GÖRÜNTÜ İŞLEME - (6.Hafta)

GÖRÜNTÜ NETLEŞTİRME ALGORİTMALARI

1. YÖNTEM: KENAR GÖRÜNTÜSÜNÜ KULLANARAK RESMİ NETLEŞTİRME

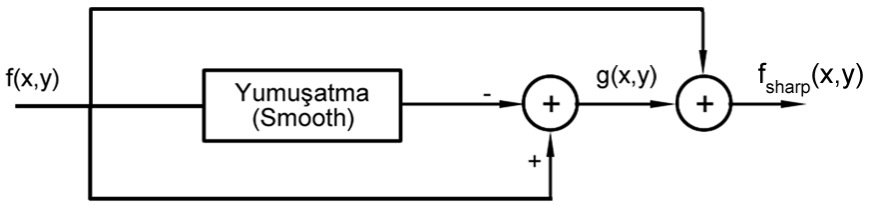
Bu algoritma orjinal görüntüden, görüntünün yumuşatılmış halini çıkararak belirgin kenarların görüntüsünü ortaya çıkarır. Daha sonra orjinal görüntü ile belirginleşmiş kenarların görüntüsünü birleştirerek, kenarları keskinleşmiş görüntüyü (netleşmiş görüntü) elde eder. Görüntünün netleştirilmesi fotogrametri ve baskı endüstrisinde yaygın olarak kullanılır.

g(x,y) çıkış görüntüsü, f(x,y) giriş görüntüsü ve $f_{smooth}(x,y)$ de bu görünün yumuşatılmış hali olmak üzere bu işlem aşağıdaki şekilde gösterilebilir.

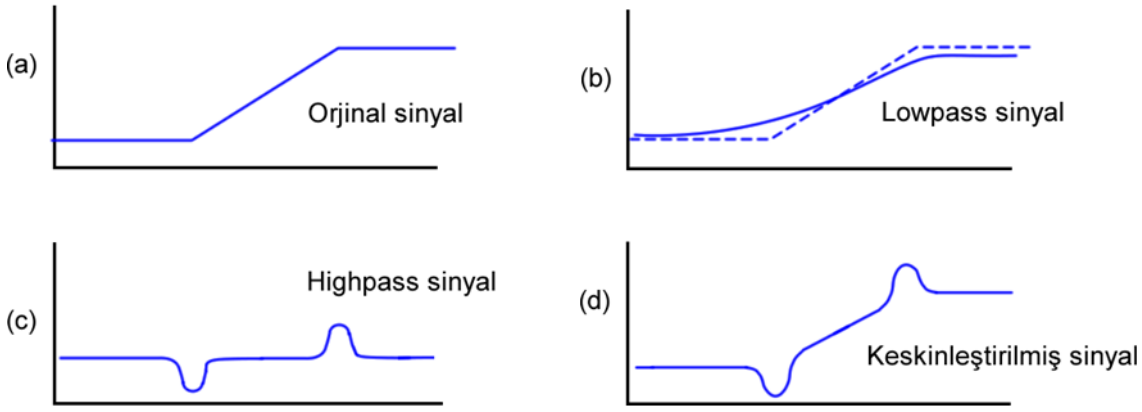
$$g(x,y) = f(x,y) - f_{smooth}(x,y)$$

$$f_{sharp}(x,y) = f(x,y) + k * g(x,y)$$

Burada k bir ölçekleme sabitidir. k için makul değerler 0,2-0,7 arasında değişir. k büyüdükçe keskinleştirme miktarı artar.



Filtrenin frekans cevabını inceleyerek işlemi daha iyi anlayabiliriz. Eğer bizim elimizde aşağıda Şekil a daki gibi bir sinyal varsa, bu sinyal Şekil b'de olduğu gibi lowpass filtreden geçirilirse, yani düşük frekanslar geçirilerek resmin yumuşatılması sağlanırsa, ortaya Şekil c'deki gibi highpass görüntü yani yüksek frekansların geçtiği keskin kenarların ortaya çıktığı bir görüntü elde edilir. Bu keskin kenarlar, orjinal görüntü ile birleştirilirse netleştirilmiş (kenarları keskinlşetirilmiş) görüntü oluşur (Şekil d).



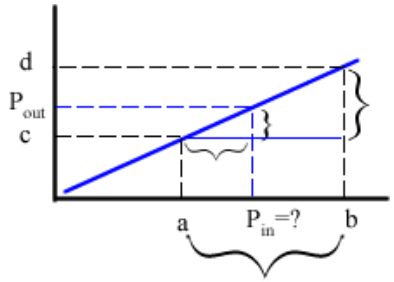
Şekil. Keskinleştirme filtresi için frekans cevabı ile kenar görüntüsü hesaplama

Keskinleştirme filtresi konvolüsyon işlemi ile yapılır. Yani çekirdek bir matris kullanılarak, resim alanı üzerinde gezdirilir ve gerekli çarpma işlemleri ile yeni resim elde edilir. Görüntünün yumuşatılmış versiyonunu oluşturmak için 3x3 lük Ortalama filtresi (mean filter) kullanılabilir. Ardından yumuşatılmış görüntü, orjinal

görüntüden piksel değerleri üzerinden çıkarılır. Bu esnada ortaya çıkan değerler negatif yada 255 i geçebilir. Bu durumda değerleri normalize etmek gerekir.

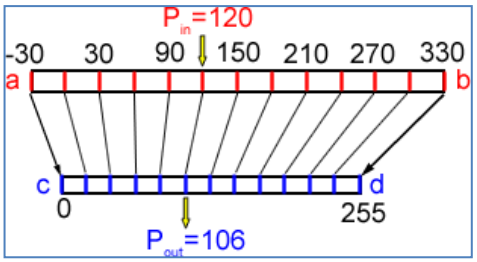
Değerleri normalize etme: Matematiksel işlemler sonunda görüntüdeki R,G,B değerleri 0-255 değerlerinin dışına çıkarsa tüm resmi aynı orantıya tabi tutarak değerleri yeniden normalize etmek gerekir.

Resim içerisindeki limit değerleri a,b ile, bu değerlerin olması gereken aralık olan 0,255 sayılarını da c,d ile, normalize edilecek değeri Pin, normalize olmuş halinide Pout ile gösterirsek, bu işlemi aşağıdaki şekilde formülüze edebiliriz. Dikkat edilirse bu formül verilen aralıklarda değerleri orantı kurarak bir gerdirme işlemidir. Aşağıdaki resimde üçgenlerdeki benzerliği kullanarak formülü bulabiliriz.



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

Örneklersek; Keskinleştirme işlemi yapıldıktan sonra resim içindeki üst ve alt sınırlar yani a ve b değerleri -30 ile 330 değerlerini alsın. Bu takdirde sınırlarımız ve aradaki herhangi bir 120 değeri şu şekilde hesaplanır.



$$P_{out} = (120-(-30))\left(\frac{255-0}{330-(-30)}\right)+0 = 106$$

Not: Burada normalleştirme öncesi üst ve alt sınırlardan biri (a ve b değerlerinden biri) 0 ve 255 değerlerini aşmaz ise (her ikiside aşmıyorsa zaten normalleştirmeye gerekte yoktur!), çıkış resmindeki düzeltilmiş üst ve alt sınırlar olan d ve c sayıları 0 ve 255 alınamaz. Bu durumda normalleşmiş resmindeki d ve c sayıları giriş resmindeki a ve b değerleri ile aynı alınmalıdır. Diğer türlü orijinal resimde hiç beyaz yada siyah bölge yokken, çıkış resminde bu bölgeler ortaya çıkmaya başlar. (dikkat! Netleştirilen resimde her iki uçta 0-255 aşmıyorsa zaten normalleştirmeye de ihtiyaç yoktur. Sınırlardan biri 0 yada 255 aşıyorsa ve diğeri aşmıyorsa, aşmayan taraf önceki değeri olarak alınır. Yani buradaki örnekte -30 yerine bu a değeri +10 olsaydı o zaman c değeri +10 alınmalı. Bu takdirde normalleştirme için 10-330 aralığının 10-255 aralığı olarak yapılmasını gerektirir.

Programlama (Görüntü Netleştirme)

Orjinal Resim

Mean 9x9 matris ile Bulanıklaştırma

Kenar Görüntüsü

Mean 9x9 matris ile netleşmiş Görüntü

Orjinal Resim

Gauss (5x5) matrisi ile netleştirilmiş görüntü

Burada orjinal resim daha başlangıçta bulanık ise, kullanılan matrisin boyutu artırılmalıdır. Gauss işlemi daha hassas bulanıklaştırma yaptığından etkisi, ince resimlerde daha iyi gözükecektir.

Not: Aşağıdaki kodlarda normalleştirme limit değerleri kullanarak yapılmadı. Basit düzeyde sınırı geçen değerler (0-255) sınırlarına çekildi. İlgili programlama (normalleştirme) ödevlerde istenmiştir.

```
      //NETLEŞTİRME-------------------------------
       private void netlestirmeToolStripMenuItem_Click_1(object sender, EventArgs e)
       {
          Bitmap OrjinalResim = new Bitmap(pictureBox1.Image);

          Bitmap BulanikResim = MeanFiltresi();
          //Bitmap BulanikResim = GaussFiltresi();

          Bitmap KenarGoruntusu = OrjianalResimdenBulanikResmiCikarma(OrjinalResim,
BulanikResim);
          Bitmap NetlesmisResim = KenarGoruntusuIleOrjinalResmiBirlestir(OrjinalResim,
KenarGoruntusu);

          pictureBox2.Image = NetlesmisResim;
       }

//*********************************************************************************************

       public Bitmap OrjianalResimdenBulanikResmiCikarma(Bitmap OrjinalResim, Bitmap
BulanikResim)
       {

          Color OkunanRenk1, OkunanRenk2, DonusenRenk;
          Bitmap CikisResmi;

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

          CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);
```

```
        int R, G, B;
        double Olcekleme = 1.7; //Keskin kenaları daha iyi görmek için değerini artırıyoruz.
        for (int x = 0; x < ResimGenisligi; x++)
        {
            for (int y = 0; y < ResimYuksekligi; y++)
            {
                OkunanRenk1 = OrjinalResim.GetPixel(x, y);
                OkunanRenk2 = BulanikResim.GetPixel(x, y);

                R =Convert.ToInt16(Olcekleme * (OkunanRenk1.R - OkunanRenk2.R));
                G = Convert.ToInt16(Olcekleme * (OkunanRenk1.G - OkunanRenk2.G));
                B = Convert.ToInt16(Olcekleme * (OkunanRenk1.B - OkunanRenk2.B));

                //================================================================
                //Renkler sınırların dışına çıktıysa, sınır değer alınacak. (Dikkat: Normalizasyon
yapılmamıştır. )
                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);

            }
        }

        return CikisResmi;
    }

//*********************************************************************************************

    public Bitmap KenarGoruntusuIleOrjinalResmiBirlestir(Bitmap OrjinalResim, Bitmap
KenarGoruntusu)
    {

        Color OkunanRenk1, OkunanRenk2, DonusenRenk;
        Bitmap CikisResmi;

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

        CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);


        int R, G, B;

        for (int x = 0; x < ResimGenisligi; x++)
        {
            for (int y = 0; y < ResimYuksekligi; y++)
            {
                OkunanRenk1 = OrjinalResim.GetPixel(x, y);
                OkunanRenk2 = KenarGoruntusu.GetPixel(x, y);

                R = OkunanRenk1.R + OkunanRenk2.R;
```

```
                G = OkunanRenk1.G + OkunanRenk2.G;
                B = OkunanRenk1.B + OkunanRenk2.B;

                //================================================================
                //Renkler sınırların dışına çıktıysa, sınır değer alınacak. //DİKKAT: Burada sınırı aşan değerler NORMALİZASYON yaparak programlanmalıdır.
                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);

            }
        }

        return CikisResmi;
    }
```

İKİNCİ YÖNTEM (Konvolüsyon yöntemi (çekirdek matris) ile netleştirme)

Aşağıdaki çekirdek matris kullanarak da Netleştirme yapılabilir. Bu matriste temel mantık üzerinde işlem yapılan pikselin kenarlarındaki 4 tane piksele bakar (Köşelere bakmıyor, sıfırla çarpıyor, onları toplama katmıyor) bu piksellerin değerini aşağıya indirir, üzerinde bulunduğu pikselin değerini yukarı çıkarır. Eğer her tarafı aynı olan bir bölgede ise sonuç değişmez. Fakat bir sınır bölgesine gelirse üzerine bastığın pikselin değerini yükseltir. Böylece sınır olan yerler daha parlak gözükür. Bu anlatılanları kendiniz örnek piksel değerleri deneyin. Matris değerleri ile piksel değerleri çarpılıp toplandıktan sonra matris toplamına da bölmek gerekir (yani buradaki 1/3 sayısı bunu anlatıyor).

$$\begin{bmatrix} 0 & -2 & 0 \\ -2 & 11 & -2 \\ 0 & -2 & 0 \end{bmatrix} \times \frac{1}{3}$$


```
        private void netlestirme2ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Color OkunanRenk;
            Bitmap GirisResmi, CikisResmi;
            GirisResmi = new Bitmap(pictureBox1.Image);

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

            CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);

            int SablonBoyutu = 3;
            int ElemanSayisi = SablonBoyutu * SablonBoyutu;


            int x, y, i, j, toplamR, toplamG, toplamB;

            int R, G, B;
            int[] Matris = { 0, -2, 0, -2, 11, -2, 0,-2,0};
            int MatrisToplami = 0 + -2 + 0 + -2 + 11 + -2 + 0 + -2 +0;

            for (x = (SablonBoyutu - 1) / 2; x < ResimGenisligi - (SablonBoyutu - 1) / 2; x++) //Resmi
taramaya şablonun yarısı kadar dış kenarlardan içeride başlayacak ve bitirecek.
            {
                for (y = (SablonBoyutu - 1) / 2; y < ResimYuksekligi - (SablonBoyutu - 1) / 2; y++)
                {
                    toplamR = 0;
                    toplamG = 0;
                    toplamB = 0;

                    //Şablon bölgesi (çekirdek matris) içindeki pikselleri tarıyor.
                    int k = 0; //matris içindeki elemanları sırayla okurken kullanılacak.
                    for (i = -((SablonBoyutu - 1) / 2); i <= (SablonBoyutu - 1) / 2; i++)
                    {
                        for (j = -((SablonBoyutu - 1) / 2); j <= (SablonBoyutu - 1) / 2; j++)
```

```
            {
                OkunanRenk = GirisResmi.GetPixel(x + i, y + j);

                toplamR = toplamR + OkunanRenk.R * Matris[k];
                toplamG = toplamG + OkunanRenk.G * Matris[k];
                toplamB = toplamB + OkunanRenk.B * Matris[k];


                k++;
            }

        }
            R = toplamR / MatrisToplami;
            G = toplamG / MatrisToplami;
            B = toplamB / MatrisToplami;

            //============================================================
            //Renkler sınırların dışına çıktıysa, sınır değer 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;
             //============================================================

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


        }
    }
    pictureBox2.Image =  CikisResmi;

}
```

Normalleştirme Yapılmış Kodlar

Orijinal Resim

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

     int ResimGenisligi = GirisResmi.Width;
```

```
            int ResimYuksekligi = GirisResmi.Height;

            CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);

            int SablonBoyutu = 3;
            int ElemanSayisi = SablonBoyutu * SablonBoyutu;


            int x, y, i, j, toplamR, toplamG, toplamB;

            int R, G, B;
            int[] Matris = { 0, -2, 0, -2, 11, -2, 0, -2, 0 };
            int MatrisToplami = 3;

            int EnBuyukR = 0;
            int EnBuyukG = 0;
            int EnBuyukB = 0;

            int EnKucukR = 0;
            int EnKucukG = 0;
            int EnKucukB = 0;

            for (x = (SablonBoyutu - 1) / 2; x < ResimGenisligi - (SablonBoyutu - 1) / 2; x++)
//Resmi taramaya şablonun yarısı kadar dış kenarlardan içeride başlayacak ve bitirecek.
            {
                for (y = (SablonBoyutu - 1) / 2; y < ResimYuksekligi - (SablonBoyutu - 1) / 2;
y++)
                {
                    toplamR = 0;
                    toplamG = 0;
                    toplamB = 0;

                    //Şablon bölgesi (çekirdek matris) içindeki pikselleri tarıyor.
                    int k = 0; //matris içindeki elemanları sırayla okurken kullanılacak.
                    for (i = -((SablonBoyutu - 1) / 2); i <= (SablonBoyutu - 1) / 2; i++)
                    {
                        for (j = -((SablonBoyutu - 1) / 2); j <= (SablonBoyutu - 1) / 2; j++)
                        {
                            OkunanRenk = GirisResmi.GetPixel(x + i, y + j);

                            toplamR = toplamR + OkunanRenk.R * Matris[k];
                            toplamG = toplamG + OkunanRenk.G * Matris[k];
                            toplamB = toplamB + OkunanRenk.B * Matris[k];


                            k++;
                        }

                    }
                    R = toplamR / MatrisToplami;
                    G = toplamG / MatrisToplami;
                    B = toplamB / MatrisToplami;

                    //============================================================
                    //Renkler sınırların dışına çıktıysa, sınır değer alınacak.


                    if (R > 255)
                    {
                        if (EnBuyukR < R)
                            EnBuyukR = R;
                        R = 255;

                    }
                    if (G > 255)
                    {
```

```
                    if (EnBuyukG < G)
                        EnBuyukG = G;
                    G = 255;
                }


                if (B > 255)
                {
                    if (EnBuyukB < B)
                        EnBuyukB = B;
                    B = 255;
                }

                //------------------------
                if (R < 0)
                {
                    if (EnKucukR > R)
                        EnKucukR = R;
                    R = 0;

                }
                if (G < 0)
                {
                    if (EnKucukG > G)
                        EnKucukG = G;
                    G = 0;
                }


                if (B < 0)
                {
                    if (EnKucukB > B)
                        EnKucukB = B;
                    B = 0;
                }
                //============================================================

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


            }
        }
        //txtUygulama.Text = EnBuyukR.ToString();
        //txtDeger1.Text = EnBuyukG.ToString();
        //txtDeger2.Text = EnBuyukB.ToString();

        txtUygulama.Text = EnKucukR.ToString();
        txtDeger1.Text = EnKucukG.ToString();
        txtDeger2.Text = EnKucukB.ToString();


        pictureBox2.Image = CikisResmi;


        ResmiNormallestir(GirisResmi, SablonBoyutu, ResimGenisligi, ResimYuksekligi,
EnBuyukR, EnBuyukG, EnBuyukB, EnKucukR, EnKucukG, EnKucukB );

    }


    public void ResmiNormallestir(Bitmap GirisResmi, int SablonBoyutu, int ResimGenisligi,
int ResimYuksekligi, int EnBuyukR, int EnBuyukG, int EnBuyukB, int EnKucukR, int EnKucukG, int
EnKucukB)
    {
        int toplamR = 0;
        int toplamG = 0;
```

```
            int toplamB = 0;

            Bitmap CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);

            int x, y, i, j;

            int R, G, B;
            int[] Matris = { 0, -2, 0, -2, 11, -2, 0, -2, 0 };
            int MatrisToplami = 3;

            for (x = (SablonBoyutu - 1) / 2; x < ResimGenisligi - (SablonBoyutu - 1) / 2; x++)
//Resmi taramaya şablonun yarısı kadar dış kenarlardan içeride başlayacak ve bitirecek.
            {
                for (y = (SablonBoyutu - 1) / 2; y < ResimYuksekligi - (SablonBoyutu - 1) / 2;
y++)
                {
                    toplamR = 0;
                    toplamG = 0;
                    toplamB = 0;

                    //Şablon bölgesi (çekirdek matris) içindeki pikselleri tarıyor.
                    int k = 0; //matris içindeki elemanları sırayla okurken kullanılacak.
                    for (i = -((SablonBoyutu - 1) / 2); i <= (SablonBoyutu - 1) / 2; i++)
                    {
                        for (j = -((SablonBoyutu - 1) / 2); j <= (SablonBoyutu - 1) / 2; j++)
                        {
                            Color OkunanRenk = GirisResmi.GetPixel(x + i, y + j);

                            toplamR = toplamR + OkunanRenk.R * Matris[k];
                            toplamG = toplamG + OkunanRenk.G * Matris[k];
                            toplamB = toplamB + OkunanRenk.B * Matris[k];


                            k++;
                        }

                    }
                    R = toplamR / MatrisToplami;
                    G = toplamG / MatrisToplami;
                    B = toplamB / MatrisToplami;

                    //NORMALİZASYON------------------------
                    int YeniR = ((255 - 0) * (R - EnKucukR)) / (EnBuyukR - EnKucukR) + 0;
                    int YeniG = ((255 - 0) * (G - EnKucukG)) / (EnBuyukG - EnKucukG) + 0;
                    int YeniB = ((255 - 0) * (B - EnKucukB)) / (EnBuyukB - EnKucukB) + 0;

                    //============================================================

                    CikisResmi.SetPixel(x, y, Color.FromArgb(YeniR, YeniG, YeniB));


                }
            }


        }
```

Ödev 1: Netleştirme işleminde resmi bulanıklaştırırken kullanılan buradaki Mean yönteminin dışında Median ve Gauss yöntemlerini kullanarakda netleştirme yapın.

Ödev 2: Bulanık bir resmi netleştirirken daha büyük ölçekli matrislere ihtiyaç olacaktır. Böyle bir resim üzerinde kullanılan matris boyutunu aşama aşama artırarak resmin netliğinde nasıl bir etkisi olduğunu gösterin.

Ödev 3: Aşağıdaki matrislerden hangisi kenar bulma matrisi, hangisi netleştirme matrisidir? Programlayarak deneyin. Deneme öncesi kağıt üzerinde kendiniz örnek değerler vererek bulmaya çalışın.

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

0	-2	0
-2	18	-2
0	-2	0

1	-2	1
-2	4	-2
1	-2	1

-1	-1	-1
-1	8	-1
-1	-1	-1

Ödev 4: Yukarıdaki tüm uygulamaları Normalizasyon yaparak düzenleyin. Eğer sınırlar 0-255 dışına çıkıyorsa ve elimizdeki resimde hiç beyaz yada siyah yoksa yani (30 – 230) gibi bir değerde ise Dışarı çıkan bu sınırlar resmin en üç noktaları arasında normalize edilmelidir.