BIV-Übung 7

1 Drehung, Interpolation und Ortsfrequenzraum

Das Bild `bild.png` soll mit Hilfe der Funktion `imrotate` gedreht werden. Die Größe des Bildes soll sich dabei nicht ändern (dazu den Hilfetext durchlesen: `doc imrotate`).

a) Drehen Sie das Bild um 10 Grad, das Ergebnis wieder um 10 Grad, und wiederholen Sie diesen Vorgang mit einer `for`-Schleife bis insgesamt eine Drehung um 360 Grad erfolgt ist.

   Die Funktion `imrotate` bietet drei verschiedene Interpolationsvarianten an. Führen Sie die beschriebene mehrfache Rotation für jede der drei Verfahren aus und lassen Sie sich das Originalbild und die drei Ergebnisse nach 360 Grad Rotation darstellen.

b) Vergleichen Sie die vier Bilder. Wie ändert sich das Original (insbesondere der Hirnstamm = die senkrechte Linie in der Bildmitte) nach der Rotation mit den drei verschiedenen Interpolationsmethoden im Vergleich zum Original und im Vergleich untereinander?

   Warum entstehen bei Nächster-Nachbar-Interpolation starke Artefakte?

c) Lassen Sie sich die logarithmierten Powerspektren (logarithmierten Betragsquadrate der Fouriertransformierten) des Originals und der Ergebnisse anzeigen.

   Hierzu müssen Sie zuerst die Fouriertransformation mit `fft2` durchführen und können dann die Funktion `ShowLogPower(fft_image, caption)` von der Moodle-Seite verwenden.

   Wieder die Frage, nun im im Fourierraum (=Ortsfrequenzraum): Wie unterscheidet sich das Original von dem Ergebnissen nach der Rotation?

d) Lassen Sie sich die logarithmierten Powerspektren der Bilder `arches.png` und `theorem.png` anzeigen.

   Wie unterscheiden sich die Spektren der beiden Bilder? Was vermuten Sie als Ursache?

e) Extrahieren Sie Kanten im Bild `theorem.png` mittels eines (theoretisch/scheinbar) *idealen* XXXpass-Filters (XXX steht entweder für „Hoch“ oder für „Tief“).

   Hierfür finden Sie die Hilfsfunktion `CreateCenteredCircle` in den Dateien zum Aufgabenblatt.

   - Steht XXX für „Hoch“ oder für „Tief“?
   - Welche Artefakte im Ergebnisbild entstehen, die diese Methode zu einem schlechten Kantenfilter machen?

2 Optional: Fouriertransformation mit Matlab

Machen Sie sich mit den Funktionen von MATLAB zur Berechnung und Anzeige von fouriertransformierten Bildern vertraut. Lesen Sie dazu die Hilfeseite von MATLAB unter *Image Processing Toolbox > Image Analysis > Image Transforms > Fourier Transform.*[1]

Beachten Sie insbesondere die folgenden Besonderheiten von MATLAB:

- Die Ortsfrequenz $\vec{k} = \vec{0}$ des Bildes steht *nicht* in der Mitte, sondern zu Beginn des Arrays.
- Verschieben Sie daher vor der Anzeige das fouriertransformierte Bild mit der MATLAB-Funktion `fftshift` so, dass die Ortsfrequenz $\vec{k} = \vec{0}$ in der Mitte des Bildes ist.

[1] Auch im Internet: `http://www.mathworks.de/de/help/images/fourier-transform.html`

- Alle Werte (insbesondere also der Wert bei $\vec{k} = \vec{0}$) sind in MATLAB zudem mit der Anzahl der Bildpunkte multipliziert.
- Außerdem ist es für die Anzeige sinnvoll, das fouriertransformierte Bild auf ein quadratisches Gitter zu entzerren (MATLAB-Funktion `imresize`). *Warum?*

Beachten Sie, dass MATLAB für die Fouriertransformation ein Bild mit Datentyp `double` erwartet. Als Wertebereich für `double`-Bilder wird (insbesondere bei der Anzeige) $0 \ldots 1$ angenommen. Ist der Wertebereich ein anderer, ist der zweite Parameter von `imshow` nützlich: `imshow(I,[low high])` und `imshow(I,[])` (automatische Histogrammspreizung) (siehe auch die Beispiele zur Fouriertransformation auf der oben erwähnten Hilfeseite).

a) Vollziehen Sie die Funktion `ShowLogPower` nach. Sie stellt das logarithmierte Powerspektrum (Logarithmus des Betragsquadrats[2] der Fouriertransformierten) als Bild dar (denken Sie an die Verschiebung mit `fftshift`!). MATLAB-Befehl zur Berechnung des Betrags: `abs`.

   Statt der Darstellung als Grauwert-Bild wird eine Regenbogen-Farbtabelle zur Anzeige verwenden: Der Befehl MATLAB-Befehl hierfür ist `colormap(jet)`.

   Warum ist es sinnvoller, das logarithmierte Powerspektrum statt das Powerspektrum selbst anzuzeigen?

b) Das Ergebnisbild der `fft2`-Funktion besteht aus komplexen Zahlen. Eine komplexe Zahl z lässt sich darstellen als Summe $z = a + i \cdot b$ eines rein reellen Anteils a und eines rein imaginären Anteils $i \cdot b$ mit reellen Zahlen a, b (MATLAB: `real` bzw. `imag`) oder in Exponentialdarstellung als $z = re^{i\phi}$, wobei $r = |z|$ der Betrag von z ist und ϕ als Phase (auch: Winkel oder Argument) bezeichnet wird (beides reelle Zahlen). Es gilt $a = r \cdot \cos\phi$ und $b = r \cdot \sin\phi$. Die Phase ϕ lässt sich demnach mit Hilfe des Arcustangens berechnen (Umkehrfunktion des Tangens tan, wobei $\tan\phi = \frac{\sin\phi}{\cos\phi}$). Die entsprechende MATLAB-Funktion heißt `atan2` bzw. `angle`.

   - Berechnen Sie die Phasenbilder der Fouriertransformierte der beiden Bilder.
   - In welchem Wertebereich kommen Werte in den Phasenbildern vor und warum?
   - Warum kommen häufig große Sprünge zwischen benachbarten Pixeln vor?

c) Erzeugen Sie im Fourierraum ein „Bild“, das den Phasenanteil von `arches.png` hat, aber überall den Betrag 1. Erzeugen Sie zudem ein zweites „Bild“, das sich von dem vorigen nur dadurch unterscheidet, dass es als Betrag den Betrag von `theorem.png` hat. Transformieren Sie die Bilder in den Ortsraum zurück und übernehmen Sie die Ergebnisse in die Auswertung.

   Welches Bild ist auf dem zweiten Ergebnis besser erkennbar?

   Was ist also „wichtiger“: Phase oder Betrag?

[2] Das Quadrieren kann weglassen werden, da das Ergebnis mit einer Histogrammspreizung angezeigt wird (am besten mit `imshow(ergebnis, [])`). Warum wird das Quadrieren bei der Anzeige mit automatischer Histogrammspreizung keine Veränderung verursachen? Tipp: Es hat etwas mit dem Logarithmus zu tun.