WS 2019/20	Exercises Digital Image Processing	Solution Exercise No. 14

Color images

Objective of exercises:

Allgemein

- Segmentation of RGB and HSV images
- Conversion of RGB to HSV

Solution for task 14.1a

Task: *Write a function segmenting an RGB image into its channels.*

To separately display the individual color channels of an RGB input image, each channel has to be assigned separately to the image values of the output image. The resulting image is either a gray scale image or an RGB image with the same values for R, G and B.

Determination of the interface:

```
// This function segments the input image <input> into the individual RGB
// color channels, depending on the <type> and writes the result into <output>.
void createChannelRGB (RgbImage& input, GrayImage& output, ColorTypeRGB type)
```

Implementation:

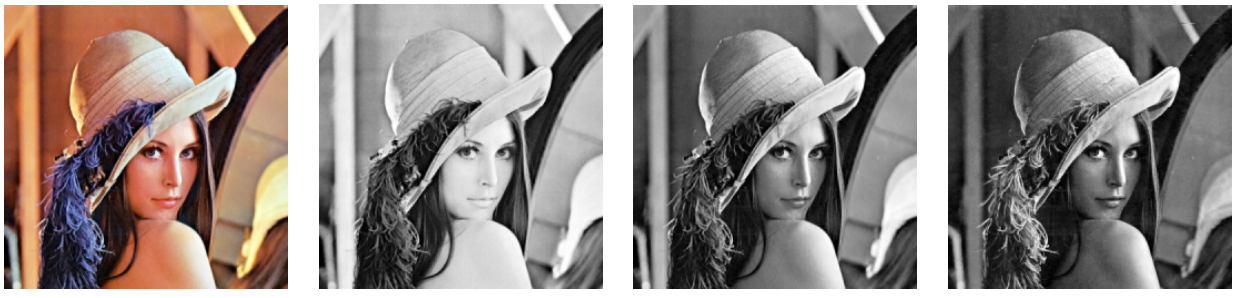
```
//
//
//
enum ColorTypeRGB {RED, GREEN, BLUE};

//
//
//
unsigned char getColorRGB (Rgb color, ColorTypeRGB type)
{
    switch (type)
    {
        case RED:   return color.r;
        case GREEN: return color.g;
        case BLUE:  return color.b;
        default:    return 0;
    }
}

//
//
//
void createChannelRgb (RgbImage& input, GrayImage& output, ColorTypeRGB type)
{
    Rgb   *idata = input.getData();
    float *odata = output.getData();

    for (int i=0; i<input.getSize(); i++)
    {
        odata[i] = (float)getColorRGB(idata[i], type);
    }
}

// ------------------------------------------------------------------------------
```



(a) (b) (c) (d)

Figure 1: RGB Eingabebild (a), zerlegt in R, G und B (b, c, d)

Solution for task 14.1b

Task: *Implements the conversion of RGB to HSV through the given algorithm.*

Determination of the interface:

```
// This function converts an RGB image <input> into an HSV image <output>.
void Rgb2Hsv (RgbImage& input, HsvImage& output)
```

Implementation:

```
void Rgb2Hsv (RgbImage& input, HsvImage& output)
{
    Rgb* idata = input.getData();
    Hsv* odata = output.getData();

    for (int i=0; i<input.getSize(); i++)
    {
        float h, s, v;

        float r    = (float)idata[i].r / 255.0;
        float g    = (float)idata[i].g / 255.0;
        float b    = (float)idata[i].b / 255.0;

        float maxi = max(r,g,b);
        float mini = min(r,g,b);

        v = maxi;

        if (maxi != 0)
        {
            s = 1.0 - mini/maxi;

            if (s != 0) {
                if (maxi == r) h  =   0 + 60 * (g-b) / (maxi-mini);
                if (maxi == g) h  = 120 + 60 * (b-r) / (maxi-mini);
                if (maxi == b) h  = 240 + 60 * (r-g) / (maxi-mini);
                if (h<0)       h += 360;

                h = h / 2.0;
            }
            else
            {
                h = 0.0;
            }
        }
        else
        {
            h = 0.0;
            s = 0.0;
        }

        s *= 255.0;
        v *= 255.0;

        odata[i].h = (unsigned char)h;
        odata[i].s = (unsigned char)s;
        odata[i].v = (unsigned char)v;
    }
}

// ------------------------------------------------------------------------------

//
//
//
enum ColorTypeHSV {HUE, SATURATION, VALUE};

//
//
//
```

Solution for task 14.1c

Task: *Writes a function which converts an RGB image to HSV by using the function described in exercise A and then segments the HSV image into its channels.*

The process can be compared with exercise a) but first converting RGB to HSV from exercise b).

Determination of the interface:

```
// This function segments the input image <input> into the indivdidual HSV
// parts, depending on the <type> and writes the result into <output>.
void createChannelHSV (HsvImage& input, RgbImage& output, ColorTypeHSV type)
```

Implementation:

```
//
//
//
void createChannelHsv (HsvImage& input, GrayImage& output, ColorTypeHSV type)
{
    Hsv   *idata = input.getData();
    float *odata = output.getData();

    for (int i=0; i<input.getSize(); i++)
    {
        odata[i] = (float)getColorHSV(idata[i], type);
    }
}
```

(a) (b) (c) (d)

Figure 2: RGB input image (a), converted to HSV, segmented into H, S and V (b, c, d)

Main Program:

```
int main (int argc, char** argv)
{
    /* ********************************** */
    /* Exercise 13                        */
    /* ********************************** */

    RgbImage input;
    input.load();
    input.show();

    int width  = input.getWidth();
    int height = input.getHeight();

    cout << "Option:\n";
    cout << "1 - RGB Bild zerlegen\n";
    cout << "2 - RGB in HSV verwandeln und zerlegen\n";

    int option;
    cin >> option;
```

```
    switch (option)
    {
        case 1:
        {
            GrayImage output1(width, height);
            GrayImage output2(width, height);
            GrayImage output3(width, height);

            createChannelRgb(input, output1, RED);
            createChannelRgb(input, output2, GREEN);
            createChannelRgb(input, output3, BLUE);

            output1.show();
            output2.show();
            output3.show();

            break;
        }
        case 2:
        {
            GrayImage output1(width, height);
            GrayImage output2(width, height);
            GrayImage output3(width, height);

            HsvImage inputHsv(width, height);

            Rgb2Hsv(input, inputHsv);

            createChannelHsv(inputHsv, output1, HUE);
            createChannelHsv(inputHsv, output2, SATURATION);
            createChannelHsv(inputHsv, output3, VALUE);

            output1.show();
            output2.show();
            output3.show();

            break;
        }
        default:
        {
            exit(0);
        }
    }

    cout << "FINISHED.\n";
    return 1;
}
```