



Interactive Systems

Image Descriptors und Panoramabilder

WiSe 2017/18

Ziel dieser Übungsaufgabe ist es, sich ein tieferes Verständnis für markante Bildmerkmale (Image features/descriptors), ihre Implementierung und Anwendungsbereiche zu erarbeiten und dabei NumPy und OpenCV Kenntnisse zu erweitern. Laden Sie den vorgegebenen Skelett-Sourcecode aus Moodle und implementieren Sie die notwendigen Funktionen. Erweitern und verändern Sie den Sourcecode nach Belieben. Der vorgegebene Sourcecode enthält viele weitere Anmerkungen.

Aufgabe 1: SIFT in OpenCV (1 Punkt)

Bitte setzen Sie sich mit dem SIFT descriptor in OpenCV auseinander (http://docs.opencv.org/3.1.0/da/df5/tutorial_py_sift_intro.html). Bauen Sie in Ihr bisheriges Videostream-Beispiel die SIFT feature Extraktion ein und visualisieren Sie die Keypoints wie im Bild 1 dargestellt. Hinweis: Die Kamerakalibrierung brauchen wir im Moment erstmal nicht mehr. Nutzen Sie das mitgelieferte 02_features.py zur Implementierung.

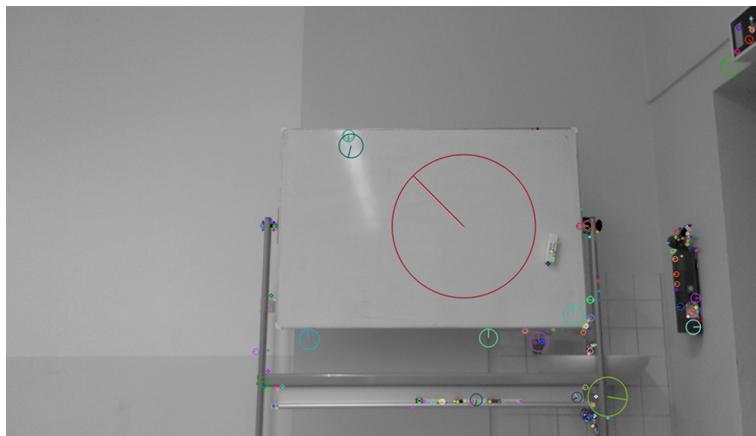


Abbildung 1: Bild des Videostreams mit der Visualisierung der SIFT Features

Aufgabe 2: Panoramabilder (2 Punkte)

Der SIFT Featuredescriptor kann unter anderem für die Erzeugung von Panoramafotos genutzt werden. Ein Beispiel dafür finden Sie in Abbildung 2. Die obere Zeile zeigt passende Featurepunkte in den Überlappungsbereichen, die über Linien verbunden werden. Die untere Zeile zeigt ein Ergebnisbild des zu implementierenden Algorithmus. Dafür sind bereits die Klasse *ImageStitcher* und das Skript *02_stitching.py* vorgegeben, die einige Funktionalität enthalten. Bitte vervollständigen Sie den Sourcecode so, dass die Liste von mitgelieferten Einzelbildern zu einem Panoramafoto zusammengesetzt wird. Beachten Sie dabei, dass nicht alle Bilder gleichzeitig zu einem Panorama zusammengesetzt werden sollen, sondern, dass immer nur zwei Bilder nacheinander verbunden werden können. D.h. dass Sie der Klasse eine manuell sortierte Liste von passenden Einzelbildern übergeben müssen. Weitere Anmerkungen finden Sie im Sourcecode.

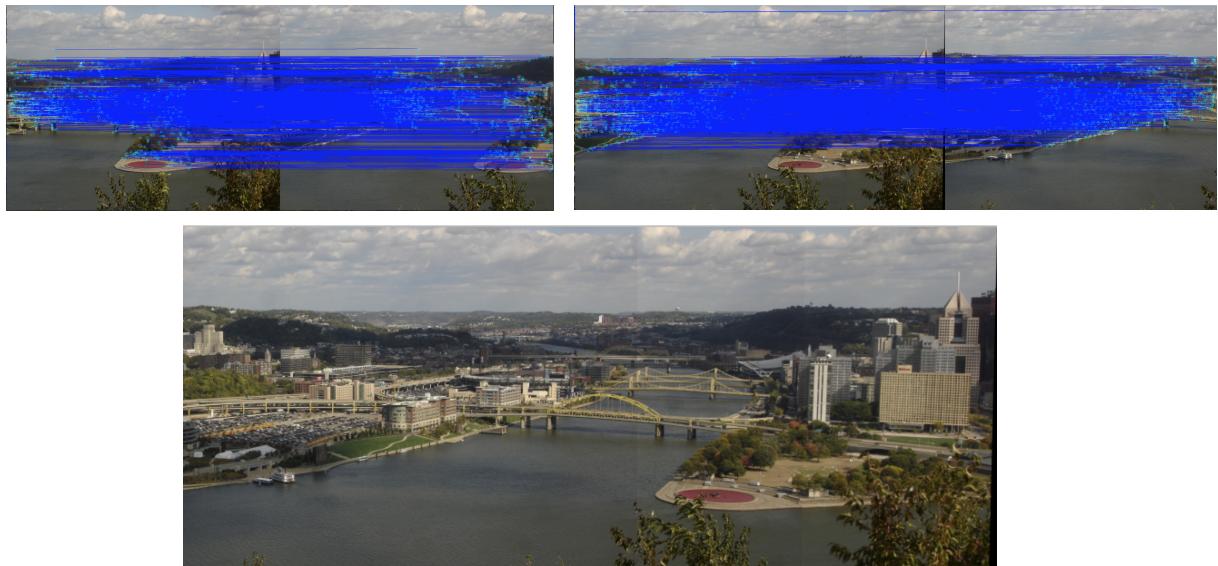


Abbildung 2: Oben: Passende Bildmerkmale in Überlappungsbereichen der Einzelbilder. Unten: Ergebnispanorama der mitgelieferten Eingabebilder.

Aufgabe 3: Simple Content-based Image Retrieval (3 Punkte)



Abbildung 3: Ergebnisse der Bildsuche von links oben nach rechts unten.

Das Ziel einer inhaltsbasierten Bildsuche ist, ähnliche Bilder bezogen auf ein Eingabebild zu finden und dabei ausschliesslich Bildmerkmale zu nutzen (also keine Keywords oder Ähnliches). Diese Suche basiert ebenfalls häufig auf der Verwendung von lokalen Featurevektoren, wie z.B. die Bildmerkmale von SIFT. Bei der Suche müssen diese Vektoren verglichen werden, um Ähnlichkeiten festzustellen. Dafür können unterschiedliche Distanzmetriken benutzt werden. In diesem sehr reduzierten Beispiel sollen Sie eine Suche implementieren, die die Vektoren auf Basis der L2-Norm vergleicht (also der euklidische Distanz der Vektoren). Gehen Sie dabei wie folgt vor und nutzen 02_image_retrieval.py:

- Laden Sie alle Bilder der mitgelieferten 'Datenbank'. Es gibt insgesamt 3 unterschiedliche Bildklassen mit etwa 5-6 Bildern.
- Erzeugen Sie eine Menge von Keypoints die in einem uniformen Gitter über dem Bild verteilt sind und an denen Sie die SIFT-Featurevektoren berechnen.
- Benutzen Sie die Keypoints, um ausschliesslich an diesen Stellen die SIFT-Featurevektoren zu berechnen. Das erzeugt genau einen Descriptor für jedes Bild in der Datenbank. Diese soll-

ten in einem Array zwischengespeichert werden. Hinweis: Wir benutzen hier nicht den SIFT-Keypointdetector.

- d) Laden Sie ein Anfragebild und extrahieren Sie für die gleichen Keypoints einen Bilddeskriptor. Vergleichen Sie den Deskriptor gegen die Deskriptoren aus der Datenbank. Nutzen Sie dafür die L2-Norm.
- e) Speichern Sie das Ergebnis in eine *PriorityQueue*.
- f) Geben Sie die Ergebnisse der Anfrage in der Reihenfolge der kürzesten Distanz aus. Das Ergebnis sollte für die Gesichter so aussehen wie in Abbildung 3 dargestellt.

Aufgabe 4: Write your own image descriptor (3 Punkte)

Implementieren Sie ihren eigenen lokalen Featurevektor und testen ihn mit den mitgelieferten Eingabebildern (02_simple_hog.py). Der Deskriptor soll ein Histogramm von Gradientenrichtungen, wie Sie von einem einfachen Kantendetektions-Filter erzeugt werden (z.B. Sobel-Filter), abspeichern. Im speziellen Fall soll der Deskriptor nur für einen einzigen Keypoint (in der Mitte des Bildes) und einer vorgegebenen Fentergröße (z.B. 11) extrahiert werden. Hinweis: Schauen Sie sich die Funktionen *cv2.Sobel*, *cv2.phase* und *cv2.magnitude* an. Die Ergebnishistogramme sollen **in etwa** so aussehen wie in Abbildung 4 dargestellt.

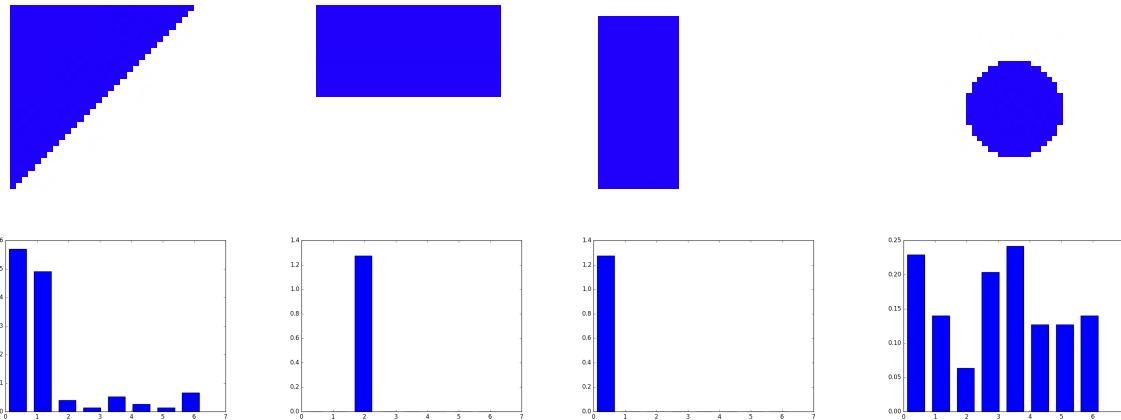


Abbildung 4: Histogram of Oriented Gradients Testauswertung. Obere Reihe zeigt eine Reihe von mitgelieferten Beispielbildern. Untere Reihe zeigt das Histogramm für die Gradienten im Bild.

Aufgabe 5: Eigenvalues zur Kantenerkennung (1 Punkt)

Wie in der Vorlesung besprochen, benötigt man zur Berechnung der Harris Corner Detectors die Eigenwerte der Matrix

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_{xx} & I_x I_y \\ I_y I_x & I_{yy} \end{bmatrix}$$

Die Eigenwerte unterscheiden sich dann für Kanten, glatte Regionen und Ecken in ihren Wertebereichen. Für glatte Regionen im Bild sind beide Eigenwerte sehr klein. Bei Kanten ist ein Eigenwert um mindestens eine Größenordnung größer als der andere und bei Ecken sind beide in der gleichen Größenordnung nicht relativ groß (> 1). 02_eigen.py enthält drei 'Beispielbilder' 3x3 Pixel groß. Berechnen Sie die Eigenwerte der Bilder und vollziehen sie die Unterschiede in den Wertebereichen nach und geben Sie die Werte aus. Nutzen Sie für die Implementierung bitte die Kommentare im Sourcecode.

Aufgabe 6: Readings (freiwillig)

Bitte lesen sie die Veröffentlichungen, die im Bereich *Readings* im Moodle zu finden sind und rekapi-tulieren Sie die Vorlesung.

Abgabe: Dies ist *Übungsaufgabe 2*; die Bearbeitungszeit der Aufgabe ist für ca. 2 Wochen ausgelegt.
Die Abgabe erfolgt via Moodle zum angegebenen Zeitpunkt.