

GÖRÜNTÜ İŞLEME - (6.Hafta)

GÖRÜNTÜ NETLEŞTİRME ALGORİTMALARI

1. YÖNTEM: KENAR GÖRÜNTÜSÜNÜ KULLANARAK RESMİ NETLEŞTİRME

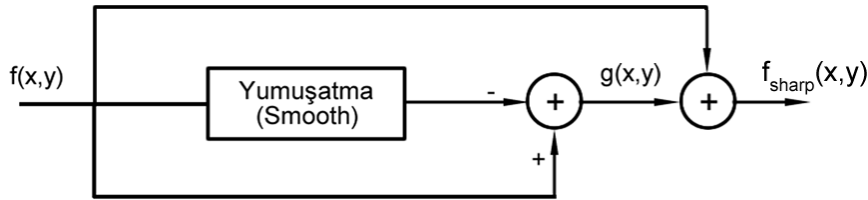
Bu algoritma orjinal görüntüden, görüntünün yumuşatılmış halini çıkararak belirgin kenarların görüntüsünü ortaya çıkarır. Daha sonra orjinal görüntü ile belirginleşmiş kenarların görüntüsünü birleştirerek, kenarları keskinleşmiş görüntüyü (netleşmiş görüntü) elde eder. Görüntünün netleştirilmesi fotogrametri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

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

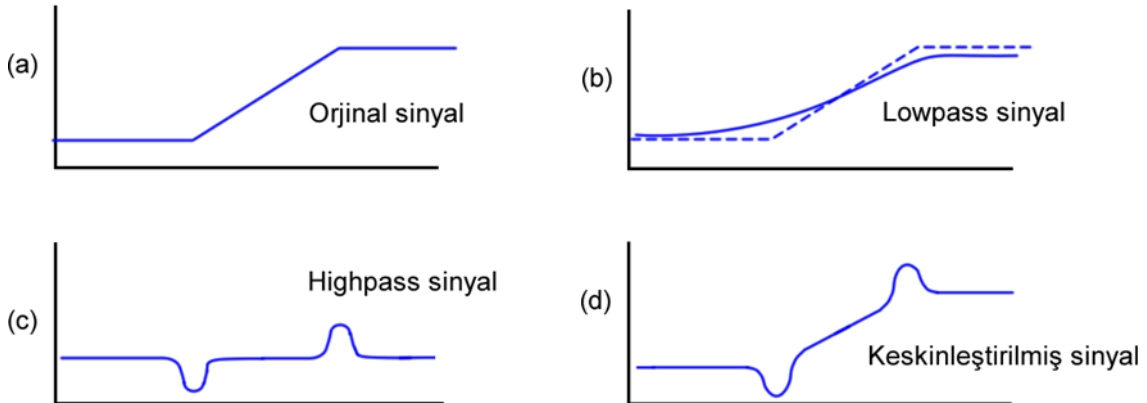
$$g(x,y) = f(x,y) - f_{smooth}(x,y)$$

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

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 keskinleştirilmiş) görüntü oluşur (Şekil d).



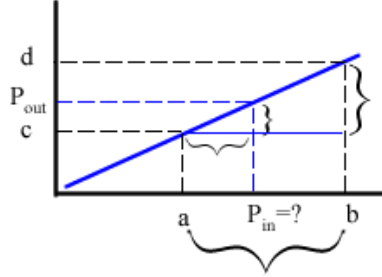
Şekil. Keskinleştirme filtresi için frekans cevabı ile kenar görüntüsü hesaplama

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.

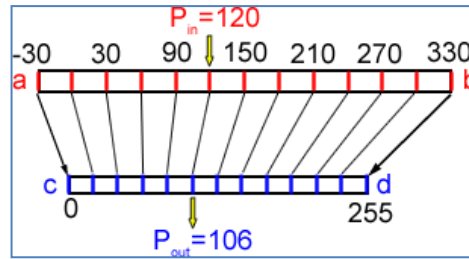
Değerleri normalize etme: Matematiksel işlemler sonunda görüntüdeki R,G,B değerleri 0-255 değerlerinin dışına çıkarsa tüm resmi aynı orantıya tabi tutarak değerleri yeniden normalize etmek gerekir.

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



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

Örneklesek; 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 gerek yoktur!), çıkış resmindeki düzeltilmiş üst ve alt sınırlar olan d ve c sayıları 0 ve 255 alınmaz. 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 aşmıyorsa 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.

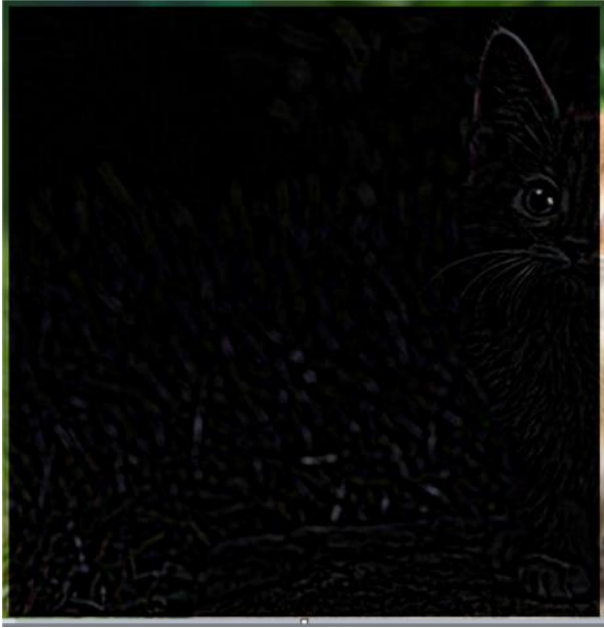
Programlama (Görüntü Netleştirme)



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 gözükcektir.

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 = OrjinalResimdenBulanikResmiCikarma(OrjinalResim,
BulanikResim);
    Bitmap NetlesmisResim = KenarGoruntusulleOrjinalResmiBirlestir(OrjinalResim,
KenarGoruntusu);

    pictureBox2.Image = NetlesmisResim;
}

//*****

public Bitmap OrjinalResimdenBulanikResmiCikarma(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 = 2; //Keskin kenarları 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 * Math.Abs(OkunanRenk1.R - OkunanRenk2.R));
        G = Convert.ToInt16(Olcekleme * Math.Abs(OkunanRenk1.G - OkunanRenk2.G));
        B = Convert.ToInt16(Olcekleme * Math.Abs(OkunanRenk1.B - OkunanRenk2.B));

        //=====
        //Renkler sınırların dışına çıktıysa, sınır değeri 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 KenarGoruntusulleOrjinalResmiBirlestir(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 değeri 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 gözüktür. 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).

0	-2	0
-2	11	-2
0	-2	0

Kullanım şekli;

$$\begin{array}{|c|c|c|} \hline 1 & 1 & 5 \\ \hline 1 & 5 & 5 \\ \hline 5 & 5 & 5 \\ \hline \end{array} \times \begin{array}{|c|c|c|} \hline 0 & -2 & 0 \\ \hline -2 & 11 & -2 \\ \hline 0 & -2 & 0 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 0 & -2 & 0 \\ \hline -2 & 55 & -10 \\ \hline 0 & -10 & 0 \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|} \hline 1 & 1 & 5 \\ \hline 1 & 5 & 5 \\ \hline 5 & 5 & 5 \\ \hline \end{array} + \begin{array}{|c|c|c|} \hline 0 & -2 & 0 \\ \hline -2 & 11 & -2 \\ \hline 0 & -2 & 0 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 0 & -2 & 0 \\ \hline -2 & 55 & -10 \\ \hline 0 & -10 & 0 \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|} \hline 0 & -2 & 0 \\ \hline -2 & 55 & -10 \\ \hline 0 & -10 & 0 \\ \hline \end{array} \div 3 = \begin{array}{|c|c|c|} \hline 0 & -2 & 0 \\ \hline -2 & 18.33 & -3.33 \\ \hline 0 & -3.33 & 0 \\ \hline \end{array}$$



```
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 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;
//=====

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 = GirişResmi.Width;
int ResimYuksekligi = GirişResmi.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 = GirişResmi.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 değeri 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));

    }
}

txtX1.Text = EnBuyukR.ToString();
txtX2.Text = EnBuyukG.ToString();
txtX3.Text = EnBuyukB.ToString();

txtY1.Text = EnKucukR.ToString();
txtY2.Text = EnKucukG.ToString();
txtY3.Text = EnKucukB.ToString();

int EnBuyuk=0, EnKucuk=0;

if (EnBuyukR > EnBuyuk)
    EnBuyuk = EnBuyukR;

if (EnBuyukG > EnBuyuk)
    EnBuyuk = EnBuyukG;

if (EnBuyukB > EnBuyuk)
    EnBuyuk = EnBuyukB;

if (EnKucukR > EnKucuk)
    EnKucuk = EnKucukR;

```

```

        if (EnKucukG > EnKucuk)
            EnKucuk = EnKucukG;

```

```

        if (EnKucukB > EnKucuk)
            EnKucuk = EnKucukB;

```

```

        Bitmap NormallesmisNetResim = ResmiNormallestir(GirisResmi, SablonBoyutu,
ResimGenisligi, ResimYuksekligi, EnBuyuk, EnKucuk);

```

```

        pictureBox4.Image = NormallesmisNetResim;    }

```

```

        public Bitmap ResmiNormallestir(Bitmap GirisResmi, int SablonBoyutu, int ResimGenisligi,
int ResimYuksekligi, int EnBuyuk, int EnKucuk)
        {
            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 * (R - EnKucuk)) / (EnBuyuk - EnKucuk) ;
                    int YeniG = (255 * (G - EnKucuk)) / (EnBuyuk - EnKucuk) ;
                    int YeniB = (255 * (B - EnKucuk)) / (EnBuyuk - EnKucuk) ;

```

```

        if (YeniR > 255) YeniR = 255;
        if (YeniG > 255) YeniG = 255;
        if (YeniB > 255) YeniB = 255;

        if (YeniR < 0) YeniR = 0;
        if (YeniG < 0) YeniG = 0;
        if (YeniB < 0) YeniB = 0;
        //=====

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

    }
}

return CikisResmi;
}

```

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 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 uç noktaları arasında normalize edilmelidir.

Ödev 5: Bir resmin üzerindeki bir bölgeyi seçmek için köşe noktalarına mouse tıklayalım. Tıklaştığımız her noktanın arasına bir çizgi çizsin. Tüm bölgeyi kapattıktan sonra, çerçevenin iç bölgesinde netleştirme uygulasin, dış bölgesinde ise Bulanıklaştırma uygulasin. Hangi algoritma ile netleştirme, hangi ile bulanıklaştırma uygulayacağımı ise yan taraftaki menüden kişi seçebilsin. Bunun için RadioButton seçeneği kullanılabilir.

Çoklu nokta şeklinde yapamayanlar, dikdörtgen çerçeve şeklinde bölgeyi seçsinler. Bu şekilde yapanlar daha düşük puan alır.

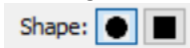


Ödev 6: Sharpen (netleştirme) aracı kullanarak bir resim üzerinde mouse takip eden bulur işlemi yaptırın. Bunun için aşağıdaki adımları takip edin.

- Mouse ın etrafında büyüklüğü ayarlanabilir belli bir bölge olsun. Örneğin açılır menüden 50 piksel seçilirse mouse etrafında 50x50 lik kare bir şekilde dolaşsın. Bu alanın içi netleşsin.



- Kişi ister bu şekli Kare yada Daire olarak değiştirebilsin. Bu bölge Kare yada Daire olarak seçilebilsin. Bu bölgenin içerisi mouse hareket ettikçe, bulanıklaşsın.



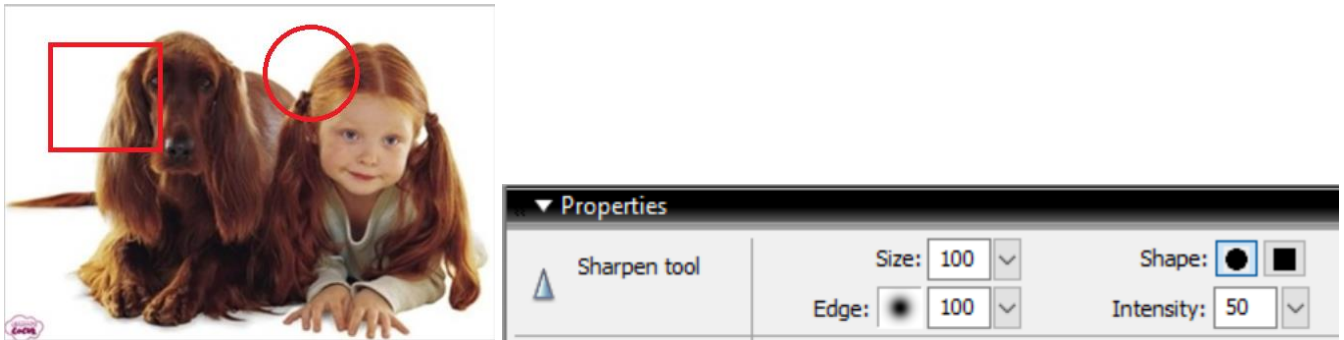
- c) Ayrıca bu aracın kenarları üzerinde geçişli bir netlik oluşturmak için bir ayar daha eklensin. Örneğin burada %100 seçilirse Dairenin merkezinden kenara kadar yavaş yavaş netlik azalarak gitsin. %50 seçilirse dairenin yarısından sonra yavaş yavaş netlik azalsın.



- d) Netliğin yoğunluğunun ayarlanabildiği bir seçenek daha ekleyin. Aşağıda Intensity olarak gösterilmiş.



- e) Kullanılan algoritma açılır menüden, yada radyo buton listesinden seçilebilsin. Örneğin; Kenar bulma algoritması yada Matris algoritması gibi.



Ödev 7: Aşağıdaki gibi bir resmin üzerinde belirlenen bölgenin iç kısmında netleştirme, dış kısmında bulanıklaştırmayı otomatik olarak yapan kodları geliştirin. Bunu gerçekleştiremeyenlerde dikdörtgen şeklinde bir çerçevenin içerisine ve dışına algoritmaları uygulasin. Hangi algoritmaların uygulanacağı menülerden seçilebilsin. Ayrıca iç ve dış tarafın bulanık/net olması değiştirilebilsin. Belirlenen bölgenin kenarlarına bir çizgi çizdirirseniz daha anlaşılır olacaktır.

