Übung Nr. 4:

Dem Programm `DIPintro` soll als weitere Funktionalität die Berechnung und Visualisierung des Fourier-Spektrums von Bildern hinzugefügt werden (Abb. 1). Da hierzu die Schnelle Fourier-Transformation (Fast Fourier Transform, FFT) genutzt wird, ist die Berechnung nur auf Bilder anwendbar, deren Seitenlängen (gemessen in Pixeln) Potenzen von 2 sind.[1]

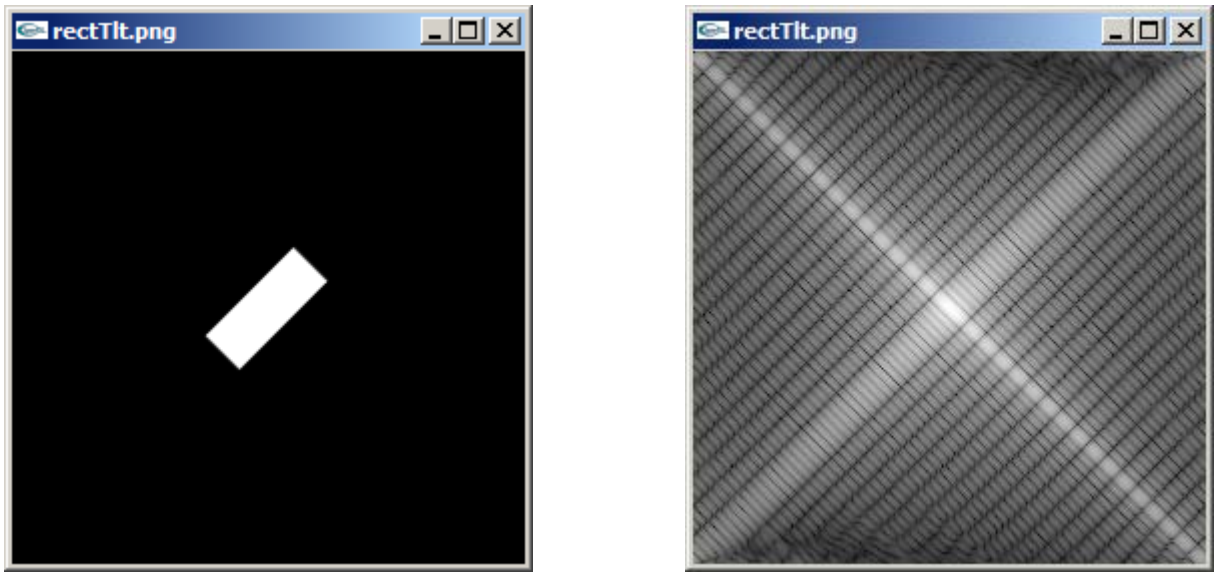

Abb. 1 `DIPintro.exe` Bild und sein Fourier-Spektrum (logarithmische Darstellung)

Das Menü in `DIPintro.c` ist für die beabsichtigte Erweiterung vorbereitet. Alle Aufgaben dieser Übung betreffen Funktionen in der Datei `FFTops.c`. Die zu behandelnden Stellen sind im Code mit den dazugehörigen Präprozessor-Anweisungen und dem Bezeichner `MORE_FFT` gekennzeichnet (s.a. Datei `FFTops.h`).

Da die FFT (hier: die Funktion `fft()`) in ihrem ursprünglichen Entwurf eindimensional ist (z.B. zur Anwendung auf Ton-Signale als Zeitfunktionen), wird sie für die Zeilen und Spalten eines Bildes nacheinander getrennt aufgerufen (s. `fft2D()`). Die darüber hinaus erforderlichen Schritte werden in `Fspec()` organisiert.

1. Teilaufgabe: Berechnung des Betrags der Elemente der Fourier-Transformierten des geladenen Bildes und Speicherung im (danach nicht mehr benötigten) Realteil der Transformierten, dabei Ermittlung des Minimums und des Maximums unter diesen Elementen. (Sie werden später zur Skalierung der Grauwerte gebraucht.)

Nach diesem Code-Zusatz kann für das Muster nach Abb.1 das linear skalierte Spektrum eingesehen werden, s. Abb. 2 (li.). Trotz des hier gewählten günstigen Motivs (senkrecht zueinander stehende Rechteck-Funktionen) ist es jedoch nur schwer erkennbar.

[1] Der Code für den FFT-Algorithmus von Cooley & Tukey ist hier (mit kleinen Änderungen) dem Buch von H. Bässmann, Ph. W. Besslich: „Ad Oculos Digital Image Processing“ entnommen. Die dort angegebene Version ist auf quadratische Bilder beschränkt.

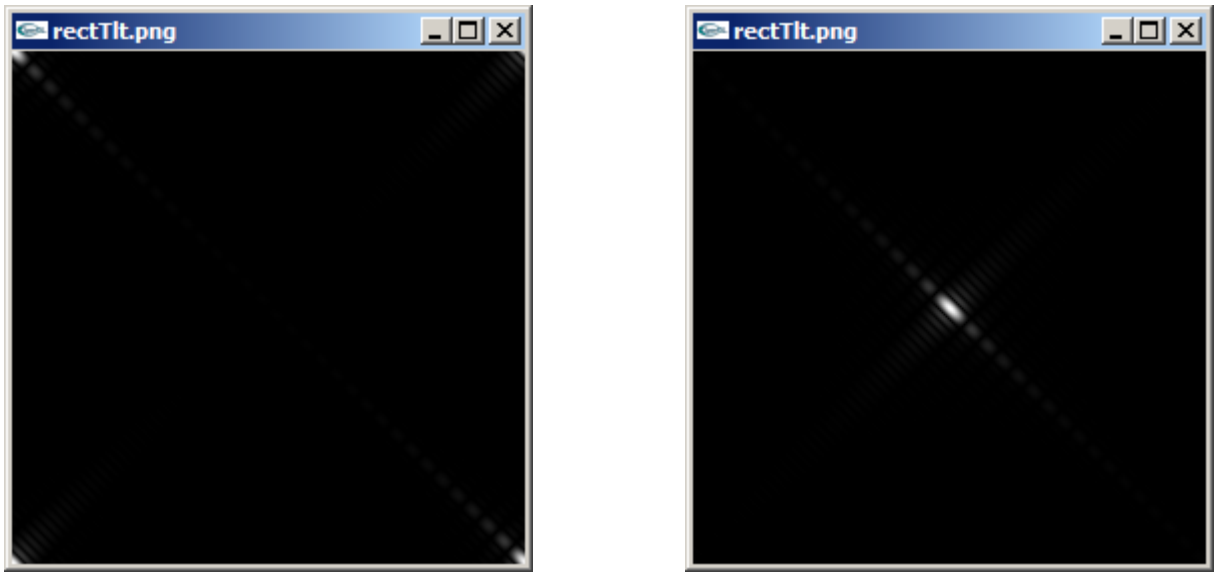

Abb. 2 Linear skaliertes nicht-zentriertes (li.), zentriertes (re.) Fourier-Spektrum zu Abb. 1

In der Praxis wird die zentrierte Form des Spektrums nach Abb. 2 (re.) verwendet. Sie entsteht, wenn das Bild um eine halbe Periode nach rechts unten (bzw. das Achsenkreuz nach links oben) verschoben wird. Das geschieht, wenn das Bild (d.h. die Bildfunktion vor der Transformation, also im Ortsbereich) elementweise mit $(-1)^{x+y}$ multipliziert wird (s. Vorlesung). Diese Umformung ist in der Funktion `centerSpec()` codiert. Sie wird aktiviert, wenn im oberen Teil der Datei `FFTops.c` der Bezeichner `SHIFT4CUT` definiert, in `Fspec()` die Variable `center=1` gesetzt und in `centerSpec()` der Bezeichner `SIMPLE_MINDED` aktiv ist.

Gegenüber dieser direkten Implementierung kann man hier, mit etwas Nachdenken über die Wirkung dieser Maßnahme (evtl. mit Hilfestellung des Autors), `centerSpec()` umcodieren und Rechenzeit einsparen.

2. Teilaufgabe: Verbesserung der Funktion `centerSpec()`

Dieselbe Zentrierwirkung läßt sich erzeugen, wenn erst nach Bildung der nicht-zentrierten Form das Spektrum geviertelt und die Viertel links/rechts bzw. oben/unten vertauscht werden.[2] Für diesen Vorgang (hier: für die Gleitkomma-Darstellung, noch bevor das Pixelbild erzeugt wird) ist die Funktion `Center()` vorbereitet und zur Hälfte codiert worden. Sie tritt in Aktion, wenn der Bezeichner `SHIFT4CUT` nicht definiert ist (und in `Fspec()` die Variable `center=1` weiterhin gesetzt ist).

3. Teilaufgabe: Vervollständigung der Funktion `Center()`

[2] Dabei macht man sich die Tatsache zunutze, daß Bild und Spektrum periodische Funktionen sind; d.h., alle vorkommenden Funktionswerte können einer einzigen Periode entnommen werden.

Zur leichteren Erkennung der doppelten sinc-Funktion, die der Betrag der Transformierten des eingesetzten Motivs aus der Abb. 1 (li.) bildet, eignet sich eine logarithmische Transformation der darin vorkommenden Werte; sie wurde in der Vorlesung vorgestellt. Die in Abb. 1 (re.) wiedergegebene Variante nimmt den (Zehner-)Logarithmus der um eins erhöhten Differenz zwischen Maximum und Minimum des Transformierten-Betrags, nach ähnlichen Erwägungen skaliert wie im linearen Fall.

Beim Versuch, eine passende Visualisierung zu erreichen, stellt man fest, daß bereits kleinere Abweichungen von der ursprünglich avisierten Transformation zu starken visuellen Veränderungen führen. Die dabei entstehenden (manchmal psychedelisch anmutenden) Muster können für eine visuelle Weiterverarbeitung entsprechend hilfreich oder hinderlich sein. Abb. 3 zeigt ein Beispiel (realisiert in der mitgelieferten Demo-Version von `DIPintro.exe`).

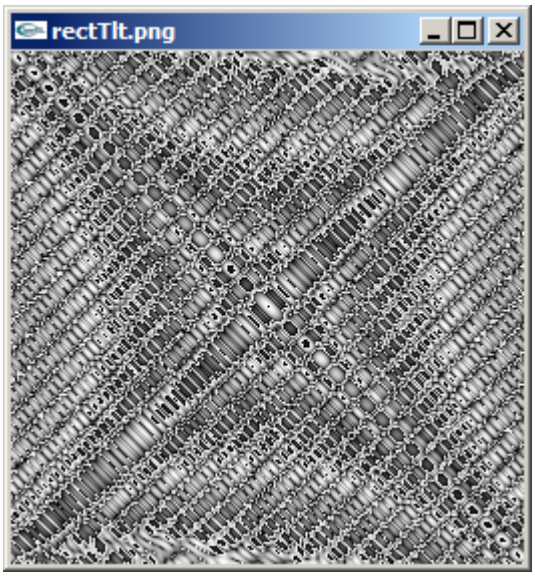

Abb. 3 Zentriertes Fourier-Spektrum zu Abb. 1 mit leicht abweichender (logarithmischer) Skalierung

4. Teilaufgabe: Implementierung der logarithmischen und (freiwillig) einer kreativ-phantasievollen eigenen Skalierung

Mit der Fourier-Transformation läßt sich auch eine schnelle Umsetzung der Faltung mit einer Punkt-Spreiz-Funktion (PSF) berechnen. Die dafür (nur für Grautonbilder) implementierte Funktion `Ffilt()` benötigt dazu PSF-Masken, die in „leere“ (d.h.: mit Nullen besetzte) Matrizen (hier: mittig) gesetzt und auf die Größe des Bildes gebracht („gepaddet“) werden.[3] Das soll in der Funktion `padMat()` passieren, die zwangsläufig den Speicherplatz für die übergebene Matrix erweitert und somit auch ihren Speicherort wechselt. (Deshalb erhält sie über die Parameterliste die Möglichkeit, auch die Speicheradresse der Maske zu verändern.)

[3] Die gemeinsame Bild- und Maskengröße wird in der Praxis manchmal auf das Doppelte der ursprünglichen Bildgröße gesetzt; das wird in `Ffilt()` mit der Einrichtung der beiden Variablen `padW` und `padH` für eine mögliche Weiterentwicklung berücksichtigt.

5. Teilaufgabe: Vervollständigung der Funktion `padMat()`

Nun kann `Ffilt()` um die Berechnung der Faltung ergänzt werden.

6. Teilaufgabe: Teil-Ergänzung von `Ffilt()` (für `purpose == CONV`)

Bei Verwendung einer (in der Datei `filter.flt` bereitgestellten) 16x16-Maske, die eine leicht schräge Verwacklung simuliert, entsteht aus einem der Bilder der angebotenen Galerie das Motiv nach Abb. 4 (li.). Bei korrekter bisher erfolgter Codierung sollte ihr visualisiertes Fourier-Spektrum die typische Streifen-Struktur der Faltung mit einer Rechteck-Funktion aufweisen.

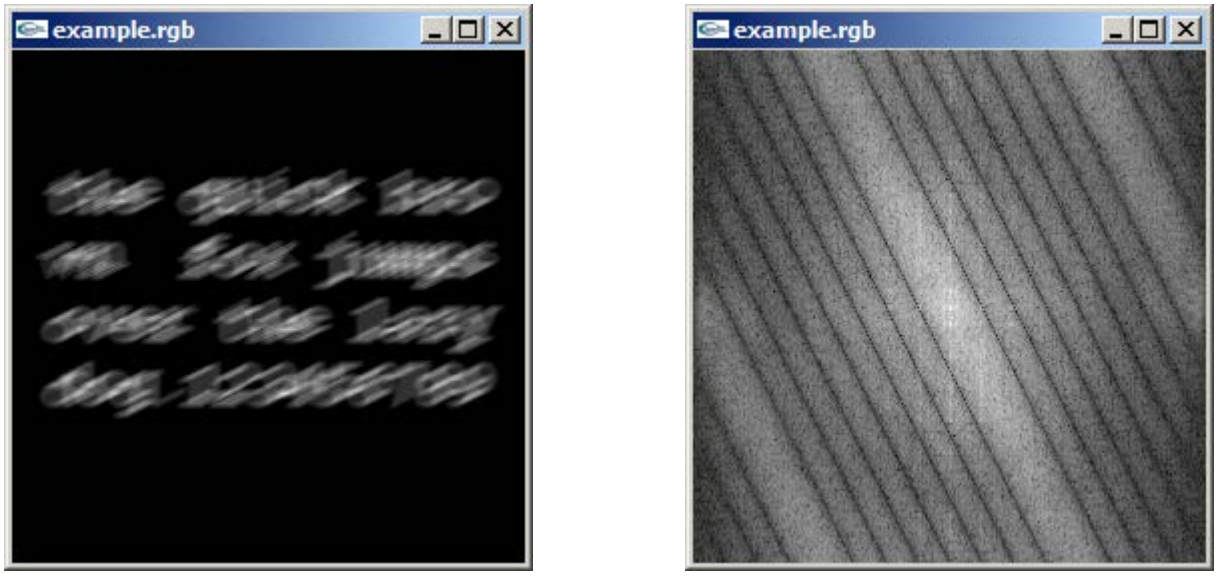

Abb. 4 `DIPintro.exe` Digital erzeugte Verwacklung (li.) und ihr Fourier-Spektrum (re.)

Aufgrund der Kenntnis der verwendeten PSF kann jetzt eine möglichst gute Näherung an das scharfe Original der unscharfen Vorlage errechnet werden (Abb. 5). Dazu ist die Fertigstellung von `Ffilt()` für die Inversfilterung notwendig. Um Divisionen durch null auszuschließen, ist im Code eine Variable eingerichtet und mit einem praxisnahen Wert initialisiert worden (`eps=.00015`). Die Variable kann so verwendet werden oder zu Testzwecken mit anderen Werten initialisiert werden.

Aufgrund der Annahme symmetrischer Signalfunktionen (sich unendlich oft wiederholender „Bild-Kacheln“) und der Symmetrie-Eigenschaft der Fourier-Transformation kann hier wieder wahlweise im Fourier-Bereich oder im Ortsbereich zentriert werden.

7. Teilaufgabe: Ergänzung von `Ffilt()` (für `purpose == INVR`)

Abb. 5 Anwendung der Inversfilterung auf Abb. 4 (li.)

Ein Aspekt, der in dieser Übung bewußt ausgespart wurde, betrifft die Reihenfolge der Verarbeitungsschritte, vor allem in der Funktion `Ffilt()`. Manche Aufrufe sind in ihrer Reihenfolge austauschbar, andere (z.B. betreffend die Zentrierung) stehen in einer logisch-mathematischen Folge. Das Erkennen und Begreifen der Zusammenhänge um Faltung und Filterung sind essentiell für ein tieferes Verständnis dieser Thematik.