

HÖHERE TECHNISCHE BUNDESLEHRANSTALT

HOLLABRUNN

Höhere Abteilung für Elektronik – Technische Informatik

Klasse/ Jahrgang: 5BHEL	Gruppe: 2b	Übungsleiter: Prof. Wihsböck
Übungsnummer: II/6	Übungstitel: Bildverarbeitung	
Datum der Übung: 24.11.2015	Teilnehmer: Graussam, Holzinger, Mischling, Schwarzmann	
Datum der Abgabe: 15.12.2015	Schrittführer: Schwarzmann Philipp	Unterschrift:

	Beurteilung
Aufgabenstellung	
Dokumentation	
Messschaltungen	
Messtabellen	
Berechnungen	
Programmlistings	
Auswertung	
Diagramme	
Berechnungen	
Simulationen	
Schlussfolgerungen	
Kommentare	
Inventarliste	
Messprotokoll	
Form	
Summe	

Allgemeiner Teil:

Titel der Übung: Bildverarbeitung
Datum der Übung: 24.11.2015
Klasse: 5BHEL
Übungsleiter: Prof. Wihsböck
Gruppe: 2b
Übungsteilnehmer: Graussam, Holzinger, Mischling, Schwarzmann
Schriftführer: Schwarzmann

Inhaltsverzeichnis

1.	Bildverarbeitung in Eclipse.....	3
1.1.	Projekt in Eclipse	3
1.2.	Programmieraufgabenstellungen.....	3
1.2.1.	Invertierung	3
1.2.2.	Helligkeit erhöhen	5
1.2.3.	Grenzwert	6
1.2.4.	Medianfilter	7
1.2.5.	Linearer Filter.....	8
1.3.	Weitere Anwendungen von Bildverarbeitung	9
1.3.1.	Zählen von Objekten	9
1.3.2.	Messen von Objekten	10
1.3.3.	Flächenmessung.....	11
2.	Bildverarbeitung mit RaspberryPi.....	12
2.1.	LED mit Simulink blinken lassen.....	12
3.	Erkenntnisse.....	12

1. Bildverarbeitung in Eclipse

1.1. Projekt in Eclipse

Zu Beginn der Übung wurde vom Übungsleiter eine .zip-Datei mit einem fertigen Eclipse-Projekt zur Verfügung gestellt. Nach dem Starten von Eclipse wurde ein neuer Workspace erstellt und der Ordner aus der .zip-Datei in den Workspace kopiert um das Projekt zu importieren. Anschließend wurde der Launcher konfiguriert:

Rechtsklick auf Projekt → Run As → Run Configuration → Arguments → In das untere Fenster die oberste Zeile aus der Datei JVM-Arguments.txt kopieren.

ImageJ wird in Java geschrieben.

1.2. Programmieraufgabenstellungen

Für jede Aufgabenstellung wurde eine neue Klasse erstellt welche nach dem starten von ImageJ unter Plugins ausgewählt werden kann. Zum Erstellen einer neuen Klasse muss man folgendes machen:

Rechtsklick auf (default package) → New → Class

Bei der Namensvergabe muss darauf geachtet werden, dass ein Underline-Zeichen im Namen vorkommt, was normalerweise vermieden werden sollte bei ImageJ aber notwendig ist, dass die Klasse erkannt wird. Anschließend muss noch mit klicken auf den Button „Add“ ein Plugin hinzugefügt werden. Es wurde „PlugInFilter“ ausgewählt.

1.2.1. Invertierung

Die erste Aufgabenstellung war es, eine Invertierungsfunktion zu programmieren. Die Bilder mit denen gearbeitet wurde waren 8 Bit Graustufenbilder. Zur Berechnung der neuen Bildpunkte muss nur 255 minus dem aktuellen Wert gerechnet werden.

```
public class PointOp_Inverter implements PlugInFilter
{

    @Override
    public int setup(String arg, ImagePlus imp)
    {
        return DOES_8G;
    }

    @Override
    public void run(ImageProcessor ip)
    {
        int w = ip.getWidth();
        int h = ip.getHeight();

        for(int x=0; x<w; x++)
        {
            for(int y=0; y<h; y++)
            {
                int pixelValue = ip.getPixel(x, y);
                int newPixelValue = 255 - pixelValue;
                ip.putPixel(x, y, newPixelValue);
            }
        }
    }
}
```

Der Returnwert DOES_8G ist für 8 Bit Graustufenbilder. Die Methoden getWidth und getHeight geben jeweils die Breite und Höhe des Bildes zurück. Es wird in zwei Schleifen jeder Bildpunkt durchgegangen. Mit der Methode getPixel wird der Wert des Pixels mit den entsprechenden Bildkoordinaten zurückgegeben und mit putPixel wird ein neuer Wert für den entsprechenden Pixel festgelegt.

Beim Starten von ImageJ muss zuerst ein Bild mit File → Open geöffnet werden und anschließend kann unter Plugins die Entsprechende Funktion ausgewählt werden. Vor dem Schließen von ImageJ muss darauf geachtet werden, dass das geöffnete Bild geschlossen wird. Vergisst man auf das Schließen des Bildes muss man unter Window → Show View die Console geöffnet und der Prozess beendet werden.

1.2.2. Helligkeit erhöhen

Die zweite Aufgabenstellung war es die Helligkeit zu erhöhen. Dazu wurde wieder eine neue Klasse erstellt.

```
public class PointOp_Brightness implements PlugInFilter
{
    @Override
    public int setup(String arg, ImagePlus imp)
    {
        // TODO Auto-generated method stub
        return DOES_8G;
    }

    @Override
    public void run(ImageProcessor ip)
    {
        // TODO Auto-generated method stub
        int w = ip.getWidth();
        int h = ip.getHeight();

        for(int x=0; x<w; x++)
        {
            for(int y=0; y<h; y++)
            {
                int pixelValue = ip.getPixel(x, y);
                int newPixelValue = pixelValue + pixelValue / 2;
                if(newPixelValue < 255)
                    ip.putPixel(x, y, newPixelValue);
                else
                    ip.putPixel(x, y, 255);
            }
        }
    }
}
```

Das Bild wurde wieder Pixel für Pixel durchgegangen und jeder Pixel eigens berechnet. Der neue Pixelwert ergab sich aus 1.5 mal dem ursprünglichen Wert.

1.2.3. Grenzwert

Die letzte Aufgabenstellung für Punktoperationen war es anhand eines Grenzwertes das Bild in ein schwarz-weiß-Bild umzurechnen.

```
public class PointOp_Grenzwert implements PlugInFilter
{
    /**
     * Setzen des Grenzwertes
     */
    public static final int GRENZWERT = 128;

    @Override
    public int setup(String arg, ImagePlus imp) {
        // TODO Auto-generated method stub
        return DOES_8G;
    }

    @Override
    public void run(ImageProcessor ip)
    {
        // TODO Auto-generated method stub
        int w = ip.getWidth();
        int h = ip.getHeight();

        for(int x=0; x<w; x++)
        {
            for(int y=0; y<h; y++)
            {
                int pixelValue = ip.getPixel(x, y);
                int newPixelValue = 0;

                if(pixelValue > GRENZWERT)
                    newPixelValue = 255;

                ip.putPixel(x, y, newPixelValue);
            }
        }
    }
}
```

Es wurde wieder jeder Pixel durchgegangen und der Wert des Pixels mit dem Grenzwert verglichen. War der Wert größer als der Grenzwert wurde der Wert auf 255 (weiß) gesetzt, andernfalls auf 0 (schwarz).

1.2.4. Medianfilter

Die nächste Aufgabenstellung war es einen Medianfilter zu programmieren. Bei einem Medianfilter werden die Werte des zu bearbeitenden Pixels und die der umliegenden Pixel gespeichert, sortiert und der Wert in der Mitte als neuer Wert des Pixels übernommen.

```
public class Filter_Median implements PlugInFilter
{
    @Override
    public int setup(String arg, ImagePlus imp)
    {
        // TODO Auto-generated method stub
        return DOES_8G;
    }

    @Override
    public void run(ImageProcessor ip)
    {
        // TODO Auto-generated method stub
        ImageProcessor copy = ip.duplicate();
        int [] filterArray = new int[9];
        int w = ip.getWidth();
        int h = ip.getHeight();
        int index;

        for(int x=1; x<w; x++)
        {
            for(int y=1; y<h; y++)
            {
                index = 0;
                for(int x_new = x-1; x_new <= x+1; x_new ++)
                {
                    for(int y_new = y-1; y_new <= y+1; y_new ++)
                    {
                        filterArray[index] = copy.getPixel(x_new, y_new);
                        index ++;
                    }
                }
                Arrays.sort(filterArray);

                ip.putPixel(x, y, filterArray[4]);
            }
        }
    }
}
```

Zur Vereinfachung wurden die äußeren Pixel nicht gefiltert. Der zu bearbeitende Pixel und die umliegenden Pixel wurden in einem Array gespeichert und mit der Methode Arrays.sort sortiert. Zum Schluss wurde der mittlere Wert des Arrays als neuer Wert übernommen. Der Medianfilter ist gut dafür geeignet das Sogenannte „Salz- und Pfefferrauschen“ zu unterdrücken. In ImageJ kann dieses Rauschen mit Process → Noise → Salt and Pepper hinzugefügt werden.

1.2.5. Linearer Filter

Die letzte Filteraufgabe bestand darin einen linearen Filter zu testen. Das File mitsamt Programmcode wurde von Übungsleiter zur Verfügung gestellt. Durch das Umschreiben des zweidimensionalen Arrays mit den Wertungen der einzelnen Pixel konnte umgeschrieben werden um z.B. einen Rechteckfilter oder Gaußfilter gemacht werden.

```
public class Filter_ArbitraryMxN implements PlugInFilter
{
    public int setup(String arg, ImagePlus imp)
    {
        return DOES_8G;
    }

    public void run(ImageProcessor original)
    {
        int w = original.getWidth();
        int h = original.getHeight();

        //filter matrix
        int[][] filter =
        {
            {0,1,2,1,0},
            {1,3,5,3,1},
            {2,5,9,5,2},
            {1,3,5,3,1},
            {0,1,2,1,0}
        };

        int calcSum = calcFilterSum(filter);

        double s = 1.0 / calcSum;

        int K = filter[0].length/2;
        int L = filter.length/2;

        ImageProcessor copy = original.duplicate();

        for (int y = L; y <= h - L - 1; y++)
        {
            for (int x = K; x <= w - K - 1; x++)
            {
                // compute filter result for position (x,y)
                int sum = 0;
                for (int j = -L; j <= L; j++) {
                    for (int i = -K; i <= K; i++) {
                        int p = copy.getPixel(x + i, y + j);
                        int c = filter[j + L][i + K];
                        sum = sum + c * p;
                    }
                }
                int q = (int) (s * sum);

                // clamp result
                if (q < 0) q = 0;
                if (q > 255) q = 255;
                original.putPixel(x, y, q);
            }
        }
    }
}
```

```

private int calcFilterSum(int filter[][])
{
    int sum=0;

    for(int i=0;i<filter.length;i++)
    {
        for(int j=0;j<filter[i].length;j++)
        {
            sum = sum+filter[i][j];
        }
    }

    return sum;
}
}

```

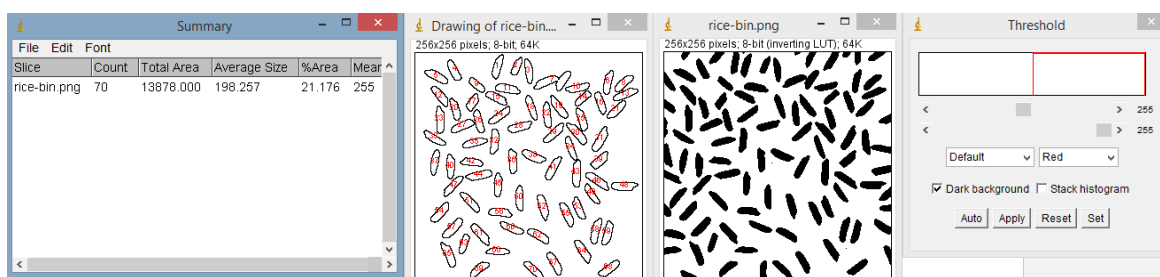
1.3. Weitere Anwendungen von Bildverarbeitung

Für die Weiteren Aufgabenstellungen wurden die verschiedensten Funktionen von ImageJ verwendet um Objekte zu zählen oder zu Vermessen.

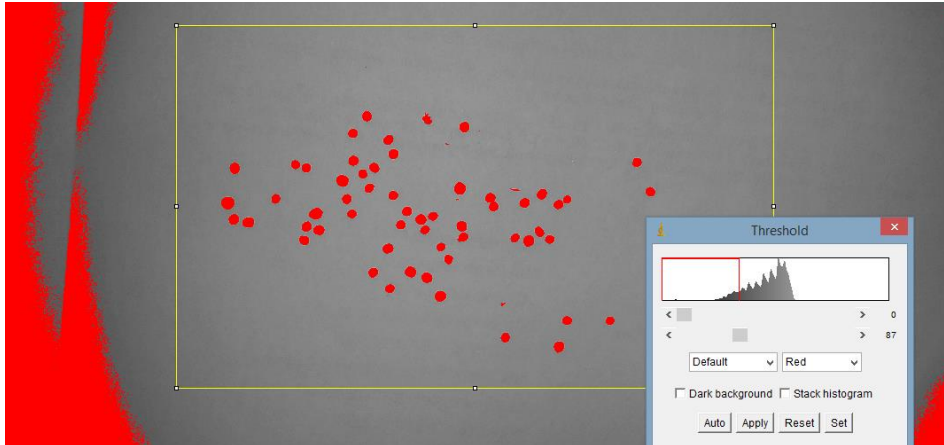
1.3.1. Zählen von Objekten

Vom Übungsleiter wurde ein schwarz-weiß-Bild mit Reiskörnern zur Verfügung gestellt, welche mithilfe von ImageJ gezählt werden sollten.

Zuerst wurde der Bildtyp auf 8 Bit Graustufen eingestellt. Dies erfolgte einem Klick auf das Image-Menü und anschließendem Auswählen des gewünschten Bildtyps unter „Type“. Anschließend wurde unter Image → Adjust → Threshold Ausgewählt, was gezählt werden sollte. Bei dem Bild wurde nach auswählen des Punktes „dark Background“ alle weißen Reiskörner rot markiert. Anschließend wurde die Auswahl mit „Apply“ bestätigt. Um die Objekte zu zählen wurde unter Analyse → Analyse Particles die Größe der Elemente eingestellt. Bei der Übung wurde die Mindestgröße auf 20 Pixel festgelegt um einzelne schwarze Punkte auszuschließen. Mit „Exclude Edges“ werden Objekte am Rand des Bildes nicht mitgezählt. Mit „Show Outlines“ werden die Umrisse der Objekte angezeigt.



Anschließend wurden vom Übungsleiter Rapskörner ausgeteilt, welche auf die gleiche Art gezählt werden sollten. Dazu wurde ein Foto von den Rapskörnern auf einem hellen Untergrund gemacht und dieses Foto im Anschluss bearbeitet. Nach öffnen des Bildes wurde ein rechteckiges Fenster um die Körner gezogen, um Schatten am Rand des Bildes nicht mitzuzählen. Es wurden die gleichen Einstellungen getroffen wie zuvor nur, dass zusätzlich unter „Analyse Particles“ eine „Circularity“ eingestellt wurde, da die Körner annähernd rund waren.

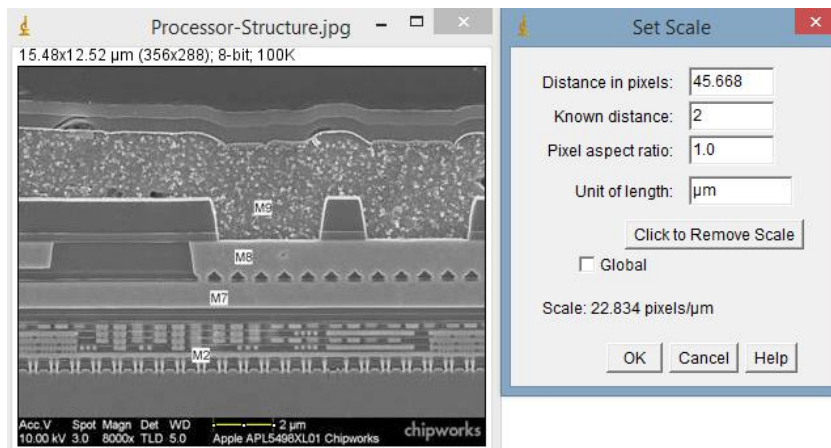


Bei der Zählung wurden 39 Rapskörner gezählt.

1.3.2. Messen von Objekten

Als nächstes sollte eine Streckenmessung anhand eines Bildes eines Prozessordurchschnittes durchgeführt werden.

Dafür wurde eine Linie auf den Maßstab des Bildes gezogen. Mit Analyse → Set Scale wird diese Linie als Referenzwert für die Messungen angenommen. In diesem Fenster kann der Wert, dem diese Linie entspricht festgelegt werden. Des Weiteren kann auch eine Einheit dafür eingestellt werden. Wenn nach den Einstellungen wieder eine Linie gezogen wird, wird die Länge automatisch auf die entsprechende Länge umgerechnet und unter den Werkzeugen der Wert angezeigt.

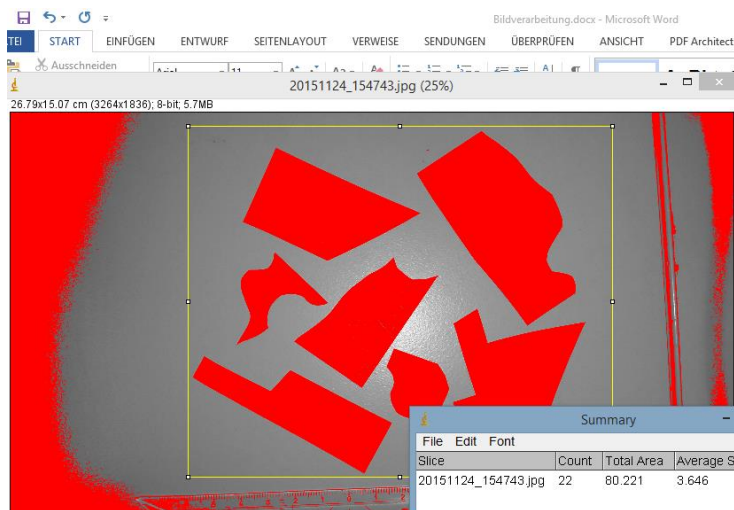


1.3.3. Flächenmessung

Die letzte Aufgabe mit ImageJ war es bestimmte Flächen auf Bildern zu berechnen. Dazu wurden vom Übungsleiter rote Zettel ausgeteilt die zerschnitten wurden und dann ein Foto davon gemacht wurde. Als Maßstab wurde ein Lineal mitfotografiert.

Die Referenzlänge wurde wie bei der vorangegangenen Aufgabenstellung eingestellt. Des Weiteren wurde abermals der Threshold eingestellt und die zu Messenden Objekte mit einem Rechteck markiert um Schatten am Rand auszuschließen. Wenn das Rechteck-Tool ausgewählt ist, kann man mit „Alt“ + Ziehen einen Bereich markieren, der nicht mitberechnet werden soll. Unter Analyse → Set Measurements muss anschließend noch „Area“ ausgewählt werden.

Die Messung wurde mit einem 1cm und einem 10cm Maßstab durchgeführt. Die berechnete Fläche mit dem 1cm Maßstab betrug 81.3cm² und mit dem 10cm Maßstab 79.6cm².

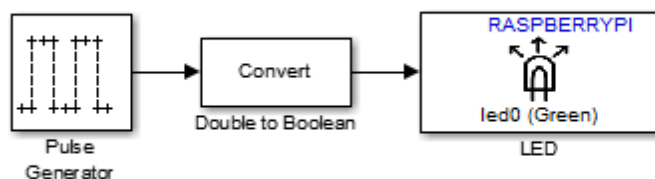


2. Bildverarbeitung mit RaspberryPi

2.1. LED mit Simulink blinken lassen

Der letzte Teil der Übung bestand darin, mithilfe von Simulink eine LED am Board des Raspberry blinken zu lassen.

In Simulink wurde ein Pulsgenerator mit einem Conversion-Block aufgebaut, der den Datentyp umwandelt und das Signal weiter an einen Funktionsblock gibt, der die LED ansteuert. Der Raspberry wurde mit einem Netzkabel mit dem Laptop verbunden. Um die Funktion auf den Raspberry übertragen zu können musste zuerst unter Tools → Run on Target Hardware → Options die Hardware sowie der Port eingestellt werden. Des Weiteren wurde die IP-Adresse sowie Username, Passwort und Pfad angegeben werden. Anschließend konnte die Funktion mit „Run“ übertragen werden.



Bildverarbeitung mit dem Raspberry und Simulink konnte nicht mehr durchgeführt werden, da die Zeit zu knapp war.

3. Erkenntnisse

In der Übung wurden die Grundlegenden Dinge über Bildverarbeitung gelernt. Des Weiteren wurde der Umgang mit ImageJ gelernt sowie das benutzen einiger wichtigen Features. Außerdem wurde das Objektorientierte Programmieren durch das Schreiben der Bildverarbeitungsprogramme aufgefrischt.