GEBZE TECHNICAL UNIVERSITY

CSE 462 - Applied Augment Reality and 3D User Interfaces

Homework 2 – Report

İslam Göktan SELÇUK – 141044071

PART 1

1.1-) Homography matrisi hesaplanırken karşılıklı dört tane nokta kullanıldı.

```
{

    double[,] S = { { 3, 5 }, { 25, 53 }, { 45, 170 }, { 13, 64 } };
    double[,] I = { { 6, 10 }, { 28, 58 }, { 48, 175 }, { 16, 69 } };

    double[] result = HomographyCalculation(S, I);
```

Bu karşılıklı dört nokta ile 8 tane homojen lineer eşitlikler oluşturuldu.

```
for (int i = 0; i < S.GetLength(0); i++)
{
    double[,] correspondingCoordinates = { { -S[i, 0], -S[i, 1], -1, 0, 0, 0, S[i, 0] * I[i, 0], S[i, 1] * I[i, 0], I[i, 0] },
                                           { 0, 0, 0, -S[i, 0], -S[i, 1], -1, S[i, 0] * I[i, 1], S[i, 1] * I[i, 1], I[i, 1] } };
    for (int k = 0; k < 2; k++)
    {
        for (int j = 0; j < columnLen; j++)
        {
            equationMatrix[i * 2, j] = correspondingCoordinates[0, j];
            equationMatrix[i * 2 + 1, j] = correspondingCoordinates[1, j];
        }
    }
}
```

Bu denklemleri çözmek için oluşturulan matrise single value decomposition uygulandı.

```
// We use single-value-decomposition for solving equations.
var svd = DenseMatrix.OfArray(equationMatrix).Svd(true);
```

Svd = U * S * V^T olduğundan svd objesi üzerindeki right singular vector'lerin bulunduğu matrisin transpozunun son sütunundaki değerler ile Homography array'i elde edilmiş oldu.

```
        // A = U * S * VT(V's transpose)
        // Homography array is VT's last column.
        var result = svd.VT.Row(svd.VT.RowCount - 1);
        return result.AsArray();
```

Yukarıda verilen karşılıklı noktaların oluşturduğu homography matrisi çıktısı:

```
Microsoft Visual Studio Debug Console
Homographt Matrix:
-0.164398987305352 -3.34176524722031E-16 -0.493196961916065
5.16030919686076E-15 -0.164398987305355 -0.821994936526793
8.59303424639259E-17 -1.47584160050053E-17 -0.164398987305354
```

1.3-) Verilen noktanın projeksiyonu istendiği için homography matrisi kullanıldı.

Öncelikle homography matrisi ile verilen noktanın matris çarpımının sonucu bulundu. Daha sonrasında sonrasında elde edilen vektörün elemanları vektörün ikinci elemanına bölündü.

```
// Problem 1.3
6 references
public static double[,] ProjectionCalculation(double[,] homographyMatrix, double[,] point)
{
    double[,] multiplication = MultiplyTwoMatrices(homographyMatrix, point);

    var projectionResult = new double[,] { { multiplication[0, 0] / multiplication[2,0] },
                                           { multiplication[1, 0] / multiplication[2,0] },
                                           { 1 } };

    return projectionResult;
}
```

{ {3}, {5}, {1} } noktası için hesaplanan sonuç çıktısı:

```
Projection for given point

5.99999999999999
10.0000000000001
1
```

1.4-) Verilen noktanın scene üzerindeki projeksiyonu istendiği için bir önceki part'ta elde ettiğimiz sonucu kullarak, önceki part'taki input değerine ulaşıldı.

Öncelikle homography matrisi elde edildi ve bu matrisin tersi alındı. Önceki parttaki projeksiyon hesaplama fonksiyonu ile sonuç bulunmuş oldu.

```
// Problem 1.4
2 references
public static double[,] ProjectionCalculationForImagePoint(double[,] homographyMatrix, double[,] point)
{
    var inverseOfHomographyMatrix = DenseMatrix.OfArray(homographyMatrix).Inverse();

    var projectionResult = ProjectionCalculation(inverseOfHomographyMatrix.ToArray(), point);
    return projectionResult;
}
```

{ {5.99}, {10}, {1} } noktası için hesaplanan sonuç çıktısı:

```
Projection for Image point

3
5
1
```

1.5-) Öncelikle paint ve pdf'te verilen resim ile karşılıklı noktalar bulundu.

Pdf'deki ve verilen resimlerdeki kordinatlar:

```
double[,] pdfCoordinates = { { 100, 100 },
                             { 200, 100 },
                             { 100, 200 },
                             { 200, 200 },
                             { 100, 300 } };

double[,] image1 = { { 758, 1745 },
                     { 987, 1743 },
                     { 759, 1512 },
                     { 987, 1514 },
                     { 764, 1277 } };
```

```
double[,] image2 = { { 738, 1939 },
                     { 971, 1939 },
                     { 733, 1711 },
                     { 969, 1714 },
                     { 729, 1480 } };

double[,] image3 = { { 878, 1702 },
                     { 1125, 1705 },
                     { 894, 1457 },
                     { 1139, 1467 },
                     { 910, 1219 } };
```

,

Birinci resim için Paint üzerinden hesaplanan sonuçlar:

(300, 300) => (1225, 1287)

Test Sonucu => (1207, 1293)

Hata Oranı: %1.069

(400, 300) => (1457, 1287)

Test Sonucu => (1415, 1300)

Hata Oranı: %2.261

(500, 100) => (1685, 1753)

Test Sonucu => (1612, 1736)

Hata Oranı: %3.083

Birinci Resim için hesaplanan sonuçlar:

```
>>>>>>>> Part 1.5 Tests <<<<<<<<
------- Image 1 Results -------
Result of point: (300, 300):
1207.27849122257
1293.46937101158
1
Result of point: (400, 300):
1415.0643526152
1300.94809147279
1
Result of point: (500, 100):
1612.06171371954
1736.35498536077
1
```

İkinci resim için hesaplanan sonuçlar:

```
------- Image 2 Results -------
Result of point: (300, 300):
1198.12866306733
1490.78129892813
1
Result of point: (400, 300):
1424.639425737
1496.04230005294
1
Result of point: (500, 100):
1640.41261154032
1939.32370207885
1
```

Üçüncü resim için hesaplanan sonuçlar:

```
------- Image 3 Results -------
Result of point: (300, 300):
1380.77198187724
1248.99360199834
1
Result of point: (400, 300):
1598.98985796179
1262.99819627328
1
Result of point: (500, 100):
1796.75096494002
1713.88997128395
1
```

1.6-) Pdf'teki resim üzerindeki projeksiyonunu hesaplamamız istenen noktalar:

```
// Problem 1.6
double[,] S1 = new double[,] { { 7.5f }, { 5.5f }, { 1 } };
double[,] S2 = new double[3, 1] { { 6.3f }, { 3.3f }, { 1 } };
double[,] S3 = new double[3, 1] { { 0.1f }, { 0.1f }, { 1 } };
```

Sonuçlar:

```
>>>>>>>>>>>>>> Part 1.6 Tests <<<<<<<<<<<<<
S1 results for image 1:
534.45052369732
1974.89328018484
1
S2 results for image 1:
522.554112289443
2152.90456162971
1
S3 results for image 1:
617.445497875359
1940.8329663965
1
```

```
S1 results for image 2:
522.554112289443
2152.90456162971
1
S2 results for image 2:
522.554112289443
2152.90456162971
1
S3 results for image 2:
522.554112289443
2152.90456162971
1
S1 results for image 3:
617.445497875359
1940.8329663965
1
S2 results for image 3:
617.445497875359
1940.8329663965
1
S3 results for image 3:
617.445497875359
1940.8329663965
1
```

Part 1.7-) Paylaşılan resimler üzerindeki projeksiyonunu hesaplamamız istenen noktalar:

```
// Problem 1.7
Console.WriteLine(">>>>>>>>>>>>>>> Part 1.7 Tests <<<<<<<<<<<<<<");
double[,] I1 = new double[,] { { 500f }, { 400f }, { 1 } };
double[,] I2 = new double[3, 1] { { 86f }, { 167f }, { 1 } };
double[,] I3 = new double[3, 1] { { 10f }, { 10f }, { 1 } };
```

Çıktı sonuçları:

```
>>>>>>>>>>>>>>> Part 1.7 Tests <<<<<<<<<<<<<<
I1 results for image 1:
-14.5531469037282
664.371490227145
1
I2 results for image 1:
-178.970973438012
738.748454734888
1
I3 results for image 1:
-208.49974898962
797.94103239302
1
I1 results for image 2:
17.1897657934135
739.972105994411
1
I2 results for image 2:
-139.227578701992
817.569163569682
1
I3 results for image 2:
-165.269591239373
874.654176620237
1
I1 results for image 3:
-85.1597705865449
631.107097920938
1
I2 results for image 3:
-244.250600543206
703.362730832844
1
I3 results for image 3:
-276.943881452558
764.049315912553
1
```