Hinweise zur Bildverarbeitung für das Argonexperiment

Inhaltsverzeichnis

Warum so kompliziert?

Bei der Bestimmung des Drehwinkels gab es bestimmte Winkel, bei denen ein Sprung in einigen Funktionen auftrat (z.B. in der Standardabweichung über dem Winkel), was auf Probleme beim Drehen hindeutet. Bei der normalen Drehung muß ja eine Interpolation in 2 Dimensionen durchgeführt werden die gerade bei verrauschten Daten ungenau ist. Mit Hilfe der Radontransformation will man diese Ungenauigkeiten vermeiden.

1 die ImProcAr1Sec - Klasse und Grundlagen

1.1 Radontransformation

Die Radontransformation ist grob gesprochen ein Linienintegral durch das Bild in einer beliebigen Richtung.

Wir wollen zur Auswertung einen Schnitt einer Section haben, wozu bisher das Bild gedreht und geschert, anschließend ein Teil ausgeschnitten (die gewünschte Section) und zum Schluß darüber summiert wurde.

Die Idee der neuen Bildverarbeitung ist, nicht mehr das Bild zu drehen, sondern die Summation durch ein "Linienintegral" in der gewünschten Richtung zu ersetzen und damit das Orginalbild gar nicht mehr drehen zu müssen.

Die Radontransformation bietet genau dieses Werkzeug, jedoch wird bei dieser immer über das gesamte Bild integriert/summiert, wir wollen jedoch nur die Summe über eine einzelne Section haben. Um dennoch die Radontransformation benutzen zu können, legt man vorher über das Bild eine Maske, welche eine 1 an allen Stellen der Maske hat, und ansonsten 0 ist. Wendet man nun die Radontransformation auf das maskierte Bild an, so erhält man das gewünschte Linienintegral nur über diese Section.

Da wir noch eine Scherung benötigen, muß man die Maske in dem ungedrehten Koordinatensystem (eine Achse ist senkrecht zum Beugungsbild) scheren, was aber analytisch geht und kein Problem ist.

1.2 genauere Betrachtung der Radontransformation

Die Radontransformation wird üblicherweise über eine 2-dimensionale Fouriertransformation berechnet, was erstens einen Geschwindigkeitsvorteil mit sich bringt, andererseits aber dafür sorgt, daß man keinerlei Interpolationen benötigt.

Sieht man sich aber die 2-dimensionale Fouriertransformation des maskierten Bildes an, so muß man aufgrund des Faltungstheorems leider feststellen, daß es zu einer überbetonung einiger hoher Frequenzen kommen kann.
Faltungstheorem:

$$F[f(x) \cdot g(x)] = F[f(x)] \otimes F[g(x)]$$

Würde man für die Maske in y''-Richtung nun einfach eine Rechteckfunktion benutzen, so bedeutet das im Fourierraum eine Faltung mit der sinc-Funktion (der Fouriertranformierten der Rechteckfunktion), was eine überbetonung hoher Frequenzen nach sich ziehen würde. Hohe Frequenzen kommen in dieser Richtung aber ausschließlich durch Rauschen zustande und sind eher unerwünscht. Durch die anschließende Integration in dieser Richtung wird der Einfluß des Rauschens zwar wieder vermindert, es wäre jedoch deutlich besser, wenn die Überbetonung hoher Frequenzen gar nicht erst auftreten würde.

1.3 die Wahl des Fensters - `WindowFunc`

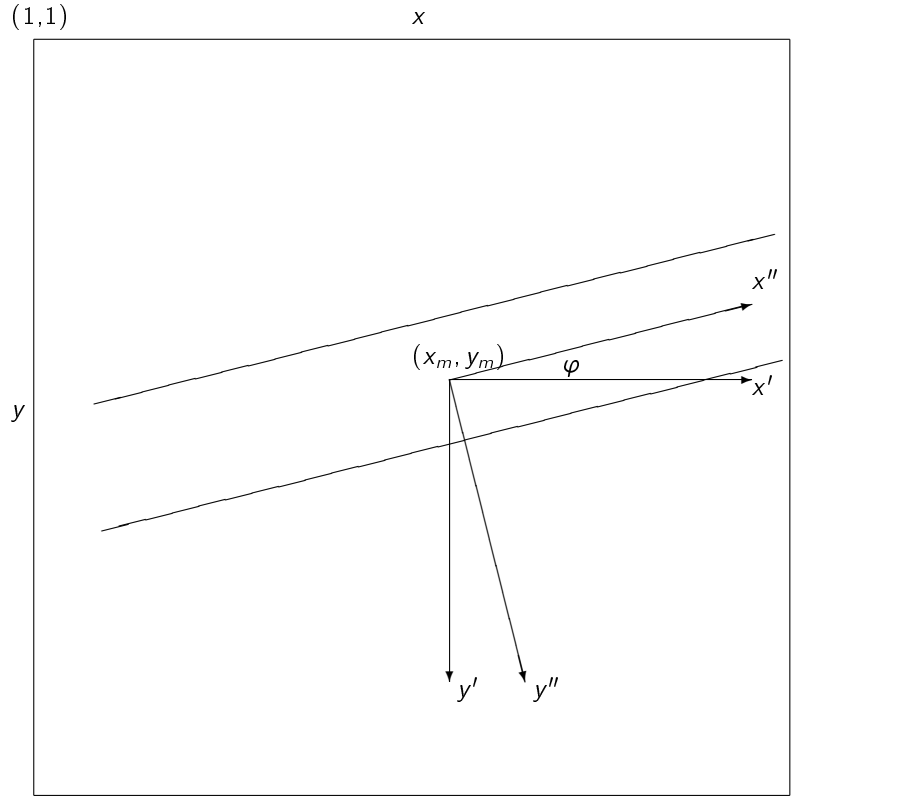
Zu diesem Zweck muß die Maske in y''-Richtung einem Fenster entsprechen, dessen Fouriertransformierte hohe Frequenzen nicht überbetont sondern gleichmäßig dämpft. Typischerweise werden für solche Zwecke z.B. Hamming oder Hann-Fenster benutzt, diese haben für unseren Fall jedoch alle gemeinsam den Nachteil, daß die Flanken nur langsam ansteigen und das Maximum dafür nur auf einem sehr kleinen Teil angenommen wird. Für uns würde das heißen, daß man sich entweder nur auf den Bereich einer Section beschränkt und dafür viel weniger Fläche unter dem Fenster hat (das heißt viel weniger Statistik) oder daß man die Fläche in etwa beibehält, dafür aber mit dem Fenster weit in andere Sections hineinragt und diese ebenso (wenn auch mit kleineren Anteilen) zu dem Ergebnis dieser Section beitragen.

Um diesem Dilemma zu entgehen, wurde ein eigenes Fenster zusammengebastelt, welches einerseits möglichst viel aus der Mitte der Section mitnimmt, aber andererseits hohe Frequenzen dennoch halbwegs passabel dämpft. Die Wahl viel auf ein Fenster, welches in der Mitte ein Rechteckfenster ist, an dessen Flanken aber jeweils ein cos-förmiges Fenster anschließt.

Für eine 25 Pixel breite Section wäre eine gängige Wahl ein Fenster mit einer Breite von 19 Pixel (`BreiteFenster` ≡ Breite des Rechteckfensters) und der Breite des Anstiegs von 5 Pixel (`BreiteAnstieg` ≡ Breite des cos-Anstiegs). Auf diese Weise ragt man nur etwa 2,5 Pixel in die benachbarten Sections hinein, und diese werden nur stark gedämpft mitgenommen.

Die Fensterfunktion findet man unter `@ImProcAr1Sec\private\WindowFunc.m` , falls man eine bessere Idee für ein Fenster hat, so kann man dieses dort ändern. Die Funktion gibt für einen eingegebenen y''-Wert den Funktionswert des Fensters an dieser Stelle zurück. Das Fenster muß symmetrisch um 0 sein, darf nur vom y-Wert abhängen (keine x-Abhängigkeit, die Drehung und Scherung werden woanders gemacht), und muß Matrizen und zwei skalare Werte (Breite des Fensters und Breite des Anstiegs) als Eingabe akzeptieren (ob beide skalaren Werte gebraucht werden, bleibt jedem selbst überlassen).

1.4 Berechnung der Maske - CreateMask



Die Fensterfunktion ist nur von y'' abhängig und wird mit F bezeichnet, die Maske mit M.

Die Scherung berechnet sich in $''$-Koordinaten zu:

$$\begin{aligned} & y'' &=& sx'' \\ \Rightarrow \qquad & M(x'', y'') &=& F(y'' - sx'') \end{aligned}$$

Für die Drehung braucht man eine passive Drehung (da man an den $'$-gestrichenen Koordinaten den Wert der $''$-gestrichenen braucht).

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \begin{pmatrix} x'' \\ y'' \end{pmatrix}$$

Die Maske in $'$-gestrichenen Koordinaten ist:

$$M(x', y') = F\left[x' \sin\varphi + y' \cos\varphi - s(x' \cos\varphi - y' \sin\varphi)\right]$$

am Ende kommt nur noch die Verschiebung (und da die obere Ecke in Matlab sowieso die $(1, 1)$ ist, braucht man nicht noch zusätzlich wie sonst üblich die Koordinaten um 1 zu verschieben).

$$\begin{aligned} M(x,y) &= F\,[(x-x_m)\sin\varphi + (y-y_m)\cos\varphi \\ &\quad -s((x-x_m)\cos\varphi - (y-y_m)\sin\varphi)] \end{aligned}$$

Die Koordinatentransformation ist in `@ImProcAr1Sec\CreateMask.m` implementiert. Es wird ein `ImProcAr1Sec`-Objekt übergeben, aus diesem die benötigten Parameter extrahiert und die berechnete Maske zurückgegeben.

1.5 Berechnung der Radontranformierten - `GetRadon`

Über die Funktion `GetRadon` kann man den Querschnitt für eine Section eines Bildes berechnen. Dazu wird das `ImProcAr1Sec`-Objekt und das Bild übergeben, dann die Maske geladen und auf das Bild angewandt. Anschließend wird die Radontransformation des maskierten Bildes berechnet und zurückgegeben.

Falls die Größe des Bildes nicht mit der Bildgröße in c übereinstimmt, so wird die Maske in neuer Größe berechnet und auf das Bild angewandt. Da jedoch nicht mehr die Richtigkeit der Mitte sichergestellt werden kann, wird zusätzlich eine Warnung ausgegeben.

1.6 Funktionen der `ImProcAr1Sec`-Klasse

In folgender Tabelle sind alle Funktionen für die `ImProcAr1Sec`-Klasse zusammengefasst:

`c = ImProcAr1Sec(param)` ist der Klassen-Constructor und erzeugt ein Objekt dieser Klasse. `param` kann leer sein (dann wird ein Standardobjekt erzeugt), oder kann aus einer Aufzählung aller benötigten Parameter bestehen. Die möglichen Parameter sind in Tabelle 1 zusammengefasst.

`c = set(c,param)` Ändert die übergebenen Parameter in dem Objekt c. Die möglichen Parameter können als Liste übergeben werden und sind wieder in Tabelle 1 zusammengefasst.

`val = get(c,descript)` gibt den Wert der Eigenschaft `descript` von c zurück. Dabei ist jeweils nur eine Eigenschaft erlaubt.

`display(c)` gibt alle Parameter aus c formatiert in dem Command-Window aus. Die Kurzform `c Enter` ist ebenfalls möglich.

`plot(c,b,nrfigure)` plottet das maskierte Bild in das Fester Nr. `nrfigure`, wobei auch der unmaskierte Teil noch sichtbar gelassen wird, um die Position besser sichtbar zu machen. Wird das Bild weggelassen, so wird nur die Maske gezeichnet. Ebenso kann `nrfigure` weggelassen werden, wobei dann ein neues Figure geöffnet wird.

`[f r] = GetRadon(c,b)` berechnet den Querschnitt durch die Section wie schon im vorhergehenden Abschnitt beschrieben. Der Querschnitt ist dann in f enthalten, in r wird der Abstand von der natürlichen Mitte des Bildes zurückgegeben. Die Abfrage von r kann auch weggelassen werden.

`[f r fn] = GetRadon(c,b)` wie schon eben beschrieben wird die Radontranformierte berechnet, jedoch wird jetzt zusätzlich noch eine auf diese Section normierte Radontransformation an `fn` zurückgegeben. (Die Normierung erfolgt so, daß $\sum fn = 1$ ist)

Parameter	Standardwert	Beschreibung
'Winkel'	0	Winkel in Grad gegen den UZS gegenüber der Horizontalen
'Scherung'	0	Scherparameter
'Mitte'	[128 128]	Position der Mitte in Pixel es sind auch nichtganzzahlige Werte möglich
'BreiteFenster'	19	Breite des Rechteckfensters (in Pixel, möglichst ungerade)
'BreiteAnstieg'	5	Breite des Cos-Anstiegs (in Pixel, jeweils für beide Seiten)
'BildGroesse'	[256 256]	Größe des Bildes (in Pixel)
'HBreiteAusschnitt'	80	halbe Breite des zurückgegebenen Ausschnittes
'MitteAusschnitt'	5	Abstand der Mitte des Ausschnittes von der Bildmitte in Pixel

Tabelle 1: Liste aller möglichen Parameter

Da die zurückgegebene Radontransformierte über die gesamte Bildbreite (oder gar Diagonale) reicht, werden die letzten beiden Parameter benötigt, um den zurückgegebenen Ausschnitt zurechtzuschneiden. Die Mitte des Atomstrahls liegt nicht unbedingt in der Mitte des Bildes, diese Relation kommt in der Größe 'MitteAusschnitt' zum Ausdruck (diesen Wert kann man sehr einfach finden, indem man die Radontransformierte für ein Spaltbild berechnet und dann diese über dem Rückgabevektor `r` plottet – der x-Wert für das Maximum ist der benötigte Wert). Die Section wird dann jeweils 'HBreiteAusschnitt' Pixel links und rechts von der Mitte beschnitten. Die Normierung bezieht sich übrigens auf die beschnittene Section, eine zu kleine Wahl von 'HBreiteAusschnitt' birgt einen Fehler für die Normierung.

2 die `ImProcAr` - Klasse

Die `ImProcAr` - Klasse verwaltet alle Daten, die man für die Bildverarbeitung eines Bildes benötigt, auch wenn mehrere Sections verarbeitet werden müssen.

Sie stellt neben den Standardmethoden (Klassen-Constructor und set-Funktion) eine bequeme Methode zur Verfügung, alle zur Bildverarbeitung benötigten Daten aus dem MainIndex auszulesen.

Zusätzlich zur ImProcAr1Sec-Klasse wird das Einlesen eines Referenzbildes unterstützt. Bei der Verarbeitung eines Bildes wird dieses dann vor der Radontransformation durch das Referenzbild geteilt.

2.1 neue oder geänderte Funktionen gegenüber der ImProcAr1Sec-Klasse

Im folgenden sollen neue oder gegenüber der ImProcAr1Sec-Klasse geänderte Funktionen aufgelistet werden:

`c = ImProcAr(param)` ist der Klassen-Constructor und erzeugt ein Objekt dieser Klasse. `param` kann leer sein (dann wird ein Standardobjekt erzeugt), oder kann aus einer Aufzählung aller benötigten Parameter bestehen. Die möglichen Parameter sind in Tabelle 1 zusammengefasst.

`c = set(c,param)` Für die set-Funktion gibt es einige Einschränkungen bezüglich der Parameter: die `AnzahlSections` kann nach der Erzeugung nicht mehr geändert werden

`val = get(c,descript)` gibt den Wert der Eigenschaft `descript` von c zurück. Dabei ist jeweils nur eine Eigenschaft erlaubt.

`display(c)` gibt alle Parameter aus c formatiert in dem Command-Window aus. Die Kurzform `c Enter` ist ebenfalls möglich.

`plot(c,b,nrfigure)` plottet das maskierte Bild in das Fester Nr. `nrfigure`, wobei auch der unmaskierte Teil noch sichtbar gelassen wird, um die Position besser sichtbar zu machen. Wird das Bild weggelassen, so wird nur die Maske gezeichnet. Ebenso kann `nrfigure` weggelassen werden, wobei dann ein neues Figure geöffnet wird.

`GetRadon` diese Funktion ist für die `ImProcAr`-Klasse nicht direkt anwendbar

`[f r fn]=BildVerarbeiten(c,b)` Diese Funktion tritt an die Stelle von `GetRadon` und übernimmt die komplette Bildverarbeitung. Falls ein Referenzbild angegeben wurde, so wird das Bild durch dieses dividiert, und anschließend für jede Section die Radontransformierte berechnet. Anschließend werden alle Sections beschnitten und normiert (das Beschneiden erfolgt wie im vorangegangenen Kapitel für die ImProcAr1Sec-Klasse beschriebenen). Falls man keine normierten Daten benötigt, so kann man diesen Ausgabeparameter auch weglassen.

2.2 neue oder geänderte Übergabeparameter gegenüber der ImProcAr1Sec-Klasse

Da die ImProcAr-Klasse mehrere Sections unterstützt, kommen zusätzlich zu der ImProcAr1Sec-Klasse einige Parameter hinzu. Für alle Sections werden dieselben Winkel, Breite und Anstieg benutzt, nur die Mitten unterscheiden sich.

Man hat grundsätzlich 2 verschiedene Möglichkeiten die Mitten anzugeben: entweder man gibt die Position für jede Mitte getrennt an (also z.B. `Mitte1=[7 7] Mitte2=[7 7] ... MitteN=[... ...]`), oder man gibt nur die Position einer Mitte und zusätzlich den Abstand der Sections (genauer den Abstand der Mitten) an und die fehlenden Mitten werden selbstständig aus dem Winkel berechnet.

Alternativ kann man auch alle Parameter in den MainIndex schreiben und erzeugt das Objekt dann mit `...=ImProcAr('MainIndex')` , dabei wird automatisch der MainIndex aus dem aktuellen Ordner nach den Parametern durchsucht. Alle Bildverarbeitungsparameter müssen mit einem `&` beginnen, es dürfen hinter den Werten keine zusätzlichen Zeichen erscheinen (wie ; oder %). Wenn möglich, sollte man den Parameter-Block mit 3 Fragezeichen abschließen, dies beschleunigt die Bildverarbeitung, da alle Zeilen danach nicht mehr eingelesen werden.

Ein Auszug aus einem MainIndex könnte folgendermaßen aussehen:

```
& Winkel=17.38
& Scherung=-0.315
& BildGroesse=[256 256]
& ReferenzBild=referenz_070522_100sec_norm.txt
& BreiteFenster=15
& BreiteAnstieg=5
& MitteAusschnitt=5
& HBreiteAusschnitt=80
& AnzahlSections=7
& AbstandSections=20
& Mitte4=[129.8 133.5]
???
```

Parameter	Standardwert	Beschreibung
'Winkel'	0	Winkel in Grad gegen den UZS gegenüber der Horizontalen
'Scherung'	0	Scherparameter
'BreiteFenster'	19	Breite des Rechteckfensters (in Pixel, möglichst ungerade)
'BreiteAnstieg'	5	Breite des Cos-Anstiegs (in Pixel, jeweils für beide Seiten)
'HBreiteAusschnitt'	80	halbe Breite des zurückgegebenen Ausschnittes
'MitteAusschnitt'	5	Abstand der Mitte des Ausschnittes von der Bildmitte in Pixel
'BildGroesse'	[256 256]	Größe des Bildes (in Pixel)
'ReferenzBild'	'none'	Pfad zum Referenzbild oder 'none'
zusätzliche Parameter		
'AnzahlSections'	4	Anzahl der Sections (kann später nicht mehr verändert werden)
'AbstandSections'	0	Abstand der Mitten der Sections sollte man nicht angeben, wenn alle Mitten übergeben werden
'MitteN'	[128 128]	die Mitte für Section N
'MainIndex'		kein zusätzlicher Wert zu übergeben

2.3 bekannte Probleme, bzw. noch nicht behobene

In diesem Abschnitt sollen bekannte Probleme und ihre Abhilfe aufgelistet werden.

- bei Angabe aller Parameter in der MainIndex-Datei sollte man keine Semikolon hinter den Zahlenwerten setzen, sonst werden diese nicht erkannt.