Spalatin-Facsimile
–
Bildnachbearbeitung von Doppelseitenscans

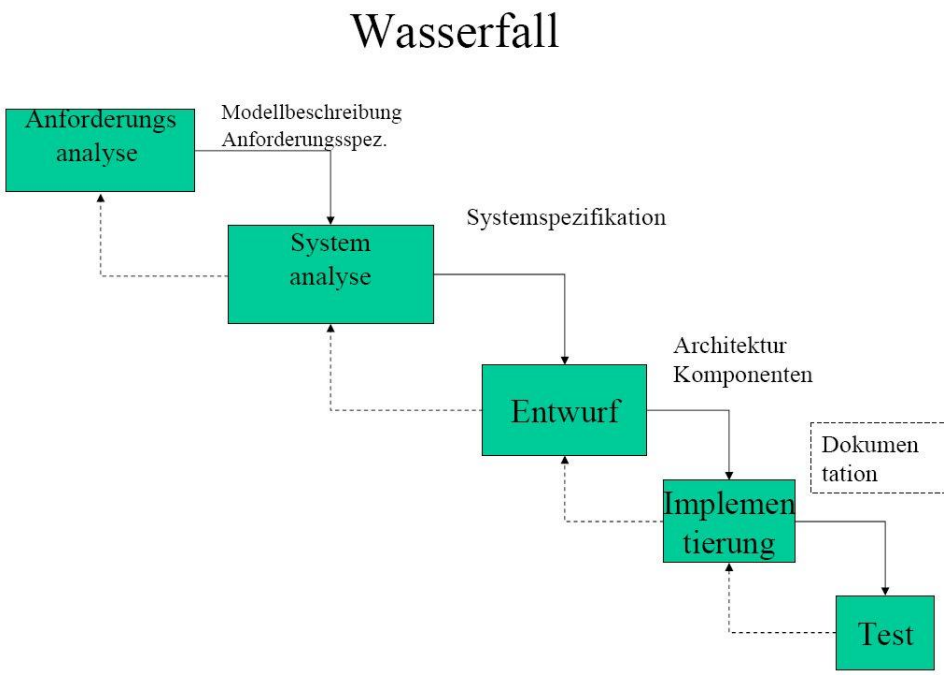
Ergebnisse des Projekts aus dem SS 2006

Projektteilnehmer:
Doris Aschenbrenner, Alexandra Secas, Jan Wiefel

30. November 2007

Inhalt

1. Thema
2. Ausgangssituation
3. Ziele
4. Recherche
5. Umsetzung
6. Fazit



Aufgabenstellung

- Eingabe:
  (Foto-) Scans von alten Büchern
- Problem:
  Verlauf von Texten und Bildern schräg zur Falz
- Aufgabe:
  Automatische Nachbearbeitung um Text und Bild gerade auszurichten

Verfeinerung der Aufgabenstellung

Entstehung der Scans

- Aufschlagen des Buches auf einer speziellen Wippe
- Abfotografieren der einzelnen Doppelseiten
- Entstehung von Grafiken im TIF-Format mit einer Auflösung von ca. 100 Megapixel und einer Genauigkeit von 400 dpi

Aufgabenstellung

- Bearbeitung der Grafiken durch automatisierten Prozess
- Gefordertes Ergebnis
  - Buchseiten ohne perspektivische Verzerrung
  - Skalierung auf Ursprungsgröße
  - Digital Iron (Name des Projekts)

Inhalt

1. Thema
2. Ausgangssituation
3. Ziele
4. Recherche
5. Umsetzung
6. Fazit

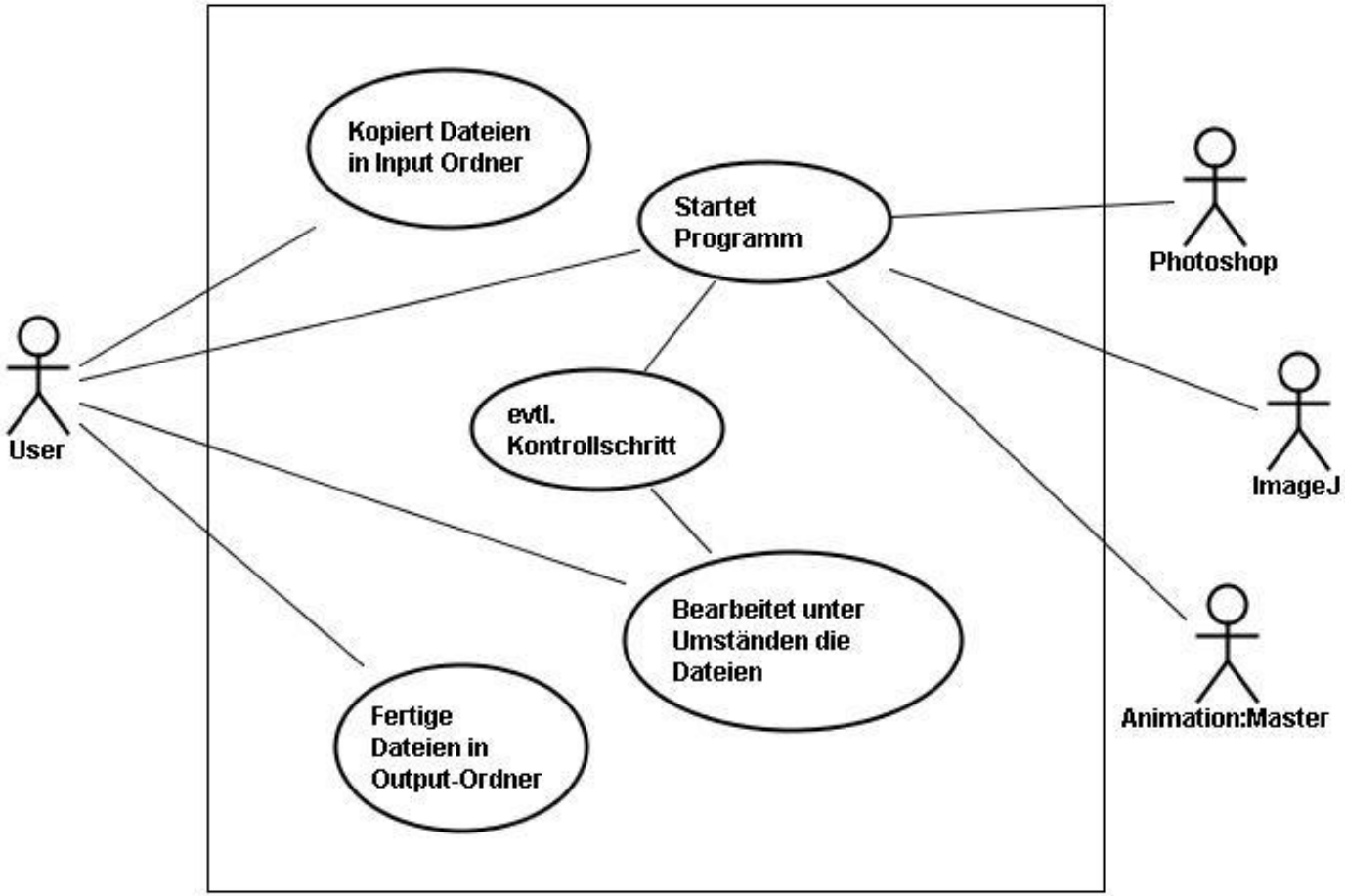
Ausgangssituation – Phase: Planung

- Testen einer Vielzahl unterschiedlicher Grafikprogramme
  - Übernehmen von Teilaufgaben
  - Möglichkeit der Plug-in-Anfertigung
  - Überprüfung von Lizenzen
  - Schnittstellen zwischen Programmen
- Übergeordnetes (ausführendes) Tool zur Koordinierung der Schnittstellen
- Erstellen des Lastenheftes auf Basis der Ausgangssituation

Lastenheft – Phase: Planung

- Bestimmung der Basisanforderungen
  - Ziel:
    Bessere Erfassung der Doppelseitenscans als bisher
  - Einsatz:
    Online-Projekte der Universität Würzburg
  - Anwendungsbereich:
    Spalatin-Bände und andere (foto-) gescannte Bücher

Use Case Diagram – Phase: Planung



Inhalt

1. Thema
2. Ausgangssituation
3. Ziele
4. Recherche
5. Umsetzung
6. Fazit

Ziele – Phase: Analyse

- Erstellen eines vollständigen, konsistenten und eindeutigen Produktmodells
- Erste Version des Pflichtenheftes
  - Ziele
    - Musskriterien
    - Wunschkriterien
  - Einsatz
  - Umgebung (Hardware, Software)
  - Funktionalität
  - Daten
  - Leistungen
  - Benutzungsoberfläche
  - Qualitätsziele

Ziele (Musskriterien) – Phase: Analyse

Musskriterien

- Aus Bearbeitung der Doppelseiten:
  - Entfernung der durch Prozess des Einscannens entstandenen Veränderungen (so gut wie möglich)
    - Krümmung der Einzelseiten
    - Schattierung an der Falz
  - Ausgabe rechteckiger Einzelseiten in ursprünglichem Format
  - Bessere Erfassung als bisher unter

    http://spalatin.informatik.uni-wuerzburg.de
- Gute Handhabung des Programms unter Zuhilfenahme des Benutzerhandbuchs

Ziele (Wunschkriterien) – Phase: Analyse

Wunschkriterien

- Wenige Eingriffe durch den Benutzer notwendig
- Individuelle Einstellungsmöglichkeiten für erfahrene Nutzer
- Einsatz von Windows (Einsatzgebiet v.a. Bibliotheksrechner)

Ziele – Phase: Analyse

- Anwendungsbereiche
  - Wiederverwendung für andere gescannte Handschriften
  - Verarbeitung der *Spalatin*-Bände
- Nutzergruppe
  - Lehrstuhl für Informatik II der Universität Würzburg
  - Mitarbeiter der Universitätsbibliothek

Abgabepacket – Phase: Analyse

- Aus Erweiterung des Lastenhefts entstandenes *Pflichtenheft*
- *Programm*, welches den Anforderungen des Pflichtenhefts genügt (inklusive Quelltext Dateien)
- *Handbuch* mit Erklärung der Programmfunktion und der Einstellungsmöglichkeiten für erfahrene Nutzer
- Aus Programm extrahierte *Javadoc* Dokumentation

Inhalt

1. Thema
2. Ausgangssituation
3. Ziele
4. Recherche
5. Umsetzung
6. Fazit

Recherche – Phase: Analyse

- Aufteilung des gegebenen Arbeitsprozesses in:
  - Finden der Falz (Teilen der Doppelseiten)
  - Extraktion der Seiten
  - Glätten der Seiten
  - Skalierung auf Ursprungsgröße

Grafikprogramme – Phase: Entwurf

- Auswahl möglicher Programme
  - Animation:Master
  - Photoshop
  - ImageJ

Grafikprogramme – Phase: Entwurf

- Animation:Master
  - Automatische Erkennung des dreidimensionalen Gitters schwierig
  - Nutzung einer Standardvorlage unmöglich
  - Ungeeignet
- Photoshop
  - Gute Möglichkeit zur Extraktion der Einzelseiten
  - Umfangreiche Werkzeuge
  - Keine Lizenz für Plug-in-Herstellung
- ImageJ
  - Einfache Möglichkeit des Einbindens von Plug-ins
  - Programmiersprache Java (Erfahrung aus Programmierpraktikum)

Schnittstellen – Phase: Entwurf

- Nutzung von mehreren Programmen erfordert Schnittstellen
  - Festlegen der Ausgabeformate der Zwischenschritte
  - Eigenschaften der temporäre Dateien zur Weiterverarbeitung
  - Vereinheitlichung des Zugriffs mittels Input, Temp und Output Ordnern
- Kombination der verschiedenen Arbeitsschritte
  - Modul zur Automatisierung von mausgesteuerten Prozessen in Windows: GuiTester

Inhalt

1. Thema
2. Ausgangssituation
3. Ziele
4. Recherche
5. Umsetzung
6. Fazit

Arbeitsschritte – Phase: Entwurf

- Finden der Falz (Teilen der Doppelseiten)
  - Nutzung des Sobel-Operators und der Hough-Transformation
- Extraktion der Seiten
  - Nutzung von Plug-ins in Photoshop
- Glätten der Seiten
  - Einsatz von linearer Regression zur besseren Abschätzung des oberen bzw. unteren Seitenrandes
- Skalierung auf Ursprungsgröße

Sobel-Operator – Phase: Entwurf

- Sobel-Operator ist Kantendetektionsfilter
- Möglichkeit der Verstärkung von Strukturen in Bildern
  - Gewinnung der horizontalen und vertikalen Richtungsinformation
  - Kombination der Richtungsinformation
- Faltung des Originalbilds bzw. der einzelnen Pixel der Reihe nach mit einer 3 x 3-Matrix um Richtungsinformationen herauszufiltern

Ergebnisse des Sobel-Operators

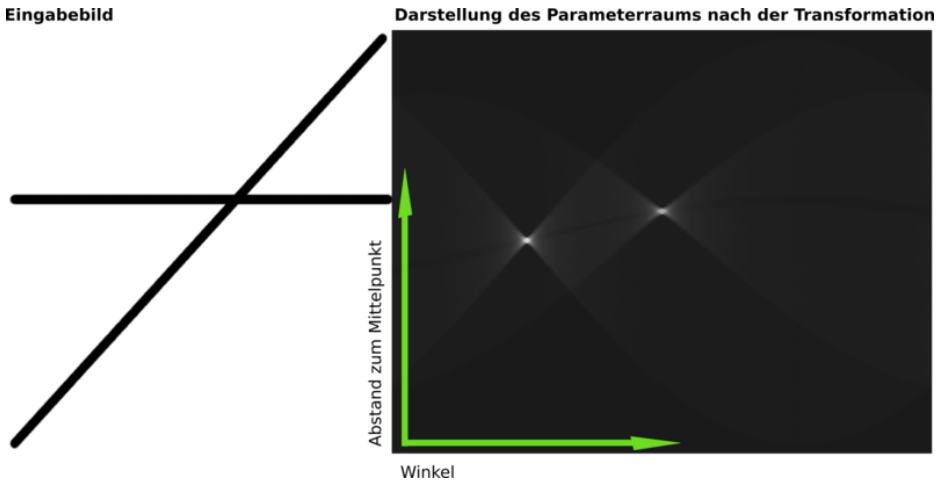
- Vertikale Faltung
- Horizontale Faltung

- Kombination der beiden Richtungsinformationen

Hough-Transformation – Phase: Entwurf

Finden von Linien mittels Hough-Transformation

- Hough-Transformiertes Bild enthält für jede mögliche Linie einen Punkt
- Darstellung dieser Gerade (Linie) in Hessescher Normalform
  - Winkel der Normalen der Gerade
  - Abstand vom Ursprung
- Maxima in diesem Bild repräsentieren eine Linie
- Entscheidung der Falz anhand der Maxima aus Hough-Transformation

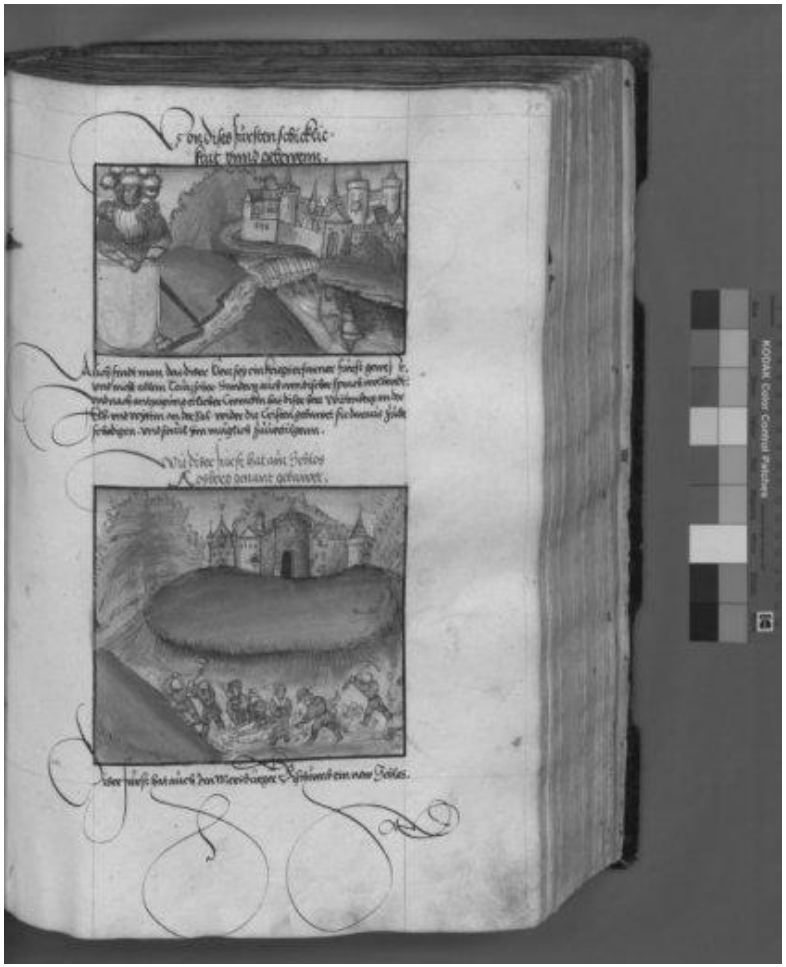


Hough-Transformation – Phase: Implementierung

- Nutzung von ImageJ als Programm
- Implementieren der Spezifikationen
- Verbalisierung
  - Erklärende Namen für Variablen
  - Selbsterklärender Code
  - Programmkommentare

```
IJ.write("Bild geladen");

// in graustufen wandeln
ip = ip.convertToByte(true);

// Breite des uebergebenen Bildes
w = ip.getWidth();

// Hoehe des uebergebenen Bildes
h = ip.getHeight();

// Bild fuer die Richtung
direction = new ByteProcessor(w - 2, h - 2);

get_falz();
```

Ergebnisse der Hough-Transformation



- 1. Versuch: Finden der Falz

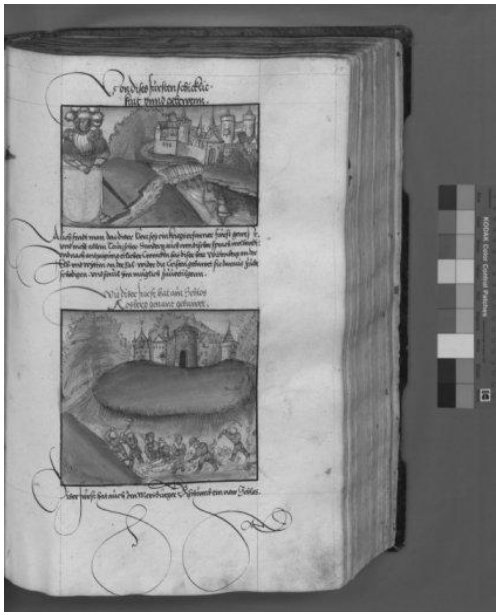
Photoshop – Phase: Implementierung

Farbtrennungs- und Glättungsfilter

- Färbung der Randdaten
- Reduzierung der Tonstufen (stärkere Tontrennung)
- Glättung der Ränder durch den Filter "Staub und Kratzer entfernen"
- Extrahierung der Seite in zusätzlichem temporären File

Sobel-Operator – Phase: Entwurf

- Rückgriff in die Entwurfsphase
  - Photoshop hat Probleme mit dem Finden der rechten bzw. linken Bildbegrenzung
  - Erweiterung der Hough-Transformation in ImageJ um das Suchen der äußeren Bildbegrenzungen



Ergebnisse der Hough-Transformation

- Finden der Falz und der linken bzw. rechten Bildbegrenzung
- Auseinanderschneiden an der Falz

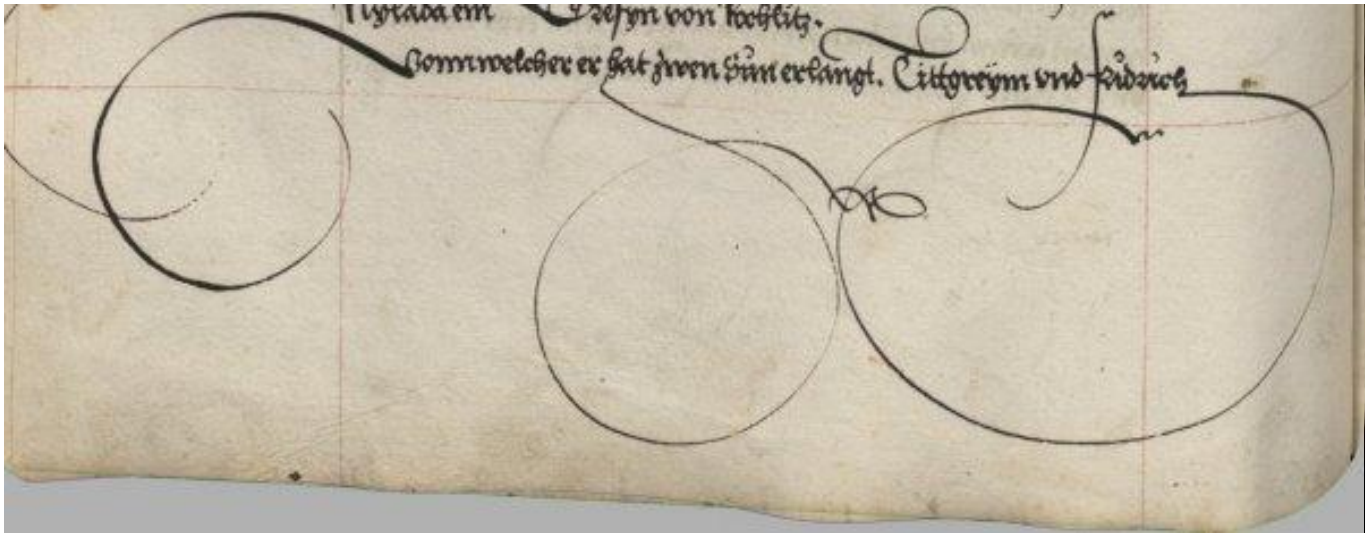
Abschätzen der Randdaten – Phase: Entwurf

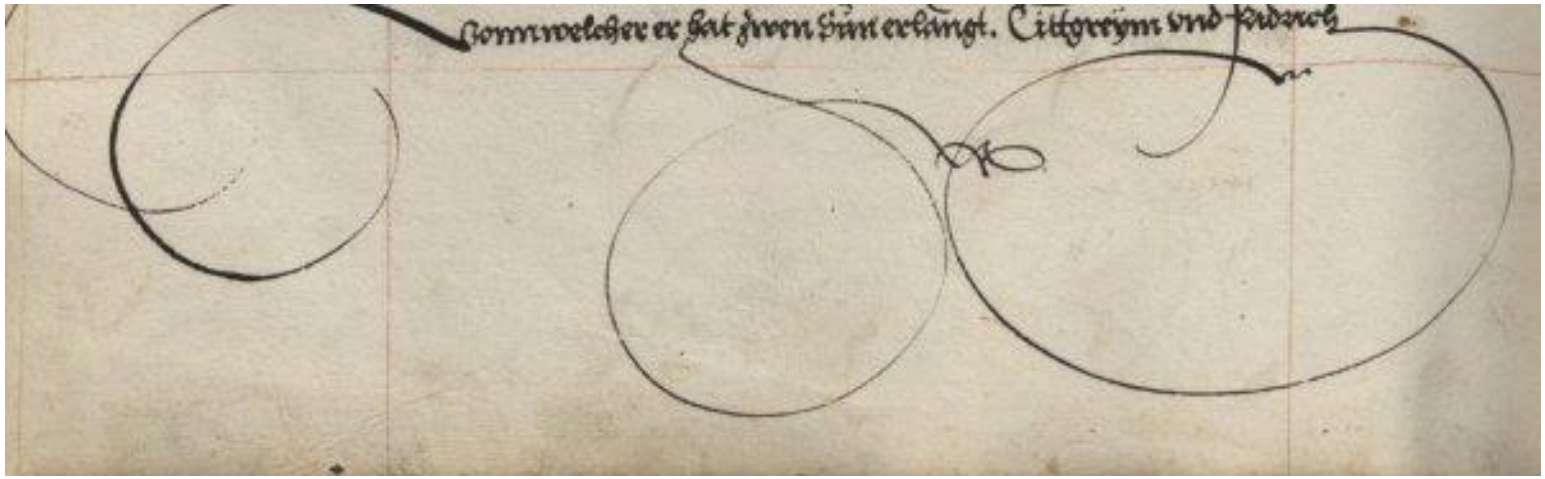
- Auslesen des unteren bzw. oberen Randdaten aus dem temporären File
- Anwendung eines Median-Filters
- Zusätzliche Korrektur des Randes mittels polynomialer Regression

Polynomiale Regression – Phase: Entwurf

- Suche eines Polynoms n-ten Grades, so dass die Summe der Fehlerquadrate minimal wird
- Approximation der unteren bzw. oberen Seitenbegrenzung
  - Ausgleichen der lokalen Fehler durch Wasserflecken o.ä.
  - Bestimmung der Krümmung um Falz aufzubiegen

Polynomiale Regression – Beispiel 1





Polynomiale Regression – Beispiel 2

Abschätzen der Randdaten – Phase: Implementierung

```java
/**
 * method that implements the polynomiell regression
 *
 * @param tmp Array where the approximation tooks place
 * @param grad the degree of the approximation
 * @param reg how many points are responsible for approximation
 * @return approxed pixel
 */
private double polyReg(int[] tmp, int grad, int reg) {
```

- Methode polyReg zur polynomiellen Regression
  - Erklärende Namen für Variablen
  - Selbsterklärender Code
  - Programmkommentare

Javadoc – Phase: Implementierung

Package Class Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS | FRAMES NO FRAMES All Classes
SUMMARY: NESTED | FIELD | CONSTR | METHOD | DETAIL: FIELD | CONSTR | METHOD

Class Transform3_

```
java.lang.Object
  └Transform3_
```

All Implemented Interfaces:
ij.plugin.filter.PlugInFilter

```
public class Transform3_
```

extends java.lang.Object
implements ij.plugin.filter.PlugInFilter

class to transform a cropped image in front of a black background to a rectangle with maximal width and height

Author:
Doris Aschenbrenner & Jan Wiefel

Field Summary

Fields inherited from interface ij.plugin.filter.PlugInFilter
CONVERT_TO_FLOAT, DOES_16, DOES_32, DOES_8C, DOES_8G, DOES_ALL, DOES_RGB, DOES_STACKS, DONE, FINAL_PROCESSING, NO_CHANGES, NO_IMAGE_REQUIRED, NO_UNDO, PARALLELIZE_STACKS, ROI_REQUIRED, SNAPSHOT, STACK_REQUIRED, SUPPORTS_MASKING

Constructor Summary

Transform3_()	

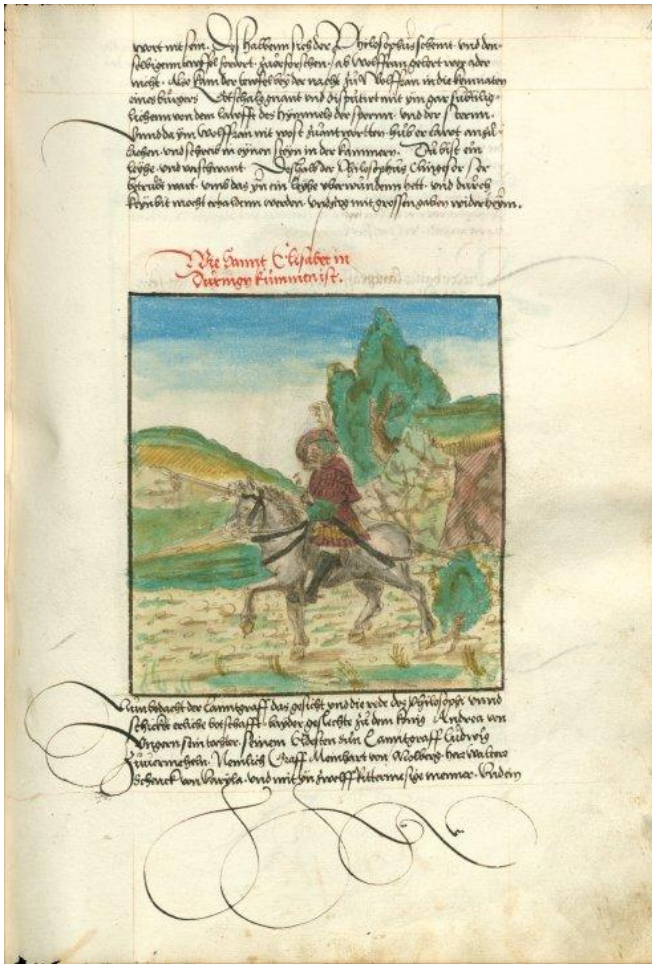
Method Summary

void	run(ij.process.ImageProcessor ip) method to transform the image actually
int	setup(java.lang.String arg0, ij.ImagePlus imp) required setup method
int[]	xtend(int[] data, int length) method extends an array proportionally, it should work with compressing, too, but does no merging

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Ziehen auf ein Rechteck – Phase: Implementierung

- Aufbiegung der Falz mit Hilfe der gewonnenen Krümmung
- Aufhellen der Schattierung an der Falz



Ziehen auf ein Rechteck – Phase: Implementierung

Inhalt

1. Thema
2. Ausgangssituation
3. Ziele
4. Recherche
5. Umsetzung
6. Fazit

Fazit

- Wiederholung und Anwenden der Inhalte der Vorlesung Softwaretechnik
- Arbeiten im Team

- Kennenlernen von Grafikprogrammen und Plug-in-Erstellung
- Schwierigkeit: Automatisierung, Wiederverwendbarkeit

Vielen Dank für Ihre Aufmerksamkeit