Grafika i komunikacja człowiek - komputer

Rok akademicki 2022

6.05.2022

Uniwersytet w Białymstoku
Autor: Mateusz Jabłoński

Druga praca zaliczeniowa

Laboratoria : 5, 6, 7

Technologia C#

Opcja rozjaśniania:

```
// rozjaśnienie
        private void button7_Click(object sender, EventArgs e)
        {
            Bitmap b1 = (Bitmap)pictureBox1.Image;
            Bitmap b2 = (Bitmap)pictureBox2.Image;
            int _b = 127;
            Color k;
            int r, g, b;

            // pętla po mapie bitowej
            for(int i = 0; i < width; i++)
            {
```

Funkcja przyciemniania:

```
private void button8_Click(object sender, EventArgs e)
        {
        Bitmap b1 = (Bitmap)pictureBox1.Image;
```

```
    Bitmap b2 = (Bitmap)pictureBox2.Image;
    int _b = 127;
    Color k;
    int r, g, b;
    for (int i = 0; i < width; i++)
    {
       for (int j = 0; j < height; j++)
       {
          k = b1.GetPixel(i, j);

          r = k.R - _b;
          g = k.G - _b;
          b = k.B - _b;
          if (k.R - _b < 0)
          {
             r = 0;
          }
          if (k.G - _b < 0)
          {
             g = 0;
          }
          if (k.B - _b < 0)
          {
             b = 0;
          }

          b2.SetPixel(i, j, Color.FromArgb(r, g, b));

       }
    }
    pictureBox2.Refresh();
}
```

NEGATYW:

```
private void button6_Click(object sender, EventArgs e)
        {
          Bitmap b1 = (Bitmap)pictureBox1.Image;
          Bitmap b2 = (Bitmap)pictureBox2.Image;
          Color k;
          int r, g, b;
          for (int i = 0; i < width; i++)
          {
             for (int j = 0; j < height; j++)
             {
                // od 255 odejmuję poszczególne wartości r, g, b
                k = b1.GetPixel(i, j);
                r = 255-k.R;
                g = 255-k.G;
                b = 255-k.B;

                k = Color.FromArgb(r, g, b);
                b2.SetPixel(i, j, k);
             }
          }
          pictureBox2.Refresh();
       }
```

SUMA:

```
if (k1.R + k2.R > 255) r = 255;
        else r = k1.R + k2.R;

        if (k1.G + k2.G > 255) g = 255;
        else g = k1.G + k2.G;

        if (k1.B + k2.B > 255) b = 255;
        else b = k1.B + k2.B;

        b2.SetPixel(i, j, Color.FromArgb(r, g, b));
```

Odejmowanie:

```
r = k1.R + k2.R - 255;
        g = k1.G + k2.G - 255;
        b = k1.B + k2.B - 255;

        if (r < 0) r = 0;
        if (g < 0) g = 0;
        if (b < 0) b = 0;

        b2.SetPixel(i, j, Color.FromArgb(r, g, b));
```

MNOŻENIE :

```
r = (k1.R * k2.R)/255;
g = (k1.G * k2.G)/255;
b = (k1.B * k2.B)/255;

if (r > 255) r = 255;
if (g > 255) g = 255;
if (b > 255) b = 255;
```

RÓŻNICA:

```
r = Math.Abs(k1.R - k2.R);
g = Math.Abs(k1.G - k2.G);
```

```
b = Math.Abs(k1.B - k2.B);
```

MNOŻENIE ODWROTNOŚCI:

```
r = 255 - (((255 - k1.R) * (255 - k2.R)) / 255);
if (r > 255) r = 255;
if (r < 0) r = 0;

g = 255 - (((255 - k1.G) * (255 - k2.G)) / 255);
if (g > 255) g = 255;
if (g < 0) g = 0;

b = 255 - (((255 - k1.B) * (255 - k2.B)) / 255);
if (b > 255) b = 255;
if (b < 0) b = 0;
```

NEGACJA:

```
r = 255 - Math.Abs(255 - k1.R - k2.R);
g = 255 - Math.Abs(255 - k1.G - k2.G);
b = 255 - Math.Abs(255 - k1.B - k2.B);
```

Ciemniejsze:

```
if (k1.R < k2.R)
{
   r = k1.R;
}
else r = k2.R;

if (k1.G < k2.G)
{
   g = k1.G;
}
else g = k2.G;

if (k1.B < k2.B)
{
   b = k1.B;
```

```
}
else b = k2.B;
```

Jaśniejsze :

```
if (k1.R > k2.R)
{
   r = k1.R;
}
else r = k2.R;

if (k1.G > k2.G)
{
   g = k1.G;
}
else g = k2.G;

if (k1.B > k2.B)
{
   b = k1.B;
}
else b = k2.B;
```

WYŁĄCZENIE:

```
r = k1.R + k2.R - 2 * ((k1.R * k2.R) / 255);
g = k1.G + k2.G - 2 * ((k1.G * k2.G) / 255);
b = k1.B + k2.B - 2 * ((k1.B * k2.B) / 255);
```

NAKŁADKA:

```
if(k1.R < (255 / 2))
{
  r = 2 * ((k1.R * k2.R) / 255);
} else r = 255 - 2 * (((255 - k1.R) * (255-k2.R))/255);

if (k1.G < (255 / 2))
{
   g = 2 * ((k1.G * k2.G) / 255);
```

```
}
else g = 255 - 2 * (((255 - k1.G) * (255 - k2.G)) / 255);

if (k1.B < (255 / 2))
{
   b = 2 * ((k1.B * k2.B) / 255);
}
else b = 255 - 2 * (((255 - k1.B) * (255 - k2.B)) / 255);
```

OSTRE ŚWIATŁO:

```
 if (k2.R < 0) r = (2 * k1.R * k2.R) / 255;
else r = 255 - (2 * (255 - k1.R) * (255 - k2.R)) / 255;
if (r < 0) r *= -1;

if (k2.G < 0) g = (2 * k1.G * k2.G) / 255;
else g = 255 - (2 * (255 - k1.G) * (255 - k2.G)) / 255;
if (g < 0) g *= -1;

if (k2.B < 0) b = (2 * k1.B * k2.B) / 255;
else b = 255 - (2 * (255 - k1.B) * (255 - k2.B)) / 255;
if (b < 0) b *= -1;
```

WYPALANIE :

```
if (255 - k2.R != 0)
{
   r = k1.R / (255 - k2.R);
}
else r = k1.R / 10;

if (255 - k2.G != 0)
{
   g = k1.G / (255 - k2.G);
}
else g = k1.G / 10;

if (255 - k2.B != 0)
```

```
                {
                   b = k1.B / (255 - k2.B);
                }
                else b = k1.B / 10;
```

REFLECT MODE:

```
                if (255 - k2.R == 0) r = 0;
                else r = (k1.R * k1.R) / (255 - k2.R);

                if (r > 255) r = 255;
                if (255 - k2.G == 0) g = 0;
                else g = (k1.G * k1.G) / (255 - k2.G);
                if (g > 255) g = 255;
                if (255 - k2.G == 0) b = 0;
                else b = (k1.B * k1.B) / (255 - k2.B);
                if (b > 255) b = 255;
```

KONTRAST:

```
private void button17_Click(object sender, EventArgs e)
      {
         Bitmap b1 = (Bitmap)pictureBox1.Image;
         Bitmap b2 = (Bitmap)pictureBox2.Image;
         Color k;
         int r, g, b;
         byte[] LUT = new byte[256];
         double a = 127;

         for (int i = 0; i < 256; i++)
         {
            if ((a * (i - 127) + 127) > 255)
            {
               LUT[i] = 255;
            }
            else if ((a * (i - 127) + 127) < 0)
            {
```

```
            LUT[i] = 0;
        }
        else
        {
            LUT[i] = (byte)(a * (i - 127) + 127);
        }
    }

    for (int x = 0; x < width; x++)
    {
        for (int y = 0; y < height; y++)
        {
            k = b1.GetPixel(x, y);
            r = k.R;
            g = k.G;
            b = k.B;

            r = LUT[r];
            g = LUT[g];
            b = LUT[b];

            b2.SetPixel(x, y, Color.FromArgb(r, g, b));
        }
    }
    pictureBox2.Refresh();

}
```

HISTOGRAM DLA OBRAZU:

```
private void button18_Click(object sender, EventArgs e)
        {
            chart1.Visible = true;
            Bitmap b1 = (Bitmap)pictureBox1.Image;
            red = new int[256];
            green = new int[256];
            blue = new int[256];
            Color k;
            for(int i = 0; i < width; i++)
            {
                for(int j = 0; j < height; j++)
                {
                    k = b1.GetPixel(i, j);
                    red[k.R]++;
                    green[k.G]++;
                    blue[k.B]++;
                }
            }


            chart1.Series["red"].Points.Clear();
            chart1.Series["green"].Points.Clear();
            chart1.Series["blue"].Points.Clear();
            for(int i = 0; i < 256; i++)
            {
                chart1.Series["red"].Points.AddXY(i, red[i]);
                chart1.Series["green"].Points.AddXY(i, green[i]);
                chart1.Series["blue"].Points.AddXY(i, blue[i]);
            }
            chart1.Invalidate();
        }
```

LUT dla wyrownania

```
private int[] calculateLUT(int[] values, int size)
{
   double minValue = 0;
   for (int i = 0; i < 256; i++)
   {
      if (values[i] != 0)
      {
         minValue = values[i];
         break;
      }
   }

   int[] result = new int[256];
   double sum = 0;
   for (int i = 0; i < 256; i++)
   {
      sum += values[i];
      result[i] = (int)(((sum - minValue) / (size - minValue)) * 255.0);
   }

   return result;
}
```

WYROWNANIE:

```
private void button19_Click(object sender, EventArgs e)
{
   Bitmap b1 = (Bitmap)pictureBox1.Image;
   Bitmap b2 = (Bitmap)pictureBox2.Image;
   Color k, k2;

   int[] LUTred = calculateLUT(red, width * height);
   int[] LUTgreen = calculateLUT(green, width * height);
```

```
    int[] LUTblue = calculateLUT(blue, width * height);

    red = new int[256];
    green = new int[256];
    blue = new int[256];

    for(int i = 0; i < width; i++)
    {
       for(int j = 0; j < height; j++)
       {
          k = b1.GetPixel(i, j);
          k2 = Color.FromArgb(LUTred[k.R], LUTgreen[k.G], LUTblue[k.B]);
          b2.SetPixel(i, j, k2);

          red[k2.R]++;
          green[k2.G]++;
          blue[k2.B]++;
       }
    }
    pictureBox2.Refresh();

    chart1.Series["red"].Points.Clear();
    chart1.Series["green"].Points.Clear();
    chart1.Series["blue"].Points.Clear();
    for (int i = 0; i < 256; i++)
    {
       chart1.Series["red"].Points.AddXY(i, red[i]);
       chart1.Series["green"].Points.AddXY(i, green[i]);
       chart1.Series["blue"].Points.AddXY(i, blue[i]);
    }
    chart1.Invalidate();
}
```

LUT DLA SKALOWANIA

```
private int[] calculateLUT2(int[] values)
{
   int minValue = 0;
   for (int i = 0; i < 256; i++)
   {
      if (values[i] != 0)
      {
         minValue = i;
         break;
      }
   }

   int maxValue = 255;
   for (int i = 255; i >= 0; i--)
   {
      if (values[i] != 0)
      {
         maxValue = i;
         break;
      }
   }

   int[] result = new int[256];
   double a = 255.0 / (maxValue - minValue);
   for (int i = 0; i < 256; i++)
   {
      result[i] = (int)(a * (i - minValue));
   }

   return result;
}
```

SKALOWANIE:

```
private void button20_Click(object sender, EventArgs e)
{
   Bitmap b1 = (Bitmap)pictureBox1.Image;
   Bitmap b2 = (Bitmap)pictureBox2.Image;
   Color k, k2;

   int[] LUTred = calculateLUT2(red);
   int[] LUTgreen = calculateLUT2(green);
   int[] LUTblue = calculateLUT2(blue);

   red = new int[256];
   green = new int[256];
   blue = new int[256];

   for(int i = 0; i < width; i++)
   {
      for(int j = 0; j < height; j++)
      {
         k = b1.GetPixel(i, j);
         k2 = Color.FromArgb(LUTred[k.R], LUTgreen[k.G], LUTblue[k.B]);

         b2.SetPixel(i, j, k2);
         red[k2.R]++;
         green[k2.G]++;
         blue[k2.B]++;
      }
   }
   pictureBox2.Refresh();

   chart1.Series["red"].Points.Clear();
   chart1.Series["green"].Points.Clear();
   chart1.Series["blue"].Points.Clear();
   for (int i = 0; i < 256; i++)
```

```
        {
            chart1.Series["red"].Points.AddXY(i, red[i]);
            chart1.Series["green"].Points.AddXY(i, green[i]);
            chart1.Series["blue"].Points.AddXY(i, blue[i]);
        }
        chart1.Invalidate();
    }
```

FILTRY ROBERTSA, PREWITTA, SOBELA, itd. :

```
private void button12_Click(object sender, EventArgs e)
    {
        // Przypisuję do zmiennych wartości kontrolek
        dane.M1 = (int)numericUpDown3.Value;
        dane.M2 = (int)numericUpDown4.Value;
        dane.M3 = (int)numericUpDown5.Value;

        dane.M4 = (int)numericUpDown6.Value;
        dane.M5 = (int)numericUpDown7.Value;
        dane.M6 = (int)numericUpDown8.Value;

        dane.M7 = (int)numericUpDown9.Value;
        dane.M8 = (int)numericUpDown10.Value;
        dane.M9 = (int)numericUpDown11.Value;

        Bitmap b1 = (Bitmap)pictureBox1.Image;
        Bitmap b2 = (Bitmap)pictureBox2.Image;

        Color k1, k2, k3, k4, k5, k6, k7, k8, k9;

        int suma_maski = 0;

        for (int i = 0; i < 3; i++)
        {
            for(int j = 0; j < 3; j++)
            {
```

```
            suma_maski += maska[i, j];
          }
        }

        for (int i = 1; i < width - 1; i++)
        {
         for (int j = 1; j < height - 1; j++)
         {
            k1 = b1.GetPixel(i - 1, j - 1);
            k2 = b1.GetPixel(i, j - 1);
            k3 = b1.GetPixel(i + 1, j - 1);

            k4 = b1.GetPixel(i - 1, j);
            k5 = b1.GetPixel(i, j);
            k6 = b1.GetPixel(i + 1, j);

            k7 = b1.GetPixel(i - 1, j + 1);
            k8 = b1.GetPixel(i, j + 1);
            k9 = b1.GetPixel(i + 1, j + 1);

            kk[0] = k1; kk[1] = k2; kk[2] = k3; kk[3] = k4; kk[4] = k5; kk[5] = k6; kk[6]
= k7; kk[7] = k8; kk[8] = k9;

            r = k1.R * dane.M1 + k2.R * dane.M2 + k3.R * dane.M3 + k4.R *
dane.M4 + k5.R * dane.M5 + k6.R * dane.M6 + k7.R * dane.M7 + k8.R * dane.M8 +
k9.R * dane.M9;
                    if (suma_maski != 0)
                    {
                       r = r / suma_maski; // normowanie
                    }
                    if (r > 255) r = 255; // obcinanie
                    if (r < 0) r = 0;

            g = k1.G * dane.M1 + k2.G * dane.M2 + k3.G * dane.M3 + k4.G *
dane.M4 + k5.G * dane.M5 + k6.G * dane.M6 + k7.G * dane.M7 + k8.G * dane.M8
+ k9.G * dane.M9;
```

```
            if (suma_maski != 0)
            {
               g = g / suma_maski; // normowanie
            }
            if (g > 255) g = 255; // obcinanie
            if (g < 0) g = 0;

            b = k1.B * dane.M1 + k2.B * dane.M2 + k3.B * dane.M3 + k4.B *
dane.M4 + k5.B * dane.M5 + k6.B * dane.M6 + k7.B * dane.M7 + k8.B * dane.M8 +
k9.B * dane.M9;
            if (suma_maski != 0)
               if (suma_maski != 0)
            {
               b = b / suma_maski; // normowanie
            }
            if (b > 255) b = 255; // obcinanie
            if (b < 0) b = 0;

            b2.SetPixel(i, j, Color.FromArgb(r, g, b));
         }
      }
      pictureBox2.Refresh();
```

FILTR STATYCZNY:

```
 private void button12_Click(object sender, EventArgs e)
    {
       dane.M1 = (int)numericUpDown3.Value;
        dane.M2 = (int)numericUpDown4.Value;
        dane.M3 = (int)numericUpDown5.Value;

        dane.M4 = (int)numericUpDown6.Value;
        dane.M5 = (int)numericUpDown7.Value;
        dane.M6 = (int)numericUpDown8.Value;

        dane.M7 = (int)numericUpDown9.Value;
        dane.M8 = (int)numericUpDown10.Value;
        dane.M9 = (int)numericUpDown11.Value;

        Bitmap b1 = (Bitmap)pictureBox1.Image;
        Bitmap b2 = (Bitmap)pictureBox2.Image;

        Color k1, k2, k3, k4, k5, k6, k7, k8, k9;
        int keepR, keepG, keepB;
        Color[] colors = new Color[9];

        for (int i = 1; i < width - 1; i++)
        {
           for (int j = 1; j < height - 1; j++)
           {
              k1 = b1.GetPixel(i - 1, j - 1);
              k2 = b1.GetPixel(i, j - 1);
              k3 = b1.GetPixel(i + 1, j - 1);

              k4 = b1.GetPixel(i - 1, j);
              k5 = b1.GetPixel(i, j);
              k6 = b1.GetPixel(i + 1, j);
```

```
            k7 = b1.GetPixel(i - 1, j + 1);
            k8 = b1.GetPixel(i, j + 1);
            k9 = b1.GetPixel(i + 1, j + 1);

            colors[0] = k1; colors[1] = k2; colors[2] = k3; colors[3] = k4; colors[4] =
k5; colors[5] = k6; colors[6] = k7; colors[7] = k8; colors[8] = k9;

            // Ustawiam na początkowe wartości w celu obliczeń minimalnej i
maksymalnej wartości
            keepR = colors[0].R; keepG = colors[0].G; keepB = colors[0].B;

            . // (miejsce na wykonywane operacje)
            .
            .
            .
            b2.SetPixel(i, j, Color.FromArgb(keepR, keepG, keepB));
          }
        }
        pictureBox2.Refresh();
```

Filtr maksymalny

```
        // filtr maks
            for(int x = 1; x < colors.Length; x++)
            {
              if (colors[x].R > keepR) keepR = colors[x].R;
              if (colors[x].G > keepG) keepG = colors[x].G;
              if (colors[x].B > keepB) keepB = colors[x].B;
            }
```

Filtr minimalny

```
// filtr min
        for (int y = 1; y < colors.Length; y++)
        {
           if (colors[y].R < keepR) keepR = colors[y].R;
           if (colors[y].G < keepG) keepG = colors[y].G;
           if (colors[y].B < keepB) keepB = colors[y].B;
        }
```

Filtr medianowy

```
// mediana
        List redList = new List();
        List greenList = new List();
        List blueList = new List();

        for (int z = 0; z < colors.Length; z++)
        {
           redList.Add(colors[z].R);
           greenList.Add(colors[z].G);
           blueList.Add(colors[z].B);
        }

        redList.Sort();
        greenList.Sort();
        blueList.Sort();

        keepR = redList[4];
        keepG = greenList[4];
        keepB = blueList[4];
```

INNOWACYJNOSC:

Metoda wykonująca dany algorytm po naciśnięciu przycisku

```
    private void button13_Click(object sender, EventArgs e)
  {
    Bitmap b1 = (Bitmap)pictureBox1.Image;
    Bitmap b2 = (Bitmap)pictureBox2.Image;
    Color k, k1, k2;
    int r, g, b;
    for (int i = 0; i < width; i++)
    {
      for (int j = 0; j < height; j++)
      {
        k1 = b1.GetPixel(i, j);
        k2 = b2.GetPixel(i, j);

        . // (miejsce na algorytm)
        .
        .
        .
      }
     pictureBox2.Refresh();
    }
```