

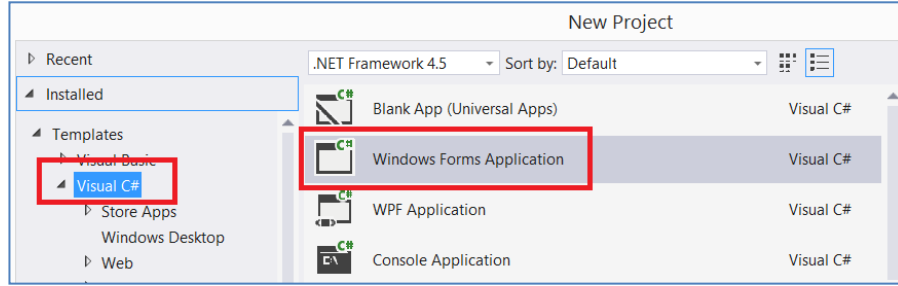
GÖRÜNTÜ İŞLEME - (2.Hafta)

C# PROGRAMLAMA İLE GÖRÜNTÜ İŞLEME

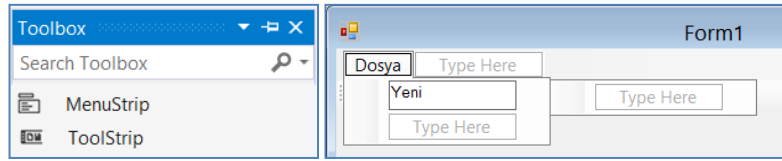
Görüntü işleme kapsamında geliştirilecek algoritmalar C# diliyle yazılacaktır. Bu amaçla bilgisayarımızda Visual Studio programının kurulu olması gerekmektedir. Öncelikle programın altyapısının oluşturulup, dersler ilerledikçe içerisine kodlar eklenecektir.

Program Altyapısının Oluşturulması

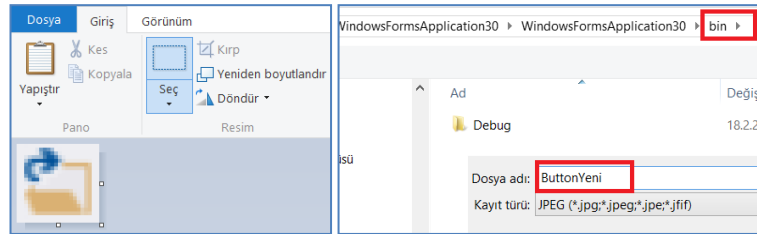
Geliştirilecek program Windows Form Application uygulaması olacaktır.



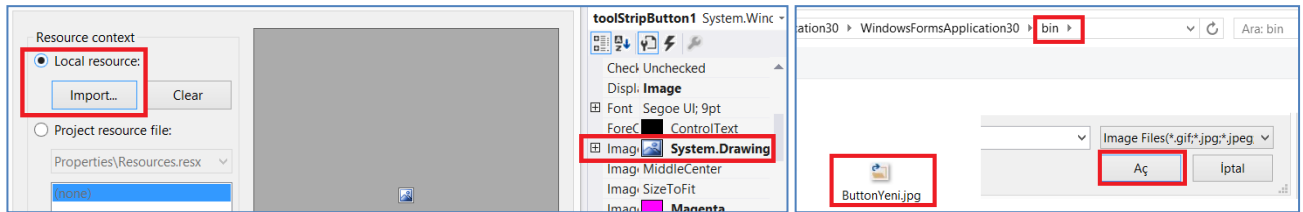
Program içerisinde Menü ve Araç çubuğu kullanılacaktır. Bu amaçla MenuStrip ve ToolStrip nesneleri kullanılacaktır.



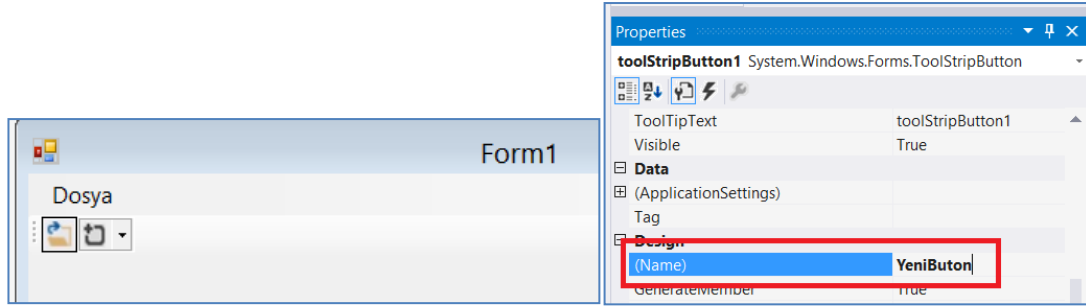
ToolStrip (araç çubuğu) üzerindeki butonların resimlerini atarken, Paintte oluşturulan örnek resimler projenin içerisindeki Bin klasörünün altına kopyalayalım.



Daha sonra ToolStrip üzerindeki butonu seçip Image özelliğinden butonu yükleyelim.



Butonun resmi yüklendikten sonra Properties penceresinden başka gerekli ayarlamalar yapılabilir. Örneğin butonun adı yapacağı işle ilgili olarak değiştirilebilir.



Butona tıkladığımızda ona ait olan fonksiyon kodları açılacaktır.

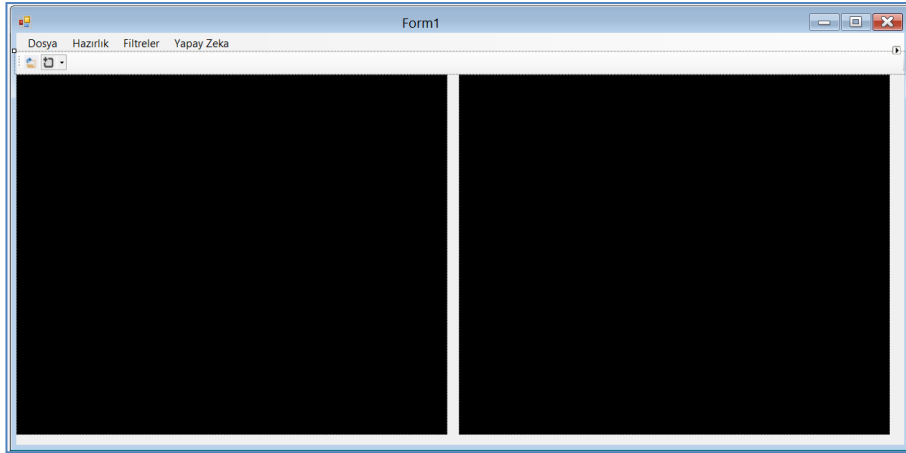
```

public partial class Form1 : Form
{
    1 reference
    public Form1()
    {
        InitializeComponent();
    }

    1 reference
    private void YeniButon_Click(object sender, EventArgs e)
    {
    }
}

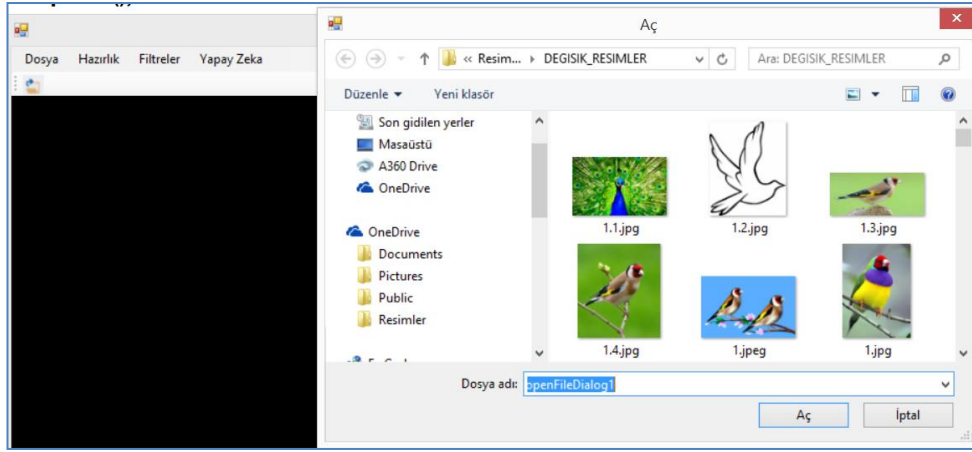
```

Programımızın tasarımını üstte bir menu çubuğu, altında ise Araç çubuğu olacak şekilde tasarlayalım. Form üzerinde ise iki tane PictureBox olsun. Bunlardan birincisi orjinal resmi, diğeri ise dönüştürülmüş resmi gösterebilir. Program ilerledikçe ekleme ve çıkarmalar olacaktır. Bu altyapı üzerinde devam edelim.



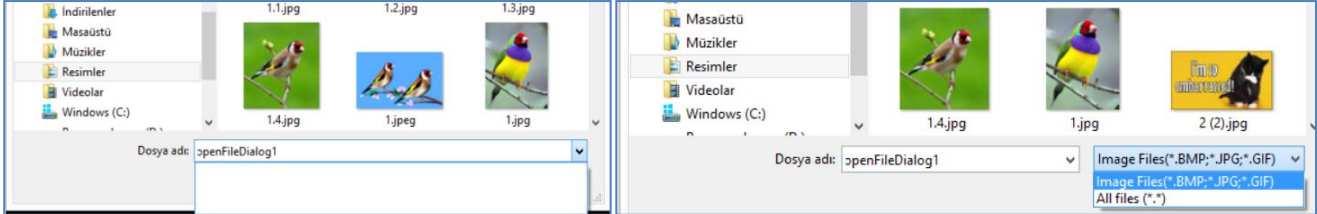
RESİM YÜKLEME

Bilgisayarımızdaki herhangi bir resmi PictureBox1 yüklemek için aşağıdaki kodları kullanabiliriz. Bunu fonksiyon şeklinde yazarsak hem menüden hem de araç çubuğunda aynı kodlar kullanılarak resim yüklenebilir. Burada geliştirilen programda iki tane pictureBox kullanılmaktadır. Birincisinde orjinal resim (giriş resmi), ikincisinde ise dönüştürülmüş resim (çıkış resmi) görüntülenecektir.



Dosya açarken uzantıların görüntülenmesi süzmek için uygun olacaktır. Bunun için aşağıdaki kodları kullanabiliriz.

```
openFileDialog1.DefaultExt = ".jpg";
openFileDialog1.Filter = "Image Files(*.BMP;*.JPG;*.GIF)|*.BMP;*.JPG;*.GIF|All files (*.*)|*.*";
```



Eğer resim yükleme çalışırken openFileDialog açıldıktan resim seçilmeyip "iptal" tuşuna basılırsa, resim yolu değişkeni "null" olacağı için program hata verir. Bunu engellemek için Try-Catch kodları içine yazılması gerekir.

```
private void DosyaAc_menu_Click(object sender, EventArgs e)
{
    DosyaAc();
}
private void DosyaAc_toolbar_Click(object sender, EventArgs e)
{
    DosyaAc();
}
//DOSYA AÇ -----
public void DosyaAc()
{
    try
    {
        openFileDialog1.DefaultExt = ".jpg";
        openFileDialog1.Filter = "Image Files(*.BMP;*.JPG;*.GIF)|*.BMP;*.JPG;*.GIF|All files (*.*)|*.*";

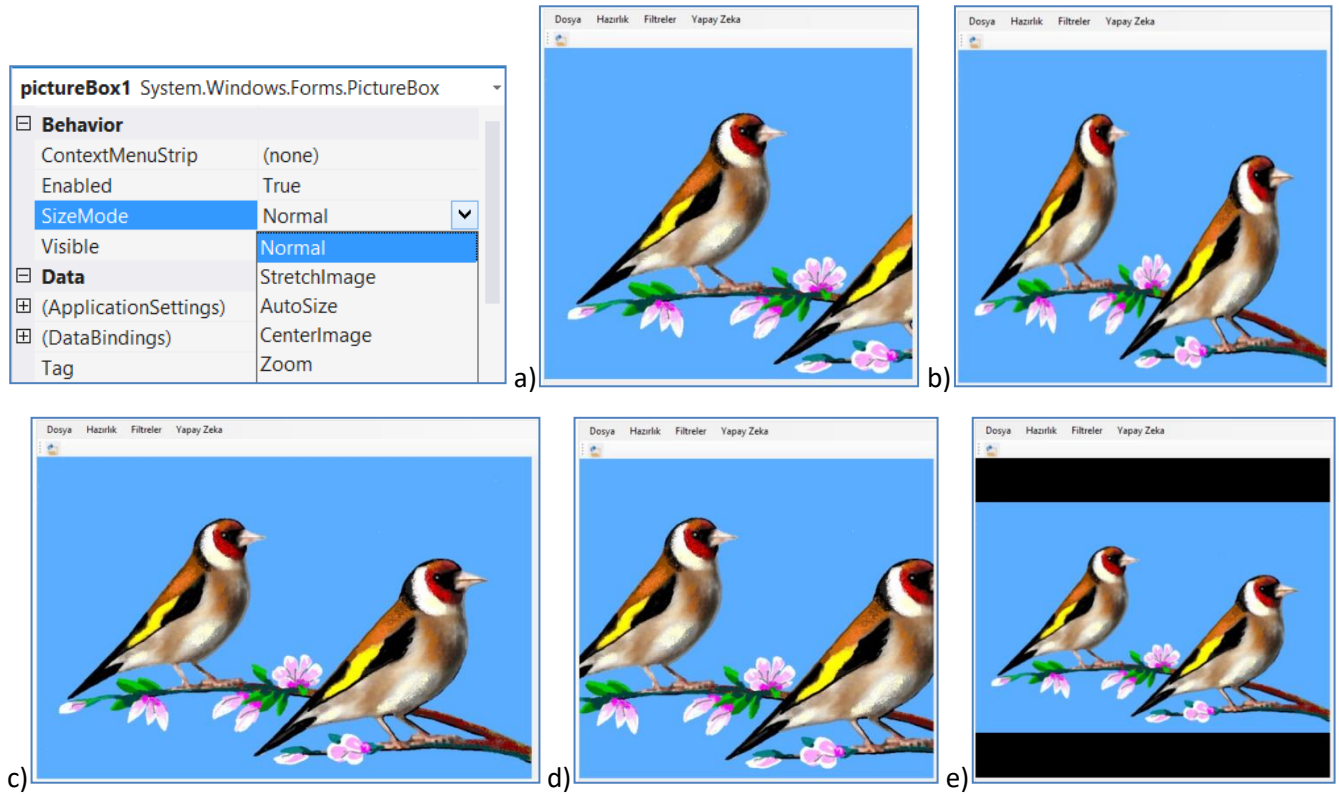
        openFileDialog1.ShowDialog();

        String ResminYolu = openFileDialog1.FileName;
        pictureBox1.Image = Image.FromFile(ResminYolu);
    }
    catch {}
}
```

Görüntülenen resim çeşitli boyutlarda olabilir. Bu nedenle resmin pictureBox içerisinde düzgün görüntülenmesi için SizeMode seçeneğini aşağıdaki seçeneklerden uygun olanla yapmak gerekir. Burada;

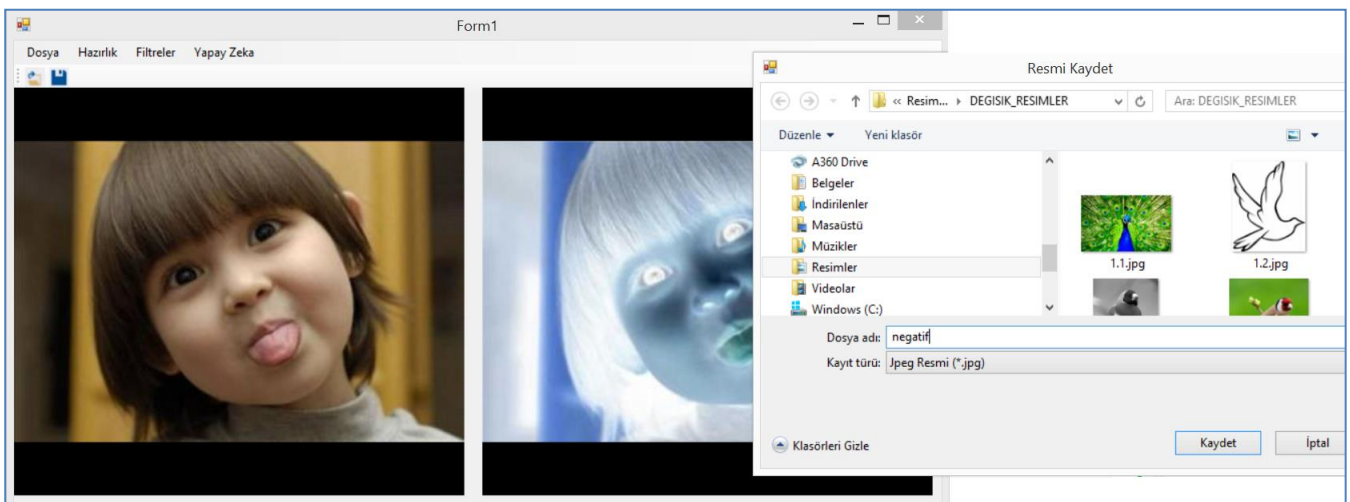
- a) **Normal** (resmin sığıdığı kadar kısmı normal görüntülenir)
- b) **StretchImage** (resim çerçevenin içine tam sığacak şekilde daraltılır yada genişletilir,
- c) **AutoSize** (çerçeve resmin boyutlarını alır)

- d) **CenterImage** (resmin orta bölgesi çerçeve içinde normal şekilde görüntülenir, sığmayan kısımlar dışarıda kalır,
- e) **Zoom** (resmin orantısı bozulmadan tamamı çerçeve içinde görüntülenir).



RESMİ KAYDETME

Birinci picturebox da görüntülenen orjinal resim, belli filtrelerden geçirildikten sonra ikinci picturebox da görüntülenecektir. Görüntülenen bu ikinci resim gerektiğinde bilgisayara kaydedilebilmelidir. Bu amaçla aşağıdaki kodları kullanabiliriz. Burada üç tip resim kaydetme yapılmaktadır. Bunlar jpeg, bitmap ve gif resimleridir.



//RESMİ KAYDETME-----

```
public void ResmiKaydet()
```

```
{
```

```
    SaveFileDialog saveFileDialog1 = new SaveFileDialog();
```

```
    saveFileDialog1.Filter = "Jpeg Resmi*.jpg|Bitmap Resmi*.bmp|Gif Resmi*.gif";
```

```
    saveFileDialog1.Title = "Resmi Kaydet";
```

```

saveFileDialog1.ShowDialog();

if (saveFileDialog1.FileName != "") //Dosya adı boş değilse kaydedecek.
{
    // FileStream nesnesi ile kayıtlı gerçekleştirilecek.
    FileStream DosyaAkisi = (FileStream)saveFileDialog1.OpenFile();

    switch (saveFileDialog1.FilterIndex)
    {
        case 1:
            pictureBox2.Image.Save(DosyaAkisi, System.Drawing.Imaging.ImageFormat.Jpeg);
            break;

        case 2:
            pictureBox2.Image.Save(DosyaAkisi, System.Drawing.Imaging.ImageFormat.Bmp);
            break;

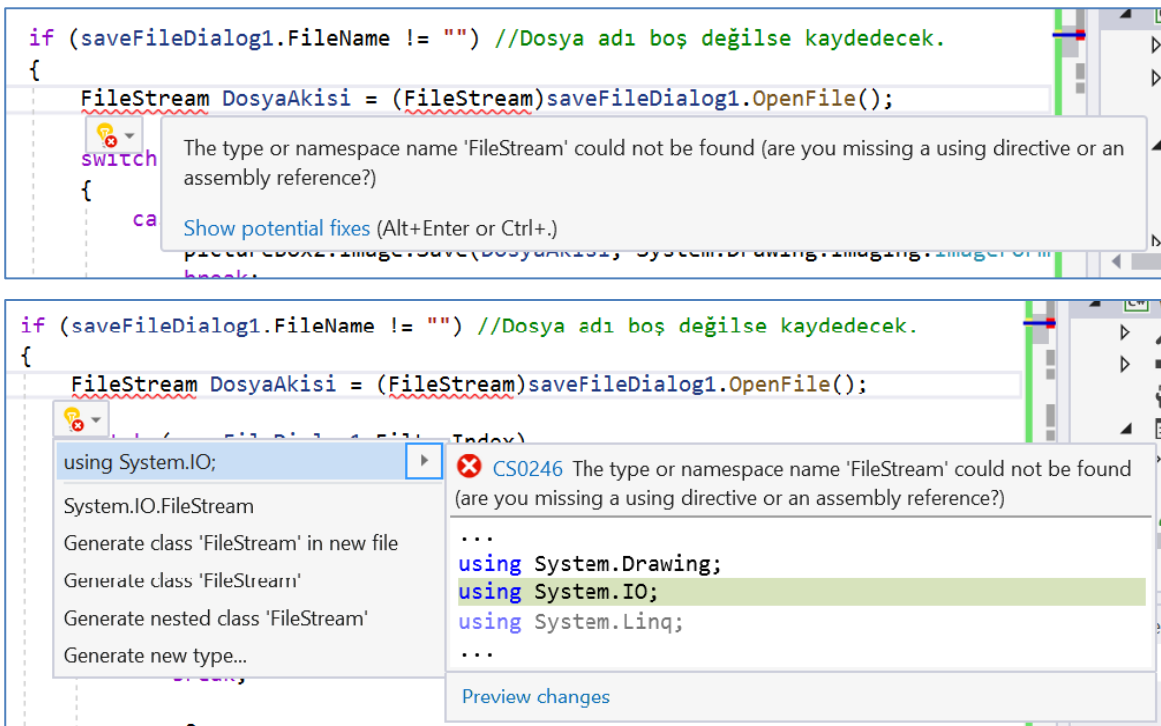
        case 3:
            pictureBox2.Image.Save(DosyaAkisi, System.Drawing.Imaging.ImageFormat.Gif);
            break;
    }

    DosyaAkisi.Close();
}
}

```

Not:

2019 Versiyonunda bir komutun kütüphanesini (reference) eklemek için aşağıdaki gibi mouse ile üzerine gelince potansiyel seçenekleri gösterecektir. Bunun için “Show potential fixes” tıklayınız. Daha sonra çıkan listede using.System..IO; kütüphanesini seçin. Kendisi otomatik olarak programın en üstüne ekleyecektir.



RESMİ PİKSEL OLARAK AKTARMA



```
private void btnResmiPikselOlarak_Click(object sender, EventArgs e)
{
    Color OkunanRenk, DonusenRenk;
    int R = 0, G = 0, B = 0;

    Bitmap GirisResmi, CikisResmi;
    GirisResmi = new Bitmap(pictureBox1.Image);

    int ResimGenisligi = GirisResmi.Width; //GirisResmi global tanımlandı. İçerisine görüntü
    yüklendi.
    int ResimYuksekligi = GirisResmi.Height;
    CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi); //Cikis resmi oluşturuyor.
    Boyutları giriş resmi ile aynı olur. Tanımlaması globalde yapıldı.

    int i = 0, j = 0; //Çıkış resminin x ve y si olacak.
    for (int x = 0; x < ResimGenisligi; x++)
    {
        for (int y = 0; y < ResimYuksekligi; y++)
        {
            OkunanRenk = GirisResmi.GetPixel(x, y);

            R = OkunanRenk.R;
            G = OkunanRenk.G;
            B = OkunanRenk.B;

            DonusenRenk = Color.FromArgb(R, G, B);
            CikisResmi.SetPixel(x, y, DonusenRenk);
        }
    }

    pictureBox2.Image = CikisResmi;
}
```

RESMİN NEGATİFİNİ ALMA (Negative)

Bu işlem basit olarak şu şekilde yapılır. Her piksel renk değeri bir döngü içinde 255 sayısından çıkarıldığında geriye kalan değer, negatif rengi verecektir. Renkli resimlerde ise 3 renk de aynı işleme tabi tutulur.

Örneğin; siyah renkli bir pikselin değeri 0 dır. Bu pikselin değeri $255-0=255$ olarak ayarlanırsa bu renk de beyaz olacaktır. Yada beyaz yakın bir değer olan 212 değerini dönüştürürsek, $255-212=43$ olacaktır. Bu da siyaha yakın bir renktir. Bunu matematiksel olarak ifade edersek, $g(x;y)$ giriş resminin Kırmızı piksel değeri, $f[x;y]$ çıkış resminin kırmızı piksel değeri olacaktır.

$$f(x; y) = 255 - g(x; y)$$

Programlama



```
//NEGATİFİNİ ALMA -----
private void btnNegatif_Click(object sender, EventArgs e)
{
    Color OkunanRenk, DonusenRenk;
    int R = 0, G = 0, B = 0;

    Bitmap GirisResmi, CikisResmi;
    GirisResmi = new Bitmap(pictureBox1.Image);

    int ResimGenisligi = GirisResmi.Width; //GirisResmi global tanımlandı. İçerisine görüntü
    yüklendi.
    int ResimYuksekligi = GirisResmi.Height;
    CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi); //Cikis resmi oluşturuyor.
    Boyutları giriş resmi ile aynı olur. Tanımlaması globalde yapıldı.

    int i = 0, j = 0; //Çıkış resminin x ve y si olacak.
    for (int x = 0; x < ResimGenisligi; x++)
    {
        for (int y = 0; y < ResimYuksekligi; y++)
        {
            OkunanRenk = GirisResmi.GetPixel(x, y);

            R = 255 - OkunanRenk.R;
            G = 255 - OkunanRenk.G;
            B = 255 - OkunanRenk.B;

            DonusenRenk = Color.FromArgb(R, G, B);
            CikisResmi.SetPixel(x, y, DonusenRenk);
        }
    }

    pictureBox2.Image = CikisResmi;
}
```

GRİ TON RESİM (Siyah-Beyaz) (Grayscale)

Griton resimler renkler olmadan sadece ışığın yoğunluğunu gösteren resimlerdir. Eski teknoloji siyah beyaz resimlere benzer.

Bu resimler bilgisayarda 8 bit formatında saklanır. Bunun anlamı her piksel 8 bit ikili kod ile saklanır. Yani $2^8=256$ olarak gösterilirse 0-255 arasında değer alır. 0 siyah rengi gösterirken, 255 beyaz rengi gösterecektir.

Bir griton resim 800x600 piksel boyutlarında olursa içerisinde 480.000 piksel olacaktır. Her piksel 8 bit=1 byte hafızada yer alacağına göre resmin tamamı 480 kByte=0,48 MByte yer kaplayacaktır. Aynı resim renkli olsaydı,

her renk (RGB-RedGreenBlue) benzer şekilde 256 ton renk olarak $2^8 \times 3 = 2^{24}$ yani 24 bit yer kaplar. Bu durumda aynı resim bu sefer 3 katı yer kaplar.

Griton resimleri, kenar tespiti yada eşik uygulamaları gibir farklı amaçlar için ileriki uygulamamızda kullanacağız.

Renkli resmin Gri-Ton a dönüştürülmesi

Renkli sayısal bir görüntüyü gri-ton bir görüntüye dönüştürme işlemi aslında RGB renk modelinde belirtilen her bir renk bandına karşı düşen gri-ton görüntülerin ölçeklendirilmesinden başka bir şey değildir. Normalde üç rengin değerini toplayıp üçe bölerek gri tonu elde edebiliriz. Fakat bu gözümüzün farklı renkleri farklı algılama hassasiyetini tam yansıtmaz. Bu nedenle aşağıdaki formüllerde verildiği gibi ölçekleme yapılmalıdır. İlk verilen formül en gerçekçi olanıdır.

$$\text{GriDegeri} = (R + G + B)/3$$

$$\text{GriDegeri} = 0.299 \times R + 0.587 \times G + 0.114 \times B$$

$$\text{GriDegeri} = 0.21 \times R + 0.71 \times G + 0.071 \times B$$

Gri renk değeri bulunduktan sonra, R,G,B renk değerlerinin hepsi aynı renk koduna sahip olmalıdır. Yani;

$$R=\text{GriDegeri}, G=\text{GriDegeri}, B=\text{GriDegeri}$$

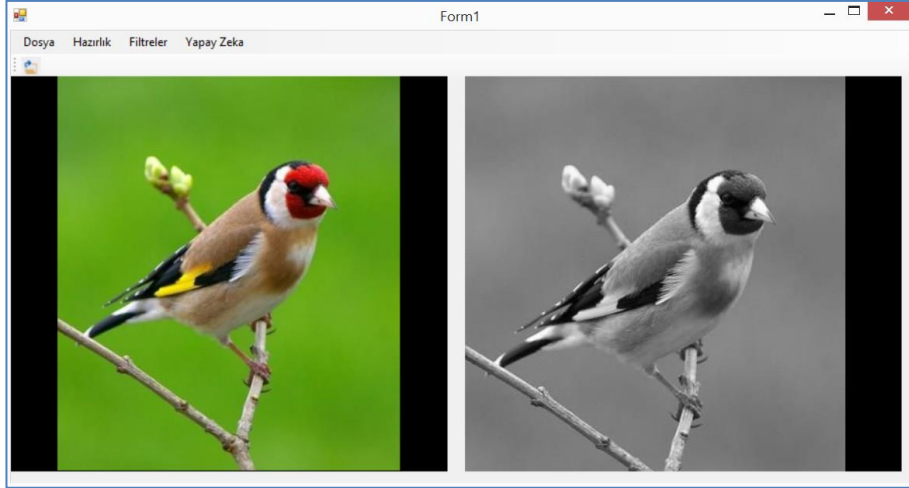
Tabii bu şekilde resim yine 24 bit olarak saklanmış olur (hafızada renkli resim gibi yer tutar). Bunu 8 bit olarak saklamak gerekir.

Araştırma: 24 bit resim 8 bit olarak nasıl saklanır. Yani renkler üç tane formatta değilde tek renk formatında nasıl saklanır. Konuyu araştırınız. Her iki formatla saklanan gri-ton resmin hafızda kapladığı alanın farklı olduğunu gösteriniz.



Şekil. Doğal renkli görüntüden Gri ton görüntünün elde edilmesi.

Programlama



```
//GRI TONLAMA -----
public Bitmap ResmiGriTonaDonustur()
{
    Color OkunanRenk, DonusenRenk;
    int R = 0, G = 0, B = 0;

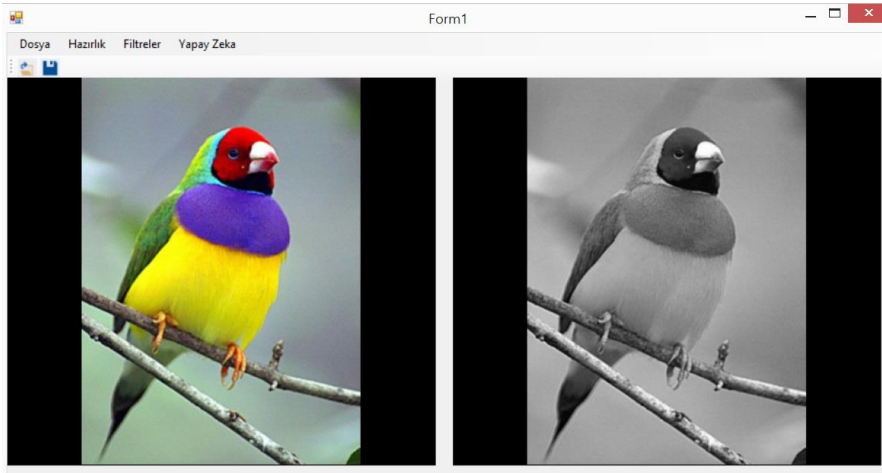
    int ResimGenisligi = GirisResmi.Width; //GirisResmi global tanımlandı. Fonksiyonla gelmedi.
    int ResimYuksekligi = GirisResmi.Height;
    CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi); //Cikis resmini oluşturuyor. Boyutları giriş resmi
    ile aynı olur.

    int i = 0, j = 0; //Çıkış resminin x ve y si olacak.
    for (int x = 0; x < ResimGenisligi; x++)
    {
        j = 0;
        for (int y = 0; y < ResimYuksekligi; y++)
        {
            OkunanRenk = GirisResmi.GetPixel(x, y);
            int GriDegeri = Convert.ToInt16(OkunanRenk.R*0.299 + OkunanRenk.G*0.587 + OkunanRenk.B*0.114);
            //Gri-ton formülü
            R = GriDegeri;
            G = GriDegeri;
            B = GriDegeri;
            DonusenRenk = Color.FromArgb(R, G, B);
            CikisResmi.SetPixel(i, j, DonusenRenk);
            j++;
        }
        i++;
    }

    return CikisResmi;
}
```

Gri-ton formülü olarak 3 kanal (R,G,B) renk ortalamasını alarak da gri-ton a dönüştürebiliriz. Ama bu formül daha az gerçekçi bir sonuç verecektir. Bunun için yukarıdaki programda ilgili satırı aşağıdaki kodlarla değiştirmeliyiz.

```
int GriDegeri = (int)(OkunanRenk.R + OkunanRenk.G + OkunanRenk.B)/3; //Ortalama Gri-ton formülü
```



Ortalama Formülü ile oluşturuldu



Gri-ton ölçekleme formülü ile oluşturuldu

Not:

Resmin üzerindeki noktaların renk değerlerini öğrenmek için aşağıdaki kodları kullanabilirsiniz. Böylece hangi rengin dönüşümlerde nasıl değiştiğini daha iyi yorumlarsınız.

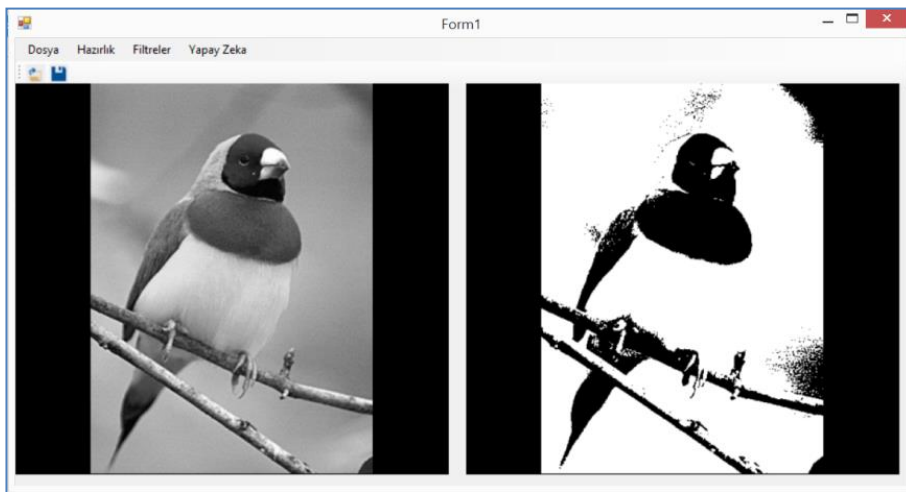
```
private void pictureBox1_MouseClick(object sender, MouseEventArgs e)
{
    Color Renk = ((Bitmap)pictureBox1.Image).GetPixel(e.X, e.Y);
    txtRGB.Text = string.Format("R:{0} G:{1} B:{2}", Renk.R, Renk.G, Renk.B);
}
```

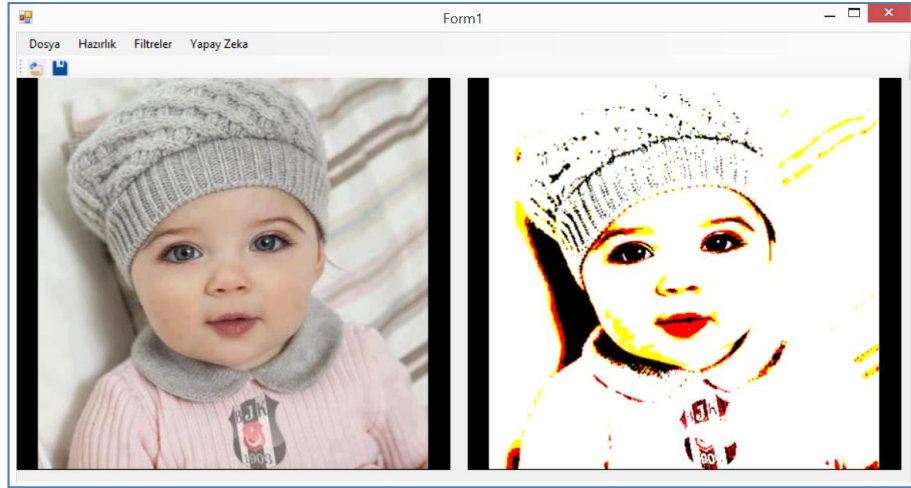
EŞİKLEME (Thresholding)

Bu teknikte resim içerisindeki her pikselin renk değeri belli bir eşik değeri ile kıyaslanır. Eğer bu eşik değerinin altında ise bir renk, üzerinde ise başka bir renk atanır. Örneğin gri-ton bir resmi bu filtreden geçirirken, renk değeri 128 gibi bir eşik değerinin altındaysa 0-siyah olarak, üzerinde ise 255-beyaz olarak atanabilir. Matematiksel olarak aşağıdaki gibi gösterilebilir.

$$f(x,y) = \begin{cases} 0 & \text{eğer } g(x,y) < T = 128 \\ 255 & \text{diğer} \end{cases}$$

Programlama





```
//EŞİKLEME -----
public Bitmap ResmiEsiklemeYap()
{
    int R = 0, G = 0, B = 0;
    Color OkunanRenk, DonusenRenk;
    Bitmap GirisResmi, CikisResmi;
    GirisResmi = new Bitmap(pictureBox1.Image);

    int ResimGenisligi = GirisResmi.Width; //GirisResmi global tanımlandı.
    int ResimYuksekligi = GirisResmi.Height;
    CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi); //Cikis resmini oluşturuyor. Boyutları
    giriş resmi ile aynı olur.

    int EsiklemeDegeri = Convert.ToInt32(textBox1.Text);

    for (int x = 0; x < ResimGenisligi; x++)
    {
        for (int y = 0; y < ResimYuksekligi; y++)
        {
            OkunanRenk = GirisResmi.GetPixel(x, y);

            if (OkunanRenk.R >= EsiklemeDegeri)
                R = 255;
            else
                R = 0;

            if (OkunanRenk.G >= EsiklemeDegeri)
                G = 255;
            else
                G = 0;

            if (OkunanRenk.B >= EsiklemeDegeri)
                B = 255;
            else
                B = 0;

            DonusenRenk = Color.FromArgb(R, G, B);
            CikisResmi.SetPixel(x, y, DonusenRenk);
        }
    }

    pictureBox2.Image = CikisResmi;
}
```

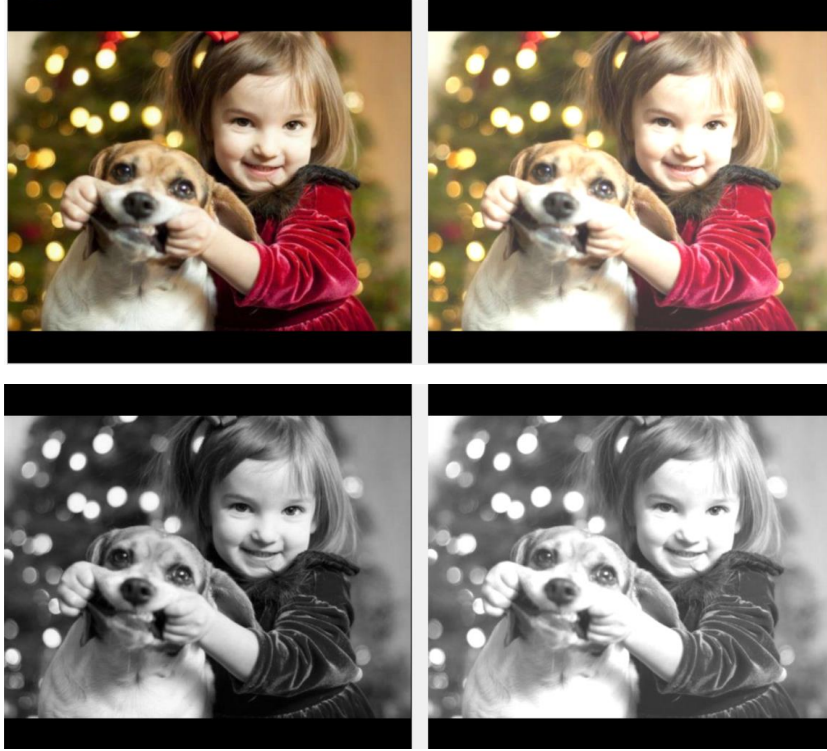
PARLAKLIK (BRIGHTNESS)

Bir resmin parlaklığını artırma yada rengini açma, beyaz tona doğru ilerlemek için renk kanalları üzerine aynı büyüklükte eşdeğer bir sayı eklemekle gerçekleştirilir. Resmi koyulaştırmak için tam tersi sayı çıkarılır. Burada dikkat edilmesi gereken üzerine eklenen sayılarla sonuç 255 değerini geçmemelidir yada çıkarılan sayılar nedeniyle 0 altına düşmemelidir. Şu şekilde formülize edebiliriz.

$$f(x,y) = g(x,y) (R \mp b, G \mp b, B \mp b)$$

Program Kodları

Birinci resimler orjinal resim, ikinciler rengi 50 değerinde açılmış resimdir.



```
//PARLAKLIĞINI DEĞİŞTİRME -----
public Bitmap ResminParlakliginiDegistir()
{
    int R = 0, G = 0, B = 0;
    Color OkunanRenk, DonusenRenk;
    Bitmap GirisResmi, CikisResmi;
    GirisResmi = new Bitmap(pictureBox1.Image);

    int ResimGenisligi = GirisResmi.Width; //GirisResmi global tanımlandı.
    int ResimYuksekligi = GirisResmi.Height;
    CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi); //Cikis resmini oluşturuyor. Boyutları giriş resmi ile aynı olur.

    int i = 0, j = 0; //Çıkış resminin x ve y si olacak.
    for (int x = 0; x < ResimGenisligi; x++)
    {
        j = 0;
        for (int y = 0; y < ResimYuksekligi; y++)
        {
            OkunanRenk = GirisResmi.GetPixel(x, y);
            //Rengini 50 değeri ile açacak.
            R = OkunanRenk.R + 50;
            G = OkunanRenk.G + 50;
```

```

    B = OkunanRenk.B + 50;

    //Renkler 255 geçtiyse son sınır olan 255 alınacak.
    if (R > 255) R = 255;
    if (G > 255) G = 255;
    if (B > 255) B = 255;

    DonusenRenk = Color.FromArgb(R, G, B);
    CikisResmi.SetPixel(i, j, DonusenRenk);
    j++;
}
i++;
}
return CikisResmi;
}

```

KARŞITLIK (CONTRAST)

Sözlük anlamı “karşıtlık” olan kontrast, görüntünün ayırt edilebilirliğini ifade etmek için kullanılır. İnsan gözünün gri değerlerini ayırt etme kapasitesi ortalama olarak otuz adet gri değeriyle sınırlıdır. Bir görüntüdeki en düşük ve en yüksek gri değerleri arasındaki farkın bu sınırdan düşük olması durumunda, bu görüntüdeki detayların insan gözü tarafından yeterince algılanamadığı söylenebilir. Bu tipteki görüntülerin histogramlarında, gri düzeyli bir görüntünün dar bir aralığında çok sayıda pikselin yoğunlaştığı görülür. Bu görüntülere düşük kontrastlı görüntüler adı verilir. Kontrast kavramı bir görüntü içindeki grilik seviyesi farklarının genliğini ifade etmektedir. İnsan incelemesine uygun olmayan düşük kontrastlı görüntüler çeşitli dönüşüm fonksiyonları yoluyla insan için daha uygun duruma getirilebilirler. Farklı türdeki düşük kontrastlı görüntüler için farklı fonksiyonlar kullanmak gerekir. Koyu tonların ağırlıklı olduğu bir görüntüyü geliştirmek için gri değerlerini açık tonları gösteren yüksek değerlere doğru kaydırmak gerekir

$$g(x, y) = F[f(x, y)]$$

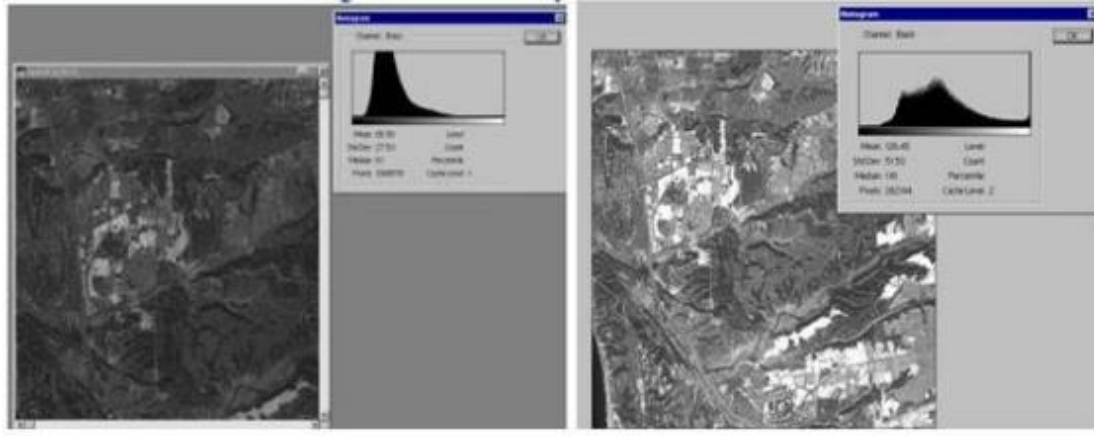
$$= af(x, y) \quad s = ar \quad a > 1 \text{ ise karşıtlık artar.}$$

$$a < 1 \text{ ise karşıtlık azalır.}$$



Şekil. Sabit sayıya göre kontrast ayarı.

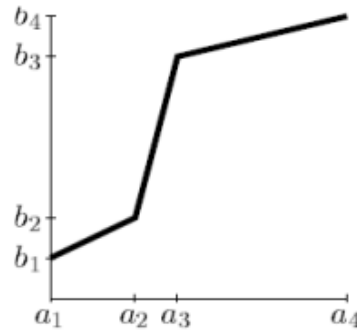
Kontrast germe; kötü kontrastlı görüntülerin (histogramı dar bir alana yayılmış görüntüler), kontrastlarını iyileştirmek için yapılan genel bir yöntemdir. Amacı, histogramı geniş bir bölgeye yaymaktır. Gri seviye değerlerini, parça parça lineer bir fonksiyon uygulayarak, orijinal aralığın dışına yayabiliriz. Şekilde gösterilen parça parça lineer germe fonksiyonu a_i , b_i ve a_{i+1} ve b_{i+1} koordinatları arasında kalan piksel değerlerini bulmak için kullanılır [Perihanoğlu G. M., “Dijital Görüntü İşleme Teknikleri Kullanılarak Görüntülerden Detay Çıkarma” Yüksek Lisans Tezi, Ocak 2015.].



(a)

(b)

Şekil. a) kontrast germe işleminden önceki görüntü b) kontrast germe işleminden sonraki görüntü.



Şekil. Lineer germe fonksiyonu

$$y = \frac{b_{i+1} - b_i}{a_{i+1} - a_i} (x - a_i) + b_i$$

Doğrusal Kontras Artırma

Gri seviye görüntüde 0-255 arasında olan parlaklık değerleri bazı görüntülerde dar bir aralıktadır. Örneğin bir görüntünün minimum parlaklık değeri 50 maksimumu 100 olabilir. Böyle bir durumda görüntüde bulunan ayrıntıların birbirinden ayırt edilmesi zordur. Kontras artırma ile görüntünün parlaklık değerleri 0-255 arasına çekilerek görüntü noktasal olarak zenginleştirilmiş olur. Kontrast iyileştirme, görüntülenecek nesne histogramının ne kadar genişlediği ile ilgilidir. Yani görüntülenmek üzere minimum ve maksimum değerlerle belirtilen aralığın boyutlarıdır. Bu aralık daha iyi görüntülenirken, bu aralığın alt ve üst sınırında kalan değerler görüntülenmeyecektir. Kontrast iyileştirme için doğrusal ve doğrusal olmayan metotlar kullanılır. Doğrusal kontrast iyileştirme metodunda orijinal görüntü değerleri ile görüntülenen değer arasında doğrusal bir ilişki kurulur. Doğrusal kontrast iyileştirme en iyi sonucu görüntü histogramının Gaus dağılım gösterdiği durumlarda verir. İdeal bir gaus histogram, değerlerin tek modlu bir aralıkta bulunduğu ve modun her iki yanında simetrik biçimde dağılmış olduğu histogramdır. Görüntüde birbirine yakın yansıma değerleri veren metaryeller bulunmakta ise histogram gaus yada gausa benzer olacaktır. Doğrusal kontrast iyileştirme metodunda görüntüleneyecek değerler aşağıdaki Eşitlikçe göre belirlenir.

$$g(x, y) = \left(\frac{f(x, y) - f(x, y)_{\min}}{f(x, y)_{\max} - f(x, y)_{\min}} \right) * 255$$

$g(x, y)$: görüntüleneyecek piksel değeri
 $f(x, y)$: orijinal görüntü piksel değeri
 $f(x, y)_{\min}$: orijinal görüntü en küçük piksel değeri
 $f(x, y)_{\max}$: orijinal görüntü en büyük piksel değeri

Üstte verilen formül genel kontrast genişletme için kullanılır ve hassas değildir. Daha hassas kontrast genişletme işlemi için aşağıda verilen formül kullanılır

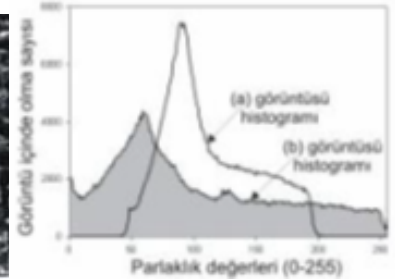
$$g(x, y) = \begin{cases} 0 & f(x, y) \leq \%f(x, y)_{\min} \\ 255 \left(\frac{f(x, y) - \%f(x, y)_{\min}}{\%f(x, y)_{\max} - \%f(x, y)_{\min}} \right) & \%f(x, y)_{\min} < f(x, y) < \%f(x, y)_{\max} \\ 255 & f(x, y) \geq \%f(x, y)_{\max} \end{cases}$$



a)



b)



c)

Şekil Kontrast arttırma, (a) Orijinal görüntü, (b) Kontrast arttırılmış görüntü, (c) Orijinal ve zenginleştirilmiş görüntünün histogram dağılımı [Karakuş D. "Görüntü Analiz Yöntemleri ile Kayaçların Yapısal Özelliklerinin Tanımlanması" Doktora Tezi, Kasım 2006.]

Önceki notlar

Görüntünün kontrastını ayarlama da ilk adım, aşağıdaki formülle verilen bir kontrast düzeltme faktörünü hesaplamaktır:

$$F = \frac{259(C + 255)}{255(259 - C)}$$

F: Kontrast düzeltme faktörü (double)
C=Kontrast seviye katsayısı

Algoritma'nın doğru çalışması için kontrast düzeltme faktörü (F) değerinin bir tamsayı değil, ondalık sayı (double) olması gerekir. Formüldeki C değeri, istenilen kontrast seviyesini belirtir.

Bir sonraki adım, gerçek kontrast ayarını yapmaktır. Aşağıdaki formüller kırmızı (R), yeşil (G), mavi (M) renkler için verilmiştir.

$$R' = F(R - 128) + 128$$

$$G' = F(G - 128) + 128$$

$$B' = F(B - 128) + 128$$

Bu formüllerden çıkan R,G,B renk değerlerinin 0-255 geçerli renk aralığında olması gerekir.

Kırmızı, yeşil ve mavi değerlerin 0'dan 255'e kadar geçerli bir aralıkta olmasını sağlar. Kontrast seviyesi (C) ise +255 ile -255 arasında olmalıdır. Pozitif değerler kontrast miktarını artırırken, Negatif değerler kontrast miktarını düşürecektir.

Program Kodları

Aşağıdaki resimler Orijinal resim, +128 Kontrast ve -128 kontrast miktarlarını göstermektedir.



```

public Bitmap Karsitlik()
{
    int R = 0, G = 0, B = 0;
    Color OkunanRenk, DonusenRenk;
    Bitmap GirisResmi, CikisResmi;
    GirisResmi = new Bitmap(pictureBox1.Image);

    int ResimGenisligi = GirisResmi.Width; //GirisResmi global tanımlandı.
    int ResimYuksekligi = GirisResmi.Height;
    CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi); //Cikis resmini oluşturuyor. Boyutları
    giriş resmi ile aynı olur.

    double C_KontrastSeviyesi = Convert.ToInt32(textBox1.Text);

    double F_KontrastFaktoru = (259 * (C_KontrastSeviyesi + 255)) / (255 * (259 -
    C_KontrastSeviyesi));

    for (int x = 0; x < ResimGenisligi; x++)
    {
        for (int y = 0; y < ResimYuksekligi; y++)
        {
            OkunanRenk = GirisResmi.GetPixel(x, y);
            R = OkunanRenk.R;
            G = OkunanRenk.G;
            B = OkunanRenk.B;

            R = (int)((F_KontrastFaktoru * (R - 128)) + 128);
            G = (int)((F_KontrastFaktoru * (G - 128)) + 128);
            B = (int)((F_KontrastFaktoru * (B - 128)) + 128);

            //Renkler sınırların dışına çıktıysa, sınır değeri alınacak.
            if (R > 255) R = 255;
            if (G > 255) G = 255;
            if (B > 255) B = 255;

            if (R < 0) R = 0;
            if (G < 0) G = 0;
            if (B < 0) B = 0;

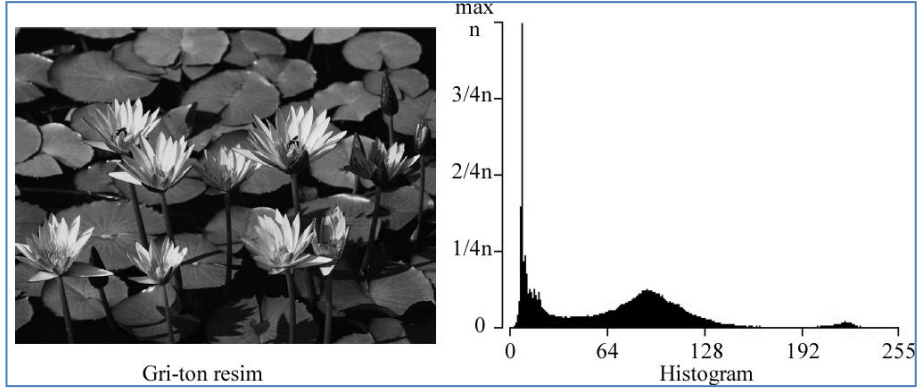
            DonusenRenk = Color.FromArgb(R, G, B);
            CikisResmi.SetPixel(x, y, DonusenRenk);
        }
    }
    pictureBox2.Image = CikisResmi;
}

```

HİSTOGRAM

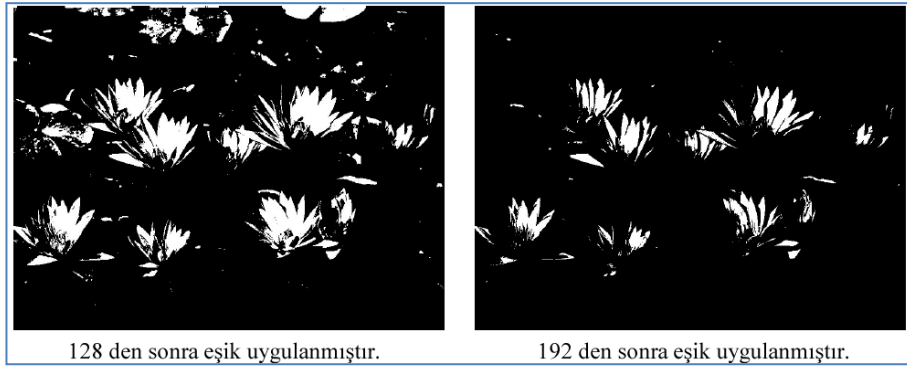
Resim üzerinde eşikleme işlemi yapıldığında belli özellikleri ortaya çıkarmak için, uygun eşik değeri kullanmamız gerekir. Fakat bu eşik değerinin hangi değer olması gerektiğini tam olarak bilemeyiz. İşte tam bu noktada histogram konusu devreye girmektedir.

Görüntü histogramı, gri-ton resmin üzerindeki piksel bilgilerini gösteren bir çubuk grafikdir. Grafiğin yatay x-ekseni 0-255 arasındaki gri ton değerlerini gösterir. y-ekseni ise bu değerlere sahip piksel sayılarını gösterir.

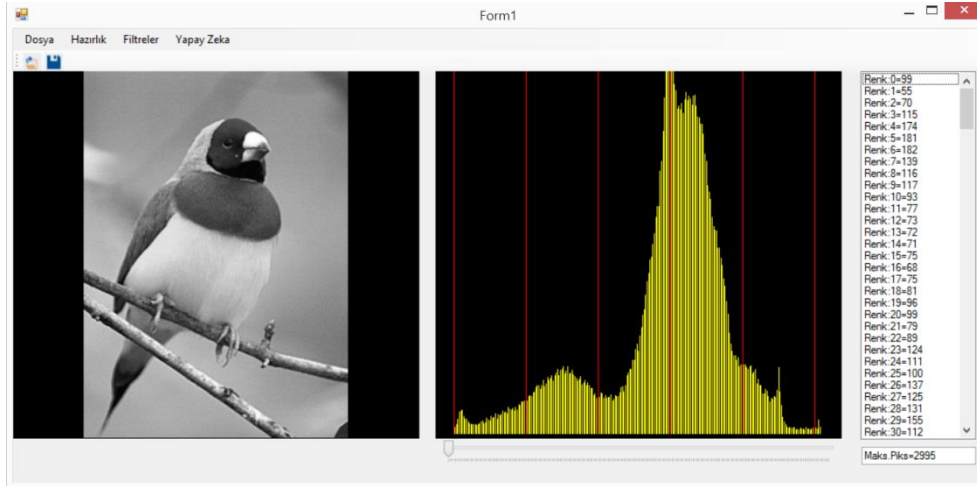


Aşağıdaki grafikte yatay eksen 0-255 arası eşit 4 parçaya bölünmüştür. Dikey eksen ise hangi renk tonundan en fazla adet varsa onun sayısına bağlı olarak (n ile gösterilmiştir) eşit kısımlara bölünmüştür. Buna göre en fazla piksel adedi 10 renk tonu civarında olduğu görülmektedir.

Resme bakıldığında renklerin koyu piksellerden oluştuğu gözükmemektedir. Bu durum histogramın sol tarafa siyaha yakın bölgede toplanmasına yol açmıştır. Aynı zamanda histogramın orta bölgeye yakın bir yerde bir zirve yaptığını görmekteyiz. Bu bölgedeki renkler zambakların yapraklarının renginden kaynaklanmaktadır (suyun üstündeki geniş yapraklar). 128 den sonra renk sayılarının azaldığını görmekteyiz. Bunlar açık renkli tonlardır. Aşağıdaki gibi yapılacak iki farklı eşikle (128 veya 192) renklerin hangi nesnelere ait olduğunu görebiliriz.



Programlama



```
//HİSTOGRAM -----
public void ResminHistograminiCiz()
{
    ArrayList DiziPiksel = new ArrayList();

    int OrtalamaRenk = 0;
    Color OkunanRenk ;
    int R = 0, G = 0, B = 0;
    Bitmap GirisResmi; //Histogram için giriş resmi gri-ton olmalıdır.
    GirisResmi = new Bitmap(pictureBox1.Image);

    int ResimGenisligi = GirisResmi.Width; //GirisResmi global tanımlandı.
    int ResimYuksekligi = GirisResmi.Height;

    int i = 0; //piksel sayısı tutulacak.
    for (int x = 0; x < GirisResmi.Width; x++)
    {
        for (int y = 0; y < GirisResmi.Height; y++)
        {
            OkunanRenk = GirisResmi.GetPixel(x, y);
            OrtalamaRenk = (int)(OkunanRenk.R + OkunanRenk.G + OkunanRenk.B) / 3; //Griton resimde üç kanal
            //renge aynı değere sahiptir.

            DiziPiksel.Add(OrtalamaRenk); //Resimdeki tüm noktaları diziye atıyor.
        }
    }

    int [] DiziPikselSayilari = new int[256];
    for (int r = 0; r < 255; r++) //256 tane renk tonu için dönecek.
    {
        int PikselSayisi=0;
        for (int s = 0; s < DiziPiksel.Count ; s++) //resimdeki piksel sayısınıca dönecek.
        {
            if (r == Convert.ToInt16(DiziPiksel[s]))
                PikselSayisi++;
        }
        DiziPikselSayilari[r] = PikselSayisi;
    }

    //Değerleri listbox'a ekliyor.
    int RenkMaksPikselSayisi = 0; //Grafikte y eksenini ölçeklerken kullanılacak.
    for (int k = 0; k <= 255; k++)
    {
```

```

        listBox1.Items.Add("Renk:" + k + "=" + DiziPikselSayilari[k]);

        if(DiziPikselSayilari[k]>RenkMaksPikselSayisi)
        {
            RenkMaksPikselSayisi=DiziPikselSayilari[k];
        }
    }

    //Grafığı çiziyor.
    Graphics CizimAlani;
    Pen Kalem1 = new Pen(System.Drawing.Color.Yellow, 1);
    Pen Kalem2 = new Pen(System.Drawing.Color.Red, 1);
    CizimAlani = pictureBox2.CreateGraphics();

    pictureBox2.Refresh();
    int GrafikYuksekligi = 400;
    double OlcekY = RenkMaksPikselSayisi / GrafikYuksekligi, OlcekX = 1.6;
    for (int x = 0; x <= 255; x++)
    {
        CizimAlani.DrawLine(Kalem1, (int)(20 + x * OlcekX), GrafikYuksekligi, (int)(20 + x * OlcekX),
        (GrafikYuksekligi - (int)(DiziPikselSayilari[x] / OlcekY)));
        if(x%50==0)
            CizimAlani.DrawLine(Kalem2, (int)(20 + x * OlcekX), GrafikYuksekligi, (int)(20 + x * OlcekX), 0);
    }
    textBox1.Text = "Maks.Piks=" + RenkMaksPikselSayisi.ToString();
}

```

Ödev 1:

Sürgü kullanarak bir resmin parlaklığını artırma yada düşürme işlemini dinamik olarak gerçekleştirin.



Ödev 2:

Picturebox içerisindeki bir resmin istenen bir bölgesine tıklandığında, o noktanın etrafındaki noktalarındaki 9 adet noktanın renk değerlerini okutup ortalama bir renk değeri elde edin. Daha sonra belirlenen renk moduna bağlı olarak, tıklanan noktanın etrafındaki tüm yakın renkleri bularak (belli bir yakalam toleransı belirlenebilir) o bölgenin

a) Parlaklığını artırın yada azaltın

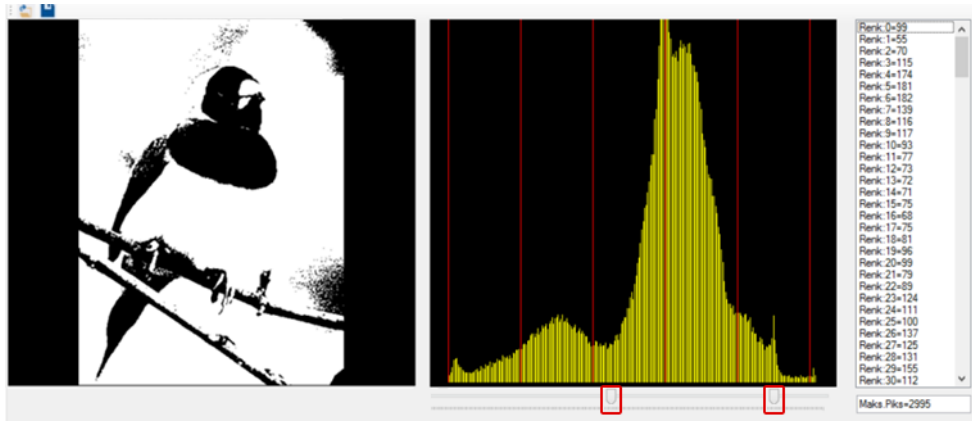
b) Rengini değiştirin.

Aşağıda verilen örnek resimde arabanın kapısı üzerindeki tüm bölge bulunmuş ve o bölgenin rengi değiştirilmiştir. Uygulamanız buna benzer olsun.



Ödev 3:

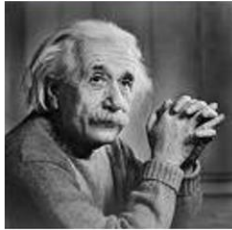
Form üzerinde iki tane sürgü kullanarak Eşikleme yapan bir program yazınız. Sürgüler grafik altında x eksenı boyunca olsun. Sürgü ilerletildikçe resmin nasıl değıştiğı gözlemlenebilsin.



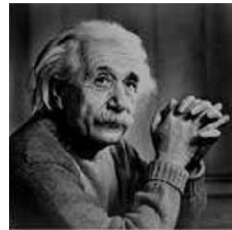
Ödev 4:

Ařağıdaki gibi bir siyah beyaz resmin belli bir sınırın ötesindeki beyaz renkleri sıfırlayan ve o bölgelerin koyulařmasını sağılayan ve bunu histogram üzerinden ayarlayan programı yazınız. Histogramın altındaki sürgü sürüklendiğinde resim değışsin ve resmin histogramı tekrar yenilensin. Ařağıdaki histogramda tek bir sürgü kullanılmıřtır. Beyaz bölge geriye çekildiğinde tüm 0-255 arasındaki sayılar ölçeklenerek yenilenmesi gerekir. Diyelim sürgü 180 çekilirse, önceden 255 olan renk artık 180 değeri gelmiş olması gerekir. Tüm diğerk renk değerkleri de 0-180 arası ölçeklenmelidir.

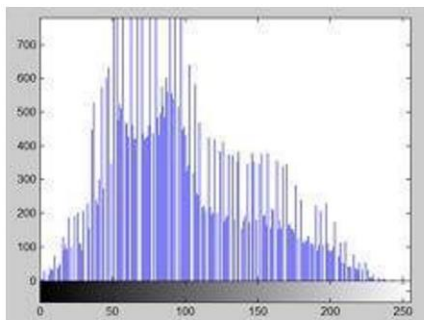
Orjinal Resim



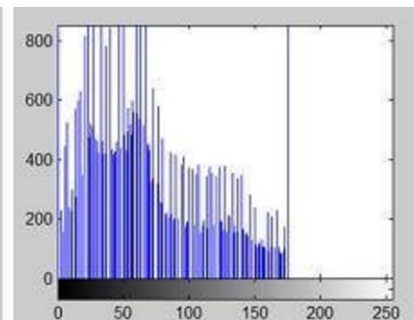
Yeni Resim



Orjinal Histogram



Yeni Histogram



Renk Dönüşümleri ve Dönüşüm Matrisi

<http://www.cescript.com/2013/09/renk-donusumleri-ve-donusum-matrisi.html>

Renk dönüşüm filtreleri

<http://www.cescript.com/2013/09/instagram-nedir-nasil-calisir.html>

HSV-RGB Uzak Dönüşümü

<http://www.cescript.com/2012/07/hsv-rgb-uzay-donusumu.html>

RGB-HSV Uzak Dönüşümü

<http://www.cescript.com/2012/07/rgb-hsv-uzay-donusumu.html>

İmge Renksizleştirme (Image Decolorization)

<http://www.cescript.com/2015/10/imge-renksizlestirme-image.html>

Histogram İşlemleri

<http://www.cescript.com/2012/06/histogram-saysal-bir-resim-icerisinde.html>

Otsu Metodu ile Adaptif Eşikleme

<http://www.cescript.com/2012/07/otsu-metodu-ile-adaptif-esikleme.html>

Görüntü Renk İndirgeme

<http://www.cescript.com/2013/10/goruntu-renk-indirgeme.html>

C++ BMP Okuma

<http://www.cescript.com/2012/03/c-bmp-okuma.html>

C++ BMP Yazma

<http://www.cescript.com/2012/03/c-bmp-yazma.html>

Bağlantılı Bileşen Etiketleme

<http://www.cescript.com/2012/09/baglantili-bilesen-etiketleme.html>

Perspektif dönüşümü

<http://www.cescript.com/2013/12/perspektif-donusumu.html>

