YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

VEKTÖR KUANTALAMA ile GÖRÜNTÜ SIKIŞTIRMA

Elektronik ve Haberleşme Mühendisi Nilgün DURSUNOĞLU

FBE Bilgisayar Mühendisliği Anabilim Dalında
Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Yrd. Doç. Dr. Songül ALBAYRAK

İSTANBUL, 2006

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

x_i	i. giriş örneği
w_k	k. merkez
α	Öğrenme katsayısı
S_k	k. merkeze ait örnekler kümesi
N_k	k. merkeze ait örnekler kümesinin eleman sayısı
N_c	Komşuluk fonksiyonu
η	NGas modeli için öğrenme katsayısı
H	Ödül fonksiyonu

KISALTMA LİSTESİ

SOM	Self Organizing Map
1D SOM	One Dimensioned Self Organizing Map
2D SOM	Two Dimensioned Self Organizing Map
HSOM	Hierargichical Self Organizing Map
VQ	Vector Quantization
NGas	Neural Gas
DC	Differential Coding
MSE	Mean Square Error
MIT	Massachusetts Institute of Technology
LZW	Lempel-Ziv-Welch Algorithm
JPEG	Joint Photographic Experts Group
GIF	Graphics Interchange Format
BMP	Batch Message Processor
RLE	Run Length Encoding
TIFF	Tagged Image File Format
MP3	MPEG (Moving Pictures Experts Group) Audio Layer 3

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

ÖNSÖZ

Bilgi teknolojisinin gelişmiyle verilerin depolanabilmesinin sağlanmasını takiben verinin bir ortamdan diğerine aktarımı ve bunun hızlı, verimli ve kayıpsız yapılabilmesi çabaları ortaya çıkmıştır. Bu amaca ulaşabilmek için verilerin daha küçük boyutlara getirilmesi, bununla birlikte veri kayıbının engellenmesi veya en aza indirilmesi amacıyla çeşitli sıkıştırma yöntemleri geliştirilmeye çalışılmıştır. Her geçen gün verinin depolanabilmesi ve aktarım kalitesinin arttırımı için yöntemler denenmektedir.

Bu çalışmada da görüntü sıkıştırmada vektör kuantalama uygulamaları üzerinde çalışılarak yukarıda bahsedilen çabalara bir katkı sağlama amacı güdülmüştür.

Danışmanım Sayın Yrd. Doç. Dr. Songül ALBAYRAK'a tez çalışmam boyunca beni her açıdan desteklediği, bilgisini, görüş ve deneyimlerini benimle paylaşarak, bilimsel yayınlar yapmam için beni yönlendirdiği için çok teşekkür ederim.

Sayın Prof. M. Yahya KARSLIGİL'e, yüksek lisans öğrenimim boyunca, problemlere etkin ve kısa çözümler üreten yaklaşımları ve daha pekçok şeyi bana öğrettiği için çok teşekkür ederim.

Gerek lisans ve yüksek lisans hayatıma, gerekse bilimsel çalışmalarıma görüşleri ve farklı yorumları ile katılan, bu süreçlerde karşılaştığım problemlerin çözümünde bana yardımcı olan, akademik çalışmalarda beni cesaretlendirici tavırlarda bulunan, hayatımın her anında beni destekleyen eşim Volkan DURSUNOĞLU'na çok teşekkür ederim.

Yıldız Teknik Üniversitesi'ndeki akademik hayatım boyunca ve sonrasında her zaman her konuda yanımda olan, Sayın Yrd. Doç. Dr. Banu DİRİ'ye teşekkür ederim. Görüntü işleme konusunda kendisinden çok şey öğrendiğim Sayın Yrd. Doç. Dr. Elif KARSLIGİL'e teşekkür ederim.

Araştırma Görevlisi olduğum sürece aynı odayı paylaştığım, Arş. Gör. M. Fatih AMASYALI ve Arş. Gör. Hakan HABERDAR'a görüş ve yorumlarıyla akademik hayatıma katkıda bulundukları için, değerli arkadaşlığını benden esirgemeyip her zaman yanımda olan YTÜ Elektronik ve Haberleşme Mühendisliği bölümündeki Arş. Gör. Burcu ERKMEN'e teşekkür ederim.

Hayatım boyunca desteklerini hiçbir zaman eksik etmeyen, benim için her türlü fedakarlıkta bulunan, beni yüreklendiren ve bana güvenen sevgili annem Hatice ERDEM'e ve kızkardeşim Nilüfer ERDEM'e, her zaman yanımda olan sevgili babam Musli ERDEM'e çok teşekkür ederim.

ÖZET

Bu çalışmada, bir veri sıkıştırma örneği olarak görüntü sıkıştırma işlemi iki aşamalı olarak gerçekleştirilmiş, ilk aşamada resmin renk sayısı düşürülmüş ikinci aşamada ise renk sayısı azaltılmış olan resim sıkıştırılmıştır.

Resimdeki renk sayısı düşürülürken K-Means ve SOM kuantalama algoritmaları farklı güncelleme yöntemleri kullanılarak ve hiyerarşik yapılı olarak 256 renk Lenna resmine uygulanmıştır. K-Means algoritması için neural gas, batch ve online güncelleme yöntemleri, SOM algoritması için geleneksel ve literatürde ilk defa neural gas güncelleme yöntemleri kullanılmıştır. Vektör kuantalama algoritmaları kayıplı yöntemler olduğundan bu aşamanın sonunda elde edilen renk sayısı azaltılmış resimlerden orjinal resmin bire bir elde edilmesi mümkün değildir. Uygulamada Lenna resminin renk sayısı sırasıyla 16, 64 ve 100'e indirilmiştir. Renk sayısının azaltılması sonucunda, kuantalama algoritmalarının hız ve hata oranları ölçülmüş ve karşılaştırılmıştır.

Sıkıştırma aşamasında iki yol izlenmiştir. Birincisi renk sayısı azaltılmış resme Huffman kodlamasının doğrudan uygulanması, ikincisi ise resme önce fark kodlamasının sonra Huffman kodlamasının uygulanmasıdır. Fark kodlaması ve Huffman kodlaması kayıpsız yöntemlerdir. Bir başka deyişle bu algoritmaların uygulandığı resim, tersine işlemler uygulanarak aynen elde edilebilmektedir. Bu iki yol sonucunda elde edilen sıkıştırma oranları, tüm kuantalama algoritmaları ve farklı güncelleme yöntemleri için karşılaştırılmıştır.

ABSTRACT

In this paper, an image compression application is constituted as a compression example with two steps, in the first step the number of colors of the picture is reduced, in the second step color number reduced image is compressed.

On color reducing process, k-Means and SOM quantization algorithms are applied on 256 colored Lenna picture with different updating methods and also hierarchically. For k-Means algorithm NGas, batch and online updating methods and for SOM algorithm traditional and – for the first time- NGas updating methods are used. For the reason that vector quantization algorithms are lossly methods, obtaining the same of the original picture from the color number reduced picture is not possible. In this application color number of Lenna picture is reduced to respectively 16, 64 and 100. The speed and error values occurred in the process of color number reduction are measured and compared.

Two ways are followed in compression process. First is the application of Huffman coding directly to the color number reduced picture, second is the application of Huffman coding after differential coding to the color number reduced picture. Differential coding and Huffman coding are lossless methods. In other words, the same picture can be obtained with using reverse mehods. Compression rates obtained with these two ways are compared for all quantization algorithms with different updating methods used in this paper.

1. GİRİŞ

Geçmişten günümüze her zaman veri saklama ve veri iletme ihtiyacı olmuştur. Bilgisayarların kullanılmaya başlanması ve hızla gelişen teknolojinin daha büyük diskler, daha hızlı iletişim sunmasına paralel olarak, insanoğlunun depolama ve iletim ihtiyacı da artmaktadır. Bilgisayarların günlük yaşamımızda kullanımının yaygınlaşması ve ihtiyaç duyulan bilginin her koşulda ve her an erişilebilir olması isteği, verinin çok küçük alanlarda depolanabilmesi ve hızlı bir şekilde iletilebilmesi için artan ihtiyacı karşılamada daha etkili sıkıştırma algoritmalarına her zaman ihtiyaç duyulacağını göstermektedir.

Bu çalışmada görüntü sıkıştırma uygulaması olarak tasarlanan sistem, vektör kuantalama, fark kodlama ve Huffman kodlama adımlarını içermektedir. Kuantalama algoritması olarak, k-Means, 1D SOM ve 2D SOM algoritmaları kullanılmış ve renk kuantalama yapılmıştır.

Vektör kuantalama algoritmaları veri öbekleme için çok önemli algoritmalardır. Kayıplı görüntü sıkıştırmanın vazgeçilmez bir parçası olduğu gibi sınıflandırma algoritmalarında ön işlem olarak kullanıldığı yerler de vardır.

Vektör kuantalama kayıplı görüntü sıkıştırmada resim kuantalama ya da renk kuantalama olarak karşımıza çıkmaktadır. Resim kuantalama kullanılarak görüntü sıkıştırılırken öncelikle resim 4x4 veya 8x8 bloklara ayrılır. Her blok bir giriş örneği olarak alınır ve kuantalama algoritması uygulanarak bloklar daha az sayıda yeni bloklar olarak ifade edilir. Elde edilen yeni bloklar kullanılarak resim seçilen sıkıştırma yöntemine göre sıkıştırılır. (Laha vd.,2004; Costa vd., 2003)

Renk kuantalama yapılırken ise resimde kullanılan her bir renk giriş örneği alınır ve kuantalama algoritması uygulanarak renk sayısı azaltılır. Kuantalama sonucu elde edilen renkler üzerine sıkıştırma algoritması uygulanarak resim sıkıştırılır.

Sıkıştırma adımında Huffman kodlaması basitliği ve kolay uygulanabilirliği sebebiyle tercih edilen bir yöntemdir. Sıkıştırma oranını arttırmak için sıkıştırılacak veriye Huffman kodlama öncesi fark kodlaması, JPEG'de olduğu gibi DCT (kayıplı) dönüşümü gibi yöntemler uygulanmaktadır.

Bu çalışmada kayıplı bir yöntem olan vektör kuantalama ve kayıpsız bir sıkıştırma yöntemi olan Huffman kodlama 256 renk Lenna resmi üzerine uygulanmıştır. Vektör kuantalama algoritmalarından k Means batch, online ve neural gas yöntemi ile güncellenerek (k-means batch, k-means online, k-means NGas), geleneksel yöntemlerle ve NGas yöntemiyle

güncellenen 1D ve 2D SOM algoritmaları (1D SOM Std1D, SOM NGas, 2D SOM Std, 2D SOM NGas) kullanılmıştır. Bu yöntemler aynı zamanda hiyerarşik yapıda kullanılmış ve sıkıştırma oranlarına etkileri incelenmiştir.

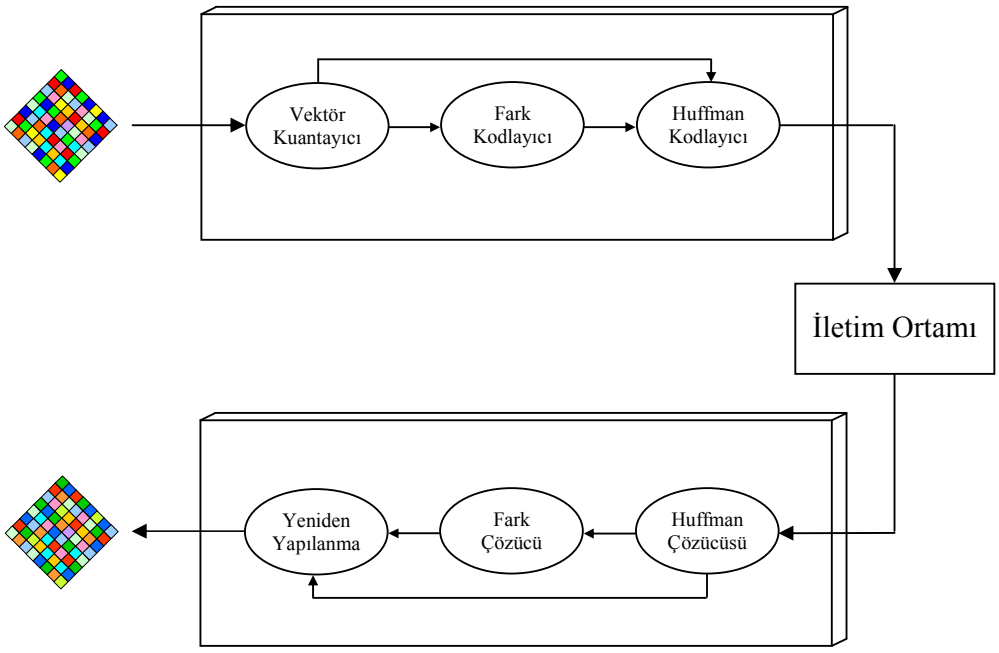
2. bölümde bu çalışmada kullanılan sistemin genel yapısından bahsedilmiştir. 3. bölümde vektör kuantalamanın ne olduğu kuantalama algoritmalarının temel adımları, 4. bölümde vektör kuantalama algoritmaları ve 5. bölümde hiyerarşik yapılı vektör kuantalama algoritmaları yer almıştır. Bu algoritmalar 256 renk Lenna resmine uygulanmış ve algoritmaların hız ve hata sonuçları 6. bölümde incelenmektedir. Sıkıştırma adımında kullanılan Huffman kodlaması 7. bölümde ve Lenna resmine uygulanmasıyla elde edilen sonuçlar 8. bölümde yer almaktadır. Son bölümde sonuçlar hakkında değerlendirme yapılmıştır. Kuantalama algoritmalarının uygulanması ile elde edilen resim sonuçları Ek 1'de yer almaktadır.

2. SİSTEMİN GENEL YAPISI

2.1 Genel

Bu çalışmada kullandığımız sistem Şekil 2.1'de görülgüğü gibi vektör kuantalayıcı, fark kodlayıcı ve Huffman kodlayıcı olmak üzere üç parçadan oluşmaktadır. Sistem girişi olarak 256 renk(8-bit) bmp resim uygulanmıştır. Vektör kuantalayıcı olarak, k-means, 1D SOM ve 2D SOM kuantalama algoritmaları geleneksel güncelleme ve NGAS güncelleme yöntemleri ile kullanılmıştır. Aynı zamanda k-means ve 1D SOM algoritmaları hiyerarşik yapıda kullanılarak vektör kuantalayıcı olarak uygulanmıştır. Vektör kuantalayıcının çıktısı olarak elde edilen yeni resim, direk ve fark kodlayıcıdan geçirilerek Huffman kodlayıcıya giriş olarak verilmiş ve bu iki durumda sistemin sıkıştırma performansı karşılaştırılmıştır.

Renkli resimlerin sıkıştırılmasında kullanılan vektör kuantalama kayıplı bir yöntemdir. Fark kodlayıcı Huffman kodlama sırasında sıkıştırma oranını arttırmak için kullanılan, kodlama yöntemlerinden biridir. Huffman kodlayıcı kayıpsız bir sıkıştırma yöntemidir. Sıkıştırılmış resmin açılması aşamasında son elde edilen sonuç vektör kuantalayıcının çıkışındaki kayıplı resim olacaktır.



Şekil 2.1 Renkli görüntü sıkıştırmada kullanılan metodlar [2]

3. VEKTÖR KUANTALAMA

Vektör kuantalama, depolanmak ya da iletilmek istenen verinin, kendisini en iyi şekilde yansıtabilen daha az miktarda veriyle ifade edilmesidir. En iyi şekilde yansıtan ifadesinde anlatılmak istenen, verinin miktarını azaltırken oluşacak kaybın (bozulmanın) en az olduğu veri kümesini bulabilmektir(Duda vd, 2001)[1].

3.1 Vektör Kuantalamanın amacı

Aynı merkeze ait, giriş uzayı vektörlerinin birbirine benzerliği yüksek olmalı, farklı merkezlere ait giriş uzayı vektörleri birbirinden olabildiğince farklı olmalıdır. Benzerliği uzaklık ile yorumlayacak olursak, vektör kuantalama algoritmaları, aynı merkezin örnekleri birbirine yakın, farklı merkezlerin örnekleri birbirinden uzak olacak şekilde merkezlerin yerleri belirlemeyi amaçlar (Duda vd,2001).

3.2 Vektör Kuantalamanın Temel Prensibi

Vektör kuantalamanın temel prensibi, sürekli girdi uzayı için, sürekli girdi uzayını, bilgi kaybını minimum tutacak şekilde ayrık çıktı uzayında yansıtmaktır (Şekil 3.1 (a)). Ayrık girdi uzayı için ise, Q özellik içeren P vektörden oluşan bir veritabanını, N<P koşulu ile, Q özellik içeren N vektörden oluşan veritabanı olarak ifade etmektir[1] (Şekil 3.1 (b)).

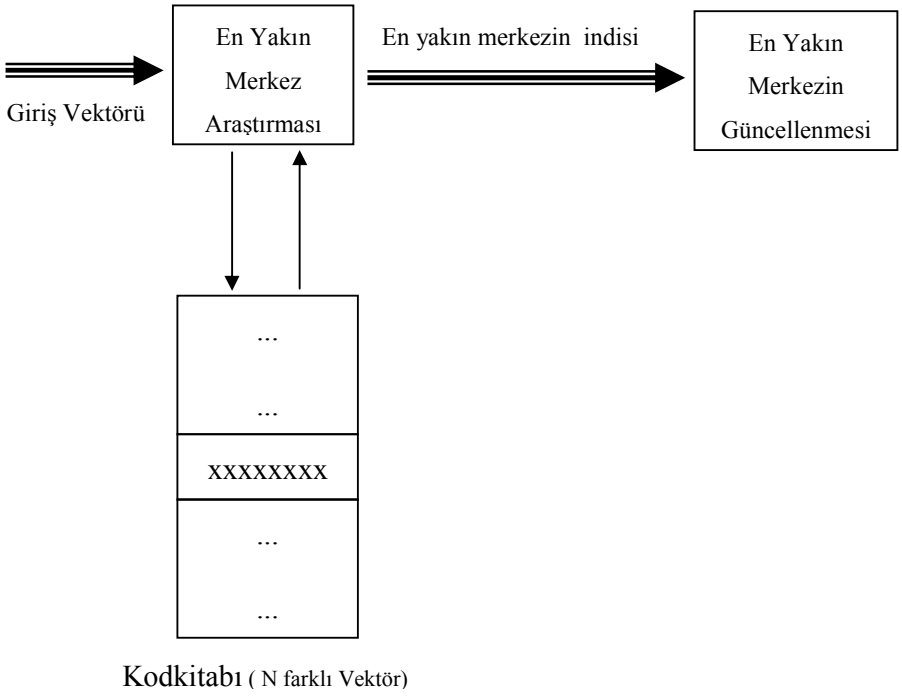
Şekil 3.1 (a) : sürekli girdi uzayının kuantalanması (b) : ayrık girdi uzayının kuantalanması

3.3 Vektör Kuantalama Algoritmalarının Temel Yapısı

Vektör kuantalama algoritmalarının temel yapısı, giriş uzayını kendisini en iyi şekilde ifade edecek belli sayıdaki çıkış kümesi ile belirlemeye dayanmaktadır. Bu, girişi gruplandırmak olarak da tanımlanabilir. Öncelikle giriş uzayının kaç adet çıkış ile ifade edileceği belirlenir. Çıkış sayısı ne kadar büyük seçilirse çıkıştaki bozulma o oranda daha düşük olmaktadır.

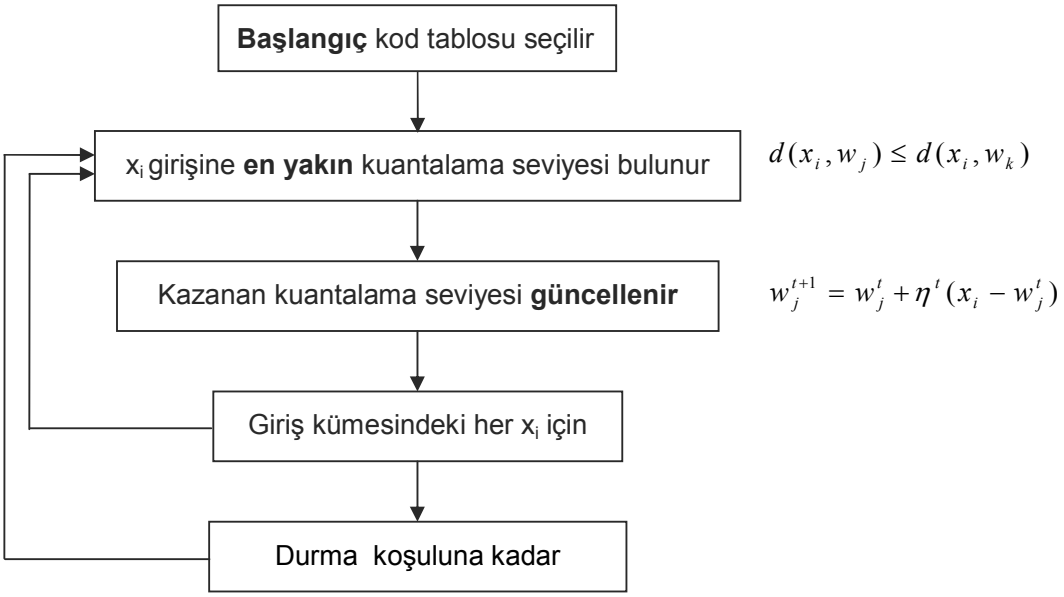
Literatürde her bir çıkış -kuantalama seviyesi-, kodvektör; bu çıkışlar kümesine ise kod tablosu adı verildiği gibi, her bir çıkışın merkez (k Means) veya nöron (SOM) olarak da adlandırıldığı görülmektedir. Giriş uzayından seçilen bir giriş vektörü için kuantalamanın temel yapısı Şekil 3.2'de görüldüğü gibi olmaktadır.[1]

İlk adımda çıkış olarak ifade edilecek kuantalama seviyelerine ilkdeğer ataması yapılmalıdır. Daha sonrasında giriş vektörüne en yakın kuantalama seviyesi belirlenir. Kullanılan vektör kuantalama algoritmasına göre en yakın kuantalama seviyesi veya en yakın seviye ile birlikte komşuları güncellenir.



Şekil 3.2 Vektör kuantalamanın temel prensibi [1]

Vektör kuantalama algoritmalarının adımları Şekil 3.3'te gösterildiği gibidir. Algoritmanın adımları olan kuantalama seviyelerine ilk değer atama, en yakın seviyenin hesaplanması ve kuantalama seviyelerinin güncellenmesi için farklı yöntemler geliştirilmiştir. Bu adımlar için kullanılacak metodlar kuantalama başarısını doğrudan etkilemektedir. [1]



Şekil 3.3 Vektör kuantalamanın temel adımları [1]

3.3.1 İlk Değer Ataması

Kuantalama seviyelerine ilk değer ataması için rasgele değerler kullanılabilir, giriş kümesinden örnekler seçilebilir veya giriş kümesi işlemden geçirilerek elde edilen sonuçlar ilk değer olarak kullanılabilir. Değer atama için kullanılacak yöntem kuantalama seviyelerinin yakınsama hızını ve sonucun hata oranını doğrudan etkilemektedir.

3.3.2 Vektör Kuantalama Algoritmalarında Kullanılan Güncelleme Kuralları

Çoğu Vektör Kuantalama algoritmasında, sadece kazanan merkez güncellenir. Bu şekilde, birbirine çok yakın iki merkezi ayırmak zordur. Kazanan merkezin yanı sıra diğer merkezleri de güncellemek bu zorluğu ortadan kaldıracaktır. Aynı zamanda diğer merkezlerin güncellendiği algoritmalar Vektör Kuantalamada yakınsamayı hızlandırmaktadır[12].

3.3.3 Durma Koşulu

İlk değer ataması yapıldıktan sonra giriş kümesindeki her bir örnek için kazanan merkez hesaplanır ve güncellenme işlemi yapılır. Merkez belirleme ve güncelleme işleminin bütün örnekler için tekrarlanmasına epok denir. Bu işlemin algoritma içerisinde kaç kez tekrarlanacağı (döngü sayısı) sisteme giriş olarak verilir. Her döngü sonrasında durma şartının sağlanıp sağlanmadığı kontrol edilir. Durma şartı merkez bulma ve güncelleme işlemlerinin

giriş kümesindeki her bir örnek için döngü sayısı kadar tekrarlanması olabileceği gibi, yakınsama miktarı da olabilir. Yakınsama; kuantalama seviyelerinin değerlerinin değişme oranı veya kazanan merkeze en yakın örneklerin sayısı gibi değişik koşullar olabilir. Yakınsama miktarı sisteme giriş olarak verilen değer (epsilon) kadar olduğunda veya döngü sayısına ulaşıldığında durma şartı sağlanmış olur.

4. VEKTÖR KUANTALAMA ALGORİTMALARI

Bu bölümde, temel adımları bir önceki bölümde anlatılan vektör kuantalama algoritmaları anlatılmıştır.

4.1 K –Means Algoritması

K means algoritması adından da anlaşılabileceği gibi giriş uzayını belirli sayıda (k adet) merkezle (kuantalama seviyesi) ifade etmeye çalışan bir yöntemdir. Merkezlere ilk değer ataması yapıldıktan sonra merkez değerlerinin güncellenmesi için iki farklı yöntem kullanılılır. Birinci yöntemde (batch metod) giriş kümesindeki her bir örneğin hangi merkeze yakın olduğu hesaplanır. Aynı merkeze yakın olan örneklerin ortalaması alınarak merkezin değeri güncellenmiş olur. Durma koşulu sağlanana kadar bu işlem tekrar edilir. İkinci yöntemde (online metod) giriş kümesinden bir örnek seçilir ve bu örneğin merkezlere olan uzaklığına bakılır. Örneğin en yakın olduğu merkez bulunarak bu merkezin değeri güncellenir. Her bir örnek için bu işlem tekrarlanır. Merkez değeri güncellenirken merkezle örnek arasındaki mesafe değeri her epokta azalan bir öğrenme katsayısıyla çarpılarak kullanılır. Bu sayede ilk adımlarda merkezlerin yer değiştirmesi büyük miktarlarda olurken zamanla yer değiştirme azalır ve merkezler yakınsar. Durma şartı sağlanıncaya kadar her örnek için işlem tekrarlanır. Örneklerin işlenme sırası her epokta aynı olacağı gibi rasgele sırada da olabilir (Duda vd,2001).

Bu çalışmada, merkezlerin güncellenmesinde tüm merkezlerin güncellendiği Neural Gas güncelleme yöntemi de kullanılmış ve algoritma NGas k-means olarak isimlendirilmiştir.

4.1.1 Batch k-means Algoritması

Giriş uzayı d boyutlu, x_i: i. girdi; w_k: k. merkez; $d(x_i, w_k)$: i. girdi'nin k. merkeze olan uzaklığı olmak üzere,

1- *İlk Değer Atama.* Kuantalama seviyelerine(K adet merkez) ilk değer ataması yapılır.
2- For e = 1 to epoch do
   a. For i = 1 to n_input do
      i. Giriş kümesinden bir örnek alınır, xi ∈ X;
      ii. x_i örneğine en yakın kuantalama seviyesi bulunur.

$$d(x_i, w_k) \le d(x_i, w_j),\ 1 \le i \le N \quad 1 \le j, k \le K \text{ ise } x_i \in S_k(t) \tag{4.1}$$

      iii. x_i örneği ait olduğu merkezin sınıfına S_k, dahil edilir. S_k k. Merkeze ait olan örnekler kümesidir.
   b. Tüm merkezler kendisine ait örneklerin ortalaması alınarak güncellenir. N_k $S_k(t)$'deki örnek sayısıdır.

$$w_k(t+1) = \frac{1}{N_k} \sum_{x \in S_k(t)} x \tag{4.2}$$

4.1.2 Online k-means Algoritması

Giriş uzayı d boyutlu, x_i: i. girdi; y_k: k. merkez; α : öğrenme katsayısı; $d(x_i, y_k)$: i. girdi'nin k. merkeze olan uzaklığı olmak üzere,

1- *İlk Değer Atama.* Kuantalama seviyelerine(K adet merkez) ilk değer ataması yapılır.
2- For e = 1 to epoch do
   a. For i = 1 to n_input do
      i. Giriş kümesinden bir örnek alınır, xi ∈ X;
      ii. x_i örneğine en yakın kuantalama seviyesi bulunur.

$$d(x_i, w_k) \le d(x_i, w_j), \quad 1 \le i \le N \quad 1 \le j, k \le K \tag{4.3}$$

      iii. Kazanan merkez güncellenir(kazanan merkez, x_i' ye yaklaştırılır)

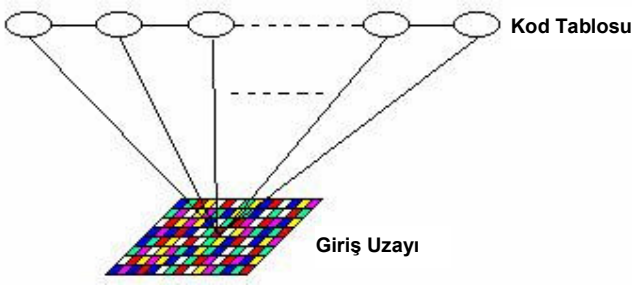
$$w_k(t+1) = w_k(t) + \alpha(x_i - w_k(t)) \tag{4.4}$$

4.2 Kendi kendine organize olan özellik haritası (Self-Organizing Feature Map-SOFM)

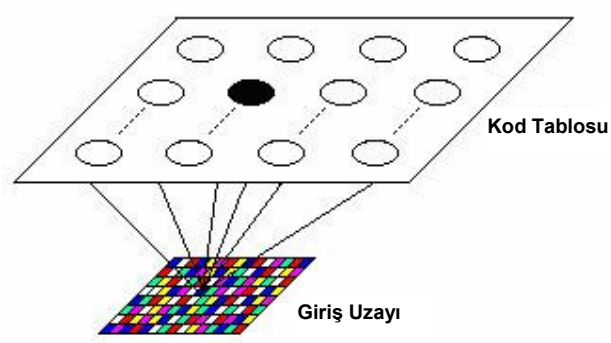
Kendi kendine organize olan sistemler, Self Organizing Future Maps (SOFM) olarak adlandırılan yapay sinir ağlarının özel bir sınıfıdır. Kod tablosundaki vektörler arasında 1 boyutlu veya 2 boyutlu komşuluk tanımlaması yapılarak 1D-SOM ve 2D-SOM ağları oluşturulur. Bu ağlar yarışmalı öğrenmeye dayanırlar (Haykin, 1998; Kohonen , 1989).

Vektör kuantalama algoritmalarının dezavantajı, performansının ve yakınsama hızının başlangıç kod tablosu seçimine bağlı olmasıdır. SOM algoritmasının, başlangıç kod tablosu seçiminden bağımsız olarak güvenilir, daha hızlı ve ölü seviyelerden kaçınan etkili bir algoritma olduğu örneklerle gösterilmiştir (Luo vd, 2003). SOM'da kod tablosu harita ve kod tablosunun her bir elemanı ise nöron olarak adlandırılır.

Algoritmanın temeli, giriş uzayında birbirine yakın olan vektörlerin, çıkışlarının da birbirine yakın olacağı görüşüne dayanmaktadır. Öğrenme işleminde, bir özellik haritası hesaplanırken, haritanın doğruluğu SOFM algoritmasının iterasyon sayısıyla doğrudan ilişkilidir. Buna ek olarak, haritanın başarısı, algoritmanın ana parametrelerinin seçilmesine (öğrenme oranı parametresi α ve komşuluk fonksiyonu N_c) kritik olarak bağlıdır. Maalesef, bu parametrelerin seçilmesinde temel bir kural yoktur, genellikle deneme ve yanılma yöntemiyle bulunur. Öğrenme oranı parametresi α(n), nöronların güncellenmesi için kullanılır [2]. Şekil 4.1 ve Şekil 4.2'de, 1D-SOM ve 2D-SOM modelleri gösterilmektedir.



Şekil 4.1 1D SOM modeli



Şekil 4.2 2D SOM modeli

Bu çalışmada, nöronların güncellenmesinde Neural Gas güncelleme yöntemi de kullanılmış ve Ngas 1D SOM, Ngas 2D SOM olarak isimlendirilmiştir. Neural Gas güncelleme yöntemi için nöronlar girişe yakınlıklarına göre değil, kazanan nörona komşuluk derecelerine göre güncellenmişlerdir.

4.2.1 SOM Algoritması

Giriş uzayı d boyutlu, x_i: i. Girdi; y_k: k. merkez; α : öğrenme katsayısı; $d(x_i, y_k)$: i. girdi'nin k. merkeze olan uzaklığı olmak üzere,

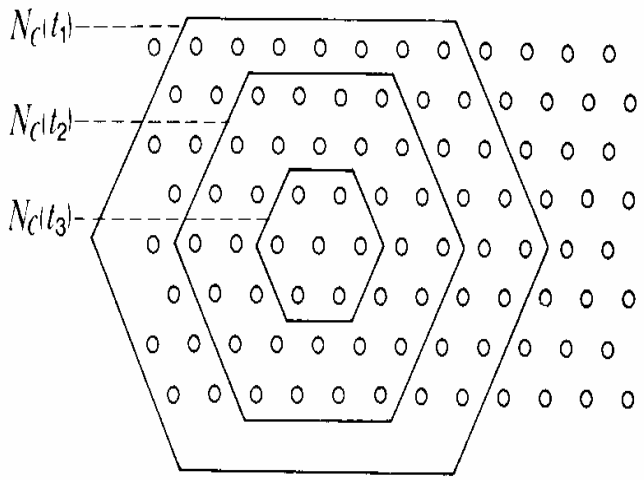
1- *İlk Değer Atama.* Kuantalama seviyelerine(nöron) ilk değer ataması yapılır.
2- For e = 1 to epoch do
   a. For i = 1 to n_input do
      i. Giriş kümesinden bir örnek alınır, xi ∈ X;
      ii. x_i örneğine en yakın nöron bulunur.

$$d(x_i, w_k) \le d(x_i, w_j), \quad 1 \le i \le N \quad 1 \le j, k \le N \tag{4.5}$$

      iii. Tüm nöronlar, kazanan nörona komşuluklarına göre güncellenir.

$$w_k(t+1) = \begin{cases} w_k(t) + \alpha(x_i - w_k(t)) & \text{eger } i \in N_c(t) \\ w_k(t) & \text{eger } i \notin N_c(t) \end{cases} \tag{4.6}$$

Algoritmada kullanılan komşuluk fonksiyonu N_c'nin belirlediği alan zaman içinde daraltılmalıdır. Şekil 4.2'de 2 boyutlu SOM için komşuluk alanının değişimi görülmektedir. t_1 anında büyük olan komşuluk fonksiyonu t_2, t_3 zamanlarında gittikçe daraltılmaktadır. Şekil 4.3'de altıgen formunda bir komşuluk ilişkisi gösterilmektedir. Ancak uygulamada komşuluk için kare formunda bir ilişki kullanılmıştır.



Şekil 4.3 N_c komşuluk fonksiyonunun 2D SOM'da zamanla değişimi

Bu çalışmada 2 boyutlu SOM için, bir nöronun komşu sayısı 8 olarak alınmıştır.

4.3 Neural GAS Algoritması

"Neural Gas" (NGas) Model (Martinez vd, 1993), kuantalama seviyelerinin döngülü olarak ayarlandığı, her döngüde tüm kuantalama seviyelerinin giriş örneğine benzerliklerine göre güncellendiği etkili fakat basit bir güncelleme prensibine dayanmaktadır (Ancona vd, 1997) . NGas algoritmasında her iterasyonda analitik maliyet fonksiyonunun bir stokastik bayır inişi (gradient descent) gerçeklenir (Zhang vd, 1998). Güncelleme adımı esnasında yerel bir minimuma sınırlı kalmayı engellemek için genel bir yaklaşım, kazanan merkezleri güncellemenin yanı sıra, seçilen giriş vektörüne yakınlıklarına dayanan bütün merkezleri etkileyen bir softmax güncelleme kuralı uygulamaktır (Martinez vd, 1993; Ancona vd, 1997). Genişletilmiş bir k-means öbekleme yöntemi olarak değerlendirebilir (Martinez vd, 1993). Literatürde farklı uygulamaları vardır (Zhang vd, 1998; Qin ve Suganthan, 2004).

Bu çalışmada Neural Gas güncelleme yöntemi k-Means öbekleme (Duda vd,2001; Linde vd, 1980) ve SOM (9;10;Albayrak, 2002) vektör kuantalama algoritmalarına uygulanmış ve 256

renk renkli görüntülerde renk sayısı 16, 64 ve 100'e indirilerek sistemin başarısı klasik metodlarla karşılaştırılmıştır. K-means öbekleme'de tüm merkezler giriş vektörüne yakınlıklarına dayanan softmax güncelleme kuralı uygulanarak (Martinez vd, 1993; Ancona vd, 1997), 1 ve 2 boyutlu SOM'da ise tüm merkezler, kazanan merkeze komşuluk derecelerine dayanan softmax adaptasyon kuralı uygulanarak güncellenmiştir.

NGas modelinde, güncelleme sırasında, kuantalama seviyelerinin giriş vektörüne uzaklıkları sıralanmakta ve her seviyenin giriş vektörüne kaçıncı en yakın seviye olduğu bilgisi kullanılmaktadır (Martinez vd, 1993; Ancona vd, 1997). Bu modelin SOM'a uygulanması sırasında, kuantalama seviyelerinin giriş merkezine yakınlıklarını bulmak yerine, kazanan merkeze göre komşuluk seviyeleri kullanılmıştır. Bu şekilde sıralama için harcanacak süre kazanılmış olmaktadır. Öğrenme katsayısı $\eta \in [0,1]$, H() ödül fonksiyonu. η and λ zamanla azaltılmalıdır.

$$\eta^{t} = 1/\sqrt{e+1} \quad ; \quad H^{t}(j) = e^{-h(j)/\lambda^{t}} \tag{4.7}$$

4.3.1 NGas k-means Algoritması :

Giriş uzayı d boyutlu olmak üzere,

1- *İlk Değer Atama.* Kuantalama seviyelerine ilk değer ataması yapılır
2- For e = 1 to epoch do
   a. For i = 1 to n_input do
      i. Giriş kümesinden bir örnek alınır, xi ∈ X;
      ii. x_i örneği ile tüm quantalama seviyeleri arasındaki uzaklıklar hesaplanır.
         1. For j = 1 to n_output do

$$d(x_i, w_j) = \|x_i - w_j\| = \sqrt{\sum_{k=1}^{d} (x[i,k] - w[j,d])^2} \tag{4.8}$$

      iii. Uzaklıkların indisleri, uzaklıklar en küçük değerden en büyük değere doğru yer alacak şekilde sıralanır. İndisler n_output boyutlu *h* dizisine atılır, *h(j)* w_j'nin girişe kaçıncı en yakın kuantalama seviyesi olduğunu gösterir.
      iv. Kuantalama seviyeleri, sıralı sıralı dizideki pozisyonlarına göre güncellenir.

$$w_j^{t+1} = w_j^{t} + \eta^{t} H^{t}(j).\left(x_i - w_j^{t}\right) \tag{4.9}$$

4.3.2 1D-2D SOM NGas Algoritması :

Giriş uzayı d boyutlu olmak üzere,

1- *İlk Değer Atama.* Kuantalama seviyelerine ilk değer ataması yapılır
2- For e = 1 to epoch do
   a. For i = 1 to n_input do
      i. Giriş kümesinden bir örnek alınır, xi ∈ X;
      ii. x_i örneğine en yakın kuantalama seviyesi bulunur.

$$d(x_i, w_k) < d(x_i, w_j) \quad j = 1,2,..n\_output \tag{4.10}$$

$$d(x_i, w_j) = \left\| x_i - w_j \right\| = \sqrt{\sum_{k=1}^{d} (x[i,k] - w[j,d])^2} \tag{4.11}$$

iii. Kuantalama seviyeleri, w_k'ya komşuluk derecelerine göre güncellenir, *h(j)* w_j'nin w_k'ya kaçıncı komşulukta olduğunu gösterir.

$$w_j^{t+1} = w_j^{t} + \eta^{t} H^{t}(j).\left(x_i - w_j^{t}\right) \tag{4.12}$$

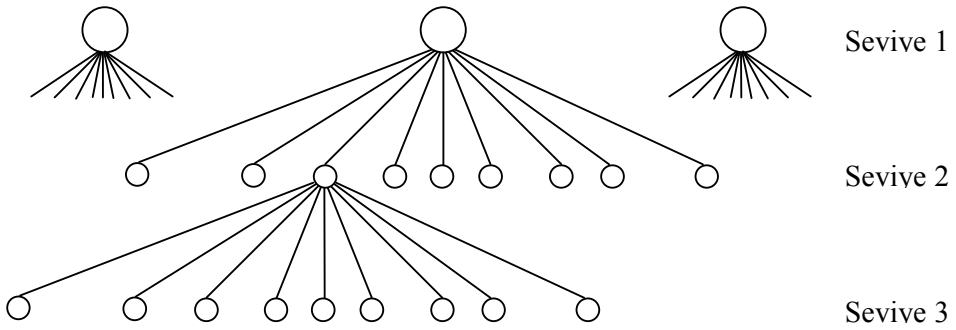
5. HİYERARŞİK VEKTÖR KUANTALAMA

Hiyerarşik yapılı vektör kuantalama algoritmaları, seviyelerden ve bu seviyelerde bulunan vektör kuantalama düğümlerinden oluşur. Her düğüm bir önceki seviyede bağlı olduğu düğümün bölgelere ayırdığı çıkış uzayının bir bölgesini, kendisinin yeni giriş uzayı olarak kullanır. Başka bir deyişle hiyerarşik vektör kuantalama algoritmaları, giriş uzayını her seviyede bölgelere ayırır ve bir sonraki seviyede bu bölgeleri giriş uzayı olarak kullanır. Bu algoritmaları kullanmaktaki temel amaç, kuantalama süresinin azaltılmasıdır. (Endo vd,2000; Barballho vd, 2001; Luo vd, 2003)

Başlangıçta giriş kümesi büyük, çıkış kümesi ise elde edilmek istenen çıkış kümesinden küçüktür. İlerleyen seviyelerde düğümlerin giriş kümesi azaltılır. (Endo vd,2000)

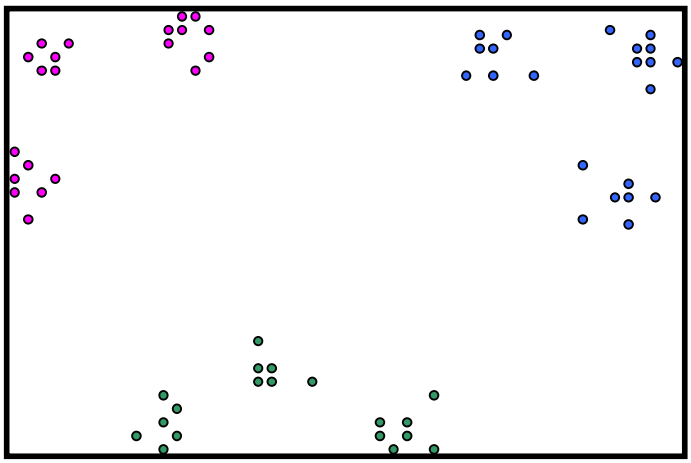
İlk seviyede başlangıç düğümü, tüm giriş kümesini kullanmaktadır. Bu giriş kümesi için vektör kuantalama algoritması çalıştırılır ve merkezlerin değerleri belirlenir. Sonraki seviyede, her bir merkeze ait olan giriş değerleri için bu giriş değerlerini giriş kümesi olarak kullanan vektör kuantalama algoritması çalıştırılır. Ve yeni merkezler belirlenir. Sonraki seviyelerde aynı adımlar izlenir. (Endo vd,2000; Barballho vd, 2001; Luo vd, 2003)

Hiyerarşik vektör kuantalamayı her dalı bir vektör kuantalama yöntemi olan bir ağaç gibi düşünebiliriz. Algoritmanın ilk aşamasında ağacın kaç seviye olacağı, her seviyede kullanılacak vektör kuantalama algoritması ve bu algoritmalardaki merkez sayısı belirlenmelidir. Şekil 5.1 de ağacın genel yapısı gösterilmiştir. Düğümler kuantalama algoritmasını ifade eder ve her düğüm kendinden önceki düğümün ürettiği çıkışın bir bölümünü kullanır. Seviyelerde farklı kuantalama algoritması kullanılabilir ve her seviyede farklı sayıda çıkış üretilebilir. Bu projede iki seviyeli bir hiyerarşi kullanılmışır. (Barballho vd, 2001)



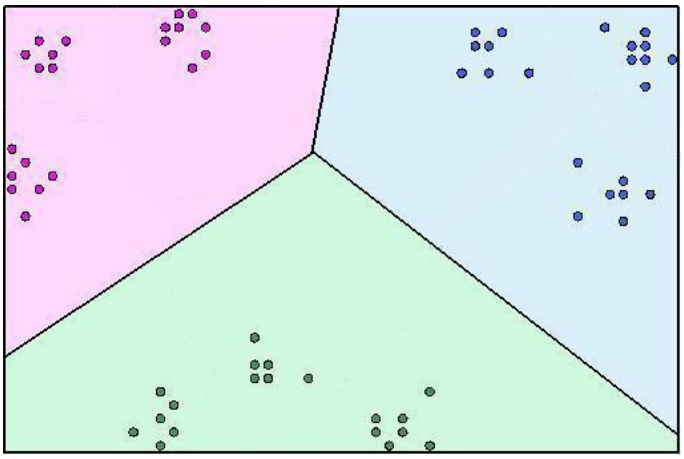
Şekil 5.1 Hiyerarşik VQ modeli

Giriş uzayı bir resim ve resimdeki renk değerleri iki boyutlu olarak Şekil 5.2 deki gibi olsun. Hiyerarşik vektör kuantalama algoritması kullanarak resmimizi 9 renk ile ifade etmeye çalışalım. Algoritmanın ilk adımı olarak ağacın seviyesi, seviyelerde kullanılacak yöntemler ve seviyelerdeki çıkış sayısı belirlenmelidir. Seviye sayısı iki ve her seviyede elde edilecek çıktı sayısı üç olsun.



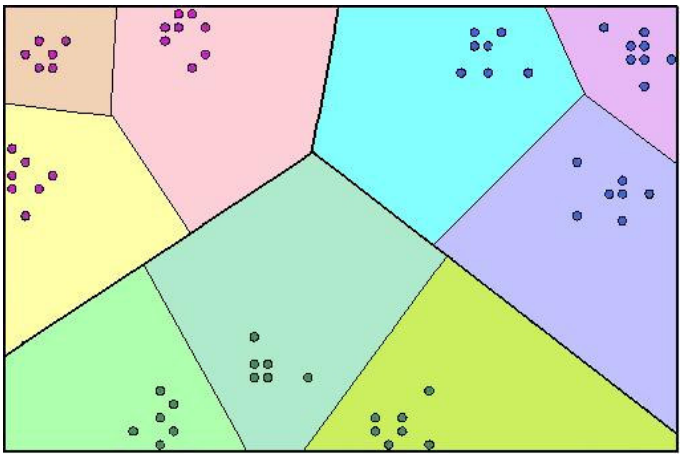
Şekil 5.2 Örnek giriş uzayı

İlk seviyede giriş uzayı olarak resmin tamamı kullanılır. İlk seviyenin çalışması sonunda resim ifade edilebileceği üç temel renge ayrılır. Şekil 5.3 te giriş uzayının bölgelere ayrılışı gösterilmiştir. Seviye sonunda giriş uzayı üç ana bölgeye ayrılmıştır ve bir sonraki seviyede bu üç bölge giriş olarak kullanılır.



Şekil 5.3 Örnek giriş uzayının 1. seviyede temel bölgelere ayrılması

İkinci seviyede giriş uzayı olarak ilk seviyede üretilen bölgeler kullanılır. Bu sayede seviyeler arttıkça giriş uzayı küçülmüş olur ve bu sistemin hızlanması sağlanır. Bu seviyenin sonucu olarak alt bölgeler oluşur, temel renkler tonlarına ayrılır. İlk seviyenin oluşturduğu üç bölge alt bölgelere ayrılarak dokuz bölge elde edilir. Resim işlemlerin sonucunda dokuz renk ile ifade edilmiş olur. Örnek giriş uzayının ayrıldığı dokuz bölge Şekil 5.4 deki gibidir.



Şekil 5.4 Örnek giriş uzayının 2. seviyede alt bölgelere ayrılması

5.1.1 HSOM Algoritması

Hiyerarşik yapılı vektör kuantalamanın temel yararı, eğitim ve kodlama fazlarında her iterasyonda kazanan merkezin bulunması için harcanan hesaplama eforunu azaltmasıdır. Hiyerarşik vektör kuantalamaya örnek olarak hiyerarşik SOM algoritması anlatılacaktır. Hiyerarşik SOM, geleneksel SOM'un genişletilmiş bir halidir. Her düğümü, belirlenmiş veri setiyle eğitilen bir SOM olan ağaç yapısı tanımlanır. 1. seviyedeki harita, tüm veri kullanılarak eğitilir. Her ağaç seviyesindeki eğitimin bitiminden sonra, haritanın merkezleri, kendilerinin işaret ettikleri verileri giriş kümesi olarak alan, yeni yavru haritalar oluşturur. HSOM'un yapısı öğrenmenin başlangıcından önce belirlenir ve eğitim yukardan aşağıya doğru sırayla gerçekleştirilir. Şekil 5.1'de 3 seviyeli 1 boyutlu HSOM yapısının örneği görülmektedir.(Barballho vd, 2001)

1.adım:İlk değer atama – Bu adımda ağacın yapısı belirlenir. Ağacın kaç seviye olacağı, ve her seviyedeki SOM düğümünün kaç nöron içereceğine karar verilir. Her bir SOM düğümünün başlangıç nöron değerleri atanır.

2 .adım: seviye = 1 için tüm eğitim seti girdi olarak alınarak SOM algoritması çalıştırılır.

3. *adım:* Düğümdeki her nöron için yavru düğümler oluşturulur. seviye bir arttırılır.

4. *adım:* Bir önceki seviyede bağlı olduğu nöronu merkez alan örnekler giriş alınarak SOM algoritması çalıştırılır.

5. *adım:* Belirlenen seviye sayısına ulaşana kadar 3. adıma geri dönülür.

6. VEKTÖR KUANTALAMA METODLARININ HIZ VE DOĞRULUK SONUÇLARI

Bu bölümde, grafiksel gösterimde kullanılan hata ve hız değerleri Lenna resminin işlenmesi ile elde edilmiştir. Bu çalışmada, vektör kuantalama algritmalarındaki ilk değer ataması iki farklı yöntemle yapılmıştır. Merkezlere atanan ilk değerler ya 0-255 aralığından rasgele ya da resimde kullanılan renklerden seçilerek yapılmıştır. Resimden seçim yapılırken, resim paletindeki renkler, ilk renkten başlanarak belli aralıklarla merkezlere atanmışlardır. Bahsedilen aralık "kullanılan renklerin sayısı / merkez sayısı" şeklinde belirlenmiştir.

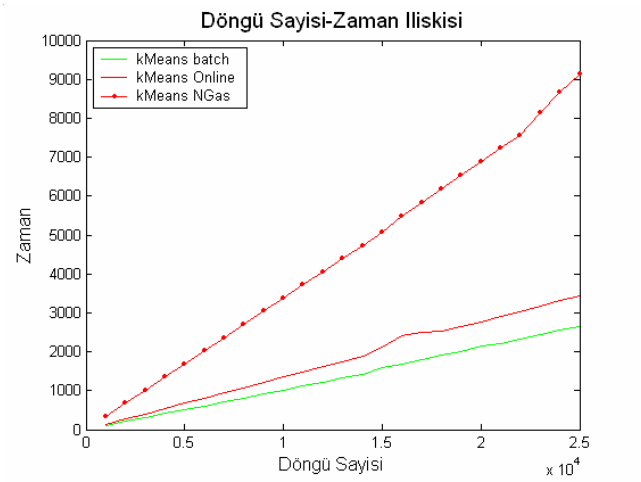
İlk değer atamasının rasgele yapıldığı durumlar için algoritmaların isimlerinin sonuna "R" kısaltması eklenmiştir. "R" eki bulunmayan algoritmalarda ilk değer ataması resimden seçilerek yapılmış anlamına gelmektedir.

6.1 Algoritmaların Hız Sonuçları

Hızların karşılaştırıldığı grafiklerde yatay eksen döngü sayısını, dikey eksen ise zamanı göstermektedir. Döngü sayısı 1 ile 25.000 arasında değişmektedir. Zamanın birimi ise 10^{-2} saniye olarak alınmıştır. Zamanın değişimini daha iyi gözlemleyebilmek için, algoritma 1.000 döngü aralıklarla çalıştırılmış ve çalışma süresi ölçümü yapılmıştır.

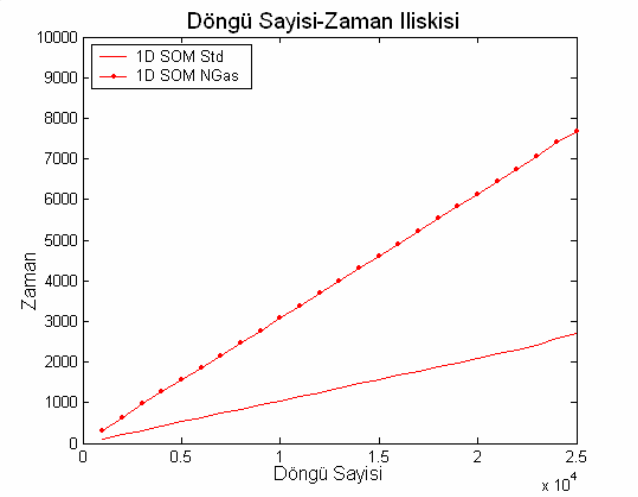
6.1.1 VQ Algoritmalarının Hızlarının Karşılaştırılması

K-Means algoritmasının, üç farklı güncelleme yöntemi kullanıldığında hızının değişimi Şekil 6.1'de görülmektedir. Elde edilen sonuçlar, k-Means algoritmasında batch güncelleme yöntemi kullanıldığında, diğer güncelleme yöntemlerine göre daha hızlı olduğunu göstermiştir.

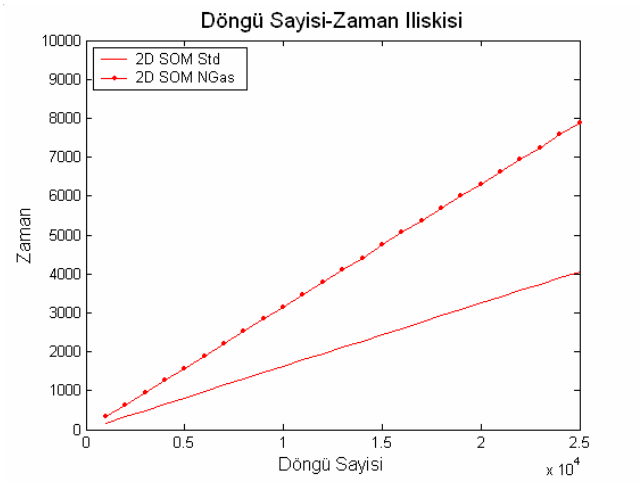


Şekil 6.1 k-Means algoritmasının batch, online ve NGas güncelleme yöntemleri kullanıldığında, döngü sayısına göre sürelerinin değişimi

1D SOM ve 2D SOM algoritmaları için iki farklı güncelleme yöntemi kullanılmış ve bu yöntemlerin algoritmaların hızlarını nasıl etkilediği Şekil 6.2 ve Şekil 6.3'te gösterilmiştir. NGas güncelleme yönteminin, algoritmaların hızını yavaşlattığı görülmüştür.

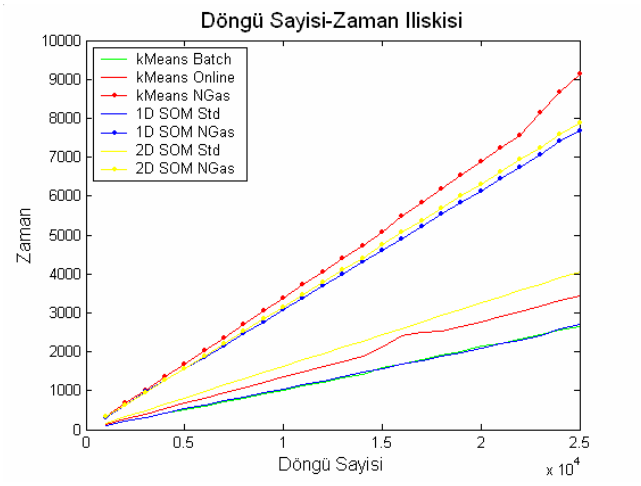


Şekil 6.2 Bir boyutlu SOM algoritmasının geleneksel ve NGas güncelleme yöntemleri kullanıldığında, döngü sayısına göre sürelerinin değişimi



Şekil 6.3 İki boyutlu SOM algoritmasının geleneksel ve NGas güncelleme yöntemleri kullanıldığında, döngü sayısına göre sürelerinin değişimi

Şekil 6.4'te farklı güncelleme yöntemleri için, vektör kuantalama yöntemlerinin döngü sayısına göre hız değişimini gösteren bir grafik görülmektedir. Grafikte NGas güncelleme yöntemi kullanılan algoritmaların sürelerinin, geleneksel güncelleme yöntemi kullanılan algoritmalara göre daha uzun olduğu gözlemlenmiştir. Batch k-Means algoritması ile geleneksel güncelleme yöntemini kullanan SOM algoritmasının döngü sayısı – zaman grafiği hemen hemen aynı olmuştur.

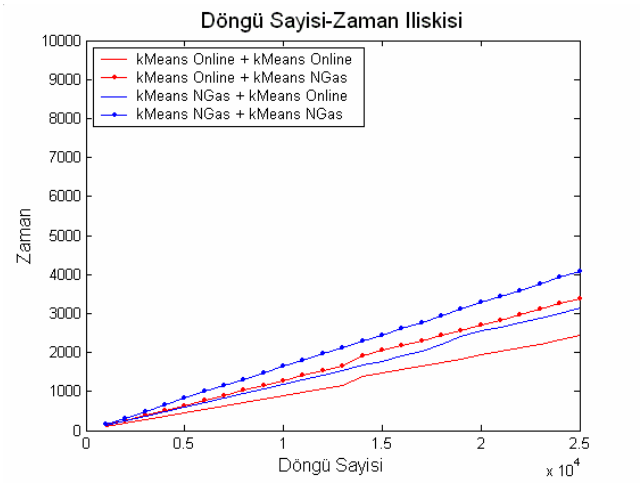


Şekil 6.4 Vektör kuantalama algoritmalarının farklı güncelleme yöntemleri kullanıldığında, döngü sayısına göre sürelerinin değişimi

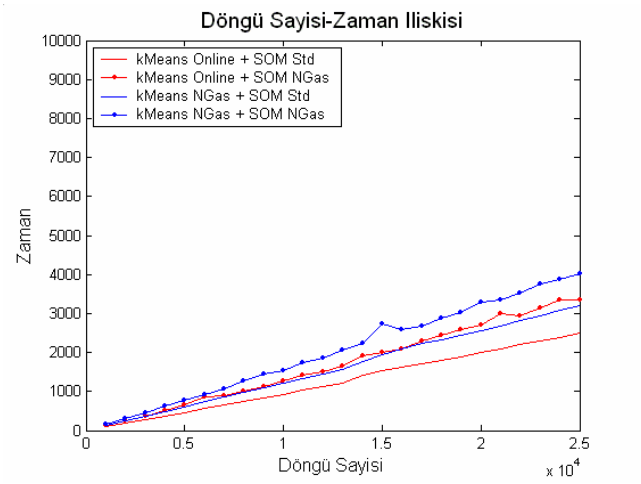
6.1.2 Hiyerarşik Yapılı VQ Algoritmalarının Hızlarının Karşılaştırılması

Şekil 6.5'te hiyerarşik k-means algoritmasının, her seviyede farklı güncelleme yöntemleri

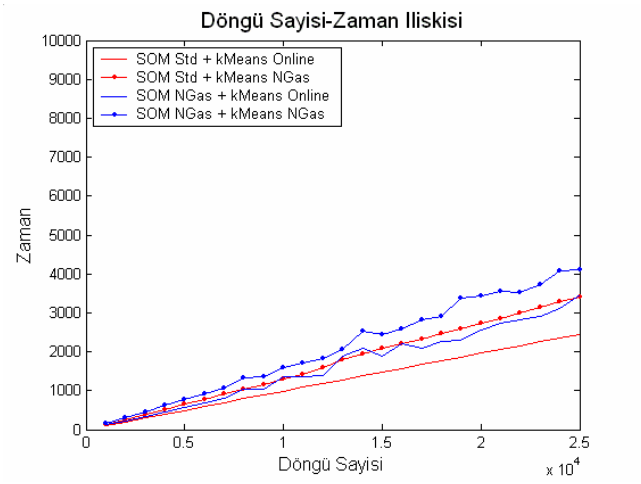
kullanıldığında, hızın, döngü sayısına göre değişimi görülmektedir. Hiyerarşik VQ yapısında da, VQ'da olduğu gibi, NGas güncelleme yönteminin kullanıldığı algoritmaların, hızı düşürdüğü görülmüştür.



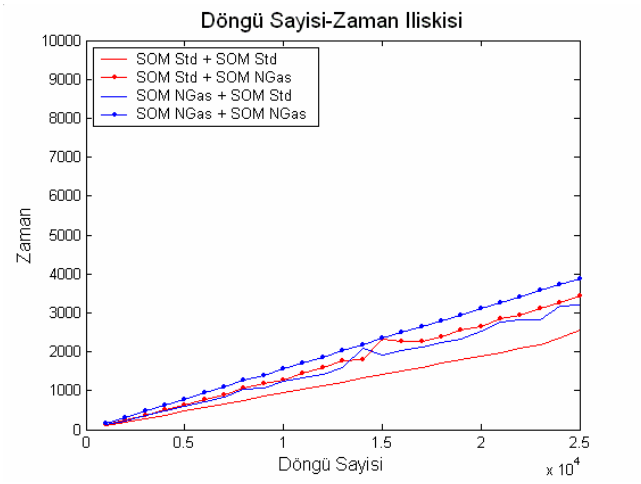
Şekil 6.5 Hiyerarşik k-Means algoritmasında kullanılan güncelleme yöntemlerine göre algoritmanın hızının döngü sayısına göre değişimi



Şekil 6.6 1. seviyede k-Means 2. seviyede 1D SOM kullanan hiyerarşik yapılı VQ algoritmasının hızının döngü sayısına göre değişimi



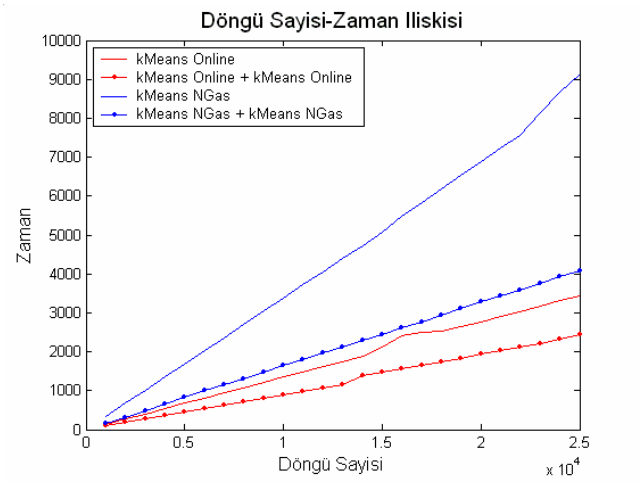
Şekil 6.7 1. seviyede 1D SOM 2. seviyede k-Means kullanan hiyerarşik yapılı VQ algoritmasının hızının döngü sayısına göre değişimi



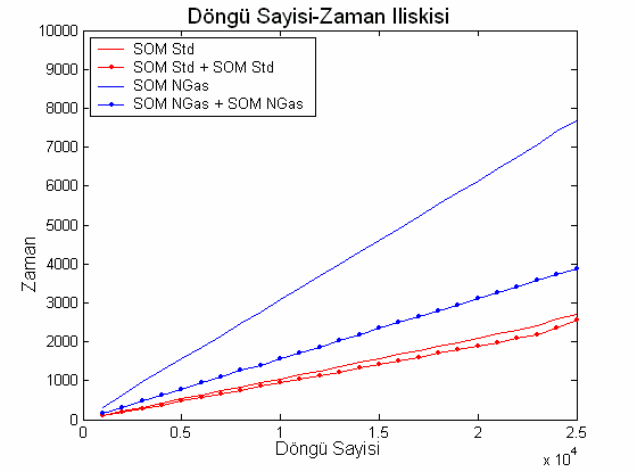
Şekil 6.8 Hiyerarşik 1D SOM algoritmasında kullanılan güncelleme yöntemlerine göre algoritmanın hızının döngü sayısına göre değişimi

6.1.3 VQ - Hiyerarşik Yapılı VQ Algoritmalarının Hızlarının Karşılaştırılması

Şekil 6.9'da k-means algoritması ve hiyerarşik yapılı k-means algoritmasının online ve NGas güncelleme yöntemleri için döngü sayısına göre hız değişimi görülmektedir. Şekil 6.10'da ise 1 boyutlu SOM algoritması ve hiyerarşik bir boyutlu SOM algoritmasının online ve NGas güncelleme yöntemleri için döngü sayısına göre hız değişimi görülmektedir. Elde edilen sonuçlara göre, hiyerarşik yapılı vektör kuantalama algoritmaları, vektör kuantalama algoritmalarına göre daha hızlı olduğu görülmüştür.



Şekil 6.9 k-Means ve hiyerarşik k-Means algoritmalarının hızlarının döngü sayısına göre karşılaştırılması



Şekil 6.10 1D SOM ve hiyerarşik 1D SOM algoritmalarının hızlarının döngü sayısına göre karşılaştırılması

6.2 Algoritmaların Hata Sonuçları

Renk kuantalama tekniği, resimden elde edilen $X=\{x_i, i=1,..,M \in X\}$ giriş renk kümesini daha az sayıda yeni renk kümesi $Q = \{q_j, j=1,..,N\}$ ile ifade etmektedir. Renk kuantalama adımından sonra her renk kendine en yakın olan yeni renk değeriyle ifade edilir ve (6.1)'deki eşitlikle ifade edilen bir renk hatası hata oluşur. $Q(x_i)$ x_i rengine en yakın çıkış rengini $d(x_i,q(x_i))$ ise x_i renginin kendisine en yakın çıkış rengi ile arasındaki öklit mesafesini göstermektedir. D boyutlu uzay için öklit mesafesi (6.2)'deki gibi hesaplanmaktadır.(Albayrak, 2002)

$$E_{MSE}(X,Q)=\frac{1}{M}\sum_{i=0}^{M-1}d(x_i,q(x_i)) \tag{6.1}$$

$$d(x_i,q_j)=\left\|x_i-q_j\right\|=\sqrt{\sum_{k=1}^{d}(x[i,k]-q[j,k])^2} \tag{6.2}$$

Hataların karşılaştırıldığı grafiklerde yatay eksen döngü sayısını dikey eksen ise hata değerini göstermektedir. Hata değeri olarak ortalama karesel renk hatası(MSE) alınmıştır.

Hata sonuçları değerlendirilirken iki türlü hata incelemesi yapılmıştır. Birinci değerlendirme için döngü sayısı 1.000 alınarak kuantalama algoritmalarının her iki döngüde bir hata değerleri hesaplanmıştır. Bu şekilde algoritmalarının hatalarının yakınsayıp yakınsamadığı, yakınsamasının ne şekilde olduğu grafiksel olarak karşılaştırılmıştır. İkinci hata değerlendirmesi için, döngü sayısı 1 ile 5.000 arasında değişmektedir. Yöntemler 20 döngü aralıklarıyla çalıştırılatak hata hesabı yapılmış ve döngü sayısı seçiminin hatayı nasıl etkilediği incelenmiştir.

Kuantalama algoritmalarının ve hiyerarşik yapılı kuantalama algoritmalarının farklı güncelleme yöntemleri kullanılarak çalıştırılması sonucunda elde edilen resim ve renk sonuçları Ek 2, Ek 3 ve Ek 4'te verilmiştir.

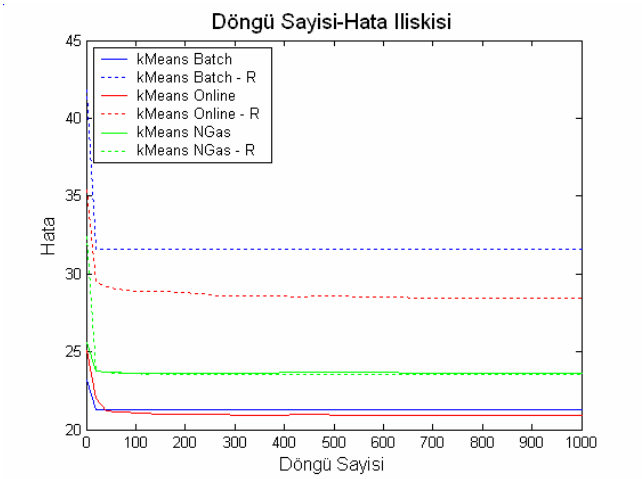
6.2.1 VQ Algoritmalarının Hatalarının Karşılaştırılması

Şekil 6.11, Şekil 6.12 ve Şekil 6.13'de döngü sayısı 1 ile 1.000 arasında değişmektedir. Her bir yöntem için, 2 döngüde bir hata hesaplanmış ve döngü sayısı boyunca hatanın nasıl değiştiği gözlemlenmiştir. Tüm algoritmalar döngü sayısı arttıkça yakınsayan bir özellik göstermişlerdir.

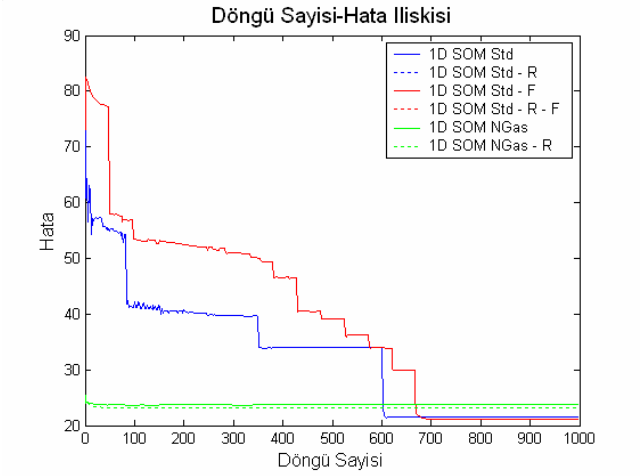
Şekil.6.11'den çıkarılabilecek ilk sonuç, k-means kuantalama yönteminde merkezlere resimden ilk değer ataması yapıldığında hatanın, daha düşük bir değere yakınsadığıdır. İlk değer atamasının resimden seçilerek yapıldığı taktirde batch ve online güncelleme kullanılarak NGas güncelleme yöntemine göre daha başarılı sonuç elde edilmiştir. Çıkarılabilecek ikinci sonuç, k-means algoritmaları, güncelleme yöntemine bağlı kalmaksızın düşük döngülerde yakınsamaktadır.

1D ve 2D SOM algoritmasında ilk değer atamasının rasgele ya da resimden seçilerek yapılmasının hata sonuçlarına bir etkisinin olmadığı görülmüştür (Şekil 6.12). NGas uygulanan 1D ve 2D SOM algoritmaları diğer yöntemlere göre daha düşük döngülerde daha

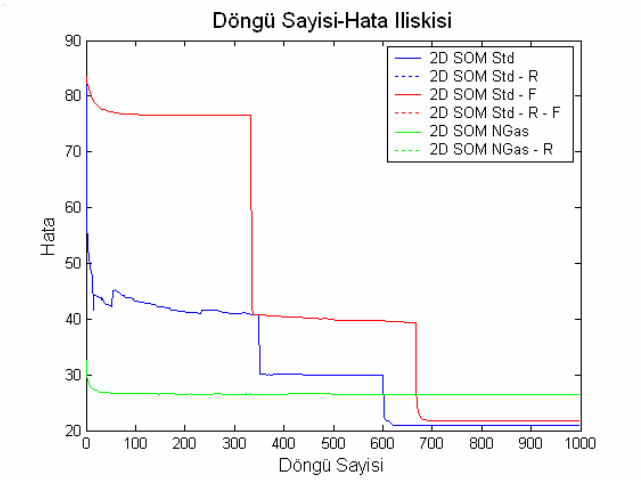
yüksek hata değerine yakınsamıştır (Şekil 6.12, Şekil 6.13).



Şekil 6.11 k-Means algoritmasında, farklı güncelleme yöntemlerinin ve farklı ilk değer atamanın algoritmanın hatasına etkisi



Şekil 6.12 1D SOM algoritmasında, farklı güncelleme yöntemlerinin, farklı komşuluk fonksiyonunun ve farklı ilk değer atamanın algoritmanın hatasına etkisi



Şekil 6.13 2D SOM algoritmasında, farklı güncelleme yöntemlerinin, farklı komşuluk fonksiyonunun ve farklı ilk değer atamanın algoritmanın hatasına etkisi

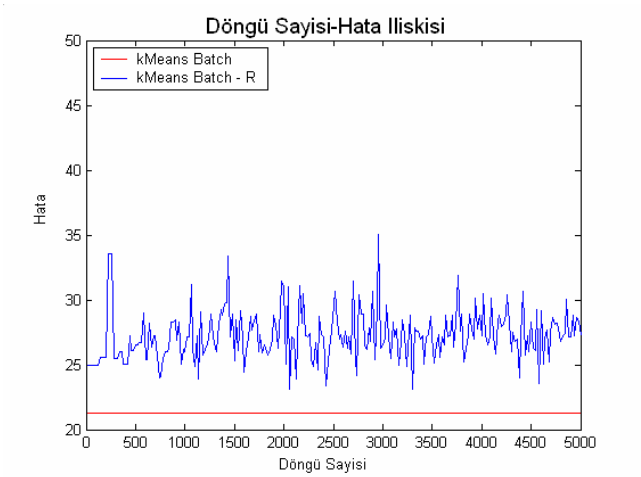
Şekil 6.14 ve Şekil 6.20 arasındaki grafiklerde, döngü sayısı 1 ile 5.000 arasında değişmektedir. Yöntemler 20 döngü aralıklarıyla çalıştırılarak hata hesabı yapılmış ve döngü sayısı seçiminin hatayı nasıl etkilediği incelenmiştir.

K means algoritmasında, batch ve online güncelleme yöntemleri için, ilk değer ataması rasgele yapıldığında hata değeri yüksek aralıklar arasında salınım yapmaktadır (Şekil 6.14 ve 6.15). Rasgele ilk değer ataması yapıldığında algoritma herhangi bir döngü sayısında küçük bir hata değerine yakınsayabilir. Algoritma aynı döngü sayısı için tekrar çalıştırıldığında maksimum hata değeri de elde edebilir. En küçük hatanın bulunmasında döngü sayısına bağlı bir düzen olmadığı görülmektedir.

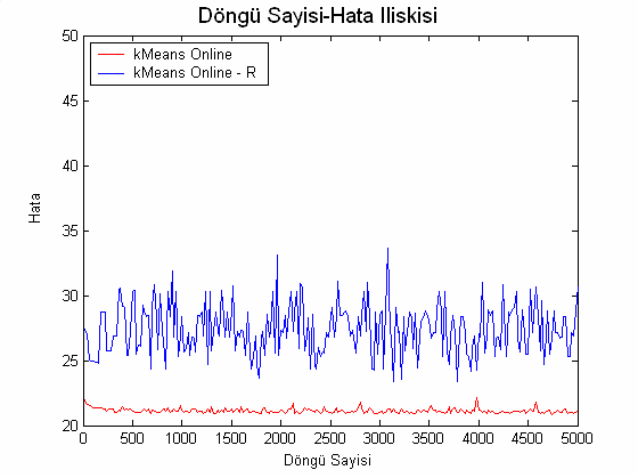
K means algoritmasında, batch ve online güncelleme yöntemleri için, kuantalama seviyelerinin ilk değerlerinin resimden seçildiği durumlarda hatanın küçük salınımlar yaparak yakınsadığı görülmektedir (Şekil 6.14 ve 6.15). Batch güncelleme yönteminin kullanıldığı k means, 20 döngü sayısına ulaşmadan önce yakınsamıştır. Daha büyük seçilen döngü sayılarında aynı değere yakınsayacaktır. Bu grafiklerden çıkaracağımız sonuç, k-means öbekleme yönteminde kuantalama seviyelerinin ilk değerlerinin resimdeki renklerden seçilmesi, düşük döngü sayılarında yakınsamayı ve kararlı bir yapısının olmasını sağlamaktadır. Tüm bunlar, güncelleme yöntemi olarak hangi yöntem kullanılırsa kullanılsın, k-means algoritmalarında ilk değer atamasının ne kadar önemli olduğunu göstermektedir.

K means algoritmasında, NGas güncelleme yönteminde dikkati çeken farklı bir unsur, rasgele ilk değer ataması yapıldığında oluşan hata değerlerinin yaptığı salınımların, resimden renkler

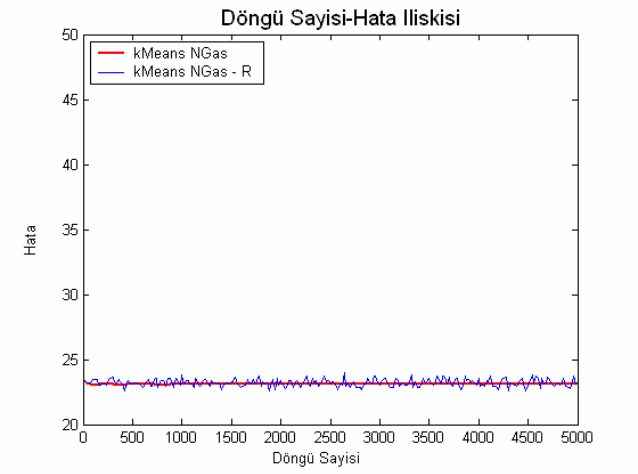
seçildiği durumda elde edilen hata değerlerini ortalama alacak şekilde olduğudur (Şekil 6.16).



Şekil 6.14 Batch k-Means algoritmasında ilk değer atamasının hata sonucuna etkisi

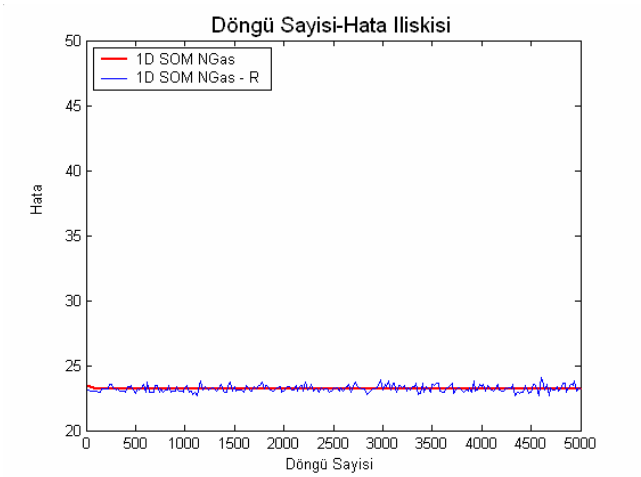


Şekil 6.15 Online k-Means algoritmasında ilk değer atamasının hata sonucuna etkisi

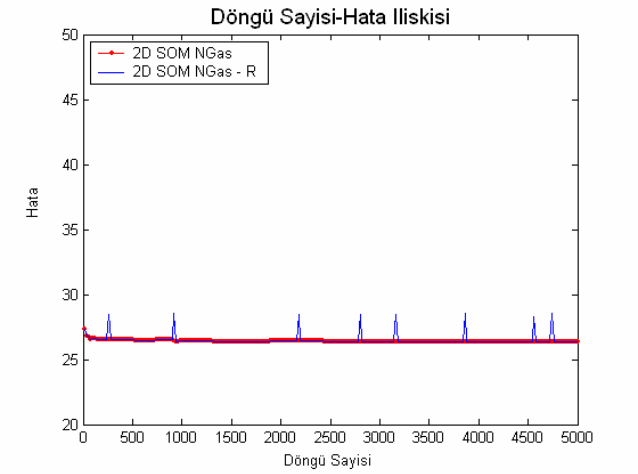


Şekil 6.16 NGas k-Means algoritmasında ilk değer atamasının hata sonucuna etkisi

SOM algoritmalarında geleneksel ve NGas güncelleme yöntemleri için, iki seçim kriterinin döngü sayısına göre hata sonucunu nasıl etkilediği gözlemlenmiştir. Bunlardan birincisi ilk değer atama ikincisi ise komşuluk fonksiyonu seçimidir. NGas güncelleme yöntemi tüm kuantalama seviyelerini güncellediği için komşuluk değerlendirmesi yapılmamıştır. SOM algoritmalarında da NGas güncelleme yöntemi uygulandığında, hata sonuçlarının ilk değer atamadan etkilendiği ve hata sonuçlarının salınımlı bir yapıya sahip olduğu gözlenmiştir (Şekil 6.17, Şekil 6.18).



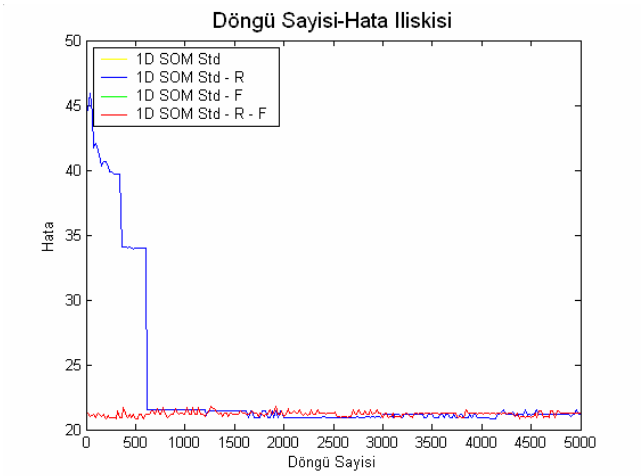
Şekil 6.17 NGas 1D SOM algoritmasında ilk değer atamasının hata sonucuna etkisi



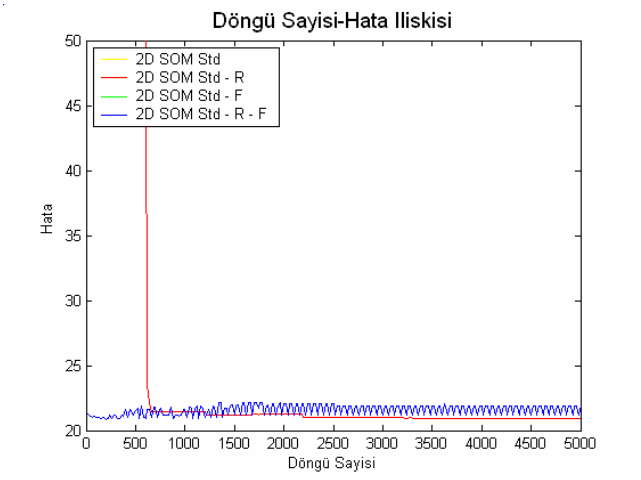
Şekil 6.18 NGas 2D SOM algoritmasında ilk değer atamasının hata sonucuna etkisi

Geleneksel güncelleme yöntemini kullanan 1D ve 2D SOM algoritmalarında, ilk değer atama hangi şekilde yapılırsa yapılsın sonucun yakınsamasına olumlu veya olumsuz bir etkisinin olmadığı görülmüştür(Şekil 6.19, Şekil 6.20).

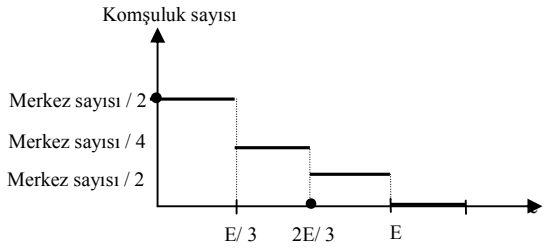
Geleneksel güncelleme yöntemi kullanılan SOM algoritmalarında iki farklı komşuluk tanımlanmıştır. Birincisi, döngü sayısının belli oranlarında, komşuluk sayısı nöron sayısının azalan bir katı olarak belirlenir (Şekil 6.21). İkincisi komşuluk değeri belirleme yöntemi bir fonksiyon kullanılarak gerçekleştirilmiştir. Bu fonksiyon Şekil 6.22'de görüldüğü gibi, döngü sayısının 2/3'üne kadar lineer olarak azalmakta, döngü sayısının son üçtebirinde 0 değerini almaktadır.



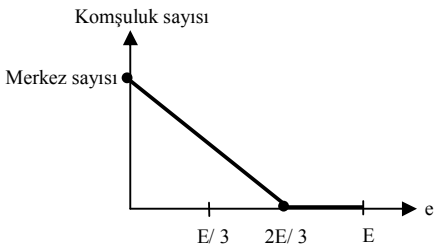
Şekil 6.19 Geleneksel 1D SOM algoritmasında ilk değer atamasının ve komşuluk fonksiyonu kullanmanın hata sonucuna etkisi



Şekil 6.20 Geleneksel 2D SOM algoritmasında ilk değer atamasının ve komşuluk fonksiyonu kullanmanın hata sonucuna etkisi



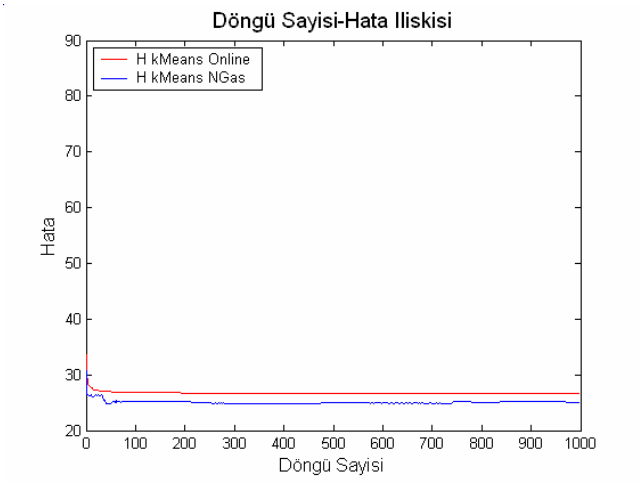
Şekil 6.21 SOM algoritmalarında güncellenecek komşuların sayısının belirlenmesinde kullanılan 1. komşuluk fonksiyonu



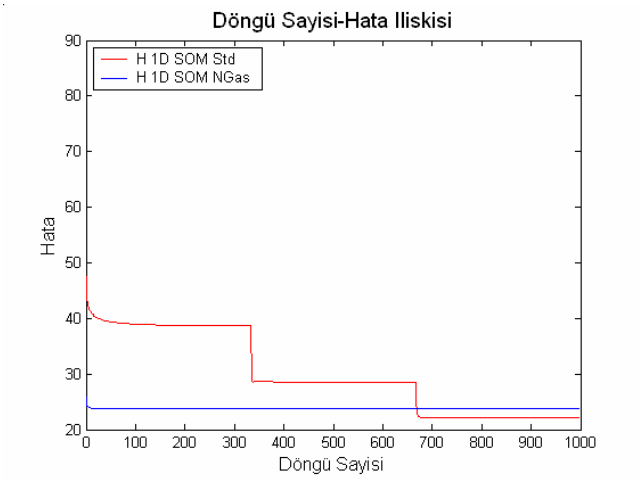
Şekil 6.22 SOM algoritmalarında güncellenecek komşuların sayısının belirlenmesinde kullanılan 2. komşuluk fonksiyonu

6.2.2 Hiyerarşik yapılı VQ Algoritmalarının Hatalarının Karşılaştırılması

Bu grafiklerde hiyerarşik yapılı kuantalama algoritmasının 1. seviyelerindeki MSE hata değerleri kullanılmıştır. Şekil 6.23 ve Şekil 6.24'te döngü sayısı 1 ile 1.000 arasında değişmektedir. Her bir yöntem için, 2 döngüde bir hata hesaplanmış ve döngü sayısı boyunca hatanın nasıl değiştiği gözlemlenmiştir. Hiyerarşik yapılı kuantalama algoritmalarının, 1. seviyesinde kullanılan kuantalama yönteminin karakteristik özelliklerini gösterdiği görülmüştür.



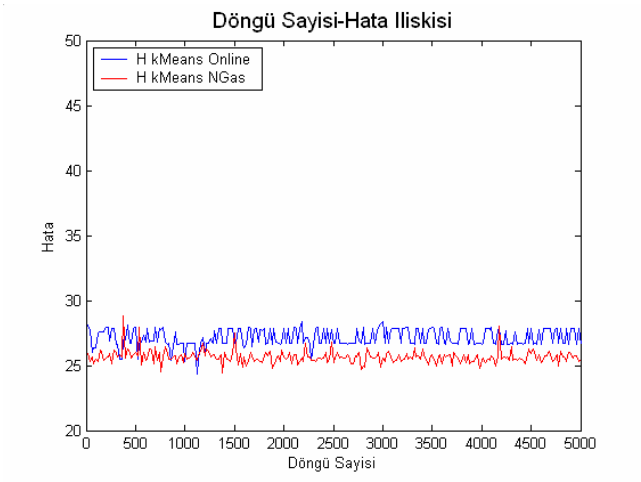
Şekil 6.23 Hiyerarşik yapılı online ve NGas k-Means algoritmasında hatanın döngü sayısına göre değişimi



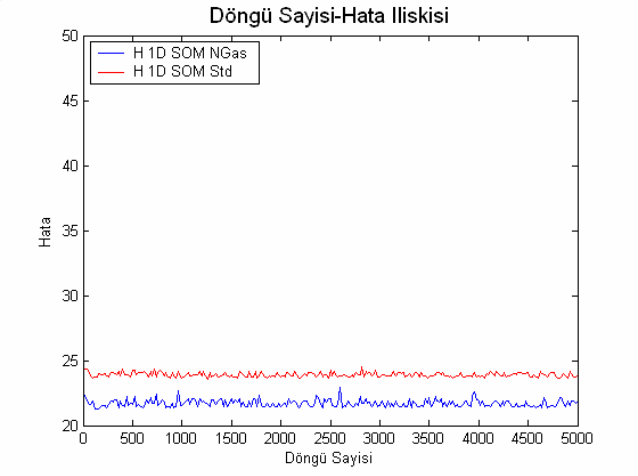
Şekil 6.24 Hiyerarşik yapılı online ve NGas k-Means algoritmasında hatanın döngü sayısına göre değişimi

Şekil 6.25 ve Şekil 6.30 arasındaki grafiklerde, hiyerarşik yapının her seviyesindeki

kuantalama yöntemleri için döngü sayısı 1 ile 5.000 arasında değişmektedir. Algoritma 20 döngü aralıklarıyla çalıştırılarak hata hesabı yapılmış ve döngü sayısı seçiminin hataya etkisi incelenmiştir. Hiyerarşik yapılı kuantalama algoritmaları, ilk değer atamasının resimden renkler seçilerek yapılmasına rağmen salınımlı bir yapı göstermiştir.



Şekil 6.25 Hiyerarşik yapılı online ve NGas k-Means algoritmasında döngü sayısının hataya etkisi

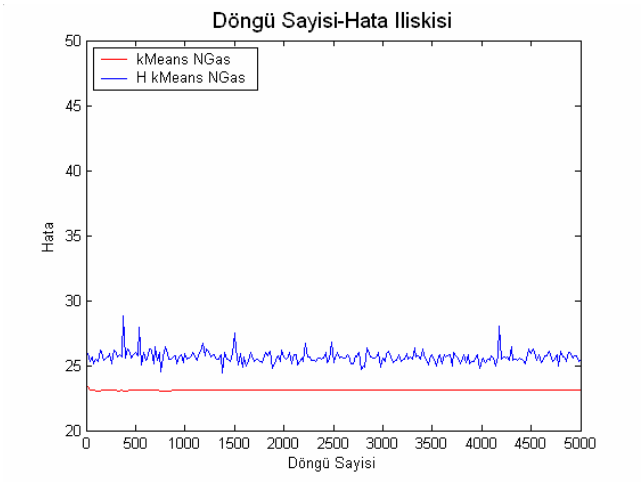


Şekil 6.26 Hiyerarşik yapılı geleneksel ve NGas SOM algoritmasında döngü sayısının hataya etkisi

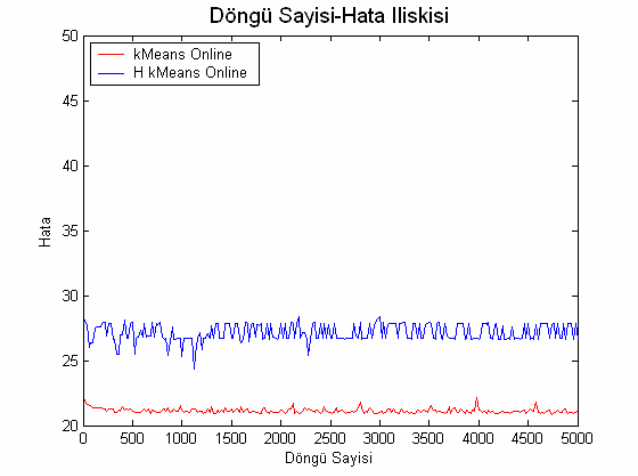
6.2.3 VQ - Hiyerarşik Yapılı VQ Algoritmalarının Hatalarının Karşılaştırılması

Şekil 6.27 ve Şekil 6.30 arasındaki grafikler döngü sayısı seçiminin k Means ve SOM algoritmalarında farklı güncelleme yöntemleri için, hiyerarşik yapılı olmasının hataya etkisi kuantalama algoritmalarının hatası ile karşılaştırılmıştır. Döngü sayısı ne seçilirse seçilsin

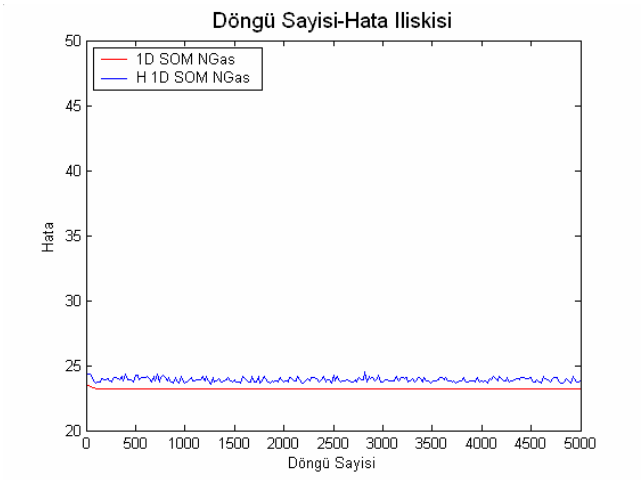
k-Means algoritması için hiyerarşik yapı kullanıldığında hata değeri daha yüksek olmuştur (Şekil 6.27 ve Şekil 6.28). Geleneksel güncelleme yöntemini kullanan SOM algoritması ve aynı algoritmanın hiyerarşik yapılısı birbirine yakın hata sonuçları üretmiştir. Hatta düşük döngü sayılarında hiyerarşik yapı kullanmanın daha iyi sonuç vereceği görülmektedir.



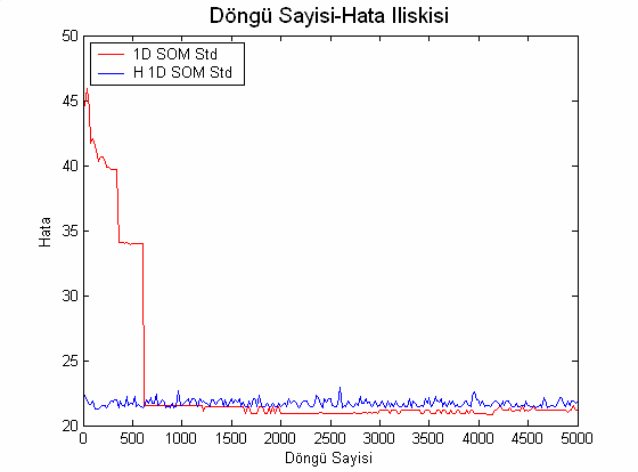
Şekil 6.27 NGas k-Means ve hiyerarşik yapılı NGas k-Means algoritmasında döngü sayısının hataya etkisi



Şekil 6.28 Online k-Means ve hiyerarşik yapılı online k-Means algoritmasında döngü sayısının hataya etkisi



Şekil 6.29 NGas 1D-SOM ve hiyerarşik yapılı NGas 1D-SOM algoritmasında döngü sayısının hataya etkisi



Şekil 6.30 Geleneksel güncellenen 1D-SOM ve hiyerarşik yapılı geleneksel güncellenen 1D-SOM algoritmasında döngü sayısının hataya etkisi

7. KODLAMA

7.1 Sıkıştırma Tarihi

1838 yılında telgraf için kullanılan mors alfabesinde, ingilizcede daha yaygın olan E ve T gibi harfler için diğer harflere nazaran daha kısa kodlar kullanılmıştır. Bu özelliğiyle mors alfabesi, bir veri sıkıştırma örneğidir. Veri sıkıştırmada modern yöntemler, bilgi teorisinin gelişmesiyle birlikte 1940'lı yılların sonunda kullanılmaya başlanmıştır. 1949 yılında Claude Shanon ve Robert Fano, blokların olasıklarına dayanan kod sözcüklerinin atanması için sistematik bir yol geliştirmişlerdir. 1951 yılında ise bunu yapmanın en uygun çözümü David Huffman tarafından bulunmuştur. 1970'li yılların ortalarında, kullanılan veriye dayanan Huffman kodlaması için dinamik olarak güncellenen sözcükler fikri ortaya çıkmıştır. Ve 1970'li yılların sonlarında, metin dosyalarının heran erişilebilir saklanmasının yaygınlaşmasıyla, neredeyse hepsi adaptif Huffman kodlamasına dayanan sıkıştırma programları oluşturulmaya başlanmıştır. 1977 yılında Abraham Lempel ve Jacob pointer tabanlı kodlama fikrini önermişlerdir. 1980'li yılların ortalarında Teri Welch'in devam eden çalışmasıyla LZW olarak adlandırılan algoritma, genel amaçlı sıkıştırma sistemlerinin çoğu için seçilen bir yöntem haline gelmiştir. 1980'li yılların sonlarında sayısal görüntü daha yaygın hale gelmiş ve bunların sıkıştırılması için standartlar ortaya çıkmıştır. 1990'lı yılların başlarında kayıplı sıkıştırma yöntemleri de geniş çaplı kullanılmıştır.[4]

1951 yılında, David Huffman ve MIT bilgi teorisi dersi sınıf arkadaşlarının, dönem ödevi veya final yazılısı arasında seçim yapılması istenmişti. Profesör Robert M. Fano, dönem ödevini en etkin ikili kodun bulunması olarak belirlemişti. Huffman, en etkin kodu kanıtlayamadığı için neredeyse vazgeçmiş ve final sınavına çalışmaya başlayacaktı ki aklına frekansa göre sıralanmış ikili kod fikri geldi ve çabucak bu yöntemin etkin olduğunu kanıtladı. Böylece, bilgi teorisinin mucidi Claude Shannon ile benzer bir kod oluşturmak için çalışan profesörünü geçmiş oldu.[5]

7.2 Entropi Kodlama

Entropi kodlama işlemi modelleme ve kodlama olmak üzere iki parçaya ayrılabilir. Modellemede sembollere olasılıkları atanır, kodlamada ise bu olasılıklardan bir bit dizisi üretilir. Sembol olasılığı ile o sembole ait bit dizisi arasında bir ilişki vardır. İyi bir sıkıştırma oranı elde edebilmek için belirli olasılık tahminine ihtiyaç duyulur. Model herbir sembolün olasılığından sorumlu olduğu için, modelleme veri sıkıştırmadaki en önemli aşamalardan

biridir.[6]

Entropi kodlaması, Huffman kodlama gibi herbir sembol için ayrık sayı kullanan veya birçok sembol için bir ayrık sayı kullanan bir sıkıştırma şeması ile gerçeklenebilir. Bir entropi kodlaması, sembollerin olasılıklarıyla kod uzunluklarını eşleştirmek üzere kodları sembollere atayan bir kodlama şemasıdır. Tipik olarak verinin sıkıştırılması için, eşit uzunluklu kodlar olarak gösterilen sembollerin olasılığın ters logaritmasıyla orantılı kodlarla gösterilen sembollerle değiştirilmesiyle entropi kodlamaları kullanılır. [7]

En genel üç entropi kodlama tekniği Huffman kodlaması, değer (range) kodlama ve aritmetik kodlamasıdır[7]. Huffman kodlaması, kayıpsız veri sıkıştırması için kullanılan bir entopy kodlama algoritmasıdır. Bu terim, kaynak sembolün her olası değeri için tahmin edilen görünme olasılığına dayanarak belirli bir şekilde oluşturulan değişken-uzunluk kod tablosunun kullanımına işaret etmektedir. [6]

7.3 Huffman Kodlama

Huffman kodlaması, çok kullanılan sembolleri, daha az kullanılan sembollerininkinden kısa bit katarlarıyla ifade eden öntakısız (prefix-free) koddur. Huffman, bu tipin en etkin sıkıştırma yöntemini dizayn etmiştir. Huffman kodlaması, “prefix-free code” oluşturulmasında o kadar yaygın bir yöntemdir ki, kod Huffman algoritması tarafından oluşturulmamış olsa da “Huffman kodlaması” terimi “prefix-free code” ile eş anlamda kullanılır.[5]

Uygulama sırasında semboller için kullanılan sıklıklar, uygulamaya özel belirlenmiş genel ortalama sıklıklar olabileceği gibi sıkıştırılacak verideki sembollerin gerçek kullanım sıklıkları da olabilir. Genel sıklıkların kullanılması durumunda sembollerin kodlarının birkez aktarılması yeterli olacaktır. Sembollerin, sıkıştırılacak verideki sıklıklarının kullanılması durumunda ise sembollere ait kod değerleri veya bu kod değerlerinin elde edilebileceği ipuçları da veri ile birlikte aktarılmak zorundadır. Bunun etkin olarak gerçekleştirilebilmesi için çeşitli yöntemler kullanılmaktadır. [5]

Herbir giriş sembolünün olasılığının ikinin eksi kuvveti olduğu durumlarda Huffman kodlaması, optimum sıkıştırma sağlamaktadır. Olasılıkların ikinin eksi kuvvetlerinin arasına düştüğü durumlarda yeterince etkin bir sıkıştırma sağlanamayabilir. Sıkıştırmanın daha etkin olmasını sağlamak için sembollerin tek tek Huffman kodlamasının yapılması yerine sembol bloklarının Huffman kodlaması yapılabilir. Örneğin metin sıkıştırmada Huffman kodlamasından önce, harflerin “kelimelere” birleşimiyle genişletilen alfabe boyutu, sıkıştırma

verimine biraz yardımcı olabilmektedir. Huffman kodlaması için en kötü durum, bir sembolün olasılığının, sınırsız etkinsizliğin üst limiti olan 2^{-1}'i geçmesidir. Bu durumu önlemek için, semboller ön işlemlerden geçirilebilir.[3]

Aritmetik kodlama, Huffman kodlamasının genelleştirilmiş hali olarak görülebilir. Her ne kadar aritmetik kodlama, Huffman kodlamasından daha iyi sıkıştırma performansına sahip olsa da Huffman kodlaması, sadeliği, yüksek hızı ve patent engeli olmaması nedeniyle hala yaygın olarak kullanılır.[3]

7.3.1 Huffman Kodlamasının Kullanıldığı Yerler

Güncel görüntü sıkıştırma standartları arasında FAX CCITT 3, GIF (LZW), JPEG, BMP (RLE, vs.), TIFF(fax, jpeg, gif, vs.) metotları bulunmaktadır.

Huffman kodlaması, günümüzde genelikle, bazı diğer sıkıştırma yöntemlerine "arka-son" (back-end) olarak kullanılmaktadır. DEFLATE ve JPEG, MP3 gibi çoklu ortam kodları bir ön-son (front-end) modeline ve Huffman tarafından takip edilen kuantalamaya sahiptir. [3]

7.3.2 Dinamik Huffman

Veri sıkıştırma, statik veya dinamik olmak üzere iki grupta sınıflandırılabilir. Statik yöntem, iletilecek veri kümesinin kod kümesi ile eşleşmelerinin, veri iletimi başlamadan sabitlendiği bir yöntemdir. Böylece verilen bir mesaj, her görüldüğünde aynı kodlar ile temsil edilmektedir. Huffman kodlaması dendiğinde statik yöntemden bahsedilmektedir. [8]

Eğer veri kümesinin kod kümesi ile eşleşimi aktarma sırasındanın herhangi bir anında değişiyorsa, kod dinamiktir. Kodların sembollere atanması, zamanda her nokta için görülmelerin göreli sıklığına dayanır. X sembolü iletiminin başlangıç aşamasında, daha sık görünmesinden dolayı bu sembol kısa bir kod ile ifade edilebilir. Halbuki veri kümesinin tamamındaki görünme sıklığı daha düşük olabilir. İletimin başlarında az görülen semboller daha sonra sık görülmeye başladığında kısa kodlar bu yüksek sıklığa sahip mesajlara atanır ve x daha uzun kod ile gösterilir. [8]

Örnek olarak, Çizelge 7.1'deki örnekte "aa bbb" öneki ile uyumlu bir dinamik Huffman kod tablosu görülmektedir. Her ne kadar boşluğun tüm mesaj içindeki sıklığı b'ninkinden daha fazla olsa da, bu noktada, b'nin sıklığı daha fazladır ve böylece daha kısa bir kod sözcüğü ile gösterilir. [8]

Çizelge 7.1 “aa bbb” prefixi için dinamik Huffman kod çizelgesi örneği

Kaynak mesaj	Olasılık	Kod Sözcüğü
A	2/6	10
B	3/6	0
Boşluk	1/6	11

7.3.2.1 Huffman Kodlama Algoritma Örneği

Huffman algoritması, $\{w_1,.....,w_n\}$ negatif olmayan ağırlıklar listesini girdi olarak alır ve yaprakları bu ağırlıklarla işaretlenmiş, tam bir ikili ağaç (full binary tree) oluşturur. Bir kod oluşturmada Huffman algoritması kullanıldığında, ağırlıklar kaynak harflerin olasılıklarını gösterir. Başlangıçta, listedeki herbir ağırlık için tekil ağaç seti vardır. Her adımda, en küçük ağırlığa sahip olan ağaçlar, w_i ve w_j, ağırlığı bu iki ağacın ağırlıklarının toplamı olan, $w_i + w_j$, yeni bir ağaç altında birleştirilir. w_i ve w_j ağırlıkları listeden çıkarılır ve w_i+w_j ağırlığı listeye eklenir. [8]

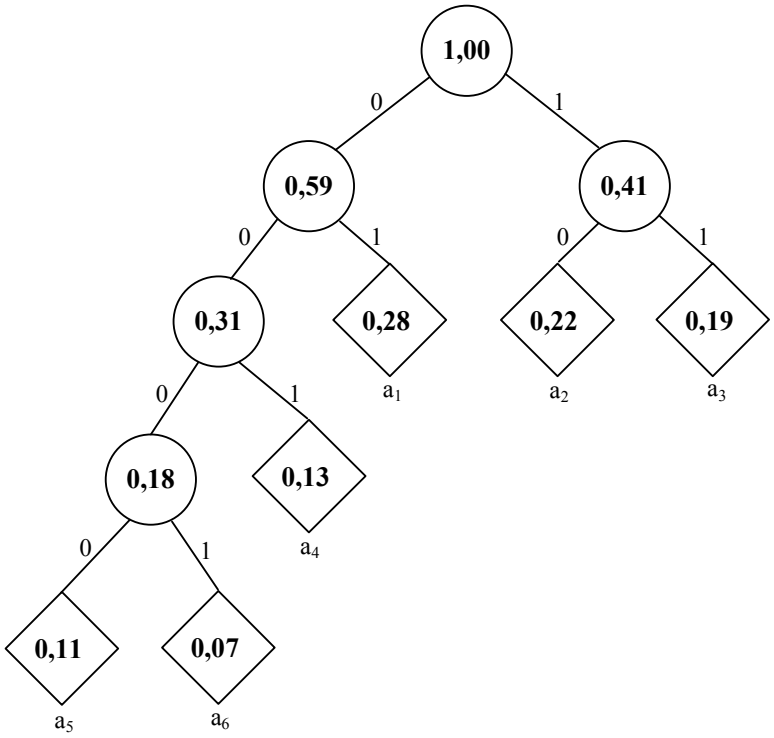
Bu işlem, ağırlık listesinde tek bir değer kalana kadar devam eder. Eğer, herhangi bir anda, en küçük ağırlığa sahip olan çiftleri seçmek için birden çok yol varsa, herhangi iki değer seçilebilir. Huffman’ın çalışmasında, işlem artmayan ağırlık listesinde başlar. Bu detay, algoritmanın doğruluğu için önemli değildir, ama daha etkin bir uygulama sağlar. Huffman algoritması, Çizelge 7.2’de gösterilmiştir. [8]

Çizelge 7.2 Huffman işlemi sonrasında elde edilen liste

Semboller	Olasılıklar	1. Adım	2. Adım	3. Adım	4. Adım	5. Adım
a_1	0,28	0,28	0,31	0,41	0,59	1,00
a_2	0,22	0,22	0,28	0,31	0,41	
a_3	0,19	0,19	0,22	0,28		
a_4	0,13	0,18	0,19			
a_5	0,11	0,13				
a_6	0,07					

Huffman algoritması, herbir kaynak harfine, a_i, eşleştirilecek kod sözcüklerinin uzunluklarını belirler. Asıl basamakların belirlenmesi için birçok alternatif vardır; tek gereklilik, kodun öntakı özelliğine sahip olmasıdır. Alışılmış atama, ebeveynden sol taraftaki çocuğa gidişi 0 ile sağ çocuğa gidişi ise 1 ile gösterimdir. Herbir kaynak harf için kodlama, kökten harfi temsil eden yaprağa olan seridir. Şekil 7.1’te gösterilen kaynak için, azalan olasılıklara göre, kodlamalar {01, 11, 001, 100, 101, 0000, 0001}’dir. Açıkça, bu işlem minimal öntakı kodu sağlar. Ayrıca, algoritma ile en uygun (optimum – minimum artıklı) kodun oluşturulmasını

garanti eder. Gallager, bir Huffman kodunun artıklığı için üst sınırın, p_n en az görülen kaynak mesajının olasılığını göstermek üzere, $p_n + \lg[(2 \lg e) / e] \approx p_n + 0{,}086$ olduğunu kanıtlamıştır. [8]



Şekil 7.1 Huffman işlemi sonrasında elde edilen ağaç

7.3.3 Çalışmada Uygulanan Huffman Kodlama Algoritması

Bu çalışmada 256 renk bmp resmi, vektör kuantalama algoritmaları kullanılarak renk sayısı 16, 64 ve 100'e indirilerek Huffman kodlaması uygulanmıştır. Huffman kodlama algoritması aşağıdaki şekilde uygulanmıştır.

1. Adım : Paletteki renklerin frekans bilgisi resimden elde edilir. Renklerin indisleri ve frekans değerleri "ilk_semboller" dizisine atılır.

2. Adım : Bu dizi frekans değerleri küçükten büyüğe olacak şekilde sıralanır.

3. Adım : Sıralanan dizi Huffman ağacını oluşturmak üzere hazırlanır : Dizinin ilk iki elemanı toplanır. Toplam değeri dizinin içinde kendisinden küçük ya da eşit ilk elemanın öncesine negatif toplam değeri ile yerleştirilir. Sembol değeri olarak resimde kullanılmayan herhangi bir sembol kullanılabilir. Bu çalışmada negatif toplam bulunan gözlere sembol

olarak ‘-1’ atandı. Dizinin bir elemanı gibi değerlendirilip toplam yapılırken mutlak değeri alınır. Dizinin sonraki iki elemanı toplanıp dizide ait olduğu yere aynı şekilde yerleştirilir. Dizinin başlangıçtaki eleman sayısı N ise bu işleme dizinin eleman sayısı 2*N-2 olana kadar devam edilir.

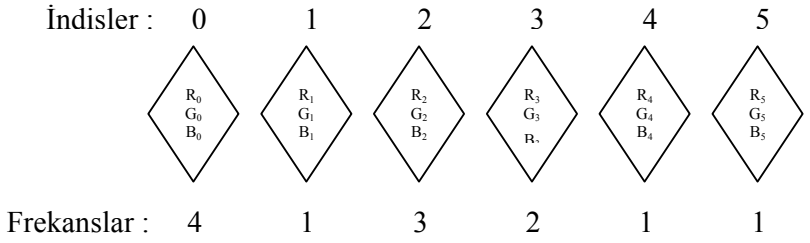
4. Adım : Hazırlanmış dizi kullanılarak Huffman ağacı oluşturulur.

5. Adım : Huffman ağacı kullanı kullanarak başlığa yazılacak binary ağaç ve bu ağaca yerleşmiş olan sembollerin ağacın en sol ve en altından başlayarak indisleri elde edilir

7.3.3.1 256 Renk bmp resmi için Huffman Kodlama Örneği

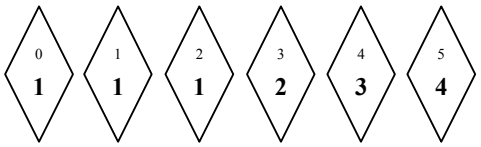
Aşağıda algoritmanın adımlarının uygulandığı bir örnek yer almaktadır.

1. Adım : Huffman kodlama uygulanacak resimdeki tüm farklı renkler ve bu renklerin frekans bilgileri elde edilmelidir. Resmin palet bilgisinden renkler alınır. Frekans bilgisini elde edebilmek için ise tüm resim baştan sona taranıp paletteki renklerden kaçar adet bulunduğu bilgisi elde edilir. Örnek resim 6 farklı renkten oluşsun, renkler ve frekansları aşağıdaki gösterildiği gibi olsun :



Şekil 7.2 Huffman kodlama uygulanacak renkleri ve renklerin frekans bilgileri

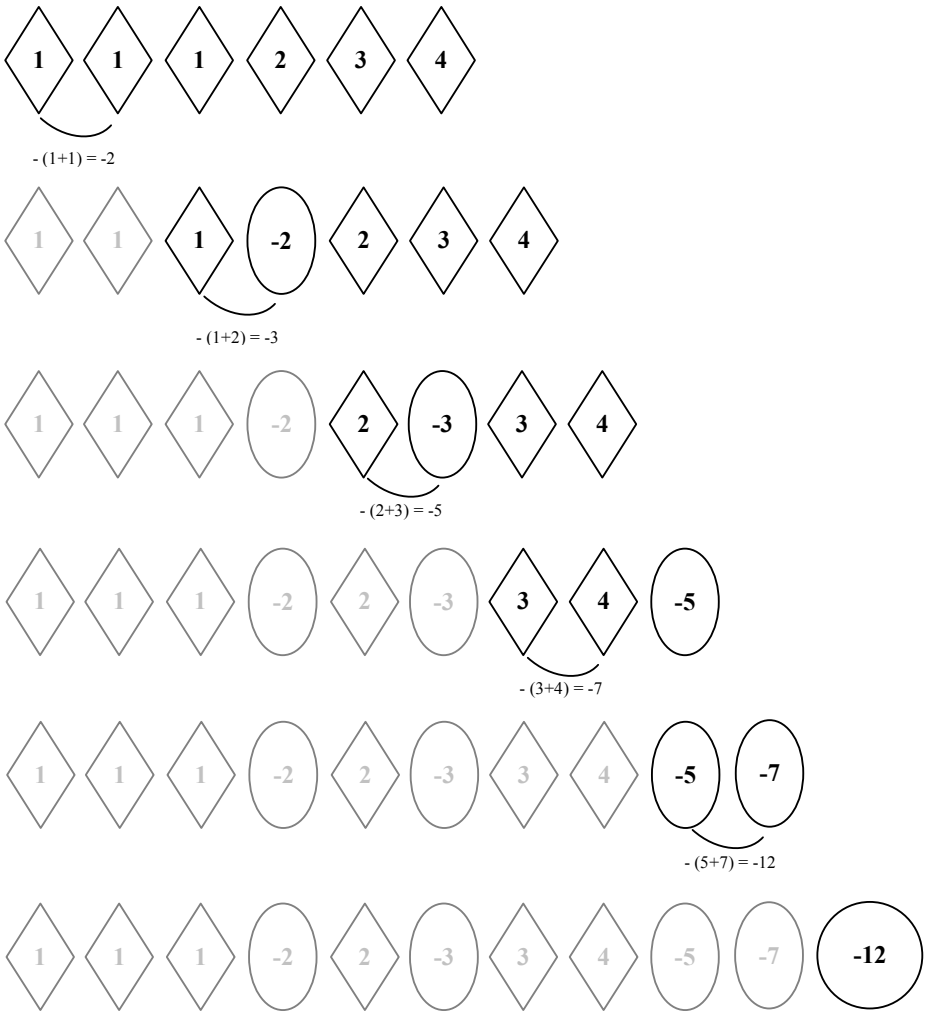
2.Adım : Frekanslar ve ait oldukları renklerin indisleri her iki bilgiyi de saklayacak yapıda kurulmuş bir dizide saklanır ve bu dizi frekans değerleri küçükten büyüğe olacak şekilde sıralanır.



Şekil 7.3 Frekans değerlerine göre sıranmış renkler

3.Adım : Sıralanan dizi Huffman ağacını oluşturmak üzere hazırlanır. Dizinin ilk iki elemanı toplanır. Toplam değeri dizinin içinde kendisinden küçük ya da eşit ilk elemanın öncesine

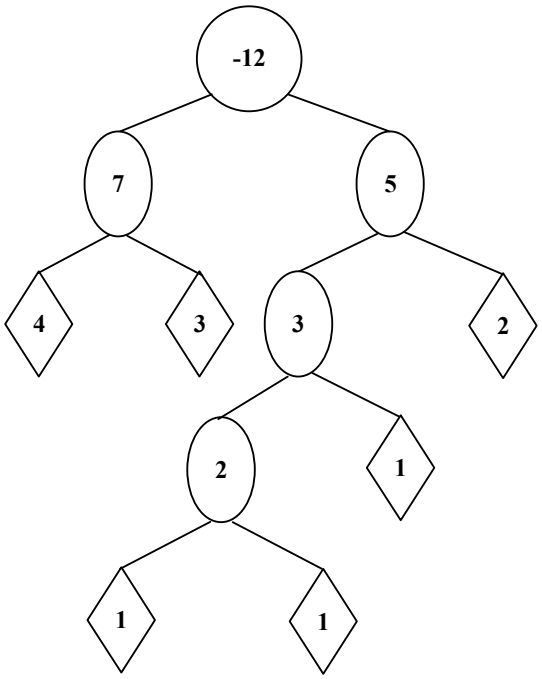
negatif toplam değeri ile yerleştirilir. Ve sembol değeri olarak resimde kullanılmayan herhangi bir sembol kullanılabilir. Bu çalışmada negatif toplam bulunan gözlere sembol olarak '-1' atandı. Bu toplam değeri dizinin bir elemanı gibi değerlendirilir, toplam yapılırken mutlak değeri alınarak toplama katılır. Dizinin sonraki iki elemanı toplanıp dizide ait olduğu yere aynı şekilde yerleştirilir. Dizinin başlangıçtaki eleman sayısı N ise bu işleme dizinin eleman sayısı 2*N-1 olana kadar devam edilir. Örnek resimde 6 adet renk olduğuna göre bu işleme dizinin eleman sayısı 11 olana kadar devam edilmelidir.



Şekil 7.4 Sıralı frekans dizisi kullanılarak Huffman ağacı oluşturulacak dizinin oluşturulma aşamaları

4.Adım : Hazırlanmış bu dizi kullanılarak Huffman ağacı oluşturulur. Dizinin 2n-1. elemanı ağacın ilk düğümü olur. Huffman ağacı yukarıdan aşağı büyüyen ikili bir ağaçtır. Dizide sondan başa doğru ilk gelen eleman düğümün sol dalına ikinci eleman sağ dalına yerleştirilerek ilerlenir. Pozitif toplamlar ağacın yaprakları, negatif toplamlar ağacın

düğümleridir.

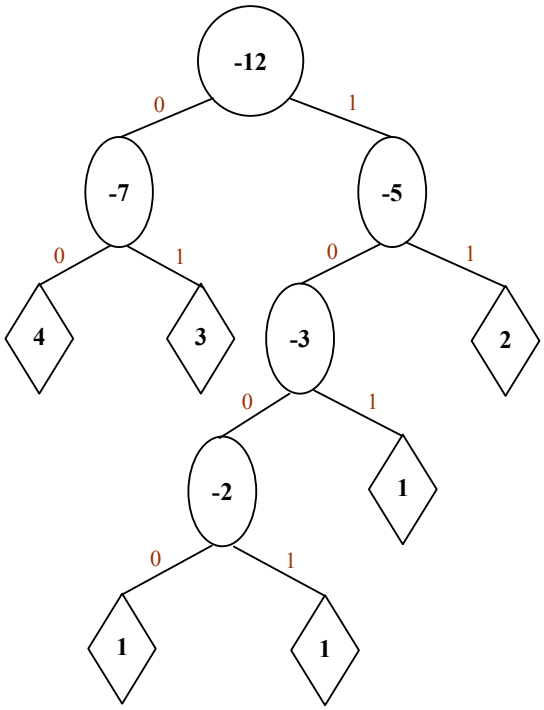


Şekil 7.5 Huffman ağacı için hazırlanmış dizi elemanlarının Huffman ağacına yerleştirilmesi

5.Adım : Renklerin kod tablosu oluşturulur. Ağacın sol dallarını 0 sağ dallarını 1 ile değerlendirdiğimizde (Şekil 7.6), Çizelge 7.3'de belirtilmiş olan kodlar elde edilecektir.

Çizelge 7.3 Resimdeki renklerin Huffman kodları

Renk			**Kod Tablosu**
R	**G**	**B**	
R_0	G_0	B_0	0 0
R_1	G_1	B_1	1 0 0 0 0 0
R_2	G_2	B_2	0 1
R_3	G_3	B_3	1 1
R_4	G_4	B_4	1 0 0 0
R_5	G_5	B_5	1 0 1



Şekil 7.6 Huffman ağacının dallarının kodlanması

7.4 Sıkıştırmada Hız ve Sıkıştırma Oranı Kavramları

Sıkıştırmada iki önemli kavramdan sözedilebilir: algoritma karmaşıklığı ve sıkıştırma oranı. Veri sıkıştırma veri aktarımında kullanılacaksa, hedeflenen algoritmanın hızıdır. İletim hızı, gönderilen bitlerin sayısına, kodlayıcının kodlu mesajı oluşturması için gereken zamana ve kod çözücünün orijinal mesajı oluşturması için gereken zamana bağlıdır. Veri depolama işleminde, algoritmanın hızından çok, sıkıştırma oranının yüksek olması ön plana çıkmaktadır. Sıkıştırma oranı eşitlik 7.1'de verilmiştir. Statik şema için analiz edilmesi gereken üç algoritma vardır: eşleştirme algoritması, kodlama algoritması ve kod çözme algoritmaları. Dinamik şema için ise iki algoritma vardır: kodlama ve kod çözme algoritmaları. (22)

$$\textit{Sıkıştırma oranı} = 100 * (1 - \textit{giriş boyutu} / \textit{çıkış boyutu}) \qquad (7.1)$$

7.5 Sıkıştırılmış Dosyanın Hazırlanması

Sıkıştırılmış dosyanın yazılmasında iki aşama söz konusudur. Birincisi sıkıştırılmış dosyanın açılabilmesi için gerekli başlık bilgisinin yazılması, ikincisi dosyayı oluşturan sembollerin Huffman kodları ile eşleştirilerek yazılmasıdır.

Başlık oluşturulurken başlığa, kaç adet sembol olduğu, sembollerin kendileri, resmin yükseklik ve genişlik bilgisi, ve Huffman ağacı veya ağacı oluşturmak için gerekli ve yeterli bilgi yazılmalıdır. Bu çalışmada, ağaç kodlanarak başlığa yerleştirilmiştir. Bu kodlama sayesinde ağaç, N adet sembol bulunan bir resim için 2N-2 bit ile ifade edilebilmektedir.

Ağacın kodlanmasında, ilk düğümden başlanarak, sol dal için 0, sağ dal için 1 kullanılır. Sola yol olduğu sürece sola gidilip, yaprağa ulaşıldığında bir üst düğümün sağındaki düğüm için değerlendirme yapılır. Her yaprağa ulaşıldığında aynı durum tekrarlanır. Bir önceki 6 farklı renk örneğinde oluşturulan Huffman ağacı bu kurallara göre kodlandığında kodlanmış ağaç Çizelge 7.4'teki gibi olur.

Çizelge 7.4 Resimdeki renklerin Huffman kodları

0	0	1	1	0	0	0	1	1	1

Kodlanmış ağaçta bazı özellikler bulunmaktadır: ağaçtaki 0 ve 1'lerin sayısı eşittir. Ağacın herhangi bir noktasında 1'lerin sayısı 0'ların sayısını geçemez. Ağaçtaki kod sayısı her zaman çifttir (2N-2).

Sıkıştırılmış resmin başlığının boyutunun ne olacağı, resimdeki renk sayısına bağlı olarak hesaplanabilmektedir. Başlığın neler içermesi gerektiği önceki paragraflarda anlatılmıştı. Buna göre N adet renkten oluşan bir resim sıkıştırıldığındaki resim başlığı, resimde kaç renk olduğunu (1 bayt), resimdeki renkleri (bir renk 3 bayt ile ifade edildiği için N renk için N*3* bayt), resmin genişlik ve yükseklik bilgisini (2*4 bayt), ikili ağacı (2*N-2 bit) içermelidir. Bu şartlar altında başlığın boyutu bit cinsinden "70 + 26 N" olmaktadır. 16 renk için 486 bit'e ihtiyaç duyulduğu açık olarak görülmektedir.

Gövdenin yazılması aşamasında, sıkıştırılacak verideki her sembol yerine, o sembole karşılık düşen Huffman kodu sıkıştırılmış dosyaya yazılmaktadır. Resim sıkıştırma için yorumlanırsa, resimdeki pikseller yerine o pikselin rengine karşılık düşen Huffman kodları sıkıştırılmış dosyaya yazılmalıdır. Huffman kodları değişken uzunlukludur. Dolayısıyla kodlar dosyaya yazılırken 1 bayt (8 bit)'lık paketler halinde birleştirilerek yazılmakdadır.

7.6 Fark Kodlamasının Resime Uygulanması

Fark kodlaması bir resime uygulanırken, ilk pikselin değeri değiştirilmeden bırakılır. İkinci piksel ilk pikselden çıkarılarak ikinci pikselin yerine fark değeri yazılır. Her pikselin değeri

bir önceki pikselin fark almadan önceki değerinden çıkarılarak fark bulunup o pikselin yerine yazılır. Yeni oluşan resme sıkıştırılmak üzere Huffman kodlaması uygulanır.

Dört farklı renkten oluşan bir resme fark kodlaması uygulandığında 4*2 -1 adet fark kodu elde edilir. Dört renk 0, 1, 2, 3 ile ifade edilirken oluşabilecek fark kodları -3, -2, -1, 0, 1, 2, 3 ile ifade edilir. Huffman kodlama bu fark kodu uygulanmış resme uygulanırken önceki bölümde anlatıldığı gibi aynen uygulanır. Huffman kodlamasında kullanılan semboller bu fark kodları, sembol sıklıkları ise bu kodların yeni resmin içinde ne kadar sayıda bulunduğu bilgisidir. Sembol sayısı 2*N-1 fark kodu sayısı olacaktır. Huffman kodlaması aynen uygulansa da başlığın içeriği değişecek ve boyutu artacaktır.

Fark kodlaması uygulandıktan sonra Huffman kodlaması ile sıkıştırılan resmin başlığının boyutu yine resimdeki renk sayısına bağlı olarak hesaplanabilmektedir. Başlık, resimde kaç renk olduğunu (1 bayt), resimdeki renkleri (bir renk 3 bayt ile ifade edildiği için N renk için N*3*1 bayt), resmin genişlik ve yükseklik bilgisini (2*4 bayt), fark kodlarını (2*N-1 bayt), ikili ağacı (2*(2*N-1)-2 bit) içermelidir. Bu şartlar altında başlığın boyutu bit cinsinden "60 + 44 N" olmaktadır. 16 renk bulunan bir resim fark kodlaması ve Huffman kodlaması kullanarak sıkıştırıldığında, başlık için 764 bit'e ihtiyaç duyulur.

Huffman kodlamasından önce fark kodlaması uygulandığında başlığın boyutu artmaktadır. Fakat fark kodlaması uygulanan resimlerde daha iyi sıkıştırma sonucu elde edilmiştir. Bunun sebebi resimlerdeki renk geçişlerinin yumuşak olmasından kaynaklanmaktadır. Birbirine komşu piksellerin arasındaki farklar sınır geçişleri haricinde, düşük olmaktadır. Gökyüzündeki renkler mavinin tonlarında iken, bir ağacın yapraklarının yeşilin tonlarında olması örnek olarak verilebilir.

7.7 Sıkıştırılmış Resmin Yeniden Yapılandırılması

Huffman kodlaması uygulanarak elde edilen resim iletildikten sonra resmin yeniden yapılandırılarak kodlamadan önceki orjinal haline döndürülmesi aşaması gelmektedir. Bu aşamada öncelikle sıkıştırılmış resimdeki başlık bilgisi okunur. Başlıktaki ikili kodlanmış ağaç kullanılarak Huffman kod ağacı elde edilir. Renk sembollerinin ağaçtaki sırası soldan sağa olduğu bilgisi kullanılarak sembollerin ağaçtaki tam yerleri (indisleri) belirlenir. Daha sonraki aşamada gövdeden bir bayt okunur. Her bayt 8 bitlik bit dizisine çevrilir. Huffman ağacının başlangıç düğümünden başlanarak, bit değeri "0" için ağacın sol dalından, "1" için sağ dalından ilerlenir. Yaprağa gelindiğinde o yaprağa düşen sembol değeri yeniden yapılandırılan resimde ilk piksel yerine yazılır. Ağacın başlangıç düğümüne geri dönülür. Bit

dizisinde kullanılmamış bitler kaldıysa kullanılır, aksi halde gövdeden bir bayt daha okunur ve işlemler sıkıştırılmış dosyanın sonuna gelene kadar ya da yeniden yapılandırılan resimdeki piksel sayısı başlıktaki genişlik*yükseklik sayısı olana kadar tekrarlanır. Eğer resme fark kodlaması uygulandıysa elde edilen veri fark kodlaması çözücüsünden de geçirilerek resmin yeniden yapılandırılması işlemi tamamlanmış olmaktadır.

8. KODLAMA SONUÇLARI

Bu çalışmada, görüntü sıkıştırma aşamasında resimlere doğrudan Huffman kodlaması uygulanmış ya da ön işlem olarak resimler fark kodlayıcıdan geçirilerek Huffman kodlaması uygulanmıştır. Dördüncü bölümde anlatılan kuantalama ve beşinci bölümde anlatılan hiyerarşik kuantalama yöntemleri kullanılarak 256 renk resimler 16, 64 ve 100 renge indirilmiş, elde edilen resimler görüntü sıkıştırma adımında kullanılmıştır. Vektör kuantalama yöntemlerinin 16, 64 ve 100 renge indirilmiş Lenna resmi için elde edilen hata ve sıkıştırma oranları sırasıyla Çizelge 8.1, Çizelge 8.2 ve Çizelge 8.3'te gösterilmiştir. Fark kodlaması uygulandıktan sonra Huffman kodlaması uygulanmasının Huffmanın doğrudan uygulanmasından daha yüksek sıkıştırma oranı sağladığı görülmüştür. Lenna resmi için fark kodlaması uygulandığında, komşuluk ilişkisi veya NGas güncelleme yöntemi kullanılan kuantalama yöntemlerinde sıkıştırma sonuçlarının daha iyi olduğu görülmüştür.

Çizelge 8.1 Lenna resminin16 renge indirilmesinde kullanılan kuantalama yöntemleri, hata oranları ve Huffman uygulandıktan sonra elde edilen sıkıştırma oranları

VQ Yöntemi	Hata (MSE)	Sıkıştırma Oranı (%)	
		Huffman	DC & Huffman
kMeans_Batch	24,98	57,79	71,22
kMeans_Online	20,93	58,23	71,65
kMeans_Ngas	23,06	56,05	74,64
SOM_1D_Std	20,92	57,50	74,30
SOM_1D_Ngas	23,16	56,60	74,89
SOM_2D_Std	20,93	57,14	74,49
SOM_2D_Ngas	26,36	57,15	74,44

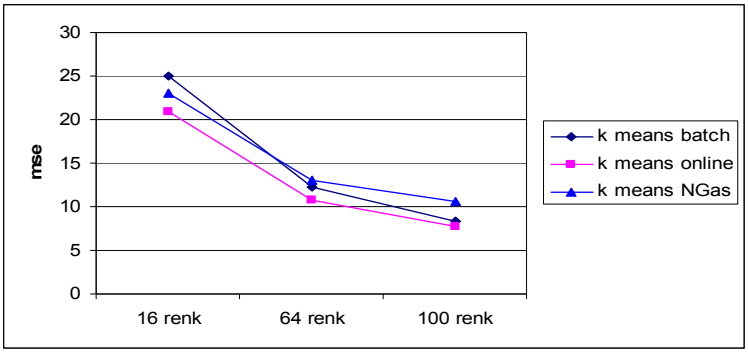
Çizelge 8.2 Lenna resminin 64 renge indirilmesinde kullanılan kuantalama yöntemleri, hata oranları ve Huffman uygulandıktan sonra elde edilen sıkıştırma oranları

VQ Yöntemi	Hata (MSE)	Sıkıştırma Oranı (%)	
		Huffman	DC & Huffman
kMeans_Batch	12,25	39,34	44,69
kMeans_Online	10,8	39,87	46,77
kMeans_Ngas	13,01	36,89	48,67
SOM_1D_Std	11,31	35,47	50,60
SOM_1D_Ngas	13,18	34,78	48,08
SOM_2D_Std	11,63	36,92	56,94
SOM_2D_Ngas	15,32	36,31	56,72

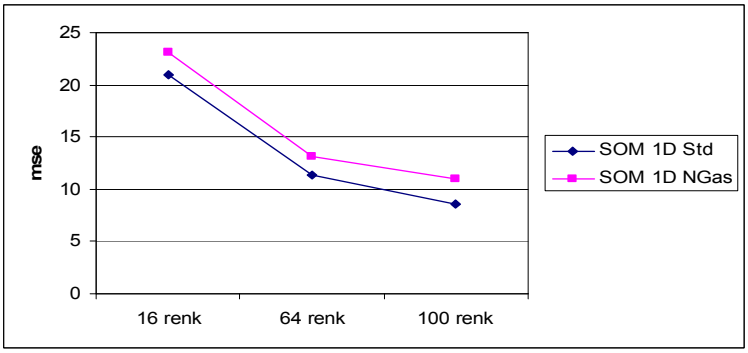
Çizelge 8.3 Lenna resminin 100 renge indirilmesinde kullanılan kuantalama yöntemleri, hata oranları ve Huffman uygulandıktan sonra elde edilen sıkıştırma oranları

VQ Yöntemi	Hata (MSE)	Sıkıştırma Oranı (%)	
		Huffman	DC & Huffman
kMeans_Batch	8,29	30,17	31,82
kMeans_Online	7,73	30,64	31,80
kMeans_Ngas	10,64	30,40	39,92
SOM_1D_Std	8,59	28,53	46,38
SOM_1D_Ngas	10,93	29,85	38,45
SOM_2D_Std	8,94	29,40	50,10
SOM_2D_Ngas	13	29,55	47,01

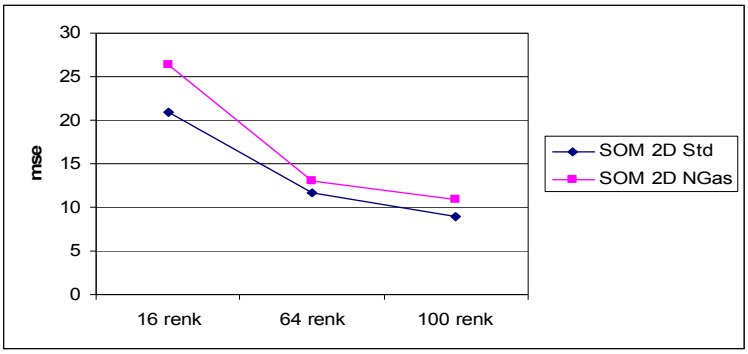
Şekil 8.1, Şekil 8.2 ve Şekil 8.3'te 256 renk Lenna resminin 16, 64 ve 100 renge indirilmesi sırasında kullanılan kuantalama yöntemlerinin hata oranlarının renk sayısına göre değişimi grafiksel olarak gösterilmiştir.



Şekil 8.1 k-Means kuantalama algoritmasının farklı güncelleme yöntemleriyle uygulandığı Lenna resminin 16, 64, 100 renge indirilmesi sonucu oluşan hataların değişimi



Şekil 8.2 1D SOM kuantalama algoritmasının farklı güncelleme yöntemleriyle uygulandığı Lenna resminin 16, 64, 100 renge indirilmesi sonucu oluşan hataların değişimi



Şekil 8.3 2D SOM kuantalama algoritmasının farklı güncelleme yöntemleriyle uygulandığı Lenna resminin 16, 64, 100 renge indirilmesi sonucu oluşan hataların değişimi

Hiyerarşik kuantalama, her seviyede aynı ya da farklı kuantalama yöntemi kullanılarak 256 renk Lenna resmine uygulanmıştır. Hiyerarşik kuantalamada her seviyede kullanılan yöntemler, 16, 64 ve 100 renge indirilmiş Lenna resmi için elde edilen hata ve sıkıştırma oranları sırasıyla Çizelge 8.4, Çizelge 8.5 ve Çizelge 8.6'da gösterilmiştir.

Çizelge 8.4 Lenna resminin16 renge indirilmesinde her seviyede kullanılan kuantalama yöntemleri, hata oranları ve Huffman uygulandıktan sonra elde edilen sıkıştırma oranları

VQ Yöntemi		**Hata (MSE)**		**Sıkıştırma Oranı**	
Seviye 0	**Seviye 1**	**Seviye 0**	**Seviye 1**	**Huffman**	**DC & Huffman**
kMeans_Online	kMeans_Online	39,07	26,37	64,62	76,29
	kMeans_Ngas	39,08	25,37	56,01	70,49
	SOM_1D_Std	38,95	21,34	57,67	71,30
	SOM_1D_Ngas	38,72	23,81	54,77	72,48
kMeans_Ngas	kMeans_Online	41,60	26,10	65,73	76,11
	kMeans_Ngas	41,73	25,77	54,64	71,05
	SOM_1D_Std	42,35	21,53	57,79	73,56
	SOM_1D_Ngas	42,53	23,45	54,52	71,15
SOM_1D_Std	kMeans_Online	38,85	26,65	63,86	75,39
	kMeans_Ngas	38,70	25,01	54,96	68,24
	SOM_1D_Std	38,62	21,59	56,98	74,01
	SOM_1D_Ngas	38,65	23,79	55,20	71,69
SOM_1D_Ngas	kMeans_Online	42,78	26,63	64,04	77,13
	kMeans_Ngas	42,36	26,11	54,45	72,11
	SOM_1D_Std	42,36	21,53	56,23	72,77
	SOM_1D_Ngas	42,36	23,27	55,09	72,55

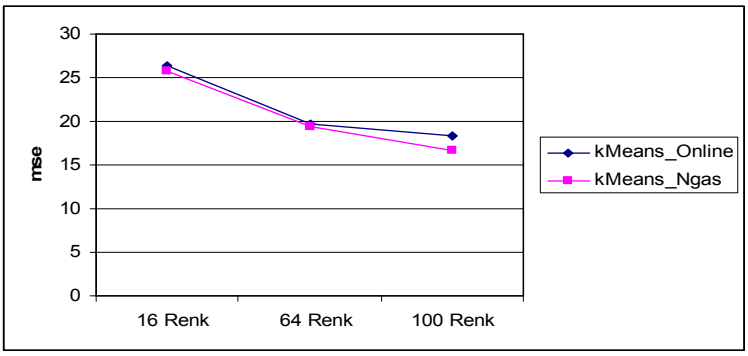
Çizelge 8.5 Lenna resminin 64 renge indirilmesinde her seviyede kullanılan kuantalama yöntemleri, hata oranları ve Huffman uygulandıktan sonra elde edilen sıkıştırma oranları

VQ Yöntemi		**Hata (MSE)**		**Sıkıştırma Oranı**	
Seviye 0	**Seviye 1**	**Seviye 0**	**Seviye 1**	**Huffman**	**Huffman & DC**
kMeans_Online	kMeans_Online	30,73	19,69	51,60	63,44
	kMeans_Ngas	28,85	20,52	43,04	55,00
	SOM_1D_Std	28,11	11,34	35,45	48,77
	SOM_1D_Ngas	28,88	13,37	37,09	50,55
kMeans_Ngas	kMeans_Online	32,23	17,65	47,69	58,07
	kMeans_Ngas	32,03	19,33	41,38	55,52
	SOM_1D_Std	31,88	11,27	35,97	52,75
	SOM_1D_Ngas	31,96	13,29	35,43	52,24
SOM_1D_Std	kMeans_Online	28,09	19,94	52,40	62,79
	kMeans_Ngas	28,48	19,98	43,83	56,02
	SOM_1D_Std	28,2	11,52	34,73	52,08
	SOM_1D_Ngas	28,21	13,2	35,30	49,98
SOM_1D_Ngas	kMeans_Online	30,77	19,42	49,19	60,62
	kMeans_Ngas	30,77	19,79	40,68	54,15
	SOM_1D_Std	31,34	11,85	35,17	52,86
	SOM_1D_Ngas	30,77	13,59	36,08	53,34

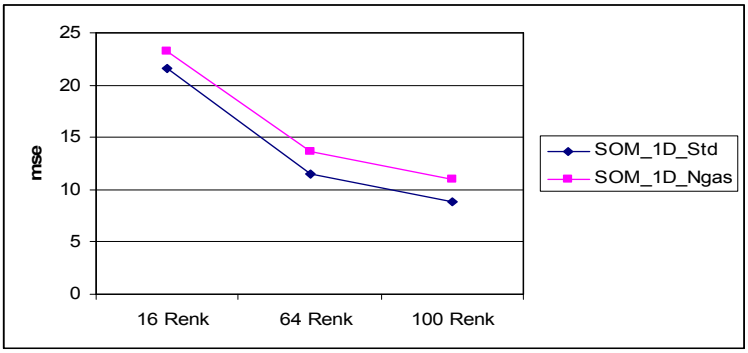
Çizelge 8.6 Lenna resminin100 renge indirilmesinde her seviyede kullanılan kuantalama yöntemleri, hata oranları ve Huffman uygulandıktan sonra elde edilen sıkıştırma oranları

VQ Yöntemi		**Hata (MSE)**		**Sıkıştırma Oranı**	
Seviye 0	**Seviye 1**	**Seviye 0**	**Seviye 1**	**Huffman**	**Huffman & DC**
kMeans_Online	kMeans_Online	26,76	18,32	51,59	61,26
	kMeans_Ngas	26,09	17,02	38,01	46,53
	SOM_1D_Std	25,89	9,36	29,74	42,88
	SOM_1D_Ngas	25,9	10,97	31,99	42,99
kMeans_Ngas	kMeans_Online	28,99	17,52	52,68	62,31
	kMeans_Ngas	29,24	16,61	36,70	47,56
	SOM_1D_Std	27,98	9,1	30,70	47,60
	SOM_1D_Ngas	21,83	11,01	28,95	43,91
SOM_1D_Std	kMeans_Online	25,76	17,72	49,80	59,67
	kMeans_Ngas	25,73	15,65	38,08	49,79
	SOM_1D_Std	25,42	8,82	30,25	45,01
	SOM_1D_Ngas	25,91	10,89	28,60	45,17
SOM_1D_Ngas	kMeans_Online	29,13	18,61	51,65	61,61
	kMeans_Ngas	29,33	16,62	35,71	49,10
	SOM_1D_Std	29,13	9,15	30,03	44,68
	SOM_1D_Ngas	29,13	11,02	29,36	45,59

Şekil 8.4, Şekil 8.5 ve Şekil 8.6'te 256 renk Lenna resminin 16, 64 ve 100 renge indirilmesi sırasında kullanılan hiyerarşik kuantalama yöntemlerinin hata oranlarının renk sayısına göre değişimi grafiksel olarak gösterilmiştir.



Şekil 8.4 2D Hiyerarşik kuantalama algoritmasının farklı güncelleme yöntemleriyle uygulandığı Lenna resminin 16, 64, 100 renge indirilmesi sonucu oluşan hataların değişimi



Şekil 8.5 Hiyerarşik kuantalama algoritmasının farklı güncelleme yöntemleriyle uygulandığı Lenna resminin 16, 64, 100 renge indirilmesi sonucu oluşan hataların değişimi

9. SONUÇ

Kullanılan kuantalama yöntemleri hız açısından karşılaştırıldığında, NGas güncelleme yönteminin tüm merkezleri güncellemesi ve üs alma fonksiyonu kullanması sebebiyle algoritmanın hızını azalttığı gözlemlenmiştir. VQ algoritmalarına hiyerarşik yapı uygulandığında, kazanan merkezi bulmak için harcanan süre kısaldığından, algoritmanın hızı artmıştır. Aynı zamanda NGas güncelleme yöntemi kullanan algoritmaların düşük döngü sayılarında yakınsadığı görülmüştür. Batch k-Means algoritması hem hızlı bir algoritma olup hem de düşük döngü sayılarında yakınsamıştır.

Hata açısından karşılaştırıldığında VQ algoritmalarında NGas güncelleme yönteminin kullanılması hata değerini artırmıştır. SOM algoritması ve NGas güncelleme yöntemi kullanan kuantalama algoritmalarının (hiyerarşik yapılılar dahil) renk kuantalama sonucunda elde edilen renkler sıralı olarak karşımıza çıkmıştır. Hiyerarşik yapı kullanmak algoritmaların hata değerini artırmıştır.

16 renk sayısı için sıkıştırma yöntemi olarak sadece Huffman kodlaması uygulandığında %58,23 sıkıştırma oranıyla online güncellenen k-Means algoritması en iyi sonucu vermiş, aynı zamanda hata değeri de 20,93 olarak en iyi yöntemlerden biri olmuştur. Fark kodlaması ve Huffman kodlaması beraber uygulandığında ise %74,89 ile NGas yöntemiyle güncellenen 1D SOM en iyi sonucu vermiş fakat hata değeri 23,16 olarak elde edilmiştir.

16 renk sayısı için, hiyerarşik yapının son seviyesinde online güncellenen k-Means kuantalama algoritması kullanıldığında ve sadece Huffman kodlaması uygulandığında ortalama %64,56; fark kodlaması ve Huffman kodlaması uygulandığında ise ortalama %76,23 olarak en iyi sıkıştırma sonuçları ile birlikte ortalama 26,44 ile en yüksek hata değeri elde edilmiştir. En düşük hata değeri ortalama 21,50 olan geleneksel güncellenen 1D SOM yönteminde %72,91 ortalama sıkıştırma oranı ile hem hata değeri hem de sıkıştırma performansı bakımından uygun bir sonuç elde edilmiştir.

Sıkıştırma oranları açısından VQ algoritma uygulamasından sonra, Huffman kodlaması doğrudan uygulandığında, tüm VQ algoritmaları için birbirlerine yakın sıkıştıma oranları elde edilmiştir. Huffman kodlaması öncesinde fark kodlaması uygulandığında ise sıralı renk dizesi üreten SOM ve NGas güncelleme yöntemi kullanılan algoritmalar ile daha iyi sıkıştırma oranları elde edilmiştir. Sadece Huffman kodlaması yöntemi kullanımı yerine fark kodlaması ile Huffman kodlaması kullanımı tüm kuantalama yöntemleri için daha iyi sıkıştırma oranı elde edilmesini sağlamıştır. Bunun nedeni resimlerde birbirine yakın piksellerin benzer

renklerde olmasıdır. Bu özellik resim sıkıştırmada yararlanılan çok önemli bir özelliktir. Hiyerarşik yapılı online güncellenen k-Means algoritması kullanıldığında en iyi sıkıştırma oranı elde edilmesine rağmen, hata değeri çok yüksek çıkmıştır. Hata değerinin çok yüksek olması sıkıştırma oranının yüksek olmasını gölgede bırakmıştır.

NGas güncelleme yönteminin k-Means ve SOM algoritmasında kullanılması, renk kuantalama ve sıkıştırmada, geleneksel yöntemlere kıyasla istenen başarıyı gösterememiş olmasına rağmen anlamlı bir sonuca ulaşılabilmesi için farklı uygulamalar üzerinde de denenmelidir.

KAYNAKLAR

Albayrak S. "A Comparison of 1-D and 2-D Self Organizing Feature Map Algorithm on Color Image Quantization", 9'th International Conference on :Neural Information Processing, November 2002, Singapure

Ancona F., Ridella S., Rovetta S. ve Zunino R., "On the Impotance of Sorting in 'Neural Gas' Training of Vector Quantizers". IEEE, 1997

Barballho José M., A. Duarte D. Netto, José A. F. Costa, ve Márcio L. A. Netto, "Hierarchical SOM Applied to Image Compression", Neural Networks, 2001 IEEE

Costa, J.A.F., Dória Neto, A.D. ve Netto, M.L.A. (2003). "A New Structured Self-Organizing Map with Dynamic Growth Applied to Image Compression", *Anais do VI Congresso Brasileiro de Redes Neurais*, São Paulo, Junho de 2003, pp. 457-462

Duda R. O., Hart P. E. ve Stork D.G., Pattern Classification 2nd Edition, Wiley-Interscience Publication, USA.2001

Endo M., Ueno M., Tanabe T. ve Yamamoto M., "Clustering method using self-organizing map", Neural Networks for Signal Processing X, 2000 IEEE

Gray R.M., "Vector quantization". IEEE ASSP Mag., vol 1, no. 2, pp. 4-29, April 1984

Haykin S., Neural Networks: A comprehensive Foundation, 2nd ed. Englewood Cliffs, NJ:Prentice-Hall, 1998

Sayood Khalid, Introduction to Data Compression, Morgan Kauffman, 1996

Kohonen T. "Self-organization and Associative Memory" 3 rd Ed., 1989, springer Verlag

Laha A.,Pal N.R. ve Chanda B., "Design of Vector Quantizer for Image Compression Using Self-Organizing Feature Map and Surface Fitting", IEEE Transactions on Image Processing Vo.13 No.10 October 2004

Linde Y., Buzo A. ve Gray R. M., "An Algorithm for vector quantizater design", IEEE Trans. Commun., vol. COM-28, pp. 84-95, Jan. 1980

Lookabaugh T., Lindbergh D. ve Richard L.Baker, Digital Compression for Multimedia, Jerry D.Gibson,Toby Berger, Morgan Kauffman, 1998

Luo M., Yu-Fei Ma ve Hong-Jiang Zhang, "A spatial constrained K-means approach to image segmentation", Information, Communications and Signal Processing, 2003 IEEE

Martinez T. M., Berkovich S. G ve Schulten K. J., " 'Neural Gas' Network for Vector Quantization and its Application to Time Series Prediction". IEEE Trans. Neural Networks 4, pp. 558-568, 1993

Qin A. K. ve Suganthan P. N., "A Robust Neural Gas Algorithm for Clustering Analysis". IEEE, icisip, pp. 342-347, 2004

Zhang B., Fu M. ve Yan H., "Application of Neural 'Gas' Model in Image Compression". IEEE, pp. 918-921, 1998

Zhang B., Fu M., Yan H., “Handwritten Signature Verification based on Neural ‘Gas’ Based Vector Quantization”. IEEE, pp. 1862-1864, 1998

INTERNET KAYNAKLARI

[1]http://www.dice.ucl.ac.be/~verleyse/lectures/elec2870/slides/vector%20quantization%20BW%202spp.pdf

[2] http://kom.aau.dk/~fn/research/vq_tour.pdf

[3] http://encyclopedia.thefreedictionary.com/Huffman%20tree

[4] http://www.wolframscience.com/reference/notes/1069b

[5] http://www.answers.com/topic/Huffman-coding

[6] http://www.data-compression.info/Algorithms/EC/index.htm

[7] http://en.wikipedia.org/wiki/Entropy_coding

[8]http://citeseer.ist.psu.edu/cache/papers/cs/809/http:zSzzSzwww.ics.uci.eduzSz~danzSzpubszSzDataCompression.pdf/lelewer87data.pdf

EKLER

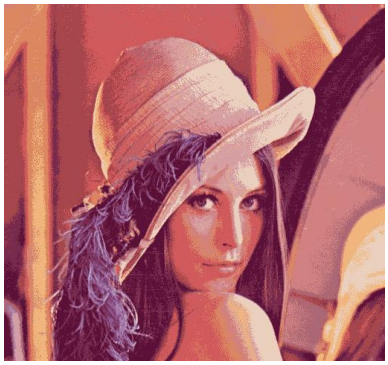
Ek 1 Kullanılan Orijinal Resim
Ek 2 Lenna resmi için 16 renk sonuçları
Ek 3 Lenna resmi için 64 renk sonuçları
Ek 4 Lenna resmi için 100 renk sonuçları

Ek 1 – Kullanılan Orijinal Resim

Şekil Ek 1.1 256 renk orijinal Lenna resmi

Ek 2 –Lenna resmi için 16 renk sonuçları

Vektör kuantalama algoritmaları sonuçları



E_{MSE} = 21,28

Şekil Ek 2.1 k-Means batch algoritması için resim, renk ve hata değeri sonuçları

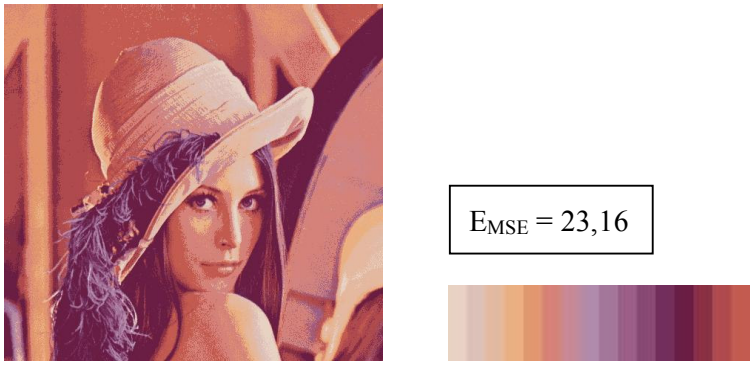
E_{MSE} = 20,86

Şekil Ek 2.2 k-Means online algoritması için resim, renk ve hata değeri sonuçları

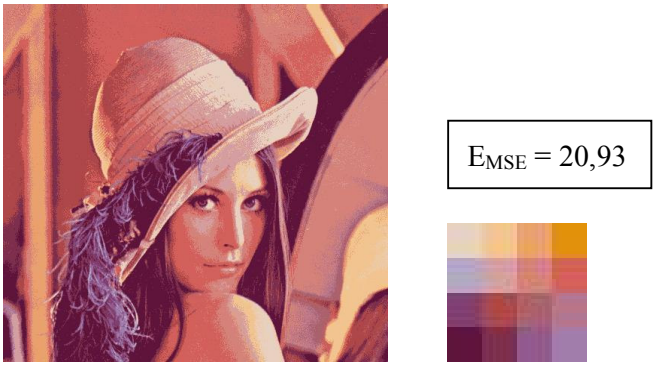
E_{MSE} = 23,04

Şekil Ek 2.3 NGas k-Means algoritması için resim, renk ve hata değeri sonuçları

Şekil Ek 2.4 Geleneksel 1D SOM algoritması için resim, renk ve hata değeri sonuçları



Şekil Ek 2.5 NGas 1D SOM algoritması için resim, renk ve hata değeri sonuçları



Şekil Ek 2.6 Geleneksel 2D SOM algoritması için resim, renk ve hata değeri sonuçları

E_{MSE} = 26,36

Şekil Ek 2.7 NGas 2D SOM algoritması için resim, renk ve hata değeri sonuçları

Hiyerarşik vektör kuantalama algoritmaları sonuçları

E_{MSE} = 39,07

(a)

E_{MSE} = 26,37

(b)

Şekil Ek 2.8 k-Means online algoritması için resim, renk ve hata değeri sonuçları
(a) Seviye 0 (b) Seviye 1

E_{MSE} = 41,73

(a)

E_{MSE} = 25,77

(b)

Şekil Ek 2.9 NGas k-Means online algoritması için resim, renk ve hata değeri sonuçları
(a) Seviye 0 (b) Seviye 1

E_{MSE} = 38,62

(a)

E_{MSE} = 21,59

(b)

Şekil Ek 2.10 Geleneksel 1D SOM algoritması için resim, renk ve hata değeri sonuçları
(a) Seviye 0 (b) Seviye 1

E_{MSE} = 42,36

(a)

E_{MSE} = 23,36

(b)

Şekil Ek 2.11 NGas 1D SOM algoritması için resim, renk ve hata değeri sonuçları
(a) Seviye 0 (b) Seviye 1

Ek 3 –Lenna resmi için 64 renk sonuçları

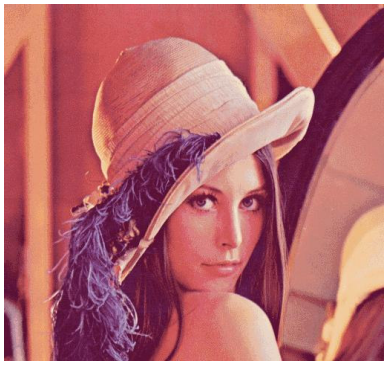
Vektör kuantalama algoritmaları sonuçları

E_{MSE} = 12,25

Şekil Ek 3.1 k-Means batch algoritması için resim, renk ve hata değeri sonuçları

E_{MSE} = 10,80

Şekil Ek 3.2 k-Means online algoritması için resim, renk ve hata değeri sonuçları



E_{MSE} = 13,01

Şekil Ek 3.3 NGas k-Means algoritması için resim, renk ve hata değeri sonuçları

E_{MSE} = 11,31

Şekil Ek 3.4 Geleneksel 1D SOM algoritması için resim, renk ve hata değeri sonuçları

E_{MSE} = 13,18

Şekil Ek 3.5 NGas 1D SOM algoritması için resim, renk ve hata değeri sonuçları

E_{MSE} = 11,63

Şekil Ek 3.6 Geleneksel 2D SOM algoritması için resim, renk ve hata değeri sonuçları

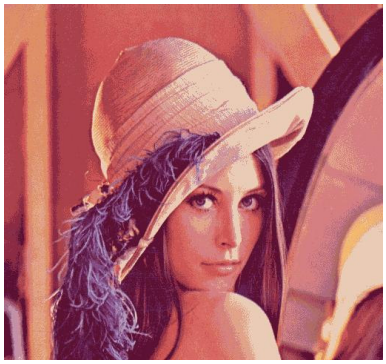
E_{MSE} = 15,33

Şekil Ek 3.7 NGas 2D SOM algoritması için resim, renk ve hata değeri sonuçları

Hiyerarşik vektör kuantalama algoritmaları sonuçları

E_{MSE} = 30,73

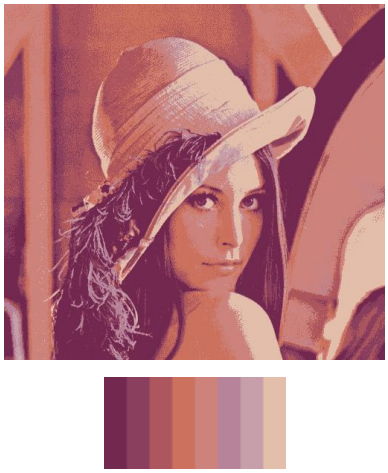
(a)



E_{MSE} = 19,69

(b)

Şekil Ek 3.8 k-Means online algoritması için resim, renk ve hata değeri sonuçları
(a) Seviye 0 (b) Seviye 1



E_{MSE} = 32,04

(a)

E_{MSE} = 19,33

(b)

Şekil Ek 3.9 NGas k-Means algoritması için resim, renk ve hata değeri sonuçları
(a) Seviye 0 (b) Seviye 1

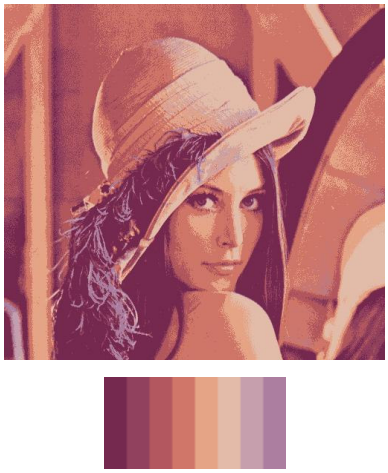
E_{MSE} = 28,20

(a)

E_{MSE} = 11,52

(b)

Şekil Ek 3.10 Geleneksel 1D SOM algoritması için resim, renk ve hata değeri sonuçları
(a) Seviye 0 (b) Seviye 1



E_{MSE} = 30,77

(a)

E_{MSE} = 13,59

(b)

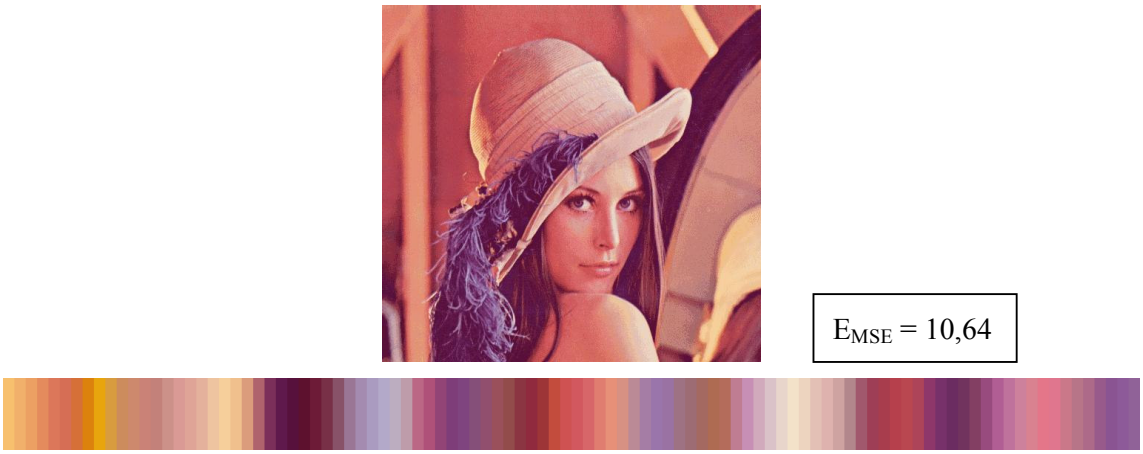
Şekil Ek 3.11 NGas 1D SOM algoritması için resim, renk ve hata değeri sonuçları
(a) Seviye 0 (b) Seviye 1

Ek 4 –Lenna resmi için 100 renk sonuçları

Vektör kuantalama algoritmaları sonuçları

Şekil Ek 4.1 k-Means batch algoritması için resim, renk ve hata değeri sonuçları

Şekil Ek 4.2 k-Means online algoritması için resim, renk ve hata değeri sonuçları



Şekil Ek 4.3 NGas k-Means algoritması için resim, renk ve hata değeri sonuçları

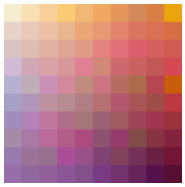
E_{MSE} = 8,59

Şekil Ek 4.4 Geleneksel 1D SOM algoritması için resim, renk ve hata değeri sonuçları

E_{MSE} = 10,93

Şekil Ek 4.5 NGas 1D SOM algoritması için resim, renk ve hata değeri sonuçları

E_{MSE} = 8,94



Şekil Ek 4.6 Geleneksel 2D SOM algoritması için resim, renk ve hata değeri sonuçları

E_{MSE} = 13,00

Şekil Ek 4.7 NGas 2D SOM algoritması için resim, renk ve hata değeri sonuçları

Hiyerarşik vektör kuantalama algoritmaları sonuçları

E_{MSE} = 26,76

(a)

E_{MSE} = 18,32

(b)

Şekil Ek 4.8 k-Means online algoritması için resim, renk ve hata değeri sonuçları
(a) Seviye 0 (b) Seviye 1

E_{MSE} = 29,24

(a)

E_{MSE} = 16,61

(b)

Şekil Ek 4.9 NGas k-Means algoritması için resim, renk ve hata değeri sonuçları
(a) Seviye 0 (b) Seviye 1

E_{MSE} = 25,42

(a)

E_{MSE} = 8,82

(b)

Şekil Ek 4.10 Geleneksel 1D SOM algoritması için resim, renk ve hata değeri sonuçları
(a) Seviye 0 (b) Seviye 1

E_{MSE} = 29,13

(a)

E_{MSE} = 11,02

(b)

Şekil Ek 4.11 NGas 1D SOM algoritması için resim, renk ve hata değeri sonuçları
(a) Seviye 0 (b) Seviye 1

ÖZGEÇMİŞ

Doğum tarihi	01.04.1978	
Doğum yeri	Bakırköy	
Lise	1991-1995	50. Yıl İnsa Lisesi
Lisans	1996-2001	İstanbul Teknik Üniversitesi Elk-Elektronik Fak. Elektronik ve Haberleşme Mühendisliği Bölümü
Yüksek Lisans	2001-2006	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği

İş Tecrübesi

2004-.....	Garanti Teknoloji A.Ş., Temel Bankacılık, Kredi Kartları Bölümü, Yazılım Mühendisi
2001-2004	Yıldız Teknik Üniversitesi, Elk.-Elektronik Fak. Bilgisayar Mühendisliği Bölümü, Bilgisayar Bilimleri Anabilim Dalı, Araştırma Görevlisi