

## JPEG

JPEG veya JPG [2], özellikle dijital fotoğrafçılıkla üretilen görüntüler için dijital görüntüler için yaygın olarak kullanılan bir kayıplı sıkıştırma yöntemidir. Sıkıştırma derecesi, depolama boyutu ve görüntü kalitesi arasında seçilebilir bir değiş tokuşa izin verecek şekilde ayarlanabilir. JPEG tipik olarak, görüntü kalitesinde çok az algılanabilir kayıpla 10:1 sıkıştırma elde eder. [3] 1992'de piyasaya sürülmesinden bu yana, JPEG, dünyada en yaygın kullanılan görüntü sıkıştırma standardı [4] [5] ve 2015 itibarıyla her gün üretilen birkaç milyar JPEG görüntü ile en yaygın kullanılan dijital görüntü formatı olmuştur. [6]

"JPEG" terimi, 1992'de standardı oluşturan Joint Photographic Experts Group için bir başlangıç/kısaltmadır. JPEG'nin temeli, ilk olarak Nasir Ahmed tarafından 1972'de. [7] önerilen kayıplı bir görüntü sıkıştırma tekniği olan ayrık kosinüs dönüşümüdür (DCT) [1]. JPEG, dijital görüntülerin ve dijital fotoğrafların İnternette ve daha sonra sosyal medyada yaygınlaşmasından büyük ölçüde sorumluydu. [8]

JPEG sıkıştırması, bir dizi görüntü dosyası biçiminde kullanılır. JPEG/Exif, dijital kameralar ve diğer fotoğrafik görüntü yakalama aygıtları tarafından kullanılan en yaygın görüntü formatıdır; JPEG / JFIF ile birlikte, fotoğrafik görüntüleri World Wide Web'de saklamak ve iletmek için en yaygın formattır. [9] Bu biçim varyasyonları genellikle ayırt edilmez ve kısaca JPEG olarak adlandırılır.

JPEG için MIME ortam türü, JPEG görüntüleri yüklerken MIME tipi görüntü / pjpeg sağlayan eski Internet Explorer sürümleri dışında, resim / jpeg'dir. [10] JPEG dosyaları genellikle .jpg veya .jpeg dosya adı uzantısına sahiptir. JPEG / JFIF maksimum 65,535 × 65,535 piksel [11] görüntü boyutunu destekler, dolayısıyla 1: 1 en boy oranı için 4 gigapiksele kadar çıkar. 2000 yılında, JPEG grubu halefi olması amaçlanan bir format olan JPEG 2000'i tanıttı, ancak baskın görüntü standardı olarak orijinal JPEG'nin yerini alamadı. [12]

### Arka Plan

1992'de yayınlanan orijinal JPEG spesifikasyonu, CCITT (şimdi ITU-T, ITU-T Study Group 16) ve Joint Photographic Experts Group tarafından atıfta bulunulan daha önceki çeşitli araştırma makaleleri ve patentlerden süreçleri uygular. [1] JPEG'in kayıplı sıkıştırma algoritmasının ana temeli, ilk kez 1972'de Nasir Ahmed tarafından bir görüntü sıkıştırma tekniği olarak önerilen ayrık kosinüs dönüşümüdür (DCT) [1] [13]. [7] [13] Ahmed, 1973'te Kansas Eyalet Üniversitesi'nden T. Natarajan ve Arlington'daki Texas Üniversitesi'nden K. R. Rao ile pratik bir DCT algoritması geliştirdi. [7] Onların çığır açan 1974 makalesi [14], Wen-Hsiung Chen, C.H. Smith ve S.C. Fralick, hızlı bir DCT algoritmasını tanımlayan [1] [15], ayrıca N.J. Narasinha ve S.C. Fralick'in 1978 tarihli bir makalesi ve B.G. Lee. [1] Spesifikasyon ayrıca Wen-Hsiung Chen ve W.K.'nin 1984 tarihli bir makalesine atıfta bulunmaktadır. Pratt, [1] [16] ve David A. Huffman'ın Huffman kodlama algoritması için 1952 tarihli makalesi. [1]

JPEG spesifikasyonu, birkaç şirketin patentlerine atıfta bulunur. Aşağıdaki patentler, aritmetik kodlama algoritmasının temelini oluşturmuştur. [1]

### **IBM**

- ABD Patenti 4,652,856 - 4 Şubat 1986 - Kottappuram M.A. Mohiuddin ve Jorma J. Rissanen - Çarpma içermeyen çoklu alfabe aritmetik kodu
- ABD Patenti 4,905,297 - 27 Şubat 1990 - G. Langdon, J.L. Mitchell, W.B. Pennebaker ve Jorma J. Rissanen - Aritmetik kodlama kodlayıcı ve kod çözücü sistemi
- ABD Patenti 4,935,882 - 19 Haziran 1990 - W.B. Pennebaker ve J.L. Mitchell - Aritmetik kodlayıcılar için olasılık uyarlaması

### **Mitsubishi Electric**

- JP H02202267 (1021672) - 21 Ocak 1989 - Toshihiro Kimura, Shigenori Kino, Fumitaka Ono, Masayuki Yoshida - Kodlama sistemi
- JP H03247123 (2-46275) - 26 Şubat 1990 - Fumitaka Ono, Tomohiro Kimura, Masayuki Yoshida ve Shigenori Kino - Kodlama aparatı ve kodlama yöntemi

JPEG spesifikasyonu ayrıca IBM'in diğer üç patentinden alıntı yapıyor. Patent sahibi olarak belirtilen diğer şirketler arasında AT&T (iki patent) ve Canon Inc. [1] bulunmaktadır. Listede yer almayan, Compression Labs'den Wen-Hsiung Chen ve Daniel J. Klenke tarafından Ekim 1986'da dosyalanan ABD Patenti 4,698,672'dir. Patent, DCT tabanlı bir görüntü sıkıştırma algoritmasını açıklar ve daha sonra 2002'de tartışmaya neden olur (bkz. Aşağıda patent tartışması). [17] Bununla birlikte, JPEG spesifikasyonu, Wen-Hsiung Chen tarafından 1977 ve 1984'te yayınlanan önceki iki araştırma makalesine atıfta bulundu. [1]

### **JPEG Standardı**

"JPEG", JPEG standardını ve diğer hareketsiz resim kodlama standartlarını oluşturan komitenin adı olan Joint Photographic Experts Group anlamına gelir. "Ortak", ISO TC97 WG8 ve CCITT SGVIII anlamına geliyordu. 1986'da kurulan grup, 1980'lerin sonunda JPEG standardını geliştirdi. İnceledikleri birkaç dönüşüm kodlama tekniği arasında, açık ara en verimli pratik sıkıştırma tekniği olduğu için ayrık kosinüs dönüşümünü (DCT) seçtiler. Grup, JPEG standardını 1992'de yayınladı. [4]

1987'de ISO TC 97, ISO / IEC JTC1 oldu ve 1992'de CCITT, ITU-T oldu. Şu anda JTC1 tarafında, JPEG, ISO / IEC Ortak Teknik Komitesi 1, Alt Komite 29, Çalışma Grubu 1'in (ISO / IEC JTC 1 / SC 29 / WG 1) iki alt grubundan biridir - sabit resimlerin kodlanması. [18] [19] [20] ITU-T tarafında, ITU-T SG16 ilgili yapıdır. Orijinal JPEG Grubu 1986 yılında düzenlendi, [21] 1992'de ilk JPEG standardını yayınladı ve Eylül 1992'de ITU-T Önerisi T.81 [22] ve 1994'te ISO / IEC 10918-1 olarak onaylandı.

JPEG standardı, bir görüntünün bir bayt akışına nasıl sıkıştırıldığını ve bir görüntüye nasıl açıldığını tanımlayan codec'i belirtir, ancak bu akışı içermek için kullanılan dosya biçimini tanımlamaz. [23] Exif ve JFIF standartları, JPEG ile sıkıştırılmış görüntülerin değişimi için yaygın olarak kullanılan dosya formatlarını tanımlar.

JPEG standartları resmi olarak Bilgi teknolojisi - Sürekli tonlu hareketsiz görüntülerin dijital sıkıştırması ve kodlanması olarak adlandırılır. ISO / IEC 10918 aşağıdaki bölümlerden oluşur:

## YABLO

### Patent Tartışması

2002'de Forgent Networks, 27 Ekim 1986'da dosyalanan ve 6 Ekim 1987'de verilen bir patentten doğan JPEG teknolojisine sahip olduğunu ve patent haklarını uygulayacağını iddia etti: Compression Labs 'Wen- tarafından 4,698,672 ABD Patenti Hsiung Chen ve Daniel J. Klenke. [17] [28] Forgent o sırada Sıkıştırma Laboratuvarlarına sahip olmasa da Chen, daha sonra, Cisco için çalışmaya başlamadan önce Sıkıştırma Laboratuvarlarını Forgent'a sattı. Bu, Forgent'ın patent üzerinde sahiplik kazanmasına yol açtı. [17] Forgent'ın 2002 duyurusu, Unisys'in GIF görüntü sıkıştırma standardı üzerindeki haklarını koruma girişimlerini anımsatan bir heyecan yarattı.

JPEG komitesi, 2002 yılında patent iddialarını araştırdı ve çeşitli uzmanlar tarafından paylaşılan bir görüş olan [29] önceki teknik tarafından geçersiz kılındığı görüşündeydi. Patent, Nasir Ahmed, T. Natarajan ve KR Rao'nun [1] [13] [14] 1974 tarihli bir makalesinden kaynaklanan kayıplı bir görüntü sıkıştırma tekniği olan ayrık kosinüs dönüşümüne (DCT) [17] dayalı bir görüntü sıkıştırma algoritmasını açıklar. ] Wen-Hsiung Chen, C.H. Smith ve S.C. Fralick. [15] [17] 1992 JPEG spesifikasyonu, DCT algoritması için hem 1974 Ahmed kağıdına hem de 1977 Chen kağıdına ve Chen ve W.K. tarafından 1984 tarihli bir makaleye atıfta bulunur. Niceleme algoritması için Pratt. [1] [16] Sıkıştırma Laboratuvarları Chen tarafından kuruldu ve DCT teknolojisini ticarileştiren ilk şirket oldu. [31] Chen 1986'da DCT tabanlı bir görüntü sıkıştırma algoritması için patentini Klenke'ye sunduğunda, daha sonra JPEG standardı haline gelecek olanların çoğu önceki literatürde zaten formüle edilmişti. [17] JPEG temsilcisi Richard Clark da Chen'in kendisinin JPEG komitelerinden birinde oturduğunu iddia etti, ancak Forgent bu iddiayı reddetti. [17]

2002 ve 2004 yılları arasında Forgent, patentlerini yaklaşık 30 şirkete lisanslayarak yaklaşık 105 milyon ABD doları elde edebildi. Nisan 2004'te Forgent, 31 başka şirkete daha fazla lisans ödemesi yapması için dava açtı. Aynı yılın Temmuz ayında, 21 büyük bilgisayar şirketinden oluşan bir konsorsiyum, patenti geçersiz kılmak amacıyla karşı dava açtı. Ayrıca Microsoft, Nisan 2005'te Forgent'a karşı ayrı bir dava açtı. [32] Şubat 2006'da, Birleşik Devletler Patent ve Ticari Marka Ofisi, Kamu Patent Vakfının talebi üzerine Forgent'ın JPEG patentini yeniden incelemeyi kabul etti. [33] 26 Mayıs 2006'da USPTO, önceki tekniğe dayanarak patenti geçersiz buldu. USPTO ayrıca Forgent'ın önceki teknikten haberdar olduğunu, ancak kasıtlı olarak Patent Ofisine söylemekten kaçındığını da buldu. Bu, patentin eski durumuna getirilmesine yönelik herhangi bir temyiz başarısızlığına ulaşma olasılığını oldukça düşük kılar. [34]

Forgent ayrıca, Avrupa Patent Ofisi tarafından 1994 yılında verilen benzer bir patente sahiptir, ancak bunun ne kadar uygulanabilir olduğu belirsizdir. [35]

27 Ekim 2006 itibarıyla, ABD patentinin 20 yıllık süresinin sona ermiş olduğu görülüyor ve Kasım 2006'da Forgent, JPEG standardının kullanımına karşı patent taleplerinin uygulanmasından vazgeçmeyi kabul etti. [36]

JPEG komitesi, standartlarının (özellikle temel yöntemlerinin) lisans ücretleri ödenmeden uygulanabilir olması ve 20'den fazla büyük kuruluştan JPEG 2000 standardı için uygun lisans haklarını güvence altına almış olması konusundaki açık hedeflerinden biri olarak var.

Ağustos 2007'den başlayarak, başka bir şirket olan Global Patent Holdings, LLC 1993 yılında yayınlanan patentinin (ABD Patenti 5,253,341) JPEG görüntülerinin bir web sitesinde veya e-posta yoluyla indirilmesiyle ihlal edildiğini iddia etti. Geçersiz kılınmıyorsa, bu patent JPEG görüntülerini gösteren herhangi bir web sitesine uygulanabilir. Patent, 2000-2007 yılları arasında ABD Patent ve Ticari Marka Ofisi tarafından yeniden inceleniyordu; Temmuz 2007'de, Patent Ofisi patentin tüm orijinal istemlerini iptal etti, ancak Global Patent Holdings tarafından önerilen ek bir istemin (istem 17) geçerli olduğunu tespit etti. [37] Global Patent Holdings daha sonra patentinin 17. istemine dayanarak bir dizi dava açtı.

Her ikisi de Chicago, Illinois'de açılan yeniden incelemeyi takiben ilk iki davasında, Global Patent Holdings, Green Bay Packers, CDW, Motorola, Apple, Orbitz, Officemax, Caterpillar, Kraft ve Peapod'a sanık olarak dava açtı. 5 Aralık 2007'de Güney Florida'da ADT Security Services, AutoNation, Florida Crystals Corp., HearUSA, MovieTickets.com, Ocwen Financial Corp. ve Tyre Kingdom aleyhine üçüncü bir dava ve 8 Ocak 2008'de Güney'de dördüncü bir dava açıldı. Florida, Boca Raton Resort & Club'a karşı. Nevada'da Global Patent Holdings'e beşinci dava açıldı. Bu dava, Global Patent Holdings tarafından tehdit edildiği iddia edilen Zappos.com, Inc. tarafından açılmış ve '341 patentinin geçersiz olduğuna ve ihlal edilmediğine dair bir adli beyanda bulunulmasını istemiştir.

Global Patent Holdings, Gregory Aharonian [38] ve "Patent Troll Tracker" [39] olarak bilinen bir web sitesi blogunun anonim operatörü dahil olmak üzere geniş yazılım patentlerine yönelik açık sözlü eleştirmenleri dava etmek veya tehdit etmek için '341 patentini de kullandı. 21 Aralık'ta, , 2007, Chicago'lu patent avukatı Vernon Francissen, ABD Patent ve Ticari Marka Ofisi'nden, '341 patentinin geri kalan tek talebini yeni önceki teknik temelinde yeniden incelemesini istedi. [40]

5 Mart 2008'de ABD Patent ve Ticari Marka Ofisi, '341 patentini yeniden incelemeyi kabul etti ve yeni önceki teknolojinin patentin geçerliliğiyle ilgili önemli yeni sorular ortaya çıkardığını tespit etti. [41] Yeniden incelemenin ışığında, hak ihlalinde bulunan sanıklar, bekleyen beş davadan dördünde, ABD Patent ve Ticari Marka Bürosu'nun '341 patenti incelemesinin tamamlanmasına kadar davalarını askıya alma (durdurma) dilekçesi verdiler. 23 Nisan 2008'de, Illinois, Chicago'daki iki davaya başkanlık eden bir yargıç bu davalarda önermeleri kabul etti. [42] 22 Temmuz 2008 tarihinde, Patent Ofisi ikinci yeniden incelemenin ilk "Ofis Davası" nı yayınladı ve on dokuz ayrı gerekçeye dayanarak iddiayı geçersiz buldu. [43] 24 Kasım 2009'da, tüm talepleri iptal eden bir Yeniden İnceleme Sertifikası verildi.

2011'den başlayarak ve 2013'ün başlarından itibaren devam eden, Doğu Teksas merkezli Princeton Digital Image Corporation [44] olarak bilinen bir kuruluş, 4,813,056 sayılı ABD Patentini ihlal ettiği iddiasıyla çok sayıda şirkete dava açmaya başladı. Princeton, JPEG görüntü sıkıştırma standardının '056 patentini ihlal ettiğini ve çok sayıda web sitesine, perakendeciye, kamera ve cihaz üreticisine ve satıcıya dava açtığını iddia ediyor. Patent ilk olarak General Electric'e aitti ve devredildi. Patent Aralık 2007'de sona erdi, ancak Princeton

çok sayıda şirkete bu patentin "geçmiş ihlali" nedeniyle dava açtı. (ABD patent yasalarına göre, bir patent sahibi, bir dava açılmadan altı yıl öncesine kadar "geçmiş ihlal" için dava açabilir, böylece Princeton teorik olarak şirketlere Aralık 2013'e kadar dava açmaya devam edebilirdi.) Mart 2013 itibarıyla, Princeton'ın bekleyen davaları vardı. 55'ten fazla şirkete karşı New York ve Delaware. General Electric'in davaya karıştığı bilinmemekle birlikte mahkeme kayıtları, patentin 2009 yılında Princeton'a devredildiğini ve patentin belirli haklarını koruduğunu gösteriyor. [45]

### **Tipik Kullanım**

JPEG sıkıştırma algoritması, en iyi şekilde, yumuşak ton ve renk varyasyonlarına sahip gerçekçi sahnelerin fotoğrafları ve resimlerinde çalışır. Bir görüntü için kullanılan veri miktarını azaltmanın duyarlı sunum için önemli olduğu web kullanımı için, JPEG'in sıkıştırma avantajları JPEG'i popüler hale getirir. JPEG / Exif ayrıca dijital kameralar tarafından kaydedilen en yaygın formattır.

Ancak JPEG, bitişik pikseller arasındaki keskin kontrastların göze çarpan yapaylıklara neden olabileceği çizgi çizimler ve diğer metinsel veya ikonik grafikler için pek uygun değildir. Bu tür görüntüler, TIFF, GIF veya PNG gibi kayıpsız bir grafik biçiminde daha iyi kaydedilir. [46] JPEG standardı kayıpsız bir kodlama modu içerir, ancak bu mod çoğu üründe desteklenmez.

JPEG'nin tipik kullanımı, görüntü doğruluğunu azaltan kayıplı bir sıkıştırma yöntemi olduğundan, görüntüleme verilerinin tam olarak çoğaltılması için uygun değildir (bazı bilimsel ve tıbbi görüntüleme uygulamaları ve belirli teknik görüntü işleme çalışmaları gibi).

JPEG, birden çok düzenlemeden geçecek dosyalar için de pek uygun değildir, çünkü görüntü her yeniden sıkıştırıldığında, özellikle görüntü kırıldığında veya kaydırıldığında veya kodlama parametreleri değiştirildiğinde bazı görüntü kalitesi kaybolur - ayrıntılar için dijital nesil kaybına bakın. Sıralı ve tekrarlayan düzenleme sırasında görüntü bilgisi kaybını önlemek için, ilk düzenleme kayıpsız bir biçimde kaydedilebilir, daha sonra bu formatta düzenlenebilir ve daha sonra dağıtım için JPEG olarak yayınlanabilir.

### **JPEG Sıkıştırma**

JPEG, ayrık kosinüs dönüşümüne (DCT) dayalı kayıplı bir sıkıştırma biçimi kullanır. Bu matematiksel işlem, video kaynağının her karesini / alanını uzamsal (2D) etki alanından frekans etki alanına (a.k.a. dönüşüm etki alanı) dönüştürür. İnsan psiko-görsel sistemine gevşek bir şekilde dayanan algısal bir model, yüksek frekanslı bilgileri, yani yoğunluktaki keskin geçişleri ve renk tonunu atar. Dönüşüm alanında, bilgiyi azaltma sürecine niceleme denir. Daha basit bir ifadeyle, niceleme, büyük bir sayı ölçeğini (her sayının farklı oluşumlarıyla) en uygun şekilde daha küçük bir ölçeğe indirgemek için bir yöntemdir ve dönüşüm alanı, görüntünün uygun bir temsilidir, çünkü daha az katkıda bulunan yüksek frekans katsayıları diğer katsayılara göre genel resme göre, yüksek sıkıştırılabilirliğe sahip

karakteristik olarak küçük deęerlerdir. Nicelenmiř katsayılar daha sonra sıralanır ve kayıpsız bir řekilde ıktı bit akıřına paketlenir. Neredeyse tüm JPEG yazılım uygulamaları, kullanıcının sıkıřtırma oranı (ve dięer isteęe baęlı parametreler) üzerinde kontrolüne izin vererek, kullanıcının daha küçük dosya boyutu iin resim kalitesinden vazgeemesine olanak tanır. Gml uygulamalarda (benzer bir DCT sıkıřtırma řeması kullanan miniDV gibi), parametreler uygulama iin nceden seilir ve sabitlenir.

Sıkıřtırma yntemi genellikle kayıplıdır, yani bazı orijinal grnt bilgileri kaybolur ve geri yklenemez, bu da muhtemelen grnt kalitesini etkiler. JPEG standardında tanımlanan isteęe baęlı kayıpsız bir mod vardır. Ancak bu mod rnlerde yaygın olarak desteklenmemektedir.

Ayrıca, verilerin giderek daha yksek ayrıntıya sahip oklu geiřlerde sıkıřtırıldığı taramalı ařamalı bir JPEG formatı da vardır. Bu, yavaş bir baęlantı zerinden indirilirken grntlenecek byk grntler iin idealdir ve verilerin yalnızca bir kısmını aldıktan sonra makul bir nizleme saęlar. Ancak, ařamalı JPEG'ler iin destek evrensel deęildir. Ařamalı JPEG'ler, onları desteklemeyen programlar tarafından alındığında (Windows 7'den nceki Internet Explorer srmleri gibi) [47], yazılım grnty ancak tamamen indirildikten sonra grntler.

## Kayıpsız Dzenleme

Grnt boyutu 1 MCU bloęunun (Minimum Kodlanmış Birim) katı olduęu srece (genellikleher iki ynde 16 piksel, 4: 2: 0 kroma alt rnekleme iin). Bunu uygulayan yardımcı programlar řunları ierir:

- jpegtran ve GUI'si, Jpegcrop.
- IrfanView, JPG\_TRANSFORM eklentisinin yklenmesini gerektiren "JPG Lossless Crop (PlugIn)" ve "JPG Lossless Rotation (PlugIn)" kullanarak.
- "Dosyaya Kayıpsız Kırpma" ve "JPEG Kayıpsız Dndrme" kullanan FastStone Image Viewer.
- "JPEG kayıpsız dnřmler" kullanan XnViewMP.
- ACDSee, "Kayıpsız JPEG iřlemlerini zorla" seeneęiyle kayıpsız dnř destekler (ancak kayıpsız kırpmayı desteklemez).

Bloklar 90 derecelik artıřlarla dndrlebilir, yatay, dikey ve apraz eksenlerde evrilebilir ve grnt iinde hareket ettirilebilir. Orijinal grntdeki tm blokların deęiřtirilmiř durumda kullanılması gerekmez.

Bir JPEG grntnn st ve sol kenarı 8 × 8 piksellik bir blok sınırında bulunmalıdır, ancak alt ve saę kenarın bunu yapması gerekmez. Bu, olası kayıpsız kırpma iřlemlerini sınırlar ve ayrıca alt veya saę kenarı tm kanallar iin bir blok sınırında bulunmayan bir grntnn dnmelerini ve dnmelerini nler (nk kenar, yukarıda veya solda - yukarıda belirtildięi gibi - a blok sınırı zorunludur).

Görüntü genişliği ve yüksekliğinin 8 veya 16'nın katı olmadığı (kroma alt örneklemesine bağlı olarak) kayıpsız değildir. Böyle bir görüntünün döndürülmesi, blokların yeniden hesaplanmasına ve bu da kalite kaybına neden olur. [48]

Kayıpsız kırpma kullanılırken, kırpma bölgesinin alt veya sağ tarafı bir blok sınırında değilse, kısmen kullanılmış bloklardan gelen verilerin geri kalanı yine kırılan dosyada mevcut olacaktır ve kurtarılabılır. Tek fark, katsayıların dosyaya yerleştirilme sırası olduğundan, temel ve aşamalı formatlar arasında kalite kaybı olmaksızın dönüştürme yapmak da mümkündür.

Ayrıca, aynı kalitede kaydedildikleri ve kenarlar blok sınırlarıyla çakıştığı sürece birkaç JPEG görüntüsü kayıpsız bir şekilde birleştirilebilir.

### **JPEG Dosyaları**

"JPEG Değişim Biçimi" (JIF) olarak bilinen dosya biçimi, standardın Ek B'sinde belirtilmiştir. Ancak, bu "saf" dosya formatı nadiren kullanılır, çünkü öncelikle standardın tüm yönlerini tam olarak uygulayan kodlayıcıların ve kod çözücülerin programlanmasının zorluğu ve standardın bazı eksiklikleri nedeniyle:

- Renk alanı tanımı
- Bileşen alt örnekleme kaydı
- Piksel en boy oranı tanımı.

Bu sorunları ele almak için birkaç ek standart geliştirilmiştir. Bunlardan ilki 1992'de piyasaya sürüldü, JPEG Dosya Değişim Biçimi (veya JFIF), ardından son yıllarda Değiştirilebilir görüntü dosyası biçimi (Exif) ve ICC renk profilleri geldi. Bu formatların her ikisi de, farklı işaretleyicilerden oluşan gerçek JIF bayt düzenini kullanır, ancak ek olarak, JIF standardının uzantı noktalarından birini, yani uygulama işaretlerini kullanır: JFIF, APP0'ı kullanırken, Exif, APP1'i kullanır. JIF standardında ileride kullanılmak üzere bırakılan ve onun tarafından okunmayan dosyanın bu bölümleri içinde, bu standartlar belirli meta verileri ekler.

Bu nedenle, bazı yönlerden JFIF, belirli kısıtlamaları (tüm farklı kodlama modlarına izin vermemek gibi) belirlemesi açısından JIF standardının kısaltılmış bir sürümüdür, diğer şekillerde ise eklenenler nedeniyle JIF'in bir uzantısıdır. meta veriler. Orijinal JFIF standardı için belgeler şu şekildedir: [49]

JPEG File Interchange Format, JPEG bit akışlarının çok çeşitli platformlar ve uygulamalar arasında değiş tokuş edilmesini sağlayan minimal bir dosya formatıdır. Bu minimum format, TIFF JPEG spesifikasyonunda bulunan gelişmiş özelliklerin hiçbirini veya uygulamaya özel herhangi bir dosya formatını içermez. Bu basitleştirilmiş formatın tek amacı JPEG sıkıştırılmış görüntülerin değiş tokuşuna izin vermek de olmamalıdır.

JPEG sıkıştırması kullanan görüntü dosyaları genellikle "JPEG dosyaları" olarak adlandırılır ve JIF görüntü biçiminin varyantlarında saklanır. JPEG çıktısı veren çoğu görüntü yakalama cihazı (dijital kameralar gibi), aslında kamera endüstrisinin meta veri değişimi için standartlaştırdığı format olan Exif formatında dosyalar oluşturuyor. Öte yandan, Exif

standardı renk profillerine izin vermediğinden, çoğu görüntü düzenleme yazılımı JPEG'i JFIF formatında depolar ve ayrıca meta verileri neredeyse uyumlu bir şekilde dahil etmek için Exif dosyasından APP1 segmentini içerir; JFIF standardı biraz esnek bir şekilde yorumlanmıştır. [50]

Kesin konuşursak, JFIF ve Exif standartları uyumsuzdur, çünkü her biri kendi işaretleyici segmentinin (sırasıyla APP0 veya APP1) önce görüldüğünü belirtir. Pratikte, çoğu JPEG dosyası, Exif başlığından önce gelen bir JFIF işaret parçası içerir. Bu, eski okuyucuların eski biçimdeki JFIF segmentini doğru bir şekilde işlemesine izin verirken, daha yeni okuyucular da aşağıdaki Exif segmentinin kodunu çözerek ilk önce görünmesini gerektirme konusunda daha az katıdır.

### **JPEG Dosya Adı Uzantıları**

JPEG sıkıştırması kullanan dosyalar için en yaygın dosya adı uzantıları .jpg ve .jpeg'dir, ancak .jpe, .jfif ve .jif de kullanılır. JPEG verilerinin diğer dosya türlerine gömülmesi de mümkündür - TIFF kodlu dosyalar genellikle bir JPEG görüntüsünü ana görüntünün küçük resmi olarak gömer; ve MP3 dosyaları, ID3v2 etiketinde bir JPEG kapak resmi içerebilir.

### **Renk profili**

Çoğu JPEG dosyası bir ICC renk profilini (renk alanı) yerleştirir. Yaygın olarak kullanılan renk profilleri arasında sRGB ve Adobe RGB bulunur. Bu renk uzayları doğrusal olmayan bir dönüşüm kullandığından, 8 bitlik bir JPEG dosyasının dinamik aralığı yaklaşık 11 duraktır; gama eğrisine bakın.

### **Sözdizimi ve Yapı**

Bir JPEG görüntüsü, her biri bir 0xFF bayt ile başlayan ve ardından ne tür bir işaretçi olduğunu belirten bir bayt ile başlayan bir işaretleyici ile başlayan bir dizi segmentten oluşur. Bazı işaretçiler yalnızca bu iki bayttan oluşur; diğerlerini takip eden marköre özgü yük verilerinin uzunluğunu gösteren iki bayt (yüksek sonra düşük) izler. (Uzunluk, uzunluk için iki bayt içerir, ancak işaretleyici için iki bayt içermez.) Bazı işaretçilerin ardından entropi kodlu veriler gelir; böyle bir markörün uzunluğu entropi kodlu verileri içermez. Ardışık 0xFF baytlarının doldurma amacıyla doldurma baytları olarak kullanıldığına dikkat edin, ancak bu doldurma bayt dolgusu yalnızca entropi kodlu tarama verilerinin hemen ardından gelen işaretçiler için yapılmalıdır (ayrıntılar için JPEG belirtim bölümü B.1.1.2 ve E.1.2'ye bakın; özellikle "Sıkıştırılmış verilerden sonra işaretçilerin eklendiği her durumda, isteğe bağlı 0xFF doldurma baytları işaretleyiciden önce gelebilir").

Entropi ile kodlanmış veriler içinde, herhangi bir 0xFF baytından sonra, kodlayıcı tarafından bir sonraki bayttan önce bir 0x00 bayt eklenir, böylece hiçbirinin amaçlanmadığı yerde bir işaretçi yok gibi görünerek çerçeveleme hatalarını önler. Kod çözücüler bu 0x00 baytı atlmalıdır. Bayt doldurma olarak adlandırılan bu teknik (bkz. JPEG belirtim bölümü F.1.2.3), işaret yükü verilerine değil, yalnızca entropi kodlu verilere uygulanır. Bununla birlikte, entropi kodlu verilerin kendine ait birkaç işareti olduğunu unutmayın; Paralel kod çözmeye izin vermek için bağımsız entropi kodlu veri yığınlarını izole etmek için kullanılan



özellikle Sıfırlama işaretçileri (0xD0 ila 0xD7) ve kodlayıcılar bu Sıfırlama işaretlerini düzenli aralıklarla eklemekte özgürdür (ancak tüm kodlayıcılar bunu yapmaz).

Başka türden JPEG kodlamalarını tanıtan başka Çerçeve Başlangıcı işaretçileri de vardır.

Birkaç satıcı aynı APPn işaret tipini kullanabileceğinden, uygulamaya özgü işaretçiler genellikle bir standart veya satıcı adıyla (örneğin, "Exif" veya "Adobe") veya başka bir tanımlayıcı dizeyle başlar.

Bir yeniden başlatma işaretleyicisinde, bloktan bloğa tahmin değişkenleri sıfırlanır ve bit akışı, bir bayt sınırına senkronize edilir. Yeniden başlatma işaretçileri, güvenilir bir ağ üzerinden aktarım veya dosya bozulması gibi bit akışı hatasından sonra kurtarma için araçlar sağlar. Yeniden başlatma işaretleyicileri arasındaki makro blok dizilerinin kodu bağımsız olarak çözülebildiğinden, bu işlemlerin kodu paralel olarak çözülebilir.

## JPEG Codec Örneği

Bir JPEG dosyası çeşitli şekillerde kodlanabilse de, çoğunlukla JFIF kodlamasıyla yapılır. Kodlama süreci birkaç adımdan oluşur:

Görüntüdeki renklerin temsili, parlaklığı temsil eden bir luma bileşeninden (Y ') ve rengi temsil eden iki kroma bileşeninden (CB ve CR) oluşan Y'CBCR'ye dönüştürülür. Bu adım bazen atlanır.

Kroma verilerinin çözünürlüğü genellikle 2 veya 3 kat azaltılır. Bu, gözün ince renk ayrıntılarına ince parlaklık ayrıntılarından daha az duyarlı olduğu gerçeğini yansıtır.

Görüntü  $8 \times 8$  piksellik bloklara bölünür ve her blok için Y, CB ve CR verilerinin her biri ayrık kosinüs dönüşümüne (DCT) tabi tutulur. DCT, bir tür uzamsal frekans spektrumu üretmesi açısından Fourier dönüşümüne benzer.

Frekans bileşenlerinin genlikleri nicelendirilir. İnsan görüşü, geniş alanlardaki küçük renk veya parlaklık değişikliklerine, yüksek frekanslı parlaklık değişikliklerinin gücünden çok daha duyarlıdır. Bu nedenle, yüksek frekanslı bileşenlerin büyüklükleri, düşük frekanslı bileşenlere göre daha düşük bir doğrulukla depolanır. Kodlayıcının kalite ayarı (örneğin Independent JPEG Group kütüphanesinde [52] 0-100 ölçeğinde 50 veya 95), her bir frekans bileşeninin çözünürlüğünün ne ölçüde azaldığını etkiler. Aşırı derecede düşük kaliteli bir ayar kullanılırsa, yüksek frekanslı bileşenler tamamen atılır.

Tüm  $8 \times 8$  bloklar için elde edilen veriler, Huffman kodlamasının bir varyantı olan kayıpsız bir algoritma ile daha da sıkıştırılır.

Kod çözme işlemi, geri döndürülemez olduğu için niceleme dışında bu adımları tersine çevirir. Bu bölümün geri kalanında, kodlama ve kod çözme işlemleri daha ayrıntılı olarak açıklanmaktadır.

## **Kodlama**

JPEG standardındaki seçeneklerin çoğu yaygın olarak kullanılmamaktadır ve yukarıda belirtildiği gibi çoğu görüntü yazılımı, bir JPEG dosyası oluştururken daha basit JFIF formatını kullanır; bu, diğer şeylerin yanı sıra kodlama yöntemini belirler. Piksel başına 24 bit (her biri kırmızı, yeşil ve mavi olmak üzere sekiz) olan bir girdiye uygulandığında en yaygın kodlama yöntemlerinden birinin kısa bir açıklaması. Bu özel seçenek, kayıplı bir veri sıkıştırma yöntemidir.

## **Renk Alanı Dönüşümü**

İlk olarak, görüntü RGB'den Y'CBCR (veya gayri resmi olarak YCbCr) adı verilen farklı bir renk uzayına dönüştürülmelidir. Üç bileşen Y', CB ve CR'ye sahiptir: Y' bileşeni bir pikselin parlaklığını temsil eder ve CB ve CR bileşenleri krominansı temsil eder (mavi ve kırmızı bileşenlere bölünmüştür). Bu, temelde dijital renkli televizyon ve video DVD'leri içeren dijital video tarafından kullanılan renk alanıyla aynıdır ve rengin analog PAL video ve MAC'de temsil edilme şekline benzer (ancak YIQ renk alanını kullanan analog NTSC ile değil). Y'CBCR renk alanı dönüşümü, algısal görüntü kalitesi (veya aynı sıkıştırma için daha yüksek algısal görüntü kalitesi) üzerinde önemli bir etki olmaksızın daha fazla sıkıştırmaya izin verir. Görüntünün nihai algısal kalitesi için daha önemli olan parlaklık bilgisi tek bir kanalla sınırlı olduğu için sıkıştırma daha verimlidir. Bu, insan görsel sistemindeki renk algısına daha yakından karşılık gelir. Renk dönüşümü ayrıca istatistiksel ilintisizleştirme yoluyla sıkıştırmayı da geliştirir.

JFIF standardında Y'CBCR'ye özel bir dönüştürme belirtilmiştir ve elde edilen JPEG dosyasının maksimum uyumluluğa sahip olması için gerçekleştirilmelidir. Ancak, "en yüksek kalite" modundaki bazı JPEG uygulamaları bu adımı uygulamaz ve bunun yerine renk bilgilerini, görüntünün kırmızı, yeşil ve mavi parlaklık bileşenleri için ayrı kanallarda depolandığı RGB renk modelinde [53] tutar. Bu, daha az verimli bir sıkıştırma ile sonuçlanır ve dosya boyutu özellikle önemli olduğunda büyük olasılıkla kullanılmaz.

## **Altörnekleme**

İnsan gözündeki renk ve parlaklığa duyarlı alıcıların yoğunlukları nedeniyle, insanlar bir görüntünün parlaklığında (Y' bileşeni), bir görüntünün ton ve renk doygunluğundan (Cb ve Cr bileşenleri). Bu bilgiyi kullanarak, kodlayıcılar görüntüleri daha verimli bir şekilde sıkıştırmak için tasarlanabilir.

Y'CBCR renk modeline dönüşüm, Cb ve Cr bileşenlerinin ("altörnekleme" veya "kroma alt örnekleme" olarak adlandırılır) uzamsal çözünürlüğünü azaltmak olan bir sonraki olağan adımı etkinleştirir. Normalde JPEG görüntüler için altörneklemenin yapıldığı oranlar 4: 4: 4 (altörnekleme yok), 4: 2: 2 (yatay yönde 2 kat küçültme) veya (en yaygın olarak) 4: 2: 0 (hem yatay hem de dikey yönde 2 faktör azaltma). Sıkıştırma işleminin geri kalanı için Y', Cb ve Cr ayrı ayrı ve çok benzer bir şekilde işlenir.

### Bölmeyi Engelle (Block Splitting)

Alt örneklemeden sonra, her kanal  $8 \times 8$  bloğa bölünmelidir. Renk alt örneklemesine bağlı olarak bu,  $8 \times 8$  (4: 4: 4 - alt örnekleme yok),  $16 \times 8$  (4: 2: 2) veya en yaygın olarak  $16 \times 16$  (4: 2: 0). Video sıkıştırmada MCU'lara makro bloklar denir.

Bir kanala yönelik veriler tam sayı blokları temsil etmiyorsa, o zaman kodlayıcının tamamlanmamış blokların kalan alanını bir tür kukla verilerle doldurması gerekir. Kenarların sabit bir renkle (örneğin, siyah) doldurulması, kenarlığın görünür kısmı boyunca zil sesi yapaylıkları oluşturabilir; kenar piksellerinin tekrarlanması, bu tür yapay olayları azaltan (ancak zorunlu olarak tamamen ortadan kaldırmayan) yaygın bir tekniktir ve daha karmaşık kenar doldurma teknikleri de uygulanabilir.

### Ayrık Kosinüs Dönüşümü

8 bit gri tonlamalı olarak gösterilen  $8 \times 8$  alt görüntü

Daha sonra, her bileşenin (Y, Cb, Cr) her  $8 \times 8$  bloğu, normalleştirilmiş, iki boyutlu bir tip-II ayrık kosinüs dönüşümü (DCT) kullanılarak bir frekans alanı gösterimine dönüştürülür, bkz. Ayrık kosinüs dönüşümünde Alıntı 1 . DCT, ayrık kosinüs dönüşümünde olduğu gibi bir dönüşüm ailesi bağlamında bazen "tip-II DCT" olarak anılır ve karşılık gelen ters (IDCT), "tip-III DCT" olarak gösterilir.

Örnek olarak, böyle bir  $8 \times 8$  8 bitlik alt görüntü şöyle olabilir:

52	55	61	66	70	61	64	73
63	59	55	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	68	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

$8 \times 8$  bloğunun DCT'sini hesaplamadan önce, değerleri pozitif bir aralıktan sıfır merkezli bir aralığa kaydırılır. 8 bitlik bir görüntü için, orijinal bloktaki her giriş aralığı girer  $[0, 255]$  Aralığın orta noktası (bu durumda, 128 değeri) sıfıra ortalananmış bir veri aralığı oluşturmak için her girişten çıkarılır, böylece değiştirilen aralık  $[-128, 127]$ . Bu adım, takip eden DCT işleme aşamasında dinamik aralık gereksinimlerini azaltır.

Bu adım aşağıdaki değerlerle sonuçlanır

$$g = \begin{matrix} & \begin{matrix} x \\ \longrightarrow \end{matrix} \\ \begin{matrix} \downarrow y. \\ \end{matrix} & \begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix} \end{matrix}$$

Bir sonraki adım, aşağıdaki şekilde verilen iki boyutlu DCT'yi almaktır:

$$G_{u,v} = \frac{1}{4} \alpha(u) \alpha(v) \sum_{x=0}^7 \sum_{y=0}^7 g_{x,y} \cos \left[ \frac{(2x+1)u\pi}{16} \right] \cos \left[ \frac{(2y+1)v\pi}{16} \right]$$

where

- $u$  is the horizontal **spatial frequency**, for the integers  $0 \leq u < 8$ .
- $v$  is the vertical spatial frequency, for the integers  $0 \leq v < 8$ .
- $\alpha(u) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if } u = 0 \\ 1, & \text{otherwise} \end{cases}$  is a normalizing scale factor to make the transformation **orthonormal**
- $g_{x,y}$  is the pixel value at coordinates  $(x, y)$
- $G_{u,v}$  is the DCT coefficient at coordinates  $(u, v)$ .

Bu dönüşümü yukarıdaki matrisimizde gerçekleştirirsek, aşağıdakini elde ederiz (ondalık ayırıcının ötesinde en yakın iki basamağa yuvarlanır):

$$G = \begin{matrix} & \begin{matrix} u \\ \longrightarrow \end{matrix} \\ \begin{matrix} \downarrow v. \\ \end{matrix} & \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} \end{matrix}$$

Oldukça büyük büyüklükteki sol üst köşe girişine dikkat edin. Bu, tüm blok için temel tonu tanımlayan DC katsayısıdır (sabit bileşen olarak da adlandırılır). Kalan 63 katsayı, AC katsayılarıdır (alternatif bileşenler olarak da adlandırılır). [54] DCT'nin avantajı, yukarıda görülebileceği gibi, sinyalin çoğunu sonucun bir köşesinde toplama eğilimidir. Takip edilecek niceleme adımı, bu etkiyi vurgularken aynı zamanda DCT katsayılarının genel boyutunu azaltır ve entropi aşamasında verimli bir şekilde sıkıştırılması kolay bir sinyalle sonuçlanır.

DCT, 8 bitlik / bileşen görüntünün DCT katsayılarının depolanması için 11 veya daha fazla bit (DCT hesaplamasının aslına bağlı olarak) alması nedeniyle, verilerin bit derinliğini

geçici olarak artırır. Bu, codec'i bu katsayıları tutmak için geçici olarak 16 bitlik sayılar kullanmaya zorlayabilir ve bu noktada görüntü temsilinin boyutunu ikiye katlayabilir; bu değerler tipik olarak niceleme adımı ile 8 bitlik değerlere düşürülür. Bu aşamada boyuttaki geçici artış, çoğu JPEG uygulaması için bir performans sorunu değildir, çünkü tipik olarak görüntünün yalnızca çok küçük bir kısmı, görüntü kodlama veya kod çözme işlemi sırasında herhangi bir belirli zamanda tam DCT formunda depolanır.

## Niceleme

İnsan gözü, nispeten geniş bir alandaki parlaklıktaki küçük farklılıkları görme konusunda iyidir, ancak yüksek frekanslı bir parlaklık değişiminin tam gücünü ayırt etmede o kadar iyi değildir. Bu, yüksek frekans bileşenlerinde bilgi miktarını büyük ölçüde azaltmaya izin verir. Bu, basitçe frekans alanındaki her bir bileşeni o bileşen için bir sabite bölerek ve ardından en yakın tam sayıya yuvarlayarak yapılır. Bu yuvarlama işlemi, DCT hesaplaması yeterince yüksek hassasiyetle gerçekleştirilirse, tüm süreçteki tek kayıplı işlemidir (kroma alt örneklemesi dışında). Bunun bir sonucu olarak, tipik olarak, daha yüksek frekanslı bileşenlerin birçoğunun sıfıra yuvarlanması ve geri kalanların çoğunun, temsil edilmesi çok daha az bit alan küçük pozitif veya negatif sayılar haline gelmesi durumudur.

Niceleme matrisindeki öğeler, daha büyük değerler daha fazla sıkıştırma üreterek sıkıştırma oranını kontrol eder. Tipik bir niceleme matrisi (orijinal JPEG Standardında belirtildiği gibi %50 kalite için) aşağıdaki gibidir:

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}.$$

Nicelleştirilmiş DCT katsayıları aşağıdaki formül ile hesaplanır.

$$B_{j,k} = \text{round} \left( \frac{G_{j,k}}{Q_{j,k}} \right) \text{ for } j = 0, 1, 2, \dots, 7; k = 0, 1, 2, \dots, 7$$

Burada; G, nicelleştirilmemiş DCT katsayılarıdır; Q, yukarıdaki niceleme matrisidir; ve B nicelleştirilmiş DCT katsayılarıdır. Bu niceleme matrisini yukarıdaki DCT katsayı matrisi ile kullanmak şu sonuçları verir:

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

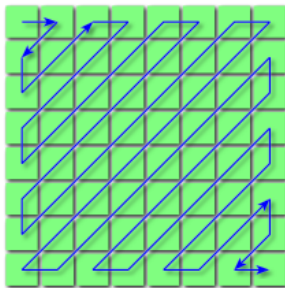
Alt bloğun yüksek frekanslı elemanlarının çoğunun (yani, x veya y uzamsal frekansı 4'ten büyük olanlar) sıfır değerlerine nicelleştirildiğine dikkat edin.

## Entropi Kodlaması

Entropi kodlaması, kayıpsız veri sıkıştırmanın özel bir biçimidir. Görüntü bileşenlerini, benzer frekansları bir arada gruplandıran uzunluk kodlama (RLE) algoritması kullanarak, uzunluk kodlama sıfırlarını ekleyerek ve sonra geriye kalan üzerinde Huffman kodlamasını kullanarak bir "zikzak" düzeninde düzenlemeyi içerir.

JPEG standardı aynı zamanda kod çözücülerin Huffman kodlamasından matematiksel olarak üstün olan aritmetik kodlamanın kullanımını desteklemesine izin verir, ancak gerektirmez. Bununla birlikte, bu özellik, tarihsel olarak telif hakkı içeren lisanslar gerektiren patentler tarafından kapsanması ve Huffman kodlamasına kıyasla kodlama ve kod çözme işleminin daha yavaş olması nedeniyle nadiren kullanılmıştır. Aritmetik kodlama, dosyaları tipik olarak yaklaşık% 5-7 küçültür.

Önceki nicelleştirilmiş DC katsayısı, mevcut nicelleştirilmiş DC katsayısını tahmin etmek için kullanılır. İkisi arasındaki fark, gerçek değer yerine kodlanmıştır. 63 nicelleştirilmiş AC katsayılarının kodlanması, bu tür tahmin farklılaşmasını kullanmaz.



Yukarıdaki nicelleştirilmiş katsayılar için zikzak dizisi aşağıda gösterilmiştir. (Gösterilen format sadece anlama / görüntüleme kolaylığı içindir.)

-26

-3 0

-3 -2 -6

2 -4 1 -3

1 1 5 1 2

-1 1 -1 2 0 0

0 0 0 -1 -1 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0

0 0 0 0 0 0

0 0 0 0 0

0 0 0 0

0 0 0

0 0

0

If the  $i$ -th block is represented by  $B_i$  and positions within each block are represented by  $(p, q)$  where  $p = 0, 1, \dots, 7$  and  $q = 0, 1, \dots, 7$ , then any coefficient in the DCT image can be represented as  $B_i(p, q)$ . Thus, in the above scheme, the order of encoding pixels (for the  $i$ -th block) is  $B_i(0, 0), B_i(0, 1), B_i(1, 0), B_i(2, 0), B_i(1, 1), B_i(0, 2), B_i(0, 3), B_i(1, 2)$  and so on.

This encoding mode is called baseline *sequential* encoding. Baseline JPEG also supports *progressive* encoding. While sequential encoding encodes coefficients of a single block at a time (in a zigzag manner), progressive encoding encodes similar-positioned batch of coefficients of all blocks in one go (called a *scan*), followed by the next batch of coefficients of all blocks, and so on. For example, if the image is divided into  $N$   $8 \times 8$  blocks  $B_0, B_1, B_2, \dots, B_{N-1}$ , then a 3-scan progressive encoding encodes DC component,  $B_i(0, 0)$  for all blocks, i.e., for all  $i = 0, 1, 2, \dots, N-1$ , in first scan. This is followed by the second scan which encoding a few more components (assuming four more components, they are  $B_i(0, 1)$  to  $B_i(1, 1)$ , still in a zigzag manner) coefficients of all blocks (so the sequence is:  $B_0(0, 1), B_0(1, 0), B_0(2, 0), B_0(1, 1), B_1(0, 1), B_1(1, 0), \dots, B_N(2, 0), B_N(1, 1)$ ), followed by all the remained coefficients of all blocks in the last scan.

Once all similar-positioned coefficients have been encoded, the next position to be encoded is the one occurring next in the zigzag traversal as indicated in the figure above. It has been found that *baseline progressive* JPEG encoding usually gives better compression as compared to *baseline sequential* JPEG due to the ability to use different Huffman tables (see below) tailored for different frequencies on each "scan" or "pass" (which includes similar-positioned coefficients), though the difference is not too large.

In the rest of the article, it is assumed that the coefficient pattern generated is due to sequential mode.

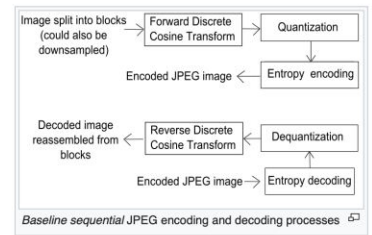
In order to encode the above generated coefficient pattern, JPEG uses Huffman encoding. The JPEG standard provides general-purpose Huffman tables; encoders may also choose to generate Huffman tables optimized for the actual frequency distributions in images being encoded.

The process of encoding the zig-zag quantized data begins with a run-length encoding explained below, where:

- $x$  is the non-zero, quantized AC coefficient.
- $RUNLENGTH$  is the number of zeroes that came before this non-zero AC coefficient.
- $SIZE$  is the number of bits required to represent  $x$ .
- $AMPLITUDE$  is the bit-representation of  $x$ .

The run-length encoding works by examining each non-zero AC coefficient  $x$  and determining how many zeroes came before the previous AC coefficient. With this information, two symbols are created:

Symbol 1	Symbol 2
( $RUNLENGTH, SIZE$ )	( $AMPLITUDE$ )




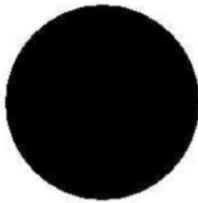
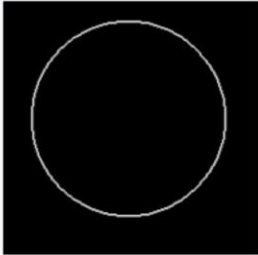
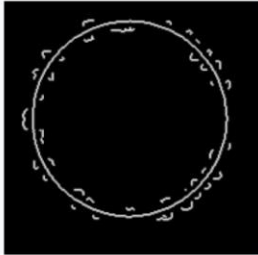
## Sıkıştırma Oranı ve Artifacts

Sıkıştırılmış  $8 \times 8$  kareler, kayıplı sıkıştırmanın diğer görsel eserleriyle birlikte büyütülmüş resimde görülebilir.

Ortaya çıkan sıkıştırma oranı, niceleme aşamasında kullanılan bölücülerde az ya da çok agresif olmakla ihtiyaca göre değiştirilebilir. Onda bir sıkıştırma, genellikle orijinalden gözle ayırt edilemeyen bir görüntüyle sonuçlanır. 100: 1'lik bir sıkıştırma oranı genellikle mümkündür, ancak orijinaline kıyasla belirgin bir şekilde yapay görünecektir. Uygun sıkıştırma seviyesi, görüntünün kullanılacağı kullanıma bağlıdır.

World Wide Web'i kullananlar, JPEG görüntülerde görünen, kontrastlı kenarlar (özellikle eğriler ve köşeler) veya "bloklı" görüntüler etrafında parazit şeklinde görünen, sıkıştırma artefaktları olarak bilinen düzensizliklere aşına olabilir. Bunlar, JPEG algoritmasının niceleme adımından kaynaklanmaktadır. Özellikle zıt renkler arasındaki keskin köşelerde fark edilirler (metin, bu tür birçok köşeyi içerdiğinden iyi bir örnektir). MPEG videodaki benzer eserler sivrisinek gürültüsü olarak adlandırılır, çünkü ortaya çıkan "kenar meşgullüğü" ve zamanla değişen sahte noktalar, nesnenin etrafında dolaşan sivrisinekleri andırır. [55] [56]

Bu eserler, daha düşük bir sıkıştırma düzeyi seçilerek azaltılabilir; Bir görüntüyü kayıpsız bir dosya biçimi kullanarak kaydederek bunlar tamamen önlenabilir, ancak bu daha büyük bir dosya boyutu ile sonuçlanacaktır. Işın izleme programları ile oluşturulan görüntüler, arazide dikkat çekici bloklı şekillere sahiptir. Bazı düşük yoğunluklu sıkıştırma kusurları, görüntüleri yalnızca görüntülerken kabul edilebilir, ancak görüntü daha sonra işlenirse vurgulanabilir, bu da genellikle kabul edilemez kaliteyle sonuçlanır. Kayıplı sıkıştırmanın kenar algılama işleme adımı üzerindeki etkisini gösteren aşağıdaki örneği düşünün.

Image	Lossless compression	Lossy compression
Original		
Processed by Canny edge detector		

Bazı programlar, kullanıcının tek tek blokların sıkıştırılma miktarını değiştirmesine izin verir. Görüntünün daha az yapaylık gösteren alanlarına daha güçlü sıkıştırma uygulanır. Bu şekilde JPEG dosya boyutunu daha az kalite kaybıyla manuel olarak azaltmak mümkündür.

Niceleme aşaması her zaman bilgi kaybına yol açtığından, JPEG standardı her zaman kayıplı bir sıkıştırma codec'idir. (Kayan nokta sayılarının hem nicelemesinde hem de yuvarlanmasında bilgi kaybolur.) Niceleme matrisi birler matrisi olsa bile, bilgi yine de yuvarlama adımında kaybolacaktır.

## Kod Çözme



Görüntüyü göstermek için kod çözme, yukarıdakilerin tümünü tersten yapmaktan ibarettir.

DCT katsayı matrisinin alınması (DC katsayısının farkını tekrar ekledikten sonra)

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

ve giriş-için-giriş ürününün yukarıdaki nicelme matrisiyle alınmasıyla sonuçlanır

$$\begin{bmatrix} -416 & -33 & -60 & 32 & 48 & -40 & 0 & 0 \\ 0 & -24 & -56 & 19 & 26 & 0 & 0 & 0 \\ -42 & 13 & 80 & -24 & -40 & 0 & 0 & 0 \\ -42 & 17 & 44 & -29 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Sol üst kısım için orijinal DCT katsayı matrisine çok benzer.

Bir sonraki adım, iki boyutlu ters DCT'yi (bir 2D tip-III DCT) almaktır;

$$f_{x,y} = \frac{1}{4} \sum_{u=0}^7 \sum_{v=0}^7 \alpha(u) \alpha(v) F_{u,v} \cos \left[ \frac{(2x+1)u\pi}{16} \right] \cos \left[ \frac{(2y+1)v\pi}{16} \right]$$

where

- $x$  is the pixel row, for the integers  $0 \leq x < 8$ .
- $y$  is the pixel column, for the integers  $0 \leq y < 8$ .
- $\alpha(u)$  is defined as above, for the integers  $0 \leq u < 8$ .
- $F_{u,v}$  is the reconstructed approximate coefficient at coordinates  $(u, v)$ .
- $f_{x,y}$  is the reconstructed pixel value at coordinates  $(x, y)$

Çıktının tam sayı değerlerine yuvarlanması (orijinalin tam sayı değerlerine sahip olması nedeniyle), değerlere sahip bir görüntüyle sonuçlanır (yine de 128'e kadar kaydırılmıştır)

$$\begin{bmatrix} -66 & -63 & -71 & -68 & -56 & -65 & -68 & -46 \\ -71 & -73 & -72 & -46 & -20 & -41 & -66 & -57 \\ -70 & -78 & -68 & -17 & 20 & -14 & -61 & -63 \\ -63 & -73 & -62 & -8 & 27 & -14 & -60 & -58 \\ -58 & -65 & -61 & -27 & -6 & -40 & -68 & -50 \\ -57 & -57 & -64 & -58 & -48 & -66 & -72 & -47 \\ -53 & -46 & -61 & -74 & -65 & -63 & -62 & -45 \\ -47 & -34 & -53 & -74 & -60 & -47 & -47 & -41 \end{bmatrix}$$

128 eklenir

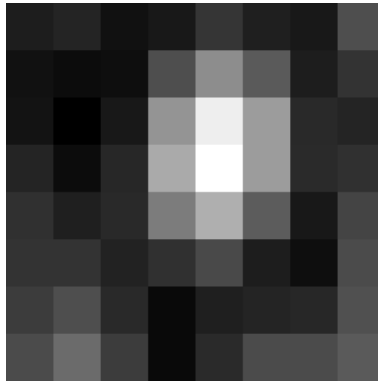
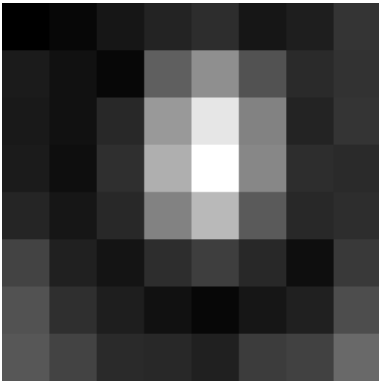
62	65	57	60	72	63	60	82
57	55	56	82	108	87	62	71
58	50	60	111	148	114	67	65
65	55	66	120	155	114	68	70
70	63	67	101	122	88	60	78
71	71	64	70	80	62	56	81
75	82	67	54	63	65	66	83
81	94	75	54	68	81	81	87

Bu, sıkıştırılmış alt görüntüdür. Genel olarak, dekompresyon işlemi, orijinal giriş aralığının dışında değerler üretebilir. [0,255]. Böyle bir durumda, kod çözücünün, orijinal bit derinliği ile sıkıştırılmış görüntüyü depolarken taşmayı önlemek için çıktı değerlerini bu aralık içinde tutacak şekilde kırpması gerekir.

Açılmış alt görüntü, aşağıdaki hata değerlerine neden olan fark (orijinal - sıkıştırılmamış) sonuçları alınarak orijinal alt görüntü ile karşılaştırılabilir (sağdaki görüntülere de bakın):

-10	-10	4	6	-2	-2	4	-9
6	4	-1	8	1	-2	7	1
4	9	8	2	-4	-10	-1	8
-2	3	5	2	-1	-8	2	-1
-3	-2	1	3	4	0	8	-8
8	-6	-4	-0	-3	6	2	-6
10	-11	-3	5	-8	-4	-1	-0
6	-15	-6	14	-3	-5	-3	7

$$\frac{1}{64} \sum_{x=0}^7 \sum_{y=0}^7 |e(x, y)| = 4.8750).$$



Hata, sol alt pikselin hemen sağındaki pikselden daha koyu hale geldiği sol alt köşede en belirgindir.

## **Gerekli Hassasiyet**

Kodlama ve kod çözme uygunluğu ve dolayısıyla kesinlik gereksinimleri ISO / IEC 10918-2'de, yani JPEG spesifikasyonunun 2. bölümünde belirtilmiştir. Bu belirtim, örneğin, test edilen bir JPEG uygulamasının bir görüntüsünden oluşturulan (ileriye doğru dönüştürülmüş) DCT katsayılarının, referans katsayıları ile karşılaştırıldığında bir nicelendirme keşçe hassasiyeti içinde bir hataya sahip olmasını gerektirir. Bu amaçla, ISO / IEC 10918-2, kod akışının kodunu çözeceği DCT katsayılarının yanı sıra test akışları da sağlar.

Benzer şekilde, ISO / IEC 10918-2, kodlayıcı hassasiyetlerini DCT alanında izin verilen maksimum hata açısından tanımlar. Bu, şu ana kadar alışılmadık bir durumdur, çünkü diğer birçok standart yalnızca kod çözücü uyumluluğunu tanımlar ve yalnızca kodlayıcıdan sözdizimsel olarak doğru bir kod akışı üretmesini gerektirir.

ISO / IEC 10918-2'de bulunan test görüntüleri, en kötü durumları kontrol etmek için (sözde) rastgele modellerdir. ISO / IEC 10918-1, renk uzaylarını tanımlamadığından ve JFIF'in YCbCr'den RGB'ye dönüşümünü de içermediğinden (şimdi ISO / IEC 10918-5), ikinci dönüşümün hassasiyeti ISO / IEC 10918-2 tarafından test edilemez.

Piksel bileşeni çıktısı başına 8 bitlik hassasiyeti desteklemek için, kod çözme ve ters DCT dönüşümleri tipik olarak optimize edilmiş kod çözücülerde en az 14 bit hassasiyetle uygulanır.

## **JPEG Sıkıştırmanın Etkileri**

JPEG sıkıştırma kusurları, ayrıntılı tek tip olmayan dokulara sahip fotoğraflarla iyi uyum sağlayarak daha yüksek sıkıştırma oranlarına olanak tanır. Daha yüksek bir sıkıştırma oranının ilk olarak görüntünün sol üst köşesindeki yüksek frekanslı dokuları nasıl etkilediğine ve karşıt çizgilerin nasıl daha bulanık hale geldiğine dikkat edin. Çok yüksek sıkıştırma oranı görüntünün kalitesini ciddi şekilde etkiler, ancak genel renkler ve görüntü biçimi hala tanınabilir. Bununla birlikte, renklerin kesinliği (insan gözü için) konturların kesinliğinden (parlaklığa bağlı olarak) daha az zarar görür. Bu, parlaklık düzleminin hassasiyetini daha fazla bilgi bitiyle korumak için kromatik düzlemleri alt örneklemeden önce (daha düşük kalitede nicelendirme de kullanılabilir) önce görüntülerin parlaklığı kromatik bilgiden ayıran bir renk modeline dönüştürülmesi gerektiği gerçeğini haklı çıkarır. .

## **Örnek Fotoğraflar**

Bilgi için, aşağıdaki sıkıştırılmamış 24 bit RGB bitmap görüntüsü (73,242 piksel) 219,726 bayt gerektirecektir (diğer tüm bilgi başlıkları hariç). Aşağıda belirtilen dosya boyutları, dahili JPEG bilgi başlıklarını ve bazı meta verileri içerir. En yüksek kaliteli görüntüler için (Q = 100), renkli piksel başına yaklaşık 8,25 bit gereklidir. [Kaynak belirtilmeli] Gri tonlamalı görüntülerde piksel başına en az 6,5 bit yeterlidir (karşılaştırılabilir bir Q = 100 kaliteli renk bilgisi yaklaşık% 25 daha fazlasını gerektirir kodlanmış bitler). Aşağıdaki en yüksek kaliteli görüntü (Q = 100) renkli piksel başına dokuz bit olarak kodlanmıştır, orta

kaliteli görüntü ( $Q = 25$ ) renk pikseli başına bir bit kullanır. Çoğu uygulama için, düşük kaliteli görüntünün gösterdiği gibi kalite faktörü piksel başına 0,75 bitin ( $Q = 12,5$ ) altına düşmemelidir. En düşük kalitedeki görüntü piksel başına yalnızca 0,13 bit kullanır ve çok zayıf renk görüntüler. Bu, görüntü önemli ölçüde küçültülmüş bir boyutta görüntüleneceği zaman kullanışlıdır.  $Q$  faktörü yerine PSNR kullanarak belirli bir görüntü kalitesi için daha iyi nicelendirme matrisleri oluşturmak için bir yöntem Minguillón & Pujol (2001) 'de açıklanmıştır. [57]

Sss

KAYNAK METİN

<https://en.wikipedia.org/wiki/JPEG>

