

Renk Kanallarını Farklı Şekilde Kodlayarak Sıkıştırma Oranını Arttırma

Emir Öztürk¹, Altan Mesut²

¹ Trakya Üniversitesi, Bilgisayar Mühendisliği Bölümü, Edirne

² Trakya Üniversitesi, Bilgisayar Mühendisliği Bölümü, Edirne

emirozturk@trakya.edu.tr, altanmesut@trakya.edu.tr.

Özet: İki boyutlu durağan görüntülerin sayısal ortamda saklanırken kayıplı veya kayıpsız sıkıştırma yöntemlerinin kullanılmasıyla daha az yer kaplamalarını sağlamak mümkündür. Bunun için fotoğraf türündeki görüntülerde etkili olan JPEG, JPEG2000 ve JPEG XR veya karmaşıklığın (entropinin) az olduğu görüntülerde iyi sonuç veren GIF ve PNG gibi yöntemler kullanılabilir. 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 çalışmada, karmaşıklığı düşük olan görüntülerde renk kanalları üzerinde yapılan işlemler ile kayıpsız sıkıştırma algoritmalarının etkinliği artırılmıştır.

Anahtar Sözcükler: Veri Sıkıştırma, Görüntü Bölümlendirme, LZMA, PPMd, Bzip2, Deflate

Increasing Compression Ratio With Encoding Color Channels In Different Ways

Abstract: It is possible to shrink (reduce the size of) two dimensional still images with lossless or lossy compression methods. Therefore, standards like JPEG, JPEG2000 and JPEG XR which are used to compress photos or GIF and PNG, used to compress low complexity (low entropy rate) images could be used. Compressing low complexity images with lossy compression methods are both staying behind lossless compression methods and the corruption of image could be detected easily by human eye. In this study, the efficiency of lossless compression algorithms is increased with some operations on color channels of low complexity images.

Keywords: Data Compression, Image Segmentation, LZMA, PPMd, Bzip2, Deflate

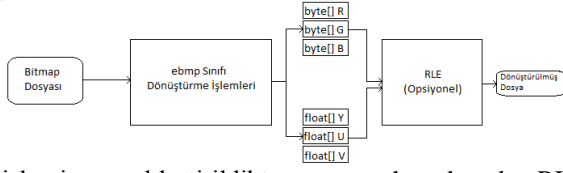
1. Giriş

JPEG [1], JPEG2000 [2,3] ve JPEG XR [4] gibi kayıplı görüntü sıkıştırma standartları, renk uzayı dönüşümü, uzamsal etki alanından frekans etki alanına dönüşüm [5,6], niceleme işlemi ve entropi kodlaması gibi aşamalarının her birinde farklı yaklaşımları temel aldıkları için birbirlerinden farklı hızlarda ve farklı kalite oranlarında sıkıştırma yapmaktadırlar. Fakat bu yöntemler genellikle fotoğraf türündeki karmaşık görüntülerin sıkıştırılması amacıyla kullanılırlar. Karmaşıklığı düşük olan görüntülerde kayıplı bir sıkıştırma yöntemi kullanıldığında sıkıştırma oranı kayıpsız sıkıştırma yöntemlerinin gerisinde kalmakta ve görüntüdeki kayıp gözle görülebilir olmaktadır. Bu yüzden karmaşıklığın daha az olduğu (entropi oranı daha düşük olan) görüntülerde GIF [7] ve PNG [8,9] gibi kayıpsız görüntü sıkıştırma yöntemleri kullanılmaktadır.

Resmin renk kanalları üzerinde yapılan işlemler ile sıkıştırma performansının artırılması mümkündür. Bu çalışmada, 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.

2. EBMP

EBMP uygulamasında 24 bpp bitmap resim alındıktan sonra 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. Bir diğer yöntem ise RGB uzayından YUV uzayında dönüşüm yaptıktan sonra Y, U ve V kanallarını ayrı float dizilerinde saklamak ve ardından bu kanallara RLE uygulamaktır. RGB ve YUV için kanallara ayırma



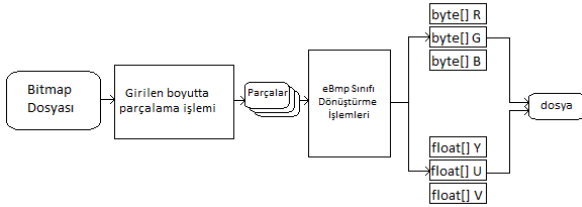
işlemi gerçekleştirildikten sonra her kanala RLE uygulanır.

Şekil 1.Ebmp sıkıştırma aşamaları

3. Statik Sabit Büyüklükte Bölümlendirme

Statik Sabit Büyüklükte Bölümlendirme (SSBB) yönteminde alınan 24 bpp resim verilen parça boyutu kadar parçalara bölünür. Örneğin, parça boyutu 64 olarak verilirse resim 64x64 piksellik parçalara bölünür. 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 2).

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



Şekil 2. SSBB Sıkıştırma Aşamaları

4. Dinamik Sabit Büyüklükte Bölümlendirme

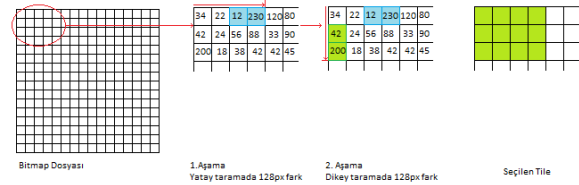
Dinamik Sabit Büyüklükte Bölümlendirme (DSBB) yönteminde resmin bölüneceği parçaların boyutu verilen resme göre dinamik olarak belirlenir. Alınan 0-255 arası bir fark değerine göre resmin herhangi bir kanalında yan yana olan pikseller taranır ve bulunana kadar yatay olarak ilerlenir. Verilen fark yan 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. Parça Boyutunun Belirlenme Aşamaları

5. 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

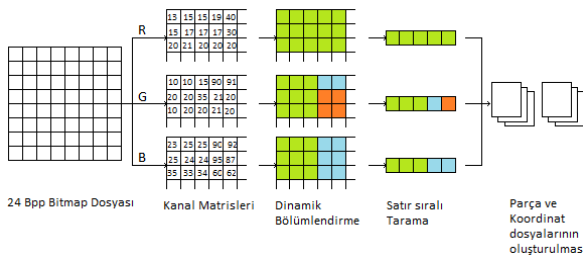
koordinat dosyası oluşturulur. Bu dosyanın içerisine bu parçanın koordinatları yazılır. Uygulama elde ettiği parçaları ve koordinatları sırasıyla dosyaya yazdığı ve okuma safhasında da aynı sırayı takip ettiği için herhangi bir okuma problemi ile karşılaşmamaktadır. Dosya içerisindeki koordinat verilerinin bir örneği Tablo 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 işaretli olan konumdan itibaren fark koşulunu tekrar arar. Fark koşulu sağlanmadan resmin sınırlarına gelirse, 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.



Şekil 4. 64 Fark koşulu için dosyanın bölümlendirme aşamaları

6. Sonuçlar

Karşılaştırma için kullanılan resim dosyaları (Şekil 4’te verilmiştir).

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.



Şekil 4. Karşılaştırma için kullanılan Ubuntu.bmp 1280x1024 (a), 2k11.bmp 1024x1024(b), Bridgetolimansk.bmp 1024x1024 (c), programmer.bmp 512x512 (d), lena.bmp 512x512 (e)

Tablo 1. Koordinat dosyasının içeriği

X1	Y1	X2	Y2
0	0	30	70
40	0	70	50
...

Tablo 2. 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	2,49 2	2,49 8	2,49 8	3,23 4	3,26 6	3,14	3,08 2	3,74 1	3,23 4
EBMP+RLE	2,74 8	2,74 9	2,74 9	2,55 9	2,52 8	2,82 3	2,81 7	4,39 5	2,55 9
EBMP	2,39 6	2,43 1	2,43 2	2,32 2	2,37 8	2,73 3	2,73 4	3,29 7	2,32 2
SSBB 64px	2,35 1	2,37 7	2,37 7	2,67 8	2,74 5	3,02 1	3,02 1	3,32 4	2,67 8
SSBB 128px	2,31 5	2,34 5	2,34 5	2,52 4	2,59 6	2,88 1	2,88 1	3,21 8	2,52 4
SSBB 256px	2,33 4	2,36 5	2,36 8	2,41 7	2,48 1	2,79 4	2,79 5	3,22	2,41 7
DSBB 64fark-R	2,39 5	2,43 4	2,43 4	2,32 1	2,37 7	2,69 5	2,69 1	3,30 3	2,32 1
DSBB 128fark-R	2,39 5	2,43 3	2,43 3	2,32 2	2,37 8	2,68 9	2,68 8	3,30 5	2,32 2
DDBB 128fark	2,39 8	2,43 3	2,43 3	2,35 1	2,41 1	2,77 7	2,77 5	3,30 1	2,35 1
DDBB 200fark	2,40 4	2,44 2	2,44 1	2,33 2	2,38 8	2,74 8	2,74 8	3,30 9	2,33 1

Tablo 2'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 3. 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	13,8 04	13,9 35	13,9 36	12,5 89	12,5 91	12,4 75	12,4 83	17,1 43	12,5 89
EBMP+RLE	15,5 69	15,5 68	15,5 71	12,8 19	13,0 05	13,6 76	13,6 73	22,0 49	12,8 19

EBMP	13,5 95	14,0 41	14,0 44	12,6 14	12,6 21	13,5 92	13,5 68	16,7 27	12,6 14
SSBB 64px	13,0 51	13,3 4	13,3 43	12,8 58	12,8 66	13,9 34	13,8 51	15,9 91	12,8 58
SSBB 128px	13,0 93	13,4 04	13,4 05	12,7 05	12,7 13	13,7 64	13,6 72	15,6 65	12,7 05
SSBB 256px	13,2 86	13,6 52	13,6 55	12,6 54	12,6 6	13,6 58	13,6 1	15,8 96	12,6 54
DSBB 64fark-R	13,9 12	14,3 65	14,3 66	12,8 53	12,8 58	13,9 12	13,9	17,3 54	12,8 53
DSBB 128fark-R	13,5 93	14,0 31	14,0 33	12,5 87	12,5 93	13,5 3	13,5 27	16,7 61	12,5 87
DDBB 128fark	13,4 76	13,8 81	13,8 84	12,6 68	12,6 76	13,6 63	13,6 3	16,4 75	12,6 68
DDBB 200fark	13,5 96	14,0 43	14,0 44	12,6 16	12,6 23	13,5 95	13,5 74	16,7 3	12,6 16

Tablo 3'te 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. 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	0,29 5	0,29 3	0,29 3	0,32 1	0,32 6	0,30 7	0,30 7	0,50 7	0,32
EBMP+RLE	0,28 3	0,28 3	0,28 3	0,40 9	0,29 3	0,39 2	0,39 1	1,13 2	0,40 9
EBMP	0,25 6	0,25 6	0,25 6	0,32 4	0,30 4	0,38 5	0,37 3	0,87 3	0,32 4
SSBB 64px	0,30 1	0,30 1	0,3	0,37 7	0,34 4	0,44 2	0,43 3	0,37 1	0,37 7
SSBB 128px	0,27 2	0,27 2	0,27 2	0,34 4	0,31 8	0,40 7	0,39 6	0,32 8	0,34 4
SSBB 256px	0,26 7	0,26 6	0,26 6	0,34 4	0,32 1	0,39 6	0,38 5	0,93 6	0,34 4
DSBB 64fark-R	0,25 8	0,25 8	0,25 8	0,32 7	0,30 7	0,38 8	0,37 7	0,87 6	0,32 7
DSBB 128fark-R	0,25 9	0,25 8	0,25 8	0,32 7	0,30 7	0,38 8	0,37 7	0,87 6	0,32 7
DDBB 128fark	0,34 6	0,34 6	0,34 7	0,45 9	0,43 3	0,52 9	0,51 5	1,04 4	0,45 9
DDBB 200fark	0,28 8	0,28 9	0,28 9	0,36 7	0,34 1	0,42 1	0,40 9	0,95	0,36 7

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).

Tablo 5'te görüldüğü gibi Programmer.bmp için tüm bölümlendirme algoritmaları, orijinal dosyanın sıkıştırma yöntemleriyle sıkıştırılmasından daha kötü sonuçlar vermiştir. Kendi aralarında ise en iyi sıkıştırma sağlayan algoritma PPMd (7z) için EBMP+RLE olmuştur.

Tablo 5. 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	0,10 3	0,10 3	0,10 3	0,08 5	0,08	0,07 6	0,07 6	0,10 6	0,08 5
EBMP+RLE	0,22 7	0,22 7	0,22 7	0,19 2	0,18 6	0,19 4	0,19 2	0,27 7	0,19 2
EBMP	0,21	0,21	0,21	0,18 7	0,19	0,19 2	0,19	0,23 5	0,18 7
SSBB 64px	0,26	0,25 9	0,25 9	0,25 6	0,26 2	0,27 5	0,27 3	0,34 1	0,25 6
SSBB 128px	0,24 5	0,24 5	0,24 5	0,21 2	0,21 6	0,22 7	0,22 4	0,29 1	0,21 2
SSBB 256px	0,23 9	0,24	0,24	0,20 5	0,20 9	0,21 1	0,20 8	0,27 6	0,20 5
DSBB 64fark- R	0,21 8	0,21 7	0,21 7	0,19 4	0,19 8	0,20 5	0,20 2	0,25 2	0,19 4
DSBB 128fark- R	0,21 8	0,21 7	0,21 7	0,19 4	0,19 8	0,20 5	0,20 3	0,25 2	0,19 4
DDBB 128fark	0,50 1	0,5	0,5	0,53 6	0,53 9	0,56 9	0,56 6	0,68 2	0,53 6
DDBB 200fark	0,45 2	0,45 1	0,45 1	0,44 5	0,44 7	0,48 9	0,48 4	0,55 8	0,44 5

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	Deflate 7z
Orijinal Resim	16,7 72	16,8 74	16,8 75	16,0 1	16,0 24	15,2 4	15,2 41	19,4 05	16,0 1
EBMP+RLE	18,9 09	18,9 08	18,9 13	16,1 39	16,1 76	16,7 73	16,7 32	26,0 38	16,1 39
EBMP	15,9 09	16,6 6	16,6 63	15,8 97	15,8 96	16,5 54	16,5 44	20,1 85	15,8 97
SSBB 64px	15,4 9	15,9 92	15,9 98	16,1 86	16,1 91	16,9 57	16,8 69	20,3 13	16,1 86
SSBB 128px	15,5 41	16,1 49	16,1 53	15,9 69	15,9 72	16,7 37	16,6 7	20,2 18	15,9 69
SSBB 256px	15,6 83	16,4 03	16,4 09	15,9 18	15,9 18	16,6 1	16,5 65	20,0 81	15,9 18
DSBB 64fark- R	15,9 51	16,7 18	16,7 18	15,9 28	15,9 26	16,5 88	16,5 82	20,8 18	15,9 28
DSBB 128fark- R	15,9 16	16,6 83	16,6 84	15,8 81	15,8 75	16,5 7	16,5 64	20,3 88	15,8 81
DDBB 128fark	15,9 02	16,6 53	16,6 55	15,9 12	15,9 11	16,5 82	16,5 77	20,1 85	15,9 12
DDBB 200fark	15,9 15	16,6 65	16,6 68	15,9 04	15,9 03	16,5 63	16,5 56	20,1 94	15,9 04

8. Kaynaklar

- [1] ISO/IEC 10918-1, CCITT Rec. T.81, 1992, "Digital Compression And Coding of Continuous-Tone Still Images – Requirements And Guidelines (JPEG)" (1994).
- [2] ISO/IEC 15444-1:2004, "JPEG 2000 image coding system: Core coding system" (2000).
- [3] Christopoulos C., Skodras A., Ebrahimi T., "The JPEG2000 still image coding system: An Overview", IEEE Transactions on Consumer Electronics, 46(4), 1103-1127 (2000).
- [4] ITU-T Rec. T.832, 2009, ISO/IEC 29199-2, "JPEG XR image coding system: Image coding specification" (2010).

[5] Rao K. R., Yip P., “Discrete Cosine Transform: Algorithms, Advantages, Applications”, Academic Press, San Diego, CA, USA (1990).

[6] Jensen A., Cour-Harbo A., “Ripples in Mathematics: The Discrete Wavelet Transform”, Springer-Verlag, Berlin, Germany (2001).

[7] CompuServe, “Graphics Interchange Format(sm), Version 89a”, CompuServe Incorporated, Columbus, Ohio (1990).

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

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