başına kod çözücüye gönderilen veri miktarında artış olacaktır.

Kodlayıcı bir bloğu kodlarken ve kod çözücü aynı blok için kod çözme işlemi yaparken, her ikisi de daha önceden kodlanmış piksellerin kod çözülmüş değerlerine sahiptir. Ayrıca şablon eşleme yönteminde alt blok için kestirim bloğu seçilirken orijinal resim (yada video çerçevesi) içerisindeki ilgili bloğun orijinal piksel değerlerine bakılmaz. Şablon seçimine karar veren etken, önceden kodlanmış piksellerden oluşan hedef alt blok şablonu ile yine önceden kodlanmış kaynak alt blok şablonunun piksel bazında yakın değerlere sahip olmasıdır.

Şablon eşlemede seçilen kestirim alt bloklarına dair bir bilgi kod çözcüye gönderilmez. Eğer bir blok için şablon eşleme ile kestirim yapılması tercih edilmişse, kod çözücü, ilgili blok için kodlayıcı hangi alt blokları seçmişse, aynı alt blokları seçmelidir. Bunun için de kodlayıcıda çalışan algoritmaların aynısı ve kodlayıcıda kullanılan referans piksel değerlerinin aynısı kod çözücüde kullanılarak aynı kestirim değerlerinin elde edilmesi sağlanır. Kod çözücü karmaşıklığını artırıcı bir etken olmakla birlikte, şablon eşlemede fazladan veri gönderilmeden büyük bir arama alanını kullanarak kestirim yapılabilmesi mümkün olur.

3.3.1.2 Çerçeve içi kestirim modu bilgisinin gönderilmesi

H.264/AVC'de kodlanan blokların kestirim modları genelde komşu bloklarla bağıntılıdır [3]. Bu nedenle bir bloğa ait kestirim modu bilgisi, bloğun üst ve sol komşuluklarında bulunan blokların kestirim modları da dikkate alınarak kodlanır. Bu iki komşu bloğun kestirim modu bilgileri karşılaştırılarak, kodlanmakta olan bloğa ait "en olası mod" (most probable mode) belirlenir. Kod çözücü de önceden kodunu çözdüğü her bir bloğun kestirim modu bilgisine sahip olduğundan, o da kodunu çözdüğü her bloğun en olası modunu aynı şekilde belirler.

Bir blok kodlanırken kestirim modu olarak bu "en olası mod" seçilmişse, kodlayıcı blok için sadece 1 bit kullanarak bu bilgiyi kod çözücüye bildirir. Eğer başka bir kestirim modu seçilmişse, o zaman diğer 8 moddan seçilmiş olan fazladan bir sözdizimi elemanı kullanılarak kod çözücüye bildirilir. Standartta indeksleri 0 ile 7 arasında olan tanımlanmış kestirim modları, bu şekilde ya 1 bit kullanılarak ya da en olası moddan büyük mod indekslerine sahip modların indeksleri bir azaltılıp, mod indeksleri 0 – 7 arasına alınıp 3 bitle kullanılarak kodlanır. Bu durum Çizelge 3.1'de özetlenmiştir.

Çizelge 3.1 : Kestirim modu sözdizimi elemanlarının kodlanması

Durum (n = seçilen mod, x = en olası mod)	**En Olası Mod Biti**	**Kestirim Modu Sözdizimi Elemanı** ($c \in [0,7]$)
$n = x$	1	-
$n \neq x$	0	$n < x \quad , \quad c = n$ $n > x \quad , \quad c = n-1$

Kodlanmakta olan bloğun en olası modu, bloğun üst ve sol komşuluklarında yer alan blokların kestirim modu indekslerinden küçük olanıdır. Eğer bu iki komşu bloktan herhangi biri yoksa o zaman en olası mod olarak DC kestirim modunun seçilmesi standartta belirtilmiştir. Komşu blokların kestirim modlarından indeksi küçük olan diğer moda göre daha avantajlı olduğundan, kodlayıcının küçük mod indeksine sahip kestirim modlarını seçme eğilimi biraz daha fazladır. Yeni bir kestirim modu eklenirken bu durum da göz önünde bulundurulmalıdır.

Kestirim modlarına bir yenisini eklemek, yukarıda anlatılan kestirim modu bilgisi kodlamasında da değişiklik yapılmasını gerektirir. Bu çalışmada CABAC entropi kodlayıcısı kullanıldığından, ilgili sözdizimi elemanlarının konteksleri göz önüne alınarak, ilgili sözdizimi elamanları ilave bir modu destekler hale getirilmelidir.

H.264/AVC kodlayıcı olarak standardın gelişimde de kullanılmış olan Joint Model (JM) [13] yazılımı kullanılmıştır. Söz konusu çerçeve içi kodlama mekanizmasındaki değişiklikler JM 14.1 kodu üzerinde yapılmıştır.

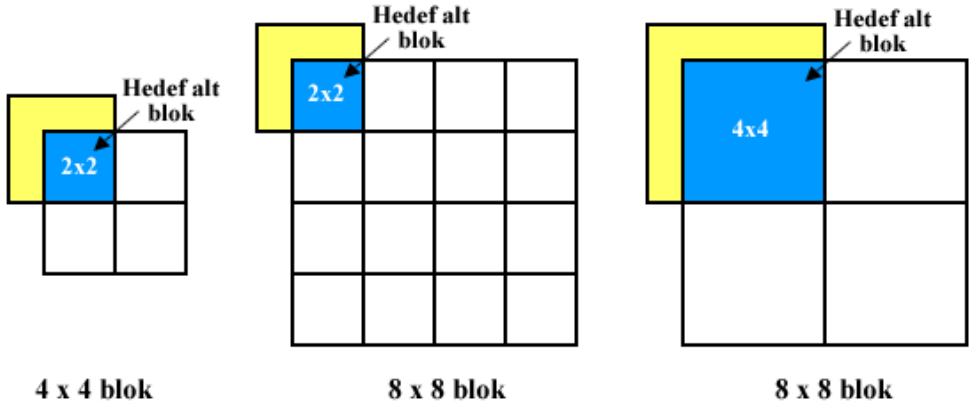
3.3.2 Şablon eşleme parametreleri

Şablon eşleme ile kestirim yapma yönteminin genel hatları yukarıda anlatılığı gibi olmakla birlikte, yöntemin uygulamaya göre optimize edilmesi gereken bazı

parametreleri vardır. Kodlanan resmin (ya da video çerçevesinin) kodlamadan önce analiz edilme imkanı varsa, bu parametreler dinamik olarak belirlenebilir. Ancak yüksek bit iletim oranlarının söz konusu olduğu videolarda, önceden bir analiz yapmak mümkün olmayabilir. Bu durumda parametrelerin sabitlenmesi ve hem kodlayıcıda hem de kod çözücüde aynı parametrelerin kullanılması gerekmektedir. Belirlenmesi gereken parametreler aşağıda özetlenmiştir.

3.3.2.1 Alt blok büyüklüğü

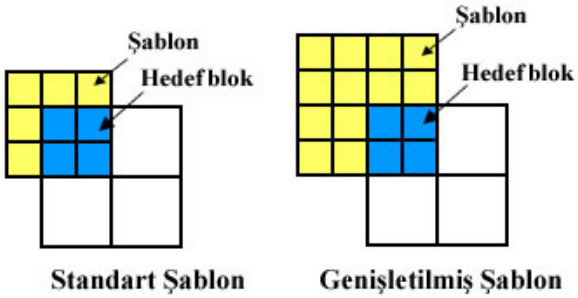
Şekil 3.2'de gösterildiği gibi şablon eşlemenin çerçeve içi kestirimde kullanılması ilk önerildiğinde, 2x2 lik bir alt blok büyüklüğünün ideal sonucu verebileceği belirtilmiştir [12]. Ancak kestirim blok büyüklüğünün 4x4 'den büyük olması durumunda (H.264/AVC 8x8 ve 16x16 da desteklemektedir), seçilen kestirim blok büyüklüğüne göre alt blok büyüklüğünün optimize edilmesi gerekir (Şekil 3.3). Bu çalışmada sadece 4x4 ve 8x8 lik kestirim blokları üzerinde şablon eşleme gerçekleştirildiğinden, optimum alt blok büyüklüğü olarak 2x2 ve 4x4 (sadece 8x8 lik bloklar için) incelenecektir.



Şekil 3.3 : Farklı blok büyüklükleri için şablon eşleme alt blok büyüklükleri

3.3.2.2 Şablonun şekli ve büyüklüğü

Alt bloğun şablonu olarak kullanılabilecek piksellerin yerleşimi, kestirim doğruluğunu etkileyen parametrelerden biridir. Alt blok büyüklüğüne göre seçilmesi gereken şablon şekli, Şekil 3.3'deki gibi "L" şeklinde olabileceği gibi, yönlü şekillerin daha iyi kestirimini sağlayacak şekilde de olabilir. Şablonu büyütmenin kestirim performansını ne şekilde etkilediği incelenecektir. Genişletilmiş şablon örneği Şekil 3.4'de görülmektedir. Yönlü şablonlar [14]'de incelenmiştir.



Şekil 3.4 : Genişletilmiş şablonun şekli

3.3.2.3 Arama bölgesi büyüklüğü

Kodlanmakta olan bloğun her bir alt bloğu için oluşturulan hedef blok şablonları, kodlanan bloğun komşuluğundaki daha önce kodlanmış ve kod çözülmüş piksellerin bulunduğu bir bölge içerisinde aranır. Bu bölgenin enine ve boyuna doğru büyüklüğü Şekil 3.2'de "x" ve "y" ile gösterilmiştir. Arama bölgesi büyüklüğü kodlama karmaşıklığını ve kodlama zamanını etkileyen bir parametre olduğundan, iyi kestirim yapılabilmesini sağlayan en küçük boyutlarda olmalıdır. Arama bölgesinin boyutları seçilirken, seçilmiş olan alt blok büyüklüğü de dikkate alınmalıdır.

3.3.2.4 Kaynak blok şablonuna karar verme kriteri

Arama bölgesi içerisinde hedef blok şablonuna piksel değerleri olarak en yakın kaynak şablonu aranırken, muhtemel her şablon ikilisi arasında piksel piksel karşılaştırma yapılır. İki şablon arasındaki benzerliğin ölçüsü olarak kullanılabilecek kriterler; mutlak farkların toplamı (sum of absolute differences – SAD), karesel farkların toplamı (sum of squared differences – SSD) yada literatürde yer alan diğer karşılatırma kriterlerinden biri olabilir. Bu çalışmada karar verme kriteri olarak; K kaynak blok şablonu pikseli ve H hedef blok şablonu pikseli olmak üzere, (3.1)'de gösterilen SAD ve (3.2)'de gösterilen SSD'nin performansları karşılaştırılacaktır.

$$SAD = \sum_{i=0}^{i=N} \left| K_i - H_i \right| \qquad \textbf{(3.1)}$$

$$SSD = \sum_{i=0}^{i=N} \left(K_i - H_i \right)^2 \qquad \textbf{(3.2)}$$

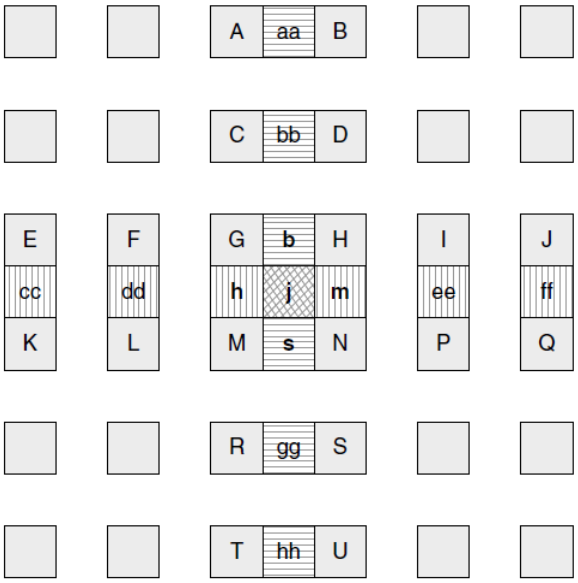
Seçilen karar verme kriterine göre, en düşük farka sahip kaynak şablonu seçilir. Eğer aynı fark değerine sahip birden fazla kaynak şablon varsa, arama bölgesi içerisinde arama yapılma sırasına göre son bulunan şablon seçilir.

3.3.2.5 Arama bölgesi piksel hassasiyeti

Arama bölgesi üzerinde hedef şablonuna en yakın kaynak şablonunu ararken, arama bölgesi çözünürlüğünü yükseltmek daha hassas sonuçlar elde edilmesini sağlayabilir. Arama bölgesi çözünürlüğünü yükseltmek, H.264/AVC standardında hareket dengeleme mekanizmasında da kullanılmaktadır. Çözünürlüğün artırılmasıyla, piksel hassasiyetinden, yarım veya çeyrek piksel hassasiyetine geçilebilir.

Çözünürlüğün hangi yöntemle artırılacağının da belirlenmesi gerekmektedir. Yine H.264/AVC standardında kullanılan yöntemler referans alınarak, yarım piksel hassasiyetine geçmek için 6 elemanlı sonlu impuls yanıtlı filtre (6 tap filter) yapısı kullanılması tercih edilmelidir. Bu filtre piksellerin arasının, komşu 3'er (yatay veya düşey) piksel kullanarak doldurulmasını sağlamaktadır.

Şekil 3.5'te görülen tam piksel değerlerinin arası üç aşamada doldurulmaktadır. A-U, tam pikselleri, diğerleri yarım pikselleri göstermektedir. İlk önce yatay doğrultuda filtreleme yapılarak resmin her bir satırındaki piksellerin arası tam piksellerin değerleri kullanılarak doldurulmaktadır. Örneğin Şekil 3.5'te b yarım pikseli, E, F, G, H, I ve J kullanılarak oluşturulmaktadır. Sonrasında düşey filtreleme yapılarak her bir sütundaki piksellerin arası yine tam piksellerin değerleri kullanılarak doldurulmaktadır. Örneğin Şekil 3.5'te h yarım pikseli, A, C, G, M, R ve T kullanılarak oluşturulmaktadır. Son aşama olarak yatay ve düşey yönde oluşturulmuş yarım piksellerin ortalarında boş kalan yarım piksel değerleri doldurulur. Bu işlem için ise ya yatay ya da düşey olarak, yeni oluşturulmuş yarım piksel değerleri kullanılır. Örneğin Şekil 3.5'te i yarım pikseli, ya cc, dd, h, m, ee ve ff yarım pikselleri kullanılarak, ya da aa, bb, b, s, gg ve h yarım pikselleri kullanılarak oluşturulmaktadır.



Şekil 3.5 : Yarım piksel hassasiyetine geçişte oluşturulan pikseller.

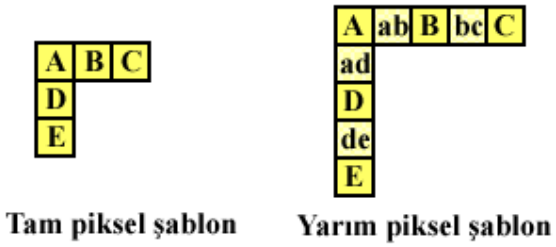
Yarım pikseller tam piksellerden oluşturulurken kullanılan filtrenin çalışması, Şekil Şekil 3.5'teki b pikselinin oluşturulması örnek verilerek (3.3)'de gösterilmektedir.

$$b = \text{yuvarla}((E - 5F + 20G + 20H - 5I + J)/32) \qquad \textbf{(3.3)}$$

Arama bölgesi şekil itibariyle dikdörtgensel olduğundan veya dikdörtgensel bölgelerin birleşimi olarak ifade edilebildiğinden dolayı, yukarıda anlatılan filtreleme yönteminin uygulanabilmesi mümkündür. Ancak arama bölgesi sınırlarında ne yapılması gerektiği de tanımlanmalıdır. Birinci alternatif eğer arama bölgesi dışında da kullanılabilir pikseller varsa, o piksellerin kullanılması yönündedir. İkinci alternatif ise her arama bölgesini dış piksellerden soyutlamak ve filtrelemeyi sadece arama bölgesi içerisindeki pikselleri kullanarak gerçekleştirmektir. Kodlayıcı (ve kod çözücü) karmaşıklığını azaltmak için ikinci yöntem tercih edilecektir. Ancak burada filtrenin sınır dışındaki pikselleri kullanmak durumunda kaldığı zaman ne yapılacağı tanımlanmalıdır.

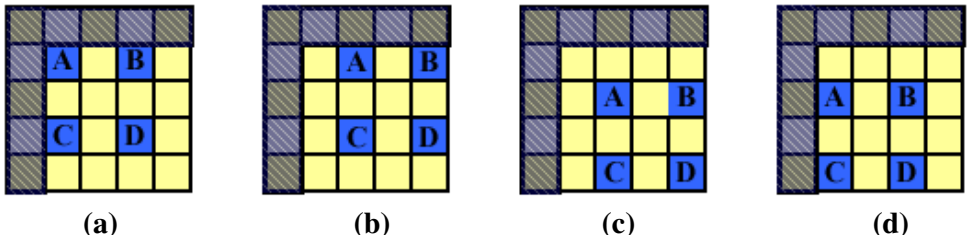
Eğer arama bölgesindeki bir piksel filtreleme için ihtiyaç duyuluyorsa, o zaman ilgili sınıra en yakın piksel kullanılır. Şekil 3.5'te gösterilecek olursa, F ile G arasındaki bir yarım pikseli (fg) oluşturmak için şu formül kullanılır:
$fg = \text{yuvarla}((E - 5E + 20F + 20G - 5H + I)/32)$

Yarım piksel hassasiyetine çıkartılmış bir arama bölgesinde arama yapabilmek için, aranacak şablonun da yarım piksel hassasiyetinde olması gerekmektedir (Şekil 3.6). Ancak şablon içerisindeki piksellerin konumu ve sayısı, 6 elemanlı filtreleme için gerekli olan uzunlukta değildir (uzatılmış bir şablon kullanılması da mümkündür, ancak kapsam dışında bırakılmıştır). Bu nedenle şablonun yarım piksel hassasiyetine çıkartılmasında seçilecek yöntemin de optimizasyonu gerekmektedir.



Şekil 3.6 : Şablon için tam piksel - yarım piksel geçişi

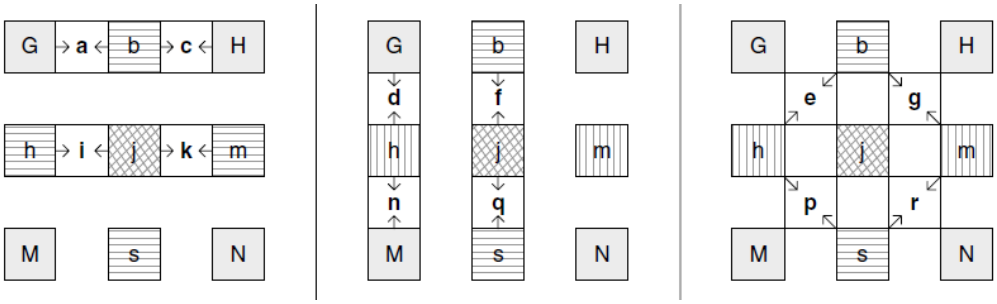
Arama bölgesi içerisinde en uygun kaynak alt blok şablonu bulunduğunda, kaynak alt bloğu yarım piksel hassasiyetinde 4x4 lük bir alandan oluşur. 2x2 lik alt blok büyüklüğü kullanıldığı için 4x4'den - 2x2'ye geçiş yapılırken kullanılacak yöntemin de belirlenmesi gerekir. Kullanılabilecek yöntemlerden biri, her iki pikselden birini atmaktır. Şekil 3.7'de piksel atma yönteminde kullanılabilecek dört farklı seçenek görülmektedir. Taralı bölge şablonu, A, B, C ve D, seçilen pikselleri göstermektedir.



Şekil 3.7 : Hedef alt bloğun oluşturulması için seçilebilecek piksel atma konfigürasyonları

Çeyrek piksel hassasiyetle şablon eşleme gerçekleştirmek içinse, arama bölgesinde önce yarım pikseller oluşturulmalı, sonrasında bu yarım pikseller kullanılarak çeyrek pikseller elde edilmelidir. Çeyrek piksellerin oluşturulma yöntemi Şekil 3.8'de görülmektedir. Her bir çeyrek piksel, kendisine komşu iki pikselin enterpolasyonu ile doldurulur. Bu komşu piksellerin ortalama değeri çeyrek piksel değerini oluşturur.

Çeyrek piksel pozisyonlarının doldurulması işlemi üç aşamada tamamlanır. İlk önce yatay doğrultuda tam ve yarım piksellerin arasında kalan çeyrek piksel pozisyonları, sol ve sağ taraflarındaki piksellerin değerleri kullanılarak doldurulur. Sonrasında dikey doğrultudaki tam ve yarım piksellerin arasında kalan çeyrek piksel pozisyonları, yukarısındaki ve aşağısında pikseller kullanılarak doldurulur. Son olarak diğer çeyrek piksel pozisyonları, çaprazında kalan (sol alt – sağ üst ikilisi yada sol üst – sağ alt ikilisi) yarım piksel pozisyonunda bulunan iki komşu değerin enterpolasyonu ile doldurulur.



Şekil 3.8 : Çeyrek piksellerin oluşturulması

Çeyrek piksellerin enterpolasyonu için komşu piksellerin ortalamasının alınmasında, örneğin Şekil 3.8'de e olarak isimlendirilmiş çeyrek piksel için, (3.4) kullanılmaktadır.

$$e = \text{yuvarla}((b+h)/2) \qquad \textbf{(3.4)}$$

Şablon eşlemenin çeyrek piksel hassasiyetle yapılacağı durumda hem arama bölgesi hem de şablonun çeyrek piksel pozisyonları doldurulur. Arama bölgesi içerisinde en uygun kaynak alt blok şablonu bulunduğunda, kaynak alt bloğu çeyrek piksel hassasiyetinde 8x8 lik bir alandan oluşur. 2x2 lik alt blok büyüklüğü kullanıldığı için 8x8'den - 2x2'ye geçiş yapılırken kullanılacak yöntemin de belirlenmesi gerekir. 8x8 lik bloktan seçilecek 4 piksel değeri ile 2x2 lik bloğun oluşturulması gerekmektedir. Yarım piksel kullanımında da benzer bir problemle karşılaşıldığı için, yarım piksel için tercih edilecek yöntem, çeyrek piksel durumu için de gerekli değişikliklerle kullanılabilir.

3.3.2.6 Kullanılan kestirim alt bloğu sayısı

Bir alt blok için en uygun kaynak şablonu bulunduğunda, kaynak alt blok kestirim alt bloğu olarak kullanır. Ancak çoğu durumda, arama bölgesi içerisinde hedef şablonu ile farkı (SAD) aynı yada yakın olan birden fazla kaynak şablonu bulunabilir. Bu durumda en düşük SAD değerlerlerine sahip olan kaynak alt blokların ortalamasının alınması daha iyi sonuç verebilir [14]. Ancak alt blokların ortalamasının alınabilmesi için, blokların SAD değerleri önceden tanımlanmış bir değerin altında olmalıdır.

Alt blokların ortalaması alınırken, bu blokların ağırlandırılması da söz konusu olabilir. Ağırlıklandırma şu iki kriterden birine göre seçilebilir:

- Kaynak alt bloğun, hedef alt bloğa uzaklığı
- Kaynak alt blok şablonunun SAD değeri

Ayrıca bu değerler, ağırlıklandırmada karesel ya da lineer olarak kullanılabilir. Örneğin uzaklık kriteri kullanılacaksa, uzaklığın karesi ile ters orantılı olacak şekilde bir ağırlıklandırma seçilebilir.

4. SONUÇLAR

Bu bölümde şablon eşleme ile çerçeve içi kestirimde kullanılabilecek parametrelerin kodlama verimliliğine etkisi incelenecektir. Sonuçların elde edilmesinde kullanılan test kriterleri ve test ortamı hakkında da bilgi verilecektir.

Şablon eşleme ilk olarak [12] gibi önceki çalışmalarda önerilen parametreler kullanılarak H.264/AVC kodlayıcısı ve kod çözücüsünün içerisine yeni bir çerçeve içi kestirim modu olarak eklenecek ve test kriterleri kullanılarak kodlama verimliliği incelenecektir.

Ardından diğer şablon eşleme parametrelerinin getirdiği ilave verimlilik kazançları incelenecek ve verimlilik / karmaşıklık bakımından optimum çözüm bulunmaya çalışılacaktır.

4.1 Test Kriterleri ve Test Ortamı

Şablon eşleme ile çerçeve içi kestirimin uygulandığı video kodlama standardı H.264/AVC olarak seçildiğinden dolayı, bu standardın geliştirilmesi sürecinde standarda dahil edilmesi düşünülen özelliklerin verimliliğe etkisini test etmek amacıyla ortaya konan kriterler, bu çalışmada da kullanılacaktır. VCEG'in (Video Coding Experts Group) yüksek profil için belirlediği koşullar [16] belirtildiği şekilde kullanılmıştır. Bu kriterlerden çerçeve içi kestirim için geçerli olanlar ve kullanılan diğer kodlayıcı parametreleri Çizelge 4.1'de verilmiştir.

Ölçümlerin yapıldığı test videoları olarak [16]'da belirtilenlere ilaveten birkaç video daha kullanılmıştır. Kullanılan videoların isimleri ve çözünürlükleri Çizelge 4.2'de verilmiştir. Bu videolar 4:2:0 renk alt örneklemesiyle oluşturulmuştur.

Çizelge 4.1 : Test için kullanılan H.264/AVC kodlayıcı parametreleri

Parametre İsmi	Kullanılan Değer	Kodlayıcı Parametresi
Profil	Yüksek Profil	ProfileIDC = 100
Entropi kodlayıcı	CABAC	SymbolMode = 1
8x8 dönüşüm	Kullanılıyor	Transform8x8Mode = 1
Harici kuantalama matrisi	Kullanılmıyor	ScalingMatrixPresent = 0
Dilim modu	Çerçeve başına tek	SliceMode = 0
Çerçeve içi kestirim periyodu	1	IntraPeriod = 1
Video başına çerçeve sayısı	150	FramesToBeEncoded = 150
Çerçeve atlama periyodu	0	FrameSkip = 0
Bit oranı – bozulma optimizasyonu	Kullanılıyor	RDOptimization = 1
Kuantalama parametresi	22, 27, 32, 37	QPISlice = (22, 27, 32, 37)

Şablon eşleme ile çerçeve içi kestirim uygulaması, standardın geliştirilmesinde de kullanılan JM [13] yazılımıyla gerçekleştirilmiştir. Yöntem hem kodlayıcı, hem de kod çözücüde değişiklik yapılmasını gerektirdiğinden, test edilen her parametre için hem kodlayıcı hem de kod çözücü kodunda eklemeler yapılmıştır.

Çizelge 4.2 : Test için kullanılan videolar

Video ismi	Genişlik	Yükseklik
Mobile	352	288
Highway	352	288
Foreman	352	288
Ice	704	576
Paris	352	288
Tempete	352	288
Container	352	288

Parametrelerin kestirim moduna ve kodlama verimliliğine etkisi iki temel değer üzerinden incelenecektir. Birincisi, kodlayıcının çerçeve başına harcadığı bit miktarıdır. İkincisi ise kodlanan orijinal video ile kodlanıp, kod çözülmüş video arasındaki farkı temsil eden ortalama PSNR değeridir. PSNR değeri tek bir çerçeve için (4.1)'de görüldüğü şekilde hesaplanır. Burada $MAKS_I$ bir pikselin alabileceği maksimum değeri temsil eder ve 255 olarak kullanılır. MSE ise (4.2)'de ifade edildiği gibi, kıyaslanan iki çerçeve arasındaki ortalama karesel hatayı göstermektedir.

$$PSNR = 20 \cdot \log_{10}\left(\frac{MAKS_I}{\sqrt{MSE}}\right) \quad \textbf{(4.1)}$$

$$MSE = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left\|I(i,j) - K(i,j)\right\|^2 \quad \textbf{(4.2)}$$

Yapılan değişikliklerin kodlama verimliliğine etkisi ölçülürken, Bjontegaard'ın video kodlama verimlilik ölçümü konusunda yaptığı önerme [17] kullanılmıştır. Bu önermede bit iletim oranına karşılık PSNR'ın değişimi göz önüne alınarak, kodlama verimliliği ölçülmektedir. Ölçüm için oluşturulan *avsnr4* yazılımına [18] test sonuçları verilerek, "BD-Rate" olarak adlandırılan Bjontegaard Delta Rate (Bit Oranı farkı) değerleri elde edilecektir (Ek A.1) .

Bir parametrenin kodlama karmaşıklığını ne kadar artırdığına dair bilgi verilen testlerde, video çerçevesi başına kodlayıcının harcadığı ortalama süre milisaniye/çerçeve cinsinden belirtilecektir. Kodlayıcı olarak kullanılan JM yazılımı, Linux işletim sisteminde çalışacak şekilde derlenmiş olup, testlerin koşturulduğu bilgisayar, Intel Q6600 2.4 GHz işlemciye ve 2 GB rasgele erişim belleğine (RAM) sahiptir. Uygulama tek izlek (thread) kullanarak çalışmaktadır.

4.2 Referans Uygulama

Parametre değişimlerinin şablon eşleme ile çerçeve içi kestirim verimliliğini nasıl etkilediğini ölçmek için referans olarak [12]'de önerilen parametreler kullanılarak alınan sonuçlar kullanılacaktır.

H.264/AVC standardında tanımlı çerçeve içi kestirim modlarına bir yenisi eklendiğinden dolayı, her blok için kod çözücüye gönderilen mod bilgisinin kodlanmasında değişiklik yapılması gerekir. Normalde Çizelge 3.1'de görüldüğü gibi kestirim modu sözdizimi elemanı 0 ila 7 arasında bir değer alır ve entropi kodlayıcı tarafından 3 bitlik kabul edilir. Yeni modun eklenmesiyle bu sözdizimi elemanının alabileceği değer aralığı 0 – 8 arasına çıkar. Bu nedenle seçilen bazı modların 3 bit olarak değil 4 bit olarak entropi kodlayıcıya gönderilmesi gerekir. Sözdizimi elemanının kodlanmasını, kodlama limitine yaklaştıran çözümlerden biri olduğu için, Çizelge 4.3'te verilen eşleştirme ikilileme için uygun görülmüştür. Bu şekilde sözdizimi elemanı 7 ya da 8 olursa bir bit fazladan gönderilmesi gerekecektir.

Çizelge 4.3 : Kestirim modu sözdizimi elemanlarının yeni modla beraber kodlanması

Kestirim Modu Sözdizimi Elemanı	Ikili Ifade
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	1110
8	1111

Şablon eşleme ile kestirim modunun indeksi olarak 2 seçilmiştir. Standartta tanımlanan mod indekslerinde 2 ve sonrasında gelen modlar bir kaydırılmıştır. Mod indeksi olarak 2 seçilmesinin nedeni, modların standart içerisinde tanımlanan indekslerinin de, modların seçilme sıklıklarına bağlı olarak belirlenmiş olmasıdır. Bu çalışmada da şablon eşleme ile kestirim modunun seçilme sıklığı analiz edilmiş ve ortalama en çok seçilen 3. kestirim modu olduğu belirlenmiştir.

Şablon eşleme parametrelerinin [12]'ye bağlı olarak seçilen değerleri Çizelge 4.4'de görülmektedir.

Çizelge 4.4 : Referans uygulama şablon eşleme parametreleri

Parametre	Kullanılan Değer
Alt blok büyüklüğü (4x4 bloklar için)	2x2
Alt blok büyüklüğü (8x8 bloklar için)	2x2
Şablon şekli ve büyüklüğü	L şeklinde 3 – 3
Arama bölgesi büyüklüğü (4x4 bloklar için)	16 x 12
Arama bölgesi büyüklüğü (8x8 bloklar için)	32x24
Şablon karar verme kriteri	SAD
Piksel hassasiyeti	Tam
Kestirim alt bloğu sayısı	1

Önceki çalışmalarda [14] elde edilen, şablon eşleme yönteminin kodlama kazancına etkisine ilişkin değerler Çizelge 4.5'de görülmektedir. Ancak bu çalışmalarda

kullanılan kestirim modu kodlama yöntemi, en olası mod belirleme şekli ve kodlayıcı yazılımı farklıdır ve yazılım versiyonu hariç diğer yöntemlerin tanımı belirgin değildir. Bu nedenle Çizelge 4.5'de verilen değerler sadece bilgi amaçlı olarak sunulmuştur. Birden fazla kestirim alt bloğunun kullanımını inceleyen bölümde de benzer yöntemi uygulayan bu çalışmada elde edilen değerlerle karşılaştırma yapılacaktır.

Çizelge 4.5 : Önceki çalışmalarda elde edilen kazanç değerleri

Video ismi	**BD-Rate kazancı (%)**
Mobile	0.67
Highway	-
Foreman	4.96
Ice	-
Paris	1.63
Tempete	0.52
Container	0.94

Uygulamanın H.264/AVC kodlayıcısı içerisine eklenmeden önce elde edilen bit iletim oranı ve PSNR değerleri ile, uygulama eklendikten sonra elde edilen değerler ve BD-Rate cinsinden bit iletim oranının değişimi Çizelge 4.6'da verilmiştir.

Şablon eşleme kullanılan referans uygulamanın kodlama karmaşıklığına etkisini yaklaşık olarak gözlemleyebilmek için Çizelge 4.7'da kodlayıcının her bir video çerçevesi için harcadığı ortalama kodlama süresi verilmiştir. Görüldüğü üzere şablon eşleme, kodlama süresini artırmaktadır.

Çizelge 4.6 : Referans uygulama ile Standart kodlamanın verimlilik farkı

		Standart JM 14.1		Şablon eşleme referans uygulama		
Test Videosu	**QP**	**PSNR**	**Bit / Çerçeve**	**SNR**	**Bit / Çerçeve**	**BD-Rate değişimi (%)**
Mobile	22	41.8	350206	41.82	349444	-0.299
	27	37.05	246579	37.06	245929	
	32	32.39	163684	32.39	163283	
	37	28.29	104056	28.29	103811	
Highway	22	42.29	82889	42.25	79941	-4.359
	27	40.02	37661	39.99	35831	
	32	37.7	19701	37.67	18528	
	37	35.3	11428	35.3	10861	
Foreman	22	42.14	126813	42.15	125273	-1.910
	27	38.55	71291	38.56	70256	
	32	35.33	39250	35.33	38426	
	37	32.5	22463	32.51	21816	
Ice	22	44.24	194101	44.24	191447	-1.333
	27	41.87	106421	41.86	104406	
	32	39.44	61147	39.42	60145	
	37	36.93	37133	36.93	36863	
Paris	22	42.2	225661	42.22	223895	-1.004
	27	37.99	151617	38	150183	
	32	33.83	96942	33.82	95897	
	37	30.07	59751	30.07	59131	
Tempete	22	41.99	254919	42	254790	-0.150
	27	37.46	169493	37.47	169336	
	32	33.04	104142	33.04	104020	
	37	29.29	60225	29.29	60159	
Container	22	42.29	153516	42.29	153435	-0.185
	27	38.33	96027	38.35	96021	
	32	34.74	55612	34.75	55623	
	37	31.61	32250	31.61	32224	

Çizelge 4.7 : JM yazılımı üzerinde şablon eşlemenin video kodlama süresine etkisi

Test Videosu	**Standart JM 14.1**	**Referans Uygulama**	**Değişim**
Mobile	491	697	%42.11
Highway	443	651	%46.80
Foreman	1642	2511	%52.98
Ice	628	844	%34.50
Paris	558	776	%38.97
Tempete	574	787	%37.12
Container	502	717	%42.66

4.3 Şablon Eşleme Parametrelerinin Kodlama Verimliliğine Etkisi

Bu bölümde daha önce açıklanan, şablon eşleme parametrelerinin alabileceği farklı değerlerin referans uygulamaya göre kodlama verimliliğine olan katkısı incelenecektir.

4.3.1 8x8 bloklar için 4x4 alt blok kullanılması

Referans uygulamada 8x8 bloklar için 2x2 lik alt bloklar kullanılmıştır. H.264/AVC standardına göre 8x8 tipindeki parlaklık makrobloklarında (16x16 büyüklüğündedir) her bir 8x8 blok için 8x8 lik dönüşüm uygulanır. Dönüşüm blok büyüklüğü ile şablon eşleme alt blok büyüklü arasındaki farkı azaltmak ve sonucunu gözlemlemek için, 8x8 lik bloklar için 4x4 lük alt blokların kullanımı test edilmiştir. Sonuçlar Çizelge 4.8'de görülmektedir. 2x2 lik alt bloklarla alınan sonuçlar referans uygulamaya aittir.

Çizelge 4.8 : 8x8 lik bloklar için farklı alt blok büyüklüklerinin oluşturduğu % BD-Rate değişimi (JM14.1'e göre)

Test Videosu	2x2 Alt Bloklarla	4x4 Alt Bloklarla	Değişim
Mobile	-0.30	-0.32	-0.02
Highway	-4.36	-4.36	0.00
Foreman	-1.91	-2.07	-0.16
Ice	-1.33	-1.55	-0.22
Paris	-1.00	-0.94	0.06
Tempete	-0.15	-0.17	-0.02
Container	-0.18	-0.25	-0.07
Ortalama	**-1.32**	**-1.38**	**-0.06**

Sonuçlardan görüldüğü üzere, 8x8 tipi makroblokların 8x8 lik blokları için şablon eşleme yapılırken alt blok büyüklüğünün artırılması daha iyi kestirim yapılabilmesini sağlamaktadır. Ancak bu değişim çok büyük değildir ve kodlayıcının 4x4 lük bloklarla 8x8 lik bloklar için farklı şekilde şablon eşleme yapmasını gerektirmektedir. Bunun sonu olarak da alt blok büyüklüğü, şablon büyüklüğü ve şablon şekli değişmektedir. Kodlayıcıda iki farklı şablon eşleme metodu yerine tek bir tanesi uygulanmak istenirse, alt blok büyüklüğünü hem 4x4 hem de 8x8 lik bloklar için 2x2 olarak kabule etmek, kodlama verimliliğinde ciddi bir kayba neden olmayacaktır.

4.3.2 Genişletilmiş şablon kullanılması

Şablonu genişletmenin kestirim doğruluğuna nasıl etki edeceği, Şekil 3.4'de önerilmiş olan bir genişletilmiş şablon şekli kullanılarak incelenecektir. Bahsedilen genişletilmiş şablonu kullanırken, arama bölgesindeki 1'er satır ve sütun şablonun yeni şeklinden dolayı kaynak şablon olarak seçilemeyecektir. En uygun kaynak alt blok şablonu bulunurken 5 piksel yerine 12 piksellik karşılaştırma yapılacaktır. Bu şekilde kaynak alt bloğun daha fazla hassasiyetle seçileceği beklenebilir. Buradaki amaç, alt blokların karakteristiğini yansıtacak en uygun şablon büyüklüğünü belirlemektir.

Genişletilmiş şablon kullanılarak alınan sonuçlar Çizelge 4.9'de gösterilmiştir. Bu çizelgede referans uygulama ve genişletilmiş şablon kullanımın JM 14.1 yazılımına göre BD-Rate değişimleri ile genişletilmiş şablonun referans uygulamaya göre BD-Rate değişimi görülmektedir. Genişletilmiş şablon kullanımın test videolarının bazılarında kodlamayı iyileştirdiği bazılarında ise kötüleştirdiği görülmektedir.

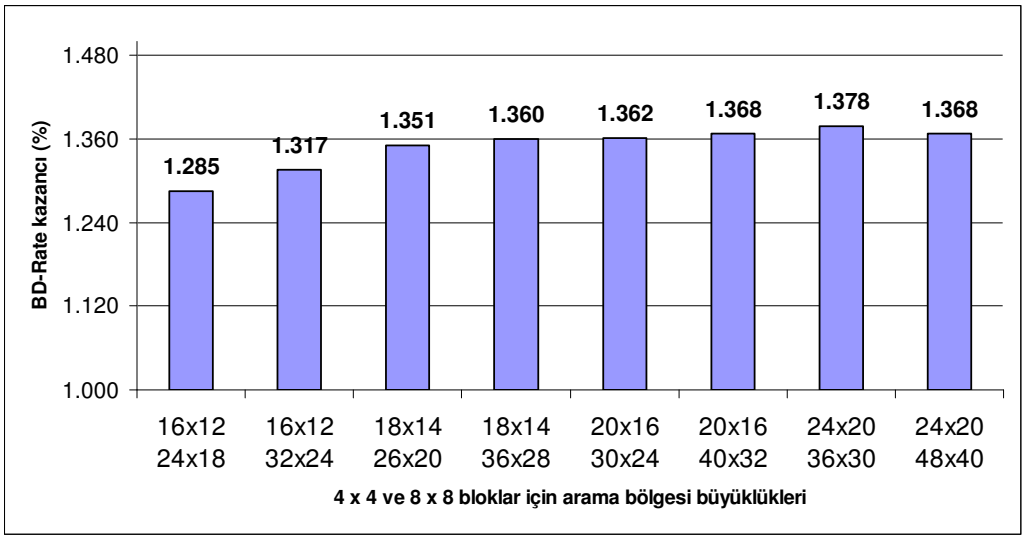
Çizelge 4.9 : Genişletilmiş şablon kullanımın oluşturduğu BD-Rate değişimleri

Test Videosu	Normal şablon (% BD-Rate değişimi)	Genişletilmiş şablon (% BD-Rate değişimi)	Referansa göre fark (% BD-Rate değişimi)
Mobile	-0.30	-0.35	-0.05
Highway	-4.36	-4.58	-0.22
Foreman	-1.91	-1.96	-0.05
Ice	-1.33	-1.25	0.08
Paris	-1.00	-0.99	0.01
Tempete	-0.15	-0.10	0.05
Container	-0.19	-0.17	0.02

4.3.3 Arama bölgesinin büyütülmesi

Kodlayıcı bir makrobloğu kodlarken hem 4x4'ü hem de 8x8'i blok büyüklüğü olarak denemektedir ve daha fazla kodlama kazancı getireni makroblok tipi olarak seçmektedir. Dolayısıyla bir makroblok kodlanırken, hem 4x4 blok kestirimleri hem de 8x8 blok kestirimleri denenmektedir. 8x8 blokların amacı daha düzgün parlaklık dağılımlı alanların kestiriminde kullanılmak olduğundan, kestirimleri için kullanılan alanın büyüklüğü de daha büyük olmalıdır. Bu nedenle şablon eşlemede de 8x8 blokların şablon eşlemesinde 4x4 bloklara göre daha büyük arama alanları kullanılmaktadır.

Referans uygulamada kullanılan arama bölgesi büyüklükleri [12]'de kullanılan büyüklerdir. Sonrasında [14]'de yapılan önermelerde 4x4 ve 8x8 lik bloklar için arama bölgesi büyüklükleri birbirlerinin iki katı olacak şekilde seçilmiştir. Bu çalışmada kullanılan referans uygulamanın farklı arama büyüklüklerinde nasıl sonuç verdiği ve 4x4 ile 8x8 lik bloklar için kullanılan arama bölgesi büyüklüklerinin birbirine yakın olduğu durumda sonucun nasıl değiştiğini incelemek amacıyla, Şekil 4.1'de verilmiş olan değerler arama bölgesi büyüklükleri olarak kullanılmıştır. Elde edilen değerlerin tümü Çizelge A.1'de verilmiştir.



Şekil 4.1 : Arama bölgesi büyüklüklerinin BD-Rate kazancına ortalama etkisi

Şekilden de görüldüğü üzere, arama bölgesi büyüklüğü arttıkça, kodlama verimi de artmaktadır. 4x4 lük bloklar için arama bölgesi büyüklüğünün artışı kodlama kazancının artışında daha belirleyici olmaktadır. Ayrıca bir büyüklüğün üzerine çıkıldığı zaman, kodlayıcının yanlış karar verme ihtimali de arttığından, elde edilen verimlilik azalmaya başlamaktadır.

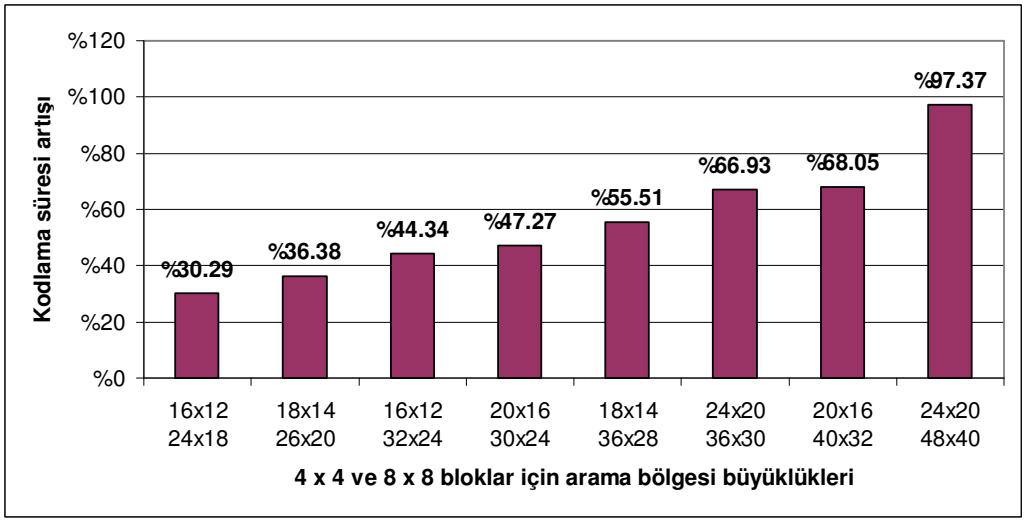
Şekil 4.2'de arama bölgesi büyüklüğünün kodlama süresine etkisi gösterilmiştir. Bir makrobloğun kodlanmasında hem 4x4 hem de 8x8'lik blokların kestirimleri denendiğinden dolayı, 4x4 ve 8x8'lik blokların tüm alt blokları için harcanan şablon eşleme süresi, şablon eşlemenin getirdiği maliyet olarak kullanılabilir. Bir makroblok için yapılan toplam şablon eşleme işlemi sayısı (4.3)'de görüldüğü gibidir.

$$\text{İşlem sayısı} = \text{alt blok sayısı} \times (\text{arama bölgesi genişliği} \times \text{arama bölgesi yüksekliği}) \quad \textbf{(4.3)}$$

Makroblok içerisinde kestirime tabi tutulan 2x2 lik alt blok sayısı 4x4 bloklar için 4×16 ve 8x8 bloklar için 16×4'dür. Böylece bir makroblokda 128 alt blok için verilen arama bölgesi büyüklüğü ile şablon eşleme yapılır. Bu durumda 2x2 lik alt bloklarla şablon eşlemenin bir makroblok için getirdiği maliyet (4.4)'de verildiği gibi olacaktır. Buradan da anlaşılmaktadır ki, 4x4 ve 8x8 lik bloklar için arama bölgesi büyüklüklerinin toplamı şablon eşlemenin maliyeti ile orantılıdır.

$$\text{Maliyet} \approx 64 \times (\text{4x4 bölge genişliği} \times \text{4x4 bölge yüksekliği}) + 64 \times (\text{8x8 bölge genişliği} \times \text{8x8 bölgesi yüksekliği}) \quad \textbf{(4.4)}$$

Şekil 4.2'den de görüldüğü üzere, arama bölgesi büyüklüklerinin toplamı ile kodlama süresini orantılı olarak artırmaktadır. Bu nedenle şablon eşlemenin kullanılacağı bir uygulamada var olan kodlayıcı performansı temel alınarak, arama bölgesi büyüklüğünün seçilmesi ideal çözümü oluşturacaktır.



Şekil 4.2 : Arama bölgesi büyüklüklerinin video kodlama süresine etkisi

4.3.4 Kaynak – hedef şablon eşleştirmesinde karesel farkların kullanımı

Kaynak alt blok şablonu ile hedef alt blok şablonu eşleştirmesi yapılırken şablonlar arasındaki farkın hesaplanmasında kullanılan SAD yerine SSD'nin kullanımı piksel farklarının düşük olduğu durumlarda daha fazla hassasiyet ile şablon seçimini

sağlayabilmektedir. Bu durumu inceleyebilmek için kodlayıcı (ve kod çözücünün) karar kriteri olarak SSD'yi kullandığı durumda elde edilen sonuçlar ve referans uygulamaya göre farkı Çizelge 4.10'de verilmiştir.

Çizelge 4.10 : Şablon eşleştirmede SSD kullanımın oluşturduğu BD-Rate değişimleri

Test Videosu	Referans Uygulama (% BD-Rate değişimi)	SSD kullanımı (% BD-Rate değişimi)	Referansa göre fark (% BD-Rate değişimi)
Mobile	-0.30	-0.27	0.03
Highway	-4.36	-4.21	0.15
Foreman	-1.91	-2.04	-0.13
Ice	-1.33	-1.46	-0.13
Paris	-1.00	-1.03	-0.03
Tempete	-0.15	-0.15	0.00
Container	-0.19	-0.25	-0.06

SSD kullanımı bazı test videolarında daha olumlu sonuçlar alınması sağlamıştır. SSD kullanırken mutlak değer işlemi yerine kare alma işlemi kullanılmaktadır ve her iki işlem arasında maliyet bakımından bir fark olmadığı kabul edilebilir. Bu nedenle şablon eşleştirilirken SSD kullanımının SAD'ye göre daha iyi sonuçlar üretmediği çıkarımı yapılabilir.

4.3.5 Şablon eşlemenin yarım piksel hassasiyetiyle yapılması

Arama bölgesinde yarım piksel hassasiyetle arama yapılmasının, şablonların daha fazla doğrulukla seçilmesine etkisini görmek için iki aşamalı bir inceleme yapmak uygundur. İlk olarak sadece 4x4 lük bloklar için arama bölgesi hassasiyetinin artırılması incelenecektir. Sonrasında, hem 4x4 hem de 8x8 lik bloklar için arama bölgesi hassasiyetinin artırıldığı durum incelenecektir.

Çizelge 4.11'da her iki inceleme için de sonuçlar yer almaktadır. Görüleceği üzere arama bölgesinde yarım piksel hassasiyet kullanmak kodlama verimini artırmaktadır. Sadece 4x4 lük bloklar içi şablon eşleme yapılırken de referans uygulamaya göre daha iyi sonuçlar alınmaktadır. 8x8 lük bloklar için de yarım piksel hassasiyetin kullanıldığı durumda bir miktar daha iyileşme olmaktadır.

Çizelge 4.11 : Arama bölgesi hassasiyetinin artırılmasının oluşturduğu % BD-Rate değişimleri

Test Videosu	4x4 tam 8x8 tam	4x4 yarım 8x8 tam	4x4 yarım 8x8 yarım
Mobile	-0.30	-0.31	-0.30
Highway	-4.36	-4.79	-4.85
Foreman	-1.91	-2.57	-2.61
Ice	-1.33	-1.75	-1.85
Paris	-1.00	-1.11	-1.11
Tempete	-0.15	-0.22	-0.22
Container	-0.19	-0.27	-0.22
Ortalama	-1.32	-1.57	-1.59

Yarım piksel hassasiyete geçişte önce arama bölgesi çözünürlüğü artırıldığından dolayı Çizelge 4.12'de görüleceği üzere, kodlama süresi de artmaktadır. 8x8 lük bloklar için 4x4 lük bloklara kıyasla daha büyük arama bölgesi kullanıldığından süre artışı daha da yükselmektedir. Şablon eşlemenin kullanılacağı uygulama piksel hasiyetinin artırılmasında belirleyici rol oynamalıdır.

Çizelge 4.12 : Arama bölgesi hassasiyetinin artırılmasının oluşturduğu ortalama çerçeve kodlama süresi artışları (milisaniye)

Test Videosu	4x4 tam 8x8 tam	4x4 yarım 8x8 tam	Değişim	4x4 yarım 8x8 yarım	Değişim
Mobile	697	874	%25.33	1596	%128.77
Highway	651	826	%26.90	1543	%137.05
Foreman	2511	3220	%28.20	6254	%149.05
Ice	844	1020	%20.80	1740	%106.03
Paris	776	951	%22.51	1671	%115.37
Tempete	787	961	%22.18	1687	%114.44
Container	717	892	%24.43	1616	%125.54

Yarım piksel hassasiyet kullanılarak arama bölgesindeki en uygun blok belirlendiğinde, tam piksele nasıl dönüleceğine dair örnek konfigürasyonlar Şekil 3.7'de verilmiştir. Çizelge 4.11'da sonuçları verilmiş uygulamada (c) konfigürasyonu kullanılmıştır. Sadece 4x4 lük bloklarda yarım piksel arama bölgesi kullanılması durumu için piksel arama konfigürasyonlarının birbirlerine göre farkları Çizelge 4.13'de verilmiştir.

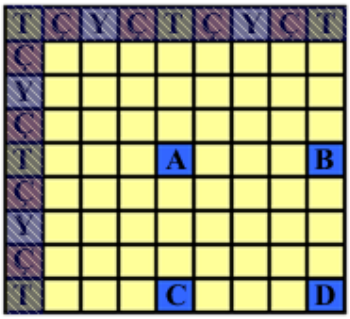
Çizelge 4.13 : Seçilen piksel atma konfigürasyonuna göre elde edilen % BD-Rate kazancı

Test Videosu	C	A	B	D
Mobile	0.30	0.16	0.10	0.15
Highway	4.34	3.47	1.79	2.52
Foreman	1.91	2.77	0.74	0.94
Ice	1.33	0.61	1.48	0.55
Paris	1.00	0.23	0.37	0.48
Tempete	0.15	0.07	0.10	0.24
Container	0.19	0.13	0.07	0.23
Ortalama	1.32	1.06	0.66	0.73

Piksel atma konfigürasyonlarından b veya d'nin seçilmesiyle elde edilen kazançların a veya c'nin seçilmesiyle elde edilenden büyük farklar içeriyor olmasının nedeni, seçilen piksellerin merkezi ile, çözünürlük artırılırken eklenen piksellerin merkezinin farklı olmasıdır. Çözünürlük artırılırken yatay ve düşey doğrultuda eşit miktarda ara piksel oluşturulmuştur. Ancak b ve d konfigürasyonlarında seçilen piksellerin merkezi yatay kenara yada düşey kenara yakındır. C ve a konfigürasyonlarında ise seçilen piksellerin merkezi, 4x4 yarım piksel hassasiyetli bloğun merkezine daha yakındır. C konfigürasyonunun en iyi sonucu vermesinin nedeni ise, seçilen piksellerle şablon piksellerinin uzaklığının ortalamasının tam sayı olmasıdır. Bu nedenle şablonun 2x2 lik kaynak alt bloğunu (tam piksel olarak) daha doğru ifade etmektedirler.

4.3.6 Şablon eşlemenin çeyrek piksel hassasiyetiyle yapılması

Arama bölgesinde yarım piksel hassasiyetle arama yapılmasının getirdiği kazancı ölçerken elde edilen verilerle, çeyrek hassasiyetle arama yapmanın getireceği kazanç da incelenebilir. Örneğin 8x8 lik bloklarda yarım piksel kullanımı önemli bir kazanç sağlamadığı için sadece 4x4 lük bloklar için çeyrek piksel kullanımı incelebilir. Aynı şekilde en uygun blok belirlendiğinde elde edilen 8x8 lik bloktan 2x2 lik bloğa nasıl geçileceğine dair yöntem de, yarım piksel hassasiyetine dair yapılan incelemede elde edilen veriler kullanılarak belirlenebilir.



Şekil 4.3 : Hedef alt bloğun oluşturulması için seçilen piksel atma konfigürasyon

Sadece 4x4 lük blokların şablon eşlemesinde çeyrek pikselle eşleme yapıldığı ve en uygun blok belirlendikten sonra 8x8 lik bloktan 2x2 lik bloğa geçiş için Şekil 4.3'de gösterilen konfigürasyonun kullanıldığı durumda elde edilen sonuçlar Çizelge 4.14'de verilmektedir.

Çizelge 4.14 : Arama bölgesinde çeyrek piksel kullanımının oluşturduğu % BD-Rate değişimleri

Test Videosu	4x4 tam 8x8 tam	4x4 çeyrek 8x8 tam	Değişim
Mobile	-0.30	-0.37	-0.07
Highway	-4.36	-4.92	-0.56
Foreman	-1.91	-2.75	-0.84
Ice	-1.33	-2.03	-0.70
Paris	-1.00	-1.28	-0.28
Tempete	-0.15	-0.29	-0.14
Container	-0.19	-0.27	-0.08
Ortalama	-1.32	-1.70	-0.38

Görüleceği üzere arama bölgesinde çeyrek piksel hassasiyet kullanmak kodlama verimini artırmaktadır. Bu artış yarım piksel kullanımında elde edilen artıştan (Çizelge 4.11) daha fazladır.

4.3.7 Birden fazla kestirim alt bloğunun kullanılması

Hedef alt blok değerlerini elde etmek için en düşük SAD değerlerine sahip birden fazla kaynak alt bloğunun kullanımı incelenerek kodlama verimi üzerine etkisi araştırılacaktır. Birden fazla kaynak alt blok kullanırken bir ağırlandırma yapılmayacak ve alt blokların piksel değerlerinin ortalaması alınmaktadır. Ortalamaya dahil edilecek alt blokların şablonlarının (kaynak şablonu) hedef şablonla olan farkının belirli bir değerin üzerinde olmaması gerekmektedir.

Önceki çalışmalarda [14] elde edilen, birden fazla kaynak alt blok kullanımının kodlama kazancına etkisine ilişkin değerler Çizelge 4.15'de görülmektedir.

Çizelge 4.15 : Birden fazla kestirim alt bloğu kullanımına ilişkin önceki çalışmalarda elde edilen değerler

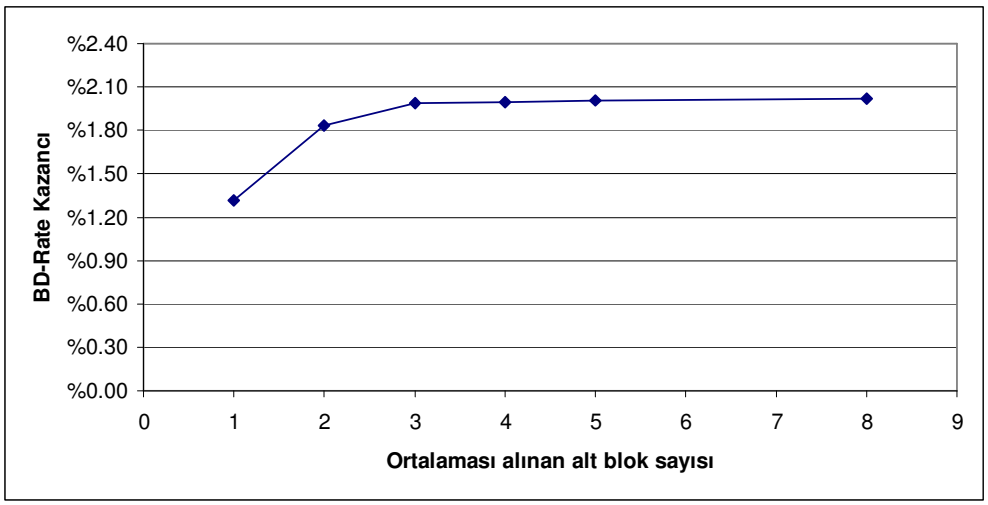
Video ismi	Tek kestirim alt bloğu ile BD-Rate kazancı (%)	Dört kestirim alt bloğu ile BD-Rate kazancı (%)
Mobile	0.67	1.10
Highway	-	-
Foreman	4.96	8.99
Ice	-	-
Paris	1.63	2.74
Tempete	0.52	1.55
Container	0.94	1.42

Bu çalışmada elde edilen kazanç değerleri ise Çizelge 4.16'de verilmiştir. Önceki çalışmalarla uyumluluğu korumak için dört alt bloğun kullanıldığı durumuna ilişkin sonuçlar verilmiştir. 1, 2, 3, 4, 5 ve 8 alt bloğun kullanımıyla elde edilen kazanç değerleri Çizelge A.2'de verilmiştir.

Çizelge 4.16 : Birden fazla kestirim alt bloğu kullanımıyla elde edilen kazanç değerleri

Video ismi	Tek kestirim alt bloğu ile BD-Rate kazancı (%)	Dört kestirim alt bloğu ile BD-Rate kazancı (%)
Mobile	0.30	0.35
Highway	4.34	5.93
Foreman	1.91	3.96
Ice	1.33	1.75
Paris	1.01	1.23
Tempete	0.15	0.39
Container	0.19	0.37

Bu uygulamada kullanılan alt blok sayısına göre kodlama verimliliğin değişimi ise Şekil 4.4'de gösterilmiştir. Burada diğer testlerde kullanılan videolar kullanılmıştır ve BD-Rate kazancı olarak gösterilen değerler, tüm videolar için elde edilen ortalama kazançtır.



Şekil 4.4 : Birden fazla alt bloğun ortalaması alınarak yapılan kestirimlerin BD-Rate kazançları

Önceki çalışmalarda kestirim için dört alt blok kullanılmasıyla BD-Rate kazancında ortalama %1.3 artış sağlanmıştır. Bu çalışmada ise dört alt blok kullanımı %0.7 artış sağlamaktadır. Buradan, implementasyonların farklı olmasına rağmen, birden fazla alt blok kullanımının her iki çalışmada da mutlak kazanç sağladığı görülmektedir.

4.3.8 Yarım ve çeyrek piksel hassasiyetle birden fazla kestirim alt bloğunun kullanılması

Buraya kadar yapılan incelemelerde elde edilen verilere göre, en fazla kodlama kazancı birden fazla kestirim alt bloğunun kullanılması durumunda elde edilmektedir. Ayrıca, tek alt blok kullanıldığı durumda arama bölgesinde yarım ve çeyrek piksel hassasiyetle şablon eşleme yapılması da, kodlama kazancına yüksek kazanç getirmektedir.

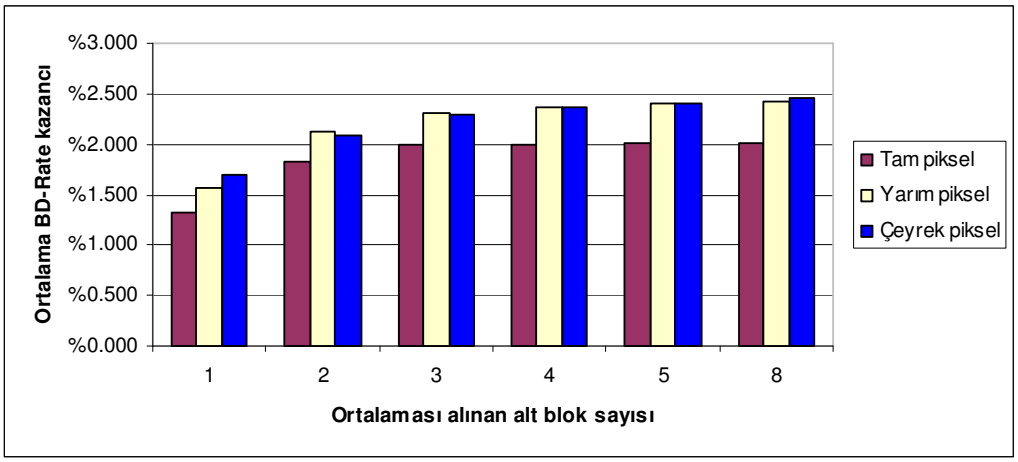
Bu sonuçlardan hareketle, kodlama kazancını artıran yöntemlerin bir araya getirilmesiyle kazancın daha yüksek seviyelere çekilebileceği öngörülebilir. Önermenin denenmesi için, yarım ve çeyrek piksel hassasiyetine çıkarılmış arama bölgelerinde, hedef şablonuna en yakın kaynak şablonlarından N adeti bir araya getirilerek kaynak alt blok belirlenmelidir. Bir araya getirmiş N adet yarım veya çeyrek hassasiyetli alt bloktan tek bir kaynak alt blok elde etmek için, bu blokların piksel bazında ortalamasının alınması gerekmektedir. Yani bir alt bloğun tek bir pikseli N adet alt bloğun aynı pozisyondaki piksellerinin (toplam N adet) ortalaması alınarak oluşturulur. Ortalama alınırken kaynak şablonun hedef şablona uzaklığıyla

(farklıyla) ağırlandırma yapılabilir. Bu çalışmada tüm alt bloklar aynı ağırlıkta kabul edilmiş ve eşit ağırlıklandırma kullanılmıştır.

Dikkat edilmesi gereken bir başka nokta da, ortalamaya dahil edilecek alt blokların şablonlarının (kaynak şablonu) hedef şablonla olan farkının belirli bir değerin üzerinde olmaması gerektiğidir. Kaynak şablon – hedef şablon farkını minimum yapan şablon belirlendikten sonra, ortalamaya dahil edilecek diğer altblokların şablonlarının hedef şablon ile farkı, bu minimum değerlerden en fazla tanımlanmış bir "v" değeri kadar sapabilmelidir. Şablonlarının sapma değeri tanımlanmış "v" değerinden daha büyük olan alt bloklar, ortalamaya dahil edilmezler.

Arama bölgesinin yalnızca tam piksellerden oluştuğu durumda, ortalamaya dahil edilecek bloklar için kullanılacak maksimum sapma değeri "v" 15 olarak seçilmiştir. Hatırlanacağı üzere, 2x2 lik alt blok için tam piksellerden oluşan bir şablonda toplam 5 piksel bulunmaktadır. Yarım piksellerin de dahil olduğu bir şablonda 9 (Şekil 3.6) , çeyrek piksellerin de eklendiği şablonda ise 17 piksel (Şekil 4.3) bulunmaktadır. Bu nedenle maksimum sapma değeri "v" yarım piksel hassasiyetli şablon eşleme için 30, çeyrek piksel hassasiyetli şablon eşleme için 50 olarak seçilmiştir.

Şekil 4.5'de tam, yarım ve çeyrek piksel kullanılarak yapılan şablon eşlemenin getirdiği kazanç değerleri verilmiştir. Ölçümler için ortalaması alınan blok sayısı 1, 2, 3, 4, 5 ve 8 olarak seçilmiştir. Tam ve çeyrek pikseller sadece 4x4 lük blokların 2x2 lik altbloklarının şablon eşlemesinde kullanılmıştır. 8x8 lik bloklar için yine tam piksel ile şablon eşleme gerçekleştirilmiştir. Nedeni ise, 8x8 lik bloklar için yarım piksel ile şablon eşleme yapılmasının fazla bir kazanç sağlamadığının ve kodlama zamanını 4 kat uzattığının önceki incelemelerde tespit edilmiş olmasıdır.



Şekil 4.5 : Tam, yarım ve çeyrek piksel hassasiyetleri ile birden fazla alt blok kullanarak şablon eşlemede elde edilen ortalama kazanç değerleri

Yarım ve çeyrek piksel hasiyetinde 1, 2, 3, 4, 5 ve 8 alt bloğun kullanımıyla elde edilen kazanç değerlerinin tamamı Çizelge A.3 ve Çizelge A.4'de verilmiştir. Elde edilen sonuçlardan görüleceği üzere, yarım piksellerin kullanıldığı ve N alt bloğun ortalamasının alındığı şablon eşleme yönteminde, tam piksellerin kullanıldığı duruma göre yaklaşık %0.4 kazanç sağlanmaktadır. Çeyrek piksellerin kullanıldığı durumda ise, sadece alt blokların ortalamasının alınmadığı durumda yarım piksele göre daha verimli kodlama mümkün olmaktadır. Ortalaması alınan alt blok sayısı N arttıkça da çeyrek piksellerin kullanıldığı kodlama performansında yarım piksellerinkine yaklaşmaktadır. Ancak çeyrek pikselle şablon eşleme yapmak için çeyrek piksele geçişte oluşturulan piksel sayısı yarım piksele geçişte oluşturulanın yaklaşık iki katıdır. Donanım seviyesinde gerçeklenmediği sürece, yarım piksele oranla önemli miktarda kodlama artışı sağlamadığı için, çeyrek pikselle şablon eşleme yapmak tercih edilmeyecektir.

Önceki çalışmalarda [14], birden fazla alt bloğun ortalamasının kullanımı ile şablon eşlemenin kodlama kazancına katkısı gösterilmiştir. Bu çalışmada da ayrıca yarım piksel kullanımının daha fazla kodlama kazancı elde edilmesini sağladığı gösterilmiş olmaktadır. Bu kazanç kodlanmakta olan videoya bağlı olarak değişmekle birlikte, test kriterleri içinde tanımlanan videoların tamamı için kazancı artırmaktadır.

YORUM VE ÖNERİLER

Bu çalışmada sadece çerçeve içi kodlama yapan video sıkıştırma teknikleri kullanarak ve doku sentezleme yöntemlerinden yararlanılarak kodlamayı iyileştiren bir yöntem optimize edilmeye çalışılmıştır. Şablon eşleştirme ile çalışan bu yöntem için kullanılabilecek parametreler tanımlanmış ve bu parametrelerin kodlama verimliliğine katkıları saptanmıştır.

Şablon eşleme ile çerçeve içi kodlama, görüntü kalitesini düşürmeden, videoyu kodlamak için harcanan bit miktarını azaltabilmektedir. Bu azalma bazı test videolarında %5'e kadar çıkmaktadır.

Yöntemde, eşleştirmenin yapıldığı alanın büyüklüğünün artırılması belirli bir düzeye kadar iyileşme sağlamaktadır. Seçilen alt blok büyüklüklerine bağlı olarak arama alanlarının ideal büyüklükleri belirlenebilmektedir. Kodlayıcı karmaşıklığı ile kodlama verimliğini dengeleyecek arama alanı büyüklükleri kodlayıcının işlem gücüne göre seçilmelidir.

H.264/AVC'de çerçeveler arası kestirim yapılırken kullanılan yarım ve çeyrek piksel hassasiyetiyle kaynak bloğun seçilmesi yöntemi, şablon eşleme tekniğinin de daha doğru sonuçlar vermesini sağlamaktadır. Bunun sonucu olarak da kodlayıcı verimliliği daha da artmaktadır. Yarım piksel hassasiyetle şablon eşleştirme yapılan testlerde, 4x4 bloklar için belirgin bir kazanç sağlanırken, 8x8 bloklar için yarım piksel hassasiyet kullanılması bit kazancını fazla artırmamaktadır ve kodlayıcıyı daha karmaşık hale getirmektedir. Yarım piksel hassasiyetin kullanılacağı durumlarda, arama bölgesinin çok büyük seçilmemesine dikkat edilmesi gerekmektedir.

Şablon eşleştirme işleminde şablonlar arası farkları SAD yerine SSD'ye göre almak daha doğru kestirimler yapılmasını sağlamaktadır. Fazladan bir maliyet getirmeyen SSD kriteri, çoğu görüntü işleme uygulamasında olduğu gibi şablon eşlemede de eşleştirme kriteri olarak seçilmelidir.

Kestirim bloğunu oluşturan alt blokların, arama bölgesi içerisindeki tek bir kaynak alt bloktan alınması yerine, şablonları benzeyen birden fazla alt bloğun ortalaması

alınarak belirlenmesi daha doğru kestirim sağlayabilmektedir. Kaynak alt blokların ortalaması alınırken ağırlıklandırma kullanmak da olumlu sonuçlar sağlayabilir, araştırılmalıdır.

Şablon eşlemesi yapılırken yarım ve çeyrek piksellerin kullanımı ise kodlama kazancını artırmaktadır. Ayrıca birden fazla alt blok kullanıldığı durumlarda yarım ve çeyrek piksellerin kodlama kazancına katkısı daha da fazla olmaktadır. Ölçümler sonucu elde edilen verilerde anlaşılmaktadır ki, çeyrek piksel kullanımı yarım piksel kullanımına göre kayda değer bir kazanç sağlamamaktadır. Ayrıca şablon eşleme karmaşıklığını ve dolayısıyla kodlama zamanını artırmaktadır. Bu nedenle sadece yarım piksellerin kullanıldığı şablon eşleme yöntemi yeterlidir ve ortalama %0.4 kazanç sağlamaktadır. Bazı test videolarında ise bu artış %1 mertebesine ulaşmaktadır.

Her ne kadar şablon eşleme kodlayıcı karmaşıklığını artırsa da, genelde kullanıcı cihazlarında olduğu gibi donanımsal kodlayıcıların kullanımıyla, kestirim işlemleri donanım içerisindeki özelleşmiş bloklar tarafından yapılabilmekte ve kodlama zamanını artırmamaktadır. Bu da şablon eşlemenin tek maliyetinin kodlayıcı ve kod çözücü yapısı üzerinde olacağı anlamına gelmektedir, kodlama süresi artışı beklenmemelidir.

Şablon eşleme, hem kodlayıcı hem de kod çözücüde, çerçeve içi kestirim karmaşıklığını artırarak bit iletim oranını düşürmektedir. Kodlayıcı ve kod çözücüde aynı anda sahip olunan bilgilerden yararlanarak, bu iki sistem arasındaki bilgi iletimini azaltmak H.264/AVC gibi modern standartların hedefleri arasındadır. Ancak kestirim bloklarının seçimi için kodlayıcı ve kod çözücüde birebir aynı şekilde yapılacak işlemlerin kullanımı pek yaygın değildir. Çerçeveler arası kestirimde yapıldığı gibi kestirim moduna ilaveten kodlayıcının hareket vektörlerini göndermesi daha çok uygulanan bir yöntemdir. Şablon eşlemede olduğu gibi, her iki tarafın da aynı kararı verebileceği kestirim yöntemleri geliştirmek, bit iletim oranını düşürmeye yardımcı olacaktır.

KAYNAKLAR

[1] **Wiegand, T., Sullivan, G. J., Bjøntegaard, G., and Luthra, A.**, 2003: Overview of the H.264/AVC video coding standard, *IEEE Trans. Circuits Syst. Video Technol.*, Vol **13** , pp. 560-576.

[2] **Wiegand, T., Schwarz, H., Joch, A., Kossentini, F.,and Sullivan, G.J.**, 2003: Rate-Constrained Coder Control and Comparison of Video Coding Standards, *IEEE Trans. Circuits Syst. Video Technol.*, Vol **13**, pp. 688-703.

[3] **Richardson, Iain E.G.,** 2003: H.264 and MPEG-4 Video Coding, John Wiley & Sons Ltd, UK

[4] ITU-T Recommendatiton H.264 and ISO/IEC 11496-10 (MPEG-4, "AVC-Advanced Video Coding for Generic Audiovisual Services", Versiyon 3

[5] **Malvar, H., Hallapuro, A., Karczewicz, M. and Kerofsky, L.,** 2003: Low-Complexity Transform and Quantization in H.264/AVC, *IEEE Trans. Circuits Syst. Video Technol.*, Vol **13**, pp. 598-603.

[6] **Marpe, D., Wiegand, T. and Gordon, S.,** 2005: H.264/MPEG4-AVC Fidelity Range Extensions: Tools, Profiles, Performance and Application Areas, *IEEE International Conference on Image Processing*, Vol **1**, ICIP, pp. 593-596

[7] **Marpe, D., Schwarz, H. and Wiegand, T.,** 2003: Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standart, *IEEE Trans. Circuits Syst. Video Technol.*, Vol **13**, pp. 620-636

[8] **Sullivan, G., and Wiegand, T.,** 1998: Rate-Distortion Optimization for Video Compression, *IEEE Signal Processing Magl.*, Vol **15**, pp. 74-90

[9] **Wei, Z., Ngan, K. and Li, H.,** 2008: An efficient intra-mode selection algoritm for H.264 based on edge classification and rate-distortion estimation, Elsevier *Signal Processing: Image Communication 23*, pp. 699-710

[10] **Bjontegaard, G.,** 1997: Coding improvement by using 4x4 blocks for motion vectors and transform, *VCEG SG16/Q15, ITU-T Doküman Q15-C23*, Eibsee, Almanya

[11] **Wei, L. and Levoy, M.,** 2000: Fast Texture Synthesis using Tree-structured Vector Quantization, *Proceeding of SIG-GRAPH 2000*, pp. 479-488

[12] **Tan, T., Boon, C. and Suzuki, Y.,** 2006: Intra Prediction By Template Matching, *Proc. ICIP 2006*, Atlanta

[13] JM 14.1 H.264/AVC Referans Yazılımı, *http://iphome.hhi.de/suehring/tml/download/*

[14] **Tan T., Boon C. and Suzuki Y.,** 2007: Intra prediction by averaged template matching predictors, *CCNC*, IEEE

[15] **Zheng, Y., Yin, P., Escoda, O., Li, X. and Gomila, C.,** 2008: Intra prediction using template matching with adaptive illumination compensation, *Proc. ICIP 2008*, California

[16] **Tan, T., Sullivan G., and Wedi T.,** 2005: Recommended Simulation Common Conditions for Coding Efficieny Experiments, *ITU-T / Study Group 16 / Question 6 – VCEG, document VCEG-AA10*, Nice,

[17] **Bjontegaard, G.,** 2001: Calculation of average PSNR differences between RD-curves, ITU-T SC16/Q6, 13th VCEG Meeting, document VCEG-M33, Texas

[18] AVSNR4, BD-Rate Ölçüm Yazılımı, *http://wftp3.itu.int/av-arch/video-site/h26L/avsnr4.zip*

EKLER

EK A.1 : Bjontegaard Delta Rate (Bit Oranı Farkı) Ölçümü

EK A.1

Bjontegaard tarafından önerilen BD-Rate ölçümünü yapmaktaki temel amaç, aynı ölçüm şartlarına sahip ancak farklı parametrelerle yapılan iki test sonucu arasındaki bit oranı ve PSNR değişimini gözlemleyebilmektir. İki test sonucuna ait bit oranı-bozulma eğrileri kullanılarak, aynı PSNR değerleri için elde edilecek olan bit oranı kazancı bulunmaya çalışılır. Bu yöntem H.264/AVC standardı geliştirilirken, standarda eklenen özelliklerin objektif testi için de kullanılmıştır.

Yöntemde, her iki test sonucundan 4 ortak nokta seçilir ve bu ortak noktalar arasındaki bit oranı ve PSNR değişimleri göz önüne alınarak, iki test sonucu arasındaki ortalama bit oranın değişimi bulunur. 4 ortak noktanın elde edilmesi için genelde uygulanan yöntem, her iki testin de 4 farklı kuantalayıcı parametresi (QP) kullanılarak yapılmasıdır. Seçilen 4 nokta (her bir test sonucu için) bit oranı - PSNR eksenleri olan bir grafiğe oturtulur. PSNR değerleri gibi, bit oranı değerleri de logaritmik olarak ifade edilir.

Bu işlemlerden sonra seçilen dört noktadan geçecek şekilde eğriler bulunur. Bu eğriler 3. dereceden denklemlerle ifade edilirler (A.1).

$$SNR = a + b \times bit + c \times bit^2 + d \times bit^3 \qquad \textbf{(A.1)}$$

Hem referans ölçümün, hem de denenmekte olan yöntemin hesaplanan eğrileri üzerinde aralık boyunca entegral alınır. Her iki eğirinin entegralinin farkı, entegral aralığına (8 noktadan en büyüğü ile en küçüğünün farkı) bölünür. Bulunan değer yüzde olarak ifade edilerek, % BD-Rate değişimi elde edilmiş olur.

EK A.2 : Detaylı ölçüm sonuçları

EK A.2

Çizelge A.1 : Arama bölgesi büyüklüklerine karşı gelen %BD-Rate kazançları

Test Videosu	16x12 24x18	16x12 32x24	18x14 26x20	18x14 36x28	20x16 30x24	20x16 40x32	24x20 36x30	24x20 48x40
Mobile	0.278	0.298	0.302	0.301	0.295	0.293	0.279	0.288
Highway	4.188	4.341	4.314	4.429	4.411	4.438	4.472	4.404
Foreman	1.870	1.908	2.041	2.074	2.070	2.065	2.196	2.209
Ice	1.344	1.330	1.402	1.309	1.339	1.393	1.386	1.403
Paris	0.997	1.005	1.046	1.047	1.049	1.017	0.935	0.904
Tempete	0.133	0.149	0.163	0.134	0.162	0.105	0.150	0.168
Container	0.189	0.185	0.190	0.228	0.208	0.262	0.227	0.198
Ortalama	1.285	1.317	1.351	1.360	1.362	1.368	1.378	1.368

EK A.2

Çizelge A.2 : Birden fazla alt bloğun ortalaması alınarak yapılan kestirimlerin % BD-Rate kazançları (tam piksel)

Test Videosu	1	2	3	4	5	8
Mobile	0.298	0.322	0.332	0.350	0.324	0.334
Highway	4.341	5.603	5.950	5.926	6.021	6.031
Foreman	1.908	3.431	3.803	3.964	3.960	4.040
Ice	1.330	1.687	1.889	1.754	1.856	1.752
Paris	1.005	1.187	1.235	1.230	1.218	1.244
Tempete	0.149	0.289	0.350	0.389	0.346	0.404
Container	0.185	0.306	0.364	0.356	0.320	0.318
Ortalama	1.317	1.832	1.989	1.996	2.006	2.018

EK A.2

Çizelge A.3 : Yarım piksel hassasiyeti ile birden fazla alt blok kullanarak şablon eşlemede elde edilen % BD-Rate kazanç değerleri

Test Videosu	1	2	3	4	5	8
Mobile	0.309	0.359	0.392	0.381	0.384	0.397
Highway	4.785	6.135	6.544	6.620	6.765	6.799
Foreman	2.574	4.101	4.738	4.933	5.012	5.092
Ice	1.748	2.164	2.331	2.311	2.406	2.347
Paris	1.106	1.312	1.416	1.426	1.448	1.421
Tempete	0.221	0.384	0.441	0.454	0.471	0.465
Container	0.269	0.371	0.362	0.450	0.388	0.465
Ortalama	1.573	2.118	2.318	2.368	2.410	2.426

Çizelge A.4 : Çeyrek piksel hassasiyeti ile birden fazla alt blok kullanarak şablon eşlemede elde edilen % BD-Rate kazanç değerleri

Test Videosu	1	2	3	4	5	8
Mobile	0.369	0.376	0.393	0.422	0.400	0.439
Highway	4.919	5.811	6.247	6.482	6.558	6.616
Foreman	2.745	3.933	4.436	4.675	4.752	4.957
Ice	2.032	2.260	2.540	2.516	2.568	2.562
Paris	1.284	1.420	1.466	1.540	1.550	1.588
Tempete	0.293	0.405	0.479	0.524	0.525	0.557
Container	0.274	0.364	0.461	0.431	0.451	0.477
Ortalama	1.702	2.081	2.289	2.370	2.400	2.457

EK B.1 : JM 14.1 H.264/AVC kodlayıcı yazılımı (CD'de)

ÖZGEÇMİŞ

Ad Soyad : Altay ERŞAHİN

Doğum Yeri ve Tarihi : İstanbul – 24.11.1984

Adres : Havacı Binbaşı Mehmet Sok. No:16 D:1
Suadiye İstanbul

Lisans Üniversite : Yıldız Teknik Üniversitesi
Elektronik ve Haberleşme Mühendisliği

Altay Erşahin 1984 yılında İstanbul'da doğdu. 2002 yılında liseden mezun olduktan sonra Gebze Yüksek Teknoloji Enstitüsü Elektronik Mühendisliği'nde lisans eğitimine başladı. 2004 yılında Yıldız Teknik Üniversitesi Elektronik ve Haberleşme Mühendisliği'ne geçiş yaptı. 2006 yılında lisans eğitimini tamamladıktan sonra İstanbul Teknik Üniversitesi'nde Telekomünikasyon Mühendisliği yüksek lisans programına kayıt oldu. Halen aynı anabilim dalında eğitimine devam etmektedir.

2006 yılından bu yana Beko Elektronik A.Ş.'de Yazılım Tasarım Mühendisi olarak çalışmaktadır. Bu süre zarfında gömülü sistemler üzerinde çalışan sayısal yayın (DVB) alıcılarla ilgili projelerde görev almıştır. Video kodlama, sayısal yayıncılık, ağ haberleşmesi ve gerçek zamanlı işletim sistemleri (RTOS) ilgi duyduğu alanlardır.