DEÜ MÜHENDİSLİK FAKÜLTESİ
FEN VE MÜHENDİSLİK DERGİSİ
Cilt: 7 Sayı: 3 s. 1-21 Ekim 2005

FRAKTAL GÖRÜNTÜ SIKIŞTIRMADA '*HASH*' FONKSİYONLARINA DAYANAN YENİ BİR SINIFLANDIRMA YÖNTEMİ

(A NEW CLASSIFICATION METHOD FOR FRACTAL IMAGE COMPRESSION BASED ON HASH FUNCTIONS)

Cengiz GÜNGÖR*, Aydın ÖZTÜRK*

ÖZET/*ABSTRACT*

Bu çalışmada '*hash*' fonksiyonları kullanılarak fraktal görüntü sıkıştırma yöntemi için bir sınıflandırma şeması önerilmektedir. Fraktal görüntü sıkıştırma yöntemi, görüntü içinde benzer parçaların bulunması esasına dayanır. Doğayla ilgili görüntülerin sıkıştırılmasında diğer yöntemlere göre çok daha etkin olan bu yöntem aynı zamanda diğer görüntü sıkıştırma yöntemlerinin iyileştirilmesinde de etkilidir. Ancak, benzer parçaların aranması oldukça fazla karşılaştırma hesabı yapılmasını gerektirmektedir. Hesaplama maliyetini düşürmek amacıyla, görüntü üzerinde ele alınan parçaların ve bunlarla eşleştirilmesi öngörülen parçaların sınıflandırılarak, benzerliklerin bu sınıflar içinde aranması genellikle tercih edilen yöntemdir. Bu çalışmada önerilen sınıflandırma yöntemi ile benzer parçaların basit bir şekilde bulunabileceği gösterilmiştir. Elde edilen sonuçlar, sıkıştırma oranının yüksek tutulduğu durumlarda, önerilen yöntemin genellikle diğer fraktal yöntemlerden daha iyi olduğunu göstermiştir. Yöntem, benzerlerinden çok daha basit ve hızlı bir şekilde uygulanabilmektedir. Ayrıca renkli resimler ve video görüntüleri için daha kaliteli sonuçlar elde etmek amacıyla diğer sıkıştırma yöntemleriyle birlikte kullanılabilmektedir.

In this paper, it is proposed a fractal image compression classification schema which is based on hash functions. Fractal image compression is based on finding similar image blocks. It is an efficient method for compression of natural images and increases quality of other image compression at the same time. However, searching similar pieces requires tedious computations. In order to reduce the computational cost, domain blocks are classified and search for finding similar pairs are performed within those classes. In this study, it is shown that similar pairs can be easily obtained by the proposed classification method. The results obtained showed that the proposed method generally obtains more realistic results than the other fractal methods on at high compression rates. The method is simplest and faster than the others. Moreover, it is suitable for using with color images, video and it can be combined with other image compression methods for better quality.

ANAHTAR KELİMELER/*KEYWORDS*

Fraktal görüntü sıkıştırma, Sınıflandırma, '*Hash*' fonksiyonları, '*Hash*' tabloları
Fractal image compression, Classification, Hash functions, Hash tables

* Ege Üniversitesi Uluslararası Bilgisayar Enstitüsü Bornova/İzmir

1. GİRİŞ

Klasik görüntü sıkıştırma yöntemlerinin esası, eldeki görüntünün orijinalinden daha az bitle ifade edilmesine dayanır. Bunu gerçekleştirmek için görüntüdeki mevcut tekrarlı bilgilerden yararlanılır. Fraktal görüntü sıkıştırma yöntemi ise klasik yaklaşımdan farklı olarak görüntü içinde birbirlerine benzeyen parçaların eşleştirilmesi esasına dayanır. Ancak, çoğu zaman benzer çiftlerin bulunma maliyetlerinin yüksek olması, önemli bir sorundur.

Mevcut yöntemler içerisinde hızlı sonuç verenler sıkıştırma oranı ve sonucun kalitesi açısından belirli bir seviyenin altında kalırken, fraktal görüntü sıkıştırma gibi yüksek kaliteli teknikler yavaş teknikler olarak bilinmektedir. Yüksek kaliteli olarak bilinen *Wavelet* teknikleri, getirdiği yenilikler nedeniyle çok hızlı sonuç vermektedir. *Wavelet* tekniği ile fraktal görüntü sıkıştırma tekniği birlikte uygulandığında (*hybrid fractal zerotree wavelet*), elde edilen sonuçlar daha da iyi olmaktadır (Kim vd. 2002). Ancak fraktal görüntü sıkıştırma mantığındaki eşleştirmeler yine işlem süresini biraz uzatmaktadır.

Fraktal görüntü sıkıştırmada kullanılan parçalar genellikle orijinal görüntü üzerinde kare ya da dikdörtgen bloklar şeklinde seçilir. Bu blokların sınıflandırılması ve eşleştirme işleminin bu sınıflar içerisinde yapılması, eşleştirme işlemini hızlandırmak için en çok tercih edilen tekniktir. Bu bağlamda, 2×2 boyutlarına indirgenmiş blokların bir özelliği olan *kanonik* formlar ve varyans değerleri dizilişlerine dayanan bir sınıflandırma şeması geliştirilmiştir (Fisher, 1995). Kominek, çok boyutlu veri uzayını indekslemek için *r-tree* üzerine kurulu bir sınıflandırma şeması geliştirmiş (Kominek, 1995), Saupe aynı amaçla *kd-tree* kullanmıştır (Saupe, 1995).

Bu çalışmamızda, '*hash*' fonksiyonlarına dayanan yeni bir sınıflandırma tekniği ileri sürülmüştür. Elde edilen ampirik sonuçlar, söz konusu yöntemle yüksek sıkıştırma oranları ve yüksek kaliteye, hızlı bir şekilde ulaşılabildiğini göstermiştir. Yöntemdeki sınıflandırma işlemi çok basit ve hızlı olarak uygulanırken, sınıflandırma tekniğinin başarısı nedeniyle birbirleriyle ilgili görüntü parçalarına ulaşım da son derece hızlı olmaktadır.

İkinci bölümde fraktal görüntü sıkıştırma ile ilgili genel bilgiler verilmiş, ileri sürülen sınıflandırma tekniği ve bununla ilgili konular üçüncü bölümde verilmiştir. Dördüncü bölümde ise elde edilen deneysel sonuçlara yer verilmiştir. Beşinci bölümde elde edilen sonuçların mevcut fraktal sıkıştırma algoritmaları ve *wavelet* yöntemi karşılaştırması ve altıncı bölümde de ileriye yönelik önerilen ve tartışma verilmiştir.

2. FRAKTAL GÖRÜNTÜ SIKIŞTIRMA

Doğal bir görüntünün, geometrik şekillerin temelleri olan noktalar, çizgiler ve düzlemler yardımıyla oluşturulması çok zor bir iştir. Dağ, bulut, bitki örtüsü gibi doğal şekillerin kolayca oluşturulması, ilk defa 1980'li yılarda Mandelbrot'un geliştirdiği fraktal geometri teknikleri ile mümkün olmuştur (Mandelbrot, 1983). Fraktal geometride bir şeklin elde edilmesi için bir başlangıç görüntüsüne belli bir dönüşümün tekrar tekrar uygulanması gerekir. Belli bir tekrar sonucunda görüntü sabitleşmekte, işleme devam etmeye gerek kalmamaktadır. Bir başka ifadeyle doğal şekiller, bir takım dönüşümleri temsil eden matrislerle oluşturulabilmektedir. İstenen bir doğa manzarasını elde etmek içinse, bu tür dönüşümler yardımıyla elde edilen objeleri birleştirmek yeterli olmaktadır.

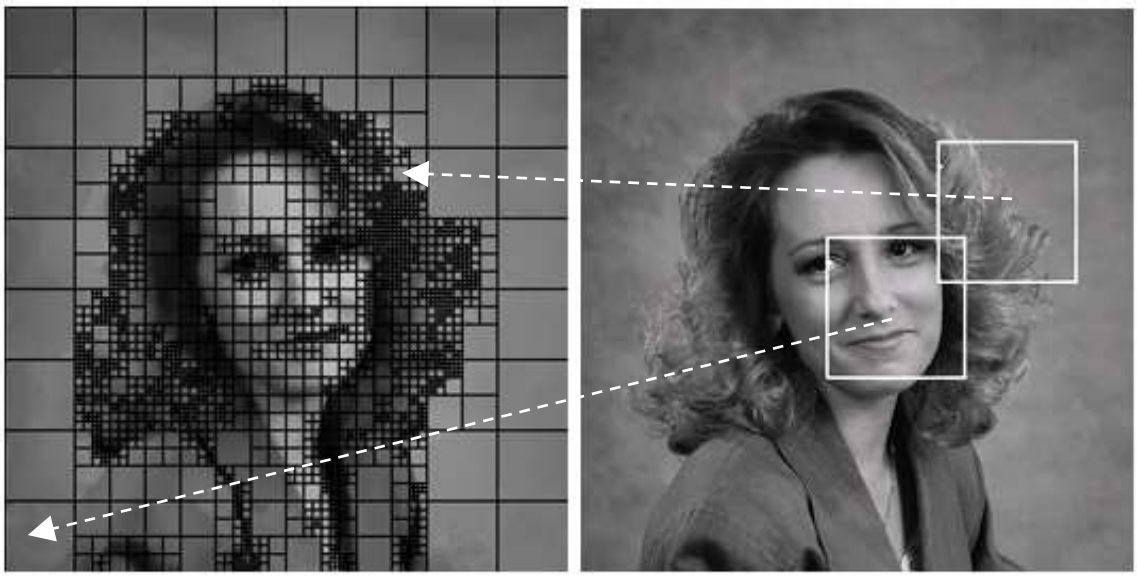
Fraktal geometrinin tersine işleyip işlemeyeceği sorusundan hareket eden matematikçiler, bir görüntü içinden alınacak parçalarla o görüntünün bir benzerini bir takım dönüşümlerden sonra elde etmenin mümkün olabileceğini göstermişlerdir. Bu konudaki ilk çalışma 1988 yılında Barnsley tarafından yapılmıştır (Barnsley, 1992). Fakat birkaç özel resim dışında tüm görüntünün bir dönüşümünü bulmak nerdeyse imkansızdır. Söz konusu soruna, Jacquin'in

çalışması ile çözüm bulunmuştur (Jacquin, 1992). Jacquin, eldeki görüntüyü parçalara ayırıp, her parça için ayrı bir dönüşüm bulup bunları birleştirerek orijinal görüntünün benzerini elde eden bir algoritma geliştirmiştir. Bu yöntemin en önemli uygulaması görüntü sıkıştırma konusunda olmuştur.

Fraktal görüntü sıkıştırma tekniği, görüntü üzerinde oluşturulan referans blokları (*range*) ile, yine aynı görüntü üzerinde referans bloklardan daha büyük olacak şekilde seçilen test bloklarının (*domain*) eşleştirilmesi esasına dayanır. Referans blokları görüntüyü oluşturduğu için birbirleriyle çakışmayacak şekilde düzenlenir. Fakat test blokları için böyle bir kısıtlama yoktur, hatta fazla test bloğu elde etmek adına her bir pikselden başlamak suretiyle bir test bloğu oluşturulabilir.

Referans blokları belli bir boyutta seçilir ve bu boyutun dışına çıkılmazsa, sıkıştırma oranı da sabit bir değerde olacaktır. Detayları yüksek olan, örneğin bir orman gibi görüntülerde küçük boyutlu blokların seçilmesi uygundur. Herhangi bir görüntü ele alınıp, görüntünün detayları incelendiğinde ise kimi bölgelerin görsel olarak düşük detaylar içerdiği, yani piksel değerleri açısından homojen dağılım gösteren bölgeleri olduğu, kimi bölgelerin ise çok fazla detay içerdiği görülmektedir. Bu durumda düşük detaylı bölgelerde büyük referans blokları, yüksek detay içeren bölgelerde ise küçük referans blokları seçmek uygundur. Bunu gerçekleştirmenin en kolay yolu, referans bloklarını büyük kareler şeklinde seçerek bunları bir test bloğu ile eşleştirmeye çalışmakla başlamak, iyi bir eşleştirme bulunamaması durumunda bloğu parçalayarak daha küçük parçaları eşleştirilmeye çalışmaktır. Bu yöntem *quadtree* yöntemi olarak bilinir (Fisher, 1995). *Quadtree* yönteminde $2^k \times 2^k$ boyutlarında seçilen referans bloğuna uygun bir eşleştirme bulunamazsa, bu blok dört eşit parçaya bölünerek, $2^{k-1} \times 2^{k-1}$ boyutlu bloklarla işleme devam edilir. Şekil 1'de örnek bir *quadtree* uygulaması görülmektedir. Bu örnekte 64×64 boyutlu referans bloklardan başlayarak, gerektiğinde 32×32, 16×16, 8×8 ve 4×4 boyutlu referans bloklara kadar inilmiştir. Ayrıca, detayların arttığı bölgelerde küçük boyutlu referansların kullanıldığı dikkati çekmektedir.

Test bloklarının boyutları işlem basitliği sağlamak amacıyla, referans bloklarının 2 katı olarak seçilir, bloklarının başlangıç noktaları arasındaki uzaklıkta da sabit bir değer seçilir. Bu çalışmamızda test blokları arasındaki mesafe 4 piksel olacak şekilde seçilmiştir.



Şekil 1. Carol görüntüsünde örnek referans ve test blokları

İyi bir eşleştirme sağlamak için referans ve test blokları hem piksel dizilişleri, hem de piksel içerikleri açısından belli dönüşümlere uğratılarak karşılaştırılırlar. Bu kapsamda en çok tercih edilen yaklaşımlar şunlardır :

- Piksellerin dizilişi farklı olsa da bloklar arasında bir benzerlik olabilir. Öyle ki birbirine benzetmek için referans ya da test bloğunun biraz döndürülmesi veya bir eksene göre simetriğinin alınması gerekebilir. Bloğun özgün dizilişi ile 90, 180 ve 270 derece dönmüş halleri ve bu dört formun herhangi bir eksene göre simetrikleri olarak sekiz farklı simetri işlemi mümkündür. Örneğin referans bloğu üzerinde bu işlem yapılırsa, bu sekiz simetri işleminden tek tek geçirilen referans bloğu bir veya birkaç dönüşümde test bloğu ile benzer bir forma gelince iyi bir eşleştirme tespit edilebilir.
- Piksel değerleri arasında parlaklık, kontrast farkı ilişkisi vardır. Bu ilişki genel olarak $E(Y)=(\alpha+\beta x)$ şeklinde basit doğrusal bir modelle ifade edilebilir. Burada α ve β modelin parametrelerini, $E(Y)$ de Y'nin beklenen değerini göstermektedir. Verilen $\{(Ri, Di) , i=1,2,...,n)\}$ değer çiftleri için α ve β parametreleri en küçük kareler yöntemine göre tahmin edilebilir. Böylece, tahmin edilen model Eşitlik 1'deki gibi bulunur.

$$\hat{R}_i = \hat{\alpha} + \hat{\beta} D_i \tag{1}$$

Buradan hareketle, referans ve test bloklarının ortalamaları Eşitlik 2'deki bağıntılar olmak üzere α (parlaklık) ve β (kontrast) tahminleri Eşitlik 3'deki gibi bulunur.

$$\overline{R} = \frac{1}{n}\sum_{i=1}^{n} R_i \text{ ve } \overline{D} = \frac{1}{n}\sum_{i=1}^{n} D_i \tag{2}$$

$$\hat{\alpha} = \overline{R} - \hat{\beta}\overline{D}$$

$$\hat{\beta} = \frac{\sum_{i=1}^{n}(R_i - \overline{R})(D_i - \overline{D})}{\sum_{i=1}^{n}(D_i - \overline{D})} \tag{3}$$

Bloklar arasındaki benzerliği belirlemek için, Eşitlik 4'deki hata kareler ortalamasının karekökü (*Root Mean Squares* (*RMS*)) bir ölçüt olarak kullanılmıştır.

$$RMS(R,D) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(R_i - (\hat{\alpha} + \hat{\beta} * D_i))^2} \tag{4}$$

Eşleştirme işlemi için her bir referans bloğu ile olası tüm test blokları arasında *RMS* testi uygulanıp, minimum *RMS* değerini döndüren blok çifti belirlendikten sonra bu değer belli bir eşikten düşükse incelenen çift benzer kabul edilip, çıktı dosyaya kodlama yapılır. Eğer eşik değeri düşük seçilirse, sonucun kalitesini arttırır, sıkıştırma oranını düşürür. Yüksek seçilirse, yüksek sıkıştırma oranları fakat düşük kalite döndürür. Bu çalışmamızda eşiğin değişik değerleri ile denemeler yapılmıştır, fakat eşiğin varsayılan değeri 8 kabul edilmiştir.

256 gri seviyeli iki görüntü arasındaki kalite farkı ise aralarındaki *RMS* ölçümünün bir fonksiyonu olan *PSNR* (*Peak Signal to Noise Ratio*) ile ifade edilir (Eşitlik 5).

$$PSNR = 20\log_2\left(\frac{255}{RMS}\right) \tag{5}$$

Eşleştirme işlemi için en sade yaklaşım, hiçbir sınıflandırma yapmadan, her referans bloğu için tüm olası test bloklarını sekiz simetri işleminden geçirip test etmektir. Bu teknik kaba kuvvet (*brute force* veya kısaca *BF*) olarak bilinir.

Fisher bir çalışmasında bloklara ait kanonik formları kullanarak karşılaştırma yapmanın yeterli kaliteyi döndürdüğünü göstermiştir (Fisher, 1995). Kanonik formlar sayesinde sekiz simetri işlemini uygulanmaya gerek kalmaz. Sadece test bloğu ile aynı kanonik forma getirmek için gerekli simetri işlemi belirlenir ve bu dönüşüm referans bloğuna uygulanır. Elde edilen dönüştürülmüş referans bloğu, test bloğu ile *RMS* testine sokulur. Böylece toplam işlem adedini sekiz kat düşürmek hedeflenmiştir. Fakat hızlı *BF* olarak adlandırılan bu yaklaşımla tam *BF* değerleri asla elde edilemez. Bu çalışmada geliştirilen teknik Fisher'in hızlı yöntemini kullandığından, hızlı *BF* değerlerine ulaşmak hedeflenmiştir.

3. ÖNERİLEN YÖNTEM

Bu çalışma ile önerilen sınıflandırma yönteminin esası '*hash*' fonksiyonlarına dayandırılmıştır. Yöntem fraktal görüntü sıkıştırmada kullanılmak üzere tasarlanmasına karşın sinyal işleme konusuna giren diğer uygulamalarda kullanılabilecek bir yapıya sahiptir. Buna göre, piksel değerlerinden oluşan test blokları için tanımlı alanlar (*slotlar*) yaratılmakta, bir '*hash*' fonksiyonu aracılığıyla söz konusu test blokları bu alanlara yönlendirilerek bellekte test bloklarından oluşan bağlı listeler (*linked lists*) oluşturulmaktadır. Böylece oluşturulan her bir '*slot*', bir test bloğu sınıfını temsil etmekte, bu sınıfa giren test blokları da birbirlerine benzeyen özellikte oldukları öngörülmektedir. Bir referans bloğu da aynı işlemden geçirildiğinde elde edilen sınıf numarası ile benzeri olan test bloklarına kolayca ulaşılmış olur.

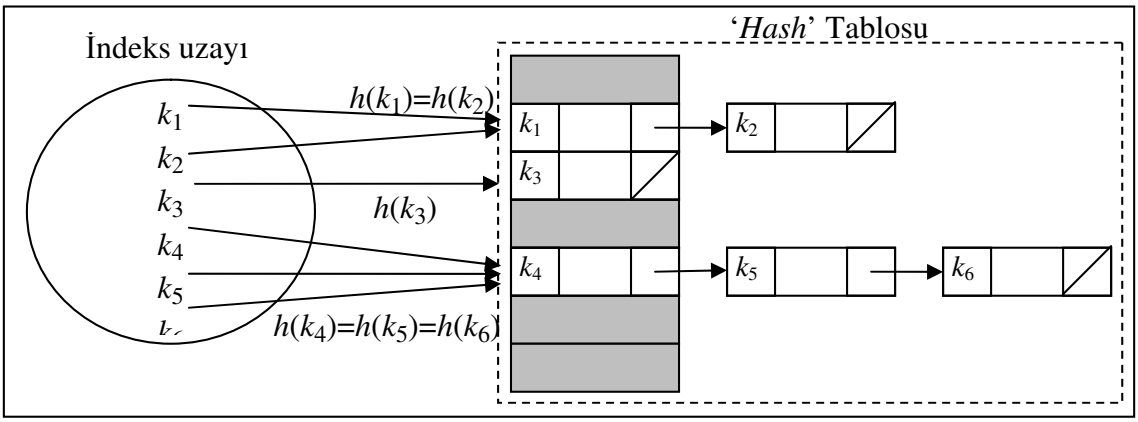
3.1. '*Hash*' Tabloları

Çoğu uygulama, dinamik bir veri yapısına ve bu yapıda bilgileri saklamak için ekleme, silme ve arama olarak bilinen temel fonksiyonlara ihtiyaç duyar. Klasik yaklaşımlarda, saklanmak istenen bilgiler bir dizide veya bağlı listede tutulabilir. Dizi yapısında, sıra ile artan indeks değerlerinin her biri için ayrı bir pozisyon bilgisi tutulur. Bağlı listede ise ardışık olmayan indeks bilgileri tutulabilir. Dizi yapısında bazı alanlar da boş kalabilirken, bağlı listelerde dolu olmayan kayıt (*node*) yoktur ve her kayıt kendinden sonrakini işaret eder. En kötü durum çalışma zamanları incelendiğinde, dizilerde bilgi arama $O(1)$ zaman alırken, bağlı listelerde $O(n)$ zaman alacaktır. Benzer indeks değerlerine sahip olan birden fazla (*duplicate*) verinin saklanması gerekirse dizi yapısında bu mümkün değildir. Bağlı listelerde bu tür bilgiler sorunsuz tutulabilir.

Bir '*hash*' tablosu yapısında, dizi ve bağlı listenin karması olan bir yapı kullanılır (Cormen vd., 2003). İndeks bilgileri dizi yapısı ile tutulurken, her bir indeks altında benzer indekse sahip olan veriler bağlı listelerde tutulur. '*Hash*' tablosu uygulamaları içinde indeks değerleri de '*hash*' fonksiyonları ile elde edilir. Bir '*hash*' fonksiyonu :

- İçeriği bilinse bile kapalı bir kutudur, çıktısından hareketle girdisi asla tahmin edilemez,
- Tutarlıdır, rasgele çıktı üretmez, aynı girdi ile her zaman aynı çıktı (indeks) elde edilir,
- Fakat farklı girdilerle benzer çıktı verebilir.

Şekil 2'de bir '*hash*' tablosu yapısı gösterilmiştir.



Şekil 2. Örnek bir '*hash*' tablosu yapısı

3.2. Önerilen Sınıflandırma Yöntemi

Test blokları havuzu üzerinde yapılan etkin bir sınıflandırma ile birbirine benzer özelliklere sahip test bloklarından oluşan farklı sınıflar oluşturulmuş olur. Bu sınıflara ve aralarında belli bir ilişki olan yakın sınıflara hızlı bir şekilde ulaşılmalıdır. En önemlisi, ulaşılan grupta aranılan test bloğunun bulunması ihtimali çok yüksek olmalıdır.

Fisher ve Saupe kendi sınıflandırma tekniklerinde *kd-tree* kullanmışlardır (Saupe, 1995). Bu sınıflandırmayı yapmak için görüntü bloğundan elde edilen standart görüntü vektörlerini kullanırlar. Bu vektörler yardımıyla bir ikili ağaç veri yapısı oluşturulur, test blokları ağaç üzerinde dağıtılır, daha sonra benzerlik arayışı da bu ağaç üzerinde yapılır.

Bu çalışmada önerilen sınıflandırma sistemine göre, üzerinde çalışılan bloklar eğer 4x4'den büyük iseler, küçültülerek 4x4 boyutlarına indirgenmekte, böylece elde edilen 16 değerin her biri bir '*hash*' fonksiyonundan geçirilerek sonuçlar bloğun sınıf numarasını belirlemek amacıyla kullanılmaktadır.

Söz konusu '*hash*' fonksiyonlarına girdi olarak verilen değer, blok ortalamasından büyük veya eşitse 1, küçükse 0 çıktısı elde edilmektedir (Eşitlik 6).

$$h_i = \begin{cases} 1 : p_i \geq \bar{x} \\ 0 : p_i < \bar{x} \end{cases} \qquad (i = 1,...,16) \tag{6}$$

Eşitlik 6'daki h_i değerleri aracılığıyla iki farklı sınıflandırma geliştirilmiştir. İlki basitçe h_i değerlerinin toplamını sınıflandırma numarası olarak kullanırken, ikincisi farklı bir teknikle, test bloklarından oluşan havuzu çok daha fazla sayıda gruplara ayırmaktadır.

3.2.1. Basit Sınıflandırma Tekniği ve Q_n İstatistiği

Eğer h_i değerlerinin toplamı sınıf numarası olarak alınırsa, test bloğunun sınıfı bir k_T değeri olarak bulunur ve bu değer 1 ile 16 arasında değişir (Güngör vd., 2004). Bu 16 değer altında blokların dağılımı tüm görüntüler için yaklaşık aynıdır ve Çizelge 1'de bir örneği görüldüğü gibi 7, 8 ve 9. sınıflarda daha fazla blok içerir.

Test bloklarının sınıf numaraları basitçe belirlendikten sonra, aynı numaralı bloklar 'hash' tablosu mantığıyla listeler altında toplanırlar. Ancak yapılan eşleştirme işlemi sayısını düşürmek ve hız sağlamak için ayrıca bir teknik gerekmiştir. Bu amaçla Q_n istatistiği adı verilen basit bir korelasyon testi geliştirilmiştir (Güngör vd., 2004).

Q_n istatistiği, (X_1, Y_1), (X_2, Y_2) ... (X_n, Y_n), gibi iki-değişkenli (*bivariate*) dağılımdan çekilen bir şans örneğinin değerlerinin, 0 ve 1'den olun tek boyutlu örneğe dönüştürülmesi esasına dayanır. Bu örnekteki i'nci (i=1, 2, ...,n) şans değişkeni çifti için, bir şans değişkeni Eşitlik 7'deki gibi tanımlanabilir.

$$I_i = \begin{cases} 1 & (X_i - \overline{X})(Y_i - \overline{Y}) \geq 0 \text{ veya } (X_i - \overline{X}) = (Y_i - \overline{Y}) = 0 \\ 0 & \text{aksi taktirde} \end{cases} \quad (7)$$

$$\overline{X} = \sum_{i=1}^{n} X_i / n$$

$$\overline{Y} = \sum_{i=1}^{n} Y_i / n$$

Bu çalışmada, Eşitlik 8'deki istatistik, korelasyonun bir ölçüsü olarak tanımlanmıştır.

$$Q_n = \frac{1}{n} \sum_{i=1}^{n} I_i \quad (8)$$

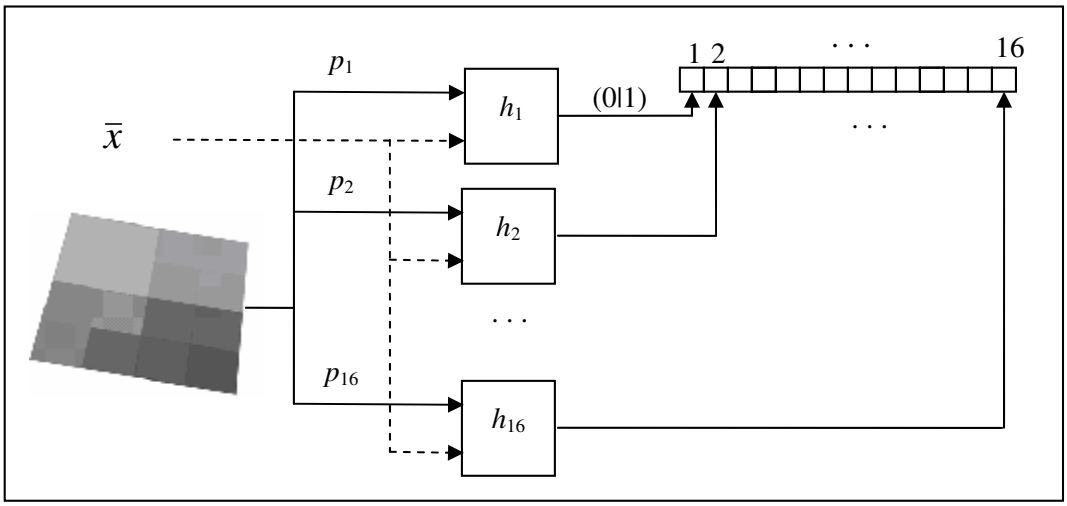
Buna göre, {(X_i, Y_i), (i=1,2 ... n)} şans örneğinde, bir değişken çiftine ilişkin değerlerin ortalamadan ayrılışlarının işaretleri aynı yada ikisinin de değeri sıfır ise, Q_n'in hesaplanmasında +1; işaretler farklı ise 0 olarak işlem görmektedir. Böylece, örnekte her iki değişkene ilişkin ortalamadan ayrılışlar hep artan bir seyir izliyorsa Q_n=1, biri hep artarken diğeri hep azalıyorsa Q_n=0 olacaktır. Diğer taraftan, ortalamadan ayrılışlar birbirinden bağımsız seyrediyorsa bu durumda da Q_n'in 0,5'e yakın bir değer alması beklenir. Sonuç olarak $0 \leq Q_n \leq 1$ 'dir. Bu özelliği bakımından Q_n, r ile örtüşmektedir.

3.2.2. Gelişmiş sınıflandırma tekniği

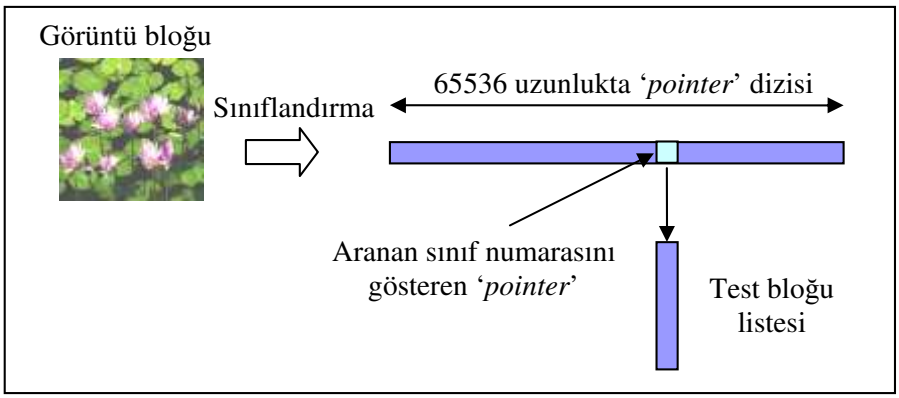
Bu teknikte, Eşitlik 6'daki '*hash*' fonksiyonları ile elde edilen 0 veya 1 çıktı bilgileri ile 16 bitlik bir sınıf numarası elde etmek amaçlanmıştır. Bu amaçla Eşitlik 9'daki toplam alınmaktadır. Şekil 3'de girdi olarak verilen bir görüntü bloğunun bu teknikle sınıf numarasının elde edilişi gösterilmiştir.

$$H = \sum_{i=1}^{16} (h_i * 2^{16-i}) \quad (9)$$

Bu yapıda sınıf numaraları 1 ile 65535 arasında değişen bir değer olacaktır. Elde edilen sınıf numaraları yardımıyla, test bloklarından oluşan bir havuz oluşturmak için öncelikle her referans boyutu için 65536 uzunlukta bir '*pointer*' listesi tutulur. Bu listelerde her '*pointer*' kendi sınıfının ilk test bloğunu işaret eder, eğer sınıf numarası altında birden fazla blok varsa, her blok bir sonrakini işaret edecek şekilde bağlı liste yapısında tutulurlar.



Şekil 3. Piksel değerlerinin '*hash*' fonksiyonları ile bir sınıf numarasına dönüştürülmesi



Şekil 4. '*Hash*' tablosunun kullandığı '*pointer*' dizisi yapısı

3.2.2.1. Sınıflar arası akrabalık ilişkisi

Sınıflandırma teknikleri ne kadar etkin olursa olsun, havuzun belli bir mantıkla parçalara ayrılması sonucu, *BF* arama yapıldığında test edilen blok çiftlerinin çoğu atlanacaktır. Özellikle, büyük boyutlu referans ve test blokları sınıflandırma amacıyla 4×4 boyutuna küçültüldüklerinden, en iyi eşleştirmeyi sağlayan referans ve test bloğu çiftlerinin sınıf numaraları arasında birkaç bit fark olabilir. Oluşan fark bir algoritma hatası değildir, oluşturulan sınıfların birbirleriyle ilişkili olduklarını gösterir. Bu ilişki sınıflar arası akrabalık ilişkisidir ve eşleştirme işlemi yapılırken, tüm akraba sınıflar içinde arama yapılmalıdır. Çünkü bir referans bloğunun benzerinin kendi sınıf numarası ile aynı sınıftaki test blokları içerisinden çıkmaması durumunda birkaç bit fark içeren, akraba sınıflar içerisinde bir benzeri olabilir.

Bu yapıda, ilgilenilen sınıflara ulaşma ve gerektiğinde akraba sınıflara geçme işlemleri basit bir şekilde gerçekleştirilmiştir. Akraba bir sınıfa ulaşmak için, içeriğinde *m* adet biti 1 olan maskelerle referans bloğunun sınıf numarasını XOR işlemine tabi tutmak yeterlidir. Kullanılan maskenin içindeki 1 olan bit pozisyonları, referans bloğun sınıf numarasında aynı pozisyondaki bitlerini tersine çevirir, yani o bitin içeriği 0 ise 1, 1 ise 0 olur. Bu işlemden geçerek elde edilen sınıf numarası ile referans bloğun akrabalık derecesi *m* olmaktadır. 4x4 boyutlarındaki bir blok için *m*'inci dereceden akraba olabilecek durum sayısı Eşitlik 10'de tanımlanmıştır.

$$\binom{16}{m} = \frac{16!}{m!(16-m)!} \qquad m = 1,\ 2,\ \text{K}\ ,\ 16 \tag{10}$$

Buna göre, 4. akrabalığa kadar olan sınıfların sayısı aşağıdaki gibidir :

- 0. dereceden 1 akraba (kendi sınıfı),
- 1. dereceden 16 akraba (sınıf numarasında 1 bit farklılık),
- 2. dereceden 120 akraba (sınıf numarasında 2 bit farklılık),
- 3. dereceden 560 akraba (sınıf numarasında 3 bit farklılık),
- 4. dereceden 1820 akraba (sınıf numarasında 4 bit farklılık).

3.2.2.2. Bloklar arası korelasyonun tahminlenmesi

Blokların eşleştirilmesi esnasında (3) ve (4)'deki hesaplamaların yapılması fraktal görüntü sıkıştırma algoritmalarının en fazla işlem süresi alan bölümüdür. Daha basit bir teknik kullanarak benzerliğin tahmin edilmesi ve blokların bu tahmine göre değerlendirildikten sonra asıl eşleştirme testine alınmaları çok büyük bir hız sağlayacaktır. Benzerliğin bulunmasında referans ve test bloklarının Eşitlik 11'deki şekilde standardize edilmiş dönüşümleri kullanılır.

$$V_i = \frac{p_i - \bar{x}}{S} \qquad i = 1,\ 2,\ \text{K}\ ,\ 16 \tag{11}$$

Burada pi, bloktaki i'nci piksel değerini, $\bar{x}$, blok ortalamasını, S ise bloğun standart sapmasını göstermektedir. Buna göre, eldeki test ve referans blokları arasındaki korelasyon katsayısı basitçe Eşitlik 12'deki gibi ifade edilebilir.

$$r = \sum_{i=1}^{16} V_i^{(r)} V_i^{(t)} \tag{12}$$

Burada $V_i^{(r)}$ ve $V_i^{(t)}$ sırasıyla standart referans ve test vektörlerini göstermektedir.

4. UYGULANAN YÖNTEMLER

Bu bölümde önerilen basit ve gelişmiş sınıflandırma tekniklerinden elde edilen sonuçlar ve yöntemlerin detaylı incelemesi verilmiştir.

4.1. Basit Sınıflandırma Tekniği

Basit sınıflandırma tekniğinde test blokları havuzu basitçe 16 parçaya ayrılır. Fakat sınıflandırma işlemi sonuçlandığında her sınıf numarası altında eşit sayıda test bloğu elde edilememektedir. Ekler kısmında verilen test görüntülerinde Çizelge 1'deki dağılıma benzer dağılımlar elde edilmiştir.

Referans blokları belirlenirken *quadtree* mantığı uygulanmıştır. Buna göre, büyük boyutlu bloklarla başlayarak, iyi bir eşleştirme bulunamaması durumunda referans bloğunu dört eşit parçaya bölünmüş, bu dört parçanın eşleştirilmesi ayrı ayrı yapılmıştır. İşlemlere uygun eşleştirmeler buluncaya kadar devam edilmiştir. Bu çalışmada 32×32 boyutundan başlanmış, gerektikçe bloklar küçültülerek, 4×4 boyutlarına kadar inilmiştir.

Sınıflandırma tekniği elde edilen 16 değer üzerinden yapıldığı için, 4×4'ten büyük blokların sınıflandırılabilmesi için önce 4×4 boyutuna küçültülmesi gerekir, bu işlem komşu piksellerin ortalaması alınarak yapılmıştır. Ancak çok ender de olsa, blok küçültme işlemi

nedeniyle, benzer olması gereken bloklar farklı sınıflara düşebilmekte ya da bunun tersi durumlarla karşılaşılabilmektedir.

Çizelge 1. 512x512 Lena görüntüsü için sınıf listeleri içinde toplanan test blokları sayıları

Blok Boyutu	k_T															
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
4x4	1	17	63	294	624	1586	3007	4961	3114	1605	586	193	53	8	0	0
8x8	0	23	141	493	691	1469	2559	4393	2792	1646	765	544	83	24	2	0
16x16	8	48	200	790	766	1465	2219	3570	2218	1414	884	862	164	28	5	0
32x32	0	6	77	671	816	1535	2160	3130	1931	1274	724	306	125	14	0	0

Referans blokları için başlangıç aşamasında sınıflandırma yerine, bloklar eşleştirme işlemine alınmadan önce sınıflandırma uygulanmıştır ve elde edilen k_R sınıf numarası ile aynı numaralı sınıf altında bulunan test blokları eşleştirme işlemine tabi tutulmuştur. İyi bir eşleştirme sağlanamadığında, bir alt ve bir üst sınıflardaki test blokları denenmiştir. Bu durumda sınıf numaraları arasında ±1 fark oluşur. Bu tür sınıflara komşu veya akraba sınıf denilir ve aradaki ilişki sınıf numarası farklarının mutlak değeri olan $\Delta = |k_R - k_T| = 1$ ile ifade edilmiştir. Yine iyi bir eşleştirme bulunamazsa, iki alt ve iki üst sınıf ($\Delta=2$), daha sonra da üç alt ve üç üst ($\Delta=3$) sınıflar denenmiş. $\Delta=3$'den daha üst ilişkilerin ihmal edilebilecek düzeyde olduğu farklı görüntüler üzerinde denenerek bulunmuştur.

BF (*Brute force*) tekniği ile çeşitli blok büyüklüklerinde elde edilen eşleştirme sayıları ile Δ değerlerinin ilişkisi Lena görüntüsü için hesaplanmış ve Çizelge 2'de verilmiştir. Diğer görüntülerde de benzer sonuçlar elde edilmiştir.

Çizelge 2. Lena görüntüsünde en iyi referans-test eşleştirmelerinin Δ değerlerine göre dağılımı

Blok Boyutu	**Δ=0**	**Δ=1**	**Δ=2**	**Δ=3**	**Toplam**
4x4	1,834 (%84,3)	213 (%9,8)	87 (%4,0)	42 (%1,9)	2,176
8x8	649 (%92,7)	43 (%6,1)	8 (%1,1)	0 (%0,0)	700
16x16	265 (%90,4)	22 (%7,5)	6 (%2,0)	0 (%0,0)	293
32x32	98 (%93,3)	7 (%6,7)	0 (%0,0)	0 (%0,0)	105

Bu sonuçlara göre, blok boyutunun 4×4 olması haricinde kendi sınıfı ve ±1 komşu sınıf ($\Delta\leq1$) bile yeterli olmaktadır.

Q_n istatistiği bloklar arasındaki korelasyonu tahminlemektedir. Dolayısıyla bir eşik değerinin üzerinde gelen tahminler yakalandığında, asıl eşleştirme testleri uygulanmış ve hızlanma sağlanmıştır. Yapılan çalışmalar sonucu eşik değeri 0,75 olarak belirlenmiştir.

Çizelge 3'de, basit sınıflandırma ve Q_n istatistiği birlikte uygulanırken, Δ değerlerine bir üst sınır konulması ve bu sınırın arttırılmasının sonuca etkisine verilmiştir.

Diğer test görüntülerde benzer sonuçlar çıkmıştır. Elde edilen değerlere göre $\Delta\leq1$ seçiminin yeterli olduğu gözlenmiştir. Fakat Saupe'un kd-tree algoritmasının hızına bu teknikle ulaşılamamıştır.

Çizelge 3. Lena görüntüsünde.farklı Δ değerleri ile elde edilen sonuçlar.

Kullanılan Yöntem	Sıkıştırma Oranı	PSNR	Test Sayısı	Süre (sn)	BF 'a göre Hızlanma
Q_n ($\Delta = 0$)	20.67 : 1	33,771	2,966,642	3.34	×10.34
Q_n ($\Delta \leq 1$)	22.08 : 1	33,523	4,817,309	5.47	× 6.31
Q_n ($\Delta \leq 2$)	22.36 : 1	33,497	5,541,897	6.61	× 5.22
Q_n ($\Delta \leq 3$)	22.54 : 1	33,454	6,271,008	7.56	× 4.57
Saupe Kd-tree	22.11 : 1	33,749	216,400	1.39	×24.84
Brute-Force (*BF*)	22.72 : 1	33,743	65,508,848	34.53	---

4.2. Gelişmiş Sınıflandırma Tekniği

Gelişmiş teknikte akraba sınıfların da test edilmesi son derece önemlidir. Bu amaçla, 3 farklı görüntü için, maksimum akrabalık sayısının arttırılmasının etkisi Çizelgede 4'de verilmiştir.

Çizelge 4. Üç görüntü için, incelenen akraba sınıfların sayısının arttırılmasının sıkıştırma oranları ve PSNR değerlerine etkisi

(a) Sıkıştırma oranları değişimi

Yöntem	Lena		Mandrill		Goldhill	
Brute-Force	22.72	(%)	5.79	(%)	11.06	(%)
Kendi sınıfı	16.77	73.81	5.37	92.75	8.60	77.76
Kendi sınıfı + 1.Akrabalar	21.02	92.52	5.63	97.24	9.89	89.42
Kendi sınıfı + 1..2.Akrabalar	22.15	97.49	5.72	98.79	10.54	95.30
Kendi sınıfı + 1..3.Akrabalar	22.53	99.16	5.75	99.31	10.78	97.47
Kendi sınıfı + 1..4.Akrabalar	22.70	99.91	5.76	99.48	10.85	98.10
Kendi sınıfı + 1..5.Akrabalar	22.72	100.00	5.77	99.65	10.88	98.37

(b) *PSNR* değerlerinin değişimi

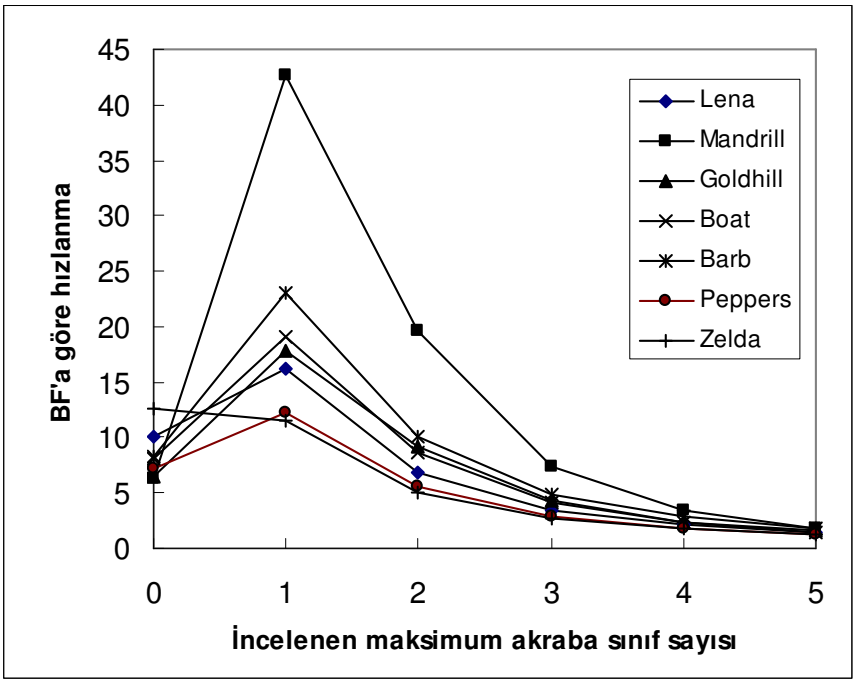
Yöntem	Lena		Mandrill		Goldhill	
Brute-Force	33.743	(%)	27.663	(%)	32.489	(%)
Kendi sınıfı	33.680	99.81	25.887	93.58	31.958	98.37
Kendi sınıfı + 1.Akrabalar	33.713	99.91	26.829	96.99	32.310	99.45
Kendi sınıfı + 1..2.Akrabalar	33.765	100.07	27.397	99.04	32.461	99.91
Kendi sınıfı + 1..3.Akrabalar	33.774	100.09	27.589	99.73	32.486	99.99
Kendi sınıfı + 1..4.Akrabalar	33.758	100.04	27.644	99.93	32.499	100.03
Kendi sınıfı + 1..5.Akrabalar	33.740	99.99	27.659	99.99	32.484	99.98

İşlem sürelerine bakıldığında, kendi sınıfı ile yetinip, hiçbir akrabalığa bakılmamasının işlem süresini uzattığı gözlenmiştir. İyi eşleştirmelerin akraba sınıflarda olması ihtimali göz ardı edilemeyecek kadar fazladır. Büyük boyutlu bir referans bloğunda uygulanan *RMS* testi, zaten fazla zaman almaktadır. Bir de büyük referans bloğuyla iyi bir eşleştirme bulunamazsa, *quadtree* uygulaması ile en son 4×4 boyutlu bloklara kadar inilmektedir, her bölme işleminde sonuçsuz kalan *RMS* testleri ek yük olarak kalmaktadır. Örneğin, Mandrill görüntüsü nerdeyse tümüyle 4×4 bloklar ile kodlanmaktadır, bu nedenle en fazla testin yapıldığı, en

uzun işlem zamanı olan görüntüdür. Sıkıştırma oranının düşük olması da daha çok 4×4 bloklar ile kodlanmasından kaynaklanmaktadır. Şekil 5'de işlem sürelerinin *BF* sürelerine göre değerlendirildiği hızlanma grafiği verilmiştir. Grafiğe göre 1. akrabalıkların eklenmesi ile yaşanan hızlanma, daha sonra hızla düşmektedir.

Elde edilen sonuçlara göre, 1. dereceye kadar akrabalıklar test edildiğinde *PSNR* fazla etkilenmediği, sıkıştırma oranının düşük kaldığı fakat hızlanmanın ×10 ile ×42 seviyelerine çıktığı gözlenmiştir. Sıkıştırma oranının önemli olmadığı, hızlanma istenen durumlarda 1. dereceye kadar akrabalıklarla yetinme tercih edilebilir bir seçenektir.

En iyi sonuçlar 3. ve daha yüksek derece akrabalıklara bakıldığı durumlarda oluşmaktadır. Sadece 3. dereceye kadar olan akrabalıklarda kalmanın ×3 ile ×10 arasında hızlanma sağladığı görülmüştür. Bu çalışma sonucunda, 4. derece ve daha yüksek akrabalık derecelerindeki sınıflara bakmanın işlem süresini uzattığı halde, görüntü kalitesini (*PSNR*) fazla etkilemediği, sıkıştırma oranını biraz daha arttırdığı gözlenmiştir. Bu nedenlerle çalışmanın devamında 3. dereceye kadar akrabalıkların kullanılmasına karar verilmiştir. Bu tercihle toplamda 697 sınıf listesi incelendiği için yeterli miktarda sıkıştırma oranı ve *PSNR* elde edilmektedir.



Şekil 5. İncelenen akraba sınıfların sayısının arttırılmasının hızlanma üzerine etkisi

4.3. Gelişmiş Teknikte Bloklar Arası Korelasyonun Tahminlenmesi

Bloklar arası korelasyonu tahminlemek için, (12)'deki standart referans ve test vektörlerinin çarpımlar toplamı kullanılmıştır. Yapılan testler sonucunda benzerlik tahmininin 0,7 eşik değeri altında olması durumunda iyi bir benzerlik elde edilemediği gözlenmiştir. Bu nedenle 0,7 eşik değeri kullanılarak, korelasyon tahminine göre *RMS* testleri uygulanmadan önce bir filtreleme yapılabilir. Çizelge 5'de *BF* arama yapılırken korelasyon filtresi kullanılmasının 8×8 boyutlu bloklar üzerine etkisi örnek olarak verilmiştir. Sonuçlardan anlaşılacağı gibi, *RMS* testlerinin çoğu hiç uygulanmadan, filtreye takıldığı için atlanmıştır.

Çizelge 5. *BF* uygulamasında, referans boyutu 8×8 iken, korelasyon tahmini 0,7 ve üstü olduğu için uygulanan ve tahmini 0,7'nin altında kaldığı için filtreye takılıp atlanan *RMS* testlerinin sayıları.

	Toplam Deneme	**Uygulanan Test Sayısı**	**Atlanan Test Sayısı**	
Lena	19,312,500	3,825,136	15,487,364	80.19%
Mandrill	56,125,000	2,694,966	53,430,034	95.20%
GoldHill	40,875,000	6,337,018	34,537,982	84.50%
Boat	35,500,000	5,114,235	30,385,765	85.59%
Barb	37,125,000	5,941,693	31,183,307	84.00%
Peppers	23,687,500	7,734,366	15,953,134	67.35%
Zelda	15,187,500	5,121,444	10,066,056	66.28%

Filtreleme işlemi tek başına bu denli etkili olmasına rağmen, önerilen sınıflandırma şemasında ve 3. derece akrabalıklara dek test uygulandığında asıl etkisini göstermiştir. Bu noktada, akrabalıkların tümünün test edilmesi ve iyi bir eşleştirme bulunca işlemin kesilmesi şeklinde iki farklı yol izlenmiştir ve sonuçlar Çizelge 6'da verilmiştir.

Çizelge 6. Önerilen yöntemde 3. dereceye dek akrabalıklarla, korelasyon filtresinin birlikte kullanılmasının "*brute-force*" yöntemi karşısında performans karşılaştırması

(a) 3. dereceye kadar olan akrabalıkların hepsinin denenmesi

	Sıkıştırma Oranı			**PSNR**			**Süre**		
	BF	**Elde Edilen**	**%**	**BF**	**Elde Edilen**	**%**	**BF**	**Elde Edilen**	**%**
Lena	22.72	22.45	98.81	33.743	33.780	100.11	34.53	8.41	24.36
Mandrill	5.79	5.76	99.48	27.663	27.591	99.74	105.41	12.13	11.51
GoldHill	11.06	10.91	98.64	32.489	32.490	100.00	61.48	12.09	19.66
Boat	11.60	11.41	98.36	33.681	33.746	100.19	57.64	12.09	20.98
Barb	9.80	9.73	99.29	30.186	30.151	99.88	66.98	11.84	17.68
Peppers	20.19	20.08	99.46	33.063	33.054	99.97	38.77	11.81	30.46
Zelda	31.20	31.00	99.36	33.447	33.465	100.05	28.06	8.92	31.79

(b) İyi bir eşleştirme yakalınca işlemin kesilmesinin etkisi

	Sıkıştırma Oranı			**PSNR**			**Süre**		
	BF	**Elde Edilen**	**%**	**BF**	**Elde Edilen**	**%**	**BF**	**Elde Edilen**	**%**
Lena	22.72	21.94	96.57	33.743	32.203	95.44	34.53	0.98	2.84
Mandrill	5.79	5.74	99.14	27.663	26.577	96.07	105.41	5.19	4.92
GoldHill	11.06	10.72	96.93	32.489	32.152	95.88	61.48	1.92	3.12
Boat	11.60	11.26	97.07	33.681	32.143	95.43	57.64	1.69	2.93
Barb	9.80	9.64	98.37	30.186	28.582	94.67	66.98	7.34	10.96
Peppers	20.19	19.68	97.47	33.063	31.597	95.97	38.77	1.69	4.36
Zelda	31.20	30.39	97.40	33.447	31.518	94.23	28.06	1.05	3.74

3. dereceye dek olan akrabalıklar incelenirken, iyi bir eşleştirme yakalandığında geri kalan akrabalıklar denenmeden, işlemin kesilmesi değerleri biraz düşürmektedir, fakat ×9 ile ×35 arasındaki hızlanma getirdiğinden, değer düşüşleri göze alınabilir.

4.4. Korelasyon Tahminlerine Göre Sıralı Liste Kullanımı

Çizelge 6 (a)'da görüldüğü gibi 3. dereceye kadar tüm akrabalıkların denenmesi *PSNR* ve sıkıştırma oranını *BF*'ya yaklaştırmaktadır. Fakat çok fazla *RMS* testi yapıldığı için bu teknik oldukça yavaştır. Bu yavaşlığı çözmek amacıyla, eşleştirme için ele alınan referans bloğu ile incelenecek olan tüm test bloklarından elde edilen tahminler alınıp, bu tahminlere göre en iyi tahmin üstte olacak şekilde sıralı bir liste oluşturmuştur. Yöntem, Saupe'un *kd-tree* algoritmasında kullanılanla aynıdır (Saupe, 1995). Bu sıralı listede, üstten *L* kadar tahmini veren çiftin *RMS* testine alınması yeterlidir. Bu yaklaşımla, her referans bloğu için en fazla *L* adet *RMS* testi yapılır. Sadece bu testleri uygulamakla, en iyi eşleştirmelerin çoğu bulunmuş olur. Bu şekilde en üst seviye sonuçlara hızlı bir şekilde ulaşılmıştır.

Yapılan testlerle *L* boyutunun 64'den fazla olması durumunda işlem süresinin arttığı fakat sıkıştırma oranı ve *PSNR* değerlerinde kayda değer artışlar olmadığı gözlenmiştir. Bu son teknikle beraber, önerilen sınıflandırma şemasının uygulanışı aşağıdaki adımlarla gerçekleştirilmiştir :

- Test blokları 65536 farklı sınıf listesi içerisine dağıtılmıştır.
- Her referans bloğu için, eşleştirme işlemine geçmeden önce, 3. dereceye kadar akrabalığı olan test bloklarının hepsi ile aralarındaki korelasyon tahminleri alınmıştır.
- Korelasyon tahmini 0,7 eşik değerinden az olan test blokları listeye eklenmemiştir.
- Test blokları elde edilen tahminlere göre 64 uzunlukta bir listede sıralanmıştır.
- Referans bloğunun eşleştirme işlemi elde edilen bu liste üzerindeki test blokları ile yapılmıştır.

Bu uygulama ile hesaplanması çok fazla işlem yükü isteyen *RMS* testlerinin her referans bloğu için en fazla 64 adet yapılması garantilenmiştir. Uygulamanın sonuçları Çizelge 7 ile verilmiştir.

Önceki uygulamalarda hızlanma görüntüye bağlı olarak çok farklılaşırken, burada sıralamadan dolayı biraz düşüş yaşanmış, ×12 ile ×21 aralığında çıkmıştır. Sıkıştırma oranı *BF* değerlerine göre düşüş göstermiş, fakat *PSNR* değerleri önceki değerlere göre çok daha yükselerek, *BF* değerleri civarında çıkmıştır.

Çizelge 7. Tüm tekniklerin birlikte kullanılmasının *BF* yöntemi karşısında performans değerleri

	Sıkıştırma Oranı			**PSNR**			**Süre**		
	BF	**Elde Edilen**	**%**	**BF**	**Elde Edilen**	**%**	**BF**	**Elde Edilen**	**%**
Lena	22.72	22.33	98.28	33.743	33.765	100.07	34.53	2.48	7.18
Mandrill	5.79	5.76	99.48	27.663	27.569	99.66	105.41	4.98	4.72
GoldHill	11.06	10.84	98.01	32.489	32.503	100.04	61.48	4.11	6.69
Boat	11.67	11.39	97.60	33.309	33.712	101.21	57.64	4.11	7.13
Barb	9.80	9.68	98.78	30.148	30.151	100.01	66.98	3.77	5.63
Peppers	20.19	20.01	99.11	32.996	33.039	100.13	38.77	3.31	8.54
Zelda	31.34	30.93	98.69	33.168	33.478	100.93	28.06	2.33	8.30

4.5. Performans Arttıran Diğer Teknikler

Eşleştirme işlemi esnasında bazı bloklarla boşu boşuna uğraşıldığı gözlenmiştir. Bu bloklarla iyi bir eşleştirme ya hiç mümkün değildir, ya da bu blokların çoğu atılsa da sonuçta elde edilen değerler fazla etkilenmemektedir. Bu tür blokların belirlenmesinde üç farklı yaklaşım incelenmiştir:

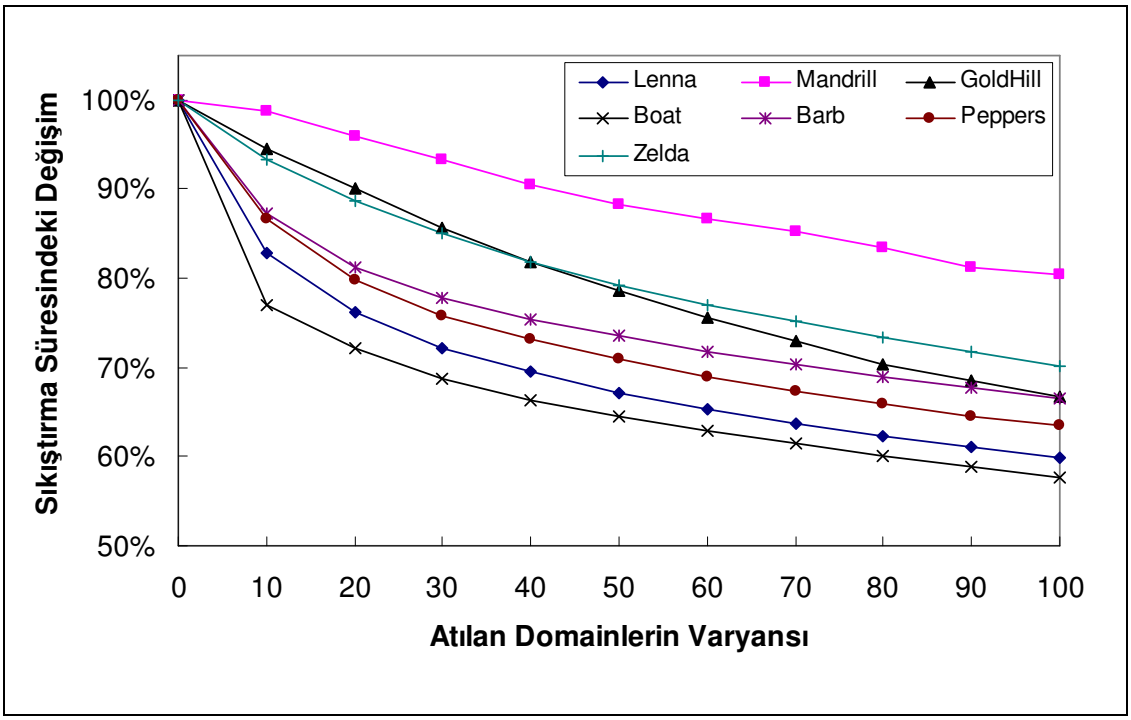
1) Homojene yakın dağılıma sahip (varyansı düşük) bloklar ilginç özellikler göstermektedir. Görüntünün özelliğine göre, içeriğinde homojene yakın pek çok blok vardır. Örneğin, Çizelge 8'de görüldüğü gibi Lena görüntüsünde çok fazla homojene yakın test bloğu olmasına karşın, bu bloklardan çok azı bir referans ile eşleştirilmektedir. Homojene yakın bir test bloğu havuzdan atıldığında, başka bir test bloğu ile eşleştirme her zaman mümkün olmalıdır, aksi taktirde elde edilen *PSNR* ve sıkıştırma oranları düşecektir.

Dolayısıyla test bloklarından oluşan havuzun hazırlanması aşamasında, homojene yakın test bloklarının bir kısmını havuza hiç eklemeyip, havuzda daha az test bloğu oluşturarak, hız sağlanmıştır. Lena görüntüsünde homojene yakın blok çok fazladır, Mandrill görüntüsünde ise tam tersi azdır. Mandrill için belli bir eşiğin altında varyansa sahip 4×4 boyutunda test blokları atıldığında, en fazla %25 bloğun atıldığı, Lena için ise %70'e varan bloğun atıldığı gözlenmiştir. Yani, Mandrill görüntüsünün test blokları havuzu bu işlemden daha az etkilenmiştir. Bunun nedeni Mandrill görüntüsünden alınan blokların piksel dağılışları ile ilgilidir. Bu teknikte, elde edilen PSNR ve sıkıştırma oranını çok çok az etkileyen bir eşik seçilmelidir. Bu değerin 50'den az olması gerektiği, incelenen görüntülerde gözlenmiştir. BF tekniğinde seçilen eşik değerine göre sürenin düşüşü ise Şekil 6'da verilmiştir.

Çizelge 8. 512×512 Lena görüntüsünde test bloklarının varyanslarına göre dağılımı

Varyans Aralığı	Tüm Test Bloklarının Dağılımı					Kullanılan Test Bloklarının Dağılımı				
	4x4	8x8	16x16	32x32	64x64	4x4	8x8	16x16	32x32	64x64
$0 \leq var < 5$	4290	1990	358	0	0	6	3	0	0	0
$5 \leq var < 10$	1795	1012	568	0	0	12	3	2	0	0
$10 \leq var < 15$	1066	864	317	13	0	5	5	0	0	0
$15 \leq var < 20$	717	686	273	20	0	3	8	2	0	0
$20 \leq var < 25$	573	501	230	23	0	8	6	2	0	0
$25 \leq var < 30$	390	420	229	41	0	7	6	2	0	0
$30 \leq var < 35$	336	359	173	45	0	8	8	2	1	0
$35 \leq var < 40$	275	324	176	38	0	11	3	3	0	0
$40 \leq var < 45$	289	284	150	50	0	6	5	2	1	0
$45 \leq var < 50$	211	253	162	29	0	10	7	1	0	0

2) Referans blokları görüntüyü oluşturduğu için asla atılamazlar. Sayıca çok fazla olan homojene yakın test blokları rahatlıkla atılabilirken, referans blokları için farklı bir yaklaşım gerekmiştir. Homojene yakın referans bloklarını, tek bir piksel değerinden oluşan homojen bir blokmuş gibi kodlamak, sıkıştırma oranından biraz kazanç sağlar. Bu tür bir referans blok için eşleştirme işlemine gerek yoktur. Bunlar sadece tek bir parlaklık değeri ile ifade edilirler, kontrast değerleri de sıfırdır. Kodlamada test bloğunun x ve y pozisyonları ile simetri işlem numarası çıktı dosyaya yazılmadığı için bit bazında elde edilen kazançlarla sıkıştırma oranını biraz artmaktadır.



Şekil 6. Homojene yakın test bloklarının varyans eşiğine göre atılmasının sıkıştırma süresi üzerine etkileri

Fakat, homojen referans blokları kullanarak kodlama yapmak görüntüde bozulmalara neden olacağı için, çok az bir süre kazancı getiren bu yaklaşımda çok düşük eşik değerlerinde kalınmalıdır, örneğin en fazla 5 veya 10 varyansa sahip referans blokları bu kapsama girebilir.

3) Bir başka performans iyileştirmesi, referans bloğu varyansı (σ_R) ile eşleştiği test bloğu varyansı (σ_T) ilişkisi incelenerek gerçekleştirilmiştir. Buna göre eşleştirme yapılmadan önce bakılan (σ_R - σ_T) farkı bize benzerlik için ipucu verebilmektedir. Bu fark değerin -200 gibi bir alt sınırdan fazla olması gerektiği gözlenmiştir. Test görüntülerle yapılan ölçümlerden elde edilen sonuçlara bakıldığında, (σ_R - σ_T) < -200 değeri döndüren referans-test blokları ile iyi eşleştirmeler yakalanamamıştır. 8×8 referans blokları için, *BF* tekniği ile bulunan eşleştirmelere varyans farkına göre filtreme uygulanan örnek bir çalışma Çizelge 9 ile verilmiştir. Her üç iyileştirme önerilen teknik üzerinde uygulandığında daha önce Çizelge 7 ile verilen sonuçlar, aşağıda Çizelge 10 ile verilen hale gelmiştir.

Elde edilen sonuçlara göre, hızlanma biraz daha artarak, ×18 ile ×25 aralığına çıkmıştır. Sıkıştırma oranı ve *PSNR* değerleri ise fazla etkilenmeden, yine *BF* değerleri civarında çıkmıştır.

Çizelge 9. Varyans farkı ≥ -200 ise eşleştirme yapılmasının sonuçları

	Toplam Deneme	**Uygulanan Test Sayısı**	**Atlanan Test Sayısı**	
Lena	19,250,000	10,927,177	8,322,823	43.24%
Mandrill	55,937,500	28,326,197	27,611,303	49.36%
GoldHill	40,812,500	28,891,311	11,921,189	29.21%
Boat	35,312,500	20,690,280	14,622,220	41.41%
Barb	37,125,000	19,291,255	17,833,745	48.04%
Peppers	23,687,500	13,222,794	10,464,706	44.18%
Zelda	15,187,500	9,899,026	5,288,474	34.82%

Çizelge 10. Tüm iyileştirmelerin uygulandığı önerilen sınıflandırma tekniğinin *BF* yöntemi karşısında performans değerleri

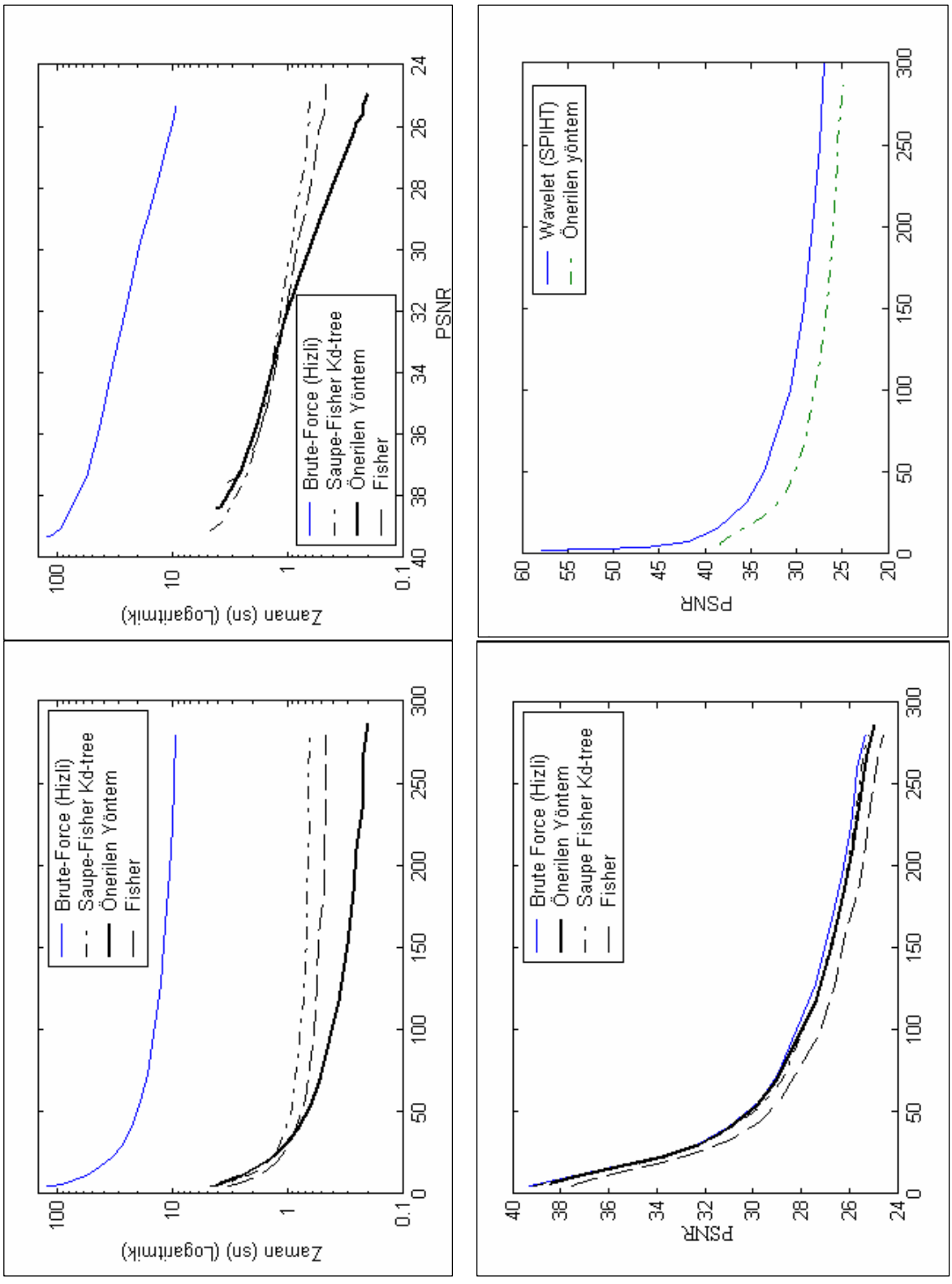
	Sıkıştırma Oranı			**PSNR**			**Süre**		
	BF	**Elde Edilen**	**%**	**BF**	**Elde Edilen**	**%**	**BF**	**Elde Edilen**	**%**
Lena	22.72	22.74	100.09	33.743	33.715	99.92	34.53	1.36	3.94
Mandrill	5.79	5.76	99.48	27.663	26.979	97.53	105.41	4.41	4.18
GoldHill	11.06	11.66	105.42	32.489	33.595	103.40	61.48	2.41	3.92
Boat	11.67	10.87	93.14	33.309	32.428	97.36	57.64	3.17	5.50
Barb	9.80	9.78	99.80	30.148	29.099	96.52	66.98	2.92	4.36
Peppers	20.19	20.49	101.49	32.996	32.951	99.86	38.77	2.00	5.16
Zelda	31.34	31.07	99.14	33.168	33.466	100.90	28.06	1.48	5.27

5. DİĞER YÖNTEMLERLE KARŞILAŞTIRMA

Şekil 7'de verilen grafiklerde, önerilen yöntem esas olarak *BF* yöntemi değerleri ile karşılaştırılmıştır. Ayrıca benzer kapsamda olan Fisher ve Saupe'un sınıflandırma teknikleri (Fisher, 1995; Saupe, 1995) ve yeni bir teknik olan *wavelet* görüntü sıkıştırma (Geoff, 1997) ile karşılaştırmalar da yapılmıştır. Elde edilen sonuçlara göre, yöntem *BF*'a yakın sonuçları, hızlı bir şekilde vermektedir. *Wavelet*, hızlı ve yüksek kaliteli bir tekniktir, Şekil 7'deki. grafikler yöntemin gücünü göstermektedir. *Wavelet* ile işlem süresi bazında karşılaştırmalar yapılmamıştır, çünkü *wavelet* önerilen yöntemden tümüyle farklı bir tekniktir ve çok hızlı sonuç vermektedir.

Elde edilen sonuçlara göre önerilen yöntem:

- Tüm sıkıştırma oranlarında iddialı, fakat yüksek sıkıştırmalarda çok daha hızlıdır.
- Belli bir *PSNR* elde edilmek istendiğinde yine çok hızlıdır.
- *BF* değerlerinde oran-bozulma özelliği göstermiştir.
- Yöntem, *wavelet* tekniği ile süre bazında yarışamasa da fraktal görüntü sıkıştırma mantığına göre elde edilebilecek en iyi değerlerle, oran-bozulma açısından *wavelet* ile elde edilen değerlere yaklaşmıştır.



Şekil 7. 512×512 boyutlu Lena görüntüsü için üstte üç farklı yöntem ile önerilen yöntemin oran-zaman ve bozulma-zaman grafikleri, altta aynı yöntemlerin oran-bozulma grafiği ile önerilen yöntemin, wavelet görüntü sıkıştırma ile oran-bozulma grafiği üzerinde karşılaştırılması.

6. SONUÇLAR VE TARTIŞMA

Bu çalışmada ulaşılmak istenen değerler, *BF* sıkıştırma oranı ve *PSNR* değerleri olmuş ve bu değerlerin hızlı bir şekilde elde edilmesi amaçlanmıştır. Sonuçta geliştirilen tekniklerle *BF* değerlerine yakın sonuçlar alınmıştır. Geliştirilen teknik, hız ve sıkıştırma oranı bakımından üst sınır değerlerine yaklaşmıştır.

Bu tekniğin başarısı, test blokları havuzunun, benzer test bloklarını aynı sınıflar içerisinde toplayan sınıf listelerine bölünmesinde kullanılan '*hash*' fonksiyonlarının etkinliğinde gizlidir. Bir referans bloğunun kendi sınıfı ve akraba sınıfları kolaylıkla tespit edilmekte ve bu listelere hızlı bir şekilde ulaşılabilmektedir.

Yöntem içerisinde hala optimize edilecek noktalar bulunmaktadır. Örneğin 64 uzunluktaki benzerlik tahminlerine göre sıralı listenin daha hızlı oluşturulması üzerinde durulabilecek önemli bir konulardan biridir.

Yöntem, renkli görüntüler ve video için de uygundur. Renkli görüntülerde yöntemin uygulanması için öncelikle *RGB* renk modunda kayıtlı olan görüntü, *YUV* veya YC_bC_r renk modlarına çevrilerek (Salomon, 2000), daha sonra elde edilen üç renk bileşeni önerilen yöntemle sıkıştırılması gerekir. C_b ve C_r bileşenleri çok fazla homojene yakın referans blokları içerirler ve çok daha fazla sıkıştırılabilirler. Renkli görüntülere bir örnek olarak, orijinal Carol görüntüsü ile Y, C_b ve C_r bileşenleri Şekil 8'de verilmiştir. Görüntünün R, G, B, Y, C_b ve C_r bileşenleri ayrı ayrı fraktal görüntü sıkıştırmaya tabi tutulmuş ve görüntülerin altında verilen değerler elde edilmiştir.

Orijinal **Y** **Cb** **Cr**

	Sıkıştırma Oranı	Boyut (*byte*)	*PSNR*	Süre (sn)	Ref. Blok Adedi			
					4×4	8×8	16×16	32×32
Bileşen *R*	27.89:1	9,401	34,160	1.08	1,556	735	211	133
Bileşen *G*	29.99:1	8,742	34,534	0.98	1,472	636	221	138
Bileşen *B*	32.66:1	8,027	33,945	0.83	1,364	539	196	152
Bileşen *Y*	31.78:1	8,250	34,515	0.94	1,324	653	210	142
Bileşen C_b	423.50:1	620	40,328	0.08	0	0	8	254
Bileşen C_r	431.16:1	609	39,524	0.09	0	0	0	256

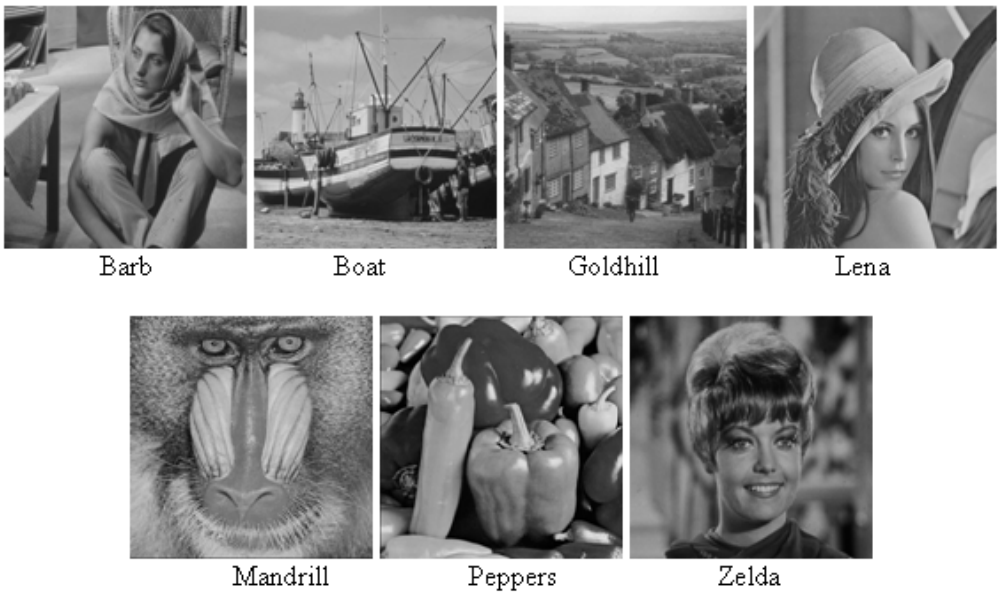
Şekil 8. 512×512 boyutlu Carol görüntüsü için üstte orijinal ve Y, C_b ve C_r bileşenleri görüntüleri, altta geliştirilen algoritmadan elde edilen değerler.

Şekil 8'de verilen değerlere göre, R, G, B ve Y bileşenleri yaklaşık aynı seviyede sıkıştırılabilirken, C_b ve C_r bileşenleri maksimum seviyede sıkışmaktadır. *PSNR* değerleri de diğerlerinden daha iyidir. Basit bir hesapla, 512×512×3 *byte* içinde saklanan örnek görüntü

8.250+620+609=9.479 *byte*'a sıkıştırıldığı görülür ki bu durumda sıkıştırma oranı yaklaşık 83:1 olmaktadır.

Fraktal görüntü sıkıştırma yöntemleri, *jpeg* tekniği veya wavelet gibi daha üstün ve yeni tekniklerle birlikte kullanıldığında her iki yöntemle elde edilen değerlerden çok daha iyi sonuçlar elde edilmektedir (Barahav v.d., 1995; Wakefield v.d., 1998; Welstead, 1999). Bu nedenle önerilen yöntem hızı sayesinde, bu tekniklere yardımcı olabilir.

Ek 1. Çalışmada Yararlanılan Test Görüntüleri



KAYNAKLAR

Barahav Z., Malah D., Karnin E. (1995): "Hierarchical Interpretation of Fractal Image Coding and Its Applications" In Y. Fisher (Editor) "Fractal Image Compression: Theory and Application" Chapte 5, 91-117, Springer-Verlag, New York, NY, USA.

Barnsley M. (1992): "Fractals Everywhere". Academic Press, San Diego, CA, USA.

Cormen T., Leiserson C., Rivest R., Stein C. (2003): "Introduction to Algortihms". MIT Press, Second Edition.

Fisher Y. (1995): "Fractal Image Compression: Theory and Application". Springer-Verlag, Newyork, USA.

Geoff D. (1998): "A Wavelet-based Analysis of Fractal Image Compression". IEEE Transactions on Image Processing, Feb. 1998.

Güngör C., Öztürk A. (2004): "A Simple Measure of Correlation for Classification of Domain Blocks in Fractal Image Compression". International Conference on Informatics, September 2004, Çeşme, İzmir, Turkey.

Jacquin A. E. (1992): "Image Coding based on a Fractal Theory of Iterated Contractive Image Transformations". IEEE Transactions on Image Processing, Vol. 1, No. 1, pp. 18-30.

Kim T., Van Dyck R.E., Miller D.J. (2002): "Hybrid Fractal Zerotree Wavelet Image Coding". Signal Processing Image Communications, Vol: 17, No:4, 347-360.

Kominek J. (1995): "Advances in Fractal Compression for Multimedia Applications".

Mandel Brot B. (1983): "The Fractal Geometry of Nature". W.H.Freeman & Co., Second Edition.

Salomon D. (2000): "Data Compression: The Complete Reference". 2nd Edition, Springer Verlag

Saupe D. (1995): "Accelerating Fractal Image Compression by Multi-Dimensional Nearest Neighbor Search" In J. A. Storer and M. Cohn (Editors), "Proceedings DCC'95 (IEEE Data Compression Conference)". 222-231, Snowbird, UT, USA, March 1995.

Wakefield P., Monro D. (1998): "Fractal Enhancement of Decompressed Images", In "Proceedings of ICIP-98 - IEEE International Conference on Image Processing". Chicago, Illinois.

Welstead S. (1999): "Fractal and Wavelet Image Compression Techniques". SPIE Press, ISBN 0-8194-3503-1.

Semboller Listesi

Δ	Referans bloğu ve test bloğuna ait sınıf numaraları arasındaki ilişki
σ_R	Referans bloğunun varyansı
σ_T	Test bloğunun varyansı
BF	'*Brute-force*' tekniği
h_i	Sınıf numarasını elde etmede kullanılan '*hash*' fonksiyonu
H	Gelişmiş teknikte bloğun sınıf numarası (1...65535)
k_R	Basit teknikte referans bloğuna ait sınıf numarası (1...16)
k_T	Basit teknikte test bloğuna ait sınıf numarası (1...16)
L	Tahminlere göre sıralı listenin boyutu
O	Karmaşıklığın ifadesinde kullanılan '*order*' fonksiyonu.
PSNR	*Peak Signa-to-Noise Ratio* değeri.
Q_n	Referans ve test blokları arasındaki korelasyonu basit bir şekilde ölçen istatistik
RGB	Görüntü kaydı için kullanılan kırmızı, mavi ve yeşil renk bileşenleri (*Red, Green, Blue*)
RMS	*Root-Mean-Square* ölçümü.
S	Bloğun standart sapması.
$V^{(r)}$	Referans bloğunun standart görüntü vektörü
$V^{(t)}$	Test bloğunun standart görüntü vektörü
$\bar{x}$	Blok ortalaması
YUV	Kırmızı, mavi ve yeşil renk bileşenlerinin farklı bir şekilde ifadesi
YC_bC_r	*YUV* renk bileşenlerinin diğer bir ifadesi