Report zu Übung 1

**Gruppe 4**
Rainer Bernhard, 0828592
Sifuentes Caccire Fausto Heraldo, 0607000
Gangl Andrea, 1025756

## Assignment 1: Colorizing Images

Für den ersten Teil der Übung sind drei Grauwertbilder der gleichen Szene gegeben.
Ziel der Aufgabe ist es, die drei Bilder als einzelne Farbkanäle zu interpretieren, diese räumlich anzugleichen und im Endeffekt ein Farbbild zu bekommen.

## Implementierung

Für die Lösung werden zuerst die drei zusammenhängende Bilder eingelesen.
Um eine gute räumliche Anpassung der Bilder zu erreichen werden zwei Farbkanäle an den dritten angeglichen. In unserer Lösung wurde der rote und grüne Kanal auf den blauen Kanal ausgerichtet. Die Funktion „alignChannel“ bewerkstelligt diese Aufgabe. Der anzupassende Kanal wird in einem Intervall von [-15, 15] Pixel in X und Y Richtung verschoben. Jede Verschiebung wird mithilfe der normalisierten Kreuzkorrelation bewertet und das beste Ergebnis (max. Wert) als endgültige und korrekte Verschiebung des Kanals genommen.

Realisiert wurde dieser Algorithmus mit einer verschachtelten for-Schleife und den Matlab-Funktionen „circshift“ und „corr2“.

## Resultat

Das Endergebnis zeigt das farbige und korrekt übereinandergelegte Bild und links daneben die drei Farbkanäle aus denen das Bild erzeugt wurde.
Die Bilder im Datensatz wurden allesamt sehr gut rekonstruiert. An mindestens einem Rand von jedem rekonstruierten Farbbild sind Farbstriche zu erkennen. Deren Breite und Entstehung hängt von der Anzahl an Pixel, um welche das Farbkanal verschoben werden musste, ab, um einen räumlichen Ausgleich zu ermöglichen. Nichtsdestotrotz sehen die Farben im Zentrum des Bildes wahrheitsgetreu aus und es sind keine bis minimalen Verschiebungen zu erkennen.

Bild „00125v“ wäre ein Bild, welches ausgezeichnet erstellt wurde.


*Abbildung 1: Bild 00125v: keine Verschiebungen erkennbar und "echte" Farben*

Abbildung 2 zeigt die Signifikanz der Angleichung der Farbkanäle anhand von Bild „00125v“.

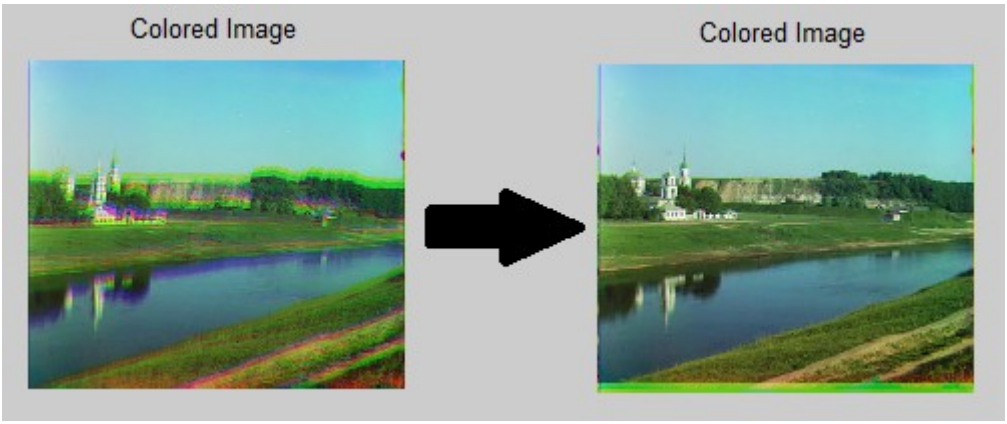


*Abbildung 2: Links: Farbkanäle nur übereinandergelegt, Rechts: Kanäle wurden angepasst und danach übereinandergelegt*

Die nächste Abbildung zeigt ein Beispiel von minimaler Verschiebung im Endergebnis. Ein Beispiel für eine minimale Verschiebung im Endresultat wäre Bild „00153v“. Auf den ersten Blick sieht das Bild gut rekonstruiert aus, wird es jedoch etwas vergrößert, lassen sich kleine Verschiebungen an den Kanten der Objekte erkennen. Dies ist womöglich darauf zurückzuführen, dass hier ein anderes Suchfenster nötig wäre als das vorgegebene.

Bild „00153v“ ist auch das einzige Bild wo diese Verschiebung erscheint. Alle anderen Bilder wurden gut rekonstruiert (Abbildungen 4 und 5).

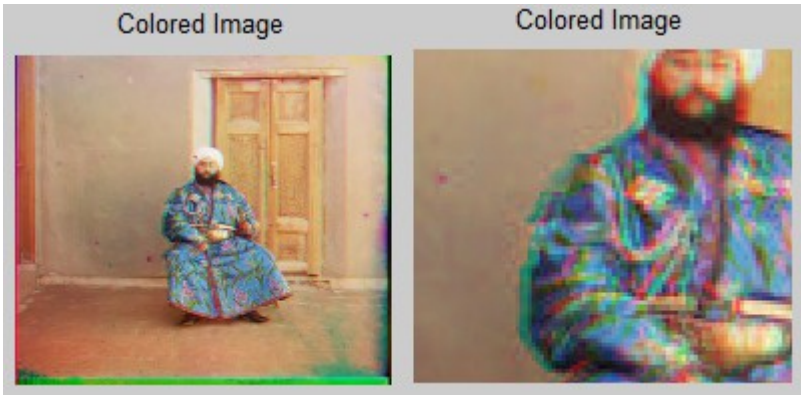


*Abbildung 3: Bild 00153: Fürs freie Auge korrekt angeglichen, beim Vergrößern fallen jedoch Verschiebungen auf*

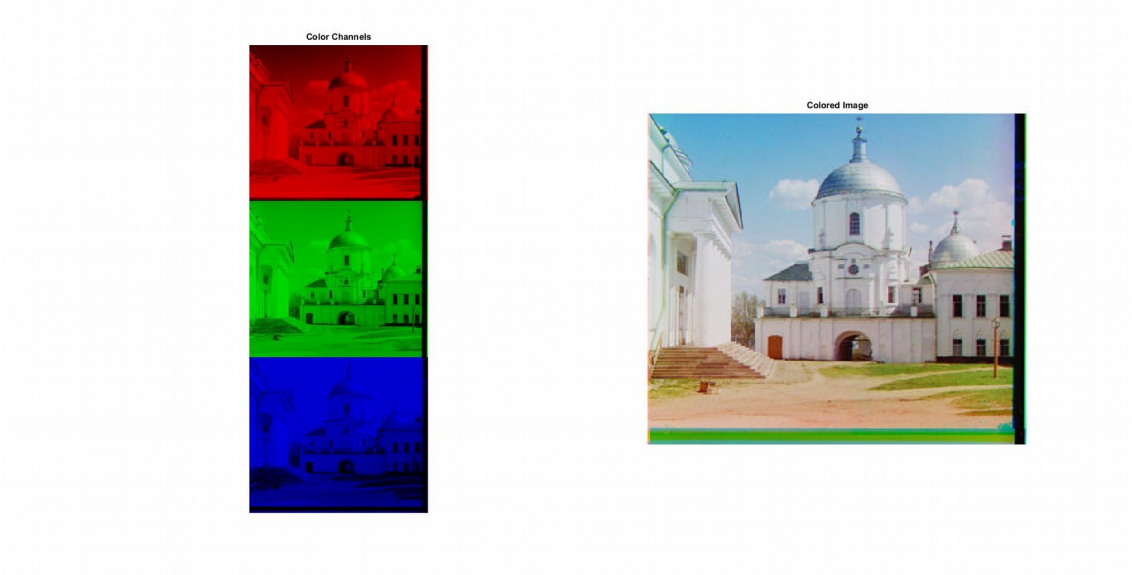


*Abbildung 4: Ergebnis für Bild 01112v*

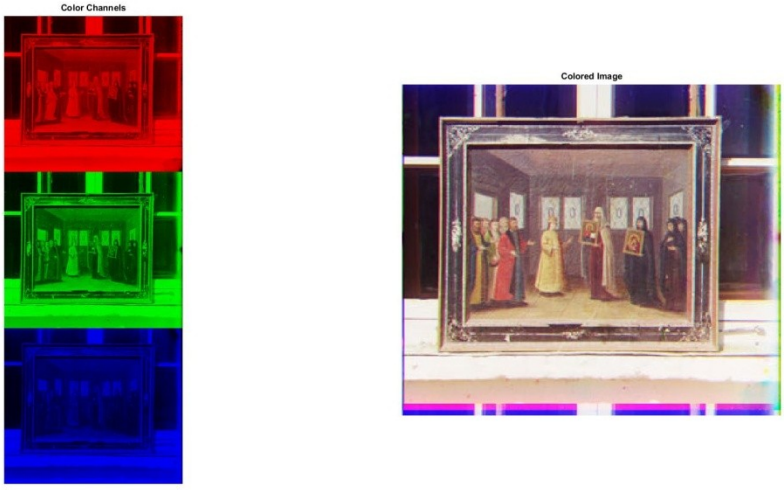


*Abbildung 5: Ergebnis für Bild 00149v*

## Assignment 2: Image Segmentation by K-means Clustering

Der zweite Teil der Aufgabe besteht darin eine Bildsegmentierung mithilfe der k-Means Methode zu durchführen.

Als Eingabe wird ein Farbbild erwartet, dieses wird (abhängig von den Farb- und Raumeigenschaften jedes Bildpunktes) in k unterschiedlichen Klassen segmentiert. Das Endergebnis ist ein RGB-Bild mit k Farben, die jeweils eine bestimmte Klasse darstellen.

## Implementierung

Der kMeans-Algorithmus besteht aus mehreren Schritten, die iterativ wiederholt werden, bis das gewünschte Resultat sich nicht mehr verändert. Als Vorverarbeitungsschritt wird auf dem Bild ein Feature-Vektor extrahiert, also alle Bildpunkte von einem 3D-Bild in einen 2D Vektor transformiert. Hierbei werden optional auch die x- und y-Koordinaten normalisiert als Features hinzugezogen. Als Start-Centroids werden k verschiedene zufällig Punkte gewählt.

Nun werden zu jedem Sample aus dem Feature-Vektor die Distanzen zu den einzelnen Centroids berechnet und das Sample mit dem Label des am kürzest entfernten Centroids klassifiziert.

Der nächste Schritt besteht darin, dass die Centroids aus den klassifizierten Samples neu berechnet werden. Dafür wird der Durchschnitt der Samples pro Centroids berechnet. Dieser dient als neuer Centroids für die jeweilige Klasse.

Der Algorithmus bricht dann ab, wenn sich die neu berechneten Centroids im Vergleich zu den vorherigen nicht mehr wesentlich verändern. Dafür berechnen wir die Distanzen der jeweiligen alten und neuen Centroids zueinander. Falls alle Distanzen unter einem Schwellwert, terminiert der Algorithmus, wenn nicht, wird der Vorgang wiederholt.

Zusätzlich beinhaltet die Lösung eine maximale Anzahl an Iterationen, wodurch der Algorithmus auch schon früher, mit einer weniger optimalen Lösung,  beendet werden kann.

## Resultat

Als Parameter für die folgenden Resultate wurden 0.003 als Schwellwert und 10 als Maximalanzahl der Iterationen ausgewählt.

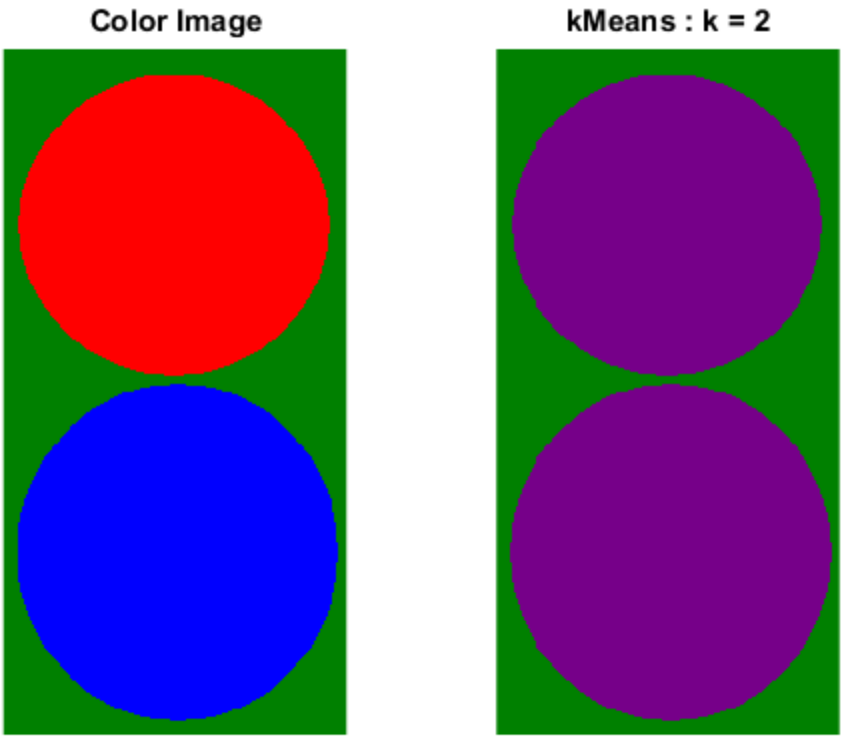


*Abbildung 6: Ergebnis für simple.png, Spatial use = true*

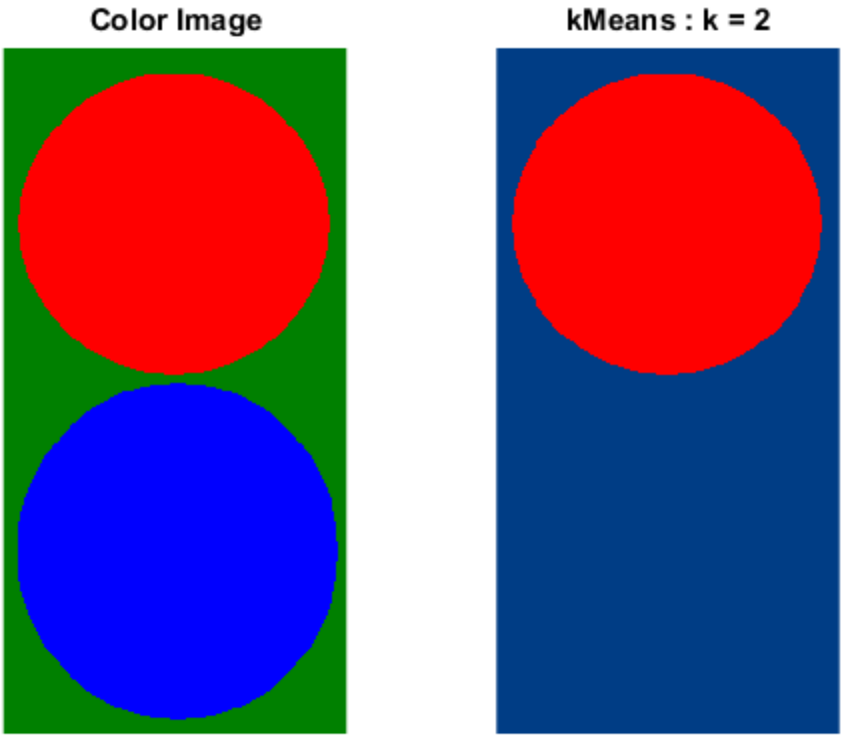


*Abbildung 7: Ergebnis für simple.png, Spatial use = false*


*Abbildung 8: Ergebnis für mm.jpg, Spatial use = true*

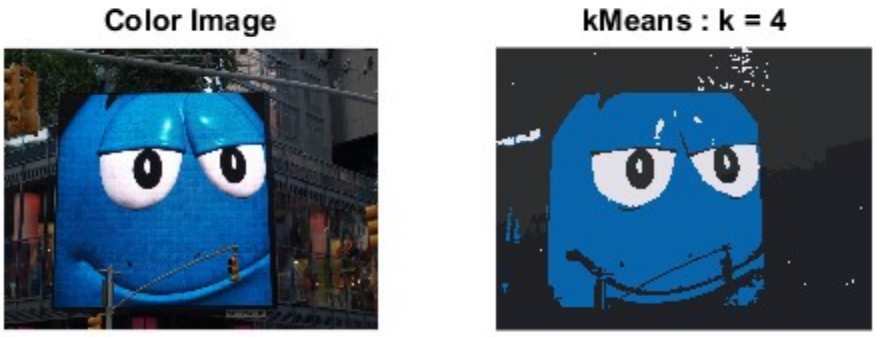


*Abbildung 9: Ergebnis für mm.jpg, Spatial use = false*

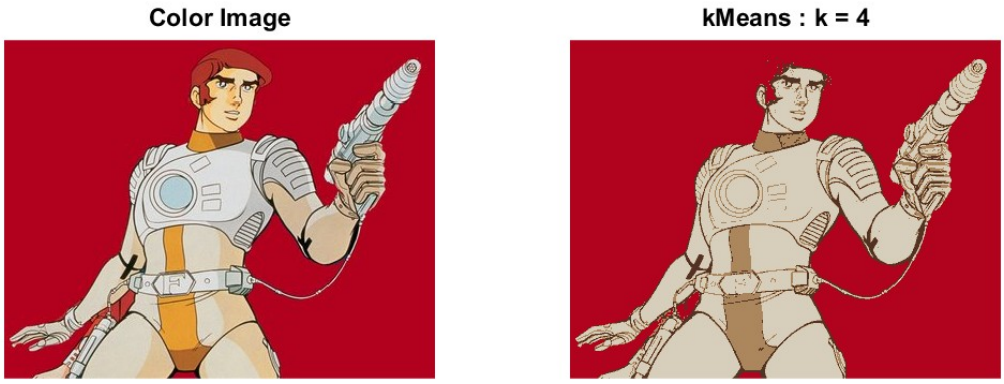


*Abbildung 10: Ergebnis für future.jpg, Spatial use = true*

## Ergebnisse für mm.jpg bei verschiedenen k

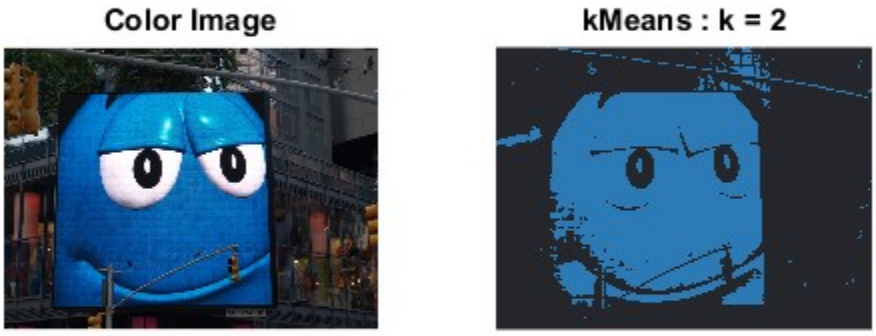


*Abbildung 11: Ergebnis für mm.jpg, Spatial use = true*


*Abbildung 12: Ergebnis für mm.jpg, Spatial use = false*


*Abbildung 13: Ergebnis für mm.jpg, Spatial use = true*


*Abbildung 14: Ergebnis für mm.jpg, Spatial use = false*


*Abbildung 15: Ergebnis für mm.jpg, Spatial use = true*

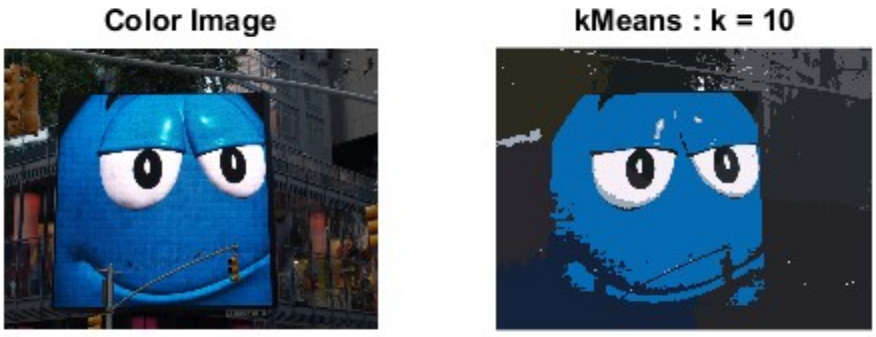


*Abbildung 16: Ergebnis für mm.jpg, Spatial use = false*

## Diskussion

Die räumlichen Merkmale eines Samples tragen dazu bei, dass oftmals größere zusammenhängende Cluster gebildet werden, obwohl sich innerhalb größere Unterschiede in der Farbe befinden. Abbildung 16 und 17 stellen dies dar. Während bei Abbildung 16 noch wesentlich mehr Struktur erkennbar ist, ist bei Abbildung 17, abgesehen von den prominentesten Formen, wesentlich weniger Struktur erkennbar.

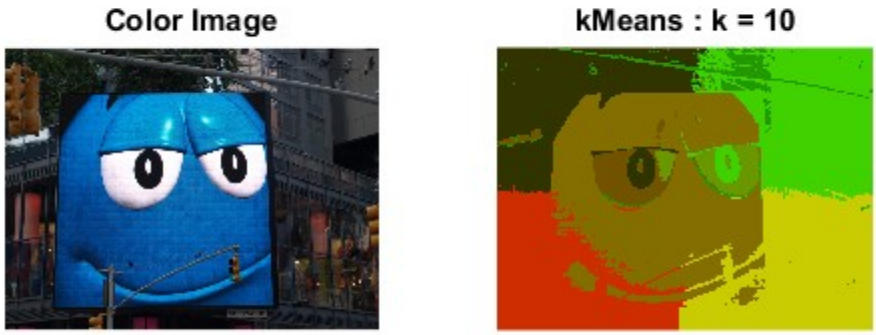


*Abbildung 17: Einfärbung der Cluster mit der räumlichen Information x = rot, y = grün*

Die räumliche Komponente kann gerade bei geringen Anzahl an Klassen das Ergebnis stark verschieben, bietet aber einen guten Anzahl sehr unstrukturierte Bilder, die z.B. durch Rauschen verunstaltet sind, sauberer zu klassifizieren. Einzelne Farb-Ausreißer können so besser ausbalanciert werden.

Ein potentieller Schwachpunkt des Algorithmus ist zweifelsohne die zufällige Auswahl der Start-Punkte. Hat man ein Bild, dass z.B. einen starken Blauton hat, einer der Startpunkte allerdings einen hohen Rotanteil besitzt, kann es passieren, dass dieser Klasse keine Werte zugeordnet werden und der Cluster somit aus dem Ergebnis fällt. Ein Umgehen dieses Problem wäre beispielsweise ein zufälliges Auswählen der Startpunkte aus den bereits vorhandenen Features. Somit wäre gesichert, dass jeder Klasse zumindest ein Element zugeordnet wird.

## Assignment 3: Scale-Invariant Blob Detection

In der dritten Aufgabe geht es darum, zusammenhängende Regionen zu erkennen („Blobs“). Dafür wird der Laplacian of Gaussian Blob Detector verwendet.

## Implementierung

Um die Aufgabe zu lösen, muss zuerst der Laplacian Scale Space errichtet werden.
Für die Anzahl an gegebenen Levels wird pro Level ein skalierungs-normalisierter LoG-Filter mit dem aktuellen Sigma erstellt.
Realisiert wurde dies mit einer Schleife, die von 1 bis zur Anzahl der Level geht.

Nachdem das aktuelle Sigma berechnet wurde, wird die Filtergröße erstellt und der LoG Filter auf das Bild angewendet. Danach wird das Filterergebnis im aktuellen Scale Space gespeichert.

Der nächste Schritt ist die non-maximum suppression.
Zuerst wird pro Level im Scale Space mit Hilfe der Funktion „ordfilt2“ die lokale Maxima gesucht und im Max Space gespeichert. Die Variable „suppression_size“ ist dabei standardmäßig auf 10 gesetzt, kann aber optional auch eingegeben werden.

Um die Ausgabe der Blobs zu erzielen wird die Methode „show_all_circles“ verwendet, die als Eingabe das Bild, die Zeilen- und Spaltenkoordinaten der Blob-Zentren (cx und cy) und den Blob-Radius (rad).
Die Werte für cx, cy und rad müssen demnach für die Übergabe vorbereitet werden.
In einer Schleife wird im Max Space nach den Werten gesucht, die über dem gegebenen Schwellwert liegen.
Der korrekte Radius wird gefunden indem der aktuelle Scale mit der Wurzel aus 2 multipliziert wird, cx und cy sind die gefundenen Zeilen und Spalten aus dem Max Space.

## Resultat

Das Resultat zeigt je nach gewähltem Schwellwert und Suppression-Größe die zusammenhängenden Regionen. Diese Blobs werden mit Kreisen angezeigt.

Beim Bild „butterfly“ sieht das Ergebnis mit den Standardwerten folgendermaßen aus:

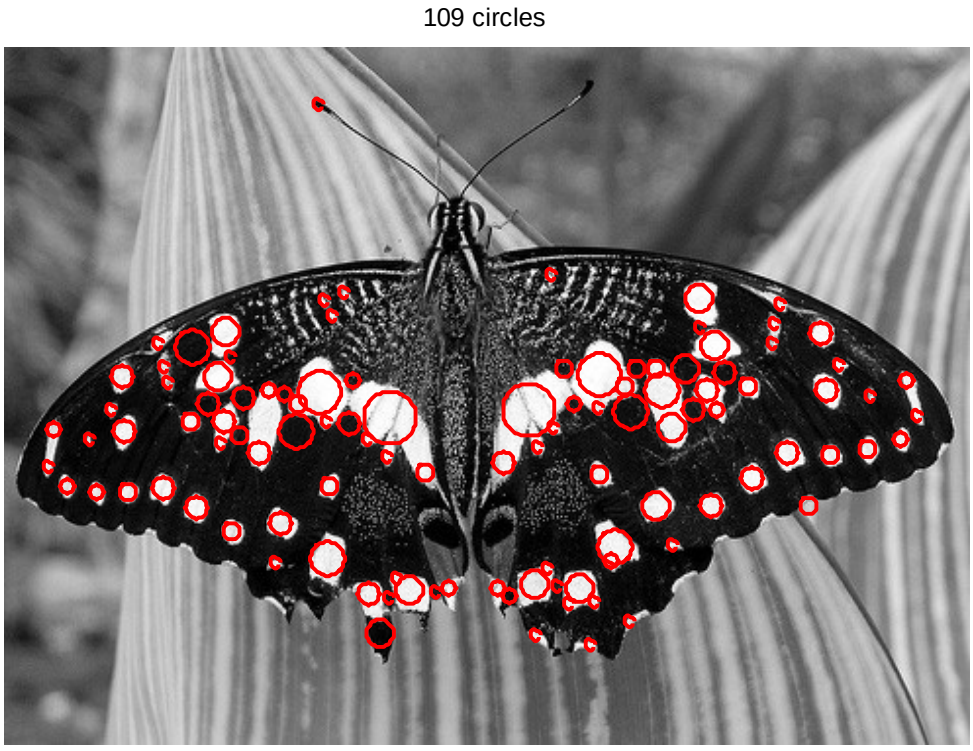


*Abbildung 18: butterfly.jpg: Originalbild, 109 Regionen erkannt*

Wenn das Bild auf die halbe Größe reduziert wird ändert sich das Ergebnis etwas. Das Resultat ist

in folgender Abbildung zu sehen:

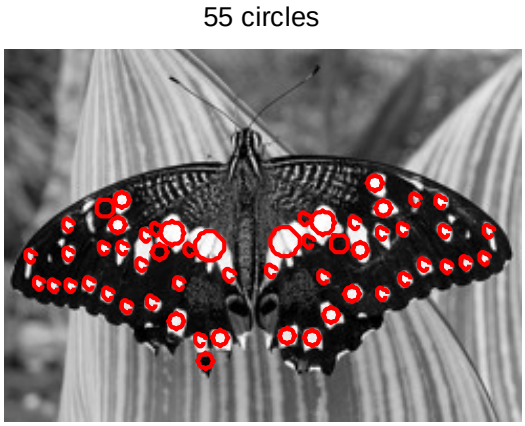


*Abbildung 19: butterfly.jpg: Verkleinerte Version, 55 Regionen erkannt*

Es werden im Großen und Ganzen dieselben Regionen gefunden, dadurch dass das Bild jedoch kleiner ist, werden Regionen, die im Original schon recht klein waren nicht mehr erkannt.
Im selbst ausgewähltem Bild „tree“ ist das Ergebnis schon eindeutiger. Im Originalbild werden 32 Regionen erkannt, in der verkleinerten Version verringert sich die Detektion auf 30.
Somit lässt sich sagen, dass die Methode zum Großteil skalierungsinvariant ist.

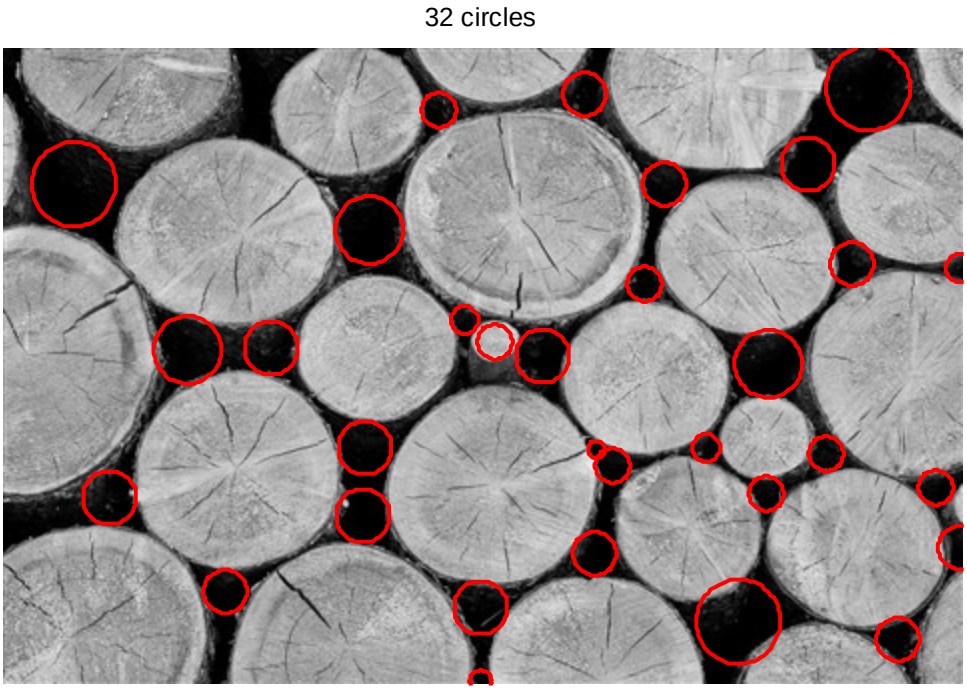


*Abbildung 20: tree.jpg: Originalbild, 32 Regionen erkannt*

Als nächstes haben wir den Keypoint 23 gewählt und stellen das Ergebnis vom LoG für alle

Skalierungen sowohl im Originalbild als auch in der verkleinerten Version dar. Die Ergebnisse für „butterfly“ und „tree“ sehen dabei so aus:

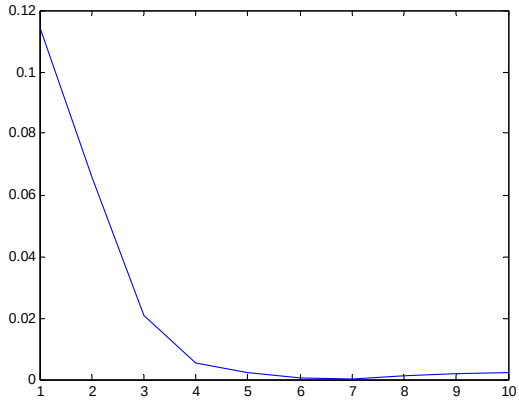


*Abbildung 21: butterfly.jpg, Originalgröße*

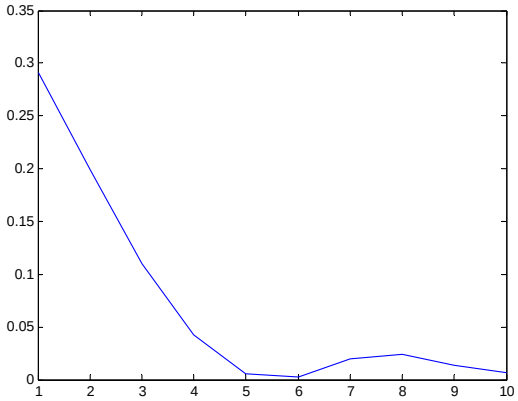


*Abbildung 22: butterfly.jpg, verkleinert*

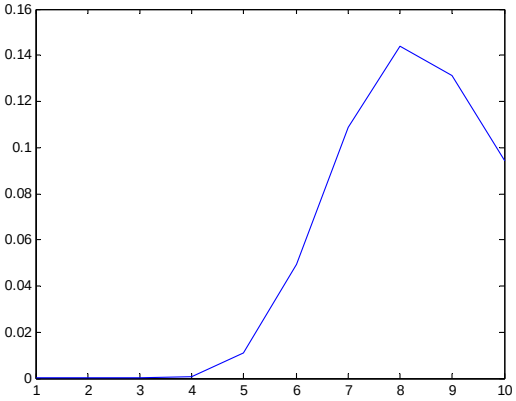


*Abbildung 23: tree.jpg, Originalgröße*

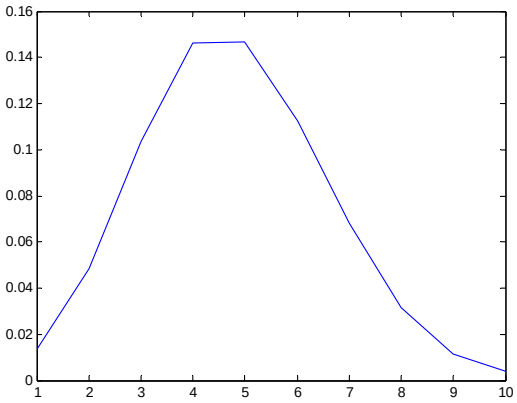


*Abbildung 24: tree.jpg, verkleinert*

Für „butterfly“ ist das Maximum des LoG 0,116 bei Level 1 im Original und 0,258 ebenfalls bei Level 1 in der verkleinerten Version.
Bei „tree“ ist das Resultat beim Originalbild 0,14 bei Level 8 und beim verkleinerten Bild 0,142 bei Level 4 und 5.

Die Unterschiede zwischen den Bilderversionen lassen sich so erklären, dass der LoG das lokale Extremum bei niedriger Skalierung für kleinere Regionen zurückgibt.