GÖRÜNTÜ İŞLEME-10.HAFTA

MANTIKSAL OPERATÖRLER

Mantıksal operatörler, resmin gerçek değerlerine dokunmadan, boolean ifadeler (true yada false) kullanılarak yeni bilgiler veya görüntüler çıkarmak için kullanılır. Örneğin "Ağaç hem yeşil hemde büyük ise" ifadesi A AND B şeklinde ifade edilebilir. Bu durumda her iki kavramda doğru (true) ise sonuç True olacaktır ve ona göre işlem yapılacaktır. Bu şekilde girilen iki tane görüntü üzerinde yada tek bir görüntü üzerinde, piksel değerlerine mantıksal operatörlerin doğruluk tablosu kuralları uygulanarak görüntüler üzerinde mantıksal işlemler gerçekleştirebiliriz. Normalde, aynı boyutta iki görüntüden çıktı görüntüsünü üretmek için, giriş görüntüsülerine ait birbirine karşılık gelen pikseller karşılaştırılır. Tek bir giriş görüntüsünü sabit bir mantıksal değerle mantıksal olarak birleştirmek de mümkündür. Bu durumda giriş görüntüsündeki her piksel karşılık gelen çıkış pikselini üretmek için aynı sabit ile karşılaştırılır.

Mantıksal işlemler, tamsayı piksel değerlerine sahip görüntüler üzerinde, bu tamsayıların ikili gösterimleri (binary) yapılarak bunlar üzerinde bitsel bir şekilde işlem yapılarak, çıkış piksel değerine bitsel sonuçlar üretilip buradan tam sayı değerine ulaşılabilir. Örneğin, 47 ve 255 tam sayılarını 8 bitlik tam sayıları kullanarak birlikte XOR yapmak istediğimizi varsayalım. Bunları ikili biçimde göserirsek 45 sayısı 00101111 ve 255 sayısı 11111111'dir. Bunları bitsel olarak birlikte XORing yaparak, ikili olarak 11010000 değerine ulaşılır. Bunun karşılığı ise tam sayı olarak 208 dir.

Mantıksal işlemler illa böyle bitsel (1/0) olarak uygulanması gerekmez. Örneğin, bazı uygulamalarda sıfır değerleri false kabul edilirken, sıfır haricindeki herhangi bir değer true olarak kabul edilebilir. Çıktı görüntüsünü elde etmek için ise 1-bit mantıksal işlemleri uygulanır. Çıktı görüntüsü basit bir ikili (binary) görüntünün kendisi olabilir ya da belki de ikili (binar) çıktı görüntüsü (sadece siyah ve beyaz görüntü) giriş görüntülerinden hehangi biriyle çarpılarak gri seviye bir görüntü elde edilebilir..

İnteger sayıyı ikili (binary) sayıya 8 bit olarak dönüştürme


```
int intSayi1 = Convert.ToInt16(txtDeger1.Text);

string binarySayi = Convert.ToString(intSayi1, 2).PadLeft(8, '0');

txtDeger1.Text = binarySayi;

int intSayi2 = Convert.ToInt32(binarySayi, 2);

txtDeger2.Text = intSayi2.ToString();
```

A-Mantıksal AND/NAND Operatörleri

Mantıksal operatörler, Boolean değerler üzerinde (1 yada 0)(Doğru yada Yanlış) işlem yapılarak elde edilir. AND ve NAND işlem tablosu aşağıdaki gibidir.

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

AND

A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

NAND

Dikkat edilirse iki tablo birbirinin tam zıttıdır. Bu Operatörlerde Birinci resmi alıp, ikinci resimle AND veya NAND işlemine tabi tutularak çıktı resmi oluşturulur. Yada sadece tek bir Girdi resmi alınır ve belli bir sabit sayı ile AND yada NAND işlemine tabi tutulabilir. Tablolara dikkat edilirse bu iki operatör birbirinin tam tersidir.

AND ve NAND operatörü giriş olarak iki adet ikili (binary-siyah/beyaz resim) veya tamsayı ile ifade edilen gri seviye görüntüyü alır ve piksel değerleri üzerinde işlem yaparak ilk görüntünün değerlerini ikinci görüntü ile işlem yaparak üçüncü bir görüntü oluşturulur. Bu operatörün farklı bir uygulaması olarak tek bir giriş görüntüsü alınıp her pikseli sabit bir değer ile işleme tutarak çıktı görüntüsü oluşturulabilir.

İşlem tek bir geçişte doğrudan gerçekleştirilir. Üzerinde çalışılan tüm giriş pikseli değerlerinin aynı sayıda bite sahip olması önemlidir. Giriş görüntülerindeki piksel değerlerinin basit 1 bitlik sayılar olmadığı durumlarda, AND işlemi normalde (ancak her zaman değil) piksel değerlerindeki her bir bit üzerinde ayrı ayrı, bitsel şekilde gerçekleştirilir.

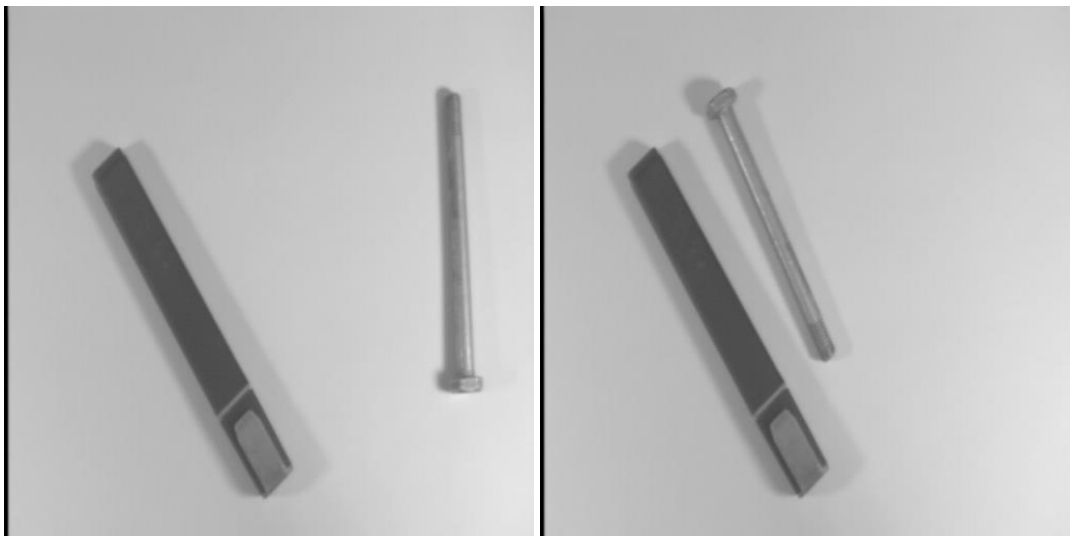
```
private void AND_NAND_IslemiToolStripMenuItem_Click(object sender, EventArgs e)
{
    Bitmap Resim1, Resim2, CikisResmi;
    Resim1 = new Bitmap(pictureBox1.Image);
    Resim2 = new Bitmap(pictureBox2.Image);

    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);

            string binarySayi1 = Convert.ToString(Renk1.R, 2).PadLeft(8, '0'); //Gri renk
olduğundan tek kanal üzerinden yapılıyor.
            string binarySayi2 = Convert.ToString(Renk2.R, 2).PadLeft(8, '0');

            string Bit1 = null, Bit2 = null, StringIkiliSayi = null;

            for (int i = 0; i < 8; i++)
            {
                Bit1 = binarySayi1.Substring(i, 1);
                Bit2 = binarySayi2.Substring(i, 1);
```

```
                ////AND İŞLEMİ
                //if (Bit1 == "0" && Bit2 == "0") StringIkiliSayi = StringIkiliSayi + "0";
                //else if (Bit1 == "1" && Bit2 == "1") StringIkiliSayi = StringIkiliSayi + "1";
                //else StringIkiliSayi = StringIkiliSayi + "0";

                //NAND İŞLEMİ
                if (Bit1 == "0" && Bit2 == "0") StringIkiliSayi = StringIkiliSayi + "1";
                else if (Bit1 == "1" && Bit2 == "1") StringIkiliSayi = StringIkiliSayi + "0";
                else StringIkiliSayi = StringIkiliSayi + "1";


            }
            R = Convert.ToInt32(StringIkiliSayi, 2); //İkili sayıyı tam sayıya dönüştürüyor.

            CikisResmi.SetPixel(x, y, Color.FromArgb(R, R, R)); //Gri resim

        }
    }
    pictureBox2.Image = CikisResmi;

}
```

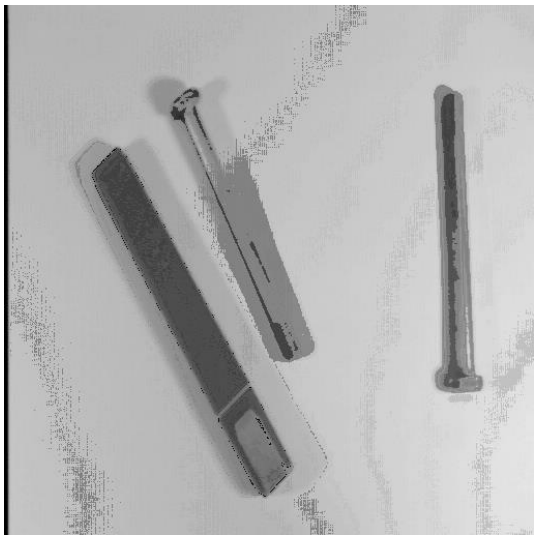
Uygulama 1

AND'in en belirgin uygulaması iki görüntünün kesişimini hesaplamaktır. Yani iki görüntü arasında hareket etmeyen, (birinci ve ikinci görüntüde aynı piksel konumlarında bulunan) nesneleri tespit etmek istediğimiz bir durumda kullanabilir.

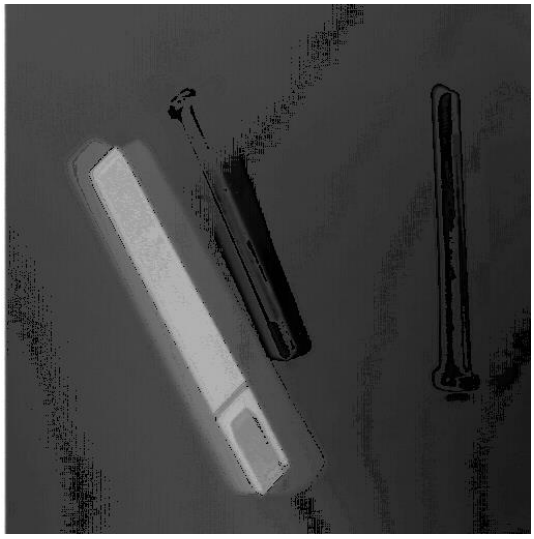
Aşağıdaki iki resmi kullanarak uygulamayı yapalım.



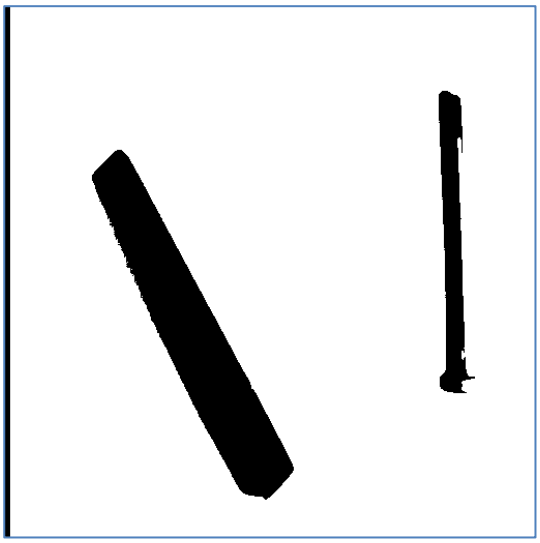
İki gray seviye görüntüye bitsel olarak basitçe "AND" uygularsak;

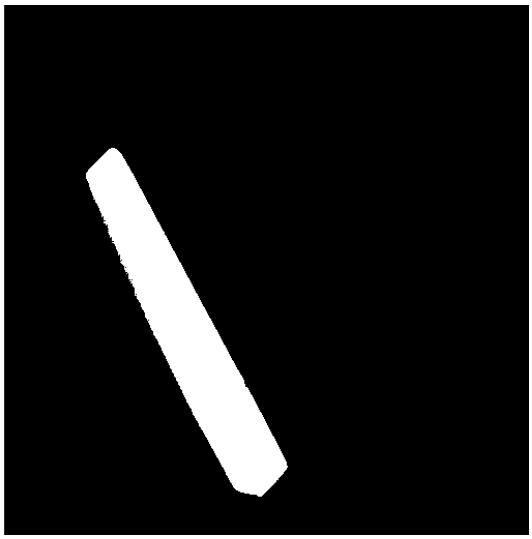


Bu uygulamada taşınan nesnenin ortaya çıkan görüntüden kaybolmasını istememize rağmen, eski ve yeni konumunda iki kez görünür olarak ortaya çıktı. Bunun nedeni, nesnenin oldukça düşük piksel değerlerine (mantıksal 0'a benzer) sahipken arka planın yüksek değerlere (mantıksal 1'e benzer) sahip olmasıdır. Bununla birlikte, normalde bir nesneyi mantıksal 1 ile ve arka planı mantıksal 0 ile ilişkilendiririz, bu nedenle aslında NOR görüntülerine eşdeğer olan iki görüntünün negatiflerini AND'ledik. İstenilen sonucu elde etmek için görüntüleri AND işlemlerinden önce tersine çevirmeliyiz.

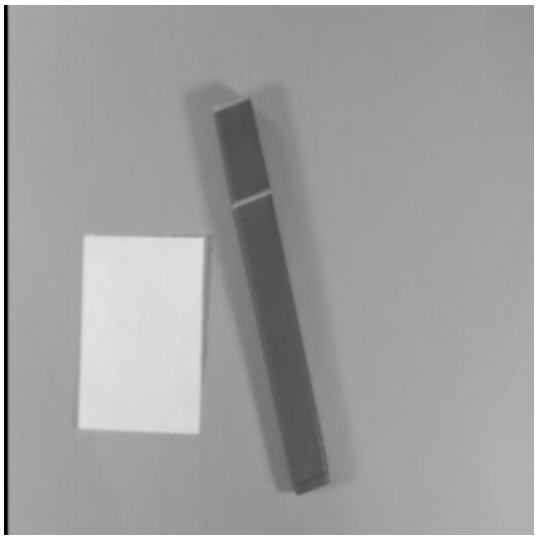


Şimdi, yalnızca her iki görüntüde de aynı konuma sahip olan nesne vurgulanır. Bununla birlikte, AND'in iki graylevel görüntüsünün bitmesi yine de sorunlara neden olabilir, çünkü iki yüksek piksel değerinin bitsel bir şekilde ANDing'inin yüksek bir çıkış değeri verdiği garanti edilmez (örneğin, 128 AND 127 0 değerini verir). Bu sorunları önlemek için, eşikleme kullanarak gri tonlamalı görüntülerden ikili (binary) sürümler üretmek en iyisidir. Aşağıdaki 2. Resim eşikleme yapılmış resimdir.

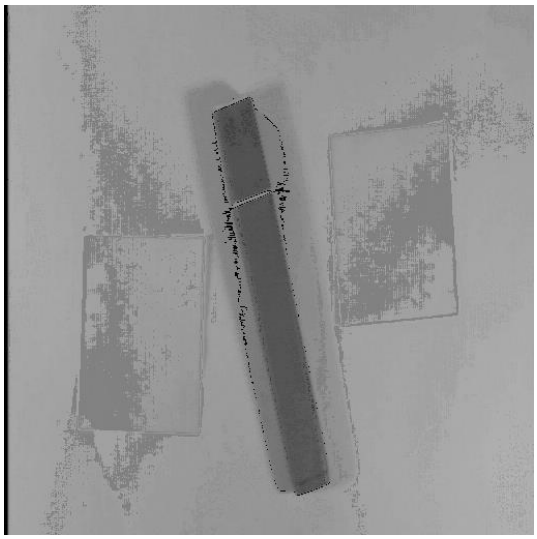




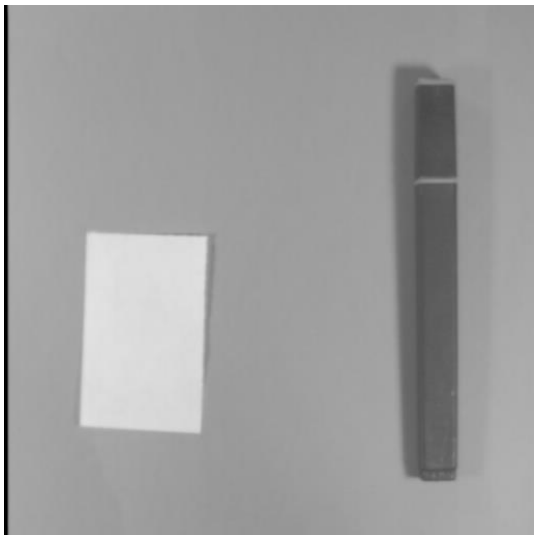
Her ne kadar AND işlemi yukarıdaki örnekte olduğu gibi iyi çalışsa da, o sahnede bazen şu şekilde problemlere neden olabilir.

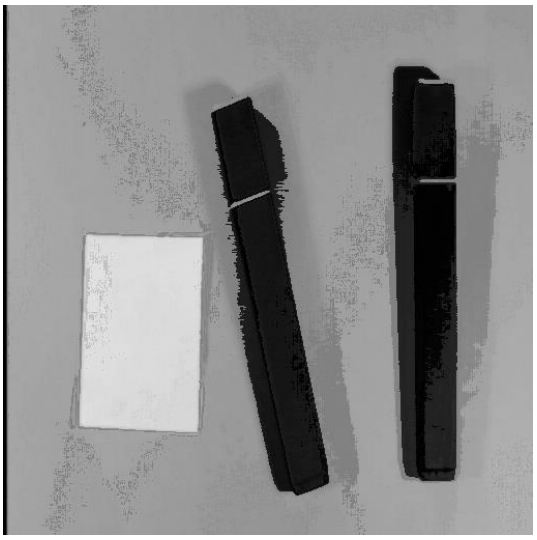


Burada, ortalama yoğunluğu arka plandan daha yüksek ve diğeri daha düşük olan iki nesnemiz var. Bu nedenle, basit eşikleme kullanarak her iki nesneyi içeren bir ikili görüntü oluşturamayız. Aşağıdaki resimlerde de görüldüğü gibi, AND gri tonlamalı görüntüleri almak da başarılı değildir. İkinci sahnede ışık kısmı,



Taşınan nesneyi zayıflatmanın istenen etkisini gösterir. Ancak, ikinci sahne bir şekilde şuna benzer ise ve siyah nesne hareket ettirilirse; ikinci resimdeki görüntü elde edilir. Bu resimde Burada, koyu nesnenin eski ve yeni konumları görülebilir.





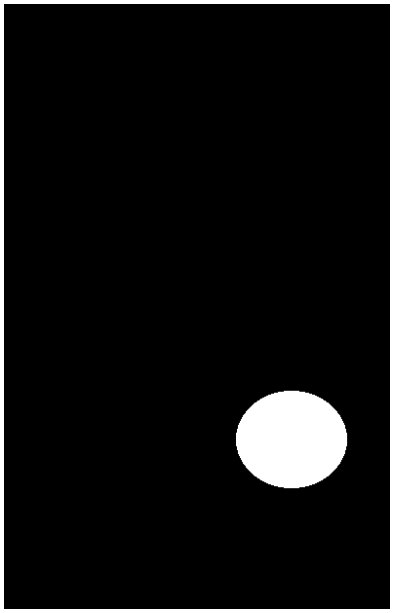
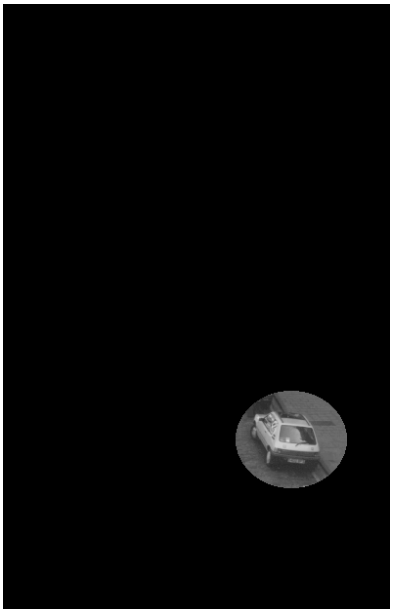
Genel olarak, aralarındaki farklılıkları veya benzerlikleri tespit etmek için AND operatörünün (veya diğer mantıksal operatörlerin) iki görüntüye uygulanması, ikili veya eşikleme kullanılarak ikili biçime dönüştürülebiliyorsa en uygunudur.

Diğer mantıksal işleçlerde olduğu gibi, AND ve NAND genellikle daha karmaşık görüntü işleme görevlerinin alt bileşenleri olarak kullanılır.

Uygulama 2

AND'in yaygın kullanımlarından biri maskeleme içindir. Örneğin, küçük bir bölgesini seçerek aydınlatmak istediğimizi varsayalım. Resimdeki belirli bir arabayı vurgulamak için.(Bunu yapmanın birçok yolu var ve biz sadece bir tanesini açıklıyoruz), önce vurgulanacak bölgeye bir boyama işlemi uygulanır (Paint programı ile yapılabilir). Burada gösterildiği gibi siyaha ayarladık.


Bu görüntü daha sonra yalnızca siyah bölgeyi seçmek için eşikleme yapılarak o bölgenin beyaz görünmesi sağlanır. Maske görüntüsü, ilgilendiğimiz bölgede 255 (11111111 binary) piksel değerine ve başka yerlerde sıfır piksel (00000000 binary) değerine sahiptir. Elde edilen maske daha sonra orijinal görüntü ile bitsel olarak AND işlemine tabi tutulursa, vurgulanacak bölge ortaya çıkarılmış olur. Son olarak, aracın olduğu bu görüntüyü 1,1 faktörle ölçeklendirerek aydınlatırız. Ardından orijinal görüntüyü ise 0,8 ölçek faktörü kullanarak karartırsak ve iki görüntüyü birleştirirsek aşağıdaki resmi elde ederiz.


Ödev 1: Benzer bir uygulama dış kısımlar bulanıklaştırılarak yada hedef bölgesi büyütülerek uygulanabilir.

Uygulama 3

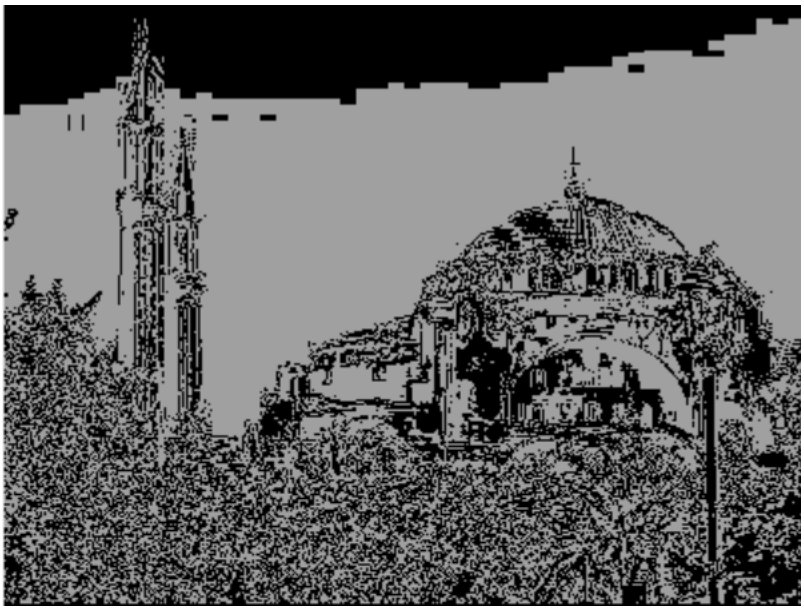
AND operatörü, 8 bitlik bir görüntüde “bit dilimleme” adı verilen işlemi gerçekleştirmek için de kullanılabilir. Belirli bir bitin görüntü üzerindeki etkisini belirlemek için, ilgili bitin 1'e ve geri kalan 7 bitin 0'a ayarlanır. sabit bir sayı ile bitsel bir şekilde AND işlemi uygulanır. Aşağıdaki örnekleri inceleyin. Ölçeklemeler resmi parlatmak için kullanılmıştır.

Orijinal resim ve 1 kat ölçeklenmiş hali.

0 bit 1x ölçekleme

1. bit 1x ölçekleme ve 3x ölçekleme

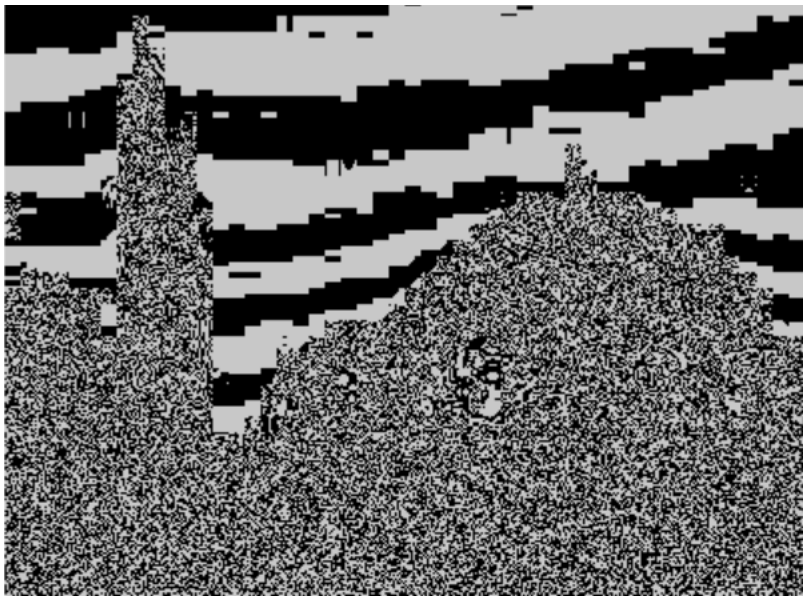
2. bit 5x öçekleme



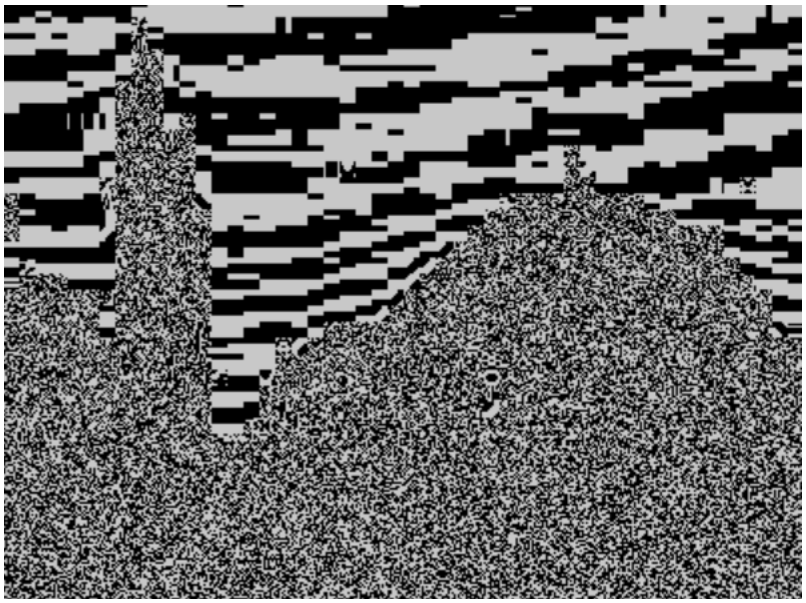
3 Bit 10x ölçekleme

4 bit 20x

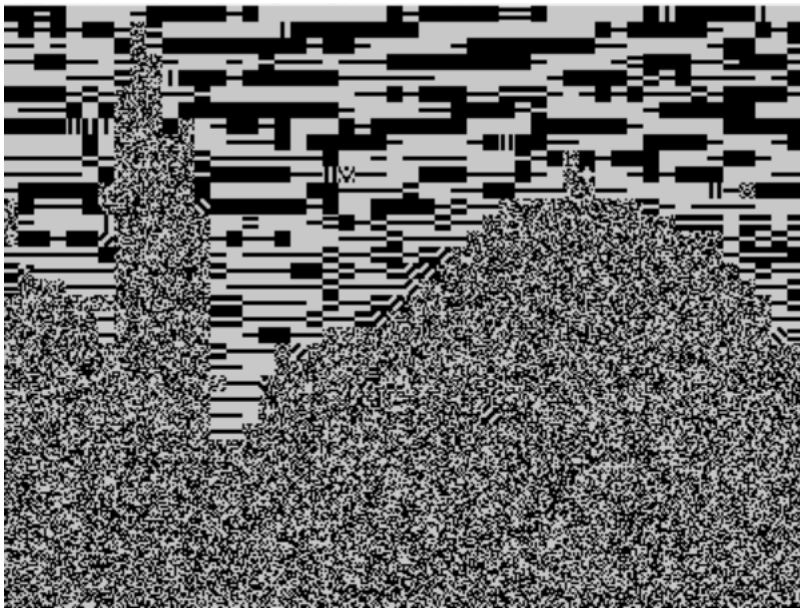
5 bit 50x



6 bit 100x

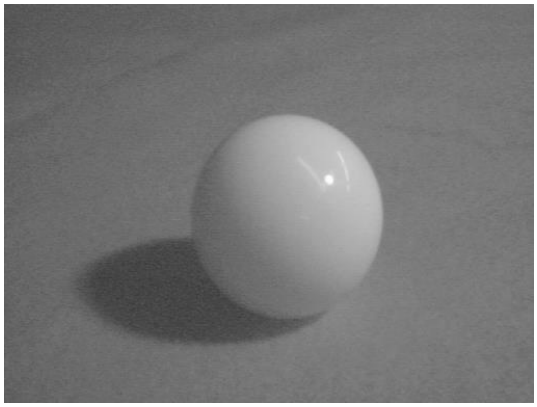


7 200x



Uygulama 4

Aşağıdaki resmi çeşitli sabit sayılar ile AND işlemine tabi tutarsak şu sonuçları elde ederiz. Sayı küçük iken görüntü koyu olduğu için görebilmek için ölçeklemek gerekmiştir.

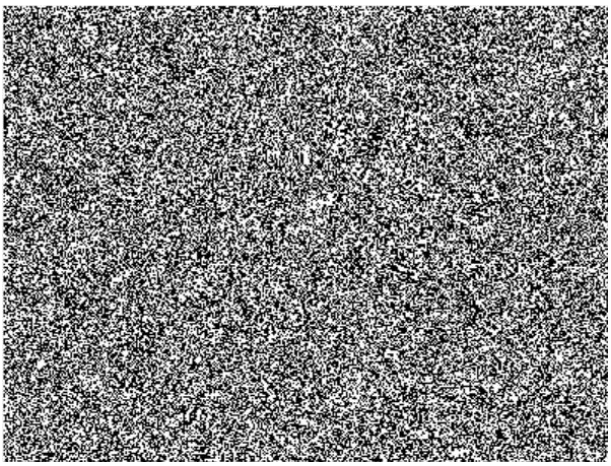


1 sayısı ile AND leme ve 1x Ölçekleme

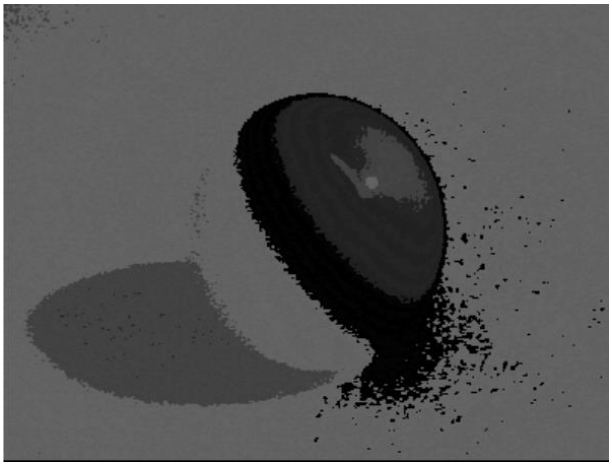
15 sayısı ile AND leme ve 10x Ölçekleme

230 sayısı ile AND leme ve 1x Ölçekleme

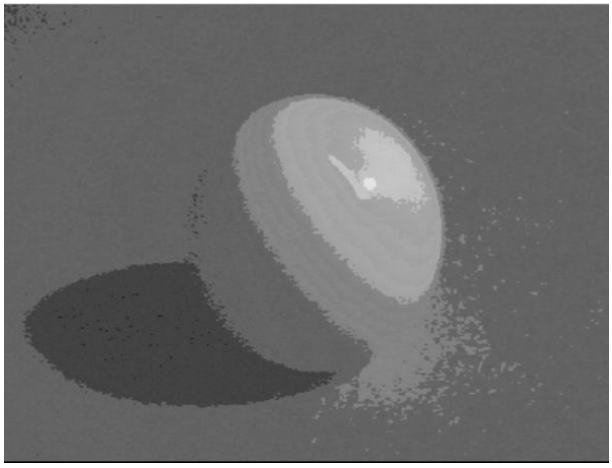
1 sayısı ile AND leme ve 255x Ölçekleme

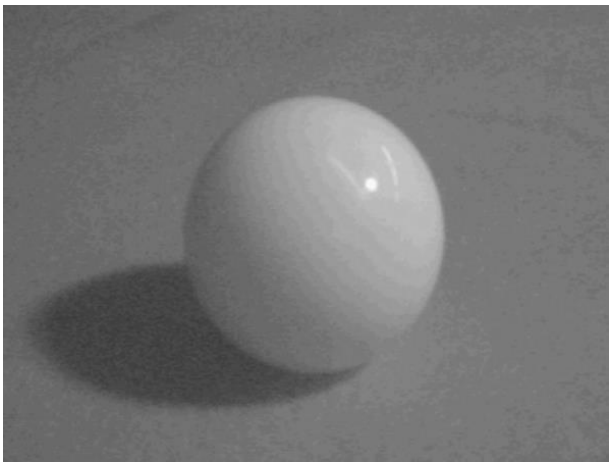


100 sayısı ile AND leme ve 1x Ölçekleme



250 sayısı ile AND leme ve 1x Ölçekleme





Benzer şekilde 0 sayısı ile AND leme yaparsak tamamen siyah resim, 255 sayısı ile AND leme yaparsak tamamen Orijinal resmi elde ederiz.

```
private void ANDSabitSayiToolStripMenuItem_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);

            string binarySayi1 = Convert.ToString(Renk1.R, 2).PadLeft(8, '0'); //Gri renk
olduğundan tek kanal üzerinden yapılıyor.

            int SabitSayi = Convert.ToInt32(txtDeger1.Text);
            string binarySayi2 = Convert.ToString(SabitSayi, 2).PadLeft(8, '0');

            string Bit1 = null, Bit2 = null, StringIkiliSayi = null;

            for (int i = 0; i < 8; i++)
            {
                Bit1 = binarySayi1.Substring(i, 1);
                Bit2 = binarySayi2.Substring(i, 1);

                //AND İŞLEMİ
                if (Bit1 == "0" && Bit2 == "0") StringIkiliSayi = StringIkiliSayi + "0";
                else if (Bit1 == "1" && Bit2 == "1") StringIkiliSayi = StringIkiliSayi + "1";
                else StringIkiliSayi = StringIkiliSayi + "0";

            }
            R = Convert.ToInt32(StringIkiliSayi, 2); //İkili sayıyı tam sayıya dönüştürüyor.

            int Olcek = Convert.ToInt32(txtDeger2.Text);
```

```
            R = R * Olcek;

            //Sınırı aşan değerleri 255 ayarlama
            if (R > 255) R = 255;
            if (G > 255) G = 255;
            if (B > 255) B = 255;

            CikisResmi.SetPixel(x, y, Color.FromArgb(R, R, R)); //Gri resim

        }
    }
    pictureBox2.Image = CikisResmi;

}
```

XOR OPERATÖRÜ BURAYA EKLENECEK..