T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

GÖRÜNTÜ SIKIŞTIRMA YÖNTEMLERİNİN ETKİNLİĞİNİ ARTTIRAN DÖNÜŞÜM VE BÖLÜMLENDİRME İŞLEMLERİ

Emir ÖZTÜRK

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

YRD. DOÇ. DR. Altan MESUT

EDİRNE 2012

T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

GÖRÜNTÜ SIKIŞTIRMA YÖNTEMLERİNİN ETKİNLİĞİNİ ARTTIRAN DÖNÜŞÜM VE BÖLÜMLENDİRME İŞLEMLERİ

Emir ÖZTÜRK

Yüksek Lisans Tezi

Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Yrd. Doç. Dr. Altan MESUT

EDİRNE 2012

T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

GÖRÜNTÜ SIKIŞTIRMA YÖNTEMLERİNİN ETKİNLİĞİNİ ARTTIRAN DÖNÜŞÜM VE BÖLÜMLENDİRME İŞLEMLERİ

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

Emir ÖZTÜRK

Bu tez ...../...../..... tarihinde Aşağıdaki Jüri Tarafından Kabul Edilmiştir.

(İmza)	(İmza)	(İmza)
.....	.....	.....
Yrd. Doç. Dr. Altan MESUT	Yrd. Doç. Dr. Deniz TAŞKIN	Yrd. Doç. Dr. Erdinç UZUN

Yüksek Lisans Tezi

Görüntü Sıkıştırma Yöntemlerinin Etkinliğini Arttıran Dönüşüm Ve Bölümlendirme İşlemleri

T.Ü. Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

ÖZET

İki boyutlu durağan görüntülerin sayısal ortamda saklanırken kayıplı veya kayıpsız sıkıştırma yöntemlerinin kullanılmasıyla daha az yer kaplamalarını sağlamak mümkündür. JPEG, JPEG2000 ve JPEG XR gibi kayıplı görüntü sıkıştırma standartları, renk uzayı dönüşümü, uzamsal etki alanından frekans etki alnına dönüşüm, niceleme işlemi ve entropi kodlaması gibi aşamalarının her birinde farklı yaklaşımları temel aldıkları için birbirlerinden farklı hızlarda ve farklı kalite oranlarında sıkıştırma yapmaktadırlar. Bu yöntemler genellikle fotoğraf türündeki karmaşık görüntülerin sıkıştırılması amacıyla kullanılırken, GIF ve PNG gibi kayıpsız görüntü sıkıştırma yöntemleri ise karmaşıklığın daha az olduğu (entropi oranı daha düşük olan) görüntülerde kullanılmaktadırlar. Düşük karmaşıklığa sahip olan görüntülerin kayıplı bir yöntem ile sıkıştırılması hem sıkıştırma oranı olarak kayıpsız yöntemlerin gerisinde kalmakta hem de görüntüdeki bozulma insan gözü tarafından daha kolay algılanabilmektedir. Bu nedenle JPEG2000 standardı kendi bünyesinde kayıpsız bir sıkıştırma yöntemi de barındırmakta ve eğer istenilen sıkıştırma oranına kayıpsız olarak erişebileceğini öngörür ise kayıpsız yöntemi kullanmayı tercih etmektedir.

Bu tez çalışmasının amaçlarından birincisi; sayısal görüntülerin farklı bir biçimde saklanması ile kayıpsız olarak daha yüksek oranda sıkıştırılabilmesinin mümkün olup olmadığını araştırmaktır. Tezin diğer bir amacı ise; bir görüntüyü farklı şekillerde bölümlendirerek, bu bölümlerin karmaşıklığına göre kayıplı veya kayıpsız sıkıştırma yapan

bir yöntemin tercih edilmesiyle, görüntünün tamamının kayıplı bir yöntem ile sıkıştırılmasına göre daha yüksek sıkıştırma oranına ve/veya daha yüksek görüntü kalitesine (düşük kayıplı) ulaşmaktır.

İkinci bölümde durağan dosyaların saklanmasında kullanılan yöntemler yer alırken, üçüncü bölümde tez kapsamında deneylerini yaptığımız farklı dönüşüm ve bölümlendirme işlemlerinin açıklamaları yer almaktadır. Dördüncü bölümde ise bu deneylerden elde ettiğimiz sonuçlar verilmiştir.

Yıl : 2012

Sayfa Sayısı : 71

Anahtar Kelimeler : Veri Sıkıştırma, Görüntü dönüşümü, görüntü bölümlendirme, JPEG, JPEG2000, JPEG XR, PNG, LZMA

MS (MASTER) Thesis

Transformation and Segmentation Operations on Images for Increasing the Effectiveness of Compression Methods

Trakya University Institute of Natural Sciences

Computer Engineering Department

ABSTRACT

It is possible to shrink (reduce the size of) two dimensional still images with lossless or lossy compression methods. Standards like (as) JPEG, JPEG2000 and JPEG XR based on different approaches in color-space transform, conversion between the spatial domain and the frequency domain, quantization and entropy encoding. Because of that, they compress images in different size and time. While these methods especially are used to compress photos, methods like GIF and PNG are used in images with low complexity (low entropy rate). Compressing images with low complexity with lossy compression method is both staying behind lossless compression methods and the corruption of image could be detected easily by human eye. Because of that JPEG2000 standard embraces a lossless compression method and if it predicts that lossless compression method can access the desired compression ratio, it prefers to use the lossless method.

First aim of this thesis is to investigate whether it is possible to use lossless compression and get higher compression ratios by storing digital images in different ways. Another aim of the thesis is acquiring higher compression rates and/or better visual quality (with low-loss) than compressing the whole image with a lossy compression method with the help of splitting the image with different ways and preferring a lossy or lossless compression method according to complexity of image pieces.

The methods of storing still images are given in the second chapter. The third chapter includes explanation of various transform and segmentation methods we made experiments on. The fourth chapter involves the results of the experiments we made.

Year : 2012

Number of Pages **:** 71

Keywords **:** Data Compression, Image Transformation, Image Segmentation, JPEG, JPEG2000, JPEG XR, PNG, LZMA

TEŞEKKÜR

Lisans eğitimimden yüksek lisans eğitimimin sonuna kadar her zaman bana yardım eden, her konuda desteğini ve tavsiyelerini benden esirgemeyen, istisnasız her durumda beni anlayışla karşılayan ve bana çözüm sunan danışman hocam Yrd. Doç. Dr. Altan MESUT'a teşekkür ederim.

Ayrıca tez jürisinde yer aldıkları için ve ders aşamasında bana verdikleri eğitim için Yrd. Doç. Dr. Deniz TAŞKIN ve Yrd. Doç. Dr. Erdinç UZUN'a, lisans eğitimimden bu yana desteklerini ve bilgilerini benden esirgemedikleri için de bölüm hocalarıma teşekkür ederim.

Her zaman yanımda olan, her durumda bana yardım eden, beni daha rahat bir yaşama kavuşturmak için kendilerinden fedakârlık gösteren ve bunu yılmadan bunca senedir yapan anneme ve babama minnettar olup, teşekkür etmeyi bir borç bilirim.

Yoğun ve gergin anlarımda anlayışları biraz olsun eksilmeyen, beni her zaman dinleyen ve ellerinden gelen yardımı benden esirgemeyen Arş. Gör. Nazan DEMİRCİ, Arş. Gör. Turgut DOĞAN, Arş. Gör. Mustafa ARDA ve Abdullah ÇELİKÇİ'ye teşekkür ederim.

Son olarak, verdiği burs karşısında sadece bir teşekkürü kâfi gören TÜBİTAK'a ve proje desteği için Trakya Üniversitesi Bilimsel Araştırma Projeleri Birimi'ne de teşekkürlerimi sunarım.

İÇİNDEKİLER

Tablo Listesi

Şekil Listesi

BÖLÜM 1

GİRİŞ

Sayısal görüntüler genellikle her bir pikselinin Kırmızı (Red), Yeşil (Green) ve Mavi (Blue) bileşenleri (kanalları) ayrı ayrı belirli bir bit oranında nicelenmesi ile saklanırlar (Örn: her biri 8-bit ile nicelenirse, bir piksel için toplam 24-bit saklama alanı kullanılır). Piksellerin dosyada saklanma sırası genellikle görüntünün sol üst noktasından sağ alt noktasına doğru, ya da sol alt noktadan sağ üst noktaya doğru (Örn: BMP) satır öncelikli olur. Görüntünün belirli kısımlarının karmaşıklık düzeyine göre bölümlendirilmesi ile, satır veya sütun öncelikli piksel saklama sırasının zig-zag tarama düzeni ile değiştirilmesi ile, veya kanalların farklı bölümler olarak düşünülüp ayrı ayrı ele alınması ile bilinen kayıpsız sıkıştırma yöntemleri bu dosyaları daha iyi oranda sıkıştırabilmektedir. Bu yöntemler arasında en çok bilinenleri: LZ77'nin [Ziv, Lempel, 1977] uyarlaması olan LZMA, LZ77 ve Huffman [Huffman, 1952] kodlamasını birlikte kullanan Deflate [Deutsch, 1996], PPM'in [Cleary, Witten, 1984] uyarlaması olan PPMd ve BWT [Burrows, Wheeler, 1994] dönüşümünü kullanan Bzip2'dir [Seward, 1996].

Tez kapsamında resimdeki piksellerin hangi sıra ile saklandığının ve bileşenlerinin (R, G ve B kanallarının) birlikte ya da ayrı ayrı saklanmasının yukarıda verilen kayıpsız sıkıştırma yöntemlerinin sıkıştırma oranına etkisi incelenmiştir. Daha sonraki aşamada sayısal bir görüntüyü karmaşıklık düzeyine göre bölümlendirme ve her bir bölümün karmaşıklık düzeyine göre uygun olan bir sıkıştırma yöntemi ile sıkıştırılması üzerine çalışmalar yapılmıştır.

Daha önce yapılan çalışmalar göstermiştir ki; birbirini takip eden piksellerin büyük oranda aynı olduğu (düşük karmaşıklıkta) şekil, çizim, yazı, tablo, vs. türü görüntülerin GIF [CompuServe, 1987] ve PNG [ISO/IEC, 2004] gibi kayıpsız sıkıştırma yapan yöntemlerle, yüksek karmaşıklığa sahip fotoğraf türü görüntülerin ise JPEG [ISO/IEC, 1994], JPEG2000 [ISO/IEC, 2000] ve JPEG XR [ITU-T, 2009] gibi kayıplı sıkıştırma yapan yöntemlerle sıkıştırılması daha uygun olmaktadır (JPEG2000 ve JPEG XR ile istenirse kayıpsız sıkıştırma da yapılabilmektedir) [Mesut, 2006] [Santa-Cruza vd., 2000].

Kullanıcıların çoğu her türde sayısal görüntüyü JPEG ile sıkıştırarak saklamayı tercih etmekte, bu da düşük karmaşıklıkta olan görüntüler için hem daha kötü bir sıkıştırma oranı, hem de orijinal kalitede görüntü yerine daha bozuk (kayıplı) bir görüntü anlamına gelmektedir. Tez kapsamında geliştirilen ve adına Tahmin Algoritması-3 dediğimiz yöntem sayesinde bir sayısal görüntüyü saklamak için hangi sıkıştırma yönteminin seçilmesi gerektiği kullanıcıların karar vermesi gereken bir mesele olmayacak, görüntünün tamamının kayıplı, tamamının kayıpsız veya belirli kısımlarının kayıplı diğer kısımlarının kayıpsız sıkıştırılmasının sıkıştırma oranı açısından daha faydalı olacağına bu yöntem karar verebilecek ve en uygun olanı ile sıkıştırma yapacaktır.

BÖLÜM 2

DURAĞAN GÖRÜNTÜLERİN SAYISAL ORTAMDA SAKLANMASI

BMP ve TIFF (Tagged Image File Format) gibi 2 boyutlu görüntülerin sayısal ortamda saklanması amacıyla kullanılan dosya biçimleri, ilk geliştirildiklerinde herhangi bir veri sıkıştırma işlemini kullanmazlarken, sonraki yıllarda RLE ve LZW gibi bazı veri sıkıştırma algoritmalarını kullanabilir hale gelmişlerdir. 1980'li yıllarda LZW [Welch, 1984] veri sıkıştırma algoritmasını temel alarak geliştirilen GIF ve 1990'lı yıllarda Deflate algoritmasını temel alarak geliştirilen PNG görüntü dosya biçimleri de, sayısal görüntülerin kayıpsız olarak sıkıştırılması amacıyla sıklıkla kullanılmaktadır.

Fotoğraf gibi karmaşıklık oranı yüksek olan görüntülerin kayıplı ve kayıpsız olarak sıkıştırılması konusunda çalışan Joint Photographic Experts Group tarafından geliştirilen ve her biri ayrı bir standart olarak yayınlanan JPEG, JPEG-LS [ISO/IEC, 1994], JPEG2000 ve JPEG XR gibi karmaşık yöntemler de mevcuttur.

Bu bölümde en çok kullanılan görüntü saklama biçimlerinden BMP'nin ve PNG'nin yapılarından kısaca bahsedilecek, kayıplı görüntü sıkıştırma standartları olan JPEG, JPEG2000 ve JPEG XR hakkında ise daha detaylı bilgi verilecektir.

2.1. Bitmap dosyası

Bitmap, sayısal resimleri saklamak için en çok kullanılan dosya biçimlerinden biridir. 1, 4, 8, 16, 24 ve 32 bpp (bit per pixel – piksel başına bit sayısı) renk derinliğini destekler. 1 bpp saklanan bitmap resimlerde her pikselin siyah mı yoksa beyaz mı olduğu saklanır. Eğer renk tablosu oluşturulmuşsa 1 bit seçilen iki rengi de belirtebilir. 8 bpp bitmap dosyasında 256 farklı renk desteklenir. Dosya başlığından sonra bir renk tablosu saklanır. Bu renk tablosunda resim içerisinde kullanılan renk değerleri indekslenir. Renk tablosunda renkler 3 byte (8.8.8) veya 4 byte (8.8.8.8) ile ifade edilebilir. 16 bpp ve üzeri resimlerde renk tablosunun kullanılmasına gerek yoktur. Ancak sınırlı renk gösterebilme kapasitesine sahip cihazların optimizasyonu için kullanılabilmesi mümkündür. 24 bpp

dosyada her kanal (R, G veya B) 1 bayt (8 bit) ile temsil edilir. 32 bpp resimlerde ise 4.294.967.296 farklı renk desteklenmektedir. R, G, B kanallarına ek olarak A (alpha – saydamlık) kanalı bulunur. Her kanal için 8 bit ayrılabileceği gibi, her kanala farklı büyüklükte bit dizileri ayrılabilir. Bitmap dosyasında pikseller sol alt köşeden sağ üst köşeye doğru ve B, G, R kanal sıralaması ile saklanır. Dosyanın başında 54 bayt başlık ve seçimlik 1024 bayt renk tablosu bulunur. 54 bayt başlık içerisinde ilk iki bayt "BM" karakterlerini içerir. Sonraki 4 bayt resmin bayt cinsinden boyutunu gösterir. Bir sonraki 4 bayt resmi oluşturan programın kullanması için ayrılmıştır. 18-22. ve 22-26. baytlar arasındaki veriler sırasıyla resmin enini (width) ve boyunu (height) gösterir. Geriye kalan bayt değerleri ise resmin kaç bpp olduğu, sıkıştırma türü, çözünürlük değerleri gibi verileri içerir (Tablo 2.1).

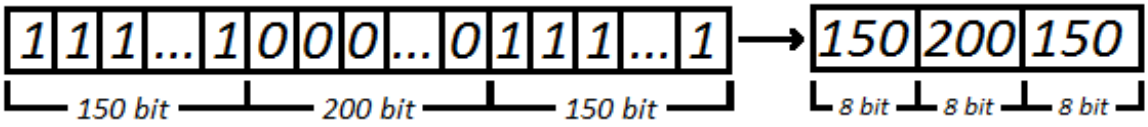
Tablo 2.1. Bitmap Başlığı

Boyut	**İçerik**
2 bayt	"BM" karakterleri
4 bayt	Resmin bayt cinsinden boyutu
2 bayt	Ayrılmış alan
2 bayt	Ayrılmış alan
4 bayt	Resim verisinin başlangıç adresi
4 bayt	BITMAPINFOHEADER yapısının boyutu
4 bayt	Resmin eni
4 bayt	Resmin boyu
2 bayt	Yüzey (plane) sayısı
2 bayt	Resmin her pikselinin kaç bit olduğu (bpp)
4 bayt	Sıkıştırma tipi (RLE-8, RLE-4)
4 bayt	Resim verisinin boyutu (boşluklar dâhil)
4 bayt	Resmin dikey çözünürlüğü
4 bayt	Resmin yatay çözünürlüğü
4 bayt	Resimdeki renk sayısı
4 bayt	Resimdeki önemli renk sayısı

BMP başlığında sıkıştırma tipi olarak belirtilen RLE (Run Length Encoding), basit bir veri sıkıştırma yöntemidir. Yöntemde tekrarlı veriler tekrarlanma sayıları ile beraber

kodlanır ve sıkıştırma sağlanır. Bir bit ile ifade edilen siyah beyaz bir resimde satır sıralı okuma yapıldığında 150 adet beyaz piksel, 200 adet siyah piksel ve ardından tekrar 150 adet beyaz piksel olduğu bir durumda bu resmin RLE ile kodlanmış hali 150, 200, 150 olacaktır. Bu sayede 500 bit ile ifade edilebilecek bir resim 3 Bayt (24 bit) ile ifade edilir ve sıkıştırma sağlanmış olur (Şekil 2.1).

RLE'nin avantajlı olduğu resimler, renk geçişinin ve detayın az olduğu düz zemine sahip resimlerdir. Bu tür resimlerde veri tekrarı fazla olduğu için her kanalda sıkıştırma sağlanır. Ancak eğer resimde birbirinden farklı sembol (piksellerin RGB değerleri) sayısı fazla ise veya ardışık piksellerin sayısal değerleri birbirinden farklı ise RLE dezavantaja dönüşür, sıkıştırma yapmak yerine verinin miktarını arttırır.



Şekil 2.1. RLE öncesi ve sonrası

2.2. PNG (Portable Network Graphics)

PNG, kayıpsız resim sıkıştırma algoritmasıdır. Patent ihtiyacı olan GIF'e rakip olarak geliştirilmiştir. 1995 yılında Thomas Boutell, Scott Elliott, Mark Adler, Tom Lane ve bir çok başka girişimcinin başlattığı PNG projesi 1996'da sonuçlanmıştır. 1999'da yayınlanan 1.2 sürümü ile birlikte kullanım oranı hızla artmıştır [Randers-Pehrson, 1999].

PNG altyapısında DEFLATE veri sıkıştırma yöntemi yer alır. 24 bpp RGB ve 32 bpp RGBA resimleri desteklediği gibi, gri tonlu resimler için de saydamlık kanalı olup olmaması farketmeksizin destek sunar. İnternet üzerinden resim iletimi için tasarlanması sebebiyle CMYK gibi profesyonel uygulamaların ihtiyaç duyacağı renk uzaylarını desteklemez.

PNG dosyasının 8 baytlık başlığı bulunur. İlk bayt 8 bit veriyi desteklemeyen iletim sistemlerinin tespiti için veya resmin metin olarak işlenmesini engellemek için kullanılır. Daha sonra gelen üç bayt P, N ve G harflerinin ASCII kodlarıyken sonraki iki bayt DOS

için satır sonu kodunu içerir. Kalan iki baytın ilki DOS için dosya sonu karakteri olarak kullanılırken, son bayt ise UNIX için satır sonu kodunu içerir (Tablo 2.2).

Tablo 2.2. PNG Başlığı

Boyut	**İçerik**
1 bayt	8 bit destek kontrolü veya metin olarak işlenme engeli
3 bayt	PNG harflerinin ASCII kodları
2 bayt	DOS satır sonu kodu
1 bayt	DOS dosya sonu kodu
1 bayt	UNIX satır sonu kodu

PNG formatının temel özellikleri şunlardır:

- Basit ve taşınabilir yapı: Geliştiriciler PNG'yi kolayca uygulayabilirler.
- Patent koruması altında değildir.
- Hem sıralı-renk hem de gerçek-renk görüntülerde diğer algoritmalar kadar etkili (hatta çoğu durumda daha etkili) sıkıştırma oranı sağlar.
- Esneklik: Geliştirmelere ve eklentilere izin veren bir yapısı vardır.

PNG'nin GIF ve TIFF'e göre üstün olduğu noktalar şunlardır:

- Piksel başına 48-bit renk derinliğine kadar renkli ve 16-bit renk derinliğine kadar gri-tonlamalı görüntülerle çalışabilme yeteneği vardır.
- Tam alfa kanalı: Genel şeffaflık maskeleri vardır.
- Görüntünün gamma bilgisi: görüntülerin doğru bir parlaklık/karşıtlık(contrast) ile otomatik olarak gösterilmesini sağlar.
- Dosya hatalarında güvenilir tespit yapabilme yeteneğine sahiptir.
- Kademeli aktarımda daha hızlı ön gösterim yapabilme yeteneğine sahiptir.

2.3. JPEG Mimarisi

JPEG, Joint Photographic Experts Group tarafından 1992 yılında geliştirilmiş bir standarttır. Geliştirilen standart, sıkıştırma ve açma aşamalarını tanımlar fakat dosya formatını tanımlamaz. JPEG, günümüzde hala yaygın olarak kullanılmaktadır. Kayıplı bir sıkıştırma yöntemi olduğundan resmin ne kadar sıkıştırılacağı, bir kalite faktörü verilerek

belirlenebilir. Kalite faktörünün düşük olması, resmin daha çok sıkışmasını sağlar fakat resimdeki kalite düşüşü de sıkıştırma miktarına bağlı olarak artar.

JPEG sıkıştırma aşamaları şu şekildedir: Renk uzayı dönüşümü, örneklemeyi azaltma, bloklara ayırma, ayrık kosinüs dönüşümü, niceleme, köşegen tarama, RLE ve QM kodlama. Bu aşamaların ayrıntıları alt başlıklarda verilmiştir.

2.3.1. Renk Uzayı Dönüşümü

JPEG, kodlamanın ilk aşamasında resmi RGB renk uzayından YUV uzayına dönüştürür. YUV uzayında Y parlaklığı ifade ederken, U ve V (mavi ve kırmızı olmak üzere) renkliliği ifade eder. RGB'den YCbCr uzayına dönüşüm işlemi sayesinde sıkıştırma miktarı artar. RGB uzayından Y, Cb ve Cr bileşenlerinin elde edilmesi için kullanılan formüller aşağıda verilmiştir.

$$Y = 0.299R + 0.587G + 0.114B$$
$$U = -0.147R - 0.289G + 0.436B$$
$$V = 0.615R - 0.515G - 0.100B$$

2.3.2. Örneklemeyi Azaltma

İnsan gözü, renk ve parlaklığa duyarlı reseptörlerin sayısındaki fark nedeniyle, bir resmin parlaklığında renkten daha çok detay algılar. Örneklemeyi azaltma aşamasında YCbCr uzayındaki bir resmin Cb ve Cr bileşenlerinin azaltılması, görüntüyü insan gözü için çok fazla farklılaştırmayacaktır. Bileşenlerin azaltılmasıyla, görüntüdeki fark sezilmeden saklanacak veri miktarı azaltılmış olur. 2x2 boyutunda 4 piksellik bir blok için 4 adet Y bileşeninin tamamı saklanırken Cb ve Cr bileşenlerinin ortalamaları kullanılarak 2'şer adedi (4:2:2) veya 1'er adedi (4:1:1) kullanılabilir.

2.3.3. Bloklara Ayırma

JPEG standardı, sıkıştırma yapmak için öncelikle resmi 8x8'lik bloklara böler. Kullanılan dönüşüm, niceleme ve kodlama aşamaları bu bloklar üzerinde uygulanır. Her blok birbirinden bağımsız olarak işlenir. Bu sayede sıkıştırma hızının yüksek olması sağlanır. Ancak, blokların birbirinden bağımsız olarak işlenmesi nedeniyle, resimde yüksek sıkıştırma oranlarında bloklama etkisi görülür (Şekil 2.2).

Şekil 2.2. 0.48 bpp Sıkıştırma Oranında Bloklama Etkisi

2.3.4. Ayrık Kosinüs Dönüşümü

Bloklara ayırma aşamasında elde edilen her 8x8 blok üzerine ayrık kosinüs dönüşümü (DCT) uygulanır [Rao, Yip, 1990]. DCT'nin amacı blokları frekans etki alanına çevirmektir. Dönüşüm yapıldığında insan gözünün ayırt etmekte zorlandığı yüksek frekans değerleri tespit edilir. 8x8 bir bloğun DCT uygulandıktan sonraki değerleri Şekil 2.3'te verilmiştir.

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix} \cdot \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.13 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.88 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}$$

(a) (b)

Şekil 2.3. 8x8 blok (a) ve DCT uygulandıktan sonra elde edilen değerler (b)

2.3.5. Niceleme

Niceleme işleminde, DCT dönüşümünden elde edilen her katsayı, belirlenen bir sabit değere bölünür ve bir tamsayı değere yuvarlanır. Bu işlemin ardından resimde kayıp

meydana gelir. Katsayıların bölüneceği değerler, bir niceleme matrisi ile belirlenir. Bu niceleme matrisindeki değerler, kullanılan kalite faktörüne göre değişiklik gösterir. DCT bloğuna uygulanan nicelemenin ardından yüksek frekans katsayıları sıfıra indirgenir (Şekil 2.4). Kalite faktörü düşürüldükçe, niceleme matrisindeki değerler büyür, bu sebeple bölme işleminden sonra 8x8 bloktaki sıfıra yaklaşan katsayı sayısı artar. Bunun sonucunda sıkıştırma miktarı artar fakat resmin kalitesinde daha fazla düşüş gözlenir.

$$\begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.13 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.88 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}$$

(a)

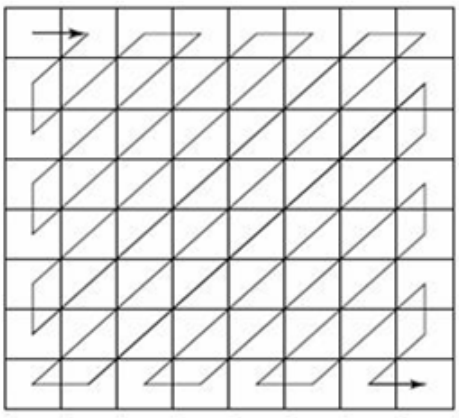
$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(b)

Şekil 2.4. DCT Bloğu (a) ve Niceleme işleminden sonra elde edilen matris (b)

2.3.6. Köşegen Tarama ile RLE ve QM Kodlama

Niceleme işleminden sonra, sıkıştırmayı arttırmak amacıyla köşegen tarama yapılır (Şekil 2.5). Köşegen tarama sayesinde sıfır olan yüksek frekans değerleri art arda gelir. Sıfırların art arda gelmesi ile RLE kodlamanın performansı arttırılır. RLE kullanılarak AC katsayıları sıkıştırıldıktan sonra, diğer AC katsayıları için ve her bloğun sol üst pikseli olan DC katsayıları için farklı QM tabloları kullanılarak, son aşama olan kayıpsız sıkıştırma gerçekleştirilir.



-26	-3	-6	2	2	-1	0	0
0	-2	-4	1	1	0	0	0
-3	1	5	-1	-1	0	0	0
-3	1	2	-1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Şekil 2.5. Köşegen Tarama

2.4. JPEG2000

2000 yılında Joint Photographic Experts Group tarafından geliştirilmiş bir standarttır. JPEG'in dezavantajlarını gidermek ve yüksek sıkıştırma oranlarında daha kaliteli resimler elde etme amacıyla geliştirilmiştir. JPEG2000 [ISO/IEC 15444-1:2004, 2000] [Christopoulos, Skodras, Ebrahimi, 2000], JPEG'de kullanılan DCT yerine ayrık dalgacık dönüşümü (DWT) [Mallat, 1989] [Jensen, Cour-Harbo, 2001] kullanır. Bu sayede JPEG'de görülen yüksek sıkıştırma oranlarındaki bloklama etkisini ortadan kaldırır. Bunun yerine, sıkıştırma miktarı arttırıldıkça, resimde bulanıklaşma meydana gelir (Şekil 2.6).

(a) (b)

Şekil 2.6. 0.24 bpp sıkıştırma oranında JPEG (a) ve JPEG2000(b) ile sıkıştırma

JPEG2000 mimarisi JPEG mimarisinden farklı olarak birkaç ek özelliğe sahiptir. JPEG2000'de sıkıştırılacak dosyanın ne kadar boyutta olacağı kodlayıcıya verilebilir. Standart, kayıplı ve kayıpsız sıkıştırma mimarilerini aynı kodlayıcı içerisinde barındırdığından, eğer verilen dosya büyüklüğüne eşit ya da daha düşük bir büyüklüğe kayıpsız sıkıştırma ile ulaşabiliyorsa, otomatik olarak kayıpsız sıkıştırmayı kullanır.

ROI(Reigon Of Interest) sayesinde resmin önemli kısımları yüksek kalitede, diğer kısımları düşük kalitede sıkıştırılabilir.

JPEG2000 ayrıca alpha kanalını da destekler. Alpha kanalı, her piksel için saydamlık değerini belirten 0-255 arası bir değerdir. RGB kanalının yanında dördüncü bir kanal olarak bulunur. Dizilim, RGBA veya ARGB şeklinde olabilir.

JPEG2000, kademeli aktarımı destekler. JPEG mimarisinde de kademeli aktarım desteği mevcuttur fakat yaygınlaşmamıştır.

JPEG2000 sıkıştırma aşamaları sırasıyla renk uzayı dönüşümü, bölümlendirme, ayrık dalgacık dönüşümü, niceleme, MQ kodlama şeklindedir. Bu aşamaların ayrıntıları şu şekildedir:

2.4.1. Renk Uzayı Dönüşümü

JPEG mimarisinde olduğu gibi, JPEG2000'de de renk uzayı dönüşümü yapılır. JPEG2000'de iki farklı renk uzayı dönüşümü desteklenir: Geri alınamayan renk dönüşümü ve geri alınabilen renk dönüşümü.

Geri alınamayan renk dönüşümü, bilinen RGB renk uzayından YCbCr renk uzayına dönüşümdür. Dönüşüm sırasında bazı kayıplar olduğundan dolayı, ters dönüşüm uygulanması mümkün değildir.

Geri alınabilen renk dönüşümünde niceleme hataları bulunmayan bir YCbCr uzayı kullanılır. Bu sayede ters dönüşüm mümkündür. Dönüşüm ve ters dönüşüm formülleri aşağıda verilmiştir.

Dönüşüm Formülleri:

$$Y = (R+2G+B)/4$$
$$Cb = B - G$$
$$Cr = R - G$$

Ters Dönüşüm Formülleri:

$$R = Y - (Cb+Cr)/4$$
$$G = Cr + G$$
$$B = Cb + G$$

2.4.2. Bölümlendirme

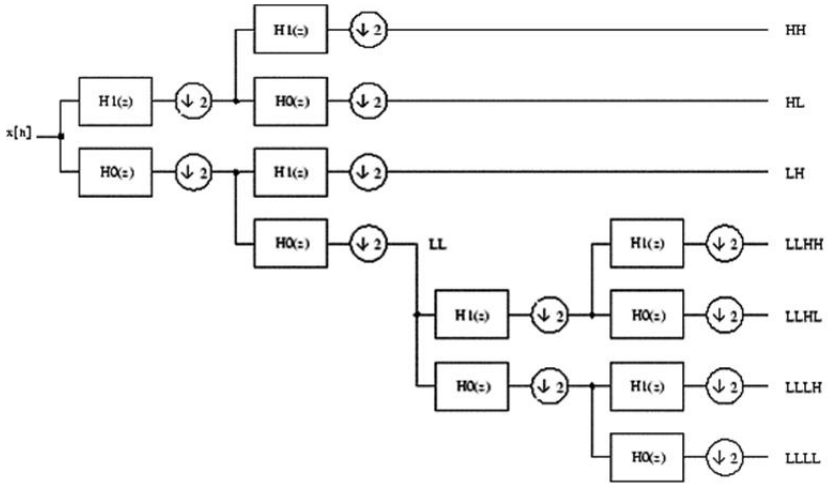
Renk dönüşümünden sonra resim "Tile" adı verilen bölümlere ayrılır. Bölümler istenilen büyüklükte seçilebilir, hatta tüm resim tek bir bölüm olarak alınabilir. Bölüm boyutu bir kez seçildiğinde eğer resmin çözünürlüğü bölüm boyutuna tam bölünemiyorsa

en alttaki ve en sağdaki bölümler hariç tüm bölümlerin boyutu aynı olur. Ardından her bölüm birbirinden bağımsız olarak sıkıştırılır.

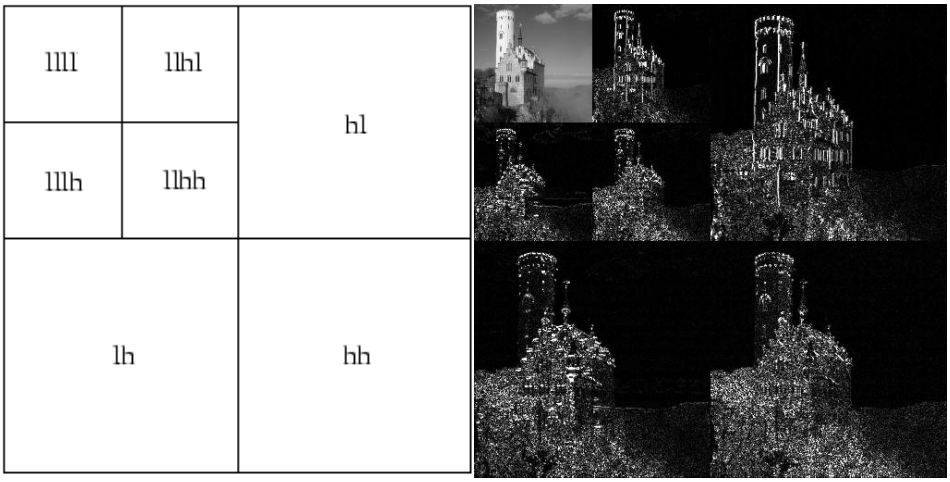
Bölümlendirme sayesinde karmaşıklık azaltıldığından, kodlayıcı daha az bellek ihtiyacı duyarak kodlama yapar.

2.4.3. Ayrık Dalgacık Dönüşümü

JPEG'in aksine JPEG2000 ayrık dalgacık dönüşümünü kullanır. DWT veriyi frekans bileşenlerine ayırır. İlk aşamada, resim yüksek frekans (High Pass) ve düşük frekans (Low Pass) filtrelerinden geçirilir ve iki adet alt bant elde edilir (L,H). Bu alt bantlar tekrar yüksek ve düşük frekans filtrelerinden geçirilerek toplamda dört adet bant elde edilir (LL,LH,HL,HH). Resmin sıkıştırma miktarına bağlı olarak bu işlem, iki kere düşük frekans filtresinden geçen resim için (LL) tekrar uygulanabilir. Dönüşüm sonrasında yaklaşık bir resim ve resmin detayları elde edilmiş olur (Şekil 2.7). Bu alt bantlar daha sonra açma esnasında orijinal resmi elde etmek için kullanılır.



Şekil 2.7(a). İki Defa Uygulanan Alt Bantlara Bölme İşlemi



Şekil 2.7(b). Alt Bantlara Bölme İşleminden Sonra Elde Edilen Detaylar

2.4.4. Niceleme

JPEG'de olduğu gibi JPEG2000'de de niceleme işlemi mevcuttur. Niceleme işleminin amacı yüksek frekans değerlerini sıfıra indirgemek ve MQ kodlamanın verimini yükselterek sıkıştırma miktarını arttırmaktır.

2.4.5. MQ Kodlama

Dönüşüm ve niceleme işlemlerinin ardından son olarak MQ kodlama uygulanır. MQ kodlama JPEG'deki QM kodlamaya benzer, içeriğe bağlı bir ikili aritmetik kodlamadır.

2.5. JPEG XR

Joint Photographic Experts Group tarafından, Windows Media Photo üzerine geliştirilmiş bir standarttır [ITU-T, 2009]. Amacı JPEG hızına ve JPEG2000 kalitesine ulaşmaktır.

Günümüze kadar standart bir resimde her kanal 8 bit ile ifade ediliyordu, yani her kanal için 256 farklı ton kullanılabilmekteydi. Hala her kanal için 8 bit kullanılmaya devam edilmesine rağmen teknolojinin ilerlemesi ve resimlerdeki kalitenin artması ihtiyacı ile doğru orantılı olarak, her kanalı ifade etmek için kullanılan bit sayısının artması kaçınılmazdır. Günümüzde her kanal için 8 bit yerine hâlihazırda 12 veya 16 bit

kullanılmaya başlanmıştır. Bu da artık maksimum 8 bit renk aralığını destekleyen JPEG'in ihtiyacı karşılayamaması anlamına gelmektedir. Eğer her kanal için 12 bit kullanan bir resim JPEG ile sıkıştırılırsa, her renk kanalı için 4 bitlik bilgi kaybolacaktır.

JPEG XR'daki XR, Extended Range'in kısaltmasıdır. Bu da JPEG XR'ın her kanalda genişletilmiş (8 bitten fazla) bilgi saklamak amaçlı geliştirildiğini kasteder. Genişletilmiş bilgi saklama özelliği sayesinde JPEG XR, yüksek çözünürlüklü ya da HDR (High Dynamic Range) resimlerin sıkıştırılması için idealdir.

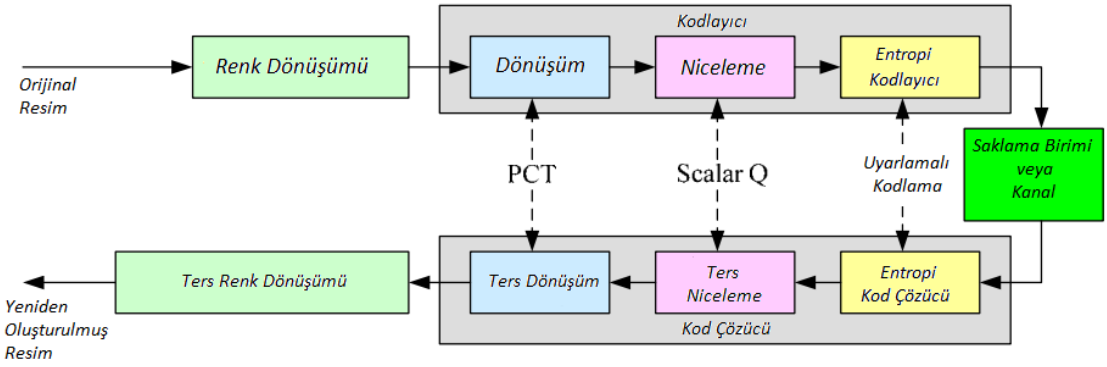
HDR resimler, görüntünün en karanlık ve en aydınlık kısımlarının arasında çok daha geniş aralıkta dinamik parlaklık değeri bulundurur. Bu resimler birden fazla farklı tonda SDR (Standard Dynamic Range) resmin birleştirilmesi ile oluşturulabilir. Yeni nesil fotoğraf makineleri ile aynı pozun farklı parlaklık oranlarında çekilip bunların birleştirilmesiyle HDR resimler elde edilebilir (Şekil 2.8).

 (1)  (2)  (3)  (4)

Şekil 2.8(a). -8 eV (1), -2 eV (2), +2 eV (3) ve +4 eV (4) parlaklık oranında çekilmiş resimler

Şekil 2.8(b). Şekil 2.8(a)'daki resimlerin bir araya getirilmesiyle oluşturulmuş HDR resim

JPEG XR'ın kodlama aşaması JPEG'in kodlama aşamasına benzerdir: Bölümlendirme, renk dönüşümü, ön filtreleme, photo core dönüşümü, niceleme, katsayı tahmini, entropi kodlama. Kodlama aşaması Şekil 2.9'da görülebilir.



Şekil 2.9. JPEG XR Kodlama-Açma Aşamaları

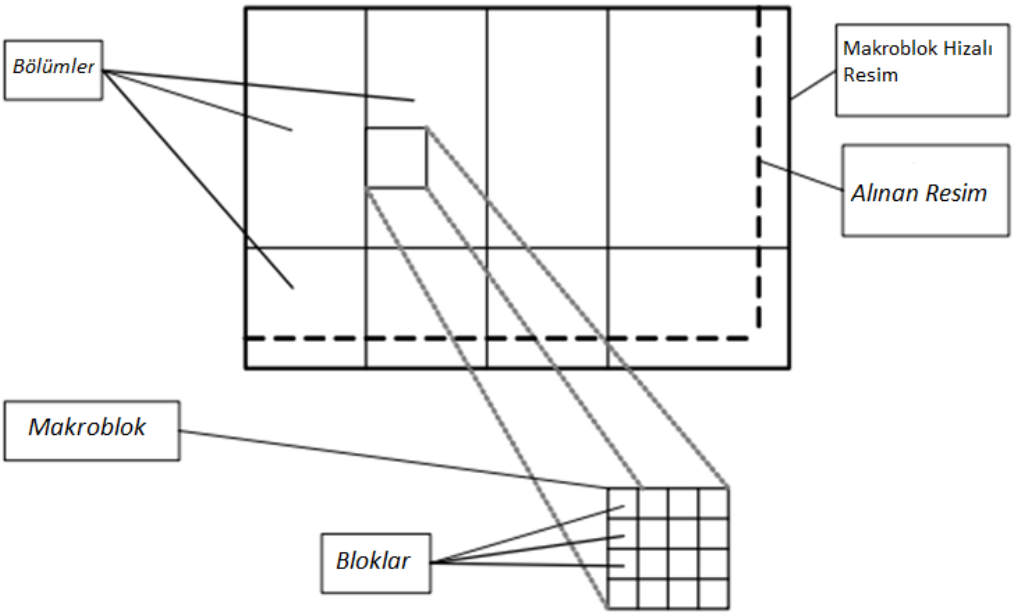
2.5.1. Bölümlendirme

Kodlamanın ilk aşamasında JPEG2000'e benzer olarak resimde bölümlendirme yapılır. Resim bölümlendirmeye tabi tutulduktan sonra her bölüm birbirinden bağımsız olarak kodlanır. Bölümlemenin sağladığı işlem kolaylığı sayesinde donanım uyarlaması da basit olur. Her bölüm, 4x4lük 16 adet bloktan oluşan makroblokları içerir (Şekil 2.11).

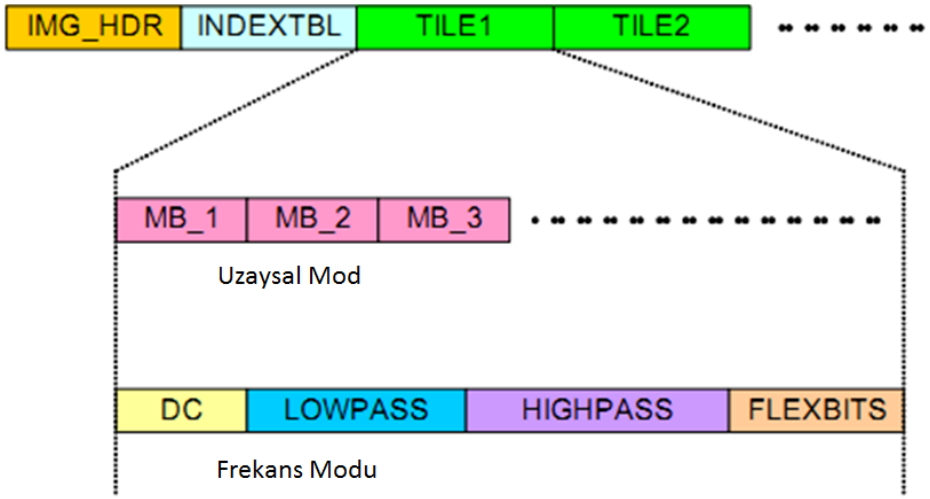
Bölümlendirme aşamasında iki farklı kip bulunur. Bunlar uzamsal mod ve frekans modudur.

Uzamsal modda, bir bölüm (tile) paketi, sol üstten başlayarak satır sıralı olarak makroblokları saklar.

Frekans modunda ise en düşük frekans katsayısı (DC), düşük frekans katsayıları (LP) ve yüksek frekans katsayıları (HP) olmak üzere üç farklı frekans bileşeni, farklı paketlerde saklanır (Şekil 2.11).



Şekil 2.10. JPEG XR Bölümlendirme Aşaması



Şekil 2.11. JPEG XR Bölümlendirme Modları

2.5.2. Renk Uzayı Dönüşümü

JPEG ve JPEG2000'de olduğu gibi dönüşüm RGB renk uzayından YUV renk uzayına yapılır. Dönüşüm kayıpsız olarak ters çevrilebilirdir. Bu sayede tekrar RGB uzayına dönüşüm mümkündür. Dönüşüm formülleri aşağıda verilmiştir.

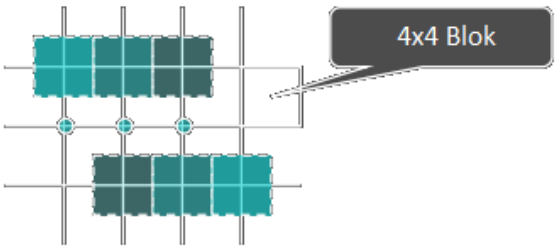
$$Y = G + Cb/2 - 128$$

$$Cb = -[R - G + Cr/2]$$

$$Cr = B - R$$

2.5.3. Ön Filtreleme

Ön filtreleme yapmanın amacı, yüksek sıkıştırma oranlarında görülen bloklama etkisini ortadan kaldırmaktır. Birbirine komşu 4 bloğun 2x2'lik kısımları birlikte işlenerek bloklama etkisinin önüne geçilmiş olur (Şekil 2.12).



Şekil 2.12. Ön Filtrelemede 4x4'lük Blokların Ortak Kısımlarının Seçilmesi

Ön filtrelemede üç farklı seçenek bulunur. Bunlar ön filtreleme yapmamak, bir aşamalı ön filtreleme ve iki aşamalı önfiltrelemedir.

Ön filtreleme yapmamak, en hızlı yöntemdir. Düşük sıkıştırma miktarları veya kayıpsız sıkıştırma için idealdir. Düşük sıkıştırma oranlarında bloklama etkisi sezilmeyeceğinden gerekli değildir fakat ön filtreleme yapılmadığı takdirde, yüksek sıkıştırma oranlarında bloklama etkisi görülür.

Bir aşamalı ön filtreleme, ön filtreleme yapmama durumuna göre daha fazla sıkıştırma sağlar fakat işlemlerin artması nedeniyle daha fazla zamana ihtiyaç duyar.

İki aşamalı ön filtreleme en karmaşık yöntemdir. En iyi kalite için kullanılması gerekmektedir. Bloklama etkisini minimuma indirir ve bütün ön filtreleme seçenekleri arasında en iyi PSNR sonuçlarını verir.

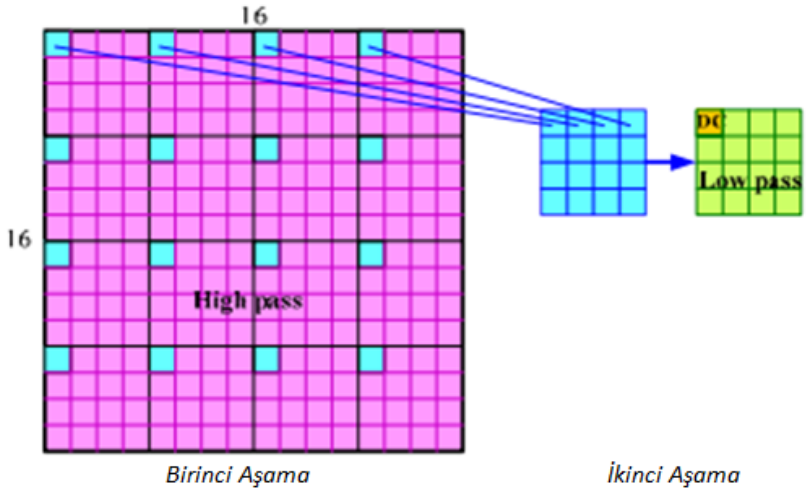
2.5.4. Photo Core Dönüşümü

JPEG XR, DCT'ye benzer bir metot olan Photo Core Dönüşümü'nü (PCT) kullanır. PCT, Hadamard [Ritter, 1996] dönüşümünün bir türüdür. 4x4'lük bloklar üzerinde 2x2'lik Hadamard dönüşümleri ve döndürme işlemlerinin uygulanmasıyla elde edilir.

PCT, piksel verisini frekans etki alanına çevirir. DCT'ye benzerlik gösterir fakat DCT'nin aksine kayıpsızdır. Bu sayede ters dönüşümler ile orijinal veri yeniden elde edilebilir. Tamsayılar üzerinde tanımlı olduğundan işlem karmaşıklığı diğer dönüşümlere göre daha düşüktür.

PCT işleminde, piksel değerleri DC (en düşük frekans katsayısı), düşük frekans katsayıları (Low-Pass) ve yüksek frekans katsayıları (High-Pass) bileşenlerine dönüştürülür.

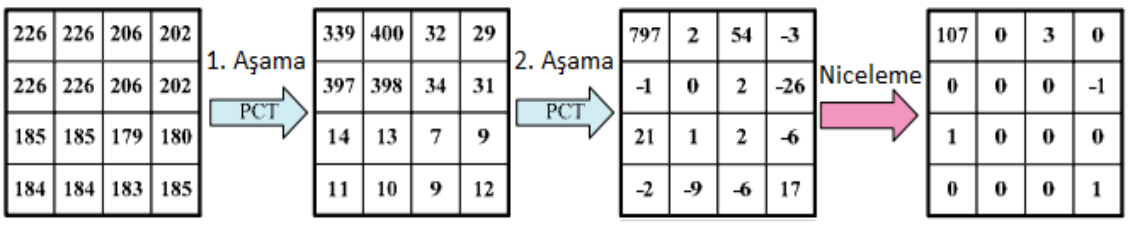
PCT iki aşamada gerçekleştirilir. Birinci aşamada makro blok ön filtrelemeden geçer ve ardından her blok 4x4'lük PCT dönüşümüne tabi tutulur. İkinci aşamada, makrobloktaki 16 adet DC değeri tek bir 4x4 blokta toplanır. Eğer kodlamada örnekleme azaltma (down-sampling) işlemi seçildiyse, bu bloğun büyüklüğü 2x4 veya 2x2 olabilir. Seçimlik ikinci bir ön filtreleme işlemi, DC katsayılarından oluşan bloğa bir kere uygulanabilir. Elde edilen DC bloğuna ikinci kez bir PCT uygulanır. Eğer örnekleme azaltma yapılmadıysa PCT ve ön filtreleme işlemi DC bloğuna 4x4 olarak uygulanır. Örnekleme azaltma mevcutsa, DC bloğunun büyüklüğü 2x2 veya 2x4 olacağından, bu bloğa uygulanacak PCT ve ön filtreleme işlemleri de 2x2'lik bloklar halinde olacaktır. İkinci aşamanın sonunda bir makroblok için 1 DC, 15 AD (low-pass) ve 240 AC (high-pass) katsayısı elde edilir (Şekil 2.13).



Şekil 2.13. PCT Birinci ve İkinci Aşama

2.5.5. Niceleme

Niceleme işleminde PCT'den elde edilen değerler bir tamsayıya bölünür ve sonrasında tamsayı değerlere yuvarlanır (Şekil 2.14). Kayıpsız sıkıştırma için bölen değer 1 olurken, kayıplı sıkıştırma için ise bu değer 1'den büyük seçilir. Nicelemede tamsayı işlemlerin kullanılmasının avantajı, bölme işlemleri için sadece kaydırma işleminin yapılması ve bu sayede hız kazanılmasıdır.



Şekil 2.14. 4x4'lük Bloğun Niceleme İşleminden Sonraki Durumu

2.5.6. Katsayı Tahmini

Katsayı tahmini, iki makroblok arasındaki benzerlik yeterli derecede ise yapılır. Dinamik olarak benzerliğin yüksek olduğu yön tespit edilir ve bu yön tahmin yönü olarak seçilir. Benzerlik bulunduktan sonra ise iki değer arasındaki fark tahmin değeri olarak kodlanır.

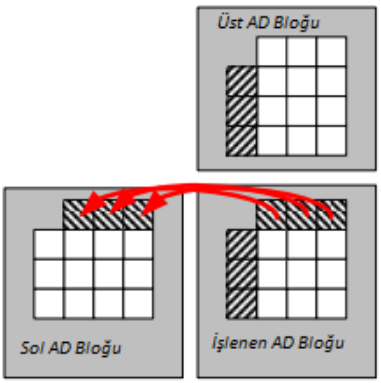
Katsayı tahmini PCT'nin 2. aşamasına girecek olan DC katsayıları için yapılan tahmin, PCT'nin 2. aşamasından sonra elde edilen AD (low-pass) katsayıları için yapılan tahmin ve PCT'nin ilk aşamasından sonra elde edilen AC (high-pass) katsayıları için yapılan tahmin olmak üzere üçe ayrılır.

DC için katsayı tahmini üç yönde (sol, üst, sol üst) yapılabilir. DC katsayıları her makrobloğun sol üst köşesinde bulunur. H_weight ve V_weight olmak üzere dikey ve yatay ağırlıklar hesaplanır.

Sol üst makrobloktaki DC katsayısından üst makrobloktaki DC katsayısının çıkartılmasıyla H_weight, sol makrobloktaki DC katsayısının çıkartılmasıyla V_weight elde edilir.

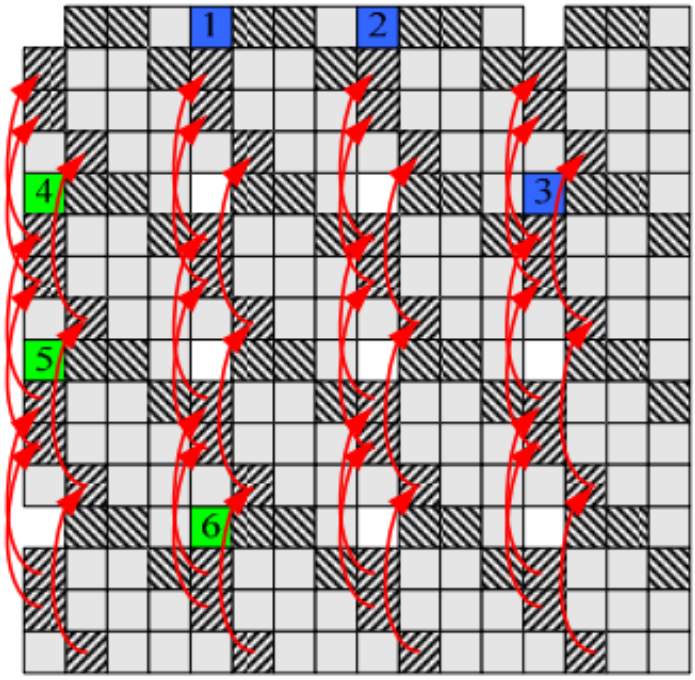
Bu ağırlıklara bakarak algoritma tahmini üstten mi, soldan mı yoksa sol üstten mi yapacağına karar verir. Eğer H_weight, V_weight'in 4 katından büyükse soldan, V_weight H_weight'in 4 katından büyükse üstten ve iki durum da sağlanmazsa sol üstten tahmin yapılmasına karar verilir. Bunun nedeni, tahmin edilecek bloğun değer olarak yakınlığa göre seçilmesidir.

AC ve AD blokların tahmini üst veya soldan yapılabilir. AD katsayıları için tahmin yönüne karar verme aşaması DC katsayıları için kullanılan algoritma ile benzerdir. Tek farkı, ilk iki koşulun sağlanmaması durumunda, tahmin yapmamasıdır (Null prediction). Algoritmanın AD (Low-pass) katsayıları için tahminin soldan yapılmasına karar vermesi durumunda, her blok için tahmin işleminin nasıl olacağı Şekil 2.15'te verilmiştir.



Şekil 2.15. AD Bloğunun Soldan Tahmini

AC (high-pass) katsayı tahmini de DC ve AD tahminlerinde olduğu gibi, iki adet ağırlık değerinin karşılaştırılması ile yapılır. Y, Cb ve Cr değerlerine bağlı olarak H_weight ve V_weight ağırlık değerleri hesaplanır. H_weight değeri, V_weight'in dörtte birinden küçük bir değerse soldan, dörtte birinden büyük fakat dört katından küçükse üstten tahmin yapılır. Eğer her iki durum da sağlanmazsa tahmin yapılmaz. Üstten yapılan tahmin için blokların durumu Şekil 2.16'daki gibi olacaktır.



Şekil 2.16. AC Bileşenlerinin Üstten Tahmin Edilmesi

2.5.7. Uyarlamalı Tarama

Entropi kodlamadan önce uyarlamalı bir tarama gerçekleştirilir. Bu tarama sayesinde iki boyutlu blok, entropi kodlama tarafından kodlanabilecek tek boyutlu bir dizi haline gelir. Kodlama sırasında her 4x4'lük blok için her konumda kaç adet sıfırdan farklı katsayı ile karşılaşıldığı saklanır. Bloktaki elemanların taranma sırası, bu sıfırdan farklı katsayıların sayısına bağlı olarak belirlenir. Kodlama esnasında eğer herhangi bir konum için sıfırdan farklı katsayı sayısı değişirse, tarama sırası da uyarlamalı olarak değişecektir. Böylece sıfırdan farklı katsayılar başlangıçta kodlanacak ve sıfırlar tek boyutlu dizinin son kısmında toplanacaktır.

Tarama sırasının Şekil 2.17(a)'daki gibi olduğunu varsayalım. Sıradaki blok okunduğunda (Şekil 2.17(b)), bu blokta sıfırdan farklı katsayıların bulunduğu konumlar için toplam değerleri değişecektir. 2, 7, 8, 12 ve 15. konumların katsayı sayısı 1 arttırılır. 0. Konumdaki DC katsayısı ise buna dahil edilmez. 12. konumun sıfırdan farklı katsayı sayısı 5. konumu geçtiğinden, tarama sırası yeniden düzenlenir. Son durum Şekil 2.17(c)'de gösterilmektedir.

Tarama Sırası	4	8	5	12	1	6
Toplam	45	36	32	32	30	27

(a)

$$\begin{bmatrix} 107 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 \\ 4 & 0 & 0 & 1 \end{bmatrix}$$

(b)

Tarama Sırası	4	8	5	12	1	6
Toplam	45	37	32	33	30	27

↓

Tarama Sırası	4	8	12	5	1	6
Toplam	45	37	33	32	30	27

(c)

Şekil 2.17. Tarama Sırasının İlk Durumu (a), Okunan Blok (b) ve Tarama Sırasındaki Değişim (c)

2.5.8. Entropi Kodlama

Uyarlamalı tarama gerçekleştikten sonra elde edilen akışa RLE uygulanır. RLE sonucu elde edilen bit dizisi Huffman kodlamaya tabii tutulur. JPEG XR'da kullanılan Huffman kodlama diğer standartlardan farklıdır. Yöntem en iyi kodlamayı elde etmek için optimum sayıda küçük boyutlu Huffman tabloları elde etmeyi amaçlar. Bu işlemlerin ardından paketleme işlemi gerçekleştirilir ve dosya oluşturulur.

BÖLÜM 3

DÖNÜŞÜM VE BÖLÜMLENDİRME İŞLEMLERİ

Bir hareketsiz görüntü üzerinde dönüşüm işlemleri yapılarak sıkıştırma işleminde kâr edilebileceği gibi bölümlendirme yapılarak her parçanın farklı bir işleme tabi tutulması ve bu sayede sıkıştırma performansının arttırılması mümkün olabilir.

"Kanalları Farklı Şekilde Kodlama Temelli Çalışmalar" alt başlığında, resmin renk kanalları üzerinde yapılan işlemler ile sıkıştırma performansının arttırılması amaçlanmıştır. Renk kanalları üzerinde yapılan dönüşüm işlemleri, dosya yapısındaki kanalların saklama sıralamalarının değiştirilmesi veya R, G, B kanallarının dizilere ayrılıp üzerinde işlemler yapılması, herhangi bir kanalın dinamik olarak bölümlendirilmesi gibi farklı yöntemler denenmiş ve her bir yöntemin sonucu alınarak karşılaştırma sonuçları incelenmiştir.

"Karmaşıklığa Göre Uygun Yöntemi Seçme Çalışmaları" alt başlığında ise resmin bölümlendirilmesiyle sıkıştırma performansının değişimi incelenmiştir. Bu kısımda resmin 4 ya da 16 parçaya bölümlendirilmesi, parçalar üzerinde farklı sıkıştırma yöntemlerinin denenmesi ve bu bilgilerin elde edilmesiyle parçaların hangi yöntem ile sıkıştırılacağının kararını verecek bir algoritmanın geliştirilmesi hedeflenmiştir.

3.1. Kanalları Farklı Şekilde Kodlama Temelli Çalışmalar

3.1.1. EBMP

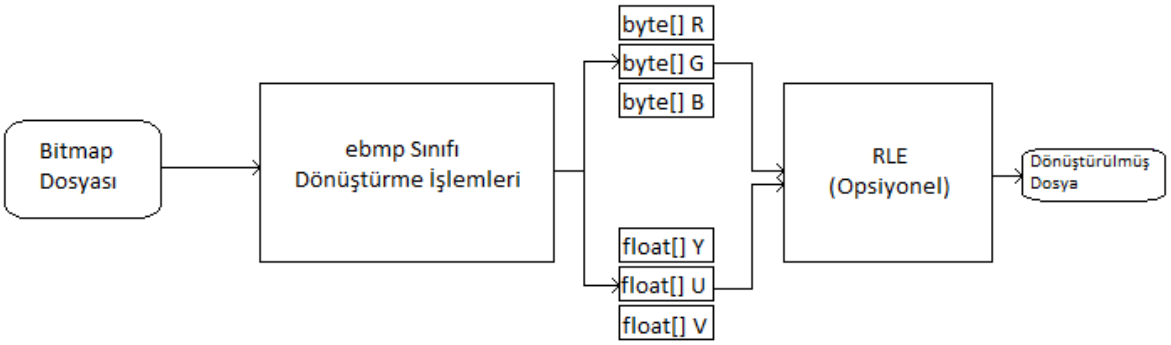
EBMP'de 24 bpp bitmap resim alındıktan sonra R,G, B kanalları için veya RGB YUV'a dönüştürüldükten sonra Y, U ve V bileşenleri için çeşitli işlemler uygulanır.

RGB ile kodlama seçildiğinde kırmızı, yeşil ve mavi kanalları ayrı ayrı 3 adet bayt dizisine atanır. Daha sonra her dizi ayrı ayrı RLE'ye tabii tutulur. Elde edilen RLE verileri R, G ve B renk kanalı sırasıyla art arda eklenir.

YUV ile kodlama seçildiği takdirde ise öncelikle RGB'den YUV'a dönüşüm işlemi gerçekleştirilir. RGB'den YUV'a dönüştürme için JPEG XR'ın kullandığı kayıpsız

dönüşüm formülleri kullanılmıştır. Dönüşüm gerçekleştirildikten sonra Y, U ve V bileşenlerine RGB kanallarına uygulanan işlemlerin aynısı uygulanır.

Bir diğer yöntem ise kanallara RLE uygulamadan önce her kanal için kendi içerisinde fark kodlama yapmaktır. RGB ve YUV için fark kodlama işlemi gerçekleştirilip her kanala RLE uygulanır (Şekil 3.1).



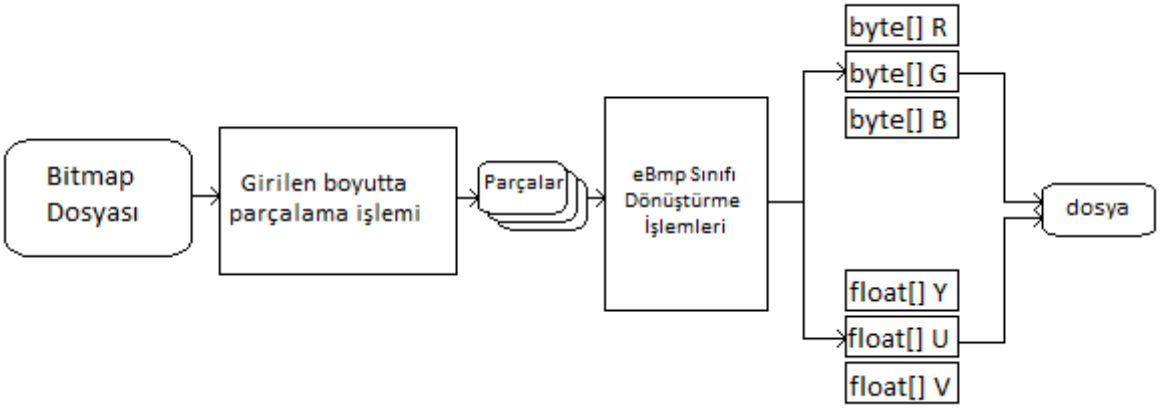
Şekil 3.1. EBMP sıkıştırma aşamaları

3.1.2. Statik Sabit Büyüklükte Bölümlendirme

Statik Sabit Büyüklükte Bölümlendirme (SSBB) yönteminde alınan resim öncelikle girdi olarak verilen parça boyutu kadar parçalara bölünür. Örneğin, girdi olarak parça boyutu 64 olarak verilirse resim 64x64'lük parçalara bölünecektir. Daha sonra her parça için RLE kodlama işlemi gerçekleştirilir.

İkinci aşamada, bölünen dosyaların her biri için renk kanalları satır sıralı olarak ayrı ayrı dizilere aktarılır ve bu diziler R-G-B sırası ile ardı sıra eklenir. Daha sonra dosyanın başına her bir parçanın piksel olarak en ve boy bilgileri girilip bu dizi dosyanın sonuna eklenir. Böylece her parça için R-G-B sıralı bir dosya elde edilmiş olur (Şekil 3.2).

SSBB'nin amacı, her bir parça için RLE kodlamanın verimliliğini, sözlük sıkıştırma yöntemlerine etkilerini ve EBMP'de olduğu gibi RGB kanallarının sırasının değiştirilmesinin RLE kodlamaya olan katkısını ölçmektir.



Şekil 3.2. SSBB Sıkıştırma Aşaması

3.1.3. Dinamik Sabit Büyüklükte Bölümlendirme

Dinamik Sabit Büyüklükte Bölümlendirme (DSBB) yönteminde resmin bölüneceği parçaların boyutu verilen resme göre dinamik olarak belirlenir. Alınan 0-255 arası bir fark değerine göre resmin herhangi bir kanalında yan yana olan pikseller taranır ve bulunana kadar yatay olarak ilerlenir. Verilen fark yan yana iki piksel arasında bulunursa parçanın yatay boyutu bulunmuş olur ve dikey taramaya geçilir.

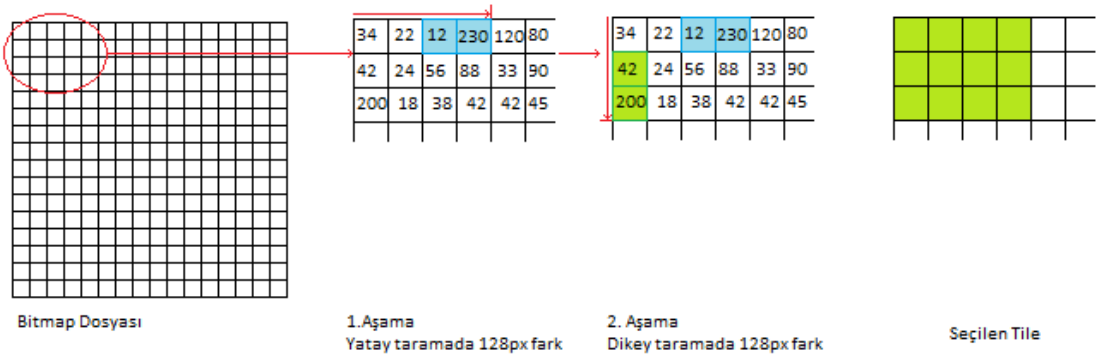
Dikey taramada da aynı fark değerine göre alt alta pikseller kontrol edilir. Fark değeri sağlandığında parçanın dikey boyutu da bulunmuş olur ve sol üstten itibaren referans alınan parça büyüklüğü tüm resme uygulanır.

Parça boyutunun belirlenmesi aşamasından sonraki diğer adımlar daha önceki uygulamalarda olduğu gibi kanalların sıralanması şeklinde gerçekleşir. Dosya, karşılaştırma sonuçları için uygulama tarafından üretilir.

Yöntemin avantajı, düz zemin olan veya tek renk içeren resimlerde parçalamaya gerek duymaması veya farklı olana kadar parçalamayarak düz kısımları tespit etmesidir. Bu sayede düz kısımlar kendi içerisinde daha çok sıkışarak, parçalamanın getirdiği dezavantajı da ortadan kaldıracaktır.

Resim çoğunlukla düz veya karmaşık olmayan alanlardan oluşurken sol üst kısmı karmaşık olabilir. Böyle bir durumda sol üst kısım referans alındığı için parça boyutu

küçük olacak ve düz alanlar için gereksiz yere bölme yapılacaktır. Bu yüzden resmin parça büyüklüğü kararının sol üstten verilmesi her zaman en iyi sonucu sağlamaz. Ayrıca verilen fark değeri bir kanal için taranmaktadır. R kanalı üzerinde taranan bir resimde hiç kırmızı renk tonu bulunmazsa resim düz olarak algılanacak ve tek parça olarak sıkıştırılmak istenecektir.



Şekil 3.3. Parça boyutunun belirlenme aşamaları

3.1.4. Dinamik Değişken Büyüklükte Bölümlendirme

Önceki çalışmada resmin bölünecek parça büyüklüğü sol üst köşeden itibaren referans alınmakta ve parça büyüklüğü tespiti herhangi bir kanaldaki fark değerine göre belirlenmekteydi. Dinamik Değişken Büyüklükte Bölümlendirme (DDBB) yönteminde ise resim öncelikle R, G ve B kanalları için 3 farklı bayt matrisine ayrılır. Daha sonra her matris üzerinde bölümlendirme işlemleri uygulanır. DSBB'den farklı olarak sabit bir parça boyutu belirlenmez. Uygulama çalıştırıldığında girdi olarak fark koşulu alınır. Her kanal matrisi için sol üst köşeden başlanmak koşuluyla önce yatay sonra dikey olmak üzere farkın sağlandığı piksel çiftleri taranır. Her iki boyut için de fark koşulu sağlandığında elde edilen parça satır sıralı olarak okunur ve parçaların saklandığı dosyaya yazılır. Bu sayede aynı sembollerin art arda gelmesi sağlanır. Ayrıca daha sonra açma işlemi için kullanılacak bir koordinat dosyası oluşturulur. Bu dosyanın içerisine bu parçanın koordinatları yazılır. Uygulama elde ettiği parçaları ve koordinatları sırasıyla dosyaya yazdığı ve okuma safhasında da aynı sırayı takip ettiği için herhangi bir okuma problemi ile karşılaşılmamaktadır. Dosya içerisindeki koordinat verileri Tablo 3.1'de verilmiştir.

Resmin tarama işlemi sırasında işlenen parçaların saklandığı bir işaret matrisi tanımlanır. Bu matriste farklı boyuttaki parçalar tespit edildikçe, matris üzerinde tekrar okunmaması için işaretlenirler. Algoritma satır sıralı olarak resmi tekrar kontrol eder ve işaretsiz olan konumdan itibaren fark koşulunu tekrar arar. Fark koşulu sağlanmadan resmin sınırlarına gelinirse, parça boyutu resmin sınırına kadar olacaktır. Ayrıca yöntem fark koşulunu sağlamadan daha önceden işaretli bir piksele rastlarsa, yine sınırı bu piksele kadar belirler. Böylece fark olmayan durumlarda aynı bölgenin tekrar işlenmesinin önüne geçilmiş olur. Algoritma tarama işlemini tüm pikselleri işaretleyene kadar, yani tüm parçaları kodlayana kadar devam ettirir.

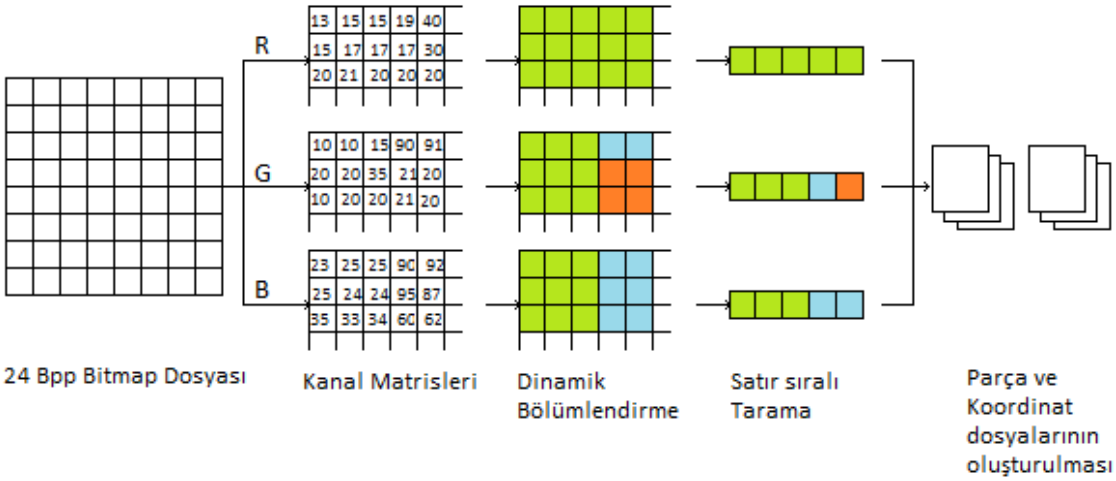
Parçalama işlemi her kanal üzerinde ayrı ayrı yapıldığından dolayı, her kanaldaki parça sayısı farklı olacaktır. Yarısı kırmızı ve yarısı yeşil olan bir resim dosyası üzerinde fark koşulu olarak 255'den küçük bir sayı verildiğinde R ve G kanalları iki parça halindeyken, B kanalı tek parça olacaktır.

Parçalamanın kanallara göre farklı yapılmasının avantajı, belirli renklerin bulunmadığı resimlerde gereksiz parçalama işlemi yapmamasıdır. Başlangıç durumunda kanallara ayırma işlemi, aynı sembolleri art arda getirmesi bakımından kar sağlayacaktır. Ardından gereksiz parçalama işlemlerinden kaçınmak ise sözlük tabanlı sıkıştırma yöntemlerindeki kârı arttıracaktır.

Parçalama işleminin dezavantajı ise verilen fark koşulunun çok fazla yerde sağlanmasıyla ortaya çıkacaktır. Parça boyutları fazlasıyla küçüleceği için, hatta bazı durumlarda piksel boyutunda parçalar olacağı için bu parçanın koordinatlarıyla beraber saklanması nedeniyle sıkıştırma yerine bazen şişirme de gerçekleşebilir.

Tablo 3.1. Koordinat dosyasının içeriği

X1 Koordinatı	Y1 Koordinatı	X2 Koordinatı	Y2 Koordinatı
0	0	30	70
40	0	70	50
…	…	…	…



Şekil 3.4. 64 Fark koşulu için dosyanın bölümlendirme aşamaları

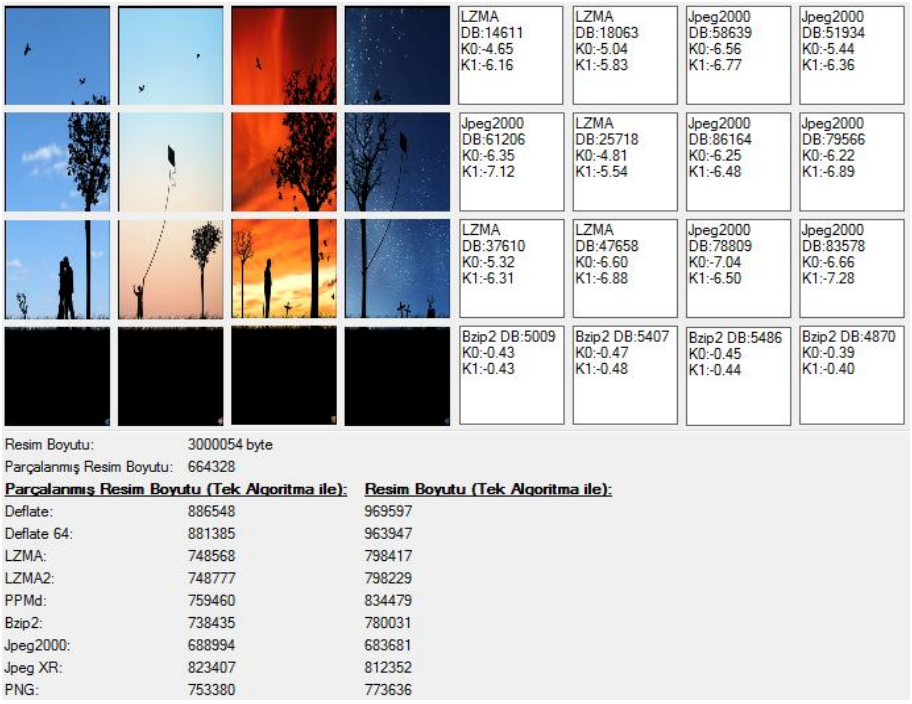
3.2. Karmaşıklığa Göre Uygun Yöntemi Seçme Çalışmaları

Öncelikle hangi karmaşıklık düzeyinde hangi kayıpsız sıkıştırma yönteminin daha iyi sonuç verdiğini belirlemek için, 9 farklı sıkıştırma yöntemi ile bir çok farklı türde görüntü dosyası ve bu görüntülerin farklı karmaşıklıktaki 4 veya 16 eşit parçası sıkıştırılmış ve elde edilen sonuçlar incelenmiştir.

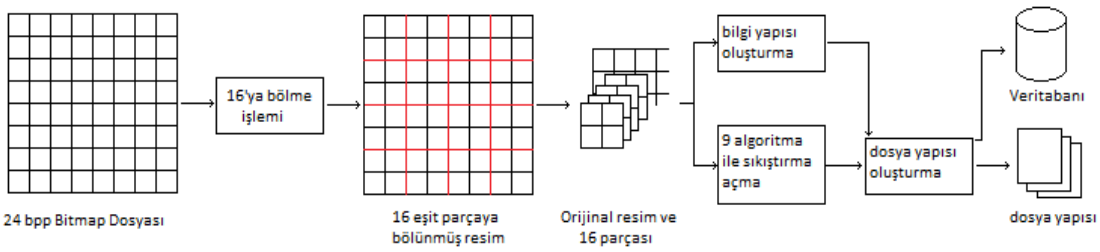
3.2.1. Karmaşıklık İstatistik-1

Karmaşıklık İstatistik-1 uygulamasında istatistiksel veriler elde edilmiş ve bu verilere dayanarak bir tahmin algoritması denemesi yapılmıştır. 24 bpp bitmap dosyasını 16 eşit parçaya bölüp her parçayı farklı bir algoritmayla sıkıştırmak, tüm resmi tek bir algoritma ile sıkıştırmaya göre daha fazla kâr sağlayabilir (Şekil 3.5). İstatistiksel veri elde etme aşamasında 24 bpp bitmap dosyası alındıktan sonra 16 eşit parçaya bölünür. Dosyanın kendisi ve bölündüğü 16 parça olmak üzere toplam 17 resim üzerinde 9 adet kayıpsız sıkıştırma algoritması ile sıkıştırma gerçekleştirilir. Sıkıştırma sonucunda resim ve 16 parçası için sıkıştırma oranları, dosya boyutları ve sıkıştırma süreleri bir "bilgi" yapısının dizisi kullanılarak saklanır. Aynı bitmap dosyasının ve parçalarının karmaşıklığı hesaplanır ve bu dosyaların belirli bilgileri bir "dosya" yapısı kullanılarak bir dizi içerisinde saklanır. Şekil 3.6'da görüldüğü gibi istatistiksel sonuçlar elde etmek amacıyla

bu bilgiler veri tabanına da eklenir. Kullanılan “bilgi” ve “dosya” yapılarının içeriği de Şekil 3.7’de verilmiştir.



Şekil 3.5. Farklı Algoritmaların Sıkıştırma ve Karmaşıklık Değerleri



Şekil 3.6. Karmaşıklık İstatistik-1 dosya oluşturma/veritabanına ekleme aşamaları

```
[Serializable()]
public struct bilgi
{
    public string dosyaBoyutu; //Sıkıştırılmış dosya boyutu
    public string sikistirmaOrani; //Sıkıştırma oranı
    public string sure; //Sıkıştırma süresi
}

[Serializable()]
public class dosya
{
    public string dosyaAdi; //Sıkıştırılan dosyanın adı
    public Bitmap orjResim; //Programda görüntülenebilmesi için resim
    public Bitmap orjResimGS; //Resmin gri kodlanmış hali
    public string dosyaBoyutu; //Orijinal dosya boyutu
    public string[] karmasiklik; //Karmaşıklık değerlerini saklayan dizi
    public bilgi[] bilgiler; //Sıkıştırma algoritmaları bilgileri
}
```

Şekil 3.7. Dosya ve bilgi yapıları

Dosya yapısında, daha sonra gösterilebilmek üzere resmin kendisi de saklanır. Bilgi yapısındaki dosya boyutu, sıkıştırılmış dosya boyutunu verirken, dosya yapısındaki dosya boyutu sıkıştırılmamış dosya boyutunu içerir. Bilgiler dizisi 9 adet sıkıştırma algoritması için kullanılır.

Kullanılan algoritmalar Deflate, Deflate64, LZMA, LZMA2, PPMd, Bzip2, Jpeg2000, Jpeg XR ve PNG'dir. Jpeg2000 ve JpegXR görüntü sıkıştırma algoritmalarının kayıpsız seçenekleri kullanılmıştır.

Karmaşıklık hesabında gri tonlu resim üzerinde hesaplanan entropi ve 24 bpp resim üzerinde R, G ve B kanallarının farklı oranlarının hesaba katıldığı ColorEntropyAVG ve ColorEntropyPER ve son olarak Matlab ile elde edilen quadtree decomposition sonucu resimdeki karelerin sayısı kullanılır. ColorEntropyAVG her kanalın entropisinin aritmetik ortalaması şeklinde hesaplanırken, ColorEntropyPER'de ise R kanalının entropisi 0.299, G kanalının entropisi 0.587 ve B kanalının entropisi 0.114 ile çarpılarak toplam değer elde edilir. Bu değerler Jpeg'deki dönüşümün sayısal değerleridir. Resmin gri tonlu hale getirilmesinde de her piksel için R, G, B kanallarının parlaklık değeri 0.299*R+0.587*G+0.114*B formülüyle hesaplanır.

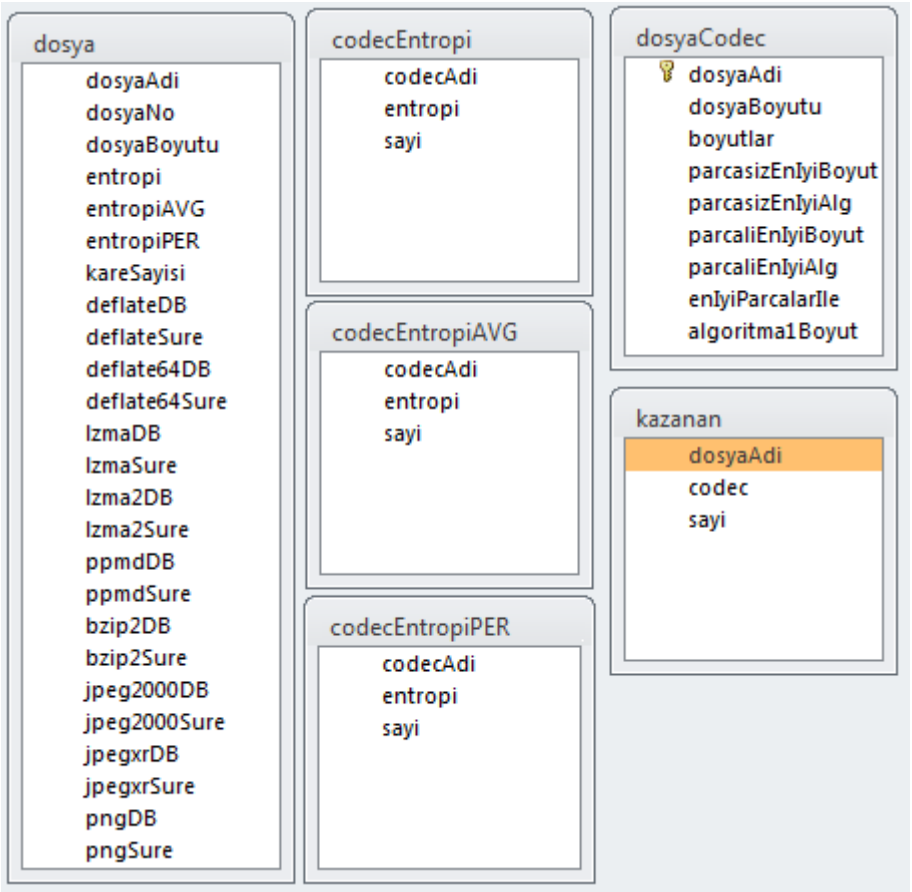
Quadtree decomposition, kare ve gri tonlu olan bir resmi 4 eşit parçaya böler. Bölme işleminden sonra her parçanın belirli bir homojenlik kriterine uyup uymadığını test

eder. Eğer homojenlik kriteri sağlanırsa o parça tekrar bölünmez. Parçanın homojenlik kriterine uymaması durumunda ise parça yeniden dört parçaya bölünür ve bu parçalarda da aynı homojenlik kriteri aranır. Her blok homojenliği sağlayana kadar bu işlem devam eder. Sonuç olarak resmin her parçası farklı sayıda alt parçaya bölünebileceği gibi, resim hiç bir parçaya da bölünmeyebilir.

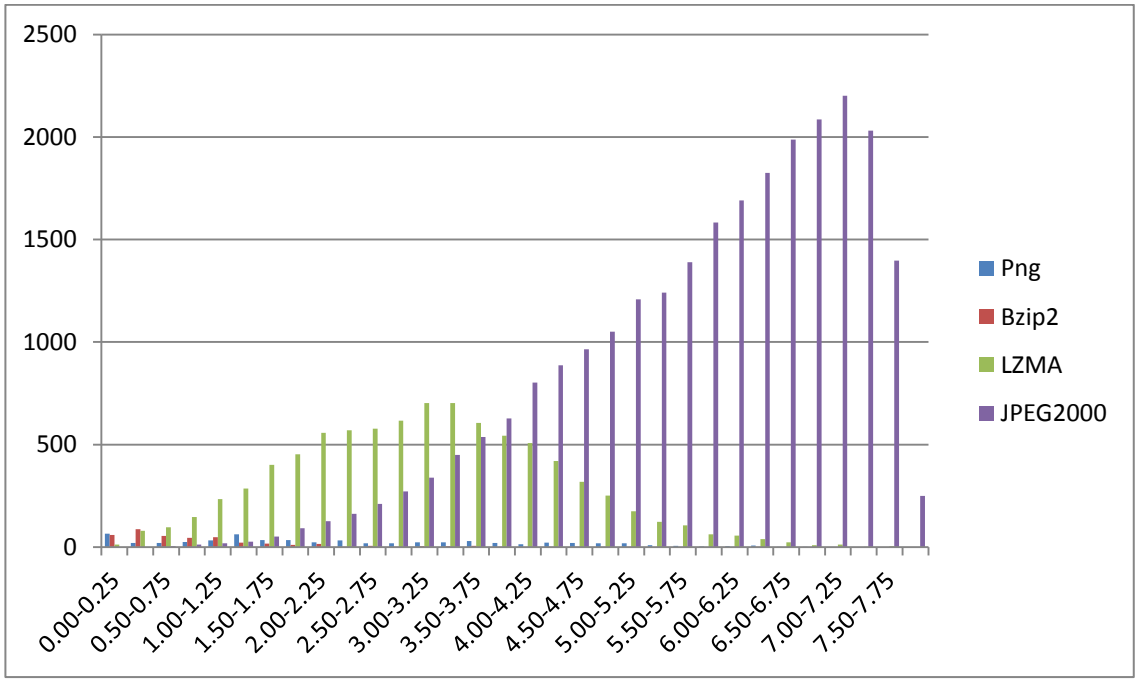
Bir resimde düz alanlara uygulanan Quadtree decomposition bölme işlemi gerçekleştirmezken, karmaşık sayılabilecek dokuların bulunduğu kısımlar veya desen çeşitliliği olan alanlar ise fazlasıyla küçük parçaya bölünecektir. Böylece bir resmin bölündüğü kare sayısı temel alınarak resmin karmaşıklık değeri tahmin/tespit edilebilir.

Uygulamadaki quadtree decomposition, Matlab'deki qtdecomp(I, threshold) fonksiyonu kullanılarak gerçekleştirilmiş, bölünen parça sayısı Matlab'den sonuç olarak alınmıştır. Fonksiyondaki I değeri gri tonlu resmi ifade ederken threshold değeri uygunluk kriterini gösterir. Eğer blok elemanlarının maksimum değeri ile minimum değeri arasındaki fark threshold değişkeninden büyükse bölme gerçekleşir. Aksi takdirde parça tekrar bölünmez.

Veritabanındaki tablolar Şekil 3.8'de verilmiştir. "CodecEntropi", "codecEntropiPER" ve "codecEntropiAVG" tabloları içinde kayıpsız sıkıştırma algoritmalarının isimleri ile her algoritmaya karşılık gelecek 1 ile 10 arasında 0.25 farkla karmaşıklık değerleri bulunur. Her değerin karşılığındaki sayı değeri ise 0'dır. Resim ve parçaları için bilgiler elde edildikten sonra hesaplanan üç farklı entropi değeri ve en iyi sıkıştırma oranına sahip algoritma tespit edilir. Üç tabloda da algoritma ve entropi değerinin aralığı tespit edilerek karşılığındaki sayı değeri bir arttırılır. Bu tablolarda sayıların saklanması ile hangi entropi değerinde hangi algoritmanın daha çok sıkıştırma yaptığı tespit edilebilir (Şekil 3.9).



Şekil 3.8. Karmaşıklık İstatistik veritabanı tabloları



Şekil 3.9. Farklı entropi değerlerinde algoritmaların en iyi çıkma sayıları

Şekil 3.9'da görüldüğü gibi LZMA algoritması çoğunlukla 0.25 ile 3.75 arasındaki codecEntropiAVG değerlerinde diğer algoritmalara göre daha iyi sonuç vermiştir. Eğer kullanılan sıkıştırma algoritmalarının hangi karmaşıklık değerinde daha iyi çıktığı tespit edilebilirse, sıkıştırma yapmaya gerek kalmadan seçilen resmin hangi algoritmayla daha iyi sıkıştırılabileceğine karar verilebilir.

Tablo 3.2. 205886_460s00.bmp dosyası için en iyi sonucu veren algoritmalar

dosyaAdi	codec	sayi
205886_460s00.bmp	Png	1
205886_460s00.bmp	Jpeg2000	10
205886_460s00.bmp	Bzip2	1
205886_460s00.bmp	LZMA	4

Kazanan tablosunda "dosyaAdi", "codec" ve "sayi" alanları bulunur (Tablo 3.2). Resim 16 parçaya bölündükten sonra her parçanın adı en iyi sıkıştırmayı yapan algoritmanın adıyla birlikte bu tabloya eklenir. Parçaların isimleri tek bir resim için aynı girilir. Böylece sorgu yazıldığında dosyanın ismiyle kaç parçanın hangi algoritma tarafından en iyi sıkıştırıldığı tespit edilebilir. Eğer 16 parça içerisinde birden fazla parça için en iyi sonucu veren bir algoritma olursa, ayrı ayrı saklanmak yerine, sayı değeri 1 arttırılır.

Örnekteki tabloda resmin 16 parçasından 10 tanesinde Jpeg2000 en fazla sıkıştırmayı gerçekleştirirken, 4 parçasında LZMA en iyi sonuca ulaşmıştır. Png ve Bzip2 birer parçada diğer algoritmalara göre daha iyi sıkıştırmıştır. Bu tablodaki verilerle, resmi tek parça sıkıştırmak yerine bölerek sıkıştırmanın kar getirip getiremeyeceği tespit edilmeye çalışılabilir. Eğer resim parçalandıktan sonra her parça en iyi sıkıştırma sağlayan algoritma ile sıkıştırılırsa ve bu kâr resmin bölünmesinden doğan zararı karşılayacak düzeydeyse resmi parçalayıp farklı algoritmalarla sıkıştırmak tercih edilebilir olacaktır.

Dosya tablosunda dosyanın yolu, dosyanın 16 parçasından hangisi olduğunu belirten dosyaNo (Parçalara ayrılmamış dosya 0 ile ifadeedilirken 1-16 arası ise dosyanın 16 parçasını satır sıralı olarak belirtir), dosyanın orijinal boyutu ve entropi değerleri, Matlab ile elde edilen quadtree decomposition sonucu resimdeki karelerin sayısı ve

kayıpsız sıkıştırma algoritmaları için sıkıştırılmış dosya boyutuyla birlikte süre bilgileri bulunur.

Test edilen resimlerden alınan entropi sonuçları, resime uygulanan quadtree decomposition sonrası resimdeki kare sayısı ve bu resmi en iyi sıkıştıran sıkıştırma yönteminin adının saklandığı tablodan bir örnek Tablo 3.3'te verilmiştir.

Tablo 3.3. Quadtree decomposition sonrası kare sayısı ve en iyi algoritma

Dosya Adı	**Entropi**	**Kare Sayısı**	**Yöntem**
3860996_460s.bmp	-1.901	476	LZMA
3858069_460s.bmp	-2.379	10844	PPMd
3994287_460s.bmp	-1.055	6068	BZip2
4060932_460s.bmp	-6.156	4788	Jpeg2000
4099788_460s.bmp	-6.905	720	Jpeg2000
4193030_460s.bmp	-7.913	856	Jpeg2000
4222843_460s.bmp	-6.032	196	PPMd

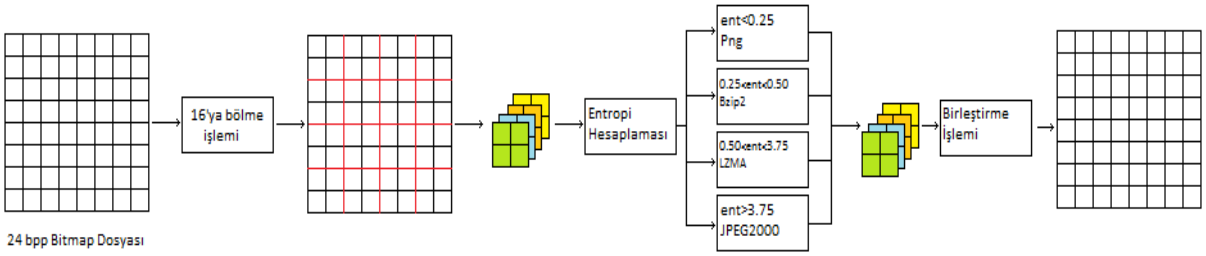
Veritabanındaki dosyaCodec tablosunda resmi parçalayarak sıkıştırmanın, resmi parçalayarak her parçanın farklı algoritmayla sıkıştırılmasının ve denenen Tahmin Algoritması-1'in sonuçları bulunur. "dosyaAdi" dosyanın adını, "dosyaBoyutu" sıkıştırılmamış dosyanın boyutunu, "boyutlar" dosyanın yükseklik ve genişlik bilgisini (yükseklikXgenişlik) saklar. "parcasizEnIyiBoyut" alanı resmi parçalara ayırmadan en iyi sıkıştıran algoritmanın elde ettiği sıkıştırılmış dosyanın dosya boyutunu, "parcasizEnIyiAlg" ise bu algoritmanın ismini saklar. "parcaliEnIyiBoyut" resmi 16 parçaya böldükten sonra her parçayı aynı algoritma ile sıkıştırma durumundaki en küçük dosya boyutunu saklarken, "parcaliEnIyiAlg" alanı ise bu boyutu elde eden algoritmanın ismini saklar. "Algoritma1Boyut" alanı ise denenen Tahmin Algoritması-1'in elde ettiği boyutu saklar.

Tahmin algoritması-1, elde edilen istatistiksel sonuçlar kullanılarak geliştirilmiştir. Karmaşıklık ve sıkıştırma sonuçları 2269 resim ve her resmin 16 parçası olmak üzere farklı boyutlarda toplam 38573 resim için elde edilmiş ve en iyi sonuç verme sayılarına göre seçilecek algoritmalar belirlenmiştir.

İstatistiklerden elde edilen sonuçlara göre karmaşıklık değeri 0.25'den küçük ise Png, 0.25-0.50 arasında ise BZip2, 0.50-3.75 arasında ise LZMA ve 3.75ten büyük ise JPEG2000 çoğunlukla en iyi sonuçları vermektedir (Tablo 3.5). Buna dayanarak algoritma resmi 16 parçaya bölecek ve her parçanın karmaşıklığını hesaplayarak, bulunduğu aralığa göre kayıpsız sıkıştırma algoritmasını seçecektir (Şekil 3.10).

Karmaşıklık olarak entropi değeri kullanılmakta olup, quadtree decomposition sonuçları istikrarlı sonuç göstermediği için algoritmaya henüz dâhil edilmemiştir.

Şekil 3.9'da da görüldüğü üzere 38573 resim için, 3.25-3.75 arasında LZMA ve JPEG2000 kazanma sayısı bakımından baş başa olmakla beraber çok az farkla LZMA önde çıkarken, 3.75-4.25 arasında da JPEG2000'in kazanma sayısı artmaktadır. Buna rağmen bu aralıklarda LZMA'nın ve JPEG2000'in kazanma sayıları birbirlerine çok yakındır.



Şekil 3.10. Tahmin algoritması-1'in şeması

Büyük boyutlu (>2mp) ve küçük boyutlu(<2mp) resimler için iki farklı sonuç elde edilmiştir.

Tablo 3.4. Algoritmaların Başarı Sonuçları

	Küçük Resimler	**Büyük Resimler**
Parçalı En İyi Boyut < Parçasız En iyi Boyut	%0	%6
En iyi parçalar İle < Parçasız en iyi	%59	%21
Tahmin Algoritması-1< Parçasız en iyi Boyut	%0	%5

Tablo 3.4'te Parçalı En iyi, bir resmin 16 parçaya bölündüğünde her parçanın aynı algoritmayla sıkıştırılmasından elde edilen en iyi sonucu gösterir. Küçük resimlerin bulunduğu sette bir resmi 16 parçaya bölerek her parçayı aynı algoritmayla sıkıştırma

yoluyla elde edilen sonuçlar, resmi tek parça halinde sıkıştırma sonuçlarına göre daha iyi sonuç vermemiştir. Büyük resimlerin olduğu sette ise 6 adet resimde iyi sonuç vermiştir.

En iyi parçalarda ise resim 16'ya bölündükten sonra her parçayı en iyi sıkıştıran algoritma seçilmiştir. Bu durumda her parça en iyi sıkıştırma performansı sağlayan algoritma ile sıkıştırılırsa 59 resimde parçalara bölme işlemi yapılmadan gerçekleştirilen sıkıştırmadan daha iyi sonuç vermiştir. Yine de 2169 resimde bu oran oldukça düşüktür. Büyük resimlerin bulunduğu sette ise bu oran %21'dir.

Parçalama işlemi büyük resimlerde küçük resimlere göre daha iyi sonuç vermektedir. Resimlerin %6'sı için kayıpsız sıkıştırma algoritmaları farklı parçaları bütüne göre daha iyi sıkıştırabilmektedir. Bu algoritmalar her parçaya başlık verisi koysa da resimleri parçalayarak sıkıştırmanın verdiği kazancın bu başlıkların kayıplarını karşılama olasılığı daha fazladır. Küçük resimlerde ise hâlihazırda bölünen parçalardan elde edilen kar başlık verisinin zararını karşılayamadığından verinin sıkışması yerine genişlemesi gözlenir.

Entropiye göre kullanılan Tahmin Algoritması-1 ise küçük resimlerde başarı sağlayamazken, büyük resimlerde %5 oranında başarı göstermiştir. Büyük resimler için bölme işleminin uygulanmasıyla elde edilebilecek maksimum değer her parçayı en iyi sıkıştıran algoritmalar seçildiğinde 21 olacaktır. Tahmin Algoritması-1 %100 doğrulukla çalışsa dahi en fazla 21 resimde kâr edebilecektir.

3.2.2. Karmaşıklık İstatistik-2

Karmaşıklık İstatistik-1'e ek olarak farklı karmaşıklık tespit yöntemleri araştırılarak bu yöntemler uygulamaya dâhil edildi. Bu yöntemlerin hesaplarının sonuçlarına veya bulduğu nesne sayısına göre karmaşıklık tespiti yapılarak, bu karmaşıklık sonuçlarına göre hangi algoritmanın daha başarılı olduğu araştırıldı. Quadtree decomposition sonucunu almak için Matlab konsolu kullanılırken, diğer yöntemler için AForge.net'in C# kütüphaneleri kullanılmıştır.

Karmaşıklık tespiti için entropi ve quadtree decomposition haricinde blob sayısı, fourier dönüşümü sonrası beyaz piksel sayısı, moravec corner sayısı, quadriliteral sayısı ve Susan Corners Detector ile bulunan köşe sayısı kullanılmaktadır (Ek-C). 9 adet farklı

algoritmanın tümünü hesaba katarak hangi kayıpsız sıkıştırma algoritmasının seçileceğinin kararını vermek, bir fonksiyon veya karar mekanizması belirlenmeden mümkün değildir. Eldeki verilerin miktarının büyük olması ve karmaşıklık tespiti algoritmalarının belirli özellikteki resimlere göre benzer sonuç vermemesi de bu mekanizmayı belirlemeyi zorlaştırmaktadır. Bu yüzden, dosya boyutu, algoritma sonuçları ve en iyi sonuç veren kayıpsız sıkıştırma algoritmasının verileri bir araya getirilerek, Weka'nın kullanabileceği bir eğitim seti oluşturulmuştur (Ek-A).

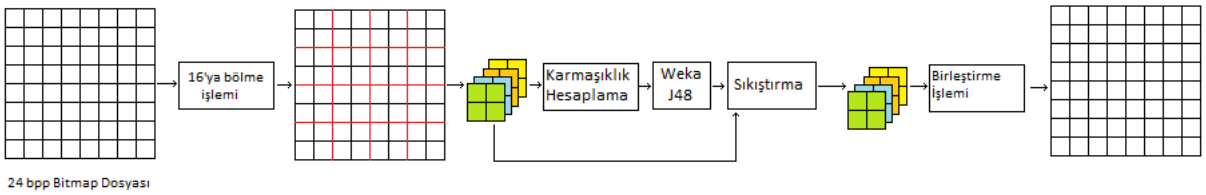
Eğitim seti, 2169 adet farklı boyutlarda ve farklı niteliklerdeki resmi ve her resmin 16 parçası için toplam 36783 resimde karmaşıklık hesap değerlerini ve bu değerlere karşılık gelen en iyi sıkıştıran kayıpsız sıkıştırma algoritmasını içermektedir. Makine öğrenmesi kullanılarak bu verilere göre hangi karmaşıklıkta hangi algoritmanın seçileceği belirlenmiştir. Sonuç değerleri sayısal değerler değil, nominal değerler (Deflate, Jpeg2000, LZMA vs.) oldukları için karar algoritması olarak J48 kullanılmıştır. Bu uygulama için oluşturulan J48 karar ağacının bir kısmı Şekil 3.11'de gösterilmişir.

```
entropiAvg <= 4.920847
|   moravecCornerSayisi <= 16
|   |   entropiPer <= 2.988
|   |   |   moravecCornerSayisi <= 6
|   |   |   |   dosyaBoyutu <= 3129.375
|   |   |   |   |   susanCornerSayisi <= 0: Png (148.0/27.0)
|   |   |   |   |   susanCornerSayisi > 0
|   |   |   |   |   |   moravecCornerSayisi <= 3
|   |   |   |   |   |   |   fourierBeyazSayisi <= 16
|   |   |   |   |   |   |   |   moravecCornerSayisi <= 2
|   |   |   |   |   |   |   |   |   entropi <= 1.476188: LZMA (3.0/1.0)
|   |   |   |   |   |   |   |   |   entropi > 1.476188: Png (7.0)
|   |   |   |   |   |   |   |   moravecCornerSayisi > 2
|   |   |   |   |   |   |   |   |   fourierBeyazSayisi <= 1: Png (6.0/1.0)
|   |   |   |   |   |   |   |   |   fourierBeyazSayisi > 1: LZMA (2.0)
|   |   |   |   |   |   |   fourierBeyazSayisi > 16: LZMA (2.0)
                                                .
                                                .
                                                .
```

Şekil 3.11. J48 karar ağacının bir parçası

Tahmin algoritması-2 öncelikle resmi 16 parçaya böler, daha sonra her parçasının karmaşıklık değerini hesaplayıp boyut ve karmaşıklık bilgisini sırasıyla karar ağacına yollar. Ağaçtan dönen sonuca göre sıradaki parçayı karar verilen sıkıştırma algoritmasına

göre sıkıştırıp boyut bilgisini hesaplar. Sonuç verilerinin incelenmesi için algoritmanın elde ettiği sonuçlar veri tabanına yazılır (Şekil 3.12).



Şekil 3.12. Tahmin algoritması-2'nin şeması

Tablo 3.5. Algoritma Sonuçları

	Sonuçlar
Parçalı En iyi < Parçasız en iyi Alg.	%8
En iyi parçalar < Parçasız en iyi Alg.	%22
Tahmin Alg. 1 < Parçasız en iyi Alg.	%0
Tahmin Alg. 2 <Parçasız en iyi Alg.	%0
Tahmin Alg. 1 = En iyi parçalar	%44
Tahmin Alg. 2 = En iyi parçalar	%0

50 resim üzerinde yapılan teste göre bir resmi parçalayarak aynı algoritma ile sıkıştırmak, 4 resimde daha iyi sonuç vermiştir. Her parçayı en iyi sıkıştırabilecek algoritmaya vermek ise 11 resimde daha iyi sonuç vermiştir.

Tahmin Algoritması-1 ile Weka kullanılarak yapılan Tahmin Algoritması-2 ise resmi parçalamadan sıkıştırmaya göre daha iyi sonuç verememişlerdir.

Resmi parçalayarak aynı algoritmayla sıkıştırmanın şişirme yaptığı durumlarda parçaları en iyi sıkıştıracak algoritmaya vererek elde edilen kar 7 resimde bu şişirmenin verdiği zararı karşıladığı gibi ayrıca bu resimleri parçasız sıkıştırmadan daha fazla sıkıştırmıştır.

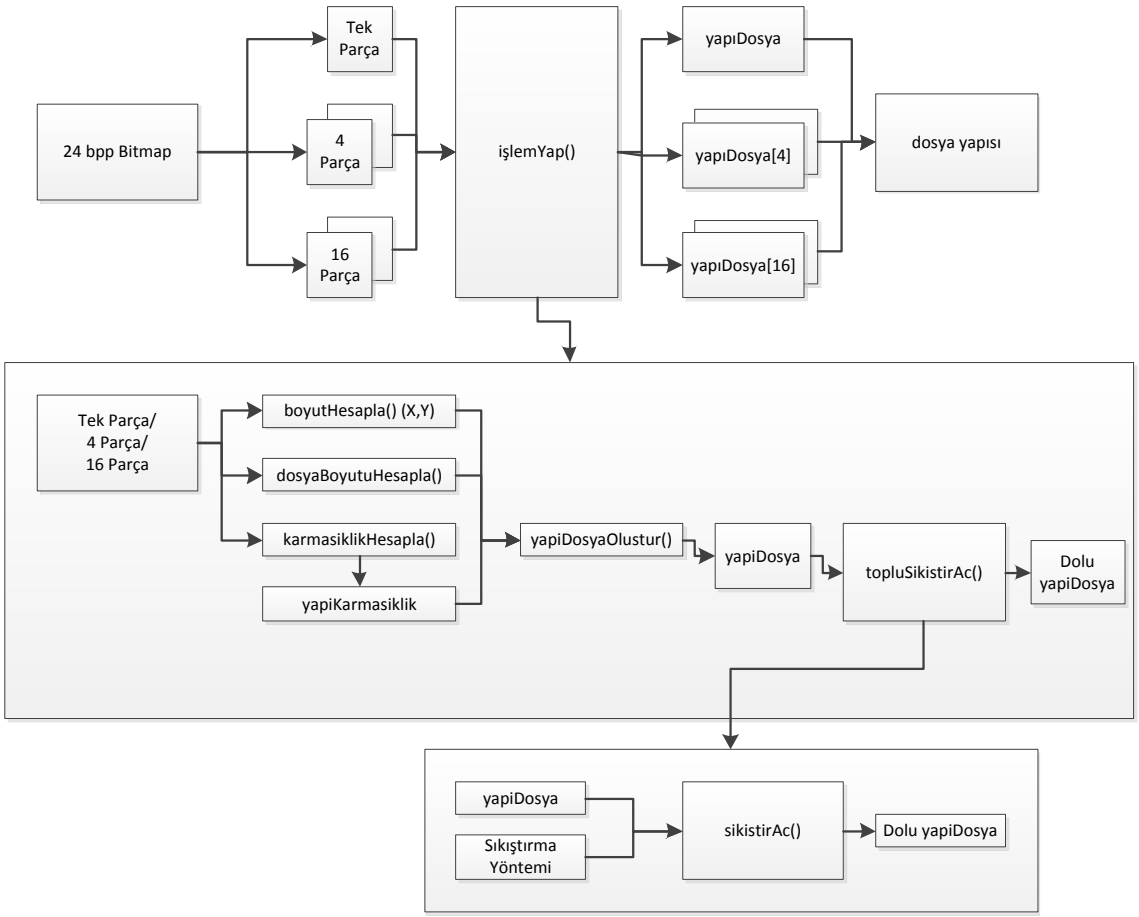
Entropi hesabına göre karar veren Tahmin Algoritması-1 22 resimde, resmi parçalayarak parçaları en iyi sıkıştıracak algoritmalara verme işlemi ile aynı oranı

yakalamıştır. Weka kullanan Tahmin Algoritması-2 ise hiçbir resimde parçalı en iyi sıkıştırma algoritmasını yakalayamamıştır.

3.2.3. Tahmin Algoritması-3

Tahmin Algoritması-3 deneyi iki ayrı bölümden oluşur. Birinci bölümde bir resmin istatistiksel sonuçları elde edilirken, ikinci bölümde ise bu istatistiksel sonuçları kullanan bir tahmin algoritması çalışır.

Birinci bölümde resim tek parça, 4 parça ve 16 parça olmak üzere 3 farklı şekilde işlenir. Her parça için sıkıştırma – açma işlemleri gerçekleştirilir. Her parçanın 9 farklı yöntem ile karmaşıklığı hesaplanır, kayıpsız sıkıştırma algoritmaları olarak JPEG2000 kayıpsız, LZMA, Png ve PPMd kullanılırken kayıplı sıkıştırma algoritmaları içinse JPEG, JPEG2000 ve JPEG XR kullanılır. Kayıpsız sıkıştırma algoritmaları içine daha önceki çalışmalarda kullanılan diğer 5 algoritmanın eklenmemesinin nedeni, kullanılan bu 4 algoritmaya göre daha düşük performans sergilemeleri ve daha kötü sonuçlar vermeleridir. Kayıplı algoritmalardan JPEG ve JPEG XR için 1-100 arasındaki tüm kalite faktörleri kullanılarak sonuçlar alınır ve her kalite faktörü için bozulma oranları (MSE, RMSE, PSNR, PAE, MAE, SSIM) hesaplanır. JPEG2000 için ise 1-100 arası sıkıştırma oranları kullanılır ve her sonuç için bozulma oranları bu algoritma için de hesaplanıp bir dosyada saklanır (Şekil 3.13). Dosya yapısı Şekil 3.14'de verilmiştir.



Şekil 3.13. Tahmin Algoritması-3'ün çalışma şeması

Şekil 3.14. Dosya yapısının hiyerarşisi

Tablo 3.6'daki verilere göre en kararlı sonuçları LZMA kayıpsız sıkıştırma algoritması vermektedir. Entropi değerlerinin 4 üzeri olması durumunda, çoğunlukla resmin boyutundan bağımsız olarak LZMA 16 parçada en iyi sonuçları verirken, entropi düşükse resmin 4 parçaya bölünmesiyle LZMA daha iyi performans verir. Tablonun tamamı incelendiğinde LZMA için (27 farklı resim üzerindeki sonuçlarda) parçalama işlemi %50'lik bir oranla, tek parça sıkıştırmaya göre daha iyi sonuç sağlar. Diğer algoritmalarda da 16 parçaya bölmek veya 4 parçaya bölmek bazı resimlerde daha iyi sonuçlar verse de çoğunlukla bu durum kararsızlık gösterdiğinden, yarı kayıplı sıkıştırma

yapacak tahmin algoritmasının kullanacağı algoritmalardan biri LZMA olarak belirlenmiştir.

Tablo 3.6. Karmaşıklık ve boyut değerlerine göre en iyi sonuç veren parça sayıları

Dosya Adı	Boyutlar	Entropi	Entropi Avg	Entopi Per	J2K	LZMA	PPMd	Png
4322118_460s	1024x1024	5.94	5.60	5.60	1	16	16	1
4345224_460s	1024x1024	3.19	2.98	2.83	1	1	1	1
4362666_460s_v1	1024x1024	6.88	7.62	7.68	1	16	1	4
4348731_460s	1024x1024	7.20	6.76	6.90	1	16	16	1
4400356_460s	1024x1024	0.22	1.25	1.25	1	4	1	1
4400526_460s	2048x2048	4.76	4.97	4.97	1	16	1	1
4401630_460s	2048x2048	6.60	7.50	7.60	1	16	1	16
4402434_460s	2048x2048	6.96	6.64	6.90	1	16	16	16
4408644_460s	2048x2048	6.69	6.26	6.83	4	16	16	16

Deneylerin istatistik toplama kısmından alınan sonuçlar, kayıplı ve kayıpsız sıkıştırma algoritmalarının performansları ve karmaşıklık hesapları göz önüne alınarak, kayıpsız olarak LZMA ve kayıplı olarak Jpeg XR algoritmalarını kullanan ve istenen boyutta sıkıştırma yapabilen bir görüntü sıkıştırma algoritması denenmiştir.

Algoritma öncelikle m5p karar algoritmasını kullanarak, resmin bölünmeden LZMA ile sıkıştırılmasıyla istenen boyuta düşürülebileceğine veya daha iyi sonucu sağlayabileceğine karar verirse, resmi LZMA ile sıkıştırır. Eğer bu sonucu elde edemeyeceğine karar verirse, daha önce entropiye göre LZMA'nın daha iyi sonuç verdiği parça sayısı tespiti için oluşturulan tablonun sonucuna göre 4 veya 16 parçaya böler (Tablo 3.6).

Her bir parça için öncelikle üç farklı entropi değerleri hesaplanır. Daha sonra liste entropi değerlerine göre büyükten küçüğe sıralanır. Sıralamanın amacı yüksek entropi değerlerine sahip parçaların karmaşıklığının daha fazla olması sebebiyle kayıplı sıkıştırma algoritmalarının kullanılmasının bu parçalarda daha belirgin olmasıdır. Böylelikle entropisi yüksek parçalar daha iyi kalite faktörlerinde sıkıştırılarak, gözle görülen kayıp azaltılacaktır.

Her adımda kalan parça sayısı ve istenen dosya boyutundan kalan dosya boyutu bilgisi yeniden hesaplanır. Bu verilerin birbirine oranı ile parça başına düşen dosya boyutu hesaplanır. Parçalar için tekrar m5p algoritması kullanılır ve istenen dosya boyutuna düşmenin LZMA ile mümkün olup olmadığına karar verilir. Karar için m5p algoritmasının kullanacağı üç farklı entropi değeri ve sıkıştırma oranı içeren eğitim verisi kullanılır. Eğer karar algoritması LZMA'nın parçayı istenen dosya boyutuna düşürmesinin mümkün olduğuna veya daha iyi sonuç vereceğine karar verirse bu parça LZMA ile sıkıştırılır. Dosya boyutu alındıktan sonra kalan dosya boyutundan bu miktar düşürülür. Böylece LZMA'nın parçayı istenen dosya boyutundan daha düşük boyutlara sıkıştırması durumunda, diğer parçalara kalacak olan parça başı dosya boyutu artmış olur. Bu sayede herhangi bir parçadan edilen kâr diğer parçaların daha iyi kalite faktörlerinde sıkıştırılması için kullanılabilir.

Eğer M5p algoritması LZMA'nın istenen sonucu veremeyeceğine karar verirse algoritma parçayı JPEG XR ile kayıplı sıkıştırır. İstenen dosya boyutunu sağlayabilecek kalite faktörünün tespiti için J48 karar ağacı ile önceden hazırlanmış bir eğitim verisi kullanılmaktadır. Bu eğitim verisinde Jpeg XR için kalite faktörü, entropi ve sıkıştırma oranı sonuçları bulunmaktadır. Bu sonuçlara göre istenen dosya boyutunu sağlayacak kalite faktörü belirlenir ve ardından sıradaki parça Jpeg XR kullanılarak bu kalite faktöründe sıkıştırılır.

Bütün parçaların sıkıştırma işlemi bittikten sonra parçalar birleştirilerek resim oluşturulur.

```
@relation resim

@attribute entropi real
@attribute entropiAvg real
@attribute entropiPer real
@attribute sikistirmaOrani real

@data
0.36749494,0.515397224,0.515397224,1.220523228
0.333848957,0.391224147,0.391224147,0.893722202
0.69476733,1.004662985,1.004662985,2.79839692
0,0,0,0.027591082
```

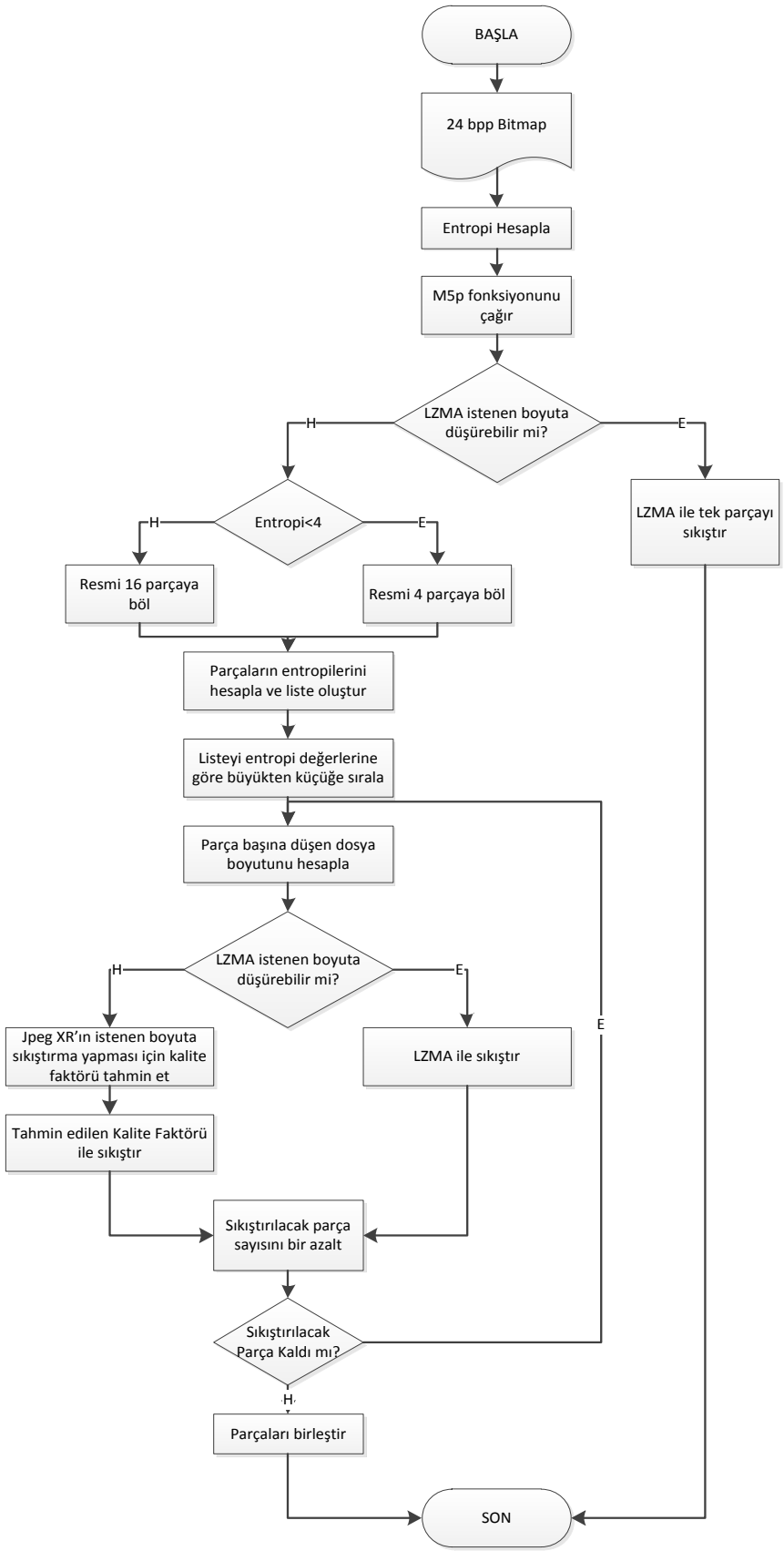
Şekil 3.15. LZMA seçim kararı için M5p algoritması eğitim verisi örneği

```
@relation resim

@attribute KaliteFaktoru
{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,3
8,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,7
3,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100}
@attribute entropiAvg real
@attribute sikistirmaOrani real

@data
66,0,0.297464685602709
0,0.515397224461887,0.158847625169195
1,0.515397224461887,0.173247860150513
2,0.515397224461887,0.200967517774595
3,0.515397224461887,0.230467336897471
```

Şekil 3.16. Jpeg XR kalite faktörü kararı için J48 karar ağacı eğitim verisi örneği



Şekil 3.17. Tahmin algoritmasının Akış diyagramı

```
if (entropiAvg <= 3.318)
        if (entropi <= 0.469)
            sikistirmaOrani = 0.0926 * entropi + 0.6597 * entropiPer - 0.01;
        else
            if (entropiAvg <= 1.929)
                sikistirmaOrani = -0.0477 * entropi + 1.0503 * entropiPer + 1.458;
            else
                sikistirmaOrani = -0.0477 * entropi + 1.8689 * entropiAvg + 0.9314 *
                                                                    entropiPer - 0.7251;
            else
                sikistirmaOrani = -2.7603 * entropi + 15.0652 * entropiPer - 26.5451;
return sikistirmaOrani;
```

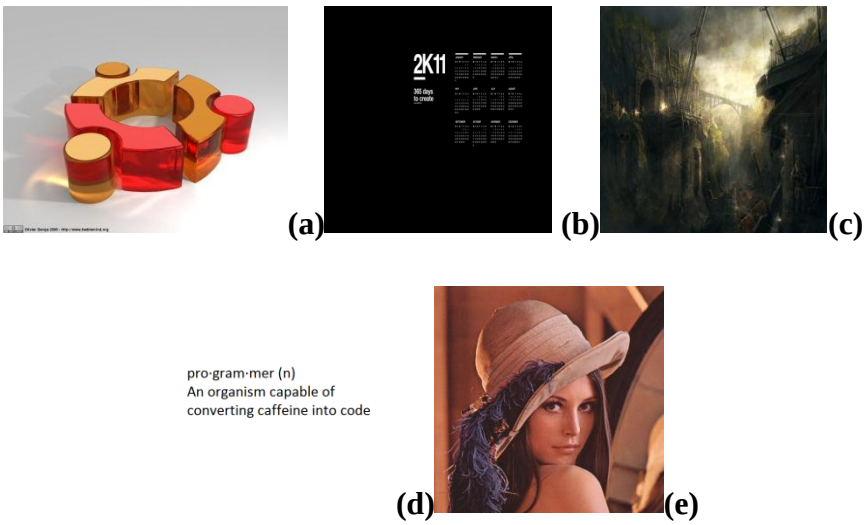
Şekil 3.18. M5P algoritmasına göre LZMA'nın entropiye göre sıkıştırma oranı tespiti

BÖLÜM 4

UYGULAMA SONUÇLARI

Bu bölümde, üçüncü bölümde verilen deneysel çalışmaların test sonuçları ve karşılaştırmaları yer almaktadır. İlk alt bölümde (4.1) resim üzerinde uygulanan kanal sıralı kodlama yöntemlerinin sonuçları ve karşılaştırmaları yer alırken, ikinci alt bölümde (4.2) tahmin algoritması ile sıkıştırma yapan Tahmin Algoritması-3'ün JPEG2000 ile karşılaştırma sonuçları verilmiştir.

Karşılaştırma için kullanılan resim dosyaları (Şekil 4.1)'de verilmiştir.



(a) **(b)** **(c)** **(d)** **(e)**

Şekil 4.1. Karşılaştırma için kullanılan Ubuntu.bmp 1280x1024 (a), 2k11.bmp 1024x1024(b), Bridgetolimansk.bmp 1024x1024 (c), programmer.bmp 512x512 (d), lena.bmp 512x512 (e)

4.1. Kanalları Farklı Şekilde Kodlama Yöntemlerinin Sonuçları

LZMA, LZMA2, PPMd, Bzip2, Deflate, Deflate64 ile sıkıştırma sonuçlarını elde etmek için Winzip v16.5 üzerinde "WinZip Command Line" ve 7z v9.2 konsol uygulaması kullanılmış, 7z sıkıştırma parametresi olarak "Ultra" seçeneği seçilmiştir. Süre sonuçları için sıkıştırma işlemi 10 defa tekrarlanmış ve ortalama alınmıştır. Sonuçların alınmasında, Intel Core i5-2500K 3.3GHz işlemcisi ve 4GB DDR3 (2.98 kullanılabilir) belleğine sahip, işletim sistemi olarak ta Windows 7 Home Edition yüklenmiş olan bir bilgisayar kullanılmıştır. DSBB için her renk kanalında sonuçlar alınmış ve kanallardaki sıkıştırma oranlarının birbirine çok yakın sonuçlar vermeleri nedeniyle tablolarda sadece bir renk kanalının sonuçları verilmiştir.

Tablo 4.1. Ubuntu.bmp için sıkıştırma sonuçları (bpp/ms)

	LZMA WinZip	**LZMA 7z**	**LZMA2 7z**	**PPMd WinZip**	**PPMd 7z**	**BZip2 WinZip**	**BZip2 7z**	**Deflate WinZip**	**Deflate64 WinZip**
Orijinal Resim	2,492 69ms	2,498 754ms	2,498 776ms	3,234 358ms	3,266 347ms	3,14 514ms	3,082 687ms	3,741 214ms	3,234 367ms
EBMP +RLE	2,748 411ms	2,749 749ms	2,749 751ms	2,559 466ms	2,528 578ms	2,823 356ms	2,817 680ms	4,395 377ms	2,559 445ms
EBMP	2,396 294ms	2,431 125ms	2,432 110ms	2,322 573ms	2,378 639ms	2,733 269ms	2,734 629ms	**3,297 416ms**	2,322 590ms
SSBB 64px	2,351 489ms	2,377 355ms	2,377 367ms	2,678 315ms	2,745 344ms	3,021 152ms	3,021 440ms	3,324 359ms	2,678 313ms
SSBB 128px	**2,315 392ms**	**2,345 295ms**	**2,345 286ms**	2,524 300ms	2,596 336ms	2,881 154ms	2,881 303ms	3,218 399ms	2,524 303ms
SSBB 256px	2,334 328ms	2,365 186ms	2,368 250ms	2,417 303ms	2,481 316ms	2,794 158ms	2,795 260ms	3,22 408ms	2,417 302ms
DSBB 64fark-R	2,395 322ms	2,434 197ms	2,434 211ms	**2,321 302ms**	**2,377 327ms**	2,695 161ms	2,691 266ms	3,303 367ms	**2,321 303ms**
DSBB 128fark-R	2,395 308ms	2,433 172ms	2,433 227ms	2,322 306ms	2,378 338ms	**2,689 140ms**	**2,688 245ms**	3,305 358ms	2,322 291ms
DDBB 128fark	2,398 281ms	2,433 188ms	2,433 179ms	2,351 299ms	2,411 322ms	2,777 155ms	2,775 224ms	3,301 383ms	2,351 305ms
DDBB 200fark	2,404 269ms	2,442 185ms	2,441 211ms	2,332 303ms	2,388 328ms	2,748 135ms	2,748 238ms	3,309 373ms	2,331 305ms

Tablo 4.1'de görüldüğü gibi Ubuntu.bmp resminde, LZMA ve LZMA2 için resmi 128x128 px karelere bölmek en iyi sonucu verirken, PPMd için resmi R kanalının 64 farkı sağladığı ilk parçanın büyüklüğünde parçalara (1280x259) bölmek en iyi sonucu vermiştir. Bzip2 için ise en iyi sonucu veren parça boyutu R kanalının 128 farkı ile elde edilen 1280x367 olmuştur. Dinamik parçalama işleminin en iyi sonucu verdiği yöntem ise Deflate64 olmuştur. Bütün denemeler arasında en iyi sonucu veren ise resmin 128x128 px karelere bölünüp LZMA (Winzip)'e verilmesi ile elde edilmiştir.

Tablo 4.2. 2k11.bmp için sıkıştırma sonuçları (bpp/ms)

	LZMA WinZip	**LZMA 7z**	**LZMA2 7z**	**PPMd WinZip**	**PPMd 7z**	**BZip2 WinZip**	**BZip2 7z**	**Deflate WinZip**	**Deflate64 WinZip**
Orijinal Resim	0,295 336ms	0,293 317ms	0,293 308ms	**0,32 99ms**	0,321 77ms	**0,306 110ms**	**0,307 456ms**	0,507 155ms	**0,32 99ms**
EBMP +RLE	0,283 280ms	0,283 271ms	0,283 266ms	0,409 77ms	**0,293 93ms**	0,392 104ms	0,391 224ms	1,132 625ms	0,409 74ms
EBMP	**0,256 325ms**	**0,256 301ms**	**0,256 305ms**	0,324 107ms	0,304 84ms	0,385 145ms	0,373 121ms	0,873 208ms	0,324 100ms
SSBB 64px	0,301 339ms	0,301 333ms	0,3 327ms	0,377 104ms	0,344 97ms	0,442 174ms	0,433 656ms	0,371 79ms	0,377 107ms
SSBB 128px	0,272 327ms	0,272 297ms	0,272 286ms	0,344 108ms	0,318 94ms	0,407 141ms	0,396 670ms	**0,328 77ms**	0,344 88ms
SSBB 256px	0,267 345ms	0,266 321ms	0,266 333ms	0,344 107ms	0,321 91ms	0,396 149ms	0,385 673ms	0,936 135ms	0,344 104ms
DSBB 64fark-R	0,258 324ms	0,258 302ms	0,258 311ms	0,327 105ms	0,307 94ms	0,388 143ms	0,377 667ms	0,876 172ms	0,327 97ms
DSBB 128fark-R	0,259 339ms	0,258 321ms	0,258 316ms	0,327 88ms	0,307 91ms	0,388 152ms	0,377 674ms	0,876 186ms	0,327 96ms
DDBB 128fark	0,346 381ms	0,346 347ms	0,347 350ms	0,459 100ms	0,433 111ms	0,529 165ms	0,515 801ms	1,044 175ms	0,459 116ms
DDBB 200fark	0,288 355ms	0,289 328ms	0,289 331ms	0,367 105ms	0,341 93ms	0,421 149ms	0,409 700ms	0,95 192ms	0,367 102ms

Tablo 4.2'de verilen sonuçlar incelendiğinde 2k11.bmp için en iyi sıkıştırma oranının, RLE kullanmayan EBMP dönüşümü ve ardından LZMA yöntemleri ile sıkıştırma ile elde edildiği görülmektedir.

Tablo 4.3. Bridgetolimansk.bmp için sıkıştırma sonuçları (bpp/ms)

	LZMA WinZip	**LZMA 7z**	**LZMA2 7z**	**PPMd WinZip**	**PPMd 7z**	**BZip2 WinZip**	**BZip2 7z**	**Deflate WinZip**	**Deflate64 WinZip**
Orijinal Resim	13,804 227ms	13,935 138ms	13,936 202ms	**12,589 418ms**	**12,591 971ms**	**12,475 322ms**	**12,483 440ms**	17,143 233ms	12,589 126ms
EBMP+ RLE	15,569 263ms	15,568 265ms	15,571 284ms	12,819 511ms	13,005 166ms	13,676 870ms	13,673 727ms	22,049 841ms	12,819 489ms
EBMP	13,595 343ms	14,041 306ms	14,044 316ms	12,614 90ms	12,621 75ms	13,592 275ms	13,568 285ms	16,727 323ms	12,614 75ms
SSBB 64px	**13,051 338ms**	**13,34 275ms**	**13,343 253ms**	12,858 790ms	12,866 946ms	13,934 338ms	13,851 743ms	15,991 342ms	12,858 876ms
SSBB 128px	13,093 299ms	13,404 201ms	13,405 188ms	12,705 688ms	12,713 921ms	13,764 347ms	13,672 685ms	**15,665 375ms**	12,705 866ms
SSBB 256px	13,286 344ms	13,652 235ms	13,655 243ms	12,654 415ms	12,66 955ms	13,658 344ms	13,61 718ms	15,896 413ms	12,654 129ms
DSBB 64fark-R	13,912 442ms	14,365 400ms	14,366 405ms	12,853 124ms	12,858 107ms	13,912 374ms	13,9 838ms	17,354 378ms	12,853 114ms
DSBB 128fark-R	13,593 349ms	14,031 252ms	14,033 235ms	**12,587 37ms**	12,593 509ms	13,53 347ms	13,527 542ms	16,761 388ms	**12,587 46ms**
DDBB 128fark	13,476 283ms	13,881 196ms	13,884 194ms	12,668 124ms	12,676 977ms	13,663 335ms	13,63 481ms	16,475 383ms	12,668 598ms
DDBB 200fark	13,596 314ms	14,043 241ms	14,044 267ms	12,616 43ms	12,623 410ms	13,595 336ms	13,574 529ms	16,73 389ms	12,616 144ms

Bridgetolimansk.bmp için BZip2 ile işlem görmemiş resim dosyasının sıkıştırılması en iyi sonucu vermiştir. En iyi sonuca en yakın sonucu veren yöntem ise R kanalında 128 fark arayan DSBB'nin PPMd ile birlikte kullanılması olmuştur (Tablo 4.3).

Tablo 4.4'te görüldüğü gibi Programmer.bmp için tüm bölümlendirme algoritmaları, orijinal dosyanın sıkıştırma yöntemleriyle sıkıştırılmasından daha kötü sonuçlar vermiştir. Kendi aralarında ise en iyi sıkıştırmayı sağlayan algoritma PPMd (7z) için EBMP+RLE olmuştur.

Tablo 4.4. programmer.bmp için sıkıştırma sonuçları (bpp/ms)

	LZMA WinZip	LZMA 7z	LZMA2 7z	PPMd WinZip	PPMd 7z	BZip2 WinZip	BZip2 7z	Deflate WinZip	Deflate64 WinZip
Orijinal Resim	**0,103 94ms**	**0,103 73ms**	**0,103 96ms**	**0,085 32ms**	**0,08 19ms**	**0,076 35ms**	**0,076 85ms**	**0,106 33ms**	**0,085 29ms**
EBMP+ RLE	0,227 93ms	0,227 73ms	0,227 74ms	0,192 26ms	0,186 15ms	0,194 30ms	0,192 216ms	0,277 49ms	0,192 23ms
EBMP	0,21 101ms	0,21 98ms	0,21 61ms	0,187 35ms	0,19 15ms	0,192 40ms	0,19 105ms	0,235 42ms	0,187 30ms
SSBB 64px	0,26 127ms	0,259 104ms	0,259 125ms	0,256 30ms	0,262 18ms	0,275 41ms	0,273 157ms	0,341 50ms	0,256 30ms
SSBB 128px	0,245 111ms	0,245 108ms	0,245 104ms	0,212 43ms	0,216 21ms	0,227 40ms	0,224 152ms	0,291 41ms	0,212 30ms
SSBB 256px	0,239 111ms	0,24 91ms	0,24 96ms	0,205 40ms	0,209 22ms	0,211 41ms	0,208 144ms	0,276 46ms	0,205 40ms
DSBB 64fark-R	0,218 116ms	0,217 99ms	0,217 87ms	0,194 40ms	0,198 19ms	0,205 40ms	0,202 124ms	0,252 40ms	0,194 37ms
DSBB 128fark-R	0,218 110ms	0,217 90ms	0,217 96ms	0,194 36ms	0,198 26ms	0,205 36ms	0,203 125ms	0,252 37ms	0,194 43ms
DDBB 128fark	0,501 119ms	0,5 107ms	0,5 96ms	0,536 38ms	0,539 26ms	0,569 44ms	0,566 191ms	0,682 41ms	0,536 43ms
DDBB 200fark	0,452 110ms	0,451 91ms	0,451 93ms	0,445 41ms	0,447 23ms	0,489 57ms	0,484 174ms	0,558 50ms	0,445 40ms

Bzip2 ve Deflate yöntemleri Lena.bmp üzerinde hiçbir ön işlem yapmadığı takdirde en iyi sonucu vermiştir. Kanalları farklı şekilde kodlama yöntemleri diğer veri sıkıştırma yöntemleri için kâr sağlayabilse de orijinal resmin Bzip2 yöntemi ile sıkıştırılmasından daha iyi sonuçlar elde edememiştir.

Tablo 4.5. Lena.bmp için sıkıştırma sonuçları (bpp/ms)

	LZMA WinZip	**LZMA 7z**	**LZMA2 7z**	**PPMd WinZip**	**PPMd 7z**	**BZip2 WinZip**	**BZip2 7z**	**Deflate WinZip**	**Deflate64 WinZip**
Orijinal Resim	16,772 235ms	16,874 191ms	16,875 199ms	16,01 261ms	16,024 227ms	**15,24 101ms**	**15,241 604ms**	**19,405 58ms**	16,01 260ms
EBMP+ RLE	18,909 358ms	18,908 289ms	18,913 305ms	16,139 522ms	16,176 547ms	16,773 218ms	16,732 460ms	26,038 501ms	16,139 512ms
EBMP	15,909 223ms	16,66 190ms	16,663 196ms	15,897 245ms	15,896 224ms	16,554 101ms	16,544 785ms	20,185 68ms	15,897 243ms
SSBB 64px	**15,49 222ms**	**15,992 197ms**	**15,998 189ms**	16,186 243ms	16,191 208ms	16,957 108ms	16,869 726ms	20,313 71ms	16,186 246ms
SSBB 128px	15,541 235ms	16,149 196ms	16,153 189ms	15,969 243ms	15,972 206ms	16,737 105ms	16,67 714ms	20,218 69ms	15,969 249ms
SSBB 256px	15,683 232ms	16,403 185ms	16,409 186ms	15,918 238ms	15,918 205ms	16,61 104ms	16,565 711ms	20,081 74ms	15,918 246ms
DSBB 64fark-R	15,951 225ms	16,718 199ms	16,718 197ms	15,928 252ms	15,926 222ms	16,588 107ms	16,582 712ms	20,818 70ms	15,928 249ms
DSBB 128fark-R	15,916 208ms	16,683 189ms	16,684 189ms	**15,881 228ms**	**15,875 208ms**	16,57 108ms	16,564 696ms	20,388 76ms	**15,881 225ms**
DDBB 128fark	15,902 221ms	16,653 197ms	16,655 196ms	15,912 241ms	15,911 203ms	16,582 113ms	16,577 714ms	20,185 69ms	15,912 233ms
DDBB 200fark	15,915 205ms	16,665 193ms	16,668 192ms	15,904 238ms	15,903 208ms	16,563 104ms	16,556 695ms	20,194 73ms	15,904 235ms

4.2. Tahmin Algoritması-3 Sonuçları

Tahmin Algoritması-3'ün başarımını test etmek amacıyla Şekil 4.1'de verilen resimler kullanılmıştır. Bu resimler için istenilen sıkıştırma oranlarına karşılık elde edilen sıkıştırma oranları ve R, G, B kanalları üzerinden hesaplanan bozulma miktarları Tablo 4.6 ile Tablo 4.10 arasındaki tablolarda verilmiştir.

Tablo 4.6. Ubuntu.bmp için JPEG2000 ve Tahmin Algoritması-3'ün sıkıştırma oranı (bpp) ve bozulma ölçümleri sonuçları

	İstenen Oran (%)	J2K Sıkışt. Oranı	TA3 Sıkışt. Oranı	J2K R	J2K G	J2K B	Alg R	Alg G	Alg B
SSIM	1	0,241268	0,248561	0,9966	0,9966	0,9966	0,9791	0,9791	0,9791
	5	1,225199	0,817132	0,9997	0,9997	0,9997	0,9981	0,9981	0,9981
	10	2,451671	1,82096	1,0000	1,0000	1,0000	0,9999	0,9999	0,9999
	15	2,801626	2,262344	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
	20	2,801626	2,735175	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
	25	2,801626	2,993898	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
PSNR	1	0,241268	0,248561	41,05	44,63	42,97	33,71	34,32	33,53
	5	1,225199	0,817132	52,20	55,16	53,13	44,42	46,03	45,62
	10	2,451671	1,82096	60,22	63,16	60,89	54,36	56,15	55,19
	15	2,801626	2,262344	92,00	103,57	102,32	58,00	59,53	58,55
	20	2,801626	2,735175	92,00	103,57	102,32	61,73	63,75	62,60
	25	2,801626	2,993898	92,00	103,57	102,32	∞	∞	∞
MSE	1	0,241268	0,248561	5,104	2,241	3,282	27,682	24,050	28,838
	5	1,225199	0,817132	0,391	0,198	0,316	2,350	1,623	1,783
	10	2,451671	1,82096	0,062	0,031	0,053	0,238	0,158	0,197
	15	2,801626	2,262344	0,000	0,000	0,000	0,103	0,072	0,091
	20	2,801626	2,735175	0,000	0,000	0,000	0,044	0,027	0,036
	25	2,801626	2,993898	0,000	0,000	0,000	0,000	0,000	0,000

Ubuntu.bmp 6 bpp oranında sıkıştırılmak istendiğinde Tahmin Algoritması-3 kayıpsız olarak 2,244 bpp oranında sıkıştırma yaparken JPEG2000 ise 2.136 bpp oranına düşürmüştür. Her iki yöntem sonucunda da resimde bozulma görülmemektedir. 1.2 bpp oranında sıkıştırma istendiğinde ise JPEG2000 1.220 bpp oranına düşürmüş ve 0,698 bpp oranına düşüren Tahmin Algoritması-3'ten daha iyi bozulma sonuçları vermiştir.

Tahmin Algoritması-3 düşük sıkıştırma oranlarında resmi JPEG2000'e göre daha fazla bozmuştur. Bunun nedeni Tablo 4.6'da da görülebileceği gibi her iki yöntemden de aynı oran istendiğinde, Tahmin Algoritması-3'ün JPEG2000'e göre daha fazla sıkıştırma yapmasıdır. Tahmin Algoritması-3 ve JPEG2000'in resmi sıkıştırma oranları arasındaki bu fark nedeniyle bozulma sonuçlarının bu şekilde olması beklenir bir sonuçtur.

Tablo 4.7. 2k11.bmp için JPEG2000 ve Tahmin Algoritması-3'ün sıkıştırma oranı (bpp) ve bozulma ölçümleri sonuçları

	İstenen Oran (%)	J2K Sıkışt. Oranı	TA3 Sıkışt. Oranı	J2K R	J2K G	J2K B	Alg R	Alg G	Alg B
SSIM	1	0,4964057	0,2342871	0,9997	0,9997	0,9997	0,9706	0,9706	0,9706
	2	0,2419239	0,2929408	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
	3	0,4910423	0,2929408	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
	4	0,4964057	0,2929408	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
PSNR	1	0,4964057	0,2342871	49,57	49,57	49,57	39,58	39,58	39,58
	2	0,2419239	0,2929408	74,58	74,58	74,58	∞	∞	∞
	3	0,4910423	0,2929408	101,35	101,35	101,35	∞	∞	∞
	4	0,4964057	0,2929408	101,35	101,35	101,35	∞	∞	∞
MSE	1	0,4964057	0,2342871	0,72	0,72	0,72	7,16	7,16	7,16
	2	0,2419239	0,2929408	0,00	0,00	0,00	0,00	0,00	0,00
	3	0,4910423	0,2929408	0,00	0,00	0,00	0,00	0,00	0,00
	4	0,4964057	0,2929408	0,00	0,00	0,00	0,00	0,00	0,00

2k11.bmp için 12 bpp sıkıştırma oranı istendiğinde her iki yöntem de 12 bpp'den daha düşük sıkıştırma oranlarına ulaşmıştır. Tahmin Algoritması-3 0,293 bpp oranına kayıpsız olarak sıkıştırma yapabilmişken JPEG2000 ise 0,494 bpp oranına sıkıştırma gerçekleştirmiş ve çok düşük de olsa bozulma göstermiştir. Resmin karmaşıklığının çok düşük olması sayesinde Tahmin Algoritması-3 LZMA'nın avantajından yararlanmıştır. 1.2 bpp oranında sıkıştırma istendiğinde ise JPEG2000, Tahmin Algoritması-3'e göre daha yakın sıkıştırma oranı vermiş ve bununla birlikte daha az bozulma sağlamıştır. Tablo 4.7'de görüldüğü gibi 0.24 bpp (%1) sıkıştırma istendiğinde Tahmin Algoritması-3 JPEG2000'e göre daha yakın sonuç vermiştir.

Tablo 4.8. Bridgetolimansk.bmp için JPEG2000 ve Tahmin Algoritması-3'ün sıkıştırma oranı ve bozulma ölçümleri sonuçları

	İstenen Oran (%)	J2K Sıkışt. Oranı	TA3 Sıkışt. Oranı	J2K R	J2K G	J2K B	Alg R	Alg G	Alg B
SSIM	1	0,2414052	0,244289	0,9710	0,9710	0,9710	0,9014	0,9014	0,9014
	5	1,2262134	1,2017387	0,9961	0,9961	0,9961	0,9969	0,9969	0,9969
	10	2,4463691	2,5033674	0,9986	0,9986	0,9986	0,9912	0,9912	0,9912
	15	3,6772669	3,7577709	0,9992	0,9992	0,9992	0,9988	0,9988	0,9988
	20	4,9013212	4,8997572	0,9998	0,9998	0,9998	0,9996	0,9996	0,9996
	25	6,1057454	5,8404835	0,9998	0,9998	0,9998	0,9999	0,9999	0,9999
PSNR	1	0,2414052	0,244289	33,25	35,57	34,74	27,87	29,27	29,97
	5	1,2262134	1,2017387	39,84	42,42	39,91	38,62	40,37	40,04
	10	2,4463691	2,5033674	43,00	45,77	42,75	37,51	39,33	39,48
	15	3,6772669	3,7577709	44,87	47,37	44,66	42,40	44,14	43,36
	20	4,9013212	4,8997572	46,45	50,07	46,61	46,12	47,68	46,78
	25	6,1057454	5,8404835	48,46	51,62	48,42	49,37	50,83	49,20
MSE	1	0,2414052	0,244289	30,75	18,03	21,81	106,31	76,97	65,54
	5	1,2262134	1,2017387	6,75	3,72	6,64	8,93	5,98	6,44
	10	2,4463691	2,5033674	3,26	1,72	3,45	11,55	7,59	7,33
	15	3,6772669	3,7577709	2,12	1,19	2,22	3,74	2,50	3,00
	20	4,9013212	4,8997572	1,47	0,64	1,42	1,59	1,11	1,36
	25	6,1057454	5,8404835	0,93	0,45	0,94	0,75	0,54	0,78

Bridgetolimansk için 18 bpp sıkıştırma oranı istendiğinde her iki algoritma da kayıpsız sıkıştırmayı seçmiş ve JPEG2000 9,454 bpp oranını yakalamışken Tahmin Algoritması-3 13,934 bpp oranını elde etmiştir. Dönüşüm işlemlerinden dolayı JPEG2000'de gözle görülmeyecek bir kayıp gerçekleşmiştir. 4.8 bpp sıkıştırma oranında Tahmin Algoritması-3 JPEG2000'e sıkıştırma oranı olarak ve bozulma miktarı olarak oldukça yakın sonuçlar vermiştir.

Algoritma kayıpsız sıkıştırma yapana kadar JPEG2000'e göre daha fazla bozulma göstermiştir. JPEG2000 daha düşük bpp değerlerinde kayıpsız olarak sıkıştırma yaptığı için, JPEG2000'in kayıpsız sıkıştırma yaptığı 9.454 bpp'den Tahmin algoritması-3'ün kayıpsız sıkıştırma yaptığı 13.934 bpp'ye kadar olan aralıkta algoritma JPEG2000'e göre daha kötü sonuçlar vermiştir.

Tablo 4.9. Programmer.bmp için JPEG2000 ve Tahmin Algoritması-3'ün sıkıştırma oranı (bpp) ve bozulma ölçümleri sonuçları

	İstenen Oran (%)	J2K Sıkışt. Oranı	TA3 Sıkışt. Oranı	J2K R	J2K G	J2K B	Alg R	Alg G	Alg B
SSIM	1	0,2495861	0,1028677	133,00	75,00	126,00	0,00	0,00	0,00
	3	0,7495213	0,1028677	27,00	13,00	23,00	0,00	0,00	0,00
	5	1,2170185	0,1028677	6,00	4,00	6,00	0,00	0,00	0,00
	7	1,4953197	0,1028677	1,00	1,00	1,00	0,00	0,00	0,00
	10	1,4953197	0,1028677	1,00	1,00	1,00	0,00	0,00	0,00
PSNR	1	0,2495861	0,1028677	33,20	38,24	33,21	∞	∞	∞
	3	0,7495213	0,1028677	47,63	52,13	47,77	∞	∞	∞
	5	1,2170185	0,1028677	59,28	63,81	59,25	∞	∞	∞
	7	1,4953197	0,1028677	87,13	87,26	83,45	∞	∞	∞
	10	1,4953197	0,1028677	87,13	87,26	83,45	∞	∞	∞
MSE	1	0,2495861	0,1028677	31,11	9,75	31,08	0,00	0,00	0,00
	3	0,7495213	0,1028677	1,12	0,40	1,09	0,00	0,00	0,00
	5	1,2170185	0,1028677	0,08	0,03	0,08	0,00	0,00	0,00
	7	1,4953197	0,1028677	0,00	0,00	0,00	0,00	0,00	0,00
	10	1,4953197	0,1028677	0,00	0,00	0,00	0,00	0,00	0,00

Jpeg2000 Programmer.bmp'yi 2.4 bpp oranında sıkıştırma istendiğinde 1,258 bpp'ye düşürürken Tahmin Algoritması-3 ise 0,077 bpp'ye düşürmüştür. Tahmin Algoritması-3 resmi kayıpsız sıkıştırdığı için bozulma söz konusu değildir. JPEG2000'de ise çok küçük miktarda da olsa bozulma mevcuttur. Tahmin Algoritması-3 bu resim için JPEG2000'den daha iyi bozulma oranlarında daha fazla sıkıştırma sağlamıştır. Bunun nedeni Tahmin Algoritması-3'ün tahmin aşamasında resmi LZMA ile sıkıştırmaya karar vermesi ve tüm resmi kayıpsız olarak LZMA ile sıkıştırmanın JPEG2000'e göre daha iyi sonuç vermesidir. 0.24 bpp sıkıştırma oranında ise Tahmin Algoritması-3 resmi hâlihazırda 0.077 bpp oranında sıkıştırdığı için herhangi bir kayıp ve değişiklik olmazken JPEG2000 için bozulma gözle görülür hale gelmiştir.

Tablo 4.9'dan de görüldüğü gibi, 0.24 bpp ile 24 bpp aralığında istenen farklı sıkıştırma değerleri için algoritma tamamen kayıpsız sıkıştırma yaptığı için sabit bir doğru çizmiştir. JPEG2000 ise 1.258 bpp'den sonra kayıpsız sıkıştırma yaparak iyileşme gösterse

de tam anlamıyla Tahmin Algoritması-3'le aynı sonuçları göstermemiş ve gözle görülemeyecek derecede de olsa bir bozulma göstermiştir.

Tablo 4.10. Lena.bmp için JPEG2000 ve Tahmin Algoritması-3'ün sıkıştırma oranı (bpp) ve bozulma ölçümleri sonuçları

	İstenen Oran (%)	J2K Sıkışt. Oranı	TA3 Sıkışt. Oranı	J2K R	J2K G	J2K B	Alg R	Alg G	Alg B
SSIM	1	0,2498913	0,3078707	0,9192	0,9192	0,9192	0,7551	0,7551	0,7551
	5	1,21415	1,2493954	0,9702	0,9702	0,9702	0,9034	0,9034	0,9034
	10	2,4665665	2,5137129	0,9854	0,9854	0,9854	0,9721	0,9721	0,9721
	25	6,1142652	5,892143	0,9979	0,9979	0,9979	0,9928	0,9928	0,9928
	50	10,905043	12,055996	1,0000	1,0000	1,0000	0,9994	0,9994	0,9994
	75	10,905043	16,874208	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
PSNR	1	0,2498913	0,3078707	28,40	31,23	29,28	22,92	25,05	24,06
	5	1,21415	1,2493954	33,74	36,08	33,87	26,71	28,74	27,92
	10	2,4665665	2,5137129	37,25	39,64	37,49	34,05	36,00	34,65
	25	6,1142652	5,892143	44,94	47,36	45,03	40,15	41,66	40,63
	50	10,905043	12,055996	80,14	85,60	85,60	48,93	50,77	49,26
	75	10,905043	16,874208	80,14	85,60	85,60	∞	∞	∞
MSE	1	0,2498913	0,3078707	93,94	48,98	76,83	331,61	203,33	255,07
	5	1,21415	1,2493954	27,46	16,03	26,65	138,78	86,94	104,91
	10	2,4665665	2,5137129	12,25	7,07	11,58	25,59	16,34	22,30
	25	6,1142652	5,892143	2,08	1,20	2,04	6,29	4,43	5,63
	50	10,905043	12,055996	0,00	0,00	0,00	0,83	0,54	0,77
	75	10,905043	16,874208	0,00	0,00	0,00	0,00	0,00	0,00

Lena.bmp için 18 bpp sıkıştırma oranı istendiğinde JPEG2000 10,277 bpp oranında sıkıştırma yaparken Tahmin Algoritması-3 11,657 bpp oranını yakalamıştır. Tahmin Algoritması-3'te herhangi bir bozulma oluşmazken JPEG2000'de ise R kanalında çok küçük de olsa bozulma mevcuttur. 2.4 bpp sıkıştırma oranında ise JPEG2000 2,136 bpp oranında sıkıştırma yaparken Tahmin Algoritması-3 2,244 bpp ile istenen orana daha yakın bir sıkıştırma yapmıştır. Tahmin Algoritması-3 kayıpsız sıkıştırma yapana kadar JPEG2000'e göre daha fazla bozulma göstermiştir.

BÖLÜM 5

SONUÇ

Kanalları farklı şekilde kodlama yöntemlerinin her biri farklı sıkıştırma algoritmaları için avantaj sağlamaktadır. Bu yöntemler yaptıkları işlem bakımından birbirlerinden farklı zaman karmaşıklığına sahiptir. İlk yöntem sadece renk kanallarının sıralanması işlemiyken ikinci yöntemde bunun üzerine resmi belirli boyuttaki parçalara bölme işlemi eklenmektedir. Üçüncü yöntemle beraber bu parçalama işleminin resmin sadece sol üst kısmı için dinamik olması sağlanmıştır. Bu da resmin bir kısmının fark bulunana kadar taranması anlamına gelmektedir. Son yöntemde ise dinamik olarak parçalanmak üzere tüm resim taranır. Bu nedenle en son yöntemdeki işlem karmaşıklığı çok daha fazladır. Yöntemlerin karmaşıklığıyla sonuçların doğrusal bir şekilde iyileşmesi gibi bir durum söz konusu değildir. Aksine bazı durumlarda en basit karmaşıklıktaki kanalların yeniden sıralanma işlemi, diğer algoritmalara göre çok daha iyi sonuç vermiştir. Resmin karmaşıklık düzeyine göre bu algoritmalar belirli durumlarda resmi sıkıştırma yöntemlerinin daha iyi sıkıştırabileceği hale getirebilmektedirler.

Tahmin Algoritması-3'ün başarısı, eğitim verisinin arttırılmasıyla artmaktadır. Eğitim verisinin miktarı ve verideki çeşitliliğin arttırılması, algoritmanın karşılaştığı farklı durumlarda daha doğru sonuçlar vermesini sağlayacaktır. Algoritmanın performansı, kullandığı LZMA ve JPEG XR yöntemlerine de doğrusal bir şekilde bağlıdır. Eldeki eğitim verisi ile Algoritma, istenen boyuta düşürme işlemini çoğunlukla yakın sonuçlar vererek gerçekleştirmektedir. Eğitim verisinin çeşitliliğinin arttırılmasıyla, bu sonuçlar en iyiye daha fazla duyarlılık ile yakınsayacaktır.

Tahmin Algoritması-3, JPEG2000'de olduğu gibi dosyayı istediği boyutun altına çekebileceğine karar verirse, LZMA algoritmasıyla kayıpsız sıkıştırma yapmakta ve daha küçük boyutlu bir dosya elde etmektedir. Algoritmanın JPEG2000'e göre avantajlı yönü ise; resmi bölüp bazı parçalarını kayıplı sıkıştırırken bazılarını ise kayıpsız

sıkıştırabilmesidir. JPEG2000 ise resmi ya tamamen kayıpsız ya da tamamen kayıplı olarak sıkıştırır.

Bazı durumlarda Algoritma JPEG2000'den daha az sıkıştırma yapmasına rağmen resimde JPEG2000'e göre daha fazla bozulma görülmektedir. Buna rağmen Algoritma, kullandığı LZMA yöntemi sayesinde, grafik veya şekil gibi karmaşıklık miktarı düşük resimlerde JPEG2000'e göre daha iyi sonuç vermektedir. Bunun nedeni karmaşıklık miktarının düşmesiyle sıkıştırılacak verinin LZMA algoritmasıyla daha iyi sıkıştırılabilmesi ve bunun kayıpsız olarak gerçekleştirilebilmesidir.

Tahmin Algoritması-3'ün önemli bir avantajı da, JPEG XR'da hali hazırda bulunmayan sıkıştırma oranı belirleme seçeneğine sahip olmasıdır. JPEG XR veya JPEG yöntemlerinde verilen kalite faktörüne göre sıkıştırılma yapılmakta ve belirli bir dosya boyutu verilip, o boyuta kadar sıkıştırma yapılması istenememekte iken Tahmin Algoritması-3 buna olanak sağlamaktadır.

EKLER

EK-A Karar Algoritmaları

Weka

Weka (Waikato Environment for Knowledge Analysis) Waikato Üniversitesi tarafından geliştirilmiş bir makine öğrenmesi yazılımıdır. 3. sürümü Java ile geliştirilmiştir ve grafik ara yüze sahiptir. Nominal, nümerik veya metin verilerle çalışılabilir. Weka'ya verilen arff dosyası içerisinde başlık ve veriler olmak üzere iki ana bölüm bulunur. Başlık kısmında özellikler ve bu özelliklerin tipleri bulunurken, veri kısmında bu özelliklerin taşıdığı değerler ve sonuç değerleri saklanır. Aşağıdaki şekilde verilen eğitim verisinin veri kısmındaki ilk satırda Jpeg2000 öncesi sayısal değerler sırasıyla dosyaBoyutu, entropi, entropiAvg vb. özelliklerini ifade ederken son eleman olan Jpeg2000, bu sayısal değerleri veren resimde en iyi sıkıştırma yapan algoritmanın JPEG2000 olduğunu gösterir.

```
@relation resim

@attribute dosyaBoyutu real
@attribute entropi real
@attribute entropiAvg real
@attribute entropiPer real
@attribute kareSayisi real
@attribute blobSayisi real
@attribute fourierBeyazSayisi real
@attribute moravecCornerSayisi real
@attribute quadriliteralSayisi real
@attribute susanCornerSayisi real
@attribute enIyiAlgoritma {Deflate, Deflate64, Jpeg2000, PPMd, LZMA, LZMA2, Bzip2, Jpeg_XR, Png}

@data
196716,7.204725293,7.053028538,7.147344435,56,1,139,492,1,300,Jpeg2000
12345.375,4.222022201,4.250522958,4.207047545,0,1,0,6,1,0,Jpeg2000
12345.375,6.313119667,6.179047649,6.282494254,0,1,7,34,1,15,Jpeg2000
12345.375,6.092810391,5.968757417,6.066807445,0,1,9,32,1,20,Jpeg2000
12345.375,4.379680283,4.37786064,4.379893913,156,1,15,58,1,56,LZMA
12345.375,2.738943279,2.738943279,2.738943279,24,1,3,7,1,3,LZMA
```

Şekil. Eğitim verisi örneği

J48

J48 C4.5 karar ağacı oluşturma algoritmasının weka içerisinde kullanılan java uyarlamasıdır.

Karar ağaçları her düğümde bir özelliği kontrol edebileceği gibi, birden fazla özelliği de taşıyabilir. Her düğümde bir özellikle ilgilenen karar ağaçları "Univariate" karar ağacı olarak adlandırılırken, birden fazla özelliği bir düğümde taşıyan ağaçlar ise "Multivariate" karar ağacı olarak adlandırılır [Korting]

C4.5 algoritması Ross Quinlan tarafından geliştirilmiş bir "Univariate" karar ağacı oluşturma algoritmasıdır [Quinlan, 1993]. Yine Ross Quinlan tarafından geliştirilen en küçük karar ağaçlarını oluşturmayı amaçlayan ID3 algoritmasına dayanır [Quinlan, 1986].

Algoritma bir eğitim verisi üzerinden karar ağacı oluşturur. Eğitim verisi olarak sayısal değerler verilebildiği gibi, nominal değerler üzerinden karar ağacı oluşturulması da mümkündür. Karar ağacına verilen test verisinin sonucu, ağaçta çocuk düğümlere doğru ilerleyerek bulunur.

M5P

M5P algoritması Ross Quinlan tarafından geliştirilmiş M5 algoritmasının weka üzerindeki Java uyarlamasıdır [Quinlan, 1992].

M5 bilinen karar ağacı yapısını düğümlerinde doğrusal regresyon fonksiyonları saklayabilecek şekilde değiştirmiştir. Ağacın oluşturulması için bir tümevarım karar ağacı algoritması kullanılır. Ardından bir bölme kıstası kullanılarak ağaç düğümleri çocuklara bölünür. Eğer düğümlerdeki sınıf değerleri düşük farklılık gösteriyorsa veya düğüm sayısı azsa bölme işlemi durur. Daha sonra ağacın her yaprağı kontrol edilerek budama işlemi gerçekleştirilir. Bu esnada budanan düğümler regresyon düzeyine dönüştürülür. Son aşamada da alt ağaçlar arasındaki süreksizlikleri (discontinuity) engellemek için bir yumuşatma (smoothing) prosedürü uygulanır.

EK–B BOZULMA ÖLÇÜMLERİ

MSE

Hataların kareleri toplamının ortalaması alınarak bulunur. Sıkıştırılmış ve orijinal resim verisi üzerinde birbirine karşılık gelen pikseller incelenir ve farklarının kareleri toplanır. Toplamın piksel sayısına oranı MSE'yi verir.

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - y_n)^2$$

RMSE

MSE'nin kareköküdür. Resim verisi üzerinde MSE hesaplandıktan sonra değerin kökü alınarak bulunur.

$$\sigma = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(x_n - y_n)^2}$$

PSNR

Hatanın büyüklüğünün orijinal piksel değerinin en büyüğü (tepe noktası) ile olan orantısıdır. Öncelikle MSE hesaplanır. Daha sonra en büyük değere sahip piksel tespit edilir. Bu değerin karesinin MSE'ye oranı PSNR'ı verir.

$$PSNR(dB) = 10\log_{10}\frac{x_{tepe}^2}{\sigma_d^2}$$

MAE

Hataların mutlak değerinin ortalamasıdır. Resim verisinde sıkıştırılmış ve orijinal resim pikselleri arasındaki farklar mutlak değer olarak toplanır ve piksel sayısına oranlanarak ortalaması bulunur.

$$MAE = \frac{1}{N}\sum_{n=1}^{N}|x_n - y_n|$$

PAE

Hataların mutlak değerinin en büyüğüdür. Pikseller arasındaki farkın mutlak değerce en büyüğü tespit edilir.

$$PAE = max_n|x_n - y_n|$$

SSIM

İki resim arasındaki benzerliği ölçen bir yöntemdir [Wang vd., 2004]. MSE ve PSNR gibi yöntemlerin hesapladıkları bozulmanın insanların karar verdiği sübjektif bozulmayla arasındaki çelişkiyi kaldırmayı amaçlamıştır. Görüntünün parçalanmasıyla her parça üzerinde ayrı ayrı hesaplanır.

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

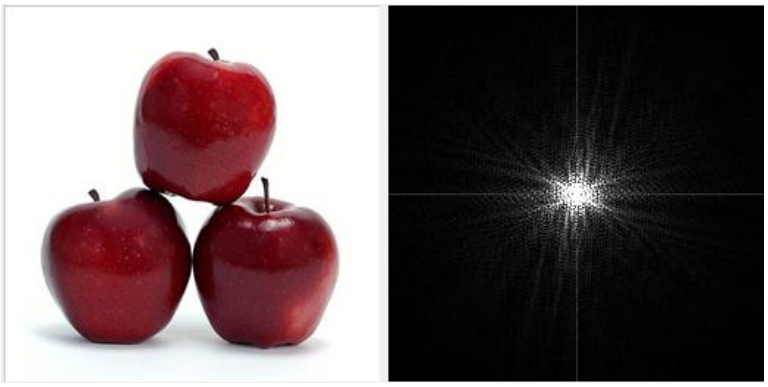
Formüldeki μ değerleri ortalamaları, σ^2 değerleri varyansı, σ_{xy} kovaryansı ifade etmektedir. c_1 ve c_2 değerleri $(k_1L)^2$ ve $(k_2L)^2$'yi ifade eder ve zayıf payda ile bölmeyi dengelemek için kullanılır. L değeri ise piksel değerinin alabileceği aralığı ifade eder.

EK–C Karmaşıklık Hesabı İçin Kullanılan Yöntemler

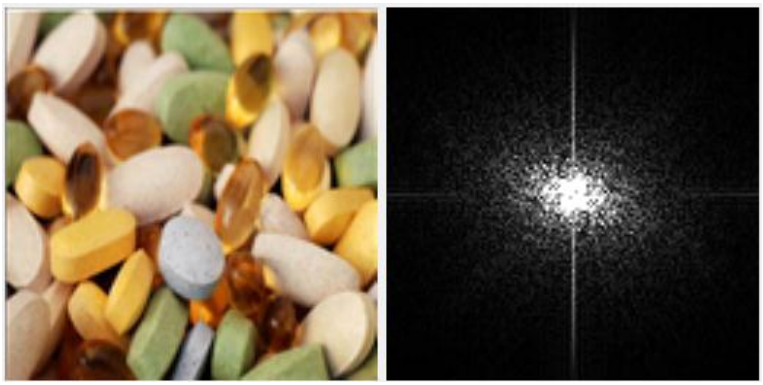
Blob sayısı fonksiyonu, resimdeki nesnelerin tespitini yaptıktan sonra sayısını geriye döndürür. Blob, renk ve parlaklık gibi özelliklerle resimden ayırt edilebilecek nesneleri ifade eder. Blob bulma fonksiyonu köşelerini bulduğu çokgenleri nesne olarak kabul eder. Algoritma piksel değeri olarak girilen sınırın üstündeki değerleri objenin, eşit ve altındaki değerleri ise arka planın pikselleri olarak kabul eder. 8 bpp gri tonlu ve 24/32 bpp RGB resimleri desteklemektedir. RGB kanallarının üçünün de hesaplamaya dâhil edildiği durumda ise herhangi bir kanal verisinden yüksek bir veriyle karşılaşılırsa bu piksel objenin olarak kabul edilir. Varsayılan sınır değeri RGB için 0, 0, 0 yani siyahtır.

Şekil. Blob Sayısı Algoritmasının Tespit Ettiği Dikdörtgen

Fourier beyaz sayısı hesabı için öncelikle AForge.NET içerisindeki complexImage fonksiyonu çalıştırılır. Fonksiyon resmi fourier dönüşümüne uygun hale getirmek için resmi karmaşık sayılar ile ifade eder. Ayrıca fonksiyonun icrası sonucunda bir siyah-beyaz çıktı elde edilir. Karmaşıklık hesabı için elde edilen bu siyah beyaz resmin beyaz piksel sayısı tespit edilerek kaydedilir. Resmin karmaşıklığına göre beyaz piksel sayısı değişiklik göstermektedir.



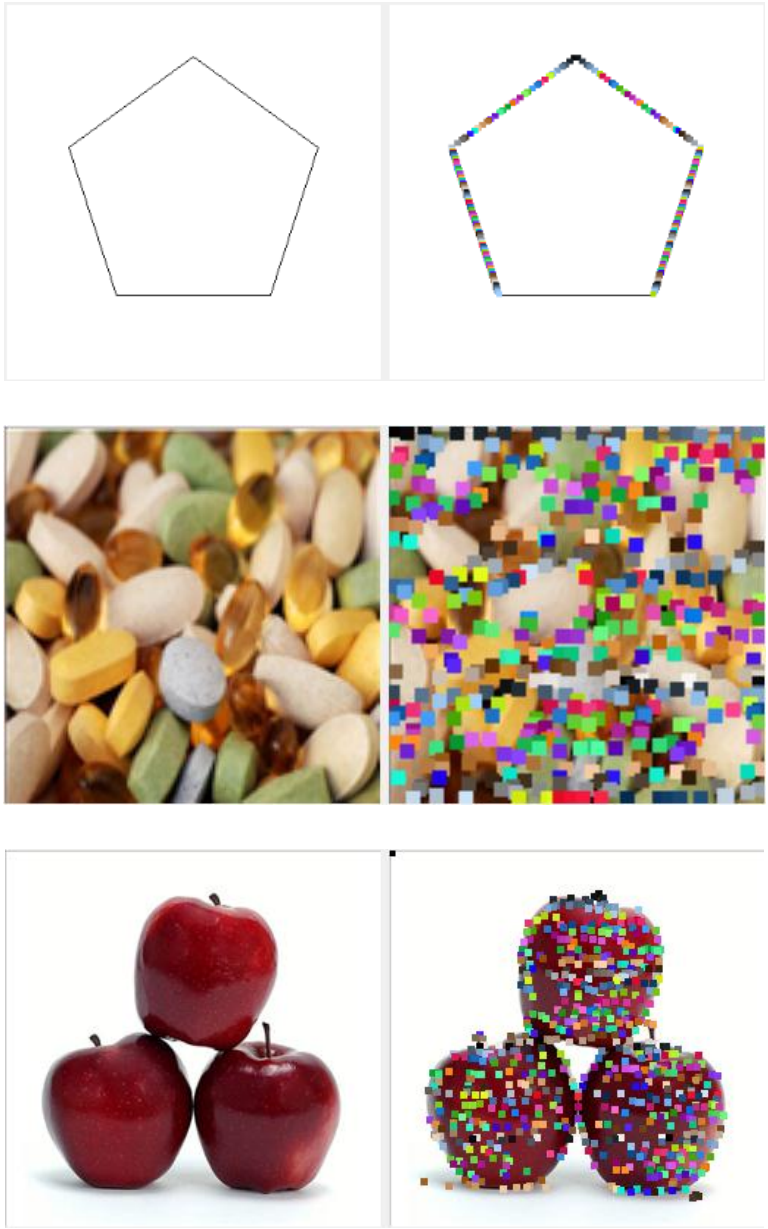
Şekil. Apple.bmp ve fourier dönüşümü sonrası elde edilen görüntü



Şekil. vitamins.bmp ve fourier dönüşümü sonrası elde edilen görüntü

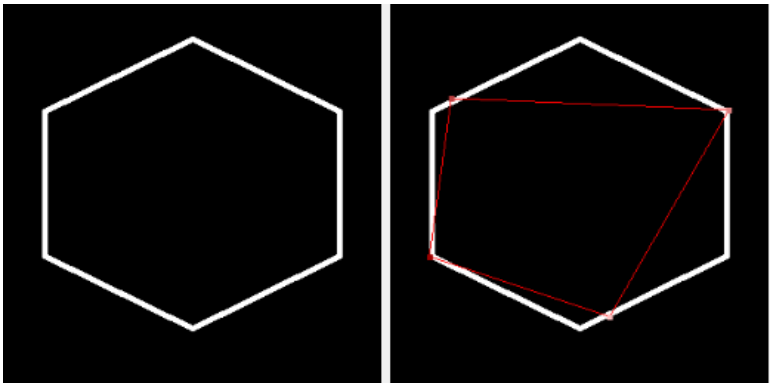
AForge.NET kütüphanesindeki MoravecCornersDetector sınıfı, Moravec Köşe Tespit etme algoritmasının uyarlamasıdır. Fonksiyon tespit ettiği köşelerin sayısını döndürür. Bulunan köşe sayısıyla nesne sayısı doğrusallık gösterir. Düz resimlerde bu sayı minimuma inerken, nesnelerin tespit edildiği karmaşık resimlerde köşe sayısı artacaktır.

Moravec köşe tanıma algoritması, ilk köşe tanıma algoritmalarından olup köşeyi, en düşük benzerliğe sahip nokta olarak tanımlar. Algoritma her pikseli araştırır. Resmi büyük boyutta belirli mantıksal parçalara ayırdıktan sonra incelediği pikseli orta nokta olarak kabul eden parçanın etrafındaki parçalarla olan benzerliğini araştırır ve bu pikselin köşe olup olmadığına karar verir [Moravec, 1980]. Benzerlik, iki parçanın piksel farklarının karelerinin toplamı olarak ifade edilir. Düşük değerler daha çok benzerlik olduğunu belirtir.



Şekil. Moravec corner finder metodunun bulduğu köşeler

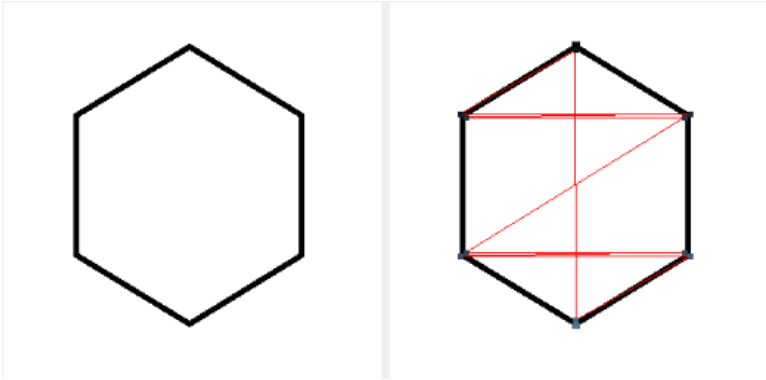
Quadrilateral, Euclid geometrisinde dört kenarı ve dört köşesi olan bir çokgendir. Quadrilateral sayısının tespit edilmesi için QuadrilateralFinder fonksiyonu kullanılır. Fonksiyon resimde quadrilateral veya üçgenlerin köşelerini tespit eder. Fonksiyon siyah pikselleri arkaplan, diğer pikselleri ise nesne olarak sayar. 8 bpp grayscale ve 24/32 bpp RGB resimlerde kullanılabilir.



Şekil. Quadrilateral finder fonksiyonunun bulduğu çokgen

Susan, "Smallest Univalue Segment Assimilating Nucleus" isminin kısaltmasıdır [Smith, 1995]. Algoritma, tanımlama işlemi için test edilecek pikselin üstüne dairesel bir maske uygular. Test edilecek piksel "nucleus" olarak adlandırılır. Daha sonra maskedeki her piksel "nucleus" olarak adlandırılan test pikseli ile karşılaştırılır. Belirli bir benzerlik kıstastı kullanılarak benzerliği sağlayan parçanın büyüklüğü tespit edilir. Eğer parça yeteri kadar büyükse parçanın kenarları bulunabilir ve parça SUSAN operatörü olarak adlandırılır.

Köşe tanıma için ise iki metod daha uygulanır. Öncelikle, bulunan parçanın ağırlık merkezi tespit edilir. Uygun bir köşe "nucleus"tan uzakta bir ağırlık merkezine sahip olacaktır. İkinci adıma göre ağırlık merkezi boyunca nucleus'tan maskenin kenarlarına kadar olan noktalar SUSAN operatörü içerisindedir.



Şekil. Susan Corners Detector algoritmasının yakaladığı köşeler

KAYNAKLAR

1. Burrows M., Wheeler D., 1994, "A block sorting lossless data compression algorithm", Technical Report 124, Digital Equipment Corporation.

2. Christopoulos C., Skodras A., Ebrahimi T., 2000, "The JPEG2000 still image coding system: An Overview", IEEE Transactions on Consumer Electronics, 46(4), 1103-1127.

3. Cleary J.G., Witten I.H., 1984, "Data compression using adaptive coding and partial string matching, IEEE Transactions on Communications", Vol. 32 (4), pp. 396–402.

4. CompuServe, 1990, "Graphics Interchange Format(sm), Version 89a", CompuServe Incorporated, Columbus, Ohio.

5. Deutsch, P., 1996, "DEFLATE Compressed Data Format Specification, version 1.3", Network Working Group, Request for Comments 1951.

6. Huffman D.A., 1952, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., pp 1098–1102.

7. ISO/IEC 10918-1, 1994, CCITT Rec. T.81, 1992, "Digital Compression And Coding of Continuous-Tone Still Images – Requirements And Guidelines (JPEG)".

8. ISO/IEC 14495-1:1994 and -2:2003. 1994, "Information technology -- Lossless and near-lossless compression of continuous-tone still images: Baseline and Extensions".

9. ISO/IEC 15444-1:2004, 2000, "JPEG 2000 image coding system: Core coding system".

10. ISO/IEC 15948, 2004, Information technology -- Computer graphics and image processing -- Portable Network Graphics (PNG): Functional specification

11. ITU-T Rec. T.832, 2009, ISO/IEC 29199-2, 2010, "JPEG XR image coding system: Image coding specification".

12. Jensen A., Cour-Harbo A., 2001, "Ripples in Mathematics: The Discrete Wavelet Transform", Springer-Verlag, Berlin, Germany.

13. Korting T. S., "C4.5 algorithm and Multivariate Decision Trees, Image Processing Division, National Institute for Space Research", INPE, SP, Brazil.

14. Mallat S., 1989, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation", IEEE Pattern Analysis and Machine Intelligence, 11(7), 674-693.

15. Mesut A., 2006, "Veri Sıkıştırmada Yeni Yöntemler", Doktora Tezi, Trakya Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı.

16. Moravec H., 1980, “Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover”, Tech Report CMU-RI-TR-3 Carnegie-Mellon University, Robotics Institute.

17. Quinlan J. R., 1993, “C4.5: Programs for Machine Learning”, Morgan Kaufmann Publishers.

18. Quinlan J. R., 1986, “Induction of Decision Trees. Mach. Learn” 1, 1 (Mar. 1986), 81-106.

19. Quinlan J.R., 1992, “Learning with Continuous Classes”, 5th Australian Joint Conference on Artificial Intelligence, Singapore, 343-348.

20. Randers-Pehrson G., 1999, “PNG (Portable Network Graphics) Specification, “Version 1.2”, PNG Development Group.

21. Rao K. R., Yip P., 1990, “Discrete Cosine Transform: Algorithms, Advantages, Applications”, Academic Press, San Diego, CA, USA.

22. Ritter T., 1996, “Walsh-Hadamard Transforms: A Literature Survey”.

23. Santa-Cruz D., Ebrahimi T., Askelof J., Larsson M., Christopoulos C., “JPEG 2000 still image coding versus other standards”, Proc. of the SPIE’s 45th annual meeting, Applications of Digital Image Processing XXIII, Vol. 4115, 2000.

24. Seward J., http://bzip.org/.

25. Smith S.M., 1995, “SUSAN - a new approach to low level image processing”, Internal Technical Report TR95SMS1, Defense Research Agency, Chobham Lane, Chertsey, Surrey, UK.

26. Wang Z., Bovik A. C., Sheikh H. R., Simoncelli E. P., 2004, “Image quality assessment: From error visibility to structural similarity”, IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612.

27. Welch T.A., 1984, “A Technique for High-Performance Data Compression”, IEEE Computer, June 1984, p. 8–19.

28. Ziv J., Lempel A., 1977, “A Universal Algorithm for Sequential Data Compression”, IEEE Transactions on Information Theory, 23(3), pp. 337–343.

ÖZGEÇMİŞ

Emir Öztürk 1988 yılında Zonguldak'ta doğdu. 2006 yılında Trakya Üniversitesi Bilgisayar Mühendisliği Bölümü'nde lisans eğitimine başladı ve 2010 yılında mezun oldu. Aynı yılda Trakya Üniversitesi Bilgisayar Mühendisliği Bölümü'nde yüksek lisans eğitimine ve araştırma görevlisi olarak göreve başladı. Halen Trakya Üniversitesi'nde araştırma görevliliğini sürdürmektedir.