

SAYISAL VİDEO İŞLEME ÖDEV#2

Yöntem:

1. Video dosyası okunarak çerçeveler bir listede tutulur.
2. Listedeki her bir çerçeve 9 eşit bloğa bölünerek kırpılır.
3. Çerçevenin kırpılan blokları için ayrı ayrı histogram hesaplanır. Histogram hesaplama adımıyla bloğun piksellerinin 0-359 aralığında değişen Hue değerleri sayılarak her bir Hue değerinden kaç tane olduğu hesaplanır. Histogram değerleri normalize edilir. Sonuçta her bir çerçeve için 9 x 360 büyüklüğünde özellik matrisi elde edilmiş olur.
4. Ardışık iki çerçevenin blokları arasında Bhattacharyya mesafesi hesaplanır. Bhattacharyya mesafesi Bhattacharyya katsayısının 0 -1 arasındaki değerine bakılarak yorumlanır. Bhattacharyya katsayısı, bir bloğun t anındaki histogramıyla t+1 anındaki histogramının çarpılıp, çarpımın karekökünün toplama katılmasıyla elde edilir. Birbiriyle çarpılan histogramlar mutlaka normalize edilmiş olmalıdır. İki bloğun benzerliği ne kadar çoksa, Bhattacharyya katsayısı 1'e o kadar yakın bir değer alır. Blokların benzerliği azaldıkça katsayı 0'a yakın değerler alır.

$bhattacharyya_coefficient = 0$

for $i = 0 : 359$

$bhattacharyya_coefficient += \sqrt{blok_histT[i]*blok_histT1[i]}$

5. t ve t+1 anındaki 9 blok için hesaplanan Bhattacharyya mesafesinin ortalama değeri alınarak ardışık iki görüntü arasındaki benzerlik hesaplanır.
6. Ardışık görüntülerin benzerliklerini içeren vektör K-Means algoritması yardımıyla sınıflandırılır. Birbirine oldukça benzer olan ardışık görüntüler aynı sınıfta, kamera geçişi olan görüntüler farklı sınıfta yer alır.
7. Kamera geçişi olmayan çerçevelerin kamera geçişi olarak algılanmasını engellemek için eşik değeri belirlenerek sonuçlar değerlendirilmiştir.

Uygulama:

Sınıflandırma sonucunda oluşturulan ardışık çerçeveler ile aralarındaki mesafeler incelendiğinde, kamera geçişi olmayan ardışık çerçevelerinin genelde 0.95 ten yüksek değerler aldığı görülmüştür.

Ancak 0.8 – 0.9 benzerliğe sahip kamera geçişi olmayan ardışık çerçeveler de bulunmaktadır. Bu çerçeveler programın yanlış tahmin yapmasında neden olmuştur.

Uygulama aşamasında BBC News, CCN International ve Al Jazeera kanallarının haber videolarında program denenmiştir. Gerçekten çekim geçişi olup program tarafından bulunan görüntülerden çeşitli örnekler aşağıdadır:

					D-980 0.78 Fp
					D-1280 0.81 Fp

					D-2024 0.79 Tp
					D-1355 0.78 Tp

Tabloda verilen D-980 değeri D videosundaki 980 numaralı çerçeveyi göstermektedir. 0.78 ardışık çerçevelerin benzerliğini, fp False Positive'i işaret etmektedir. Diğer kutucukların içindeki bilgilendirmeler de aynı biçimde yapılmıştır.

980 ve 1280 numaraların çerçevelerde geçiş olmamasına rağmen geçiş olarak algılanmasının sebebi çerçevelerin siyah-beyaz ağırlıklı resim olmalarıdır. İki çerçevede de yapılan zoom işlemi az miktarda da olsa histogramlarda büyük değişime sebep olmuş ve kamera geçişi olarak algılanmıştır.

2024 ve 1355 numaralı çerçevelerde geçiş yavaş olmasına rağmen başarılı bir şekilde tespit edilebilmiştir.

Her bir video için çerçeveler K-Means ile sınıflandırılarak hata matrisi oluşturulmuştur.

D :2325	Gerçek Kamera Geçişi	Gerçek Normal
Tahmin Kamera Geçişi	31	2
Tahmin Normal	0	2292

$$TP = 31 / 31 = 1$$

$$FN = 0 / 31 = 0$$

					E-335 0.56 Tp
					E-356 0.74 Tp
					E-1320 0.65 Fp

					E- 1440 0.59 Tp
--	--	--	--	--	--------------------------

E videosunun 335,356 ve 1440 numaralı çerçevelerinde kamera geçişi başarılı şekilde tespit edilmiştir. 1320 numaralı çerçevede futbolcunun hareket ederek yön değiştirmesi arka planda renk değişimine sebep olmuş ve kamera geçişi olarak yanlış tahmin edilmiştir.

E :1798	Gerçek Kamera Geçişi	Gerçek Normal
Tahmin Kamera Geçişi	88	2
Tahmin Normal	0	1708

$$TP = 88 / 88 = 1$$

$$FN = 0 / 88 = 0$$

					F- 25 0.89 Fp
					F- 596 0.88 Fp
					F- 84 0.86 Fp
					F- 345 0.87 Tp
					F- 511 0.82 Fp




F videosunda 345 numaralı çerçevedeki bulanık geçiş başarılı bir biçimde tespit edilmiştir. Geçişler oldukça yavaş olduğundan kamera geçişleri ile normal geçişler arasındaki Bhattacharyya katsayısı farkı azdır. Tablodaki benzerlikler incelendiğinde de 345 numaralı kamera geçişi olan çerçevede Bhattacharyya benzerliği 0.87 ve normal çerçevelerde benzerlik 0.87-0.88 aralığında birbirine yakın değerler almaktadır. 511 numaralı çerçevede ise görüntüdeki kişinin hareket etmesiyle blok histogramları değişmiş ve kamera geçişi olarak algılanmıştır.

F videosunda K-Means sonuç değerlerine 0.8 ve 0.9 eşik değerleri uygulanarak false positive değerleri düşürülmeye çalışılmıştır. Ancak false positive sayısı düştükçe true positive sayısı da düşmektedir. Bu durum sistemin hatasını daha fazla arttırmaktadır. Bu sebeptendir ki eşik seviyesi uygulaması yararlı bulunmamıştır.

F :2448	Gerçek Kamera Geçişi	Gerçek Normal
Tahmin Kamera Geçişi	90	60
Tahmin Normal	0	2238

$$TP = 90 / 90 = 1$$

$$FN = 0 / 90 = 0$$

					G-321 0.69 Fp
					G-1547 0.71 Tp
					G-2047 0.64 Fp
					G-2431 0.28 Tp





















G videosunda tespit edilen kamera geçişlerine, 1547 numaralı çerçevedeki bulanık geçiş ve 2431 numaralı çerçevedeki keskin geçiş başarılı birer örnektir. Bu iki çerçevedeki benzerlikler karşılaştırıldığında, keskin geçişlerdeki benzerlik oranı 0' a yakın, bulanık geçişlerdeki benzerlik ise 1'e yakın değer aldığı görülmüştür.

321 numaralı çerçevede öndeki kişi hareketsiz olmasına rağmen arka plan hareketli olduğu için lokal histogram değerleri değişmiş ve çerçeve kamera geçişi olarak algılanmıştır. 2047 numaralı çerçevenin yanlış kamera geçişi olmasının nedeni ise kameranın baktığı açının yavaş yavaş değişmesidir.

G :2568	Gerçek Kamera Geçişi	Gerçek Normal
Tahmin Kamera Geçişi	65	2
Tahmin Normal	0	2501

$$TP = 65 / 65 = 1$$

$$FN = 0 / 65 = 0$$

					H-2004 0.64 Fp
					H-2428 0.59 Tp
					H-2633 0.66 Fp
					H-3003 0.37 Tp

H videosunda 2428 ve 3033 numaralı çerçevelerde kamera geçişleri başarılı bir şekilde tespit edilmiştir. 2004 numaralı çerçevede az miktarda yakınlaştırma işlemi yapılmış ve çerçevenin kamera geçişi olarak algılanmasına neden olmuştur. 2633 numaralı çerçevede de kalabalık ortamdaki insan ve taşıt hareketliliği kamera geçişi tahmini yapılmasında etkili olmuştur.

H :3050	Gerçek Kamera Geçişi	Gerçek Normal
Tahmin Kamera Geçişi	83	4
Tahmin Normal	0	2963

$$TP = 83 / 83 = 1$$

$$FN = 0 / 83 = 0$$

Sonuç:

Yapılan çalışmada keskin kamera geçişlerinde çerçeveler arası benzerliğin genellikle 0.1-0.6 aralığında, bulanık geçişlerin ise 0.6-0.9 aralığında gerçekleştiği görülmüştür. Normal çerçevelerin benzerliği ise çoğunlukla 0.85-1 aralığında değişim göstermektedir. Bulanık geçiş ve normal çerçevelerde tam bir sınır ayrımı yapılamadığı için, normal çerçevelerin küçük bir kısmı, bulanık geçiş olarak algılanmıştır.

Kamerada zoom yapılması, arka plandaki hareketlilik, nesnelerin ani hareketleri ya da kameranın bakış açısının değişmesi gibi sebepler çerçevenin kamera geçişi olarak tahmin edilmesine neden olmuştur.

Gerçek kamera geçişlerinin yakalanmasında keskin ve bulanık geçişlerde herhangi bir sorunla karşılaşmamıştır.

Normal çerçevelerin kamera geçişi olarak algılanmasını engellemek için 0.9, 0.8 ve 0.85 eşik değerleri uygulanmıştır. Ancak eşik değeri fp sayısını düşürmekle beraber tp sayısını da düşürmüştür.

Uygulanan yöntem etkili bir yöntem olarak kabul görmemiştir.

Program Kodu:

```
// main Class //

using System;

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Emgu.CV;
using Emgu.CV.CvEnum;
using Emgu.CV.Structure;
using Emgu.CV.UI;
using System.Drawing.Imaging;
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        private List<Bitmap> bitmap_array = new List<Bitmap>(); // all frames list
        VideoCapture _capture;

        private Timer My_Timer = new Timer();
        int FPS = 25;
        int flag = 0;
        int height, width;

        private List<double> battDist = new List<double>(); // bhattacharyya distance
list

        public Form1()
        {
            InitializeComponent();
            //Frame Rate
            My_Timer.Interval = 1000 / FPS;
            My_Timer.Tick += new EventHandler(My_Timer_Tick);
            My_Timer.Start();
            _capture = new VideoCapture("C:\\Users\\Mukaddes\\Desktop\\h.mp4");
        }

        private void Form1_Load(object sender, EventArgs e)
        {
        }

        private void My_Timer_Tick(object sender, EventArgs e)
        {
        }
    }
}
```

```

        try
        {
            if (_capture.QueryFrame() != null &&
                _capture.QueryFrame().ToImage<Bgr, Byte>() != null)
            {
                // read video frame and save into list
                Image<Bgr, Byte> imgOrg = _capture.QueryFrame().ToImage<Bgr,
Byte>();

                pb_video.Image = imgOrg.ToBitmap();
                bitmap_array.Add(imgOrg.ToBitmap());

            }
            else
            {
                My_Timer.Stop();
                flag = 1;
                tb_finish.Text = "video yuklendi";
                Console.WriteLine("bitti");
            }

        }
        catch (NullReferenceException ne)
        {
            MessageBox.Show(ne.Message);
            return;
        }
    }

    private void pb_video_Click(object sender, EventArgs e)
    {

    }

    private void btn_compute_Click(object sender, EventArgs e)
    {
        if (flag == 1)
        {

            Functions func = new Functions();
            double[] dist = new double[9];
            double[][] histT1 = new double[9][];
            double[][] histT2 = new double[9][];

            // partite frame into 3x3 blocks and histogram computing for blocks
            for (int j=0; j<bitmap_array.Count-1; j++)
            {
                List<Bitmap> btm_blocks_t1 = func.partitionImage(bitmap_array[j],
3);

                histT1 = func.computeHistogram(btm_blocks_t1);
                //for (int i = 0; i < 9; i++)
                //{
                //    Console.Write(i);
                //    for (int m = 0; m < 360; m++)
                //        Console.Write(" " + histT1[i][m]);
                //    Console.WriteLine();
                //}

                List<Bitmap> btm_blocks_t2 =
func.partitionImage(bitmap_array[j+1], 3);
                histT2 = func.computeHistogram(btm_blocks_t2);

```

```

foreach(Bitmap btm in btm_blocks_t1)
{
    height = btm.Height;
    width = btm.Width;
}

// for t and t+1 frame compute distance block by block
for (int i = 0; i < 9; i++)
{
    //Console.WriteLine(i);
    dist[i] = func.bhattacharyyaDistance(histT1[i], histT2[i]);
}

// mean distance for blocks
double mean_dist = dist.Average();
battDist.Add(mean_dist);

}
int dim = battDist.Count();

Matrix<float> distance = new Matrix<float>(dim, 1);
Matrix<int> labels = new Matrix<int>(dim, 1);
for (int u = 0; u < dim; u++)
{
    c[u, 0] = (float)battDist[u];
}

// k means clustering for distances
MCvTermCriteria termCrit = new MCvTermCriteria(100);
CvInvoke.Kmeans(distance, 2, labels, termCrit, 2, 0);

//Console.WriteLine(j);
//panel1.Visible = false;
//btm_parts[0].Save("C:\\Users\\Mukaddes\\Desktop\\Yeni
klasör\\0.bmp", System.Drawing.Imaging.ImageFormat.Bmp);
//pb1.Image = btm_parts[0];
//pb2.Image = btm_parts[1];
//pb3.Image = btm_parts[2];
//pb4.Image = btm_parts[3];
//pb5.Image = btm_parts[4];
//pb6.Image = btm_parts[5];
//pb7.Image = btm_parts[6];
//pb8.Image = btm_parts[7];
//pb9.Image = btm_parts[8];

Console.WriteLine("yazmaya gecis");
using (System.IO.StreamWriter file =
new
System.IO.StreamWriter(@"C:\Users\Mukaddes\Desktop\outpuh\distance_cluster.txt"))
{
    for (int i = 0; i < dim; i++)

        file.WriteLine(distance[i,0] + " " + labels[i,0]);

}
int count1 = 0;
for (int i = 0; i < dim; i++)
    if (labels[i, 0] == 1)

```



```

        count1++;

    for (int i = 0; i < dim; i++)
    {
        if(count1 > (dim-count1))
        { //frames being shot boundary
            if (labels[i, 0] == 0 )
            {
                if (i > 2)
                {
                    string filename1 =
(@"C:\Users\Mukaddes\Desktop\outpath\shot" + i + "_1.jpg");
                    string filename2 =
(@"C:\Users\Mukaddes\Desktop\outpath\shot" + i + "_2.jpg");
                    string filename3 =
(@"C:\Users\Mukaddes\Desktop\outpath\shot" + i + "_3.jpg");
                    string filename4 =
(@"C:\Users\Mukaddes\Desktop\outpath\shot" + i + "_4.jpg");
                    string filename5 =
(@"C:\Users\Mukaddes\Desktop\outpath\shot" + i + "_5.jpg");
                    bitmap_array[i - 2].Save(filename1, ImageFormat.Jpeg);
                    bitmap_array[i - 1].Save(filename2, ImageFormat.Jpeg);
                    bitmap_array[i].Save(filename3, ImageFormat.Jpeg);
                    bitmap_array[i + 1].Save(filename4, ImageFormat.Jpeg);
                    bitmap_array[i + 2].Save(filename5, ImageFormat.Jpeg);
                }
            }
        }
        else
        { // frames being shot boundary
            if (labels[i, 0] == 1)
            {
                if (i > 2)
                {
                    string filename1 =
(@"C:\Users\Mukaddes\Desktop\outpath\shot" + i + "_1.jpg");
                    string filename2 =
(@"C:\Users\Mukaddes\Desktop\outpath\shot" + i + "_2.jpg");
                    string filename3 =
(@"C:\Users\Mukaddes\Desktop\outpath\shot" + i + "_3.jpg");
                    string filename4 =
(@"C:\Users\Mukaddes\Desktop\outpath\shot" + i + "_4.jpg");
                    string filename5 =
(@"C:\Users\Mukaddes\Desktop\outpath\shot" + i + "_5.jpg");
                    bitmap_array[i - 2].Save(filename1, ImageFormat.Jpeg);
                    bitmap_array[i - 1].Save(filename2, ImageFormat.Jpeg);
                    bitmap_array[i].Save(filename3, ImageFormat.Jpeg);
                    bitmap_array[i + 1].Save(filename4, ImageFormat.Jpeg);
                    bitmap_array[i + 2].Save(filename5, ImageFormat.Jpeg);
                }
            }
        }
    }
}
}
}
}

```

```
}
```

```
// Function Class //
```

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.Drawing;
```

```
using Emgu.CV;  
using Emgu.Util;  
using Emgu.CV.Structure;  
using Emgu.CV.CvEnum;
```

```
namespace WindowsFormsApplication1  
{
```

```
    class Functions  
    {
```

```
        private int w, h;
```

```
        // Görüntüyü axa boyutlarında bloklara ayırma.
```

```
        public List<Bitmap> partitionImage(Bitmap bitmap, int x)  
        {  
            List<Bitmap> blocks = new List<Bitmap>();
```

```
            for (int i = 0; i < x; i++)
```

```
            {
```

```
                for (int y = 0; y < x; y++)
```

```
                {
```

```
                    // Rect(x-coordinate, y-coordinate, width,height)
```

```
                    Rectangle r = new Rectangle(i * (bitmap.Width / x), y *  
(bitmap.Height / x), bitmap.Width / x, bitmap.Height / x);  
                    blocks.Add(cropImage(bitmap, r));
```

```
                }
```

```
            }
```

```
            return blocks;
```

```
        }
```

```
        // crop the blocks from frame
```

```
        public Bitmap cropImage(Bitmap bitmap, Rectangle cropArea)  
        {
```

```
            Bitmap bmpCrop = bitmap.Clone(cropArea,  
System.Drawing.Imaging.PixelFormat.DontCare);
```

```
            return bmpCrop;
```

```
        }
```

```
        // compute histogram for each block in frame
```

```
        public double[][] computeHistogram(List<Bitmap> blocks)  
        {
```

```
            double[][] block_features = new double[9][];
```

```
            int block_no = 0;
```

```
            int hue;
```

```
            for (int i = 0; i < 9; i++)
```

```
                block_features[i] = new double[360];
```

```

foreach (Bitmap block in blocks)
{
    w = block.Width;
    h = block.Height;
    for (int i = 0; i < w; i++)
        for (int j = 0; j < h; j++)
        {
            hue = (int)block.GetPixel(i, j).GetHue();
            //Console.WriteLine(hue);
            block_features[block_no][hue]++;
        }

    block_no++;
}

for (int i = 0; i < 9; i++)
    for (int j = 0; j < 360; j++)
        block_features[i][j] /= (w * h);

return block_features;
}
// compute distance between blocks in t and t+1 frame
public double bhattacharyyaDistance(double[] histT1, double[] histT2)
{
    double bhatt_coef = 0;

    for (int m = 0; m < 360; m++)
        bhatt_coef += Math.Sqrt(histT1[m]*histT2[m]);
    //Console.WriteLine("bhat:" + bhatt_coef);
    return bhatt_coef;
}

}
}

```