Portrait Cropping

Digital Imaging
Semesterprojekt SS05

13. Juni 2005

Fritz-Michael Gschwantner
Michaela Heinzl
Sonja Steindl

Inhaltsverzeichnis

1 Aufgabenstellung

Die Aufgabe unseres Digital Imaging 4 Projektes bestand darin, ein automatisches Portrait Cropping zu implementieren. Das heißt, aus einem Portraitfoto sollte der Kopf der Person gefunden werden, um anschließend das Bild in Bezug auf die Kopfgröße zu beschneiden und zu skalieren. Ein solch einheitliches Cropping könnte dann unter anderem zB beim Einbinden von Studentenfotos ins Web sehr hilfreich sein.

2 Mögliche Lösungsansätze

Nach Überlegungen von eventuellen Lösungsvorschlägen kamen wir auf zwei potentielle Wege das Cropping zu realisieren.

2.1 Hauterkennung

Das finden der Haut in einem Bild erschien uns zu Beginn als ein recht guter Ansatz. Dabei sollten die Regionen mit Haut gefunden und anschließend noch das Gesicht dabei herausgesucht werden. Weitere Überlegungen stellten uns jedoch vor **Probleme**:

- Wie „definiert“ sich Haut? Sättigung, Helligkeit und Farbton variieren doch sehr stark von Bild zu Bild?
- Was ist mit Personen mit starkem Bart?
- Was ist mit Personen mit dunkler Hautfarbe?
- Was ist mit nackten Personen? Dies würde noch eine komplexere Umsetzung erfordern.

Natürlich hätte diese Methode auch ihre **Vorteile**:

- Gesichtsfindung ohne Haare möglich
- Gesichtsfindung funktioniert auch bei buntem, komplexem Hintergrund

Des weiteren haben wir erfahren, dass sich bereits unsere Kollegen mit der Hauterkennung auseinandersetzen, und somit wollten wir das Problem auf eine andere Art und Weise lösen, um einen besseren Vergleich zu erhalten, welche Lösung nun „effizienter“ funktioniert.

2.2 Konturenfindung

Als eine weitere Möglichkeit untersuchten wir die Vor- und Nachteile einer Konturenfindung und somit der Trennung des Körpers vom Hintergrund mit einer möglichst scharfen und durchgehenden Linie. Liegt einmal eine

deutliche Kontur zwischen Kopf und Hintergrund vor, kann mit Hilfe einer horizontalen Grauwertprojektion der Kopf lokaliersiert werden. Die untere Begrenzung des Kopfes bildet der Hals, der die schmalste Stelle des Körpers darstellt. Auch die Auffindung der linken und rechten Kopfkante und des obersten Punktes des Kopfes erschien uns als kein Problem. Jedoch gab es **Einschränkungen** die uns bereits vor der Implementierung klar waren:

- Funktioniert nur bei einheitlichem Hintergrund ohne grobe Kanten.
- Funktioniert wahrscheinlich nur eingeschränkt bei Personen mit langen offenen Haaren, da der Hals nicht eindeutig gefunden werden kann.

Aufgrund folgender **Vorteile** haben wir uns schlussendlich aber trotzdem für die genaue Umsetzung dieser Variante entschieden:

- Unabhängig von der Hautfarbe.
- Funktioniert auch problemlos bei Personen mit Bart und nacktem Oberkörper.
- Kleine Unebenheiten im Hintergrund sind kein Problem.
- Allgemeine Implementierung erscheint uns im Gegensatz zur Hauterkennung einfacher, generell gut umsetzbar und zielführender.

3 Umsetzung und Implementierung

3.1 Programmstruktur

Unser Programm lässt sich in 3 Schritte unterteilen: (Abb.: 1)

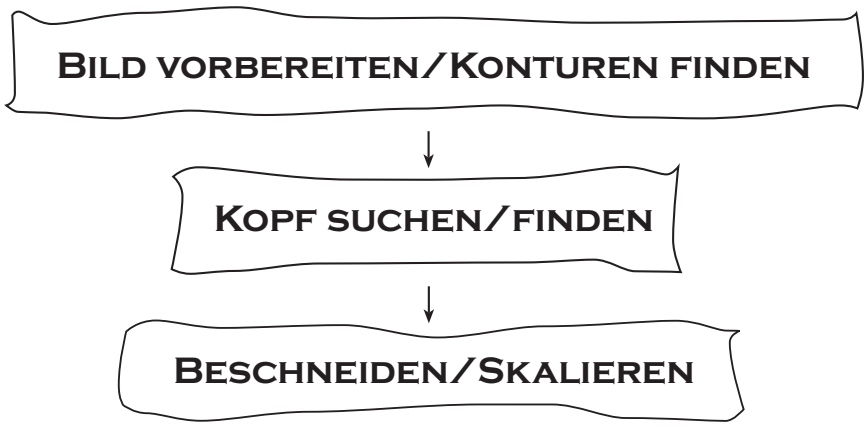


Abbildung 1: Struktur des Programms

Abbildung 2: Ausgangsbild

3.2 Bild vorbereiten/Konturen finden

Zu allererst muss unser Ausgangsbild (siehe Abb.: 2) so aufbereitet werden, dass es ein möglichst gutes Ausgangsmaterial für die spätere Suche des Kopfes darstellt, das heißt die Person muss deutlich vom Hintergrund getrennt sein.
Für die Präparation des Bildes wurden unter anderem folgende von ImageJ (Klasse `ImageProcessor`) zur Verfügung gestellten und sehr hilfreichen Operationen genutzt, deren genauer Einsatz im Weiteren noch erläutert wird:

`smooth();`	Glättung
`findEdges();`	Kantenfilterung mit Sobel-Operator
`autoThreshold();`	Umwandlung in ein Binärbild
`medianFilter();`	Beseitigung von kleinen Störungen
`erode();`	Regionen schrumpfen lassen
`dilate();`	Regionen wachsen lassen

Wir wandeln unser RGB Bild zuerst in ein 8-Bit-Grauwert-Bild um. Falls wir von einem bereits vorhandenen Graustufenbild ausgehen, wird es trotzdem nochmal in ein 8-Bit-Grauwert-Bild gewandelt, um gleiche Ausgangsbedingungen für die spätere Weiterverarbeitung zu garantieren.
Zu Beginn werden erste feine Unebenheiten durch eine Glättung beseitigt, und eine Konturenfindung mit dem Sobel-Operator angewandt. (siehe Abb.: 3)

Anschließend wird der Kontrast des Bildes abhängig von den vorhandenen Helligkeitsverhältnissen verstärkt. (siehe Abb.: 4)

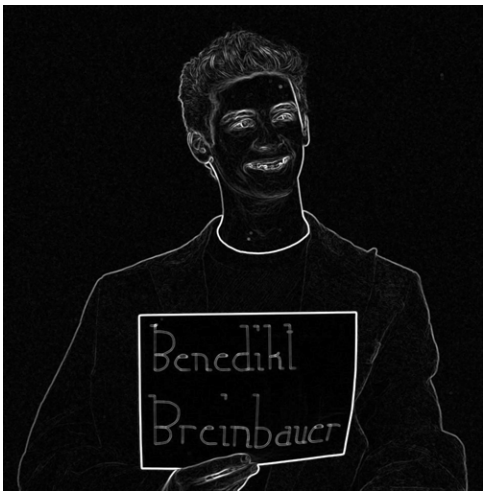


Abbildung 3: Konturenfindung mit Sobel-Operator

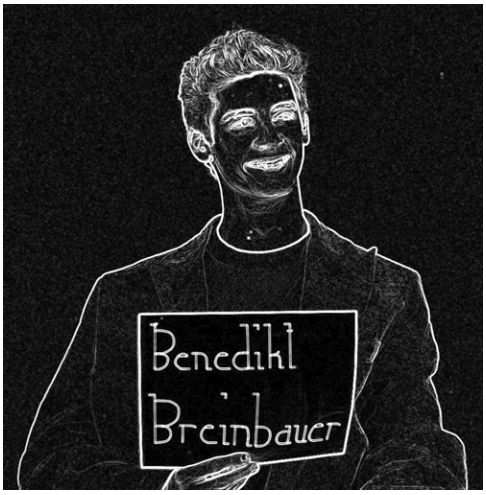


Abbildung 4: Kontrastverstärkung

Danach erfolgt die Umwandlung in ein Binärbild mittels THRESHOLD, bei dem Pixel über dem aus dem Histogramm bestimmten Schwellwert weiß werden und die darunter schwarz. (siehe Abb.: 5)

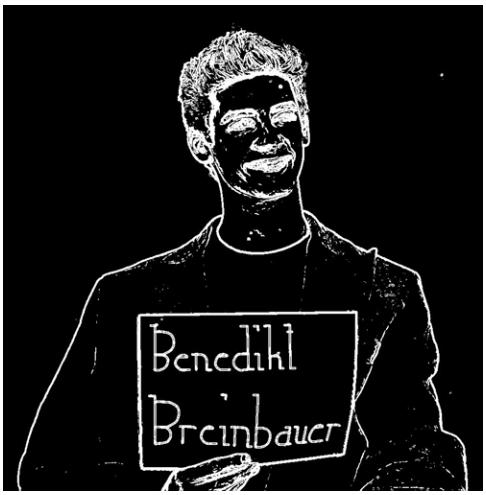


Abbildung 5: Binärbild

Durch die mehrmalige Anwendung von EROSION und DILATION sollen unterbrochene Konturen wieder miteinander verbunden werden. Mit Hilfe eines 3x3 MEDIAN-FILTERS und DILATION/EROSION werden weitere Unebenheiten im Bild (vorallem unerwünschte kleine Flecken im Hintergrund) korrigiert und die für uns wichtigen Kanten des Kopfes noch einmal verdeutlicht. (siehe Abb.: 6)

Abbildung 6: Ergebnis der Erosion, Dilation und Medianfilter

Mit der im Code verwendeten Reihenfolge dieser verschiedenen Operationen erzielten wir neben unzähligen anderen versuchten Varianten das weitaus beste Ergebnis. Die extrem erscheinende Verstärkung der Kontur ist besonders dann notwendig, wenn der ursprüngliche Kontrast nicht besonders hoch ist, und vorallem um die oft durch Haare „zart“ oder „brüchig“ erscheinende Kontur zu festigen.

3.3 Kopf suchen/finden

Mit einem idealerweise so aufbereiteten Bild machen wir uns nun auf die Suche nach dem Kopf. Jede Zeile des Bildes wird durchlaufen und die äußersten weißen Pixel jeder Zeile ermittelt - welche der Kontur des Kopfes entsprechen sollte. Danach wird der Abstand zwischen diesen beiden Pixel jeder Zeile ermittelt und damit ein Histogramm erstellt. In diesem Histogramm ist quasi der Kopf in horizontaler Lage zu sehen. (siehe Abb.: 7)

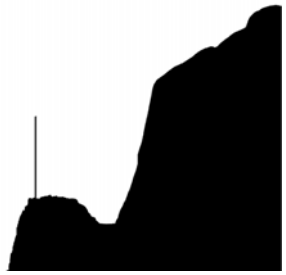


Abbildung 7: Histogramm

Um eventuell auftretende „Ausreißer“ in der Kontur auszubessern, wird dieses Histogramm „normalisiert“, um letzte kleine Fehler im Hintergrund und Einbrüche in der Kontur zu beseitigen. Dabei werden besonders einzelne hervorstechende Spitzen bzw. Löcher abgeschnitten bzw. gefüllt. (siehe Abb.: 8)

Abbildung 8: Normalisiertes Histogramm

Nun wird wirklich nach der Position des Kopfes gesucht. Der oberste Punkt vom Kopf wird an der Stelle gefunden, an der zum ersten mal Werte im

Histogramm auftreten, bzw. diese Werte mit jeder Zeile steigen. Ist der Maximalwert der stetigen Steigung erreicht, wird der linke und rechte Rand des Kopfes fixiert. Sobald die Werte wieder sinken, bewegen wir uns Richtung Hals. Sobald danach die Werte wieder zu steigen beginnen, was bei den Schultern der Fall ist, endet die Suche und der unterste Punkt des Kopfes (Übergang von Hals zu Schultern) ist gefunden.

Zur besseren Veranschaulichung des gefundenen Bereiches wird dieser für unsere Zwecke im Orignalbild mit einem Rahmen markiert. (siehe Abb.: 9)

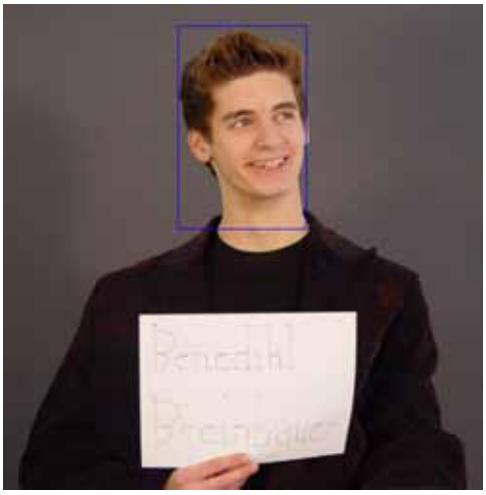


Abbildung 9: Gerahmte Kopfregion im Original

3.4 Beschneiden/Skalieren

Nachdem nun der Kopf lokalisiert wurde, soll dieser noch ausgeschnitten und anschließend auf die vom Benutzer gewünschte Endgröße skaliert werden. Aus optischen Gründen wollten wir aber nicht so knapp am Kopf beschneiden, wie es in Abb.: 9 mit dem Rahmen markiert ist. Deshalb wird zuerst die Höhe der Kopfregion um 10% erweitert, und anschließend wird die Breite je nach den zuvor vom Benutzer eingegebenen Seitenproportionen ebenfalls festgelegt.

Diese neu festgelegte Region um den Kopf wird nun als „Region Of Interest“ (ROI) definiert und wird dann mit `crop()` aus der Klasse `ImageProcessor` beschnitten. Anschließend wird dieses Portrait noch auf die gewünschte Größe mit `resize()` skaliert. Das Programm nimmt auch darauf Rücksicht, dass beim Zuschneiden des Portraits weder die Höhe noch die Breite des gefundenen Kopfes beschnitten wird. (siehe Abb.: 10)

Abbildung 10: Beschnittenes, skaliertes Portrait

3.5 Einschränkungen

Unser Programm läuft einwandfrei für RGB und Grauwertbilder mit einheitlichem Hintergrund. Das heißt es dürfen keine starken Kanten auftreten, da sonst die Detektion des Kopfes nicht mehr möglich ist. Farbverläufe funktionieren jedoch völlig einwandfrei. Kleine Fehler im Hintergrund bzw. in der Kontur des Kopfes können ausgegelichen werden.

Weiters darf auf dem Bild nur eine Person abgebildet sein. Sollten mehrere Köpfe gesucht/gefunden werden, wär die Hauterkennung eine Alternative.

Probleme ergeben sich meist bei Personen mit langen, offenen Haaren, da der Hals nicht als schmalste Stelle identifiziert werden kann. Manchmal ergibt sich trotzdem ein brauchbares Ergebnis, wobei man dies aber grundsätzlich als „Zufall“ betrachten kann. (siehe Abb.: 11). Die besten Ergebnisse erzielt unser Plugin bei Personen mit kurzen oder zusammengebunden Haaren.

Außerdem können Probleme auftauchen, wenn die Bildqualität und Kontrastwerte bereits von Anfang an zu wünschen übrig lassen. Die Person sollte sich so deutlich wie möglich vom Hintergrund abheben.

3.6 Erweiterungen und Verfeinerungen

Um die Schwachstellen in unserem Programm (siehe Kapitel 3.5) zu beseitigen, müsste man höchstwahrscheinlich das komplette Konzept umstellen. Natürlich wäre die Hauterkennung eine gute Alternative, um zum Beispiel auch Bilder mit einem kantenreichen Hintergrund zu bearbeiten.

In unserem Algorithmus könnte man versuchen, die Methode zur Normalisierung des Histogramms noch zu verbessern, um eine noch sauberere Kante zu erzielen.

Weiters wäre es wünschenswert eine bessere Reinigung von Fehlern im Hintergrund und einen ausgereifteren Algorithmus zur besseren Detektion von Personen mit langen Haaren zu implementieren.

Eine bereits eingearbeitete und in Kapitel 3.4 kurz erwähnte Verfeinerung ist die Eingabe der gewünschten Endhöhe und -breite. Somit kann der Benutzer festlegen, welche Seitenverhältnisse bzw. Größe das beschnittene Bild schlussendlich haben soll. Entsprechend dieser Werte wird das Portrait beschnitten und skaliert.

Es ist dem Benutzer jedoch nicht zu empfehlen, dass er bei einem schlecht aufgelösten Ausgangsbild zu große Seitenverhältnisse eingibt. Denn wenn das gecroppte Portrait kleiner ist, als die vom Benutzer gewünschten Werte, erfolgt eine Aufskalierung des Bildes, was die Bildqualität natürlich deutlich schmälert. In einem solchen Fall ist gegebenenfalls auf die in einem anderen Projekt realisierten Upscale-Filter zu referenzieren ;)

Weiters ist zu beachten, dass der Benutzer bei zu „extremen“ Seitenverhältnisangaben sinnvoll eingeschränkt wird. Das heißt, schneiden die eingegebenen Verhältnisse unsere festgelegte ROI (was unter Umständen zu einem Cropping mitten durchs Gesicht führen würde), wird wieder auf ein sinnvolle Verhältnis aufskaliert, damit ein wünschenswertes Cropping-Ergebnis erzielt wird. Dadurch kann es allerdings zu Verzerrungen kommen, wenn die ROI über den Bildrand kommt bzw. kommen würde.

4 Testen und Verbessern

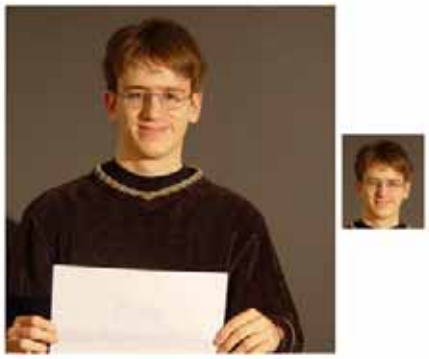
Getestet wurde während der ganzen Entwicklungsphase des Portrait-Cropping. Laufend wurde der Algorithmus erweitert und verbessert, um für so viele Bilder wie möglich ein ideales Ergebnis zu erzielen. Anfangs machte uns die Realisierung einer sauberen Kontur Probleme. Der Contrast-Enhancer von ImageJ reichte zu allererst für unsere Wünsche nicht aus, somit mussten wir eine eigene Kontrastverstärkung implementieren.

Erst durch das Testen von verschiedenen Varianten zur „Reinigung“ des Hintergrundes und die Normalisierung des Histogramms erzielten wir ein annehmbares Ergebnis, das uns die Detektion des Kopfes erst ermöglichte.

Grundsätzlich liefert das Programm sehr schnell ein Ergebnis, was auf eine einfach gehaltene Implementierung und Struktur zurückzuführen ist.

4.1 Testbilder

Hier einige Beispiele für das automatische Portrait-Cropping mit einwandfreien Ergebnissen, auch bei unterschiedlich festgelegten Seitenverhältnissen für das Endportrait:



Michaela
Hemzl


MICHAEL


Michael
PLANK

michael

Michaela

Wie in Abb.: 11 zu sehen, kann es auch bei langen Haaren zu annehmbaren Ergebnisses kommen. Im linken Beispiel, indem die Haare immer breiter werden, wird bereits gleich nach dem Kinn gecroppt. Im rechten Beispiel kann ebenfalls keine schmale Stelle am Kopf (Hals) gefunden werden, was hier zu Folge hat, dass im Endbild auch die Schultern zu sehen sind. Diese beiden Beispiele funktionieren jedoch wirklich nur „zufällig“, da die Kontur nicht durchgängig ist.

Abbildung 11: Ergebnisse langes Haar

5 Zusammenfassung

Das Ziel des Projektes wurde nach unserem geplanten Ermessen weitgehend erfüllt. Die aufgetretenen Schwierigkeiten wurden bereits erläutert, und waren eigentlich mit ein bisschen Überlegung und Tüfteln recht schnell beseitigt. Das Konzept wurde bis zum Ende durchgezogen und erfordete keine groben Änderungen.

5.1 Projektaufteilung

Da das Projekt aus 3 Grundschritten aufgebaut ist, teilten wir diese auf uns drei Mitglieder auf. Wobei aber auch nicht jeder völlig für sich allein arbeitete, sondern wir uns immer aufeinander abstimmten und einander bei der Lösung von Problemen halfen.

Die Dokumentation wurde großteils von Michaela Heinzl und Sonja Steindl erstellt, da Fritz-Michael Gschwanter durch eine ausgereifte Implementierung der Kopfsuche bereits viel Arbeitszeit investiert hat. Schlussendlich ergab sich aber ein guter Ausgleich des Zeitaufwands. Die Kommentare im Code wurden von den jeweiligen Verantwortlichen ergänzt. Teamarbeit gilt auch für die Präsentation, wobei jedes Mitglied natürlich auch einen Überblick über die jeweils anderen Programmteile hat.

Auflistung der Programmieraufgaben:

Sonja Steindl

- *Bild vorbereiten/Konturen finden:*

- run Methode

- prepareImage()

- autoContrast()

Fritz-Michael Gschwantner

- *Kopf suchen/finden:*

- showWidthHisto()

- normalizeHistogram()

- findBorders()

- showHead()

Michaela Heinzl

- *Beschneiden/Skalieren:*

- Head Klasse

- askDimensions()

- finalize()

5.2 Zeiterfassung

Besprechung, Einteilung	2h
Bild vorbereiten/Konturen finden:	
Erste Konturenfindung (Sobel-Operator, Kontrast, Erosion und Dilation)	3h
Verbesserung der Kontur mit eigener Kontraststeigerung und zusätzlichem Median-Filter	3h
Kopf suchen/finden	
Histogramm, Normalisiertes Histogramm	4h
Kopfkoordinaten finden	4h
laufende Verbesserungen am Algorithmus	5h
Beschneiden/Skalieren	3h
Dokumentation	15h
Präsentation	2h

Somit ergibt sich ein **Gesamtaufwand von 41h**
davon:

- Fritz-Michael Gschwantner: 15h
- Michaela Heinzl: 13h
- Sonja Steindl: 13h

6 Anhang

Quellcode