Report zu Übung 2

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

Assignment 4: Image Stitching

Für diese Aufgabe sollen von Sequenzbildern Interest Points gefunden und anhand dieser ein Panoramabild erstellt werden. Dieses Problem gliedert sich in drei Unterpunkte. Zuerst müssen mittels SIFT Interest Points gefunden werden, danach müssen die gefundenen Punkte miteinander zugeordnet werden und zu guter Letzt wird dann das fertige Bild erstellt.

A. SIFT Interest Point Detection

Implementierung

Für Task A sollen die SIFT Features der Bilder extrahiert werden. Dafür wird jedes Bild in ein Grauwertbild umgewandelt und danach die Funktion „vl_sift“ aufgerufen. Abschließend werden die Bilder samt den gefundenen Features geplottet.

Resultat

Folgende Grafik zeigt die gefundenen SIFT Features der Bilder „campus1.jpg“ bis „campus.5.jpg“.

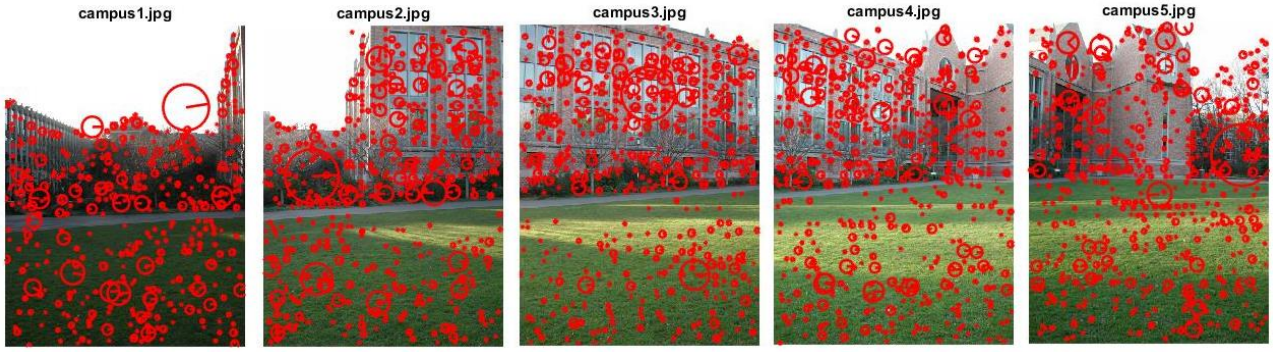

Abbildung 1 Output der SIFT Features

Die Kreise sind die gefundenen Keypoints, die Größe der Kreise gibt die Größe des Keypoints und die Linien in den Kreisen geben die Ausrichtung der Keypoints an.

B. Interest Point Matching and Image Registration

Implementierung

Um die Interest Points zuzuordnen werden einige Schritte ausgeführt.

Für je zwei aufeinanderfolgende Bilder werden folgende Schritte ausgeführt:

- Zuerst werden die SIFT Features und Deskriptoren mittels „vl_sift“ extrahiert (was eigentlich schon in Task A gemacht wurde). Mittels „vl_ubcmatch“ werden die Bilder mithilfe ihrer Deskriptoren gematcht. Um das Ergebnis zu plotten wird die Funktion „match_plot“ verwendet, die als Parameter die zwei Bilder und ihre Matching-Punkte braucht.
- Da nicht alle Zuordnungen korrekt sind, wird nun RANSAC angewandt. Hierfür wird eine Schleife von 1 bis 1000 angewandt. Bei jeder Iteration werden 4 zufällige Paare ausgewählt. Als nächstes muss die Homographie der zwei Bilder geschätzt werden, was mithilfe der Funktion „cp2tform“ passiert. Danach werden die Matching-Punkte des ersten Bildes mittels „tformfwd“ transformiert.
  Nun muss die Anzahl der „Inlier“ gefunden werden. Dazu wird die Euklidische Distanz zwischen den transformierten Matching-Punkten des ersten Bildes und den Matching-Punkten des zweiten Bildes berechnet. Als „Inlier“ gelten nun alle Distanzen, die unter einem gewissen Schwellwert liegen.
  Wenn die aktuelle Homographie mehr „Inlier“ hat als die vorige, wird diese Transformation gespeichert.
- Zuletzt wird das erste Bild auf das zweite Bild mittels der Funktion „imtransform“ transformiert und geplottet.

Resultat

Nach Schritt 2 sehen die Paare für die „campus“-Bilder so aus:

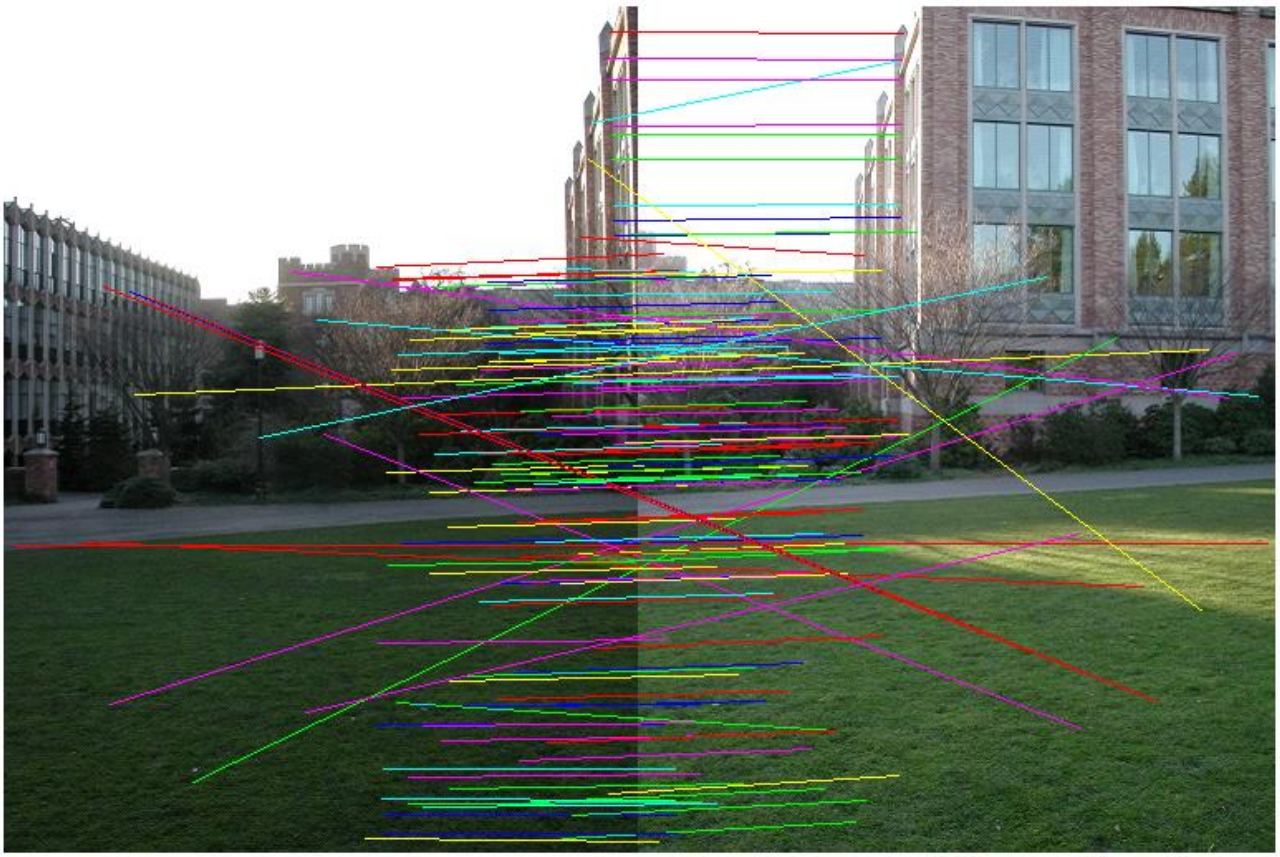

Abbildung 2 Gefundene Paare mit vl_ubcmatch vor RANSAC - campus1 auf campus2

Die Funktion „vl_ubcmatch“ verwendet als Parameter die Deskriptoren des ersten Bildes und die Deskriptoren des zweiten Bildes und findet die zusammengehörigen Punkte. Dabei wird der Algorithmus von D. Lowe verwendet, um unklare Paare auszuschließen.

VL_UBCMATCH(DESCRS QUERY,DESCRS DATABASE)

Die Funktion „vl_ubcmatch“ nimmt zwei SIFT Deskriptoren als Argumente und berechnet alle möglichen Matches dieser zwei. Dabei ist die Reihenfolge der Eingabeparameter von entscheidender Bedeutung. Das liegt daran, dass für jeden einzelnen Wert im ersten Parameter (die Query) alle möglichen bzw. geeigneten Matches im zweiten Parameter (die Datenbank) gesucht werden. Der Matchvorgang erfolgt mithilfe des „nearest neighbor“-Verfahrens (der Wert mit der kleinsten Euklidische Distanz wird als Match akzeptiert). In der Praxis ist es schwer das beste Ergebnis zu bekommen, weswegen die Hilfe von Heuristiken (wie der Best-Bin-First-Algorithmus) verwendet werden, um den „nearest neighbor“ mit der höchsten Wahrscheinlichkeit ausfindig zu machen. Da jeder Wert in der Query mehrere und nicht immer richtige Matches haben könnte, wird ein weiteres Verfahren angewandt, um falsche Matches auszuschließen. Bei diesem Verfahren wird das Verhältnis vom „nearest neighbor“ zum „second nearest neighbor“ berechnet und als Wahrscheinlichkeit verwendet, um auszusagen, ob der gefundene Match auch der richtige ist. Wie in Abbildung 2 zu sehen ist, können die angewandten Matchingvorgänge viele richtige Matches finden, dennoch kommt es aber manchmal zu „false positive“-Ergebnissen.

Nachdem RANSAC ausgeführt wurde sehen die gefundenen Paare wesentlich richtiger aus. Es gibt keine Ausreißer mehr wie man anhand Abbildung 3 sehen kann. Der Unterschied zwischen Abbildung 2 und 3 ist der, dass nach RANSAC falsche Paare eliminiert wurden und das Ergebnis nun korrekt ist.

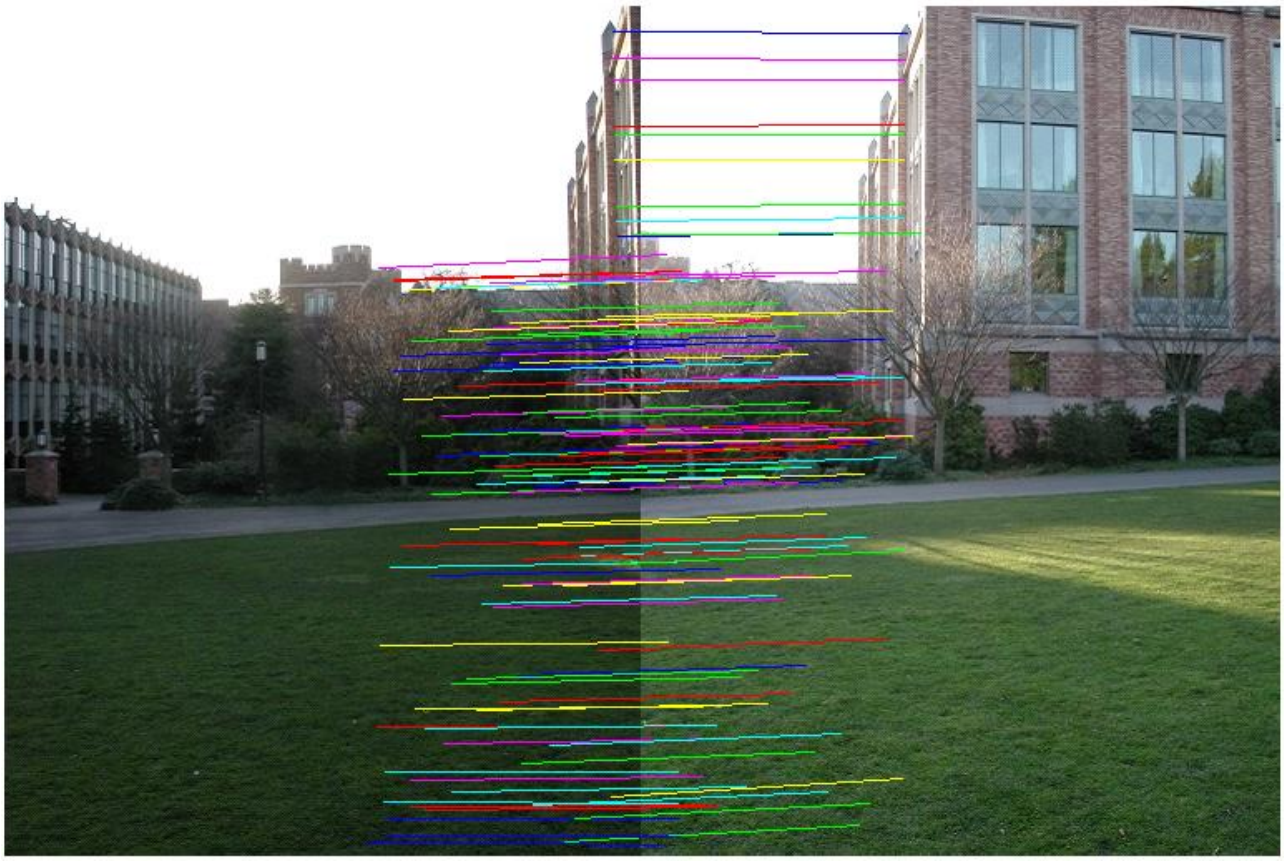

Abbildung 3 Gefundene Paare nach RANSAC - campus1 auf campus2

Abbildung 4 zeigt die absoluten Differenzen des Referenzbildes zu dem transformierten Bild. Gemeinsame Bereiche des Bildes sind im Optimalfall sehr dunkel bis schwarz. Unterschiedliche Beleuchtungen der Bilder machen dies allerdings im Normalfall nicht möglich. Trotzdem sind möglichst dunkle Ergebnisse ein Indiz für eine gute Lösung.

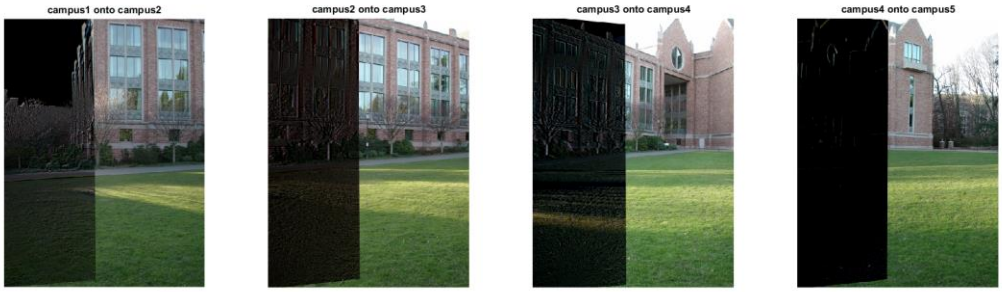

Abbildung 4 Absolute Differenzen des Referenzbildes zu dem transformierten Bild

Um zu überprüfen, ob das Verfahren invariant zu Rotation und Skalierung ist wurde das Bild „campus2.jpg“ verkleinert und rotiert. Das Ergebnis nach der Angleichung lässt sich in folgenden Abbildungen begutachten:

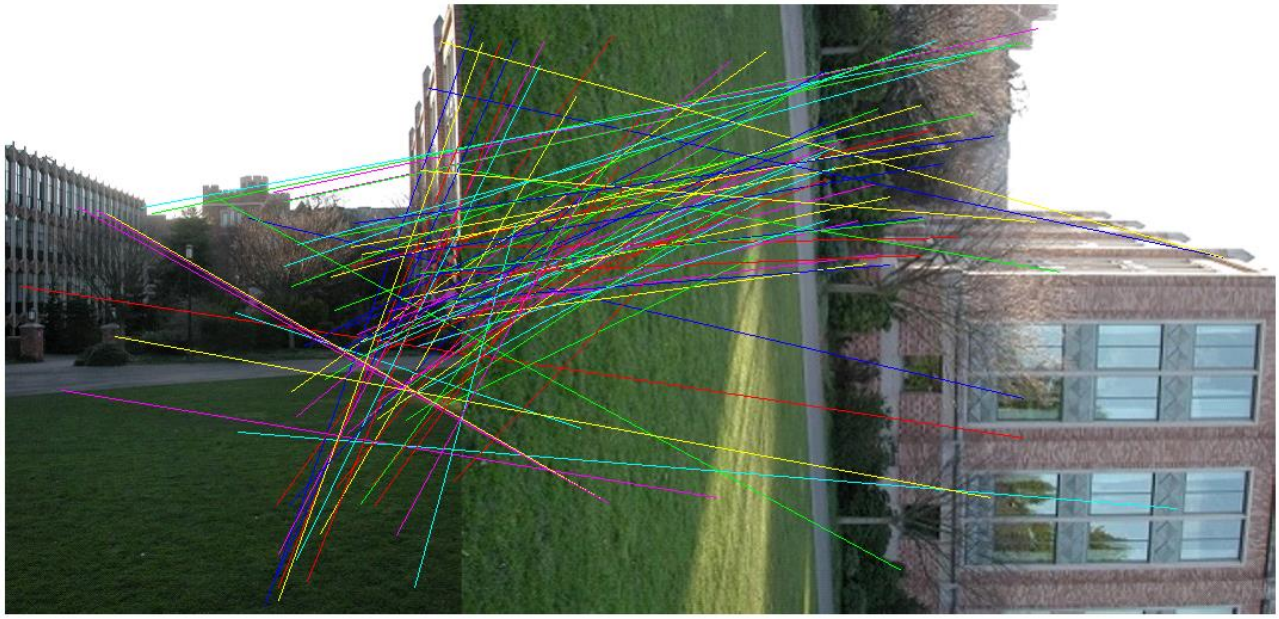

Abbildung 5 Gefundene Paare mit vl_ubcmatch - campus1 auf campus2

Abbildung 6 Gefundene Paare nach RANSAC - campus1 auf campus2

Man kann also sehen, dass beide Verfahren invariant zu Skalierung und Rotation sind, da trotz der Veränderung des zweiten Bildes noch immer die richtigen Paare gefunden wurden.

C. Image Stitching

Implementierung

Nachdem die zusammengehörigen Paare gefunden wurden kommt nun das Zusammenfügen zu einem großen Panoramabild.

Da wir mit 5 Sequenzbildern arbeiten, verwenden wir Bild 3 als Referenzbild und gleichen die anderen Bilder an dieses an.
Zuerst müssen die Homographien bestimmt werden (bereits in Task B erledigt). Mittels einer if-Abfrage wird geschaut, ob sich das gerade verwendete Bild rechts oder links vom Referenzbild befindet. Pro Bild werden die entsprechenden Transformationen angewandt und gespeichert.

Um die Breite und Höhe des endgültigen Bildes zu bestimmen werden alle Ecken aller Bilder auf das Referenzbild transformiert, danach wird eine projektive Division ausgeführt und zuletzt wird die richtige Dimension des Bildes mittels den Minimal und Maximalwerten bestimmt.

Im letzten Schritt werden alle Bilder zum Referenzbild hinzugefügt und gleichzeitig mittels „feathering“ die Überschneidungen vermischt.
Dazu wird ein alpha-Kanal erzeugt, in dem der Wert von alpha an den Randpixel 0 ist und 1 an den maximalen Distanzen vom Rand. Dies wird unter anderem mit der Funktion „bwdist“ bewerkstelligt. Mittels „imtransform“ werden dann die Bilder zur Basisebene transformiert.
Um die richtigen Farben zu bekommen wird folgende Formel angewandt:

$$O(x,y) = \frac{\sum_{i=1}^{n}(R_i, G_i, B_i) \cdot \alpha_i}{\sum_{i=1}^{n} \alpha_i}$$

Resultat

Das Ergebnis für die Bildsequenzen sieht wie folgt aus:

Abbildung 7 Ergebnis für campus

Abbildung 8 Ergebnis für officeview

Abbildung 9 Ergebnis für cat

Abbildung 10 Ergebnis für wall

Die Ergebnisse sind sehr gut geworden, weisen aber manchmal einige Ungereimtheiten auf. Beim Bild „campus“, beispielsweise, können beim Vergrößern des Bildes verschwommene Bereiche („ghosting“) wahrgenommen werden, wie folgende Abbildung zeigt:

Abbildung 11 Verschwommene Ränder beim Übergang

Beim eigenen Bild sind die Fehler natürlich aufgrund schlechter Bildqualität und Beleuchtung noch auffälliger.

Abbildung 12 zeigt das Ergebnisbild ohne Blending. Hierbei wird pro transformiertes Bild eine Maske berechnet, die dort 1 ist, wo das Bild Farbwerte enthält. Das teilfertige Panorama wird mit der invertierten Maske multipliziert und dann das weitere Bild hinzuaddiert. So werden sukzessiv die Farbwerte vom rechteren Bild übernommen und die des linken überschrieben. Wie im Beispiel erkennbar führt das unwillkürlich zu Artefakten und zu harten Grenzen in den Farbübergängen, da die Farbverläufe der Bilder unterschiedlich sind.

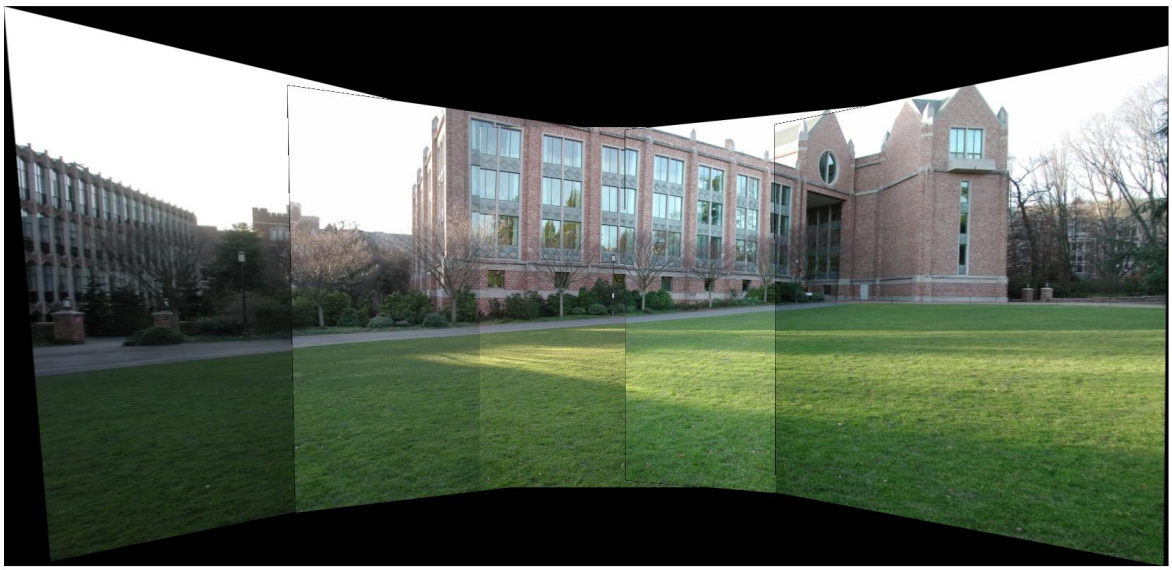

Abbildung 12 Campus Panorama ohne Blending

Assignment 5: Scene Recognition with Bag of Visual Words

Implementierung

Assignment 5 besteht aus den Dateien „main.m“, „BuildKNN.m“, „BuildVocabulary.m“ und „ClassifyImages.m“.

Zuerst wird die Funktion „BuildVocabulary“ ausgeführt, die von einem Trainingsset ein Vokabular aus visuellen Wörtern aufbaut. Dazu wird eine verschachtelte Schleife, die über die Dateiordner und die einzelnen Bilder iteriert, verwendet. Für jedes Bild wird mithilfe der Funktion „vl_dsift“ die Dense SIFT Deskriptoren berechnet, davon werden 100 zufällige Werte (die Spalten) genommen und diese in einer einzigen Deskriptorenmatrix gespeichert. Am Ende wird auf alle gefundenen Deskriptoren K-Means ausgeführt, um diese in einer als Parameter übergebene Anzahl an Cluster zu klassifizieren.

Die nächste Funktion ist „BuildKNN“. Nun soll eine Feature Repräsentation für jedes Bild aus dem Trainingsset erstellt werden welche dann für die Klassifikation später verwendet werden kann. Es wird wieder eine verschachtelte Schleife verwendet, welche über jeden Ordner und jedes Bild iteriert. Pro Bild werden wieder die Dense SIFT Deskriptoren berechnet und die Kategorie des Bildes (der Ordner aus dem das Bild gelesen wurde) wird gespeichert.
Als nächster Schritt wird „knnsearch“ ausgeführt. Dabei wird für jedes Feature (die Deskriptoren) der dazugehörige Cluster (das Ergebnis von „BuildVocabulary“) bestimmt. Das Ergebnis ist ein Vektor mit den Indizes der Cluster (Variable: IDX). Anschließend wird aus den IDX Werten ein Histogramm gebildet und die Anzahl der Elemente pro Bin ausgerechnet (histc). Das Ergebnis wird normalisiert und in die Trainingsmatrix gespeichert.

Der letzte Schritt ist die Funktion „ClassifyImages“, welche nun Bilder aus einem Testset klassifizieren soll. Es läuft dabei ähnlich wie „BuildKNN“ ab, nur das nach dem „histc“-Aufruf mithilfe der „knnclassify“-Funktion die berechneten Histogramme zu einer Kategorie klassifiziert werden. Anschließend wird in einer Tabelle mitgezählt, wie oft ein Bild richtig bzw. falsch klassifiziert wurde.

Resultat

Das Resultat unserer Klassifizierung wird in folgender Tabelle gezeigt.

Tabelle 1 Confusion Matrix

Klassen	Bedroom	Forest	Kitchen	Living Room	Mountain	Office	Store	Street
Bedroom	25	0	13	30	1	14	8	9
Forest	0	91	0	0	7	0	0	2
Kitchen	6	0	49	17	0	17	10	1
Living Room	10	0	21	36	0	11	17	5
Mountain	3	4	0	1	73	1	7	11
Office	4	0	14	9	0	73	0	0
Store	7	3	7	19	0	4	50	10
Street	3	0	3	5	1	1	25	62

Die Werte in der Diagonale sind die richtig klassifizierten Bilder. Es fällt dabei auf, dass die Klasse „Forest“ von allen Klassen am besten erkannt wurde. Es wurden Bilder aus dem Bereich „Forest“ dabei 7 Mal zur Klasse „Mountain“ und 2 Mal zur Klasse „Street“ klassifiziert.
Die Klassen „Mountain“ und „Store“ wurden ebenfalls fast immer richtig erkannt.
Anzumerken ist auch, dass die Klassen „Bedroom“ und „Living Room“ oft miteinander verwechselt werden. So wurden Bilder der Klasse „Bedroom“ 30 Mal als „Living Room“ erkannt (das ist öfters als die 25 Male die richtig erkannt wurden). Des Weiteren wurden 21 Bilder der Klasse „Living Room“ als „Kitchen“ erfasst.
Die richtige Klassifizierung liegt bei ungefähr 57,4%.

Tabelle 2 Confusion Matrix der eigenen Testbilder

Klassen	Bedroom	Forest	Kitchen	Living Room	Mountain	Office	Store	Street
Bedroom	1	0	0	0	0	1	0	0
Forest	0	7	0	0	0	0	0	1
Kitchen	0	0	1	0	0	0	0	0
Living Room	0	0	0	1	0	0	0	0

Für die eigenen Testfotos wurden Bilder aus 4 Kategorien gewählt. Es gibt 2 Fotos zu „Bedroom“, 8 Fotos zu „Forest, und jeweils ein Foto zu „Kitchen“ und „Living Room“.
Das Ergebnis ist erfreulich: „bedroom1.jpg“ wurde fälschlich als „office“ klassifiziert und „forest3.jpg“ wurde als „Street“ erkannt. Alle anderen Bilder wurden richtig erfasst, was einer Erfolgsrate von 83,3% entspricht.