GÖRÜNTÜ İŞLEME-9.HAFTA

ARİTMETİK OPERATÖRLER

Görüntü aritmetiği iki veya daha fazla görüntüye standart aritmetik işlemlerden birini (toplama, çıkarma, çarpma, bölme) veya mantıksal bir operatörü uygulayan işlemleridir. İşlem giriş resmi ile çıktı resmi arasında piksel piksel uygulanır. Bu nedenle, görüntülerin normalde aynı boyutta olması gerekir. Görüntü aritmetiği görüntü işlemenin en basit şekli ve dolayısı ile hızlı işlemlerdir.

Birçok uygulamada, işlenen görüntüler aynı sahneden farklı zaman noktalarında alınır. Örneğin, aynı sahnenin ardışık görüntüleri eklenerek rastgele gürültünün azaltılması veya iki ardışık görüntü çıkartılarak hareket algılama işlemleri yapılabilir. Mantıksal işlemler ise genellikle iki görüntüyü birleştirmek için kullanılır.

GPT

Görüntü İşlemede Aritmetik operatörler nerede kullanılır?

Görüntü işlemede aritmetik operatörler, piksel değerlerini değiştirmek veya hesaplamak için kullanılır. Aritmetik operatörler, görüntülerdeki pikseller arasında matematiksel işlemler yapmak için kullanılır ve çeşitli uygulamalarda kullanılırlar. İşte aritmetik operatörlerin bazı kullanım örnekleri:

Görüntü toplama (Addition): İki veya daha fazla görüntüyü toplama işlemiyle birleştirmek için kullanılır. Her pikselin değeri, aynı koordinatlardaki diğer görüntülerin piksel değerleriyle toplanır.

Görüntü çıkarma (Subtraction): İki görüntü arasındaki farkı hesaplamak için kullanılır. İki görüntünün piksel değerlerini çıkartarak, farklılıkları vurgulamak veya arka planı çıkarmak için kullanılabilir.

Görüntü çarpma (Multiplication): İki görüntünün piksel değerlerini çarparak yeni bir görüntü oluşturur. Bu işlem, görüntüleri birbirleriyle karıştırarak renk efektleri oluşturmak veya kontrastı artırmak için kullanılabilir.

Görüntü bölme (Division): Bir görüntüyü diğer bir görüntüye bölmek için kullanılır. Bölme işlemi, görüntüleri normalleştirme veya bir görüntüyü diğerine göre orantılı hale getirme amaçlarıyla kullanılabilir.

Görüntü ölçeklendirme (Scaling): Bir görüntüyü belirli bir oranda büyütmek veya küçültmek için çarpma veya bölme işlemleri kullanılabilir. Örneğin, görüntüyü yakınlaştırmak veya uzaklaştırmak için ölçeklendirme işlemi yapılabilir.

Bu sadece bazı örneklerdir ve aritmetik operatörlerin kullanımı çok çeşitli olabilir. Görüntü işleme algoritmalarında, piksel düzeyinde matematiksel işlemler yapmak için aritmetik operatörler sıklıkla kullanılır.

A- Piksel Toplama, Görüntü Ekleme

Bu operatör giriş olarak aynı boyutlarda iki adet görüntüyü alıp, çıkış olarak aynı boyutta üçüncü bir görünüşü üretir. Çıkış resmi, giriş resminlerinde aynı konumdaki piksellerin toplamı ile üretilir. İkiden fazla görüntüyü birleştirme yada her piksele belirli bir sabitin eklenmesi şeklinde uygulamalar da yapılabilir.

İki görüntünün eklenmesi tek bir geçişte doğrudan gerçekleştirilir. Çıktı piksel değerleri şu şekilde oluşturulur.

$$Q(x,y) = P_1(x,y) + P_2(x,y)$$

Tek bir görüntüye sabir bir sayı eklenmek isteniyorsa formülasyon aşağıdaki gibi olacaktır.

$$Q(x,y) = P_1(x,y) + C$$

Renkli görüntülerde, çıktı değerini üretmek için her bir kanal (R, G, B) kendi arasında toplanır. Sabit değer ise her bir kanala eklenir. Sabit değer eklenerek yapılan uygulamar, resmin parlaklığını artıran bir işlemdir.

Ekleme işleminden sonra, renk değerleri üst limiti aşarsa (255), taşan kısımlar için şu uygulamalar yapılabilir.

a) Taşan değerler 255 sabitlenebilir.

b) Taşan en üst değer 255 olacak şekilde tüm pikseller Normalleştirilebilir.

c) Ekleme işlemi yapılmadan önce orijinal resim belli bir sayı ile ölçeklenebilir (A=0.8 gibi).

$$Q(x,y) = P_1(x,y) * A + P_2(x,y)$$

Benzer şekilde her iki resmi de ölçekleyip toplayabiliriz (A=0.5, B=0.5).

$$Q(x,y) = P_1(x,y) * A + P_2(x,y) * B$$

c) Taşan kısımlar 0 dan başlayacak şekilde tekrar başa dönebilir.

d) Eğer bir görüntünün kenar çizgileri ile toplama yapılıyor ise, sadece kenar kısımlarında kenar görüntüsün değeri diğer alanlarda ise orijinal resmin pikselleri kullanılabilir.

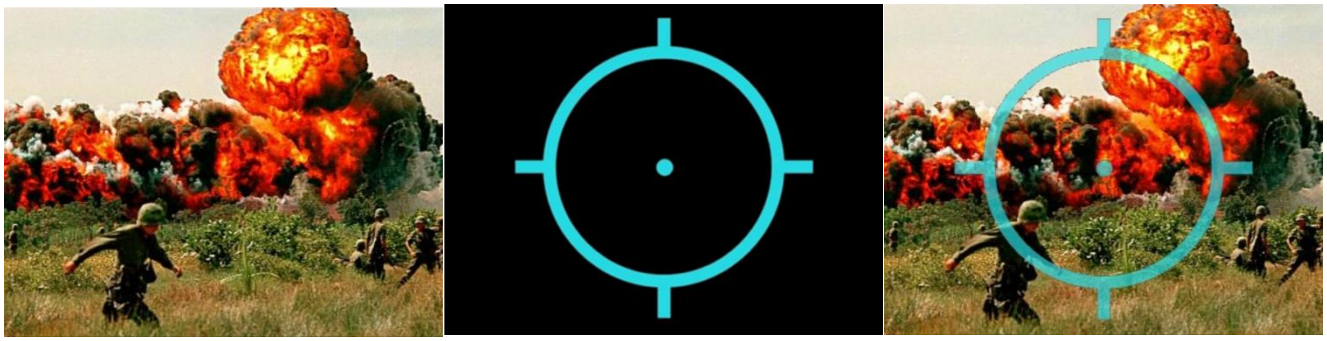
Programlama (Görüntü ekleme)

Aşağıdaki ilk iki resim farklı toplama işlemine ve sınır aşma uygulamalarına tabi tutularsa şu sonuçlar alınır.

Aşağıdaki birinci resim direk iki resim toplanıp, sınırı aşan değerler 255 de sabitlenmiştir. İkinci resimde ise her iki resimde 0.5 ölçekle çarpılarak önce karartılıp daha sonra değerler toplanmıştır.

Aşağıdaki iki resim toplanırken nişangahın renkli piksellere gelince renklerin%70 nişangahtan, %30 ana sahneden alınmıştır.

Örnek 1



```
private void pikselToplamaToolStripMenuItem_Click(object sender, EventArgs e)
{
    Bitmap Resim1, Resim2, CikisResmi;
    Resim1 = new Bitmap(pictureBox1.Image);
    Resim2 = new Bitmap(pictureBox2.Image);

    int ResimGenisligi = Resim1.Width;
    int ResimYuksekligi = Resim1.Height;

    CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);

    Color Renk1, Renk2;
    int x, y;
    int R=0, G=0, B=0;

    for (x = 0; x < ResimGenisligi; x++) //Resmi taramaya şablonun yarısı kadar dış kenarlardan
içeride başlayacak ve bitirecek.
    {
        for (y = 0; y < ResimYuksekligi; y++)
        {
            Renk1 = Resim1.GetPixel(x, y);
            Renk2 = Resim2.GetPixel(x, y);

            //İki resmi direk toplama
            //R = Renk1.R + Renk2.R;
            //G = Renk1.G + Renk2.G;
            //B = Renk1.B + Renk2.B;

            //İki resmi ölçekli olarak toplama
            if (Renk2.R > 20 && Renk2.G > 20 && Renk2.B > 20)
            {
                R = Convert.ToInt16(Renk1.R * 0.3 + Renk2.R * 0.7);
                G = Convert.ToInt16(Renk1.G * 0.3 + Renk2.G * 0.7);
                B = Convert.ToInt16(Renk1.B * 0.3 + Renk2.B * 0.7);
            }
            else
            {
                R = Renk1.R;
                G = Renk1.G;
                B = Renk1.B;
            }
            //Sınırı aşan değerleri 255 ayarlama
            if (R > 255) R = 255;
            if (G > 255) G = 255;
            if (B > 255) B = 255;

            //Sınırı aşan değerleri Başa sarma şeklinde ayarlama
            //if (R > 255) R = (R - 255);
            //if (G > 255) G = (G - 255);
            //if (B > 255) B = (B - 255);

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

        }
```

```
    }
    pictureBox3.Image = CikisResmi;
}
```

Normalizasyon Yaparak İki Resmi Toplama



```
private void PIKSEL_TOPLAMA_Click(object sender, EventArgs e)
{
    Bitmap Resim1, Resim2, CikisResmi;
    Resim1 = new Bitmap(pictureBox1.Image);
    Resim2 = new Bitmap(pictureBox2.Image);

    int ResimGenisligi = Resim1.Width;
    int ResimYuksekligi = Resim1.Height;

    CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);

    Color Renk1, Renk2;
    int x, y;
    int R = 0, G = 0, B = 0;

    int EnBuyukDeger = 0, EnKucukDeger = 255;


    for (x = 0; x < ResimGenisligi; x++) //Resmi taramaya şablonun yarısı kadar dış kenarlardan
içeride başlayacak ve bitirecek.
    {
        for (y = 0; y < ResimYuksekligi; y++)
        {
            Renk1 = Resim1.GetPixel(x, y);
            Renk2 = Resim2.GetPixel(x, y);

            ////İki resmi direk toplama
            R = Renk1.R + Renk2.R;
            G = Renk1.G + Renk2.G;
            B = Renk1.B + Renk2.B;

            int Gri = (R + G + B) / 3;

            //Sınırı aşan değerleri 255 ayarlama. Gri resim üzerinde işlem yapıldığı için Sadece
R ye bakıldı.
            if (Gri > EnBuyukDeger)
                EnBuyukDeger = Gri;

            if (Gri < EnKucukDeger)
                EnKucukDeger = Gri;

            if (R > 255) R = 255;
            if (G > 255) G = 255;
            if (B > 255) B = 255;

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

        }
    }

    pictureBox3.Image = CikisResmi;
```

```
    pictureBox4.Image = Normalizasyon(Resim1, Resim2, EnBuyukDeger, EnKucukDeger);

}
```

```
public Bitmap Normalizasyon(Bitmap Resim1, Bitmap Resim2, int EnBuyukDeger, int EnKucukDeger)
{
    Bitmap CikisResmi;

    int ResimGenisligi = Resim1.Width;
    int ResimYuksekligi = Resim1.Height;

    CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);

    Color Renk1, Renk2;
    int x, y;
    int R = 0, G = 0, B = 0;

    int UstSinir = 0, AltSinir = 0;

    if (EnBuyukDeger > 255)
        UstSinir = 255;
    else
        UstSinir = EnBuyukDeger;

    if (EnKucukDeger < 0)
        AltSinir = 0;
    else
        AltSinir = EnKucukDeger;

    for (x = 0; x < ResimGenisligi; x++) //Resmi taramaya şablonun yarısı kadar dış kenarlardan
içeride başlayacak ve bitirecek.
    {
        for (y = 0; y < ResimYuksekligi; y++)
        {
            Renk1 = Resim1.GetPixel(x, y);
            Renk2 = Resim2.GetPixel(x, y);

            ////İki resmi direk toplama
            R = Renk1.R + Renk2.R;
            G = Renk1.G + Renk2.G;
            B = Renk1.B + Renk2.B;


            int NormalDegerR = (((UstSinir - AltSinir) * (R - EnKucukDeger)) / (EnBuyukDeger -
EnKucukDeger)) + AltSinir;
            int NormalDegerG = (((UstSinir - AltSinir) * (G - EnKucukDeger)) / (EnBuyukDeger -
EnKucukDeger)) + AltSinir;
            int NormalDegerB = (((UstSinir - AltSinir) * (B - EnKucukDeger)) / (EnBuyukDeger -
EnKucukDeger)) + AltSinir;

            //EnBuyuk ve EnKucuk değerler Gri renge göre ayarlandığından Yinede bu renkler
sınırı geçebilir.
            if (NormalDegerR > 255) NormalDegerR = 255;
            if (NormalDegerG > 255) NormalDegerG = 255;
            if (NormalDegerB > 255) NormalDegerB = 255;

            if (NormalDegerR < 0) NormalDegerR = 0;
            if (NormalDegerG < 0) NormalDegerG = 0;
            if (NormalDegerB < 0) NormalDegerB = 0;

            CikisResmi.SetPixel(x, y, Color.FromArgb(NormalDegerR, NormalDegerG, NormalDegerB));

        }
    }
    return CikisResmi;
```

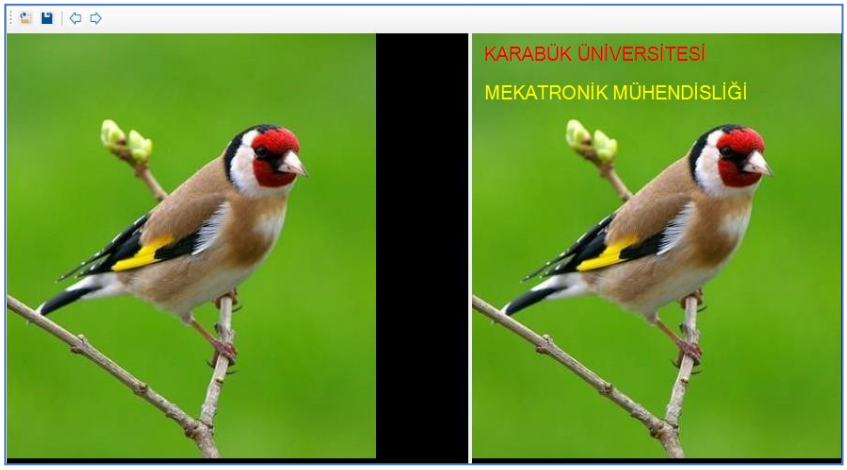
```
}
```

Ek Konu: Resim Üzerine Yazı, Çizgi ve Resim Ekleme (Hazır fonksiyonlar)

Resim üzerine yazı ve çizgi gibi eklemek yapmak istediğimizde aşağıdaki hazır fonksiyonları kullanabiliriz.

Normal kameradan alınan görüntülerin üzerine bazen tarih ve saat işlenmesi gerekebilir. Yada görüntü üzerinde nişangah (artikıl) benzeri bir şeklin düşürülmesine ihtiyaç olabilir. Ayrıca bir görüntü oluşturulurken (daha çok video görüntülerinde) köşede başka bir kameranın görüntüsünün de olmasını isteyebiliriz. Yada aynı anda bir çok kameranın görüntüsü tek çerçeve içerisinde oluşturulması gerekebilir. Video filmleri resimlerin arda arda görüntülenmesi olduğundan bir resim üzerinde yapılacak bu işlemler tekrarlandığında video görüntüsünde de yapılmış olacaktır.

Örnek 2. Resim üzerine yazı yazma




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

string Metin1 = "KARABÜK ÜNİVERSİTESİ";
string Metin2 = "MEKATRONİK MÜHENDİSLİĞİ";

Point BirinciKonum = new Point(10, 10);
Point IkinciKonum = new Point(10, 50);

Graphics graphics = Graphics.FromImage(CikisResmi);

Font ArialFontTipi = new Font("Arial", 15);

graphics.DrawString(Metin1, ArialFontTipi, Brushes.Red, BirinciKonum);
graphics.DrawString(Metin2, ArialFontTipi, Brushes.Yellow, IkinciKonum);


pictureBox2.Image = CikisResmi;

string ResimDosyaYolu = @"..\..\..\Resimler\Deneme.jpg"; //Direk dosya adı yazılırse bin/debug
klasörüne yazılır.
CikisResmi.Save(ResimDosyaYolu);
```

Örnek 3. Resim üzerine yazı yazma

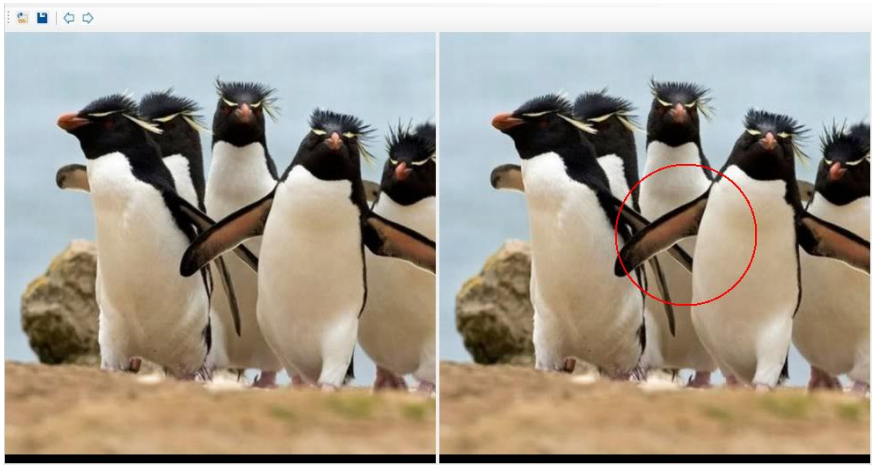

```
private void btnYaziEkle_Click(object sender, EventArgs e)
{
     Bitmap GirisResmi = new Bitmap(pictureBox1.Image);
     Bitmap CikisResmi = new Bitmap(GirisResmi);
     CikisResmi = GirisResmi;

     int FontBoyutu = 25;

     Font FontBilgisi = new Font("Times New Roman", FontBoyutu, FontStyle.Regular);

     string Metin = "KARABÜK ÜNİVERSİTESİ";

     Color Renk1 = Color.FromArgb(255,255,10,10); //Alfa,Red,Green,Blue ile tanımladık
     Color Renk2 = Color.FromName("Yellow");

     Rectangle Dikdortgen = new Rectangle(0, 0, 10, 20);

     LinearGradientBrush GecisliFirca = new LinearGradientBrush(Dikdortgen, Renk1, Renk2,
LinearGradientMode.Vertical);

     int xKonum = 100;
     int yKonum = 80;


     Graphics Grafik = Graphics.FromImage(CikisResmi);
     Grafik.DrawString(Metin, FontBilgisi, GecisliFirca, xKonum, yKonum);

     pictureBox2.Image = CikisResmi;

}
```

Örnek 4. Resim üzerine resim ekleme


```
 private void btnResimBirlestir_Click(object sender, EventArgs e)
 {
     pictureBox1.Image = Image.FromFile(@"..\..\..\Resimler\Araba2.jpg");
```

```
     pictureBox2.Image = Image.FromFile(@"..\..\..\Resimler\Araba1.jpg");

     Bitmap GirisResmi1 = new Bitmap(pictureBox1.Image);
     Bitmap GirisResmi2 = new Bitmap(pictureBox2.Image);

     int Xkoordinati = 20; int Ykoordinati = 20; int Genislik = 200; int Yukseklik = 150;

     Rectangle Dikdortgen = new Rectangle(Xkoordinati, Ykoordinati, Genislik, Yukseklik);

     Graphics Grafik = Graphics.FromImage(GirisResmi2);
     Grafik.DrawImage(GirisResmi1, Dikdortgen);

     string ResimDosyaYolu = @"..\..\..\Resimler\Deneme3.jpg"; //Direk dosya adı yazılırse bin/debug
klasörüne yazılır.
     GirisResmi2.Save(ResimDosyaYolu);

 }
```

Örnek 5. Resim üzerine çizgi ekleme



```
private void btnResimUzerineCizgiEkle_Click(object sender, EventArgs e)
 {
     Bitmap GirisResmi = new Bitmap(pictureBox1.Image);
     Bitmap CikisResmi = new Bitmap(GirisResmi);
     CikisResmi = GirisResmi;

     Graphics Grafik = Graphics.FromImage(CikisResmi);

     Color Renk = Color.Red;// Çizim kaleminin rengini beyaz atıyor
     Pen KalemCizgi = new Pen(Renk, 2);//Çizim

     Grafik.DrawEllipse(KalemCizgi, 200, 150, 160, 160);

     pictureBox2.Image = CikisResmi;

      string ResimDosyaYolu = @"..\..\..\Resimler\Deneme4.jpg"; //Direk dosya adı yazılırse bin/debug
klasörüne yazılır.
     CikisResmi.Save(ResimDosyaYolu);

 }
```

Ödev 1: Gece görüş kamerası

a) Gece ortamda ışıklar çok düşükken (odada mum ışığı altında olabilir, cep telefonu ışığı üzerine bir şey kapatılarak olabilir yada balkondan gece sokak görüntüsü alınarak denenebilir) alınan video görüntüleri üzerinde piksel değerleri üzerine sabit bir değer ekleyerek görüntüyü aydınlatın (gece görüntüsünü gündüz gibi yapın). Bunun için eklenecek sabit sayıyı bir sürgü (scrollbar) ile ayarlanabilir yaparsanız daha uygun değerde parlaklığı artırmış olursunuz. Görüntü işleme hızı nedeniyle her kare belki biraz kesikli olabilir ama olabildiğince gece görüş kamerası şeklinde olayı gösterin. (Not. Bu şekilde bir gece görüş kamerası geceyi gündüz gibi gösteren bir uygulamadır, fakat

termal kamera olayı değildir. Termal kamerada insan gözünün göremediği frekansları (kızıl ötesi bölge) görebileceğimiz frekansa dönüştürmek gerekir)

b) Aynı uygulamayı şu şekide değiştirin. Karanlık ortamda çıkan en büyük renk değerini 255 gelecek şekilde ve en küçük renk değerini de 100 gelecek şekilde Normalize ederek resmi yeniden düzenleyin. Buradaki 100 sayısını Sürgü (Trackbar) ile ayarlanabilir yapın.

B- Piksel Çıkarma , Görüntü Çıkarma

Piksel çıkarma operatöründe, iki resim birbirinde çıkarılarak üçüncü çıktı resmi oluşturulur.

$$Q(i,j) = P_1(i,j) - P_2(i,j)$$

Çıktı oluşturulurken ortaya çıkacak negatif değerler mutlak değeri alınarak pozitife dönüştürülebilir.

$$Q(i,j) = |P_1(i,j) - P_2(i,j)|$$

Sabit bir değerde çıkarılarak çıktı resmi elde edilebilir.

$$Q(i,j) = P_1(i,j) - C$$

Renkli görüntülerde her bir değer R,G,B kanallarından ayrı ayrı çıkarılır.

Sınırları taşan değerler için şu uygulamalar yapılabilir. Hangisinin uygulanacağına, yapılan uygulamanın türüne göre karar verilir. Çıkarma işleminde ortaya çıkan değerler negatif bölgede olacaktır. Buna göre;

a) Çıkış piksel değerleri negatifse direk olarak sıfıra eşitlenebilir

b) Negatif değer 255 den çıkarılarak tekrar yukarıdan geriye doğru değer atanabilir. Yani -30 için 255-30=225 değeri atanabilir.

c) Yukarıda formülü yazıldığı gibi negatif değeler mutlak değeri alınarak direk pozitife dönüştürülebilir.

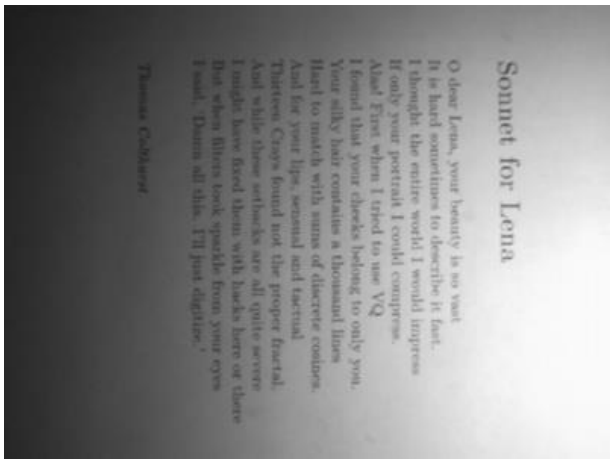
d) Bazı görüntü formatları eksi değerleri kabul eder. Bu tip görüntülerde olduğu gibi bırakılabilir.

e) Negatif değer 0 a çekilerek tüm resim değerleri Normalizasyon yapılarak, yani her pikselin değeri üst ve alt limiter arasın orantısal olarak dağıtılabilir. -30 değeri 0 yapıldığında üst limitte 250 ise bu durumda bütün pikseller 0 ile 250 arasında yeniden hesaplanarak dağıtılabilir.

Örnek 6. Arka plandan görüntü çıkarma. Yazıları okunaklı hale getirme

Görüntü yada piksel çıkarmanın yaygın bir kullanımı olarak resimde arka plan görüntülerini kaldırma gösterilebilir. Böylece ön plandaki görüntüler daha kolay analiz edilir. Örneğin: bazı tarama işlemlerinde yada yetersiz bir aydınlatma ortamında çekilen bir sayfanın görüntüsü aşağıdaki gibi olabilmektedir. Bu işlemde sahnenin boş bir görüntüsü (yazının olduğu ortama boş bir sayfa konarak oluşturulabilir), üzerinde yazı bulunan görüntüden çıkarılarsa ön plandaki yazılar daha iyi tespit edilir.

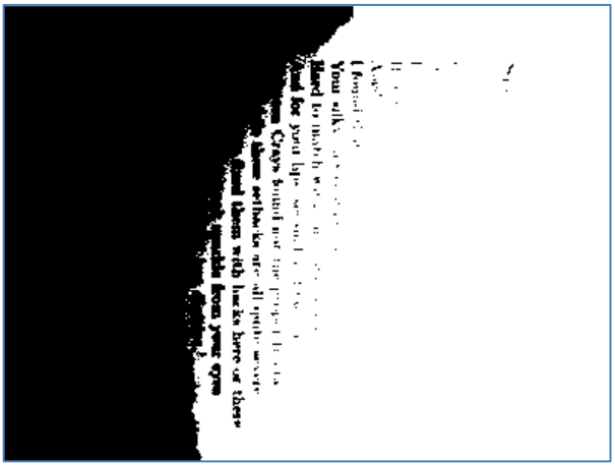
Endüstriyel bir uygulamada ise bir bant üzerindeki cisimleri homojen olmayan bir ışık altında kontrol edilirse ön plandaki cisimlerin daha anlaşılır olması için bu yöntem kullanılabilir. Görüntü alınan ortamda, kamera ve ışık konumu sabit ise, sahne boş iken arka planı oluşturan bir görüntü alınır ve ardından sahneye giren cisimler varken alınan görüntülerden bu arka plan çıkarılarak esas incelenen cisimler tespit edilebilir.



Orijinal Resim

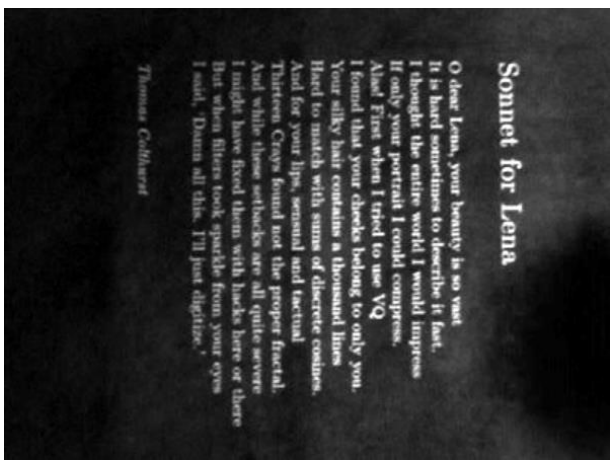
Gölge Resmi (boş sahne resmi)

Böyle bir görüntüde belli bir değerin üzerindeki beyaz kısımları tamamen beyaz, geri kalanı tamamen siyah yaparsak başarısız bir ayırma işlemi gerçekleştirmiş oluruz. Aşağıdaki resim 100 değeri ile eşiklenerek elde edilmiştir.



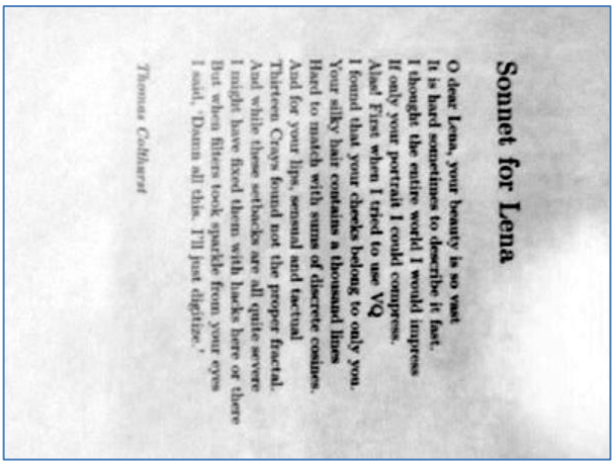
```
OkunanRenk = GirisResmi.GetPixel(x, y);

R = OkunanRenk.R;

if (R > 100)
  R = 255;
else
  R = 0;
```

Bu problemi çözmek için sahne dolu iken alınan görüntüden (Orijinal resim), sahne boş iken alınan görüntü (Gölge resmi) çıkarılırsa ön plandaki cisimlerin görüntü değeri elde edilir. Çıkarma işleminde negatif sayılar elde etmeyi önlemek yukarıda anlatılan uygun yöntemler denenerek daha iyi sonuçlar elde edilebilir. Aşağıdaki resim çıkarma işlemine mutlak değer fonksiyonu uygulanarak elde edilmiştir.




```
Renk1 = Resim1.GetPixel(x, y);
Renk2 = Resim2.GetPixel(x, y);

R = Math.Abs(Renk1.R - Renk2.R);
G = Math.Abs(Renk1.G - Renk2.G);
B = Math.Abs(Renk1.B - Renk2.B);
```

Bu görüntünün negatifi alınırsa daha iyi bir sonuç elde edilir.




```
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(i, j, DonusenRenk);
```

Elde edilen bu görüntüde arka plandaki gölgeleri 170 değeri ile eşiklersek (ayarlanabilir bir sürgü-scrollbar ile yapmak daha uygun olur) aşağıdaki sonucu elde ederiz.

```
OkunanRenk = GirisResmi.GetPixel(x, y);

R = OkunanRenk.R;

int EsiklemeDeger = ScroollBar1.Value;

if (R > EsiklemeDeger)
  R = 255;
else
  R = 0;
```

Görüntünün zayıf aydınlatılmış alanlarında kontrast (yani ön plan ve arka plan yoğunluk farkı) daha iyi aydınlatılmış kısma göre daha düşük olduğu için resim üzerindeki homojenlik henüz daha sağlanabilmiş değildir. Daha sonra gelecek konularda bu problemi biraz daha iyileştirmeye çalışacağız fakat, ışık koşulları homojen olmadığı için tam olarak %100 bir çözüm imkansızdır.

Araştırma 1: Bu uygulamayı gamma corrected yöntemini kullanarak daha da iyileştirmeye çalışın ve araştırın.

Ödev 2: Yazı Okuma

Arka plandan görüntü çıkararak yazı okuma uygulamasının bir benzerini kendiniz yapın. Bütün uygulamalarda kamerayı sabitleyin. Uygulamalarda cep telefonu kamerası anlık olarak kullanılacak ise, bu görüntülerin bilgisayara nasıl aktarılacağını bulun (blue tooth yada usb üzerinden aktarma yapılabilir, araştırın).

Yukarıdaki örnekde verildiği gibi, üzerine gölge düşmüş bir kağıdın üzerindeki yazı ve resimleri okunaklı şekilde tespit etmeye çalışın. (Cep telefonu kamerası kullanacaksanız masa kenarına telefonu sabitleyip yere kağıdı koyabilirsiniz. Yan açıdan üzerine bir cismin gölgesi düşecek şekilde aydınlatma yapabilirsiniz.

Örnek 7. Hareket algılama

Görüntüleri çıkarırken değerleri mutlak değer olarak hesaplama, sahnede değişimleri algılamak için kullanılabilir. Birbirinin aynı iki görüntüde (yada video gibi bir dizi resminde) sahnede hareket eden bir şey yoksa, iki kare arasındaki mutlak fark oluşmaz ve çıktı çoğunlukla sıfır değerli piksellerden oluşur (siyah resim). Eğer sahnede bazı cisimler hareket ediyorsa, mekansal değişimin olduğu bölgelerdeki piksellerde mutlak değer olarak farklılıklar ortaya çıkacaktır.

Bu tür bir hareket algılama için aşağıdaki iki resim arasındaki farkı çıkardığımızda üçüncü kareyi elde ederiz.

Böyle bir uygulama belli aralıklarla çekilen fotograflar işlenerek foto-kapan olarak kullanılabilir. Yada video görüntüsü işlendiğinde hereket algılama olarak kullanılabilir. Sahnede çok küçük hareketleri algılamamak için mutlak değer olarak toplam değer belli bir sayının üzerine çıktığında hareket var kabul edilebilir. Fakat bu hesaplamada gri resimleri çıkarımı üzerinden değil, siyah-beyaz resimler üzerinden hesaplamak daha doğru olur. Yani piksel farkı 0 yada 255 olacak şekilde hesaplama yapılırsa, renk derinliği işleme girmeden mekansal değişim (hareket) algılanmış olur. Bu uygulama sahnede hereketin olduğu bölgeyi de verdiği için için kamera o bölgeye odaklanıp zoomlama işlemi yaptırılabilir.

C- Piksel Çarpma ve Ölçekleme

Çarpma işleminin iki farklı türünden bahsedilebilir. Birincisi iki tane giriş resmi alınır. Bunlardan birincisinin piksel değeri, ikinci resmin piksel değerleri ile çarpılması ile çıktı görüntüsü elde edilir. Buna Piksel çarpma denir. Bu işlem iki görüntü ile yapılırken biri diğeri üzerinde maskeleme yapmak için kullanılabilir. Çarpma işlemi renk değerlerinin hızla daha yükselmesini sağlamaktadır. Sonuçta piksel toplama işlemine benzer. Piksel toplama işleminde renk değerleri daha yavaş yükselir. Bu nedenla bu operatörde değerler üst sınırı daha hızlı yaklaşır.

$$Q(i,j) = P_1(i,j)\ x\ P_2(i,j)$$

Diğer bir uygulama ise bir giriş resmi belli bir ölçek değeri ile çarpılarak çıktı resminin oluşturulması sağlanır. Buna Piksel ölçekleme denir. Bu ölçekleme Geometrik ölçekleme ile karıştırılmamalıdır. Geometrik ölçeklemede piksel koordinatları değişir (spatial), bunda ise piksel değerleri değişir (spektral). Burada C katsayısı 1 den düşük olursa resim aydınlığını azaltır. Renkli resimlerde ise her bir R, G, B kanalı ayrı ayrı sabit ile çarpılır.

$$Q(i,j) = P_1(i,j)\ x\ C$$

Çarpım katsayısı C birden büyük olursa o zaman üst sınır ya 255 i geçebilir. Bu durumda aşan değerler 255 de sabitlenebilir. Yada başa sararak aşan değeler 0 dan itibaren yeniden değer verilebilir.. Örneğin 280 ise 280-255=35 gibi. Çarpım işleminde piksel değerleri üst sınıra hızlıca yaklaşır. Sistem olarak sabit sayı ile toplamaya benzer.

Bu yöntem resimleri aydınlatmak için kullanılabilir. Çarpım katsayısı 1 den küçük değer atanırsa resmi karartır. Çarpma işlemi yukarıdaki birinci konu olan Piksel toplamadan daha doğal aydınlatma/karartma sonucu verir.

Örnek 7. İki resmi çarpma

Aşağıdaki örnekte iki resimde her kanal diğer resim değeri ile çarpılmış ve sınırını aşanlar 255 sabitlenmiştir.

```
...
Renk1 = Resim1.GetPixel(x, y);
```

```
Renk2 = Resim2.GetPixel(x, y);

R = Renk1.R * Renk2.R;
G = Renk1.G * Renk2.G;
B = Renk1.B * Renk2.B;

if (R > 255) R = 255;
if (G > 255) G = 255;
if (B > 255) B = 255;

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

...
```

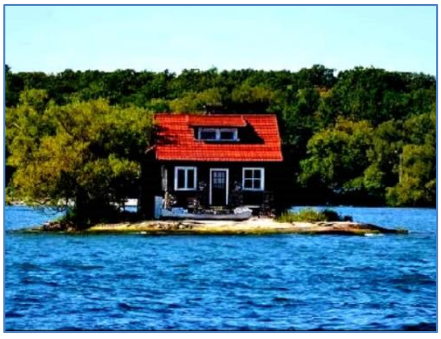
Örnek 8. Resmi Ölçekleme

Aşağıdaki resim düşük bir ışıkta çekilmiş fotograftır. Bu resmi 3x sayısı ile ölçeklersek yandaki resim elde edilir. Burada ölçekleme yaparken en üst limiti aşmamasına dikkat etmeliyiz. Aynı resim 5x ile öçeklenirse ve 255 geçen değerler başa sarılırsa aşağıdaki 3 resim elde edilir. Burada çarpma işlemi sonunda aşan sayıların ne yapılacağı kullanılacak uygulamalara uygun olarak belirlenebilir.

0.5x ile ölçekleme

Orijinal Resim (1x Ölçekleme)

1.5x ile ölçekleme



2x ile ölçekleme

2.5x ile ölçekleme

3x ile ölçekleme

4x ile ölçekleme

5x ile ölçekleme

5x ölçekleme sınırı geçenler başa sarıldı

```
Renk1 = Resim1.GetPixel(x, y);
```

```
double Olcekleme = (double)Surgu1.Value/100.0; //Sürgü sınırları 0-500 arasına ayarlandı

R = Convert.ToInt16(Renk1.R * Olcekleme);
G = Convert.ToInt16(Renk1.G * Olcekleme);
B = Convert.ToInt16(Renk1.B * Olcekleme);

//Başa sararken değerler mod alarak hesaplanıyor. Böylece 540 % 255 =30 olmuş olur.
if (R > 255|| G > 255|| B > 255)
{
    R = (R % 255);
    G = (G % 255);
    B = (B % 255);
}

//if (R > 255) R = 255;
//if (G > 255) G = 255;
//if (B > 255) B = 255;

CikisResmi.SetPixel(x, y, Color.FromArgb(R, G, B));
```

D-Piksel Bölme

İki resim değerleri birbirine bölünerek çıktı görüntüsü elde edilebilir. Sonuçta piksel çıkarma işlemine benzer. Ortaya çıkan değerler hızla küçüleceği için, çıkarma işleminden daha küçük sayılar elde edilir. Örneğin çıkarma işleminde 180 değeri ile 30 değerleri 150 sonucunu verirken, Bölme işleminde 180/30=6 değerini verir. Bu nedenla bu operatörde değerler alt sınırı daha hızlı yaklaşır.

$$Q(i,j) = P_1(i,j) \div P_2(i,j)$$

Diğer bir uygulama ise bir giriş resmi belli bir sabit değere bölünerek çıktı resminin oluşturulması sağlanır. Buna Piksel ölçeklemenin bir başka türünü verecektir. Renkli resimlerde ise her bir R, G, B kanalı ayrı ayrı sabit ile bölünür.

$$Q(i,j) = P_1(i,j) \div C$$

Uygulama: Bu konu ile uygulamaları kendiniz yaparak deneyin. Yukarıdaki örneklerde olduğu gibi sınırı geçen değerler için yada alt sınıra yakın bölgede biriken değerler için belli düzeltme faktörleri uygulayarak (ölçekleme yada normalizasyon yapılabilir) çıktı resmi daha görünür hale getirilebilir.

BLENDİNG (Görüntü Harmanlama)

Bu operatör, aynı boyutta iki giriş görüntüsünün bir karışımını belli oranda değerler ile çarparak çıkış görüntüsünü oluşturur. Görüntülerin çarpılacağı katsayılar kullanıcı tarafından belirlenir ve bunlar her görüntünün belli bir ölçekleme oranını oluşturur. Çıktı piksel değerleri maksimum sınırı aşmaması için çarpım katsayıları 1 den küçük ve toplamları bire eşit olacak şekilde belirlenir. Yani birinci resim 0,6 ile çarpılıyorsa, ikinci resim 0,4 ile çarpılmalıdır.

$$Q(i,j) = X * P_1(i,j) + (1 - X) * P_2(i,j)$$

Burada aslında iki resim belli ölçeklerle maskelenmiş olmaktadır. Gri resimlerde uygulandığı gibi, R,G,B banda yada daha fazla banda sahip resimlerde (multi spektral resimler) her banda ayrı ayrı uygulanır.

Görüntü karıştırmada resimler 1 altında ölçeklerle çarpıldığından ışık yoğunlu azalır (resim kararır). Bu nedenle hangi resim daha ön planda olması isteniyorsa onun katsayısı daha yüksek tutulabilir. Yada eklemeden sonra Normalizasyon yapılabilir. İki resmin belli oranlarda karıştırılması ile değişik sanatsal efektler oluşturulabilir.

Ek Bilgi: Normalizasyon (Kontras germe)

Kontrast uzatma (genellikle normalleştirme olarak adlandırılır), çıktı resminde oluşan piksel değerlerini alt ve üst limitlerinden tutarak belli bir aralığa lineer olarak (doğrusal olarak/orantı kurularak) çekme işlemidir. Sadece doğrusal ölçeklendirmeye izin verdiği için daha karmaşık olan "Histogram Eşitlemesinden" farklıdır. Çıktı resmi daha az serttir. Çoğu uygulama giriş resmi ve çıktı resmini Gri seviye resim olarak ister. Normalizasyon yaparken aşağıdaki formül kullanılabilir.

$$P_{out} = (P_{in} - c)\left(\frac{b-a}{d-c}\right) + a$$

Burada a ve b değerleri çıktı resminde oluşan sınırları (-30, 280 gibi sınırların dışına çıkmış olabilir), c ve d ise çekilmek istenen sınırları (örneğin 0-255 olabilir) gösterir.

Normalizasyondaki esas sorun çok yüksek veya çok düşük bir değere sahip tek bir pikselin değeri, çıktı c veya d değerini ciddi şekilde etkileyebilmesidir ve bu da çok temsilsiz ölçeklendirmeye yol açabilir. Bu nedenle, daha sağlam bir yaklaşım önce görüntünün bir **histogramını almak** ve sonrasında histogramdaki %5 ve %95 değerlerinden c ve d'yi seçmektir. Böylece aşırı uçlardaki değerler ihmal edilmiş olur. Bu durum, ölçeklendirmeyi çok etkileyen aykırı değerleri önler.

Aykırı değerlerle başa çıkmanın bir başka yolu ise, bir görüntüdeki en popüler yoğunluk seviyesini (yani histogram pikini) bulmak için yoğunluk histogramını kullanmaktır. Ortaya çıkan histogramda yığılmanın olduğu bölgenin altından ve üstünden alınan değerler ile c ve d değerleri tespit edilebilir.

Renkli görüntüler ile çalışılıyorsa, doğru renk oranlarını korumak için tüm kanallar aynı ofset ve ölçeklendirme kullanılarak gerilmelidir (normalize edilmelidir).

Programlama: Normalizasyon




Bu iki resmi 1.8 ve 1.2 katsayıları ile ölçekleyip toplarsak (Blending işlemi) üst değerler sınırı geçecektir. Bu değerler için üst sınırı geçenleri 255 e sabitlersek birinci görüntüyü elde ederiz. Üst sınırı Normalizasyon yaparak (tüm pikselleri orantısal olarak kaydırırsak) 255 ayarlarsak aşağıdaki ikinci görüntüyü elde ederiz.

```
private void blendingHarmanlamaToolStripMenuItem_Click(object sender, EventArgs e)
{
    Bitmap Resim1, Resim2, CikisResmi;
    Resim1 = new Bitmap(pictureBox1.Image);
    Resim2 = new Bitmap(pictureBox2.Image);

    int ResimGenisligi = Resim1.Width;
    int ResimYuksekligi = Resim1.Height;

    CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);

    Color Renk1, Renk2;
    int x, y;
    int R = 0, G = 0, B = 0;


    int EnBuyukDeger = 0, EnKucukDeger = 0;

    for (x = 0; x < ResimGenisligi; x++) //Resmi taramaya şablonun yarısı kadar dış kenarlardan
içeride başlayacak ve bitirecek.
    {
        for (y = 0; y < ResimYuksekligi; y++)
        {
            Renk1 = Resim1.GetPixel(x, y);
            Renk2 = Resim2.GetPixel(x, y);

            double Carpan1 = Convert.ToDouble(txtDeger1.Text);
            double Carpan2 = Convert.ToDouble(txtDeger2.Text);

            R = Convert.ToInt16(Renk1.R * Carpan1 + Renk2.R * Carpan2);


            //Sınırı aşan değerleri 255 ayarlama. Gri resim üzerinde işlem yapıldığı için Sadece
R ye bakıldı.
            if (R > EnBuyukDeger)
                EnBuyukDeger = R;

            if (R < EnKucukDeger)
                EnKucukDeger = R;
        }
    }

    Normalizasyon(Resim1, Resim2, EnBuyukDeger, EnKucukDeger);
}

//////////////////////////////////////////////////////////////
public void Normalizasyon(Bitmap Resim1, Bitmap Resim2, int EnBuyukDeger, int EnKucukDeger)
{
    Bitmap CikisResmi;
```

```
    int ResimGenisligi = Resim1.Width;
    int ResimYuksekligi = Resim1.Height;

    CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);

    Color Renk1, Renk2;
    int x, y;
    int R = 0, G = 0, B = 0;

      int UstSinir = 0, AltSinir = 0;

        if (EnBuyukDeger > 255)
            UstSinir = 255;
        else
            UstSinir = EnBuyukDeger;

        if (EnKucukDeger < 0)
            AltSinir = 0;
        else
            AltSinir = EnKucukDeger;

    for (x = 0; x < ResimGenisligi; x++) //Resmi taramaya şablonun yarısı kadar dış kenarlardan
içeride başlayacak ve bitirecek.
    {
        for (y = 0; y < ResimYuksekligi; y++)
        {
            Renk1 = Resim1.GetPixel(x, y);
            Renk2 = Resim2.GetPixel(x, y);

            double Carpan1 = Convert.ToDouble(txtDeger1.Text);
            double Carpan2 = Convert.ToDouble(txtDeger2.Text);

            R = Convert.ToInt16(Renk1.R * Carpan1 + Renk2.R * Carpan2);

            R = Convert.ToInt16(Renk1.R * Carpan1 + Renk2.R * Carpan2);

            int Deger = R; //Gri resim de tek değer kullanıldı.

            int NormalDegerR = (((UstSinir - AltSinir) * (Deger - EnKucukDeger)) / (EnBuyukDeger
- EnKucukDeger)) + AltSinir;


            CikisResmi.SetPixel(x, y, Color.FromArgb(NormalDegerR, NormalDegerR, NormalDegerR));

        }
    }
    pictureBox2.Image = CikisResmi;
}
```

Araştırma 2: Aşağıdaki konuları inceleyin. Uygulamaları programlayıp, dökümanları Türkçeleştirin.

http://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm (Fourier Transform)

http://homepages.inf.ed.ac.uk/rbf/HIPR2/pixlog.htm (Logarithm Operator)

http://homepages.inf.ed.ac.uk/rbf/HIPR2/histeq.htm (Histogram Equalization)

http://homepages.inf.ed.ac.uk/rbf/HIPR2/freqfilt.htm (Frequency Filter)

Ödev 1

a) İki tane resmi toplama işlemi yaptırın. Bu resimleri toplarken her resmi farklı ölçeklerle çarparak toplayın. Böylece ortaya çıkan birleşmiş resimi her iki resmi aynı orantıda göstermeye çalışsın. Örneğin aşağıda birinci resim koyu bir resim, ikinci daha açık bir resimdir. Bu iki resim toplanırken birinci resim daha büyük bir ölçekle çarpılarak

toplanmalı. Ortaya çıkan resim üzerinde tamamen beyazlama olmayacak şekilde ölçekleri ayarlanmalı. Ölçekleri ayarlamak için yan tarafta sürgüyü kullanarak manuel olarak yapınız.

b) Aynı uygulamayı otomatik ölçeklendirme yaparak yaptırın. (İpucu: Birinci resimdeki tüm renkleri toplasak, ortalamasını alsak -yani piksel sayısına bölsek, aslında renk düzeyinin ortalamasını bulmuş oluruz. Örneğin birinci resmin ortalaması (R, G, B hepsi için ortak bir sayı olmalı) 65 çıkmış olsun. İkinci resimde bu ortalama 130 çıkmış olursa, buradan ne anlarız, birinci resim daha koyu bir resimdir, ikinci resim ise açık bir resimdir. Bu iki resmi topladığımızda toplamları 255 geçmeli. Birinci resmi 2 le çarpsak diğerini de 1 le çarpsak yaklaşık olarak 260 gibi bir sayıyı tuttururuz. Buda iki resmi eşdeğer oranda havuza katıldığını gösterir. Buna benzer bir mantık kullanarak otomatik ölçeklendirme yaptırın.

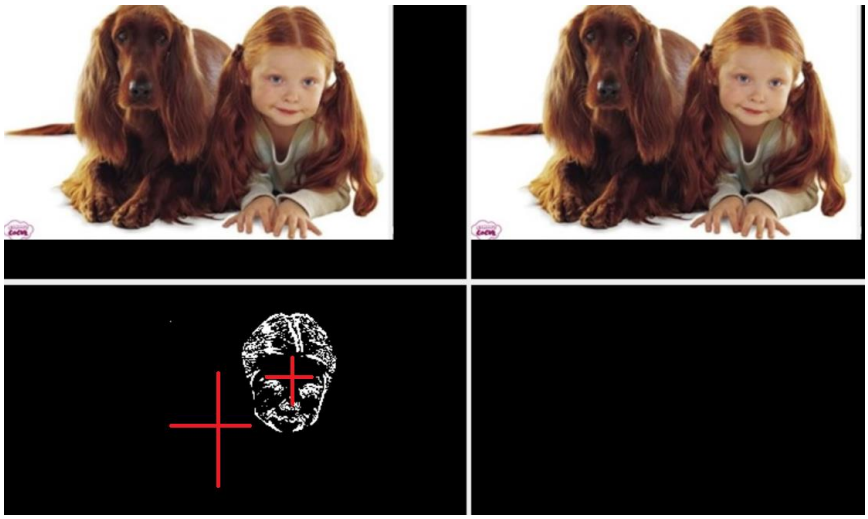
c) İki resmi kendi orijinal piksel değerleri ile toplayın. Dolayısı ile bazı piksellerin değeri 255 i geçecektir . Bu durumda ortaya çıkan resimde beyazlamış bölgeler olacaktır. İşte bu beyazlamış olan bölgeleri yok etmek Normalizasyon işlemi uygulayın.

Geliştirdiğiniz bu üç yöntemden hangisi daha uygun resim toplama işlemi yaptı. Yorumlayın.

Ödev 2: Bant üzerinde akan ürünlerin farklı ışık ortamlarında zeminden ayırma: Bant üzerinden akan ürünleri görüntü işleme ile tespit etmek istiyoruz. Bu ürünler YSA ya gönderilirken sadece siyah ve beyaz olmasını istiyoruz (2 renk olacak). Fabrikada üretim hattında ışık ortamı sürekli olarak değişmektedir. Gündüz ve akşam saatleri ile fabrikanın ışıklarını oluşturduğu farklı gölgelemeler ortaya çıkmaktadır. Bu gölgelemeler çok hızlı olmasa da yavaş bir şekilde değişmektedir. Böyle bir ortamda bant üzerindeki ürünleri doğru bir şekilde zeminden ayırabilmek için algoritmayı geliştirin.

Ödev için bu uygulamayı yaparken masa üzerine boş bir kâğıt koyun. Odanın ışıklarını kapatın. Yan taraftan mum ışığında (telefon ışığında) kâğıdın fotoğrafını üstten çekin. Kâğıt gölgeli bir fotoğraf olacaktır. Bu fotoğraf arka plan resmi olur. Sonra üzerine bir cisim koyun (Bir kaşık olabilir). Bu cismi arka plandan ayırırken eşikleme yaparak ve iki resmi birbirinden çıkararak uygulamayı gerçekleştirin. Hangisinde nesneyi daha iyi ayırdınız gösterin.

Ödev 3: Hareket sensör ile cismin merkezini yakalayıp, kamera eksenini (silah eksenini) ne kadar çevireceğini (yatayda ve dikeyde) hesaplayacak programı yazın).



Ödev 4: Hareket eden cisimlerde Ağaç ile insanı fark edebilecek bir algoritma geliştirin. Hareket cismin üzerine bir Artikel ekleyin (+ işareti) bu işaret hareket eden cismi takip etsin. Deney ortamını hazırlayın algoritmalarını yazın..

Ödev 5: Bir fabrika otomasyon sisteminde kamere çevre şartları içinde içinde içinde çalışsın. Çevredeki ışıklar değiştikçe bant sistemi üzerindeki yakalanan nesnenin görüntüsünün kalitesi değişmesin. Çevredeki ışıklar değişsin. Her değişimde hızlı bir şekilde cismi daha net bir şekilde ortamdan ayırabilsin. Deney ortamını oluşturun.

Ödev 6: Askeri uygulamalar için dürbünle bakarken Artikel, çekim saatleri, mesafe skalası gibi gerçek görüntüyü oluşturacak bir uygulama yapın. Video görüntüsü üzerinde bu uygulama nasıl yaparsınız araştırın.

ÖDEV 7: Aşağıdaki gibi bir üzerine daha ayarlanabilir bir yazıyı çapraz ve saydam olarak yazdıran komutları oluşturun.

Ödev 8: Görüntüyü zeminden ayırma

Gölgeli bir yazı görüntüsünü daha okunaklı hale getirmek için zemin görüntüsünden çıkarırsak daha iyi sonuç elde ederiz. Aşağıda bu uygulamanın aşamaları gözükmektedir. Fakat hala koyu bölgede yazılar kaybolmakta, açık bölgede ise yazıların arası dolmaktadır. Tam iyi bir sonuç elde edilememektedir.

Daha gelişmiş algoritmalar yazarak bu sorunu çözmeye çalışın. Yazı ve zemin görüntüsünü karanlık bir odada telefonun ışığı ile kendiniz elde edin.

Ödev 9: Foto kapan

a) Hareket algılama ile ilgili olarak aşağıdaki sahnelerden birinde Foto-kapan uygulaması yapın. Bunun için kamerayı sabitleyin (cep telefonu, web kamera, laptop kamerası vs)

Balkon yada cam kenarına bir yere kamerayı cadde yada gökyüzüne doğru sabitleyip, aralıklarla resim çekin (Her resmi alıp işledikten sonra tekrar resim alsın. Aralıklar ne kadar kısa olursa o kadar iyi olur). Bunun için sahne boş iken (tüm cisimler hareket etmemeli) önce sahne görüntüsünü alın. Ardından döngüyü çalıştırıp kameradan sürekli görüntü alın ve işleyin. Alınan görüntüler ile boş sahne görüntüsü arasında fark varsa, hareket var demektir ve hareket eden pikselleri yukarıda örnekte olduğu gibi belirleyebilirsiniz. Hareket eden piksellerin koordinatlarının bir ortalamasını alarak (x koordinatları kendi arasında toplanıp, kaç piksel ise ona bölünerek ve y eksenleri içinde aynısı yapılarak) hareket eden nesnenin sahnenin neresinde olduğunu bulun. Bu bölgenin üzerine bir daire yada Artikel (+) işareti kondurun. Böylelikle hedefi tespit eden bir uygulama yapmış olacaksınız.

b) Aynı ödevde hareket eden nesneler bulunduktan sonra bunları bir çerçeve içine alın. Ardından otomatik olarak hareket eden nesnenin olduğu bölgeyi zoomlasın.

Ödev 10: Arka plan görüntüsünü düzeltme.

a) İki resmi birbirinden çıkarak şöyle bir uygulama yapın. Gece vakti odanın ışığı kapalı iken Camdan dışarıdaki bir manzaranın fotoğrafını çekin (Dışarıda sokak yada karşıda evler vs çıkacaktır). Fotoğraf odanın ortasından bir yerden çekilmeli. Sonra içerinin ışığını açın, Camın iç kısmında odanın yansıması gözükecektir. Burada istenen Hem içerideki nesnelerin (cam ve etrafı) aydınlık görüntüsü olsun hem de camdan yansıma olmadan dışarının manzarası gözüksün. Buna benzer bir uygulama yapın. Burada hedef camdaki yansımayı kaldırmak. Ortamın ev içi vs olması gerekmiyor, özel alanlarda çekmeyin. Uygun bir deney ortamı kurarak yapmaya çalışın.

Ödev 11. Hareket algılama-foto kapan: İki resmi bir birinden çıkardığımızda ortaya çıkan resim bize hareket eden nesneleri gösterir. Fakat bu işlemde küçük gölgeler ve küçük nesnelerde (yaprakların oynaması gibi) hareket ediyor olarak gözükür. Böyle bir kamera uygulamasında insan ve araba gibi büyük nesnelerin hareket ettiğini algılayan, yaprak, sinek vs gibi küçük nesneleri hareket ediyor olarak algılamayacak bir algoritma geliştirin. Bulunan nesnelerin ağırlık merkezine bir (+) işaret kondurun.

Ödev 12: Gölgeli zeminden cismi ayırma: Masanın üzerine bir kağıt koyun. Kağıdın yan tarafına bir engel (Bardak, sürahi). Sürahinin arkasından ışık kağıdın üzerini aydınlatsın. (Oda karanlık olacak) Sürahinin gölgesi dalgalı bir şekilde kağıdın üzerine düşer. Bu gölgenin üzerine Kalem, silgi, Şeker (kağıdı ile), vs cisimler koyalım.

a) İlk olarak tepeden kağıdın sabit bir kamera ile (telefonu sabitleyerek) fotoğraf alalım. Bu fotoğrafın içinden cisimleri renkli bir şekilde ayıralım. Gölgeli arka plandan cisimleri zor olacaktır.
b) İkinci durumda bu sefer cisimler yokken kağıdın gölgeli görüntüsünü alalım. Bu görüntüden cisimler varken çekilen fotoğrafı çıkarırsak, cisimleri daha kolay tespit ederiz. Fakat tespit ettiğimiz yer aslında cismin silüetidir. Tam olarak kendi renginde göremeyiz. Eğer bu sülietleri olduğu görüntüyü arka plandan ayırırsak (cismin beyaz pikselleri olur) ve aynı koordinatları orijinal resmin (gölgeli kağıdın üzerindeki cisimlerin görüntüsü) üzerindeki piksellerden okursak artık aradığımız cismin renkli halini zeminden tamamen ayırarak ortaya çıkartırız. (Resim temsili konmuştur, konuyu tam anlatmaz)