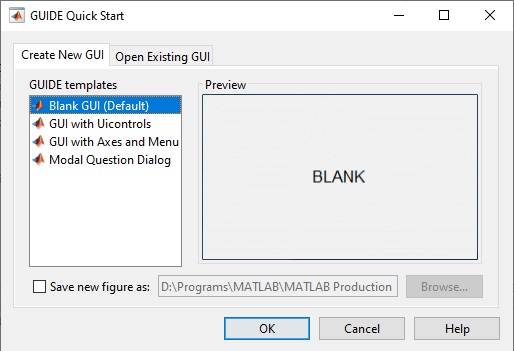
**GÖRÜNTÜ İŞLEME PROJE RAPORU**

**1-)** Bu proje kapsamında yapılan ödev MATLAB üzerinde tasarlanan bir GUI (Graphical User Interface) sistemidir. GUI sistemi yapılması iki parçadan oluşmaktadır : kullanıcı için bir arayüz tasarımı yapılması, arayüzün çalışması için kodlama yapılması.

**GUI Tasarımı**

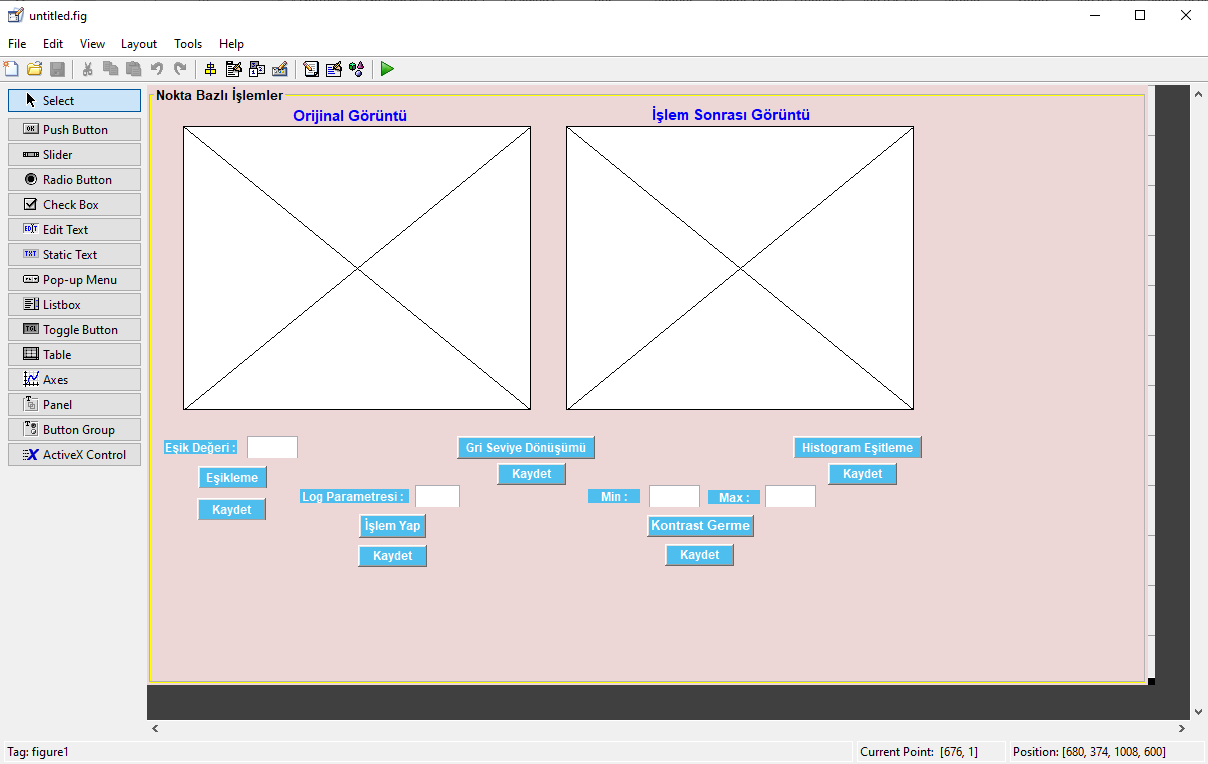
MATLAB üzerinde GUI tasarımına başlamak için öncelikle yeni bir grafiksel kullanıcı arayüzü şablonu oluşturmak gerekmektedir. MATLAB programının üzerinden menü çubuğundan yeni bir GUI oluşturmaya basılır ve aşağıdaki gibi bir şekil ortaya çıkar.



**Şekil 1.** GUI oluşturma ekranı

Yukarıdaki verilen şekil üzerinden, istenirse yeni bir GUI oluşturulabilir istenirse önceden tasarlanmış bir GUI açılabilir. “Blank GUI” olarak, boş bir GUI tasarımı oluşturuldu.

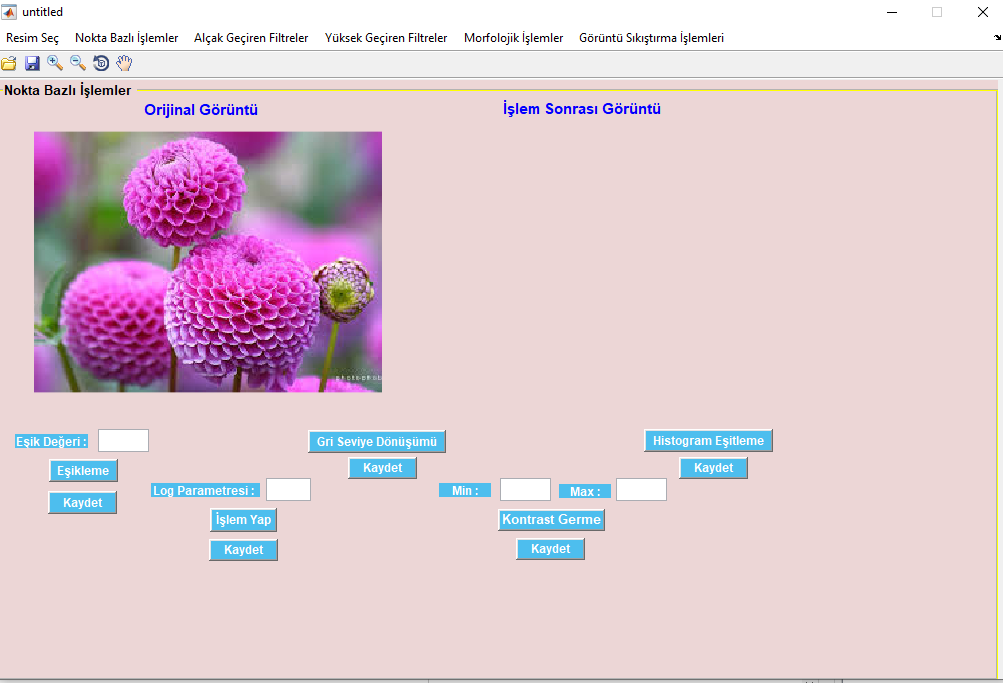
Açılan pencere üzerinden tasarıma başlanmaktadır. Boş bir GUI oluşturulduğu için gerekli olan tüm parçalar eklenmelidir. Görüntü işleme için kullanılanacak olan bu GUI üzerine ilk öncelikle panellerin eklenmesi gerekmektedir. 4 ana başlık olarak işlem yapılması gerektiği için 5 adet panel tasarımı yapılmıştır. ‘Alçak Geçiren Filtreler’ ve ‘Yüksek Geçiren Filtreler’ iki farklı panelde, diğerleri için tek bir panel tasarlanmıştır.



**Şekil 2.** GUI tasarım ekranı

Yukarıda verilen Şekil 2. üzerinde ‘Nokta Bazlı İşlemler’ başlığındaki panelin tasarımı gösterilmiştir. Panel pembe tonlarında arkaplan rengine sahip olduğu için göz yormamaktadır. Sarı highlight eklenerek de panelin görselliği ön plana alınmıştır. Panel üzerinde iki tane ‘Axes’ adı verilen nesne vardır. Axes üzerinde görüntüler gözükmekte olup kullanıcıya, seçtiği resmin orijinal halini ve daha sonra hangi işlemi yapmasına göre de işlem sonrası görüntünün durumu paylaşılmaktadır. Böylelikle kullanıcı arayüz ekranında görüntü üzerinde yapılan işlemi görebilmektedir. Alt tarafta bulunan butonlar, yapılacak olan işlemleri tetiklemektedir. Kategoriye ait olan gerekli işlemler butonlar ile sağlanmaktadır. Butonların dışında bulunan ‘Static Text’ text olarak yazılmış kısımlar için kullanılmıştır. Kullanıcıdan giriş parametreleri alınması gerektirdiği yerlerde ‘Edit Text’ nesnesi sayesinde kullanıcı değer girerek ona göre bir işlem yapılabilmektedir.

Temel olarak aynı tasarıma sahip olan bu 5 panelde sisteminin üst bölgesinde yer alan araç çubuğu sayesinde görüntü üzerinde yakınlaştırma uzaklaştırma işlemleri, görüntüyü döndürme veya sürükleyerek taşıma gibi işlemler yapılabilmektedir. Bunun yanı sıra menü çubuğu koyulmuştur. Menü çubuğu üzerinde resim seçme, resmi kaydetme ve paneller üzerinde gezinme yani görünen paneli değiştirme yapılmaktadır. Kullanıcı için kolaylık sağlayarak kullanımı kolaylaştırmaktadır.



**Şekil 3.** GUI Tam görünümü

Yukarıda verilen Şekil 1.3’te hazırlanan GUI’nin çalışması gözükmektedir. Kullanıcının eklemiş olduğu görüntü sol tarafta gösterilmekte, işlem sonrası görüntüsü ise sağ tarafta gösterilmektedir. En yukarıda menü çubuğu bulunmakta, onun altında ise araç çubuğu bulunmaktadır.

GUI tasarımının çalışması için GUI üzerindeki nesneleri tetikleyecek fonksiyonlar yazılmalıdır. Nesnelerin “Callback” fonksiyonlarına yazılacak olan kodlar bu işlemi gerçekleştirecektir.

**Kodlama ve Filtreleme İşlemleri**

Uygulama kısmı 4 ana parçadan oluşmaktadır ve her parçanın kendi altında bir çok işlem bulunmaktadır. Parçalar, farklı paneller üzerine dağılarak, sadece o panel içerisinde çalışması sağlanmıştır.

**1-) DOSYA İŞLEMLERİ**

Kullanıcının GUI sisteminde yapacağı ilk şey resim seçmek olacaktır. Seçeceği görüntüye göre işlem yapılacağı için kullanıcının öncelikle yapması gereken menü çubuğu üzerinden ‘Resim Seç’ kısmına basması gerekmektedir.

|  |
| --- |
| [filename, pathname]=uigetfile({'\*.jpg';'\*.gif';'\*.png';'\*.ppm';'\*.pgm'}, 'Görüntü Dosyasını Seç:','Multiselect','On'); |

Yukarıda verilen kod parçasında kullanıcının verilen türlerin birinde görüntü eklemesi istenilmektedir. ‘Resim Seç’ butonuna bastığı zaman dosya gezgini açılarak kullanıcı görüntü seçebilmektedir. Kullanıcının yukarıdaki kod parçasında belirtildiği gibi “.jpg, .gif, .png, .ppm, .pgm” türlerinde resim seçebilmektedir. Panel üzerindeki sol tarafta olan ‘Axes’ nesnesi üzerinde görünmekte olan resim artık işleme hazır hale gelmiştir.

Her bir işlem sonrası görüntüler kaydedilebilmektedir. Kullanıcı yaptığı işlem sonrasında oluşan görüntü için, yaptığı işlemin altında bulunan “Kaydet” butonu ile oluşmuş olan görüntüyü kaydedebilirler. Görüntü değişkeni global olarak tanımlanıp, daha sonra kaydet butonu üzerinden çağırılır ve imwrite fonksiyonu ile ‘\*’ ile belirtilmiş olan kısımda kod üzerinde istediği adı yazıp direk kaydedebilir. Fakat varsayılan olarak görüntülerin adları yaptığı işlemin adı olarak belirtilmiştir.



**2-) NOKTA BAZLI İŞLEMLER**

Nokta bazlı işlemler, gri seviye bir resimdeki her piksel için aynı dönüşüm işleminin yapılmasına denir. Dönüştürme işlemi sadece orijinal piksel üzerinden yapılır. Konumu ve komşu piksellerden bağımsızdır.

F(x, y) giriş görüntüsünün ve g(x, y) çıkış görüntüsünün (x, y) noktalarındaki gri değerleri ‘r’ ve ‘s’ olsun. Nokta bazlı işlem şu şekilde tanımlanır :

s = T (r)

T, orijinal görüntü ve çıkış görüntüsü arasındaki belli gri seviye haritalanması nokta operatörüdür. Yani dönüşüm fonksiyonudur. Genel olarak nokta operatörleri gri seviye dağılımını değiştirmek için kullanılır.

**EŞİKLEME**

Görüntü eşikleme kolay ve işlevsel bir görüntü ayırma yolu olup görüntüyü önplan ile arkaplana böler. Bu görüntü analiz tekniği, bir çeşit görüntü bölükleme olup gri seviye görüntüleri binary görüntülere dönüştürüp nesneleri izole etmektedir. Görüntü eşikleme yüksek seviyeli kontrast içeren görüntüler için çok efektif olarak çalışmaktadır.

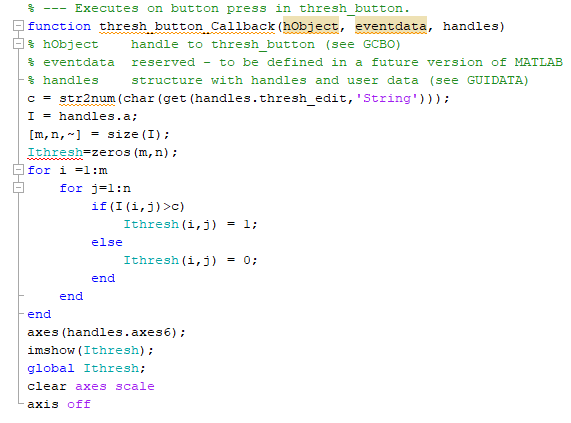
En basit eşikleme metotları şu şekilde çalışmaktadır : belirlenmiş bir T sabiti üzerindeki görüntü şiddetindeki her pikseli beyaz piksel ile; T sabiti altındakileri de siyah piksel ile değiştirir. Renkli görüntülerde birden fazla eşik değeri belirlenip her bir renk kanalı için geçerli olan RGB uzayı değeri ile işlem yapılabilir.

Yani, belirli bir eşik düzeyi belirlenir ve görüntü üzerindeki her bir piksel bu sisteme göre farklı değerlere dönüştürülür. Piksellerin her biri bu sistemden geçerek yeni değerlere sahip olur.



**Şekil 4.** Eşikleme işlemi sonrası görüntü

Yukarıda verilen şekil 4’te GUI içerisine yüklenen renkli resim, ‘166’ eşik değeri parametresi üzerinden işlem yapılarak eşiklenmiştir. 166 değerinin üstündeki değerler siyah, 166 değerinin altındaki değerler ise beyaz olarak dönüştürülmüştür.



**Kod 1.** Eşikleme Kodu

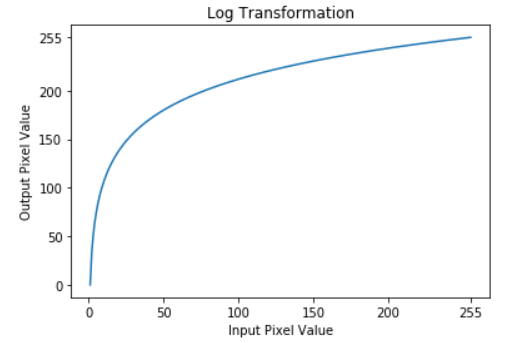
**LOG DÖNÜŞÜMÜ**

Logaritmik dönüşüm, çok kolay bir işlemdir. Görüntü üzerindeki her bir piksel değerinin logaritması alınır ve alınan değerlerle yeni görüntü oluşur.

Genel olarak logaritmik dönüşüm formülü aşağıdaki gibidir.

s = c\*log(1+r)

‘s’ ve ‘r’ değerleri çıkış ve giriş, c ise sabittir. Logaritmik fonksiyonlar 0 ve negatif değerleri alamazlar. Bu yüzden r giriş değerine 1 eklenerek, piksel değeri 0 olan görüntünün, log(0) değerinin sonsuz olması önlenerek en minimum değer 1 olarak ayarlanır.



**Şekil 5.** Logaritmik dönüşüm ile piksel değerlerin değişimi

Yukarıda verilen şekil 5’te giriş piksel değerlerinin logaritma dönüşümleri sonucu çıkış piksel değerlerinin grafiği verilmiştir. Burada c sabiti :

c = 255/(log(1 + max\_input\_pixel\_value))

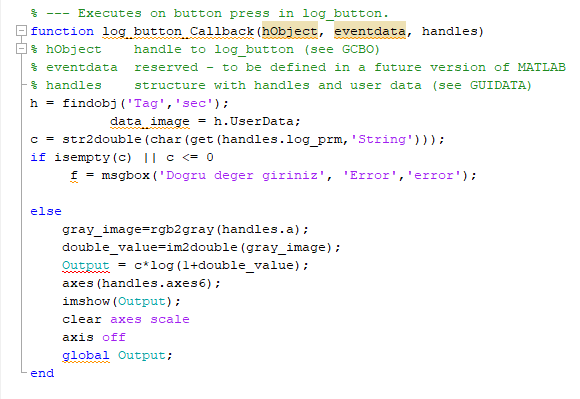
formülüyle bulunmuştur. 255 değeri, 8-bit görüntülerdeki maksimum piksel değeri olacağından, bu logaritmik dönüşüm 8-bit görüntüler için tasarlanmıştır.

Fakat yapılan GUI sisteminde, logaritmanın c sabiti kullanıcı girişine göre ayarlanmıştır. O yüzden kullanıcının girmiş olduğu c sabiti ile kurulan denklem, ona göre hesaplama yaparak görüntüyü değiştirmektedir. Yukarıdaki şekil 5’ten de anlaşılacağı üzere, c sabitinin değeri arttıkça görüntüdeki pikseller beyazlaşacak, değeri 0’a ne kadar yakın bir pozitif sayı olursa o kadar siyahlaşacaktır.



**Şekil 6.** Logaritmik dönüşüm sonrası görüntü

Yukarıda verilen şekil 6’da, GUI üzerinde yapılan logaritmik dönüşüm işlemi gerçekleştirilmiştir. C sabitinin değeri 2.5 girilmiştir ve çıktı ona göre oluşmuştur.

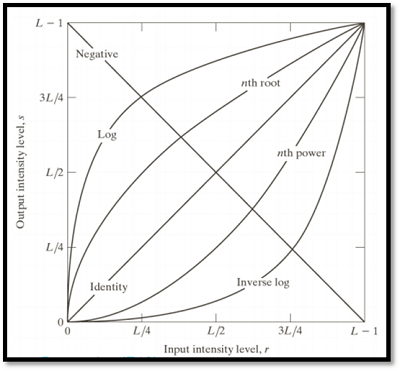


**Kod 2.** Logaritma Kodu

**GRİ SEVİYE DÖNÜŞÜMÜ**

Gri seviye dönüşümü görüntü iyileştirmede piksellerle işlem yapan önemli bir dönüşümdür. Bu işlem sonucu oluşan sonuçlar ya resim ya da karakteristik özellik gösteren bir küme olabilir. Basit efekte sahiptir fakat uygulaması karışıktır.

Resmin görselliği genelde iki özellikle belirtilir : parlaklık ve kontrast. Parlaklık ortalama şiddet seviyesiyle gösterilir ve görüntüdeki bütün piksellerin gri seviye şiddet değerlerinden etkilenirler. Gri seviye değerleri yüksek olan görüntünün parlaklığı daha yüksek olacağından, daha fazla ortalama şiddet değeri bulunduracaktır.



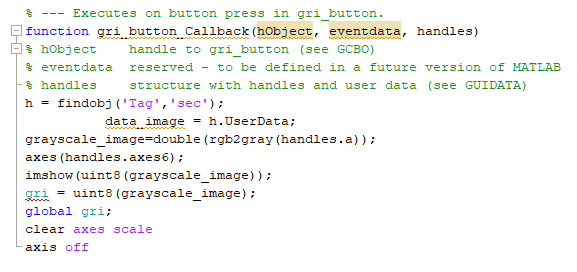
**Şekil 7.** Gri seviye dönüşümleri

Temel olarak 3 çeşit gri seviye dönüşümü vardır ve bunların grafiksel görünümü yukarıdaki şekil 7’de verilmiştir.

Aşağıda verilen şekil 8’de GUI üzerinden gri seviye dönüşümü yapılmıştır. Görüntünün piksellerindeki gri seviye değerleri kullanılarak yeni bir görüntü oluşturulmuştur.



**Şekil 8.** Gri seviye dönüşümü sonrası görüntü



**Kod 3.** Griseviye Kodu

**KONTRAST GERME**

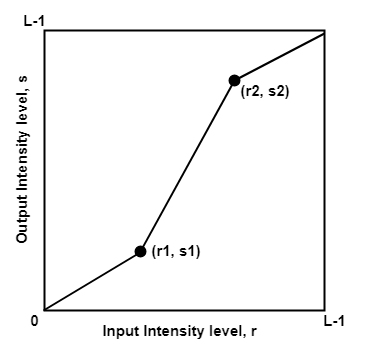
Kontrast germe dönüşümünün amacı bir görüntünün içindeki farklı parçalar arasındaki kontrastı arttırmaktır. Yani belirlenen alanların gri kontrastını arttırır, geri kalan alanların ise gri kontrastını bastırır. Kontrast germe genellikle normalizasyon olarak da adlandırılır. Görüntü üzerindeki piksel değerlerinin aralıklarında şiddetlerini istenilen aralık değerleri arasına yaymaktır yapılan işlem. Histogram eşitlemenin aksine, kontrast germe sadece linear artan fonksiyon uygulayabilir görüntü piksel değerlerine. Bu yüzden görüntü iyileştirme daha yumuşaktır. Genel olarak da gri seviye görüntü için uygulanır ve sonuç olarak başka bir gri seviye görüntüsü ortaya çıkar.

Kontrast germe işlemi yapılabilmesi için öncelikle yapılması gereken işlem alt ve üst piksel değerleri limitlerini belirlemektir. Bu limitler görüntünün piksel değerlerinin alabileceği sınırları belirlemektedir. Örneğin 8-bit gri seviye görüntülerinde alt limit 0 ve üst limit 255 olabilir.

Alt ve üst limitlere sırasıyla *a* ve *b* diyelim. En basit normalizasyon görüntüyü tarar ve görüntü üzerindeki en düşük ve en büyük piksel değerlerini bulur. Bunlara *c* ve *d* diyelim. Ve her *P* pikseli şu şekilde artar :

Eqn:eqnstr1

0 altındaki değerler 0’a, 255 üstündeki değerler ise 255 olur.



**Şekil 9.** Kontrast Germe fonksiyonu

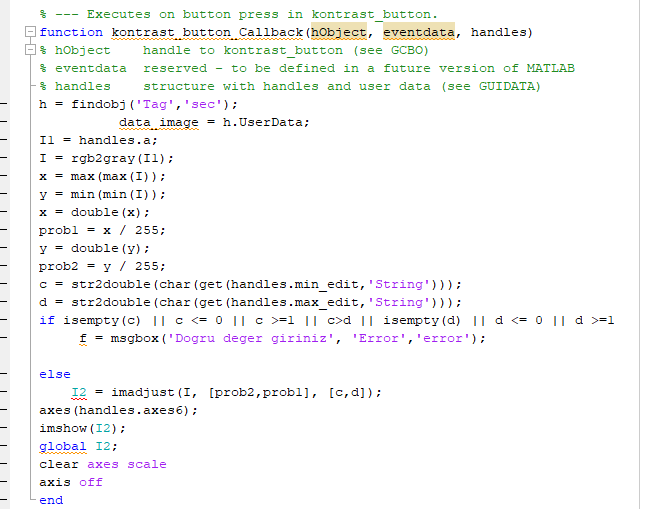
(r1,s1) ve (r2, s2) noktalarının konumunu değiştirerek, dönüşüm fonksiyonunun şeklini değiştirebiliriz.

1. r1 =s1 ve r2=s2 olduğu zaman, dönüşüm Linear fonksiyon olur.
2. r1=r2, s1=0 ve s2=L-1 olduğu zaman, dönüşüm eşikleme fonksiyonu olur.
3. (r1, s1) = (rmin, 0) ve (r2, s2) = (rmax, L-1) olduğu zaman, Min – Max Germe olur.
4. (r1, s1) = (rmin + c, 0) ve (r2, s2) = (rmax – c, L-1) olduğu zaman, Yüzdelik Germe olur.

Aşağıda verilen şekil 10’da GUI üzerinde yapılan kontrast germe işlemi verilmiştir. İşlem için kullanıcıdan iki tane parametre istenmektedir. Kontrastı germek için minimum ve maksimum değerleri kullanıcıdan alınarak, piksel değerleri o aralıklar arasında gerilir. Yapılan bu işlemde kullanıcıdan ‘0.3’ ve ‘0.6’ değerleri alınmıştır.



**Şekil 10.** Kontrast germe işlemi sonrası görüntü



**Kod 4.** Kontrast Kodu

**HİSTOGRAM EŞİTLEME**

Histogram görüntü üzerindeki şiddetlerin dağılımının grafiksel gösterimidir. Yani, her bir şiddet değeri için piksel sayısını göstermektedir.

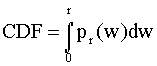
Histogram eşitleme kontrastı büyütmek için kullanılır. Bu sadece kontrastı arttırmakla sınır değildir, bazen kötü bir histogram eşitleme yapılırsa kontrast azaltılarak değerler düşürülür. Histogram eşitleme kontrastı iyileştirir diyebiliriz. Bunu yapmak için çok sık tekrarlanan şiddet değerlerini efektif bir şekilde dağıtır. Bir başka deyişle; resimdeki şiddet değerlerinin aralığını gerer. Böylelikle düşük kontrast alanları yüksek kontrast sahibi olur.

Renkli görüntülerde histogram gösterilirken her renk parçasının piksel sayısı belirlenir. Histogram eşitleme yaparken, kırmızı, yeşil ve mavi renk parçalarına ayrı ayrı işlem yapılamaz çünkü bu görüntünün renk dengesini çok bozar. Fakat görüntü öncelikle HSL/HSV gibi başka bir renk uzayına dönüştülür ise şiddet değerlerine rengin doygunluğunu etkilemeden algoritma uygulanabilir.

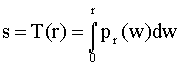
Histogram eşitleme, *r* kümülatif olasılık dağılımını bulup, *r* için *s* biçiminde çözüp ve bunu *T(r)* fonksiyonu için kullanarak elde edilmektedir. Olasılık şiddet fonksiyonu, *s* değerleri için şu şekilde tanımlanır :

https://academic.mu.edu/phys/matthysd/web226/images/Image133.gif

Kümülatif dağılım fonksiyonu *r* için bu formül ile elde edilir :



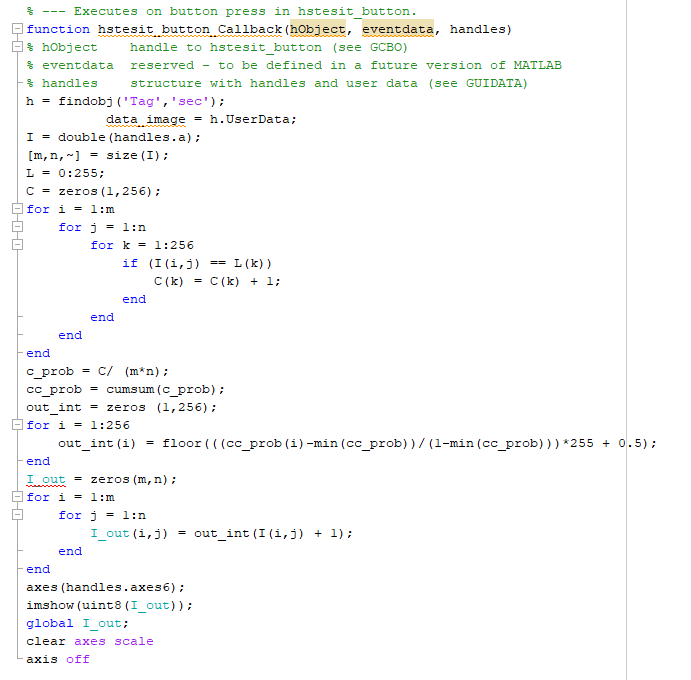
Bu dağılım fonksiyonu *T(r)* için kullanınca :



Aşağıda verilen şekil 11’de GUI üzerinde yapılan histogram eşitleme işlemi bulunmaktadır. Görüntü öncelikle double hale getirilir ve ona göre dağılım yapılır.



**Şekil 11.** Histogram eşitleme sonrası görüntü



**Kod 5.** Histogram Eşitleme Kodu

**3-) FİLTRE İŞLEMLERİ**

Görüntü işlemede, filtreler genellikle resim üzerindeki yüksek frekansları bastırmak için kullanılır. Örneğin, görüntüyü yumuşatma yani düşük frekanslar, görüntüdeki kenarları iyileştirme veya bulma.

Görüntü, frekans veya uzaysal bölgeler filtrelenebilir.

Frekans bölgelerindeki görüntü dönüştürme, görüntüyü frekans filtre fonksiyonu ile çarpar ve sonucu uzaysal bölgeye yeniden dönüştürür. Filtre fonksiyonu bazı frekansları seyreltirken diğerlerini arttırarak şekillenmiştir. Örneğin, kesik frekanstan az olan düşük geçişli fonksiyon *1* olur, diğerleri *0* olur.

Uzaysal bölgedeki belirtilen işlem giriş görüntüsü *f(i,j)* ile filtre fonksiyon *h(i,j)* ile sarmalanır. Bunu şu şekilde ifade edebiliriz :

Eqn:eqnspin1

Matematiksel işlem frekans uzayındaki çarpım ile aynıdır fakat dijital uyarlamanın sonuçları değişkenlik gösterir çünkü sonlu ve ayrık kernel ile filtre fonksiyonu yaklaştırmak zorundayız.

Ayrık konvolüsyon ‘*kaydır ve çarp*’ işlemi olarak tanımlanabilir. Kernel görüntü üzerinde kaydırılır ve ona örtüşen görüntüdeki piksel değerleri ile çarpılır. *M x M* boyutundaki kare kernel için, çıkış fonksiyonu şu formül ile hesaplanır.

Eqn:eqnspin2

**MEAN**

Mean yani ortalama filtresi basit bir şekilde uyarlanan bir filtredir. Kolayca görüntü üzerinde uygulanabilir. Görüntüyü yumuşatmak için kullanılan bu filtre, bir piksel ile diğeri arasındaki şiddet farklılığı sayısını azaltır.

Mean filtre kolay bir uzaysal filtre olup, kernel üzerindeki orta noktanın değerini, kernel üstündeki diğer değerlerin ortalamasıyla değiştirir. Kernel genellikle kare şeklindedir fakat başka şekillerde de olabilir. 3x3 boyutundaki kernel örneği aşağıda verilmiştir.

|  |  |  |
| --- | --- | --- |
| 5 | 3 | 6 |
| 2 | 1 | 9 |
| 8 | 4 | 7 |

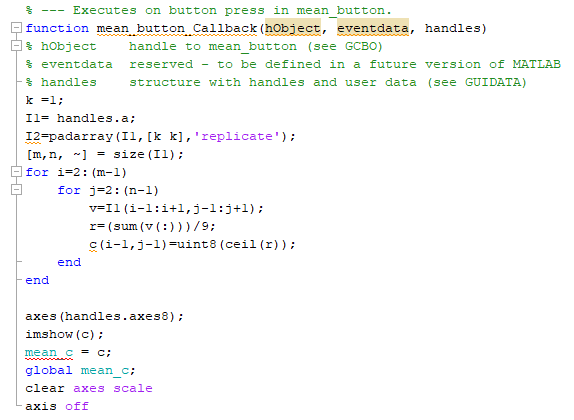
Şeklindeki 3x3 matris filtrelenmemiş değerleri göstermektedir. Mean filtre uygulandıktan sonraki hali şu şekilde olmaktadır.

|  |  |  |
| --- | --- | --- |
| \* | \* | \* |
| \* | 5 | \* |
| \* | \* | \* |

Aşağıdaki şekil 12’de GUI üzerinde yapılan mean filtresi gözükmektedir.



**Şekil 12.** Mean filtresi uygulandıktan sonra görüntü



**Kod 6.** Mean Kodu

**MEDIAN**

Median yani ortanca değer filtresi genel olarak mean filresimde gürültüyü azaltmak için kullanılır. Fakat mean filtreye göre görüntüdeki detayları korumakta daha iyi çalışmaktadır.

Mean filtresi gibi, median filtresi görüntüdeki her bir pikseli ele alır ve komşu piksellerine bakarak onlarla ilişkisi olup olmadığına bakar. Komşu piksellerin değerlerinin ortalamasına bakmak yerine, median değerine bakar. Yani ortanca değeri bulur. Median, komşu piksellerin değerlerinin numerik olarak sıralanıp ondan sonra ortadaki değeri bularak hesaplanır. Eğer ortadaki değer tek değil çift iki sayıdan oluşmakta ise bunların ortalamasını alarak hesaplanır.

Median filtresi yüksek uzaysal frekansların çoğunu geçirir. Bu sürede komşu piksellerin yarısından az olan kısmının etkilendiği durumlarda görüntüdeki gürültüyü azaltmakta başarılı bir yöntemdir.

|  |  |  |
| --- | --- | --- |
| 6 | 2 | 0 |
| 3 | 97 | 4 |
| 19 | 3 | 10 |

Yukarıda 3x3’lük bir matris verilmiştir. Piksel değerleri üzerinde yazmaktadır. Buna göre median filtresi uygulanırsa matris şu hale gelecektir :

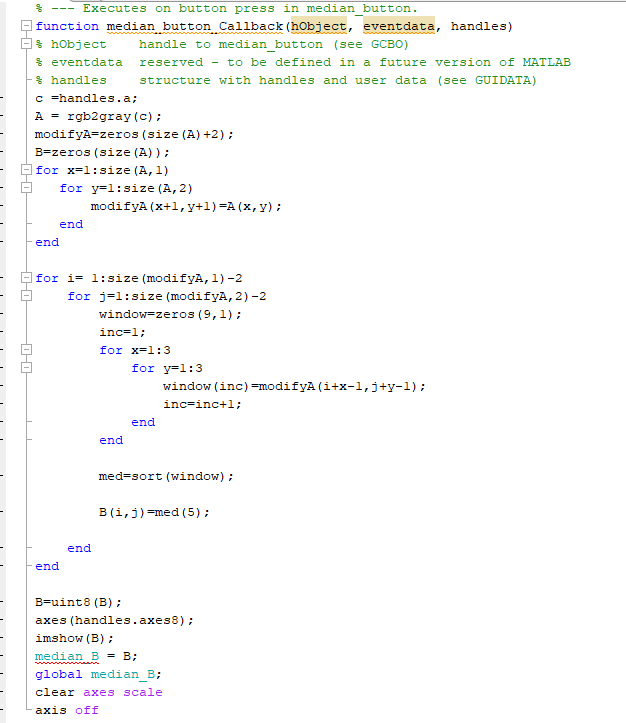
|  |  |  |
| --- | --- | --- |
| \* | \* | \* |
| \* | 4 | \* |
| \* | \* | \* |

Sayılar artacak veya azalacak şekilde sıralandıktan sonra ortanca değer 4 olarak bulunur.



**Şekil 13.** Median filtresi uygulandıktan sonra görüntü

Yukarıda verilen şekil 13’te GUI üzerinden yapılan Median filtresi gösterilmektedir. İşlem sırasında görüntüdeki gürültü azalır. Mean filtresine ne kadar benzese de çıktı sonuçları karşılaştırıldığında çok farklı bir görüntü ortaya çıktığı görülmektedir.



**Kod 7.** Median Kodu

**GAUSS**

Gaussian filtresi, görüntüyü bulanıklaştırmak için kullanılan bir filtredir. Görüntüyü bulanıklaştırırken, gürültüyü azaltır ve detayları düşürür. Bu yüzden mean filtresine benzer yapıda çalışmaktadır fakat farklı bir kernel ile tasarlanmıştır. Gaussian denilen ‘zil’ biçimindedir.

Gaussian yumuşatmanın amacı 2-D dağılımı ‘nokta-dağılım’ fonksiyonu olarak kullanmak. Görüntü ayrık pikseller olarak kaydedildiği için, konvolüsyon uygulamadan önce Gaussian fonksiyonuna ayrık hesaplama yapmalıyız. Gaussian dağılımı teorik olarak her yerde 0 harici bir değerdir, bu da sonsuz büyüklükte konvolüsyon kernel ihtiyacı doğurur. Fakat uygulamada efektif olarak ortalamadan üç standart sapmaya göre daha fazla sıfır vardır.

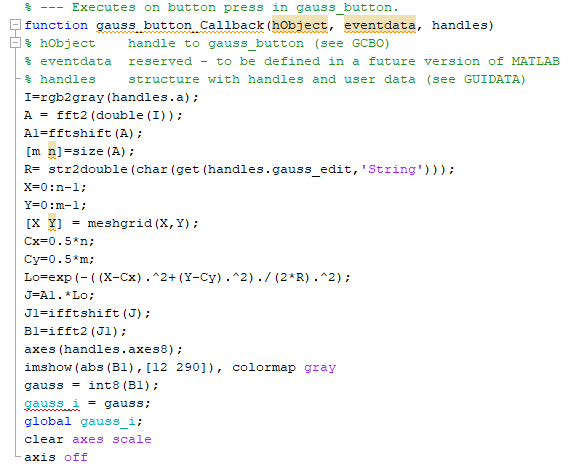
2 boyutlu işlemlerde Gaussian dönüşüm fonksiyonunun formülü şu şekildedir :

Eqn:eqngaus2

Burada *x* orijinden yatay eksendeki uzaklığı, *y* orijinden dikey eksendeki uzaklığı ve *σ* ise Guasssian dağılımın standart sapmasını göstermektedir. Aşağıdaki şekil 14’te Gauss filtresi uygulanan görüntü gösterilmiştir. Gauss filtresinin bulanıklığı kullanıcı tarafından ‘Gauss Parametresi’ olarak istenmektedir. Kullancının girmiş olduğu değer arttıkça bulanıklık azalmakta, küçük değerlerde ise bulanıklık çoğalmaktadır. Yapılan bu işlemde kullanı parametre olarak ‘6’ rakamını girmiştir ve ona göre filtre işlemi uygulanmıştır.



**Şekil 14.** Gauss filtresi sonrası görüntü



**Kod 8.** Gauss Kodu

**SOBEL**

Görüntü işlemenin diğer bir önemli işlemlerinden diğeri kenar bulmadır. Kenar bulma, görüntü içerisindeki keskin şiddet veya renk değişimleri olan alanları bulmaktadır. Yüksek değer dik değişimi, düşük değer ise sığ değişimi göstermektedir.

Kenar bulma işleminin en yaygın operatörü Sobel’dir. Sobel, görüntünün türevinin bir hesaplamasıdır. Y ve X yönlerinde farklıdır. X yönüne baktığımızda, görüntünün x yönündeki eğimi bu operatöre eşittir. 3x3 biçimindeki matrisi her bir x ve y yönü için kullanırız. Sol tarafta x yönündeki negatif sayılar ve sağ tarafta x yönündeki pozitif sayılar ve ortayı koruyoruz. Buna benzer olarak y yönündeki eğim yukardaki sayılar için pozitif, aşağıdaki sayılar için ise negatif olarak bulunur ve ortadaki kısım korunur.

|  |  |  |
| --- | --- | --- |
| +1 | +2 | +1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

|  |  |  |
| --- | --- | --- |
| -1 | 0 | +1 |
| -2 | 0 | +2 |
| -1 | 0 | +1 |

G(x) G(y)

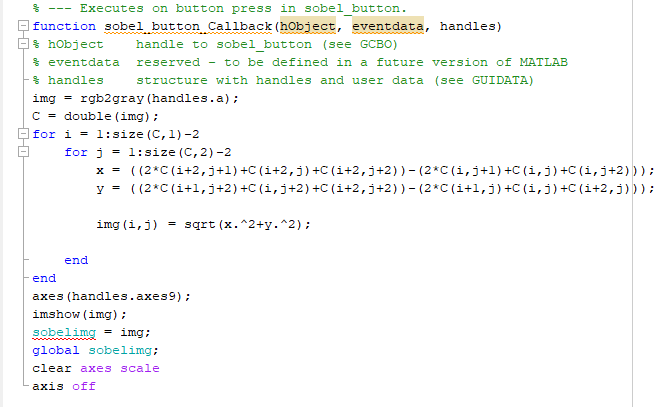
Sobel operatörü ile yapılması istenen, görüntü üzerindeki her piksel üzerine eğim matrisi konularak farklılıkların sayısı bulunur. X yönü için bir, Y yönü için bir görüntü elde ederiz.

Dijital görüntünün şiddet fonksiyonu sadece ayrık noktalarda bilindiği için, fonksiyonun türevleri görüntüdeki noktalardan alınan farklılaşabilen şiddet fonksiyonu olmadığı sürece tanımlanamaz. Buna ek varsayımlar yapılarak, sürekli şiddet fonksiyonunun türevi örneklenen şiddet fonksiyonu olarak hesaplanır. Böylelikle belirlenen noktalardaki türevler sanal görüntü noktalarındaki şiddet değerlerinin fonksiyonları olarak bulunur.

Aşağıda verilen şekil 12’de GUI üzerinde yapılan Sobel işlemi verilmiştir. Kenarların görünürlüğü artarak kenar bulma işlemi gerçekleştirilmiştir.



**Şekil 15.** Sobel filtresi uyguladıktan sonra görüntü



**Kod 9.** Sobel Kodu

**PREWITT**

Prewitt operatörü görüntüdeki kenar bulma işlemini yapan başka bir operatördür. Görüntünün eğimini hesaplayan bir şiddet fonksiyonudur. Görüntünün her noktasında, Prewitt operatörünün sonucu ya ilgili eğim vektörü ya da vektörün normudur. Prewitt işlemi görüntüyü küçük, ayrıştırılabilir ve sayısal değerlerle yatay ve dikey yönlerde sarmalar. Bunun yanısıra, eğim hesaplaması göreceli olarak Sobel operatörüne göre daha katıdır, özellikle görüntü içindeki yüksek frekanslı değişkenlerde.

Temel olarak, operatör görüntünün her noktasındaki eğimi hesaplar, olabilecek en yüksek aydınlıktan karanlığa artışa yön verir ve o yöndeki değişimin hızını gösterir. Bu yüzden görüntünün o noktadaki yumuşak veya sert geçişleri dikkat çekerek kenarları gösterdiği anlaşılır.

İki tip kenarları bulabilir : yatay kenarlar ve dikey kenarlar.

Kenarlar, bir görüntünün ilişkili piksel şiddetlerinin farklılığıyla ölçülür. Kenar bulma işlemi için kullanılan bütün maskeler türev maskeleri olarak da bilinir. Prewitt operatörü de iki tane de maskeye sahip olarak hem yatay yönde hem de dikey yönde kenar bulma işleminde kullanılır.

|  |  |  |
| --- | --- | --- |
| -1 | 0 | +1 |
| -1 | 0 | +1 |
| -1 | 0 | +1 |

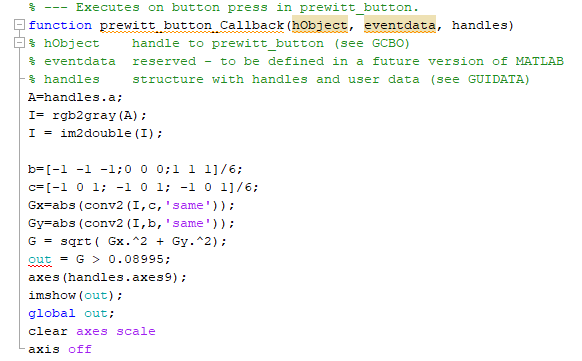
|  |  |  |
| --- | --- | --- |
| -1 | -1 | -1 |
| 0 | 0 | 0 |
| +1 | +1 | +1 |

Yukarıda sırasıyla yatay yönde ve dikey yönde kullanılan matrisler verilmiştir. Bu matrisler uygulandığında; ilk matrise göre yatay kenarlar, ikinci matrise göre ise dikey kenarlar bulunur. İlk türev alınır ve kenar bölgedeki piksel şiddetlerinin farkları hesaplanır. 0’ların bulunduğu satır veya sütuna dokunulmaz ve kenar etrafındaki komşu piksel değerleri arasındaki fark hesaplanır. Böylelikle kenarların şiddeti artar ve orijinal görüntüye göre daha fazla önplana çıkar.

Aşağıda verilen şekil 16’da GUI üzerinde yapılan Prewitt filtresi gösterilmektedir. Kenar değerlerindeki değişiklik ile kenarlar rahatlıkla bulunmaktadır.



**Şekil 16.** Prewitt filtresi sonucu görüntü



**Kod 10.** Prewitt Kodu

**LAPLACE**

Laplacian operatörü başka bir türev operatörü olup görüntü içindeki kenarları bulmak için kullanılmaktadır. Diğer operatörler olan Prewitt ve Sobel’den en büyük farkı, diğerleri birinci türev maskeleri olmalarına rağmen Laplacian operatörünün ikinci türev maskesi olmasıdır. Yani birinci türevlerin değişiminin oranını vermektedir.

Bir başka fark ise Laplace operatörü görüntüdeki kenarları belli bir yönde bulmamaktadır. İki tane sınıfta incelenir :

* İçeriye Kenarlar
* Dışarıya Kenarlar

Bir görüntünün laplace’ı, ani şiddet değişikliklerinin olduğu bölgeleri vurgular ve böylelikle görüntüdeki kenarların ani şiddet değişikleri yapmasından dolayı kenar bulmada kullanılır. Laplace operatörü genelde bir görüntü üzerinde öncelikle Gauss yumuşatma filtresi ile yumuşatma uygulanarak görüntüdeki gürültülerin azalması sonrası uygulanır. Tek bir gri seviye görüntü kullanılarak giriş yapılır ve çıkış olarak yine tek bir gri seviye görüntü elde edilir. Bir görüntünün I(x,y) piksel şiddet değerlerindeki Laplace L(x,y) olarak gösterilir ve şu şekilde tanımlanır :

Eqn:eqnlog1

Konvolüsyon filtresi kullanarak bu hesaplama rahatlıkla yapılabilir.

Giriş görüntüsü ayrık piksellerin kümesi olarak gösterildiği için, Laplace tanımındaki ikinci türevleri hesaplamak için ayrık konvolüsyon kernel’i bulmak gereklidir.

Aşağıda çok sıklıkla kullanılan iki temel Laplace kernel maskeleri verilmiştir.

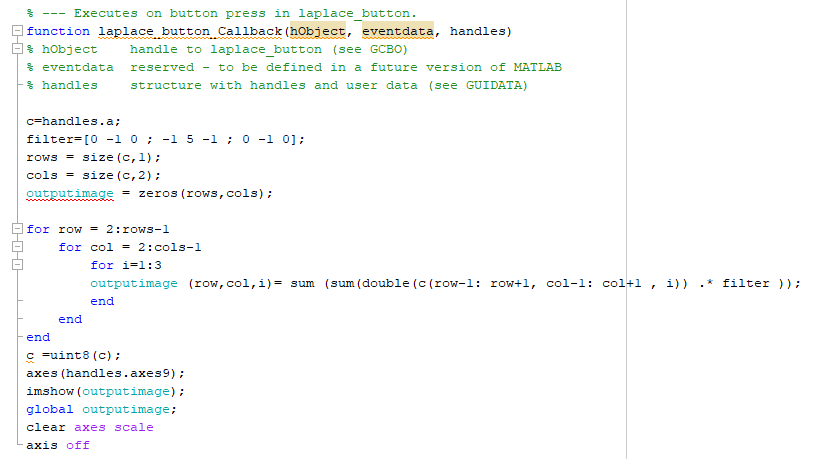
|  |  |  |
| --- | --- | --- |
| -1 | -1 | -1 |
| -1 | 8 | -1 |
| -1 | -1 | -1 |

|  |  |  |
| --- | --- | --- |
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |



**Şekil 17.** Laplace filtresi sonrası görüntü

Yukarıda verilen şekil 17’de GUI üzerinde yapılan Laplace filtresi işlemi verilmiştir. Görüntü diğer örneklerin aksine renkli şekilde işleme sokulup kenar bulma işlemi öyle yapılmıştır. Bu yüzden kenarların belli olduğu yerler renkli kalıp, geri kalan kısımlar beyazlaştırılmıştır.



**Kod 11.** Laplace Kodu

**ROBERTS**

Roberts operatörü diğer operatörler gibi basit şekilde uygulanıp, hızlı şekilde işlenip, 2-D uzaysal eğim hesabı yapmaktadır görüntü üzerinde. Bu yüzden yüksek uzaysal frekans bölgelerini ayırt ederer, yani kenarları. En yaygın kullanımında, operatöre verilen giriş gri seviyede olan bir resimdir ve çıkış olarak yine gri seviyede bir görüntü alınır. Çıkış görüntüsündeki her bir piksel değeri, giriş görüntüsünün o noktadaki hesaplanan mutlak uzaysal eğiminin büyüklüğünü gösterir.

Roberts operatörü ilk oluşturulan operatörlerden biridir. Differansiyel operatörü olmasından dolayı, Roberts operatörü bitişik çapraz pikseller arasındaki farkların karelerinin toplamını bulup, ayrık diferansiyel ile bir görüntüdeki eğimi hesaplamaktadır.

Roberts operatörü teorik olarak 2x2 büyüklüğünde bir çift konvolüsyon kernel biçiminde oluşmaktadır. Bir kernel diğerinin 90 derece çevrilmiş halidir.

|  |  |
| --- | --- |
| 0 | +1 |
| -1 | 0 |

|  |  |
| --- | --- |
| +1 | 0 |
| 0 | -1 |

G(x) G(y)

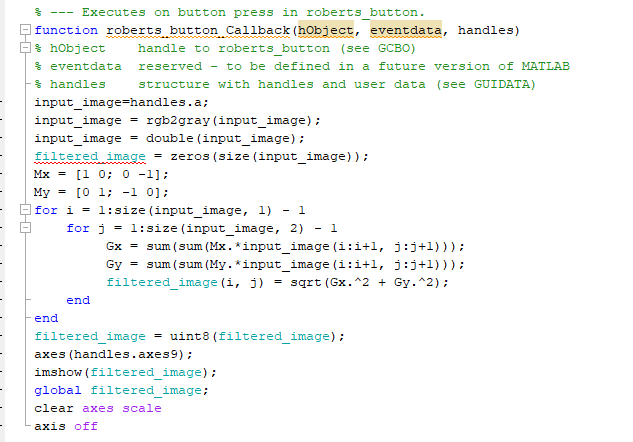
Bu kernel yapıları piksel tablosunda 45 derecelik açı yapacak şekilde kenarlara maksimum dönüş sağlar. Kernel yapıları giriş görüntüsüne ayrı ayrı uygulanarak her yönlendirmeye ayrı eğim bulunabilir. Bunlara *Gx* ve *Gy* adını verelim. Bunlar birleştirilip her noktadaki mutlak eğim büyüklüğü bulunabilir ve eğimin yönlendirmesi hesaplanabilir. Eğim büyüklüğü şu şekildedir :

Eqn:eqnrob2

Buna göre aşağıda verilen şekil 18’de GUI üzerinde Roberts işlemi yapılan görüntü bulunmaktadır. İşlem sonucunda kenarlar beyazlaştırılarak önplana çıkmakta, diğer kısımlar ise karartılarak görünürlükleri azalmaktadır.



**Şekil 18.** Roberts filtresi sonucu görüntü



**Kod 12.** Roberts Kodu

**4-) MORFOLOJİK İŞLEMLER**

Binary görüntüler, genellikle siyah ve beyaz olmak üzere sadece iki renkten oluşan pikselleri barındırmaktadır. Binary görüntüler çok fazla sorun içerebilir. Mesela, gürültü ve yapı bozulabilir binary alanlar eşik üzerinden üretildiğinde. Görüntü işlemede morfolojik işlemler bu sorunları kaldırararak görüntünün yapısını ve formunu düzeltir.

Görüntü işlemedeki morfolojik işlemler, doğrusal olmayan şekil veya görüntüdeki morfolojik özellikler bütünüdür. Morfolojik işlemler piksel değerlerinin numerik sıralamasını değil, göreceli sıralamasıyla ilgilenir ve bu yüzden binary görüntülerde çalıştırılır. Bu teknikler gri seviyeli görüntülerde de kullanılabilir. Işık transfer fonksiyonları bilinmemekte ve mutlak piksel değerleri düşük önceliktedir.

Küçük bir şekil veya şablon halindeki ‘yapısal eleman’, görüntüdeki işlenecek olan piksellerin matrisidir ve her pikselin işlenmesindeki komşuluğu tanımlar. Giriş görüntüsü üzerindeki olabilecek her lokasyona konumlandırılabilir ve ilgili piksellerin komşularıyla karşılaştırılır.

**AŞINMA**

Aşınma matematik morfolojinin temel iki işleminden birisidir, diğeri genişlemedir. Genel olarak binary görüntülere uygulanır, fakat gri seviye görüntülere de uygulanabilen versiyonları vardır. Binary görüntüdeki temel efekt önplandaki piksel alanlarının sınırlarını aşındırır genel olarak beyaz piksellerdir bunlar. Böylelikle önplandaki pikseller boyut olarak küçülür ve bu alanlardaki delikler genişler.

Aşınma operatörü iki parka veri alır giriş olarak. İlki aşınma işlemine uğrayacak olan görüntüdür. İkincisi ise genelde küçük koordinat noktaları kümesidir. Buna yapısal elemanlar ya da kernel denir. Yapısal eleman giriş görüntüsü üzerindeki aşınma efektini tam olarak yapan kısımdır. Aşınmanın binary görüntülerdeki matematiksel tanımı şu şekildedir :

*X,*  giriş binary görüntüsündeki öklid koordinatlarında küme olup, *K* ise yapısal elemanın koordinat kümesidir.

*Kx*, *K*’nın x üzerindeki orijinini göstermektedir.

Gri seviye görüntülerde aşınma aynıdır tek farkı koordinatlarn 2 boyut yerine 3 boyut üzerinde olmasıdır.

Binary görüntünün aşınmasını yaparken, giriş görüntüsündeki her önplan pikseliyle ilgilenilir. Her önplan pikseli için, yapısal eleman görüntünün üzerine yerleştirilir ve yapısal eleman ile giriş piksel koordinatları çakışır. Yapısal elemandaki her piksel için, görüntünün altındaki ilgili pikseller önplan pikselleridir. Eğer görüntüdeki ilgili piksel arkaplan ise giriş pikseli arkaplan değerini alır.

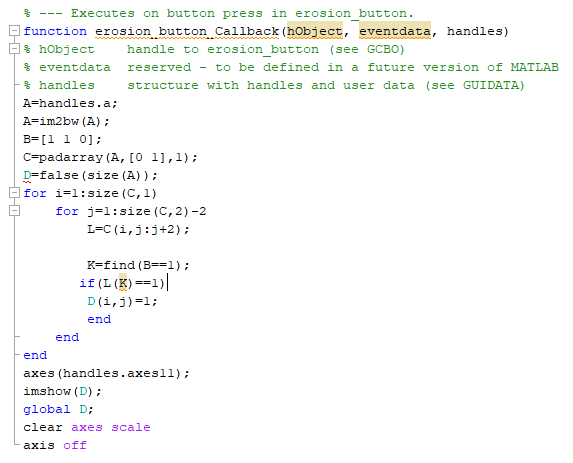
3x3 matrisindeki yapısal eleman için, bu operasyonun efekti beyaz piksellerle tamamen kapanmamış olan önplan piksellerini kaldırmaktır. Bu pikseller beyaz alanların kenarlarındadır ve uygulamada önplan alanları küçülür ve alan içindeki delikler genişler.

Bu operatör genellikle binary görüntüler üzerinde kullanıldığı için, önplan pikselleri genellikle 255 şiddet değerine, arkaplan pikselleri ise 0 şiddet değerine atanır. Bu yüzden genelde eşikleme yapılarak gri seviye görüntü binary görüntüye dönüştülür.

Aşağıda verilen şekil 19’da GUI üzerinde renkli bir görüntüde aşınma işlemi yapılmıştır. Görüntü binary hale getirilip ondan sonra yapılan işlemde siyah ve beyaz pikseller gözükmektedir.



**Şekil 19.** Aşınma işlemi sonrası görüntü



**Kod 13.** Aşınma Kodu

**GENİŞLEME**

Genişleme matematiksel morfolojinin diğer temel işlemidir. Ne kadar aşınma gibi binary görüntüler için geliştirilmiş olsa da, bu yöntem daha sonra gri seviye görüntülerde de işlem yapılması sağlanmıştır. Genişleme operasyonu giriş görüntüsü içerisindeki şekilleri genişletme ve düzenleme için yapısal eleman kullanmaktadır.

Genişleme, aşınma işlemine çok benzemektedir. Bu operasyonun binary görüntülerdeki efekti, önplan piksel alanlarının sınırlarını genişletmektir. Bu da genellikle beyaz pikselleri içerir. Bu yüzden önplan piksel alanlarının büyüklüğü artar ve bu alanlardaki boşluklar daha da küçülür.

Genişleme operatörü giriş olarak iki tane veri almaktadır. Bunların ilki genişleme operasyonuna girecek olan görüntüyü, ikincisi ise küçük koordinat noktaları kümesi olan yapısal eleman yani kernel’dir. Yapısal elemanın şekli ve boyutları genişleme işleminin değişmesine neden olur.

Binary görüntülerin matematiksel tanımı şu şekildedir :

*X,*  giriş binary görüntüsündeki öklid koordinatlarında küme olup, *K* ise yapısal elemanın koordinat kümesidir.

*Kx*, *K*’nın x üzerindeki orijinini göstermektedir.

Gri seviye görüntülerde genişleme aynıdır tek farkı koordinatlarn 2 boyut yerine 3 boyut üzerinde olmasıdır. Her arkaplan pikseli için, yapısal eleman görüntünün üzerine yerleştirilir ve yapısal eleman ile giriş piksel koordinatları çakışır. Eğer yapısal elemandaki en az bir tane piksel, altındaki görüntünün önplan pikseli ile çakışırsa, giriş pikseli önplan değerini alır. Eğer görüntüdeki her ilgili piksel arkaplan ise, o halde giriş pikseli arkaplan değerinde bırakılır.

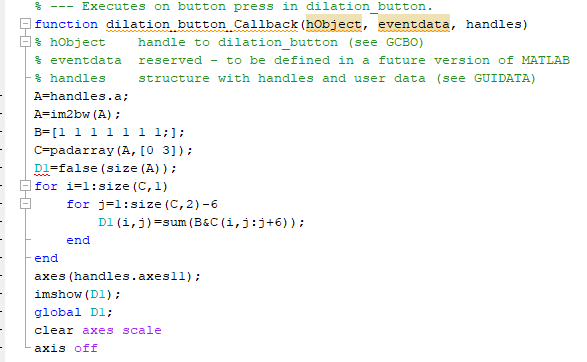
3x3 matrisindeki yapısal elemanda, bu operasyonun sonucu olarak önplan piksel komşularına sahip olan arkaplan piksellerine önplan rengi verilir. Bu pikseller beyaz alanların kenarlarındadır ve önplan alanları büyür ve alan içindeki delikler küçülür.

Genişleme, aşınmanın tam tersi işlem yapmaktadır. Yani alanların hem iç hem de dış sınırlarına piksel katmanı ekler. Önplan pikselleri genişler yani arkaplan pikselleri aşınmaya uğrar.

Aşağıda verilen şekil 20’de GUI üzerinde yapılan genişleme işlemi vardır. Görüntü renkli bir görüntü olduğu için öncelikle binary hale getirilir. Daha sonra siyah ve beyaz parçalar genişleme operatörü ile belirginleşerek arkaplan pikselleri ortaya çıkartılır.



**Şekil 20.** Genişleme işlemi sonrası görüntü



**Kod 14.** Genişleme Kodu

**AÇMA**

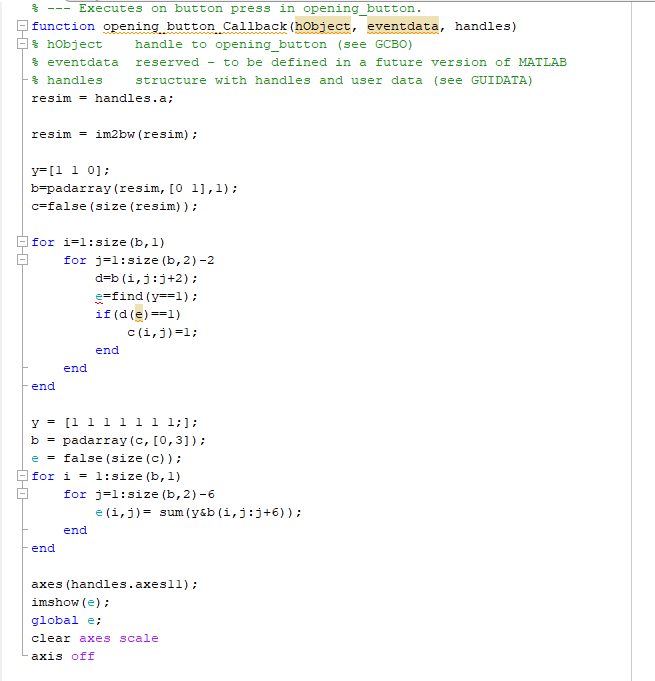
Temel morfolojik işlemler olan aşınma ve genişlemenin dışında, bu iki işlemin bir kombinasyonu olan açma işlemi vardır. Öncelikle görüntü binary hale getirilir. Ondan sonra yapılacak olan ise görüntüyü aşınma işlemine tutup daha sonra genişleme işlemine sokmaktır. Böylelikle açma işlemi gerçekleştirilmiş olur. Açma işlemi, dar bağlantıları ve iki alan arasındaki çizgileri kaldırır.

Açma işlemi öncelikle aşınma daha sonra genişleme işlemi yaparken aynı yapısal elemanı kullanır. Morfolojik açma bir görüntüdeki küçük nesneleri kaldırmak için çok kullanışlı bir operasyondur. Bunun yanısıra bu işlemi yaparken görüntü içerisindeki büyük nesnelerin büyüklüğünü ve şeklini korumaktadır. Bunun yanı sıra ayrıştırılan küçük nesneler arkaplana yerleştirilir.

Aşağıda verilen şekil 21’de GUI üzerinde yapılan açma operasyonü gerçekleştirilmiştir. Görüntü öncelikle binary hale getirilir ve daha sonra önce aşınma daha sonra genişleme işlemi yapılarak açma işlemi sağlanır.



**Şekil 21.** Açma işlemi sonrası görüntü



**Kod 15.** Açma Kodu

**KAPAMA**

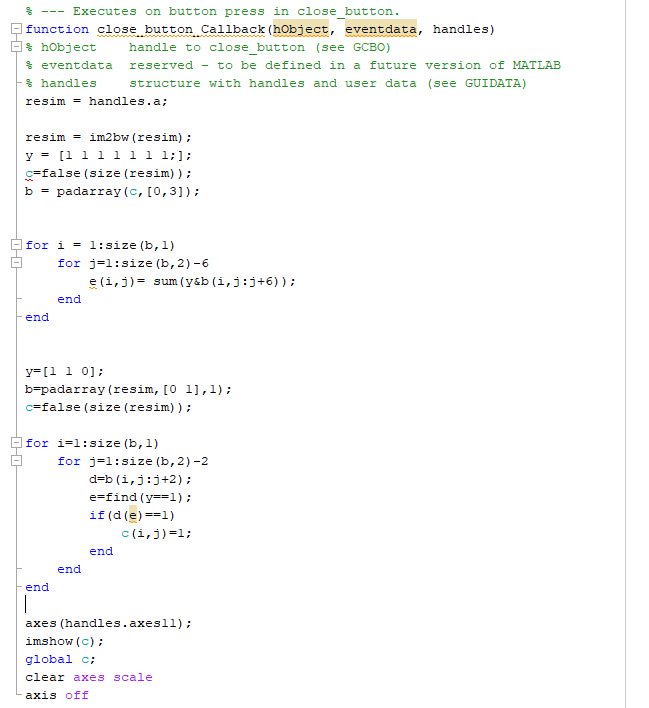
Bir diğer kombinasyonel morfolojik yöntem olan kapama işlemi, temel işlemler olan aşınma ve genişlemenin birleşimidir. Açma işleminin tam tersi olarak çalışmakta olan kapama, önce genişleme işlemi yapılan bir görüntü üzerinde daha sonra aşınma işleminin yapılmasıyla gerçekleştirilir.

Kapama dar karanlık bölgeleri veya delikleri doldurur bir görüntüde. Kapama işlemi yapılırken önce genişleme sonra aşınma işlemi için kullanılan yapısal eleman aynıdır. Farklı yapısal elemanlar üzerinden yapılmaz. Morfolojik kapama işlemi görüntüdeki küçük delikleri kapamak ve doldurmak için çok kullanışlıdır. Bunu yaparken de görüntüdeki nesnelerin boyutu ve şekli korunur.

Aşağıda verilen şekil 22’de GUI üzerinde yapılan kapama operasyonü gerçekleştirilmiştir. Görüntü öncelikle binary hale getirilir ve daha sonra önce genişleme daha sonra aşınma işlemi yapılarak kapama işlemi sağlanır.



**Şekil 22**. Kapama işlemi sonrası görüntü



**Kod 16.** Kapama Kodu

**5-) SIKIŞTIRMA İŞLEMLERİ**

Görüntü sıkıştırma veri sıkıştırmanın bir uygulaması olup, dijital bir görüntü üzerindeki depolama alanını ve veri transferini azaltmak için yapılır. Yani görüntünün kalitesini düşürmeden kapladığı yer ve görüntü üzerindeki bitlerin sayısını minimuma indirmektir. Böylelikle depolama cihazında kapladığı alan azaltılmış olur ve görüntünün web sayfalarında yüklenmesi veya karşı tarafa yollanması, upload ve download işlemlerinde daha hızlı işlenmesi için yapılmaktadır. Görüntüden belirli bölgeler çıkartılarak veya gruplandırılarak yapılmaktadır bu işlem. Görüntü sıkıştırmanın iki çeşidi vardır, kayıplı ve kayıpsız sıkıştırma.

**KAYIPLI**

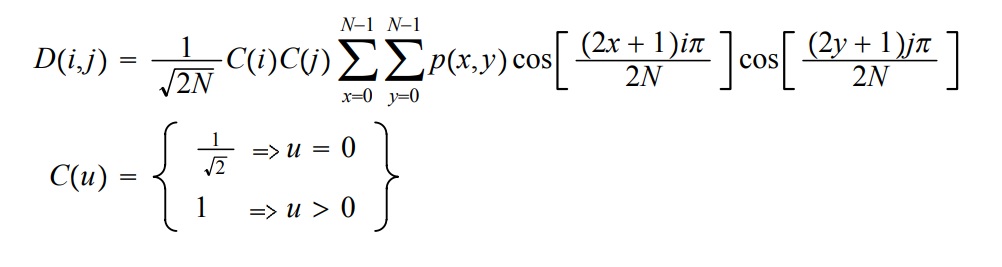
Çok sıklıkla kullanılan JPEG sıkıştırma yöntemi kayıplı sıkıştırma yöntemlerinden biridir. Bu yöntem DCT (Discrete Cosine Transform) adı verilen bir matematiksel dönüşümle yapılmaktadır. DCT matematiksel operasyonu Fast Fourier Transform (FFT) adı verilen bir sınıf içerisinde olan bir dönüşümdür. Bu operasyonların temel işlevi sinyali alıp bir gösterimden diğer gösterime dönüştürmektir. Örneğin iki boyutlu bir görüntüdeki sinyal insan görüntü sistemi tarafından algılanır. DCT bu sinyali yani uzaysal bilgiyi frekans yani numerik veriye çevirerek sıkıştırma için görüntüdeki bilgileri bir nicel forma dönüştürür.

DCT görüntüyü farklı frekanslara bölerek çalışır. Kuantizasyon denilen ara evrede sıkıştırma işlemi gerçekleşir. Daha az önem taşıyan frekanslar atılır bu yüzden kayıplı denir. Bundan sonra daha çok önem taşıyan frekanslar kalır ve görüntüyü tekrar açmak için kullanılır. Bu yüzden yeniden birleştirilmiş görüntü biraz bozukluğa sahiptir.

DCT işlemi şu şekilde gerçekleşmektedir :

1. Görüntü pikselleri 8x8 şeklinde matris haline bölünür.
2. Soldan sağa ve yukarıdan aşağıya çalışan DCT her bir matris bloğuna uygulanır.
3. Her blok kuantizasyon ile sıkıştırılır.
4. Sıkıştırılan blokların dizisi düşen alan kaplamasıyla kaydedilir.
5. Eğer istenirse, görüntü tekrar birleştirilir, yani sıkıştırmanın tersi uygulanır ve buna IDCT (Inverse DCT) denir.

DCT denklemi aşağıdaki gibidir :

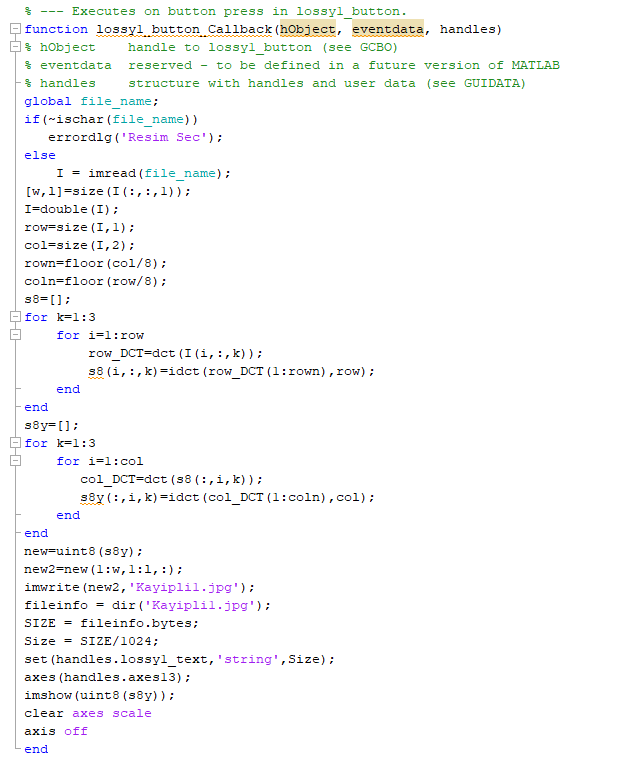


*P(x,y)* burada görüntünün x,y’inci elemanının *p* matrisindeki gösterimidir. *N*, DCT işleminin yapıldığı bloğun büyüklüğüdür. Denklem, orijinal görüntünün piksel değerlerinden dönüştürülmüş görüntünün tek bir (i,j’inci) girdisini hesaplar.

Aşağıda verilen şekil 23’te GUI üzerinde yapılan sıkıştırma işlemi verilmiştir. Kayıplı sıkıştırma olarak yapılan bu işlemde orijinal görüntü 17.48 KB iken, sıkıştırma sonrası 7.27 KB haline gelmiştir. Boyutta baya bir küçülme vardır fakat görüntünün kalitesinde düşüş rahatlıkla özlenebilmektedir.



**Şekil 23.** Kayıplı sıkıştırma işlemi sonrası görüntü



**Kod 17.** Kayıplı 1 Kodu

Aşağıda verilen şekil 24’te ise GUI üzerinde yapılan diğer kayıplı sıkıştırma yöntemi sonucu verilmiştir. Burada görüntü üzerindeki kayıp çok fazla olmaktadır. Fakat sıkıştırma işlemi sonrası görüntünün boyutunda çok fazla düşüş sağlanmaktadır. 17.49 KB olan görüntünün boyutu 0.5 KB’a kadar düşmektedir.



**Şekil 24.** Kayıplı Sıkıştırma 2 sonrası görüntü



**Kod 18.** Kayıplı 2 Kodu

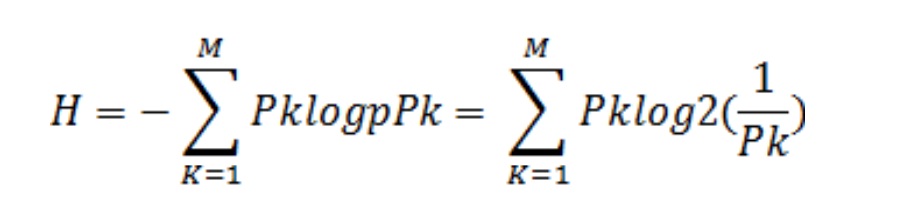
**KAYIPSIZ**

Kayıpsız sıkıştırma adı üzerinde sıkıştırma işlemi sonucu çok çok az görüntü kalitesinde kayıp yaparak çalışmaktadır. Sıkıştırma iki adımlı bir işlemdir, ilk adım static model oluşturur ve ikincisi ise bu modeli kullanmaktır. Giriş verilerinin haritalanması bit dizilimine yapılırken çok frekansa sahip bir veri daha az frekanstaki veriye göre daha kısa çıktı yapar.

Kayıpsız sıkıştırmanın avantajı olan görüntünün kalitesi bazı durumlarda çok önem taşıdığı için çok detaylı ve önemli işlerde kullanılır. Örneğin tıp alanında ve model tasarımında çok fazla yer almaktadır.

En çok kullanılan kayıpsız sıkıştırma yöntemlerinden biri olan Huffman Coding bir çeşit Entropy algoritmasıdır. Entropy bulunun verilerin bilgilerinin miktarını ölçer veya verilerdeki rastgeleliği bulur. Veri sonlu değer kümelerine kuantize olduktan sonra entropi kodlayıcı tarafından kodlanarak verilerin bulunma sıklığıyla daha fazla sıkıştırma yapar. Böylelikle statistik fazlalık azalır. Entropi kodlayıcı, gösterilecek olan bitlerin minimum sayısını verilen sembol kümesiyle gösterir.

Entropi, M giriş seviyeleri veya sembolleri (S1, S2,…SM) ve olasılıkları (P1, P2, …, PM) olduğunu varsayalım. O halde şu şekilde tanımlanabilir :



Huffman kodlamada en küçük olasılıklar birleştirilir veya eklenerek yeni olasılık kümesi oluşturur. Bu uzunluk kod tablosunu kullanarak olur. Uzunluk tablosu her olası kaynak sembolünün değerinin hesaplananan olasılığı üzerinedir. Huffman kodlamada her sembol özelleştirilmiş bir metot ile gösterilir ve en yaygın ama az cümleli karakterleri anlatır. Huffman kodlama basit binary blok kodlamayla aynıdır.

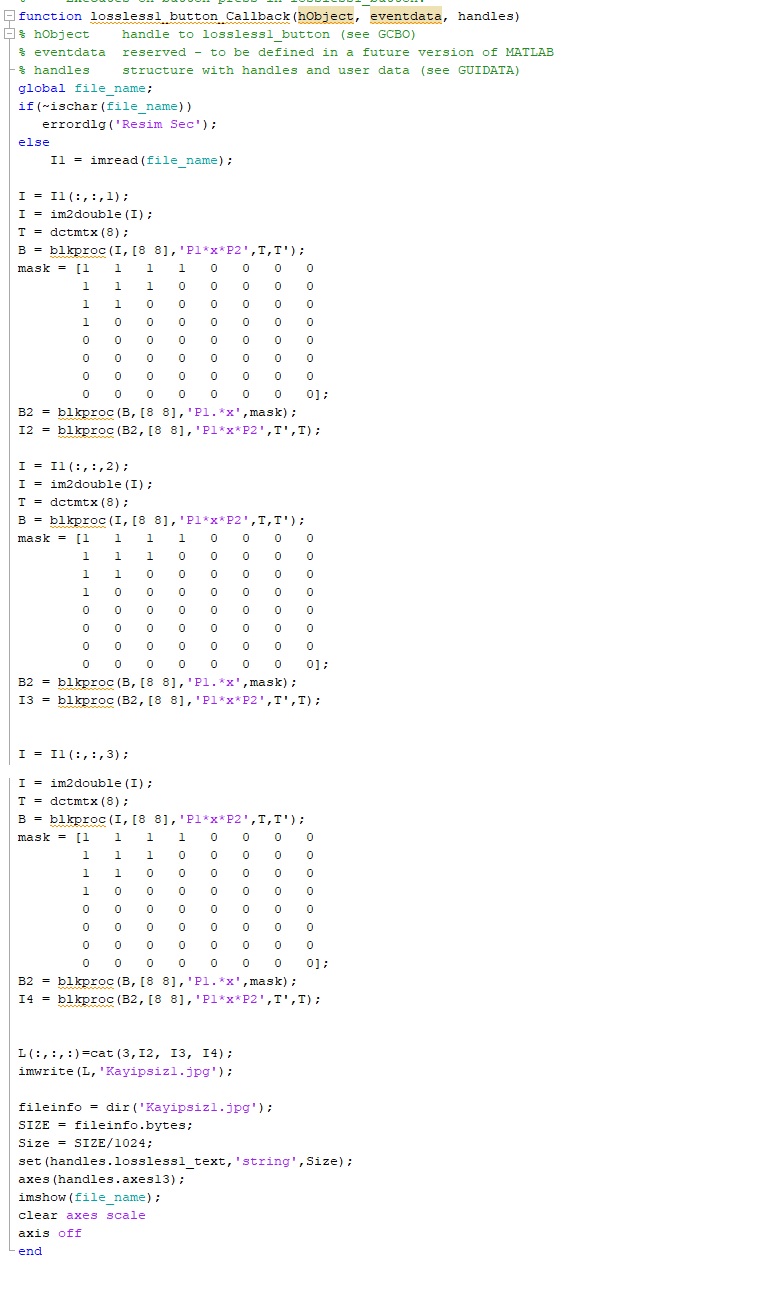
Huffman’in orijinal algoritması bilinin giriş olasılık dağılımının sembol-sembol ikilisi kodlamaya daha uygun olmasına rağmen, sembol-sembol ikilisi yasağı kaldırılınca veya ağırlık ortalama fonksiyonu bilinmiyor, birebir dağılmıyor veya bağımsız olduğunda ideal değildir.

Temel olarak tekniğin içeriği en son dizi olarak kaydedilecek bir binary ağaç yaratmaktır. Bu büyüklük olasılıkları verilen sembollerin sayılarıyla belirlenir. En düşük iki olasılık eklenir ve bir olasılık *0* olarak gösterilirken diğer olasılık ise *1* olarak gösterilir. Bu işlem toplam 1 olup bütün toplamalar bitene kadar tekrarlanır. En basit yapısal algoritma öncelik sırasını kullanır. Burada en düşük olasılıklı düğüme en yüksek öncelik tanımlanır.

Aşağıda verilen şekil 25’te GUI üzerinde kayıpsız sıkıştırma işlemi gerçekleştirilmiştir. Görüntü üzerinde DCT algoritması kullanılarak sıkıştırma yapılmıştır. Önceden anlatılan DCT algoritmasının JPEG sıkıştırmasında kayıplı olduğu söylenmiş olsa da daha sonra geliştirilen matris ve algoritmalarla DCT ile kayıpsız sıkıştırma yapılması mümkün olmuştur. Görüntü öncelikle double formatına çevirilip DCT matrisi ile maskelemeden geçer ve kuantizasyona uğrar. Böylelikle sıkıştırma gerçekleşir. Görüntünün orijinal boyutu 14.06 KB iken, sıkıştırma sonrası 7.69 KB’a düşmüş fakat görüntü kalitesinde çok minimum derecede azalma vardır.



**Şekil 25.** Kayıpsız sıkıştırma sonrası görüntü

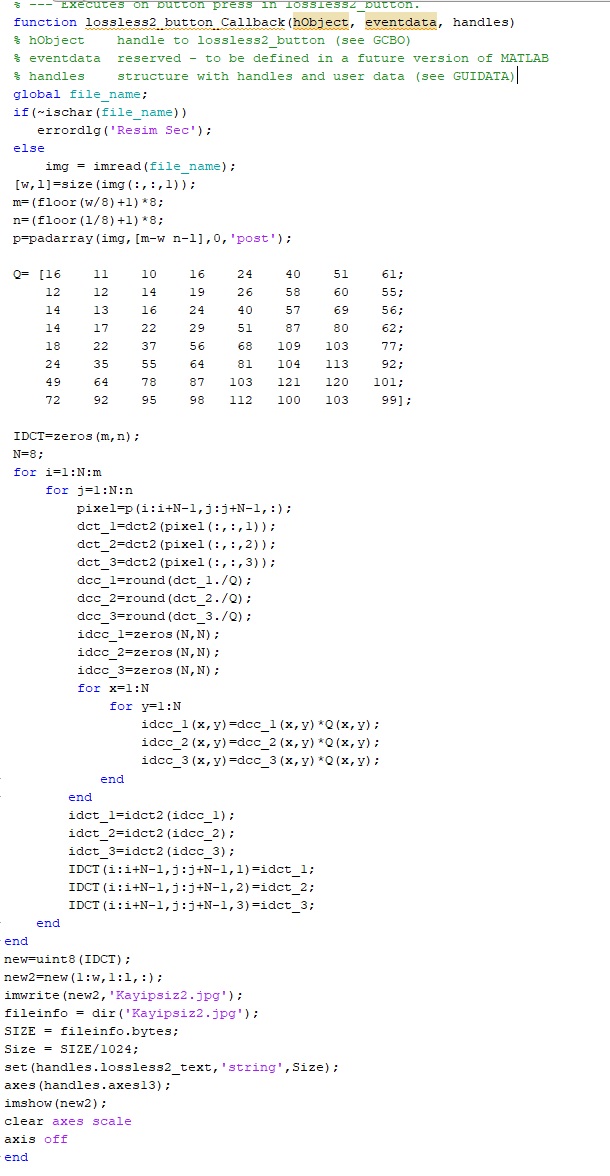


**Kod 19.** Kayıpsız 1 Kodu

Aşağıda verilen şekil 26’da ise GUI üzerinde yapılan diğer kayıpsız sıkıştırma işlemi vardır. Bu sefer görüntü üzerinde entropy coding uygulanarak yapılan Huffman Coding işlemi vardır. Entropi ile görüntüdeki piksellerin tekrarlanma olasılıkları belirli bir şekilde tablo haline getirilir ve Huffman kodlamasıyla bu olasılıklar birleştirilir. Görüntünün orijinal boyutu 14.06 KB iken, sıkıştırma sonrası 11.13 KB’a düşmüş fakat görüntü kalitesinde çok minimum derecede azalma vardır.



**Şekil 26.** Diğer kayıpsız sıkıştırma işlemi sonrası görüntü



**Kod 20.** Kayıpsız 2 Kodu