

GÖRÜNTÜ İŞLEME - (4.Hafta)

PERSPEKTİF DÜZELTME

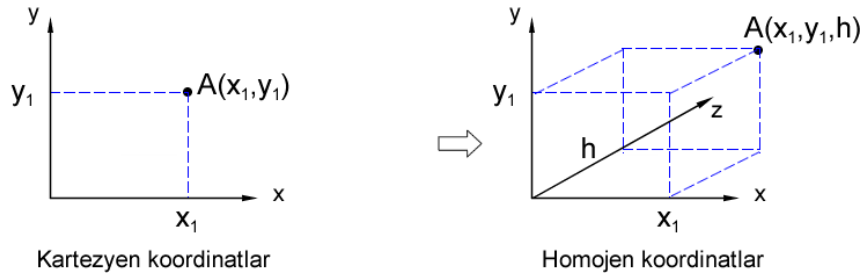
Perspektif nesnenin bulunduğu konuma bağlı olarak, gözlemcinin gözünde bıraktığı etkiyi (görüntüyü) iki boyutlu bir düzlemde canlandırmak için geliştirilmiş bir iz düşüm tekniğidir. Perspektif düzeltmede amaç kişinin veya nesnenin konum değiştirmesi sonucu oluşacak etkiyi düzeltmektir. Bu işlem sayesinde görüntü oluştuktan sonra dahi belirli kısıtlar içerisinde resme baktığımız açıyı değiştirebiliriz. Örneğin, perspektif olarak çekilmiş bir resmi, tam karşıdan çekilmiş gibi düzeltme yapabiliriz, yada çekilen kamera açısını farklı bir noktaymış gibi ayarlayabiliriz. Algoritma karakter tanıma uygulamalarında çekilmiş bir görüntüyü belirli kalıplar içerisinde oturmada, plaka tanıma, yüz tanıma gibi uygulamalarda normalizasyon sırasında sıklıkla kullanılmaktadır.

Konuya geçmeden önce dönüşüm matrislerinde işlemi kolaylaştıran "Homojen koordinatlar" konusunu bir inceleyelim.

Homojen Koordinatlar

Homojen koordinat sistemine göre, cisimler üzerinde uygulanan geometrik dönüşümlerin arka arkaya yapılabilmesini matematiksel anlamda çok esnek hale getirmektedir. Cisimlerin koordinatları matris formunda yazılarak çarpım notasyonuna tabi tutulur ve cismin yeni şekli veya konumu kolaylıkla elde edilmiş olur.

Kartezyen koordinat sisteminde bulunan bir cismin koordinatları $A(x_1, y_1)$ şeklindedir. Bu koordinatlara üçüncü bir bileşen eklenirse boyut artımına sebep olur ki, oluşan yeni sistem homojen koordinat sistemi adını alır.



Homojen koordinat sistemi yukarıda da bahsedildiği gibi, iki boyutlu kartezyen koordinat sistemine üçüncü bir boyut (z) eklenmesiyle oluşturulur. $A(x_1, y_1)$ koordinatları artık $A(x_1, y_1, h)$ üçlüsüne dönüşür. Dönüşüm için homojen koordinatlar bir h değeri ile çarpılır yada bölünür. Bu sayede iki koordinat sistemi arasında bir ilişki oluşur.

Araştırma: Homojen koordinatlara neden ihtiyaç duyuyoruz. Bu koordinat sistemi olmadan matris tersini alamaz mıyız. Araştırın?

Bu ilişkide kullanılan üçüncü koordinat bileşeni olan z, genel olarak h ile ifade edilir ve 1 alınır. Çünkü iki boyutlu eksenlerdeki bir noktanın, homojen koordinat ekseninde sonsuz gösterimi olacaktır. Örneğin kartezyen koordinat sisteminde $(x=2, y=3)$ koordinatlarına sahip bir cismin, Homojen koordinattaki gösterimi aşağıdaki gibi sonsuz sayıda olabilir.

$$\text{Kartezyen gösterim} = \text{Homojen gösterim}$$

$$A(x_1, y_1) = A(x_1, y_1, h)$$

$$A(2, 3) = A(2, 3, 1) \quad (h=1)$$

$$A(2, 3) = A(4, 6, 2) \quad (h=2)$$

$$A(2, 3) = A(6, 9, 3) \quad (h=3)$$

$$\dots = \dots$$

$$A(2, 3) = A(2n, 3n, n) \quad (h=n)$$

Bu noktaların hepsi kartezyen koordinatta aynı noktayı gösterir ve hepsi birbirine eşit demektir. Bu nedenle bir çok işlemi kolaylaştırması için homojen koordinatta $h=1$ alınır. Böylece matematiksel dönüşümler kolaylaşmış olacaktır. Dikkat edersek yukarıda tek tablo şeklinde verilen dönüşüm formüllerindeki matris gösterimleri homojen koordinatlarda verilmiştir. Örnek olarak birini yazacak olursak (döndürme matrisini seçelim) aşağıdaki şekilde olur.

Döndürme matrisinin normal gösterimi

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

Döndürme matrisinin homojen koordinatlar kullanılarak gösterimi

$$\begin{bmatrix} x_2 \\ y_2 \\ z \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Sonuç olarak homojen koordinat sistemi bize afin dönüşümlerinin tamamının çarpımsal matris formunda uygulanabilme kolaylığı sağlayacak. Geometrik hesaplamalarda esnek bir hesaplama imkanı sağlayacaktır. Bu avantajı gelecek örneklerde daha iyi anlayacağız.

-----000-----

Şimdi gelelim perspektif dönüşüm için kullanılan parametrelerin nasıl bulunabileceğine. Perspektif dönüşümünün bir matris çarpması ile yapıldığını biliyoruz. Dönüşüm için kullandığımız matrisi homojen koordinatları da dahil ederek aşağıdaki gibi oluşturabiliriz. Homojen koordinatlarda yazarken $h=1$ aldığımız için denklemler aşağıdaki gibi oluşturulur. Burada (x_1, y_1) giriş resmindeki pikselin koordinatlarını, (X_1, Y_1) ise çıkış resmindeki koordinatları temsil etmektedir (Dikkat: önceki notlarda çıkış resminin koordinatlarını x_2, y_2 şeklinde kullanıyorduk. Fakat aşağıda 4 tane noktaya ihtiyaç olacak. Dolayısıyla noktaları 1,2,3,4 indisleri ile kullanacağımızdan burada çıkış resimlerinin koordinatları büyük harflerle gösterilmiştir.) Her iki resimdeki bu noktaların koordinat değerleri bilinmektedir. Bilinmeyen $[A]$ matrisinin elemanlarıdır.

$$\begin{bmatrix} X_1 Z \\ Y_1 Z \\ z \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Burada amacımız A matrisinin (a_{xx} -elemanları ile gösterilen matris) elemanlarını bulmaktır. İlk olarak matris değerlerini açık denklemler şeklinde yazalım.

$$X_1 Z = a_{11} x_1 + a_{12} y_1 + a_{13}$$

$$Y_1 Z = a_{21} x_1 + a_{22} y_1 + a_{23}$$

$$z = a_{31} x_1 + a_{32} y_1 + 1$$

Buradaki z denklemini önceki iki denklemde yerine yazarsak ve X_1, Y_1 çekersek denklemler aşağıdaki gibi olur.

$$X_1 = \frac{a_{11} x_1 + a_{12} y_1 + a_{13}}{a_{31} x_1 + a_{32} y_1 + 1}$$

$$Y_1 = \frac{a_{21} x_1 + a_{22} y_1 + a_{23}}{a_{31} x_1 + a_{32} y_1 + 1}$$

Bu denklemlere içler dışlar çarpımı uygularsak;

$$a_{31} x_1 X_1 + a_{32} y_1 X_1 + X_1 = a_{11} x_1 + a_{12} y_1 + a_{13}$$

$$a_{31} x_1 Y_1 + a_{32} y_1 Y_1 + Y_1 = a_{21} x_1 + a_{22} y_1 + a_{23}$$

Yalnız kalan X_1 ve Y_1 terimlerini çekersek;

$$X_1 = a_{11} x_1 + a_{12} y_1 + a_{13} - a_{31} x_1 X_1 - a_{32} y_1 X_1$$

$$Y_1 = a_{21} x_1 + a_{22} y_1 + a_{23} - a_{31} x_1 Y_1 - a_{32} y_1 Y_1$$

Bu iki denklemi matris formunda yazacak olursak $[C]=[B][A]$ şeklinde bir format ortaya çıkar. Burada $[A]$ bir vektördür (Tek boyutlu matris).

$$\begin{bmatrix} X_1 \\ Y_1 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1 X_1 & -y_1 X_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1 Y_1 & -y_1 Y_1 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \end{bmatrix}$$

ifadesi elde edilir. Burada aranan [A] matrisi $[A]=[B^{-1}][C]$ işlemi ile kolaylıkla bulunabilir. Burada dikkat edilmesi gereken bir nokta elimizde sekiz bilinmeyenli bir [A] matrisi olmasına rağmen görünürde sadece iki denkleminiz olmasıdır. Sekiz bilinmeyenli bir denklemin çözümü için bağımsız sekiz tane denklem gerekmektedir. Bu amaçla tek bir nokta yerine 4 tane (x,y) noktası kullanılarak [A] matrisinin elemanları bulunacak. Bulduğumuz a_{xx} değerleri iki resim arasındaki dönüşümü yapacak olan 3x3 lük dönüşüm matrisini oluşturacaktır. Bilinen 4 nokta için matris denkleminizi tekrar yazarsak aşağıdaki şekilde olur.

$$\begin{bmatrix} X_1 \\ Y_1 \\ X_2 \\ Y_2 \\ X_3 \\ Y_3 \\ X_4 \\ Y_4 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1 X_1 & -y_1 X_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1 Y_1 & -y_1 Y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2 X_2 & -y_2 X_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2 Y_2 & -y_2 Y_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3 X_3 & -y_3 X_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3 Y_3 & -y_3 Y_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4 X_4 & -y_4 X_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4 Y_4 & -y_4 Y_4 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \end{bmatrix}$$

Bu matris denkleminde, en sağdaki [A] matrisinin elemanları hariç diğer tüm elemanları biliyoruz. [A] matrisini yalnız bırakabilmek için ortadaki 8x8 lik [B] matrisinin tersini alıp, karşı taraftaki 8x1 lik [C] matrisi ile çarpmalıyız. Yani $[A]=[B^{-1}][C]$ şeklinde bir işlem yapacağız. Bu işlem için matris tersini almayı öğrenmemiz gerekiyor. Şimdi bu konuya bakalım daha sonra kaldığımız yerden devam ederiz.

Matris Tersini Alma

Matrisler sabit değerli sayıların bir düzen içerisinde tablo şeklinde yazılması ile oluşturulur. Bu konuda geliştirilen matematik görüntü işleme, istatistik gibi pek çok alandaki problemlerin bilgisayar programı ile çözümünü kolaylaştırır. Bu amaçla matrislerle ilgili Determinant hesaplama, Tersini alma gibi konuları bilmek önem arz eder.

Matris tersini alma işlemi ile ilgili olarak Algoritmik işlemler için uygun olan **Gauss Jordan** yöntemini kullanalım. Bu yöntemde amaç bir matrisin tersini almak için, tersi olan matris ile çarpımını, birim matrise dönüştürmektir. Yani bir matrisin kendisi [A] ise tersi $[A^{-1}]$ olur, buna da [B] matrisi diyelim. Bu durumda [A] matrisi ile tersi olan [B] matrisinin çarpımı birim matrisi [I] vermelidir. Birim matris köşegeni 1 olan diğer elemanları 0 olan kare matristir.

$$[A] \times [A^{-1}] = [I]$$

Bu ifade A matrisinin tersini B matrisi ile gösterirsek, yani $[A^{-1}] = [B]$ dersek;

$$[A] \times [B] = [I]$$

olur.

Matris tersini alma ile ilgili bir çok yöntem olmasına rağmen bizi programlama kolaylığı için Gauss Jordan yöntemini öğrenelim ve buna göre programını yazalım.

Gauss Jordan Yöntemi ile Matris Tersini Alma:

Gauss-Jordan yöntemi, özellikle büyük denklem takımlarını çözmede kolaylık sağlamaktadır. Bir diğer avantajı programlamaya uygun bir yöntemdir. Pratik bir şekilde en büyük matrislerin çözümünü sağlayan bir yöntemdir. Benzeri olan Gauss Eliminasyon (yok etme) metoduna benzer.

Bir örnek üzerinde bu yöntemi inceleyelim. Aşağıdaki denklem takımını çözmeye çalışalım.

$$11 = x + 2y + 2z$$

$$4 = 3x + 2y - z$$

$$3 = 2x - y + z$$

Bu denklem takımını matris formatında yazarsak aşağıdaki şekilde olur.

$$\begin{bmatrix} 11 \\ 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 2 \\ 3 & 2 & -1 \\ 2 & -1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Bu matris takımını $[C]=[A][X]$ matris formatında gösterirsek, buradaki $[X]$ matrisindeki değerleri bilmiyoruz ve bunu bulmak için $[A]$ matrisinin tersini alıp $[C]$ matrisi ile çarpmalıyız. Yani $[X]=[A^{-1}][C]$ şeklinde olmalıdır. Buna göre $[A]$ katsayılar matrisinin tersini nasıl alırız. Bunun için Gauss Jordan yöntemini kullanacağız.

Gauss Jordan yönteminde $[A]$ katsayı matrisini birim matrise dönüştürmeye çalışacağız ve bu durumda $[C]$ matrisi de ilgili sayılarla çarpılarak belli bir değer alacak. $[A]$ matrisi birim matris olunca yeni değerlere sahip olan $[C]$ matrisi x,y,z değerlerini gösteren sonuç matrisi olacaktır, yani denklem takımının çözümü olacaktır.

$$\begin{bmatrix} 11 \\ 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 2 \\ 3 & 2 & -1 \\ 2 & -1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \Rightarrow \begin{bmatrix} * \\ * \\ * \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Buradaki işlemleri matrisi formunda kısaltmalı olarak gösterirsek,

$[C] = [A] [X]$	$[A] [A^{-1}] = [I]$
$[C] = ([I] / [A^{-1}]) [X]$	$[A] = [I] / [A^{-1}]$
$[C] [A^{-1}] = [I] [X]$	

Şimdi $[C][A]$ matrislerine çeşitli satır işlemleri uygulayarak bu dönüşümü yapalım. Burada uygulanabilecek satır işlemleri aşağıdaki işlemlerden herhangi biri olabilir.

- Bir satırın bir sabit ile çarpılması
- Bir satırın diğer bir satır ile yer değiştirmesi
- Bir satırın diğer bir satırdan çıkarılması

Daha basit gösterim için aşağıdaki formatı kullanalım. Burada $[C]$ matrisi ile $[A]$ matrisi sadece yanyana getirilmiş. Herhangi bir matematiksel anlamı yok. İşlemler yapılırken her iki matrisi de uygulanacak. Böylece sağ taraftaki $[A]$ matrisi birim matrise dönüşürken sol taraftaki $[C]$ matriside başka bir değerler alacaktır. Böyle $[A]$ matrisinin birim matris olması sonucu ortaya çıkan $[C]$ matrisi aslında çözüm olarak aradığımız $[C] [A^{-1}]$ ifadesi olmuş oluyor.

$$\begin{bmatrix} 11 \\ 4 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ 3 & 2 & -1 \\ 2 & -1 & 1 \end{bmatrix}$$

Önce köşegenin altındaki terimleri 0 yapalım. Köşegen üzerindeki değerleri de 1 yapalım. Burada hangi satırda işlem yaparsak o satırın kaçınıcı elemanı 0 yada 1 yapmaya çalışıyoruz ona bakıyoruz. Eğer 3 eleman bulunan satırın ilk elemanı değiştirmeye çalışıyorsak en yukarıdaki 1 satırda işlemleri yapmalıyız. Son 3. Elemanı değiştirmeye çalışıyorsak en son 3. Satırı kullanmalıyız. Çıkan değerleri kendi bulunduğu satır ile topluyoruz.

a) $(a_{21}=3)$ değerini 0 yapmak için birinci satırı -3 ile çarpıp (ilk terim olduğu için birinci satır ile çarpıyoruz) 2. satır ile toplayalım. Yani $[-3*11] [-3*1, -3*2, -3*2] = [-33] [-3, -6, -6]$ olur. Bu değerleri 2. satır ile toplarsak $[33+4] [-3+3, -6+2, -6-1] = [-29] [0, -4, -7]$. Bu sonucu matristeki yerlerinde gösterirsek; (Not: burada R gösterimi Row (satır manasında). İşlem yapılan satırları ifade ediyor.)

$$\begin{bmatrix} 11 \\ 4 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ 3 & 2 & -1 \\ 2 & -1 & 1 \end{bmatrix} \xrightarrow{R_2 = -3R_1 + R_2} \begin{bmatrix} 11 \\ -29 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ 0 & -4 & -7 \\ 2 & -1 & 1 \end{bmatrix}$$

b) $(a_{22}=-4)$ değerini 1 yapmak için ikinci satırı -4 bölmemiz yeterli olacaktır. Yani $[-29/-4][0/-4, -4/-4, -7/-4] = [29/4][0, 1, 7/4]$ olur. Bu sonucu matristeki yerlerinde gösterirsek;

$$\begin{bmatrix} 11 \\ -29 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ 0 & -4 & -7 \\ 2 & -1 & 1 \end{bmatrix} \xrightarrow{R_2 = R_2 / -4} \begin{bmatrix} 11 \\ 29/4 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ 0 & 1 & 7/4 \\ 2 & -1 & 1 \end{bmatrix}$$

c) $(a_{31}=2)$ değerini 0 yapmak için; o satırdaki ilk eleman olduğu için birinci satırı kullanmalıyız. Birinci satırı -2 ile çarparsak ve çıkan sonuçları da 3. satır ile toplarsak bu ilk terimi sıfır yapmış oluruz. Yani $[-2*11][-2*1, -2*2, -2*2] = [-22][-2, -4, -4]$ olur. Bu değerleri 3. satır ile toplarsak $[-22+3][-2+2, -4-1, -4+1] = [-19][0, -5, -3]$. Bu sonucu matristeki yerlerinde gösterirsek;

$$\begin{bmatrix} 11 \\ 29/4 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ 0 & 1 & 7/4 \\ 2 & -1 & 1 \end{bmatrix} \xrightarrow{R_3 = -2R_1 + R_3} \begin{bmatrix} 11 \\ 29/4 \\ -19 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ 0 & 1 & 7/4 \\ 0 & -5 & -3 \end{bmatrix}$$

d) $(a_{32}=-5)$ değerini 0 yapmak için; için ikinci satırı kullanmalıyız. İkinci satırı +5 ile çarparsak ve çıkan sonuçları da kendisi ile toplarsak bu ikinci terimi sıfır yapmış oluruz. Yani $[5*29/4][5*0, 5*1, 5*7/4] = [145/4][0, 5, 35/4]$ olur. Bu değerleri 3. satır ile toplarsak $[(145/4) -19][0+0, 5-5, (35/4)-3] = [69/4][0,0,23/4]$. Bu sonucu matristeki yerlerinde gösterirsek;

$$\begin{bmatrix} 11 \\ 29/4 \\ -19 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ 0 & 1 & 7/4 \\ 0 & -5 & -3 \end{bmatrix} \xrightarrow{R_3 = +5R_2 + R_3} \begin{bmatrix} 11 \\ 29/4 \\ 69/4 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ 0 & 1 & 7/4 \\ 0 & 0 & 23/4 \end{bmatrix}$$

e) $(a_{33}=23/5)$ değerini 1 yapmak için; üçüncü satırı ya $23/4$ bölmeliyiz yada $4/23$ ile çarpmalıyız.

$$\begin{bmatrix} 11 \\ 29/4 \\ 69/4 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ 0 & 1 & 7/4 \\ 0 & 0 & 23/4 \end{bmatrix} \xrightarrow{R_3 = R_3 * 4/23} \begin{bmatrix} 11 \\ 29/4 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ 0 & 1 & 7/4 \\ 0 & 0 & 1 \end{bmatrix}$$

Buraya kadar yapılan işlemler Gauss Eliminasyon yöntemi olarak geçmektedir. Köşegenin üst kısmındaki değerleri de 0 yaparsak Gauss Jordan yöntemi olmuş oluyor. Benzer şekilde kaldığımız yerden devam ederek köşegenin üst kısmını da 0 yapalım.

f) $(a_{12}=2)$ değerini 0 yapmak için; ikinci satırı -2 ile çarpıp birinci satır ile toplamalıyız. Yani $[-2*29/4][-2*0, -2*1, -2*7/4] = [-29/2][0, -2, -7/2]$ olur. Bu değerleri 1. satır ile toplarsak $[-29/2+11][0+1, -2+2, -7/2+2] = [-7/2][1,0,-3/2]$. Bu sonucu matristeki yerlerinde gösterirsek;

$$\begin{bmatrix} 11 \\ 29/4 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ 0 & 1 & 7/4 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{R_1 = -2*R_2 + R_1} \begin{bmatrix} -7/2 \\ 29/4 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 0 & -3/2 \\ 0 & 1 & 7/4 \\ 0 & 0 & 1 \end{bmatrix}$$

g) $(a_{13}=-3/2)$ değerini 0 yapmak için; üçüncü satırı $3/2$ ile çarpıp birinci satır ile toplamalıyız. Yani $[3/2*3][3/2*0, 3/2*0, 3/2*1] = [9/2][0, 0, 3/2]$ olur. Bu değerleri 1. satır ile toplarsak $[9/2-7/2][0+1, 0+0, 3/2-3/2] = [1][1,0,0]$. Bu sonucu matristeki yerlerinde gösterirsek;

$$\begin{bmatrix} -7/2 \\ 29/4 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 0 & -3/2 \\ 0 & 1 & 7/4 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{R_1 = 3/2*R_3 + R_1} \begin{bmatrix} 1 \\ 29/4 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 7/4 \\ 0 & 0 & 1 \end{bmatrix}$$

g) $(a_{23}=7/4)$ değerini 0 yapmak için; üçüncü satırı $-7/4$ ile çarpıp ikinci satır ile toplamalıyız. Yani $[-7/4*3][-7/4*0, -7/4*0, -7/4*1] = [-21/4][0, 0, -7/4]$ olur. Bu değerleri 2. satır ile toplarsak $[-21/4+29/4][0+0, 0+1, -7/4+7/4] = [2][0,1,0]$. Bu sonucu matristeki yerlerinde gösterirsek;

$$\begin{bmatrix} 1 \\ 29/4 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 7/4 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{R_1 = -7/4*R_3 + R_2} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Böylece katsayılar matrisini birim matrise dönüştürmüş olduk. Sonuç itibariyle denklem takımımız şöyle olur.

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Bu matris formatını açarsak x=1, y=2, z=3 olarak bulmuş oluruz.

Programlama (Matris Tersini alma)(Gauss Jordan yöntemi)

Matrisin tersinin alınması işleminin bilgisayarlar marifetiyle yapılması için algoritmik bir yaklaşıma ihtiyaç vardır. Bu bağlamda matrisin boyu ve içeriğinden bağımsız olarak matrisin tersini almak için aşağıdaki yaklaşımı kullanabiliriz.

Bir matrisin tersi, matrisin kendisi ile çarpıldığına birim matrisi veren matristir. Birim matris (identity matrix) ise diyagonu 1 ve diğer elemanları 0 olan matristir.

Öyleyse bir matrisi gerekli işlemleri yaparak birim matrise dönüştürürsek ve bu işlemler sırasında her elemana yapılan değişimi tutarsak sonuçta elde ettiğimiz bu değişim matrisi, orjinal matrisimizin tersi olacaktır.

Aşağıda bu işlemleri adım adım yapan C kodunu yazıp açıklamaya çalışalım.

İlk adımda matris boyutunu ve içeriğini dolduralım.

```

1  #include<stdio.h>
2  #include<conio.h>
3  int main()
4  {
5      float a[4][4]={1,2,3,4,7,11,9,0,9,8,7,6,1,12,3,14};
6      float b[4][4],c[4][4];
7      for(int i=0;i<4;i++){
8          for(int j=0;j<4;j++){
9              printf("    %f ",a[i][j]);
10             }
11             printf("\n");
12         }

```

Kodun ilk kısmında görüldüğü üzere 4×4 boyutlarında bir matris tanımlanmış ve bu matrisin içeriği ekrana bastırılmıştır. Yukarıdaki tanım itibariyle matrisimizin içeriği aşağıdaki şekildedir:

1.000000	2.000000	3.000000	4.000000
7.000000	11.000000	9.000000	0.000000
9.000000	8.000000	7.000000	6.000000
1.000000	12.000000	3.000000	14.000000

Şimdi bu matrisin tersini alma işlemi sırasında kullanacağımız birim matrisi döngüler ile oluşturabiliriz:

```

13  for(int i=0;i<4;i++){
14      for(int j=0;j<4;j++){
15          if(i==j)
16              b[i][j]=1;
17          else
18              b[i][j]=0;
19      }
20  }

```

Yukarıdaki kod, köşegeninde (diagon) 1 olan ve diğer elemanları 0 olan bir matris inşa etmektedir. Bu işlem için matrisin satır ve sütun koordinatlarını tutan i ve j döngü değişkenlerinin eşit olması durumu 1, diğer durumlar 0 olarak döngülerde kodlanmıştır.

Şu anda, tersi alınması istenen matris a dizisinde, birim matris ise b dizisinde tutulmaktadır. Yapılması gereken, a matrisini b matrisine dönüştürmektir.

```

21 float d,k;
22 for(int i=0;i<4;i++){
23     d=a[i][i];
24     for(int j=0;j<4;j++){
25         a[i][j]=a[i][j]/d;
26         b[i][j]=b[i][j]/d;
27     }
28     for(int x=0;x<4;x++){
29         if(x!=i){
30             k=a[x][i];
31             for(int j=0;j<4;j++){
32                 a[x][j]=a[x][j]-(a[i][j]*k);
33                 b[x][j]=b[x][j]-(b[i][j]*k);
34             }
35         }
36     }
37 }

```

Yukarıda görüldüğü üzere kodun 24-27. satırları arasında matrisin diyagonu olan değerleri 1 yapacak değerler bulunmaktadır. Bir sayının kendisine bölümü 1'dir. Dolayısıyla matrisin tersi olan sonuç matrisimizdeki, ki şu an itibariyle birim matristir, diyagon değerleri, orjinal matrisin diyagon değerlerine bölünmüştür.

Ardından diyagon olmayan değerler için dönen ve 28. satırda başlayan ikinci döngü ile diyagonda olmayan elemanlardan, diyagondaki elemanla çarpımları çıkarılmaktadır. (kodun 32. satırındaki k değişkeni o satırdaki diyagon değerini vermektedir. Bu değer orjinal matristeki değer ile çarpılıp yine aynı satır ve sütundaki değerden çıkarılmaktadır)

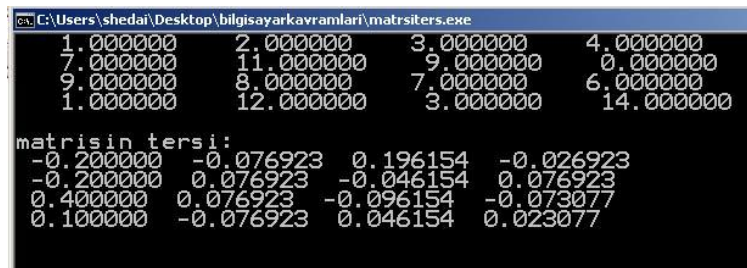
Bu işlem hem orjinal matris hem de sonuç matrisinde bire bir yapılıp ve sonuçta sonuç matrisimiz b değişkeni içinde oluşur. Bu değeri bastırmak için aşağıdaki şekilde bir döngü yazılması yeterlidir.

```

38 printf("\nmatrisin tersi: \n");
39 for(int i=0;i<4;i++)
40 {
41     for(int j=0;j<4;j++)
42         printf(" %f ",b[i][j]);
43     printf("\n");
44 }
45 getch();
46 return 0;
47 }

```

Kodumuzun çalışan hali aşağıda verilmiştir:



```

C:\Users\sheda\Desktop\bilgisayarkavramlari\matrsiters.exe
1.000000 2.000000 3.000000 4.000000
7.000000 11.000000 9.000000 0.000000
9.000000 8.000000 7.000000 6.000000
1.000000 12.000000 3.000000 14.000000

matrisin tersi:
-0.200000 -0.076923 0.196154 -0.026923
-0.200000 0.076923 -0.046154 0.076923
0.400000 0.076923 -0.096154 -0.073077
0.100000 -0.076923 0.046154 0.023077

```


C# Kodları

```

=====A Matrisi=====
1,2,3,4,
7,11,9,0,
9,8,7,6,
1,12,3,14,
=====I Matrisi=====
1,0,0,0,
0,1,0,0,
0,0,1,0,
0,0,0,1,
=====B Matrisi=====
-0,2,-0,0769230769230768,0,196153846153846,-0,0269230769230769,
-0,2,0,0769230769230769,-0,0461538461538462,0,0769230769230769,
0,4,0,0769230769230769,-0,0961538461538461,-0,0730769230769231,
0,1,-0,0769230769230769,0,0461538461538461,0,0230769230769231,
==Kontrol==AxB=I Matrisi=====
1,0,0,0,
0,1,0,0,
0,0,1,0,
0,0,0,1,

```

```

// MATRİS TERSİNİ ALMA-----
public double[,] MatrisTersiniAl (double[,] GirisMatrisi)
{
    int MatrisBoyutu = Convert.ToInt16(Math.Sqrt(GirisMatrisi.Length)); //matris boyutu içindeki
    eleman sayısı olduğu için kare matrisde karekökü matris boyutu olur.
    double[,] CikisMatrisi = new double[MatrisBoyutu, MatrisBoyutu]; //A nın tersi alındığında bu
    matris içinde tutulacak.

```

```

//--I Birim matrisin içeriğini dolduruyor

```

```

for (int i = 0; i < MatrisBoyutu; i++)
{
    for (int j = 0; j < MatrisBoyutu; j++)
    {
        if (i == j)
            CikisMatrisi[i, j] = 1;
        else
            CikisMatrisi[i, j] = 0;
    }
}

```

```

//--Matris Tersini alma işlemi-----

```

```

double d, k;

for (int i = 0; i < MatrisBoyutu; i++)
{
    d = GirisMatrisi[i, i];
    for (int j = 0; j < MatrisBoyutu; j++)
    {
        if (d == 0)
        {
            d = 0.0001; //0 bölme hata veriyordu.
        }
        GirisMatrisi[i, j] = GirisMatrisi[i, j] / d;
        CikisMatrisi[i, j] = CikisMatrisi[i, j] / d;
    }
    for (int x = 0; x < MatrisBoyutu; x++)
    {
        if (x != i)

```



```

{
    k = GirisMatrisi[x, i];
    for (int j = 0; j < MatrisBoyutu; j++)
    {
        GirisMatrisi[x, j] = GirisMatrisi[x, j] - GirisMatrisi[i, j] * k;
        CikisMatrisi[x, j] = CikisMatrisi[x, j] - CikisMatrisi[i, j] * k;
    }
}
}

return CikisMatrisi;
}

```

-----000-----

Kaldığımız yerden devam edersek, matris tersini almayı öğrendik. Şimdi aşağıdaki 4 nokta üzerinde perspektif düzeltme yapacak matris denklemindeki a_{xx} katsayılarını bulalım. Bunun için bir örnek resim kullanalım ve üzerindeki anahtar noktaların koordinatlarını paint programında okuyup dönüşüm matrisini bulalım.



Bu resimdeki 1,2,3,4 nolu noktaları sırayla a,b,c,d noktalarının olduğu köşelere doğru gerdirilecektir. Böylece plakanın yazım alanı resmin tamamını kaplayacaktır. Fakat plakanın en boyu orantısı bozulacağı için a,b,c,d noktalarının olduğu koordinatlar resmin içerisine sığdırmak yerine, istediğimiz herhangi bir dikdörtgenin köşeleri olarak da atayabiliriz. Önce birinci uygulamayı bir yapalım. Bu 8 noktanın koordinatlarını paint gibi basit bir programa atıp mouse ile üzerinde gezdirirsek koordinatları okuyabiliriz. Tabi burada resmin üzerine sonradan eklenen çizgiler işlemlerde olmayacaktır. Yukarıdaki resmin üzerindeki noktaların koordinatları aşağıda gibi okunmuştur. $x_2=806$ değeri çerçevenin dışında bir nokta olarak belirlenmiştir.

- 1.nokta, $x_1=47$, $y_1=155$
- 2.nokta, $x_2=806$, $y_2=101$
- 3.nokta, $x_3=46$, $y_3=291$
- 4.nokta, $x_4=726$, $y_4=417$

- a noktası, $X_1=0$, $Y_1=0$
- b noktası, $X_2=800$, $Y_2=0$
- c noktası, $X_3=0$, $Y_3=600$
- d noktası, $X_4=800$, $Y_4=600$

Bu değerleri aşağıdaki dönüşüm matrisimizde yerine yazalım.

$$\begin{bmatrix} X_1 \\ Y_1 \\ X_2 \\ Y_2 \\ X_3 \\ Y_3 \\ X_4 \\ Y_4 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1 X_1 & -y_1 Y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1 Y_1 & -y_1 X_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2 X_2 & -y_2 Y_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2 Y_2 & -y_2 X_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3 X_3 & -y_3 Y_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3 Y_3 & -y_3 X_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4 X_4 & -y_4 Y_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4 Y_4 & -y_4 X_4 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \end{bmatrix}$$

8 noktanın koordinatı yerlerine yazıldığında denklem aşağıdaki gibi olur.

$$\begin{bmatrix} 0 \\ 0 \\ 800 \\ 0 \\ 0 \\ 600 \\ 800 \\ 600 \end{bmatrix} = \begin{bmatrix} 47 & 155 & 1 & 0 & 0 & 0 & -47 * 0 & -155 * 0 \\ 0 & 0 & 0 & 47 & 155 & 1 & -47 * 0 & -155 * 0 \\ 806 & 101 & 1 & 0 & 0 & 0 & -806 * 800 & -101 * 800 \\ 0 & 0 & 0 & 806 & 101 & 1 & -806 * 0 & -101 * 0 \\ 46 & 291 & 1 & 0 & 0 & 0 & -46 * 0 & -291 * 0 \\ 0 & 0 & 0 & 46 & 291 & 1 & -46 * 600 & -291 * 600 \\ 726 & 417 & 1 & 0 & 0 & 0 & -726 * 800 & -417 * 800 \\ 0 & 0 & 0 & 726 & 417 & 1 & -726 * 600 & -417 * 600 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \end{bmatrix}$$

Bu matris denklemini , [C] = [B] [A] şeklinde gösterirsek B matrisinin tersini [B⁻¹] yukarıdaki program ile bulabiliriz. Bu durumda aşağıdaki sonucu buluruz.

$$[B^{-1}] = \begin{bmatrix} -0,005547 & -0,004954 & 0,002844 & 0,002039 & 0,004403 & 0,005187 & -0,001699 & -0,002272 \\ -0,007394 & -3,6E-05 & 2,1E-05 & 1,5E-05 & 0,007385 & 3,8E-05 & -1,2E-05 & -1,7E-05 \\ 2,406739 & 0,23849 & -0,136889 & -0,098176 & -1,351657 & -0,24969 & 0,081807 & 0,109375 \\ -0,000809 & -0,001922 & 0,000333 & 0,001351 & 0,000847 & 0,000607 & -0,000371 & -3,7E-05 \\ -0,011368 & -0,008493 & 0,00468 & 0,000473 & 0,011902 & 0,008536 & -0,005214 & -0,000517 \\ 1,800114 & 2,406739 & -0,741026 & -0,136889 & -1,884646 & -1,351657 & 0,825558 & 0,081807 \\ -3E-06 & -6E-06 & 1E-06 & 2E-06 & 3E-06 & 6E-06 & -2E-06 & -3E-06 \\ -8E-06 & 0 & 3E-06 & 0 & 9E-06 & 0 & -4E-06 & 0 \end{bmatrix}$$

$$[A] = [B^{-1}] [C]$$

$$\begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} & b_{16} & b_{17} & b_{18} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} & b_{26} & b_{27} & b_{28} \\ b_{31} & b_{32} & b_{33} & b_{34} & b_{35} & b_{36} & b_{37} & b_{38} \\ b_{41} & b_{42} & b_{43} & b_{44} & b_{45} & b_{46} & b_{47} & b_{48} \\ b_{51} & b_{52} & b_{53} & b_{54} & b_{55} & b_{56} & b_{57} & b_{58} \\ b_{61} & b_{62} & b_{63} & b_{64} & b_{65} & b_{66} & b_{67} & b_{68} \\ b_{71} & b_{72} & b_{73} & b_{74} & b_{75} & b_{76} & b_{77} & b_{78} \\ b_{81} & b_{82} & b_{83} & b_{84} & b_{85} & b_{86} & b_{87} & b_{88} \end{bmatrix} \begin{bmatrix} X_1 \\ Y_1 \\ X_2 \\ Y_2 \\ X_3 \\ Y_3 \\ X_4 \\ Y_4 \end{bmatrix}$$

Matris formatını denklem şekline dönüştürürsek.

$$a_{11} = b_{11}X_1 + b_{12}Y_1 + b_{13}X_2 + b_{14}Y_2 + b_{15}X_3 + b_{16}Y_3 + b_{17}X_4 + b_{18}Y_4$$

$$a_{12} = b_{21}X_1 + b_{22}Y_1 + b_{23}X_2 + b_{24}Y_2 + b_{25}X_3 + b_{26}Y_3 + b_{27}X_4 + b_{28}Y_4$$

(...şeklinde devam eder...)

$$a_{21} = b_{41}X_1 + b_{42}Y_1 + b_{43}X_2 + b_{44}Y_2 + b_{45}X_3 + b_{46}Y_3 + b_{47}X_4 + b_{48}Y_4$$

$$a_{22} = b_{51}X_1 + b_{52}Y_1 + b_{53}X_2 + b_{54}Y_2 + b_{55}X_3 + b_{56}Y_3 + b_{57}X_4 + b_{58}Y_4$$

(...şeklinde devam eder...)

----- 0000-----

b_{xx} matrisin tersinin katsayılarını bulduğumuza göre ve X_x, Y_x koordinatlarını da biliyoruz buradan a_{xx} değerlerini buluruz. Bu değerler aynı zamanda dönüşüm denklemindeki $[A]$ matrisinin elemanlarıdır. Yani aşağıdaki denklemdeki katsayılar matrisidir.

$$\begin{bmatrix} X_1Z \\ Y_1Z \\ Z \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Bu denklemi resim üzerindeki tüm noktalara uygulamak için genelleştirirsek ve aynı zamanda programlamada kullanacağımız indislere dönüştürürsek aşağıdaki şekilde olur. Programlarken diziler (0,0) değerinden başlar.

$$\begin{bmatrix} XZ \\ YZ \\ Z \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$Z = a_{20}X + a_{21}Y + 1$$

$$X = \frac{a_{00}X + a_{01}Y + a_{02}}{Z}$$

$$Y = \frac{a_{10}X + a_{11}Y + a_{12}}{Z}$$

Dikkat: Matrisleri programlarken indisler bir sayı aşağıdan başlar. Çünkü dizilerin başlangıç değerleri (0,0) dır. Buna göre programda a_{11} matris elemanı a_{00} olarak gösterilir.

Artık bu formülleri kullanarak tüm belirlediğimiz alanı istediğimiz çerçeve içine sığdırabiliriz.

Programlama (Perspektif Düzeltme)

Aşağıdaki sınırlarda plakanın köşeleri (x_x, y_x -koordinatları), resmin sınırlarına (X_x, Y_x) genişletilmiştir. Bu nedenle plaka doğal görünümünün dışına çıkmıştır.



Plakayı daha gerçekçi boyutlar görmek için dönüşeceği sınırları daha küçük tutmalıyız. ($X_1, Y_1=0,178$; $X_2, Y_2=800,178$; $X_3, Y_3=0,378$; $X_4, Y_4=800,378$). Çerçeve 800x200 olarak belirlenmiştir.



Resim ne kadar büyük bir çerçeveye yerleştirilirse aradaki boşluklar (piksel kayıpları-alias) o kadar fazla olmaktadır. Elde edilen resimler tekrar filtreden geçilip kayıp pikseller düzeltilmelidir. Eğer yerleştirilecek çerçeve daha küçük seçilirse kayıp piksel azalmaktadır. Aşağıdaki örnek (400x100) boyutlarındadır. Burada gösterilen resimler küçültülmüş boyutlardadır. Bu resimlerin her biri gerçekte 800x600 boyutlarındadır.



Yukarıdaki ilk örnekte plaka alanı (800x600) alana yerleştirilince kayıp piksel artmıştır (a resmi). Plaka 400x100 boyutlarına küçültüldüğünde kayıp pikseller daha az oluşmuştur (b resmi).



```
private void perspektifDuzeltme2ToolStripMenuItem_Click(object sender, EventArgs e)
{
    //pictureBox2.Image= PerspektifDuzelt();
    double x1 = 47, y1 = 155;
    double x2 = 806, y2 = 101;
    double x3 = 46, y3 = 291;
    double x4 = 726, y4 = 417;

    double X1 = 0, Y1 = 0;
    double X2 = 800, Y2 = 0;
    double X3 = 0, Y3 = 600;
    double X4 = 800, Y4 = 600;

    double[,] GirisMatrisi = new double[8,8];

    // { x1, y1, 1, 0, 0, 0, -x1 * X1, -y1 * X1 }
    GirisMatrisi[0,0] = x1;
    GirisMatrisi[0,1] = y1;
    GirisMatrisi[0,2] = 1;
}
```

```

GirisMatrisi[0,3] = 0;
GirisMatrisi[0,4] = 0;
GirisMatrisi[0,5] = 0;
GirisMatrisi[0,6] = -x1 * X1;
GirisMatrisi[0,7] = -y1 * X1;

//{ 0, 0, 0, x1, y1, 1, -x1 * Y1, -y1 * Y1 }
GirisMatrisi[1,0] = 0;
GirisMatrisi[1,1] = 0;
GirisMatrisi[1,2] = 0;
GirisMatrisi[1,3] = x1;
GirisMatrisi[1,4] = y1;
GirisMatrisi[1,5] = 1;
GirisMatrisi[1,6] = -x1 * Y1;
GirisMatrisi[1,7] = -y1 * Y1;

//{ x2, y2, 1, 0, 0, 0, -x2 * X2, -y2 * X2 }
GirisMatrisi[2,0] = x2;
GirisMatrisi[2,1] = y2;
GirisMatrisi[2,2] = 1;
GirisMatrisi[2,3] = 0;
GirisMatrisi[2,4] = 0;
GirisMatrisi[2,5] = 0;
GirisMatrisi[2,6] = -x2 * X2;
GirisMatrisi[2,7] = -y2 * X2;

//{ 0, 0, 0, x2, y2, 1, -x2 * Y2, -y2 * Y2 }
GirisMatrisi[3,0] = 0;
GirisMatrisi[3,1] = 0;
GirisMatrisi[3,2] = 0;
GirisMatrisi[3,3] = x2;
GirisMatrisi[3,4] = y2;
GirisMatrisi[3,5] = 1;
GirisMatrisi[3,6] = -x2 * Y2;
GirisMatrisi[3,7] = -y2 * Y2;

//{ x3, y3, 1, 0, 0, 0, -x3 * X3, -y3 * X3 }
GirisMatrisi[4,0] = x3;
GirisMatrisi[4,1] = y3;
GirisMatrisi[4,2] = 1;
GirisMatrisi[4,3] = 0;
GirisMatrisi[4,4] = 0;
GirisMatrisi[4,5] = 0;
GirisMatrisi[4,6] = -x3 * X3;
GirisMatrisi[4,7] = -y3 * X3;

//{ 0, 0, 0, x3, y3, 1, -x3 * Y3, -y3 * Y3 }
GirisMatrisi[5,0] = 0;
GirisMatrisi[5,1] = 0;
GirisMatrisi[5,2] = 0;
GirisMatrisi[5,3] = x3;
GirisMatrisi[5,4] = y3;
GirisMatrisi[5,5] = 1;
GirisMatrisi[5,6] = -x3 * Y3;
GirisMatrisi[5,7] = -y3 * Y3;

//{ x4, y4, 1, 0, 0, 0, -x4 * X4, -y4 * X4 }
GirisMatrisi[6,0] = x4;

```

```

GirisMatrisi[6,1] = y4;
GirisMatrisi[6,2] = 1;
GirisMatrisi[6,3] = 0;
GirisMatrisi[6,4] = 0;
GirisMatrisi[6,5] = 0;
GirisMatrisi[6,6] = -x4 * X4;
GirisMatrisi[6,7] = -y4 * X4;

//{ 0, 0, 0, x4, y4, 1, -x4 * Y4, -y4 * Y4 }
GirisMatrisi[7,0] = 0;
GirisMatrisi[7,1] = 0;
GirisMatrisi[7,2] = 0;
GirisMatrisi[7,3] = x4;
GirisMatrisi[7,4] = y4;
GirisMatrisi[7,5] = 1;
GirisMatrisi[7,6] = -x4 * Y4;
GirisMatrisi[7,7] = -y4 * Y4;

//-----
double[,] matrisBTersi = MatrisTersiniAl(GirisMatrisi);
//-----
//----- A Dönüşüm Matrisi (3x3) -----
double a00 = 0, a01 = 0, a02 = 0, a10 = 0, a11 = 0, a12 = 0, a20 = 0, a21 = 0, a22 = 0;
for (int i = 0; i < 8; i++)
{
    a00 = matrisBTersi[0, 0] * X1 + matrisBTersi[0, 1] * Y1 + matrisBTersi[0, 2] * X2 +
matrisBTersi[0, 3] * Y2 + matrisBTersi[0, 4] * X3 + matrisBTersi[0, 5] * Y3 + matrisBTersi[0, 6] * X4 +
matrisBTersi[0, 7] * Y4;
    a01 = matrisBTersi[1, 0] * X1 + matrisBTersi[1, 1] * Y1 + matrisBTersi[1, 2] * X2 +
matrisBTersi[1, 3] * Y2 + matrisBTersi[1, 4] * X3 + matrisBTersi[1, 5] * Y3 + matrisBTersi[1, 6] * X4 +
matrisBTersi[1, 7] * Y4;
    a02 = matrisBTersi[2, 0] * X1 + matrisBTersi[2, 1] * Y1 + matrisBTersi[2, 2] * X2 +
matrisBTersi[2, 3] * Y2 + matrisBTersi[2, 4] * X3 + matrisBTersi[2, 5] * Y3 + matrisBTersi[2, 6] * X4 +
matrisBTersi[2, 7] * Y4;

    a10 = matrisBTersi[3, 0] * X1 + matrisBTersi[3, 1] * Y1 + matrisBTersi[3, 2] * X2 +
matrisBTersi[3, 3] * Y2 + matrisBTersi[3, 4] * X3 + matrisBTersi[3, 5] * Y3 + matrisBTersi[3, 6] * X4 +
matrisBTersi[3, 7] * Y4;
    a11 = matrisBTersi[4, 0] * X1 + matrisBTersi[4, 1] * Y1 + matrisBTersi[4, 2] * X2 +
matrisBTersi[4, 3] * Y2 + matrisBTersi[4, 4] * X3 + matrisBTersi[4, 5] * Y3 + matrisBTersi[4, 6] * X4 +
matrisBTersi[4, 7] * Y4;
    a12 = matrisBTersi[5, 0] * X1 + matrisBTersi[5, 1] * Y1 + matrisBTersi[5, 2] * X2 +
matrisBTersi[5, 3] * Y2 + matrisBTersi[5, 4] * X3 + matrisBTersi[5, 5] * Y3 + matrisBTersi[5, 6] * X4 +
matrisBTersi[5, 7] * Y4;

    a20 = matrisBTersi[6, 0] * X1 + matrisBTersi[6, 1] * Y1 + matrisBTersi[6, 2] * X2 +
matrisBTersi[6, 3] * Y2 + matrisBTersi[6, 4] * X3 + matrisBTersi[6, 5] * Y3 + matrisBTersi[6, 6] * X4 +
matrisBTersi[6, 7] * Y4;
    a21 = matrisBTersi[7, 0] * X1 + matrisBTersi[7, 1] * Y1 + matrisBTersi[7, 2] * X2 +
matrisBTersi[7, 3] * Y2 + matrisBTersi[7, 4] * X3 + matrisBTersi[7, 5] * Y3 + matrisBTersi[7, 6] * X4 +
matrisBTersi[7, 7] * Y4;
    a22 = 1;
}

//----- Perspektif düzeltme işlemi -----
PerspektifDuzelt(a00, a01, a02, a10, a11, a12, a20, a21, a22, X1, Y1, X4, Y4);

```



```

    }

//===== Perspektif düzeltme işlemi =====
    public void PerspektifDuzelt(double a00, double a01, double a02, double a10, double a11, double
a12, double a20, double a21, double a22, double X1, double Y1, double X4, double Y4)
    {
        Bitmap GirisResmi, CikisResmi;
        Color OkunanRenk;

        GirisResmi = new Bitmap(pictureBox1.Image);

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

        Label1.Text = "W=" + ResimGenisligi + "; " + "H" + ResimYuksekligi;

        CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);

        double X,Y,z;

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

                z=a20*x + a21*y + 1;
                X=(a00 * x + a01 * y + a02)/z;
                Y=(a10 * x + a11 * y + a12)/z;

                if (X > X1 && X < X4 && Y > Y1 && Y < Y4) //Çapraz iki köşe kullanılarak dikdörtgen sınır
belirlendi (X1,Y1)-(X4,Y4). Bu sınırın dışına çıkan pikseller resme kaydedilmeyecek.
                    CikisResmi.SetPixel((int)X, (int)Y, OkunanRenk);
            }
        }

        pictureBox2.Image = CikisResmi;
    }

```

Ödev 1. Perspektif alanı mouse ile seçme

Yukarıdaki örnek uygulamada 1,2,3,4 ve a,b,c,d noktalarını belirlemeyi mouse ile resim üzerine tıklayarak yapın. Yani bu 8 tane nokta belirlenirken mouse kullanın. Buna göre perspektif düzeltme işlemini gerçekleştirin. Örnek resim olarak bina cephelerindeki perspektif görünümlü reklam tabelalarını kullanın. Perspektif düzeltmesi yapıldıktan sonra Alias oluşan noktalarda (hesaplama yapmadığı ve resim üzerinde boşlukları temsil eden siyah noktaları) daha sonra düzeltmeye çalışın. Bu kısımlara geldiğinde etrafındaki 9 tane pikselin değerine bakın ve ortadaki pikseli bu piksellerin ortalaması olarak aldırın. Buna benzer resmi daha da iyileştirecek bir yöntem bulmaya çalışın.

Araştırma: Gaus Jordan yönteminin ne olduğunu araştıralım. Yukarıda geçen 8x8 matrislerin tersini almak için programlamada bu yöntem mi kullanılmaktadır, değilse hangi yöntemi kullanıyor.