Dokumentation

Computer Vision Übung 1

Binder Walter, 0526638
Birkner Tamás, 1342667
Tuscher Michaela, 0827032

Aufgabe 1: Colorizing Images

In dieser Aufgabe ging es darum aus einzelnen Farbkanälen ein Farbbild zusammenzusetzen. Zu diesem Zweck waren Sets von Bildern, bestehend aus je 3 Kanälen gegeben, wobei die Schwierigkeit war, dass die einzelnen Kanäle nicht korrekt aufeinander ausgerichtet waren. Deshalb musste zusätzlich die beste Ausrichtung für die Kanäle berechnet werden.

Die von uns erstellte Funktion lässt sich mit dem Aufruf *imColor('imagenummer');* starten, wobei *imagenummer* die gewünschte Bildnummer angibt, z.B. *imColor('00125');*. Nach dem Aufruf werden alle 3 Kanäle des gewünschten Bildes eingelesen. Um die Kanäle auszurichten wurde die MATLAB-Funktion *corr2* verwendet, die die Cross Correlation zwischen Bildern, bzw. in diesem Fall für je 2 Kanäle berechnet. Hierfür wurde der grüne Kanal gleich belassen und der rote und blaue Kanal je 15 Pixel in jede Richtung verschoben. Dabei wurde bei jeder Verschiebung der Korrelationskoeffizient berechnet und im Endeffekt der rote und blaue Kanal jeweils an die Stelle geschoben, wo die höchste Korrelation mit dem grünen Kanal bestand. Zum Schluss wird das zusammengesetzte Bild ausgegeben.

Die farbigen Ränder lassen sich dadurch erklären, dass wegen der Verschiebung der Kanäle, damit sie zusammenpassen, einzelne Zeilen der Kanäle an den Rändern überstehen.
In folgendem Bild ist links das zusammengesetzte Bild vor der Ausrichtung der Kanäle zu sehen und rechts das fertige Bild mit korrekt ausgerichteten Kanälen.

Bei der Bearbeitung dieser Aufgabe fiel auf, dass die Ergebnisse für die Ausrichtung der Kanäle unterschiedlich gut waren, je nachdem für welche Kanäle man jeweils die Korrelation berechnet. Anfangs berechneten wir die Korrelation zwischen rot und grün und rot und blau. Das Ergebnis war im Prinzip für alle Bilder zufriedenstellend, außer für das mit dem sitzenden Mann. Dort konnte man deutlich sehen, dass der grüne Kanal etwas verschoben war und das Bild einen "grünen Schatten" hatte und auch sonst etwas verschwommen aussah. Aus diesem Grund versuchten wir ein besseres Ergebnis zu erzielen indem wir wie oben beschrieben die Korrelationen zwischen grün und rot und grün und blau berechneten. Für uns war erstaunlich, dass dabei ein deutlich besseres Ergebnis für dieses Bild erzielt wurde. Wir schlossen daraus, dass es wahrscheinlich aufgrund der Tatsache war, dass von der Wellenlänge her grün zwischen rot und blau liegt und daher wahrscheinlich die Berechnung der Cross Correlation bessere Ergebnisse liefert als z.B. zwischen rot und blau.
In folgendem Bild ist links das Ergebnis zu sehen, das dadurch erzielt wurde, dass der grüne und blaue Kanal mit dem roten verglichen wurden und rechts das deutlich bessere, bei dem der grüne Kanal mit den anderen beiden verglichen wurde.

Aufgabe 2: Image Segmentation by K-means Clustering

Allgemein

Die Idee von k-means Clustering ist die Segmentierung bzw. Einteilung der Bildpunkte in Gruppen, sogenannten Clustern. Dabei werden k Cluster vor Programmstart bestimmt und die Bildpunkte werden durch Berechnung des Abstandes zu ihrem nähestgelegenen Cluster zugeordnet. In weiteren Durchläufen werden die Clusterzentren neu berechnet und die Bildpunkte neu zugeordnet. Ändert sich die Zuordnung zu den Clustern je Durchlauf nur mehr gering, bricht der Algorithmus ab.

Parameter – RGB (3D) / RGBXY (5D) festes k = 10

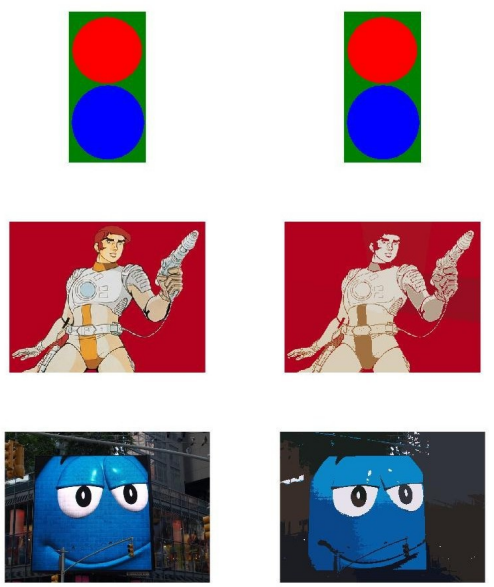


Abbildung 1: RGB +XY Paramter, k =10

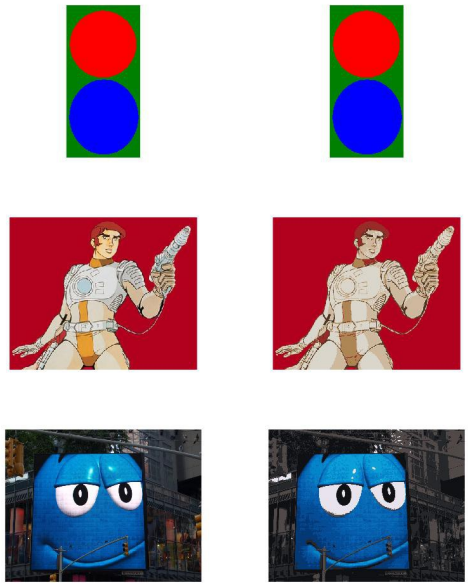


Abbildung 2: RGB Parameter, k =10

Stärken von K-means Clustering

Die Stärken von k-means Clustering liegen in der einfachen Umsetzung und in der schnellen Laufzeit. Der Algorithmus kann verschiedenste Merkmale aus n-Dimensionen zur Segmentierung von Bildinformationen verwenden. Wie in Aufgabe 2 bei der Verwendung von RGB Werten und Lageparametern (x,y Koordinaten) umgesetzt.

Schwächen von K-means Clustering

Der Algorithmus von K-means Clustering ist sehr stark davon abhängig welche Startwerte für μ_k, die Zentren der k-Cluster, auswählt wurden. Auch die Anzahl der verwendeten Cluster k bestimmt das Ergebnis maßgeblich.

In der Angabe zu Aufgabe 2 wird gefordert zufällige Werte als Startwerte für μ_k auszuwählen. Diese zufällige Auswahl führt bei der Optimierung zu jeweils unterschiedlichen lokalen Maxima pro Programmstart. Somit wird nicht eine globale Lösung gefunden sondern unterschiedliche lokale Optimierungen. In den Bildergebnissen resultiert dies in unterschiedlichen Farbwerten bei der Bildausgabe je Programmstart.

Ein weiterer Nachteil der zu sehen ist, ist dass die optimale Wahl der Anzahl von k Cluster sich je nach Semantik des zu untersuchenden Bildes unterscheidet. Ein zusätzlicher Nachteil ist, dass k vor Programmstart entweder manuell festgelegt werden oder durch eine Heuristik bestimmt werden muss.

Das Datenset des Bildes soll wenig Rauschen bzw. wenige Ausreißer besitzen, da sonst die Clusterzentren verschoben werden.

Unterschied 3D / 5D Parameter

In Bezug auf die erhaltenen Testbilder ist festzustellen, dass die Anwendung mit 3D Parametern (nur RGB) die weitaus besseren Ergebnisse bringt . Das verstärkt sich je höher der Detailgrad des Bildes ist wie z.B.: bei mm.jpg

Bei Bildern mit reduzierter bzw. beschränkter Farbintensität, die nicht überlappende Objekte besitzen, sind die Unterschiede nicht zu erkennen, wie z.B. bei simple.jpg. Bei Bildern, die eine beschränkte Farbintensität und aneinanderliegende Bildsegmente besitzen, wie future.jpg ist die Methode mit 3D Parameter besser je höher auch der Wert k gewählt wird, da dadurch Farbunterschiede von Hintergrundfarbe und Haarfarbe, die beide einen ähnliche RGB Farbwert besitzen, besser zu sehen ist.

Die Bilder future.jpg und mm.jpg Unterscheiden sich vor allem bezüglich des Hintergrundes. Future.jpg besitzt eine einfärbige Fläche und schneidet somit bei der 3D und 5D Variante fast gleich gut ab. Wiederum mm.jpg besitzt einen stark diversen Hintergrund und liefert deshalb auch sehr schlechte Werte für die RGB XY Variante.

Generell scheint die zusätzliche Berücksichtigung der Pixelkoordinaten beim Clustering (5D-Variante) eher Nachteile zu bringen. Vor allem wenn das Bild weitläufige, einfarbige Flächen hat, da hier dann die Pixelkoordinaten sehr unterschiedlich sein können und es deshalb passieren kann, dass die Fläche in mehrere Cluster geteilt wird, da aufgrund der großen Unterschiede in den Koordinaten die Punkte nicht zu einem Cluster gehörig angesehen werden.

Untersuchung mm.jpg mit Variablen k von 3 – 50 für 3D und 5D Parameter

Man kann feststellen, dass die 3D RGB Parameter Variante grundsätzlich bessere Ergebnisse liefert als die 5D RGB XY Variante. Schon bei kleinem k, 5- 15, hat die RGB Variante einen Detailreichtum, die die 5D Variante bei 50k nicht besitzt.
Die großen Unterschiede sind darauf zurückzuführen, dass speziell das Bild mm.jpg für die RGB XY Variante nicht geeignet ist. Durch das einfarbige blaue Objekt auf dem zerteilten Hintergrund ergibt sich bei der zusätzlichen Berücksichtigung der Lageparameter eine ungünstige Berechnung für Clusterzuordnung und Centroidberechnung.
Die 3D Variante ist davon natürlich nicht beeinflusst und liefert deshalb ein besseres Ausgabebild.

Abbildung 3: 3D – RGB Bildreihenfolge Original | 3k | 5k | 10k

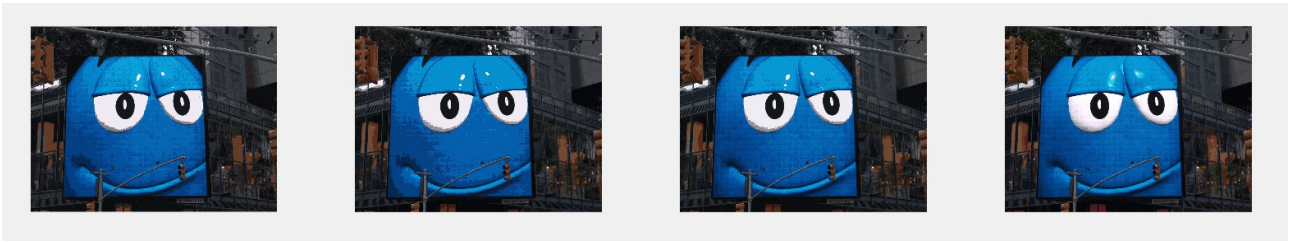


Abbildung 4: Abbildung 3: 3D – RGB Bildreihenfolge 15k | 20k | 30k | 50k

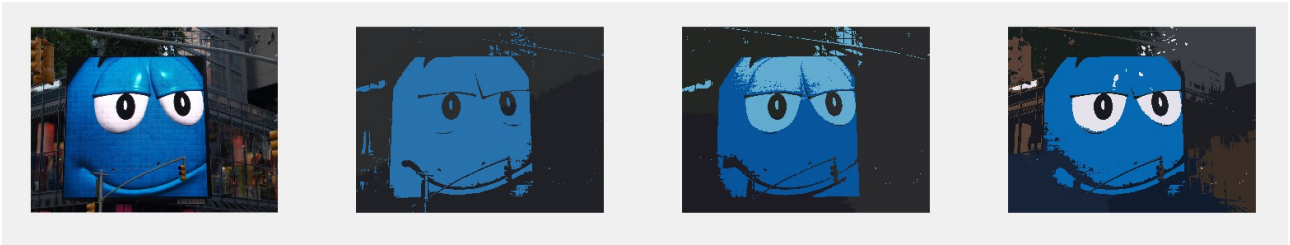


Abbildung 5: Abbildung 3: 5D – RGB XY Bildreihenfolge Original | 3k | 5k | 10k

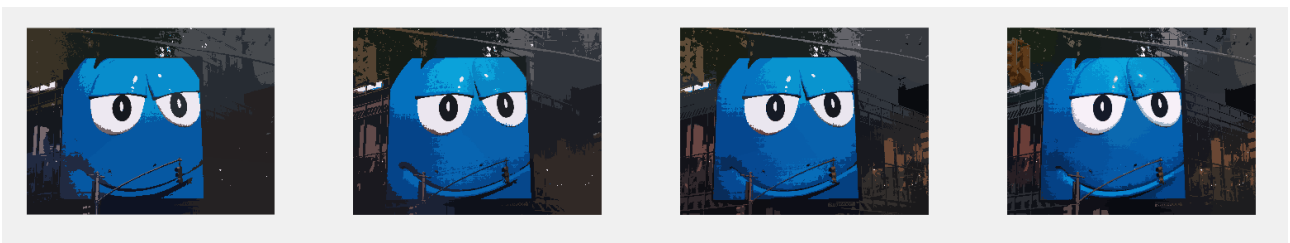


Abbildung 6: Abbildung 3: 5D– RGB XY Bildreihenfolge 15k | 20k | 30k | 50k

Aufgabe 3: Scale-Invariant Blob Detection

Bei dieser dritten Aufgabe ging es darum, sogenannte "Interest Points" zu entdecken. Diese sind solche kleine, herausstehende ("salient") Bereiche, wie Ecken oder Spitzen, die leicht auf anderen Bildern der gleichen Szene wiedergefunden werden können.

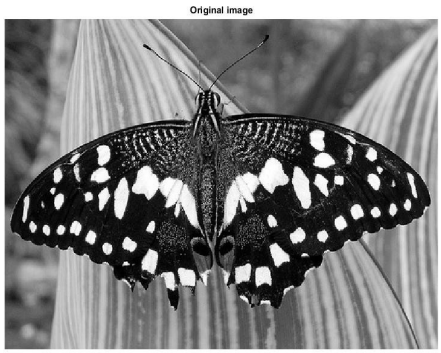
Diese Übereinstimmungen dienen dazu, um zu bestimmen, ob es sich um die selben Objekte (bzw. Szene) handelt.

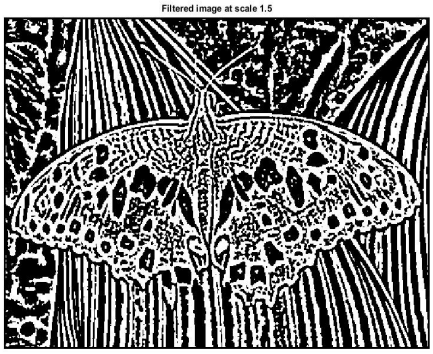
Zu diesem Zweck verwendete man die Methode der Konvolution mit einem sogenannten "Laplacian of Gaussian"-Filter (oder "Kernel").

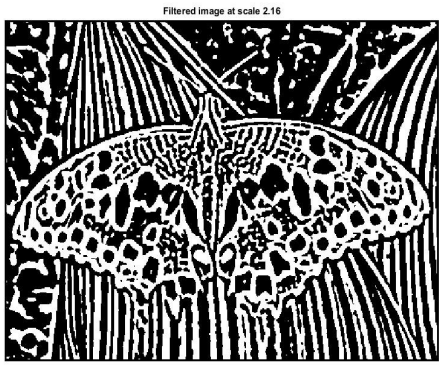
Dieses Filter hat die Eigenschaft, das es dank der unterliegenden Gauss-Funktion das Bild glättet, und das es dank der Ableitungen im Laplace-Operator auf Ungleichheiten sensibel ist. Dadurch werden, nach Eliminierung des Rauschens, die tatsächlichen Ecken und Kanten hervorgehoben.

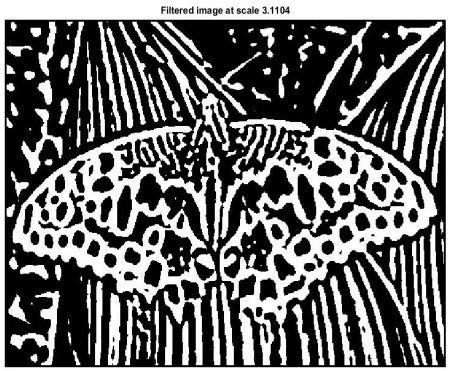
Man verwendete eine ganze Serie immer größer und grober werdender Filter (ca. 8 bis 10, mit Durchmesser in geometrischer Reihe).

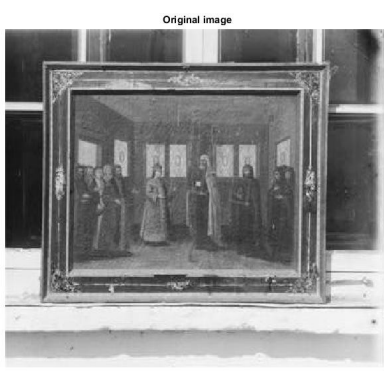
Hier sind einige der derart erzeugten gefilterten Bilder für *butterfly.jpg* und für *00149v_B.jpg*:

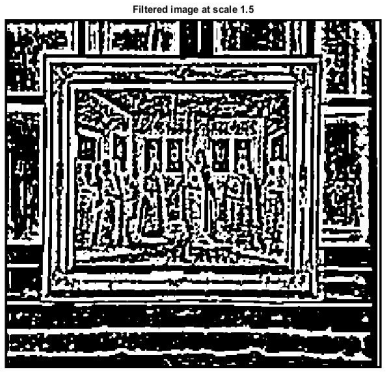


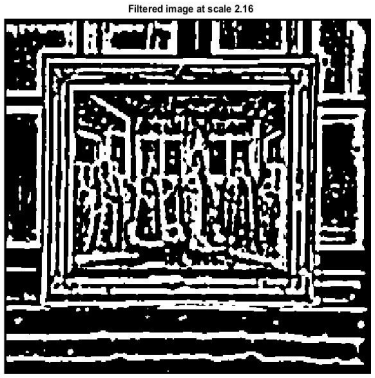


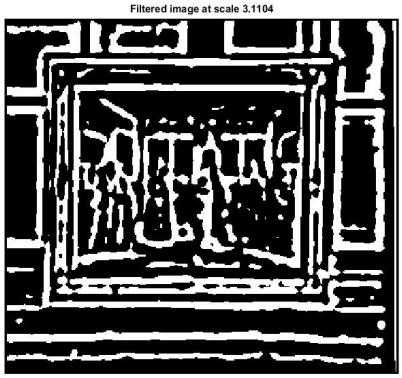




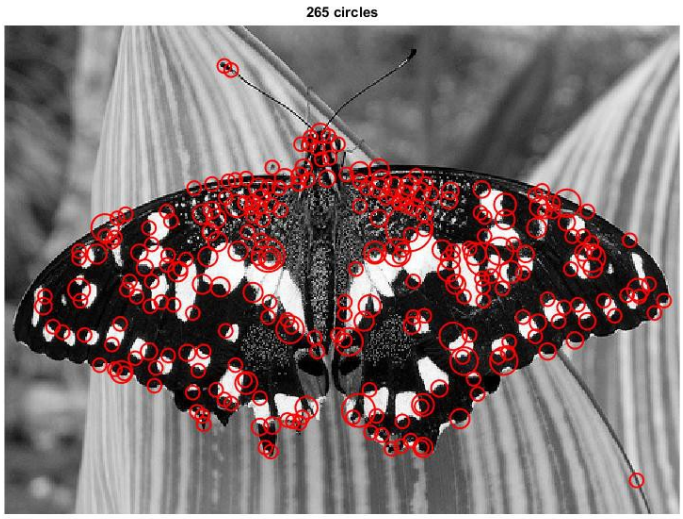


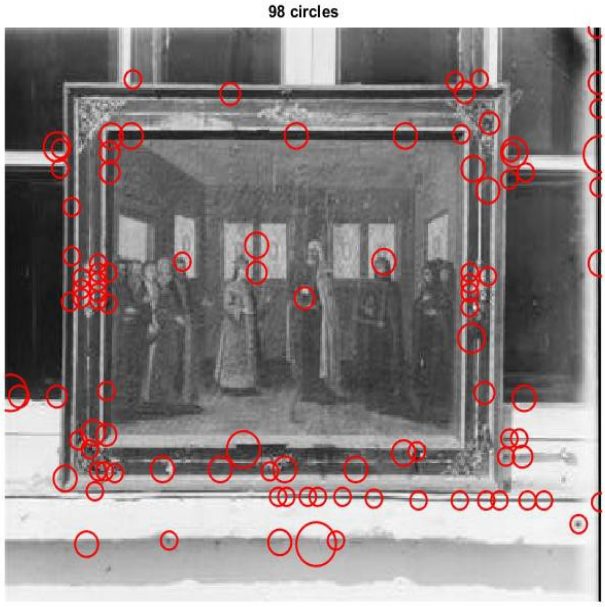






In den derart erhaltenen Grauwertbildern suchte man die lokalen Extrema aus. Diese hat man als Zentren gemerkt.



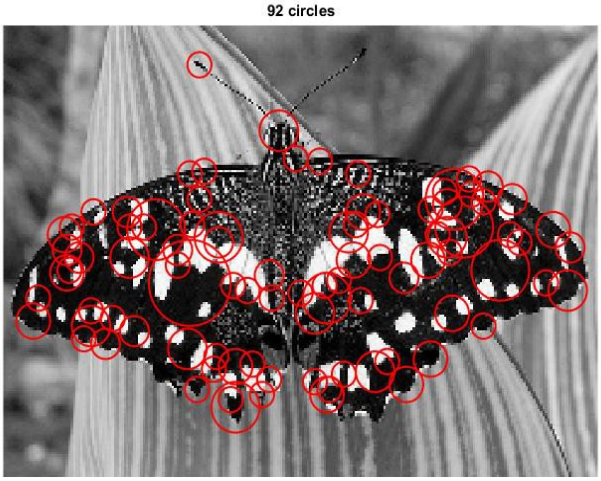


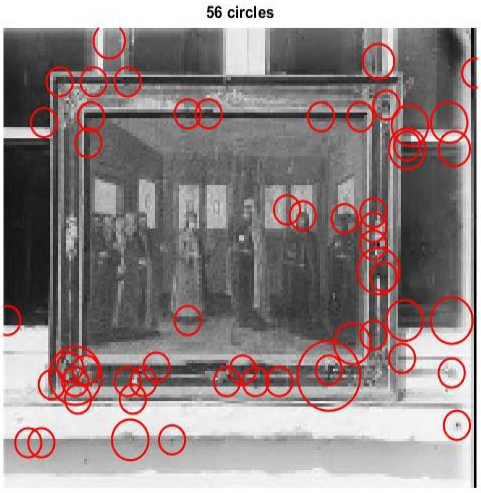
Die aus praktischer Sicht wichtigste Frage besteht insbesondere darin, ob mit dieser Methode für das selbe Bild bei verschiedenen Auflösungen die gleichen Punkte erkannt werden.

Damit modelliert man eine Situation, wo bei verschiedenen Bildern die Kameradistanz unterschiedlich ist.

(Auf verschiedene Winkel, dessen Behandlung für das Stereo-Sehen äußerst wichtig ist, wurde hier nicht eingegangen, ebenso wie auf Farbe- und Intensitätsunterschiede.)

Dazu generierte man (in der X- und Y-Richtung) 2-fache Unterstrichproben (subsampling) der Bilder.





Es hat sich herausgestellt, dass bei den weniger detaillierten Bildern mindestens 3 Mal weniger "Interest Points" erkannt wurden, als beim jeweiligen Originalbild.

Dazu ist die kleinere Punktemenge keine Teilmenge der Größeren (obwohl es eine zeimlich große Übereinstimmung gibt). Dieser Detektor ist also nicht (vollkommen) skaleninvariant.