Ludwig-Maximilians-Universität München
LFE Medieninformatik
Prof. Dr. Andreas Butz
Dominikus Baur, Sebastian Boring

Sommersemester 2009
Übungsblatt 9
26. Juni 2009

Computergrafik 1

Abgabetermin:
Die Lösung zu diesem Übungsblatt ist bis zum Freitag den **10. Juli 2009, 12:00 Uhr s.t.** (d.h. in **2 Wochen**) per Email abzugeben.

Inhalt:
In diesem Blatt erweitern Sie Ihr bereits erstelltes Fenster um weitere Filterfunktionen. Dieses Mal beschäftigen Sie sich mit der *(Fast) Fourier Transformation* und Ihrer Anwendung in der Bildverarbeitung. Die grundlegenden Konzepte der *Fourier Transformation* haben Sie in der Vorlesung bereits kennen gelernt, hier sollen Sie Ihr Wissen praktisch anwenden und vertiefen. In diesem Blatt können maximal 30 Punkte erreicht werden. Zum Bestehen des Übungsblatts müssen also **MINDESTENS** 15 Punkte erreicht werden.

Geben Sie insgesamt (d.h. ein Projekt für alle Aufgaben zusammen) alle benötigten Header-, Source-, und Projektdateien (von *Qt Creator* erzeugt) mit ab, d.h. `*.h`, `*.cpp` und `*.pro` Dateien. Abgaben die nicht kompilieren werden mit 0 Punkten bewertet. Fassen Sie alle Aufgaben zu einer zip-Datei zusammen und senden Sie diese an: `cg1_ss09@medien.ifi.lmu.de`.

Aufgabe 29 (Fast) Fourier Transformation (30 Punkte)

In dieser Aufgabe werden Sie die *(Fast) Fourier Transformation* schrittweise implementieren. Dabei werden Sie Ihr Frame dieses Mal um nur zwei dialogfreie Filter erweitern.

Allgemeine Fourier Transformation: Jedes beliebige Signal, z.B. Töne im 1-dimensionalen bzw. Bilder im 2-dimensionalen Raum, besteht aus einer Kombination von Signalen unterschiedlicher Frequenz. Anhand eines Prismas können Sie sich sehr gut veranschaulichen, wie Signale aus unterschiedlichen Signalen zusammen gesetzt sind. Wenn weißes Licht auf das Prisma trifft, wird das Signal in seine konstituierenden Signale (farbiges Licht) zerlegt. Analog lässt sich jede periodische Funktion durch eine gewichtete Summe von Sinusoiden (Sinus und Cosinus Wellen) ausdrücken.

Diese Erkenntniss ist die Grundlage der Fouriertheorie. Die einzelnen Funktionen der Summe heißen Basisfunktionen. Die Fouriertheorie ist ein Werkzeug, um die Anteile jeder einzelnen Basisfunktion in der Repräsentation einer Funktion $f(x)$ zu analysieren. Für Bilder gilt selbstverständlich, dass die Originalfunktion zweidimensional ist.

Die *Fourier Transformation* in zwei Dimensionen lässt sich folgendermaßen angeben:

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(m,n)\cdot e^{-i\cdot 2\pi(um+vn)}\,dmdn \tag{1}$$

Analog ist die *Inverse Fourier Transformation* definiert als:

$$f(m,n) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)\cdot e^{i\cdot 2\pi(um+vn)}\,dudv \tag{2}$$

Dabei bezeichnet $F(u,v)$ die Funktion im *Frequenzraum*, oder die Fourier-Transformierte. $f(m,n)$ bezeichnet dann die Originalfunktion im *Ortsraum*. Beachten Sie dabei, dass $F(u,v)$ eine komplexe Zahl ist. Im Frequenzraum bezeichnet u die Frequenz entlang der originalen m-Achse und v

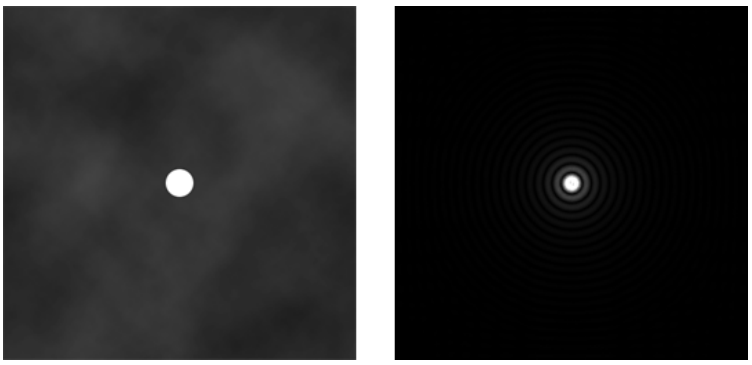

Abbildung 1: *Fourier Transformation* eines Punktes mit leichten Störungen im Hintergrund. Links der *Ortsraum*, rechts der zugehörige *Frequenzraum*

die Frequenz entlang der originalen *n*-Achse. Wie in Abbildung 1 zu sehen, ist der Ursprung des *Frequenzraumes* in der Mitte des Bildes, nicht wie im *Ortsraum* links oben.

Bei Bildern hat man nie eine kontinuierlich definierte Funktion, sondern diskrete Samples – die Pixel des Bildes. Um mit solchen Daten trotzdem im Frequenzraum arbeiten zu können, benötigen Sie einen Spezialfall der *Diskreten Fourier Transformation*. Anstatt der Integration für kontinuierliche Funktionen, verwenden Sie hier Summen über die diskreten Samples.

Die Formel um eine Bild der Dimension $M \times N$ in den *Frequenzraum* zu transformieren ist gegeben durch:

$$F(u,v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) \cdot e^{-i \cdot 2\pi \left(\frac{um}{M} + \frac{vn}{N}\right)} \tag{3}$$

Die Formel um zurück in den *Ortsraum* zu gelangen ist dann:

$$f(m,n) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} F(u,v) \cdot e^{i \cdot 2\pi \left(\frac{um}{M} + \frac{vn}{N}\right)} \tag{4}$$

Der einzige Unterschied zwischen den beiden Formeln ist das Vorzeichen im Exponenten. Daraus lassen sich zwei Dinge ablesen. 1. Die *Fourier Transformation* ist in beide Richtungen verlustfrei, also eine eindeutige Abbildung. 2. Der selbe Code lässt sich verwenden, um die *Fourier Transformation* und die *Inverse Fourier Transformation* zu implementieren.

Um nun nachvollziehen zu können, was die Fourier-Transformierte eines Bildes ausdrückt, muss man sich verdeutlichen, das $F(u,v)$ eine komplexe Zahl ist.

$$F(u,v) = R(u,v) + i \cdot I(u,v) = |F(u,v)| \cdot e^{i \cdot \phi(u,v)} \tag{5}$$

Der reelle $R(u,v)$ und imaginäre $I(u,v)$ Anteil haben noch keine besondere Bedeutung, allerdings ist $|F(u,v)|$ die Amplitude und $\phi(u,v)$ die dazugehörige Phase. Zerlegt man nun ein Bild im *Frequenzraum* in ein Array von Amplituden und ein Array von Phasen, dann erhalten wir das Amplituden-Spektrum, respektive das Phasen-Spektrum eines Bildes. Diese Spektren lassen sich wiederum als Bilder (Abbildung 2) zeichnen.

Das Amplituden-Spektrum gibt die relativen Helligkeiten von Objekten im *Ortsraum* wieder. Das Phasen-Spektrum encodiert die Information über die Position dieser Objekte. Die meisten Operationen im *Frequenzraum* lassen deswegen das Phasen-Spektrum unverändert.

Machen Sie sich mit den Konzepten der *Fourier Transformation* und insbesondere mit der *diskreten* Variante vertraut. Benutzten Sie dazu z.B. die Slides der Vorlesung oder das Buch *Grundlagen*

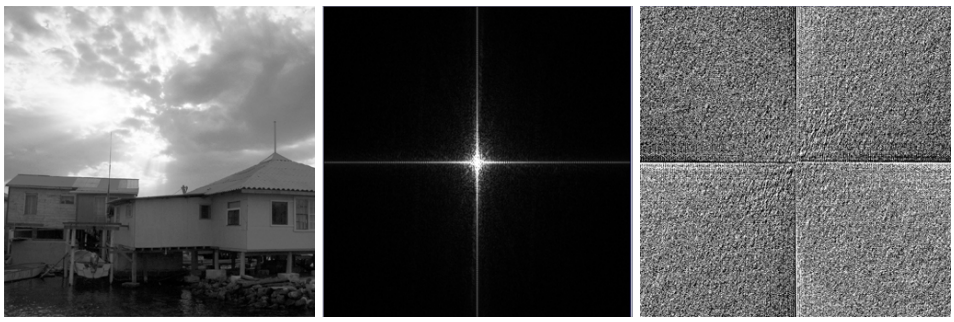

Abbildung 2: Ein Bild im *Ortsraum*, sein Amplituden-Spektrum sowie das Phasen-Spektrum.

der Bildverarbeitung[1].

Fast Fourier Transformation: Um einen Wert von $F(u,v)$ nach obiger Formel zu berechnen, muss einmal über alle Pixel summiert werden. Bei einem $N \times N$ Bild ergibt das eine Komplexität von $O(N^2)$ und folglich für alle Werte eine Gesamtkomplexität von $O(N^4)$. Das führt schon bei sehr kleinen Bildern zu nicht akzeptablen Laufzeiten. Die *Fourier Transformation* hat allerdings zwei Eigenchaften, durch die die Laufzeit deutlich verkürzt werden kann.

Separabilität Die 2-dimensionale *Fourier Transformation* lässt sich in zwei 1-dimensionale *Fourier Transformationen* zerlegen. Zuerst wird eine 1-dimensionale *Fourier Transformation* entlang den Zeilen eines Bildes ausgeführt, um ein temporäres Ergebniss zu berechnen. Danach wird eine zweite *Fourier Transformation* entlang den Spalten dieses Ergebnisses durchgeführt, um das entgültige Ergebniss zu erlangen.

Symmetrie Die 1-dimensionale *Fourier Transformation* der Länge N kann als Summe zweier *Fourier Transformationen* der Länge $\frac{N}{2}$ ausgedrückt werden. Ist nun N eine 2er-Potenz, dann kann man diesen Schritt rekursiv anwenden, bis Transformationen der Länge 2 betrachtet werden.

Die *Fast Fourier Transformation* (FFT) ist ein sogenanntes *Divide-and-Conquer* Verfahren, der obige Eigenschaften ausnutzt und damit die Komplexität auf $O(N^2 \times \log N)$ senkt. Die *Fast Fourier Transformation* für Bilder erfordert, dass beide Bild-Dimensionen 2er-Potenzen sind. Ist das nicht der Fall, muss das Bild entweder verkleinert oder vergrößert werden.

Die Fouriertheorie geht davon aus, dass die betrachteten Funktionen periodisch sind, d.h. sich endlos wiederholen. Diese Annahme gilt für Bilder in der Regel allerdings nicht. Kachelt man ein Bild mehrfach in unterschiedliche Richtungen, so sind an den Kanten starke Diskontinuitäten zu beobachten. In der *Fourier Transformation* werden die Eingabedaten aber als eine Periode einer sich unendlich wiederholenden Frequenz betrachtet. Die unpassenden Anschlussstellen an den Rändern des Bildes wirken sich wie abrupte Änderungen im Bild aus wodurch der *Frequenzraum* verzerrt wird.

Um diese unerwünschte Verzerrung im *Frequenzraum* zu verhindern, kann man die Pixelwerte des Bildes so modifizieren, dass sie sich zum Bildrand hin Null annähern. Dies geschieht, indem man eine so genannte *Fenster-Funktion (engl. Windowing)* zu den Daten multipliziert bevor man die *Fourier Transformation* durchführt. Es gibt mehrere Standard-Fenster, z.B. das Bartlett-Fenster, welches wie folgt definiert ist (r ist Abstand zum Bildmittelpunkt):

[1] Klaus D. Tönnies, ISBN 3-8273-7155-4

$$w(r) = \begin{cases} 1 - \left(\frac{r}{r_{max}}\right) & , r \leq r_{max} \\ 0 & , r > r_{max} \end{cases} \tag{6}$$

Das Hannning-Fenster ist folgendermaßen für $r \leq r_{max}$ definiert:

$$w(r) = \frac{1}{2} - \frac{1}{2} cos\left[\pi \times \left(1 - \frac{r}{r_{max}}\right)\right] \tag{7}$$

a) Machen sie sich mit dem *FFT* Algorithmus und seiner Implementierung vertraut. Gute Ausgangspunkte sind der Wikipedia Eintrag zum originalen *FFT*[2] sowie *dspGuru*[3] und *FFTW*[4].

b) Fügen Sie die Menüpunkte *Fast Fourier Transformation (Amplitude)* und *Fast Fourier Transformation (Phase)* in Ihr bereits erstelltes Untermenü *Transformation* ein (**2 Punkte**).

c) Erstellen Sie nun eine Klasse *FourierTransform*, die von *ImageFilter* erbt. Da dieser Filter ohne Dialog auskommt, können Sie hier 0 zurückgeben. Die Rückgabe des Bildes durch die Funktion `apply` ist dann entweder das Bild im *Amplituden-Spektrum* oder im *Phasen-Spektrum*. Dies ist abhängig von der gewählten *FFT* (**6 Punkte**).

d) Erzeugen Sie zusätzlich eine Klasse namens *Complex*, die Ihnen den Umgang mit komplexen Zahlen vereinfachen soll. Diese Klasse besitzt zwei Variablen (vom Typ `double`), die den Real- bzw. Imaginärteil repräsentieren. Implementieren Sie zudem Methoden, die den Winkel α bzw. die Länge zurückgeben (**6 Punkte**).

e) Implementieren Sie nun die *FFT* in der Funktion `apply`. Verwenden Sie eine der beiden hier vorgestellten Fensterfunktionen (also entweder *Bartlett* oder *Hanning*). Führen Sie die notwendigen Schritte durch falls das Bild nicht den Vorgaben entspricht, d.h. mindestens eine der Dimensionen ist keine 2er-Potenz (**Hinweis:** Skalieren oder Ausschneiden) oder das Bild ist nicht als Graustufen-Bild gespeichert (**Hinweis:** Umwandlung in 8-bit Bild). Verwenden Sie zudem Ihre zuvor erstellte Klasse *Complex* (**16 Punkte**).

[2] http://en.wikipedia.org/wiki/Cooley-Tukey_FFT_algorithm
[3] http://www.dspguru.com/info/faqs/fftfaq2.htm
[4] http://www.fftw.org/links.html