Anhang A: XRTMA-Handbuch

XRTMA V 1.1

Handbuch

0. Inhalt

1. Einführung

Das XRTMA-Paket dient der Nachbearbeitung und Auswertung von Bildern, die mit Rastertunnel-/Rasterkraft-Systemen des IFW Dresden e.V. gewonnen wurden. Es kann darüber hinaus Daten einlesen und weiterbearbeiten, die im NanoScope III®- und im DFM-Format vorliegen. Die (nachbearbeiteten) Daten können in eine IFW-, NanoScope-III-, Windows-Bitmap und PostScript®-Dateien exportiert werden und stehen damit für die Einbindung in Dokumente zur Verfügung.
Es wird eine große Anzahl von Filtern bereitgestellt, die es erlauben, die Qualität der Rohdaten so zu verbessern, daß eine Auswertung möglich wird. Die Elimination von Punkt- und Zeilenstörungen und Ausgleichskorrekturen sind ebenfalls möglich.
Im Programm sind Meßfunktionen integriert, die zusätzlich zur Auswertung von Topographiedaten, auch Potentiometrie- und Spektroskopiedaten verarbeiten können.
Die Auswertung kann sowohl in der bequem zu bedienenden graphischen Umgebung (*xrtma*), als auch in einer C++ Programmierumgebung (*librtma*) erfolgen. Für die graphische Umgebung existiert ein kontextsensitives Hilfesystem.
Bemerkung: Einige Funktionen werden aufgrund ihrer verschiedenen Aufruf-Semantik mehrfach erwähnt werden. Ihre operationelle Semantik ist aber in allen Fällen dieselbe.

Folgende Piktogramme werden verwendet:

 Abschnitt nur für Insider (der C- und UNIX-Welt).
 Der Befehl kann so oder in ähnlicher Form auch per Tastatur eingegeben werden.

2. Installation

Das System ist zur Zeit unter Linux 0.99/XFree86 3.1.1 sowie unter IBM/AIX® verfügbar. Die Programmierschnittstelle sollte auch unter MS/DOS® lauffähig sein, führt dort aber höchstwahrscheinlich bei größeren Datenmengen zu Konflikten mit der 640 kByte Begrenzung und der 64 kByte Segmentgrenze. Zur Portierung auf andere Plattformen stehen die Quelltexte beim IFW Dresden e.V. und an der Abteilung Mathematik der Technischen Universität Dresden, Institut für Wissenschaftliches Rechnen im Rahmen der für Diplomarbeiten geltenden Lizenzbedingungen zur Verfügung.

2.1. Systemvoraussetzungen für PC-Linux Version

- Intel 80386 oder höher, bei 80386 mathematischer Koprozessor empfohlen
- 4 MByte Hauptspeicher, 16 MByte oder mehr empfohlen
- 3 Tasten-Maus
- 4 MByte freier Plattenplatz
- Linux Kernel ≥ 0.99 patch level 15g, libc.4.5.26, ld.so.1.4.3 oder höher
- XFree86 3.1.1 (X11/R6) oder höher
- gcc 2.5.8 GNU C/C++ Compiler
- zusätzlich Metrolink OSF/Motif 2.0 oder höher für die dynamic linked Version

2.2. Installation

Die Standardinstallation erfordert Super-User Rechte, da das Paket sinnvollerweise nach /usr/local installiert wird. Auf der Diskette befindet sich dafür das Installations-Script *rtmainst*. Die Standardinstallation wird in vier Schritten abgearbeitet:

1. Als Super-User einloggen.
2. Die Datei *rtmainst* auf die Festplatte kopieren. Unter Linux geschieht dies mit dem Kommando *mread rtmainst rtmainst*.
3. *rtmainst* ausführbar machen mit *chmod u+x rtmainst*.
4. Starten von *rtmainst*.

rtmainst legt das Verzeichnis */usr/local/rtma* an und kopiert alle notwendigen Dateien in diesen Pfad.

☞ Soll das Paket an eine andere Stelle geschrieben werden, so muß die Datei *xrtma.taz* in die Datei *<Zielverzeichnis>xrtma.tar.Z* kopiert werden, anschließend mit *uncompress xrtma.tar.Z* dekomprimiert und mit *tar -xf xrtma.tar* dearchiviert werden. *xrtma.tar* kann dann gelöscht werden.

2.3. Konfiguration

Für jeden Nutzer, der das Programm verwenden darf, müssen in der Datei */home/<loginname>/.profile* oder in der globalen *.profile* Datei die Zeilen

```
export RTMA=/usr/local/rtma/bin
export PATH=$PATH:$RTMA:
```

nachgetragen werden.

☞ Die Teilzeichenkette "*/usr/local/rtma*" muß durch den entsprechenden Pfad ersetzt werden, wenn das Paket nicht dort zu finden ist.

2.4. Neuübersetzung

☞ Die Quellen befinden sich (falls mitgeliefert) im Verzeichnis *$RTMA/../src*. Zur Neuübersetzung sind OSF/Motif und ein C++ Compiler zwingend erforderlich. Im ersten Teil des Makefiles müssen nur die Optionen gemäß den vorhandenen Ressourcen gesetzt werden, genauere Erläuterungen finden sich in der Datei *makefile*. Vor dem Aufruf von *make* sollten *xrtma*, alle **.o* und alle **.a* Dateien gelöscht werden (*make clean)*. Im Ergebnis des Aufrufes "*make*" entstehen das Executable *xrtma* und die Bibliotheken *librtma.a* und *libxapp.a*. Durch den Aufruf von "*make install*" werden die Dateien an die entsprechenden Stellen kopiert. Die Bibliothek *librtma.a* darf auf keinen Fall aus dem Verzeichnis *$RTMA/bin* entfernt werden, da sie für die Programmierumgebung erforderlich ist (einzige Ausnahme: Verschieben nach */lib* oder */usr/lib*).

3. Erste Schritte

Nach dem Start von X-Windows wird das Auswerteprogramm durch den Aufruf *xrtma* gestartet. Die Menüleiste und das Script-Fenster erscheinen als wichtigste Bestandteile der graphischen Umgebung. Das typische Layout während der Arbeit ist in Abbildung 1 zu sehen. Zum Beenden des Programms wählt man *File→ Exit* oder gibt im Script-Fenster den Befehl *exit* ein.

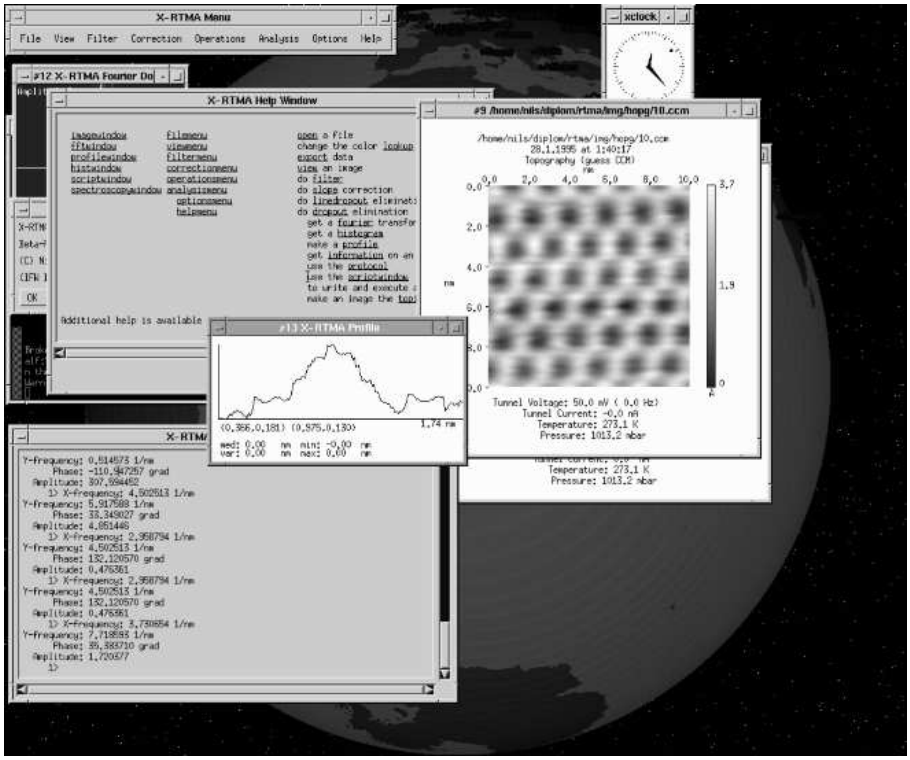

Abb. 1.: Typisches Bildschirm Layout in der graphischen Umgebung

Mit der Maus kann man jetzt zunächst den Menüpunkt *File→ Open* wählen, um eine Datei zu laden. Der Inhalt wird in einem oder mehreren Bild-Fenstern mit dem Dateinamen als Titel dargestellt. Standardmäßig ist die Draufsicht (Topview) eingestellt. Durch die Wahl von *View → Wireframe*, *View→ Solid* oder *View→ Light* kann eine 3-dimensionale Darstellung erzeugt werden. Durch die Wahl von *File→ Export* kann das Bild, nicht der Fensterinhalt, in eine Datei mit dem gewählten Format exportiert werden. Um ein Bild zu drucken, ist ein angeschlossener PostScript-fähiger Drucker notwendig.
Der Menüpunkt *Help→ Index* öffnet das Hilfefenster, mit welchem man sich im Hilfe-Baum interaktiv bewegen kann.

4. Bedienung

Die Bedienelemente der graphischen Oberfläche werden in drei Gruppen unterteilt: Controls, Views und Dialogs.
Zu den Controls gehören die Menüleiste, das Script-Fenster und das Hilfe-Fenster. Die Bild-, Profil-, Fourier-, Histogramm- und Spektroskopie-Fenster werden den Views zugeordnet. Die Einordnung aller verbleibenden Dialogboxen, Abfragen usw. ist klar.

4.1. Views

Jedes Fenster besitzt eine Grundfunktionalität. Es kann in seiner Größe geändert und frei über den Bildschirm bewegt werden. Bei einigen Fenstern erfolgt nach der Größenänderung eine automatische Anpassung an das ursprüngliche Seitenverhältnis. Durch Klicken mit der rechten Maustaste im Fenster wird ein Pulldown-Menü (Abb. 2) aktiviert. Dieses *Basic-Window Menu* erlaubt das Exportieren und Drucken des Fensterinhaltes, sowie das Schließen des Fensters. Weiterhin kann das Fenster um einen festen Faktor in seiner Größe geändert werden. *Edit→ Copy* kopiert den Fensterinhalt in die programminterne Zwischenablage und *Edit→ Paste* von dort in das Fenster. Der Inhalt eines Fensters ist das angezeigte Bild, nicht das durch das Bild repräsentierte Objekt, also z.B. bei einem Bild-Fenster werden die Ansicht und nicht die Bilddaten in die Zwischenablage geschafft.

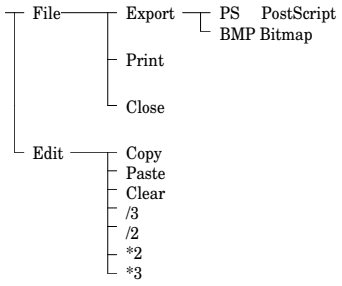

Abb. 2.: Basic-Window Pulldown

! Ein Fenster darf nur durch Wahl von *File→ Close* geschlossen werden.

4.1.1. Bild-Fenster

Geladene Daten werden im Bild-Fenster dargestellt (Abb. 3). Dieses verfügt neben der Standardfunktionalität über weitere spezielle Methoden. Das Klicken mit der mittleren Maustaste öffnet ein Pulldown-Menü, das die Wahl des Beobachterstandortes in der 3D-Ansicht (→ 4.2.1. View-Menü) sowie diverse Messungen erlaubt. Durch Klicken mit der linken Maustaste in den Farbbalken kann der untere Schwellenwert verschoben werden.

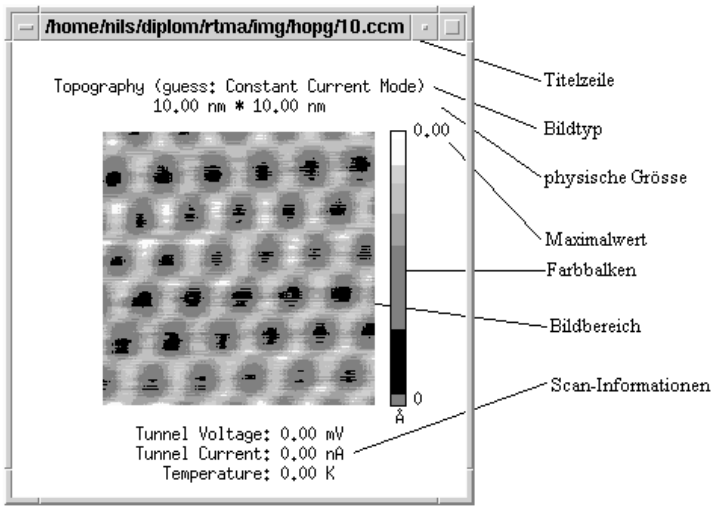

Abb. 3.: Bild-Fenster

Klicken mit der rechten Maustaste verschiebt den oberen Schwellenwert und mit der mittleren Maustaste kann das Zentrum des Farbbereiches geändert werden. Klicken und Halten der linken Maustaste im Bildbereich wirkt als Eingabe für die Meßfunktionen. Als Standardeinstellung wird ein Profil gemessen (→ 4.1.2.). Befindet sich der Mauszeiger im Topview-Modus im Bildbereich, so werden beim Drücken der Taste <F5> die Koordinaten und der Wert an der Position im Script-Fenster angezeigt. Ein Linescan ist ein spezielles Profil, das nur parallel zu den Zeilen des Bildes abgegriffen werden kann. Eine Profilmessung kann abgebrochen werden, indem der Mauszeiger aus dem Bildbereich bewegt wird. Bei der Volumenmessung wird das Volumen des markierten rechteckigen Bildausschnittes über der, durch den minimalen Wert der Messung bestimmten Ebene berechnet. *Measure→ Mark* markiert einen rechteckigen Bildausschnitt (→Ausgleichskorrektur). Mit *Measure→ Scale* kann der Maximalwert explizit gesetzt werden. Dies erlaubt einen Vergleich verschiedener Bilder. Ein Bildfenster behält bei Größenänderung sein ursprüngliches Seitenverhältnis. Mit *Measure→ Zoom* wird das kleinste, der Markierung eingeschriebene Quadrat vergrößert. *Measure→ Line Cross Correlation* bestimmt die Kreuzkorrelationsfunktion entlang der markierten Profillinie, falls zwei Fenster miteinander verbunden sind (→ *Connect*), die Autokorrelationsfunktion sonst.

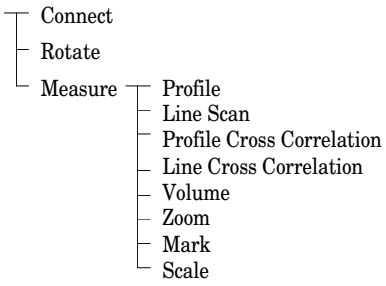

Abb. 5.: Image-Window Pulldown

Im Nicht-Topview Modus wird durch Auswahl von *Rotate* eine Dialogbox geöffnet, die die Eingabe von Azimut und Höhe des Beobachters erlaubt. Die Angaben erfolgen in Altgrad, *Azimut=0* und *Höhe=0* heißt, daß der Beobachter auf der positiven X-Achse steht. Die Winkel werden in mathematisch positivem Sinn gemessen. Alternativ kann die Wahl des Standortes durch Klicken und Ziehen mit der linken Maustaste erfolgen. Die X-Koordinate steuert den Azimut, die Y-Koordinate die Höhe. Durch Selektion von "OK" in der Dialogbox wird die Änderung übernommen, "Cancel" unterbricht die Standortwahl. Mit *Connect* können Bild-Fenster miteinander verbunden werden.

4.1.2. Profil-Fenster

Ein, in einem Bildfenster gemessenes, Profil wird in einem seperaten Fenster als Polygonzug mit Informationen über die Endpunkte der Profillinie,

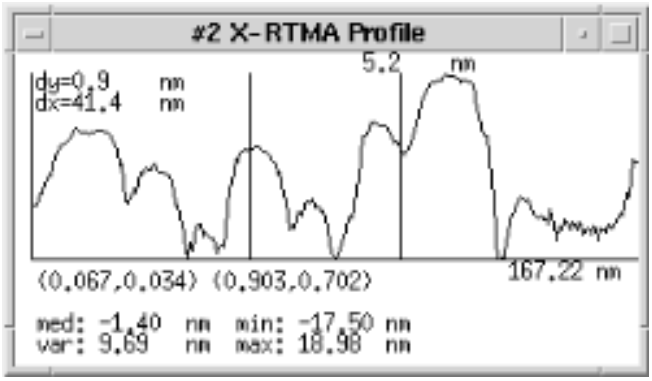

Abb. 6.: Profil-Fenster

die Länge des Profiles, sowie Minimum, Maximum, Mittelwert und Standardabweichung dargestellt. Wird mit der linken Maustaste in das Profil-Fenster geklickt und gehalten, so wird eine Maßlinie an der Position des Mauszeigers eingetragen und der Wert des Profils an dieser Stelle angezeigt. Beim zweiten Klicken wird eine weitere Maßlinie dargestellt und es werden

zusätzlich Höhen- und Wegdifferenz zum ersten Meßwert ausgegeben. Im Profil-Fenster wird auch die Kreuzkorrelation angezeigt.

4.1.3. Spektroskopie-Fenster

In diesem Fenster werden Spektroskopiekurven dargestellt. Wie im Profil-Fenster können auch hier durch Klicken mit der linken Maustaste Meßwerte erfragt werden.

4.1.4. Fourier-Fenster

Die der Analyse dienende Fouriertransformierte eines Bildes (→ 4.2.1.) wird im entsprechenden Fourier-Fenster angezeigt (Abb. 7). Es werden die Art des Spektrums (Amplituden oder Phasenspektrum) sowie die halben maximalen Frequenzkomponenten dargestellt.

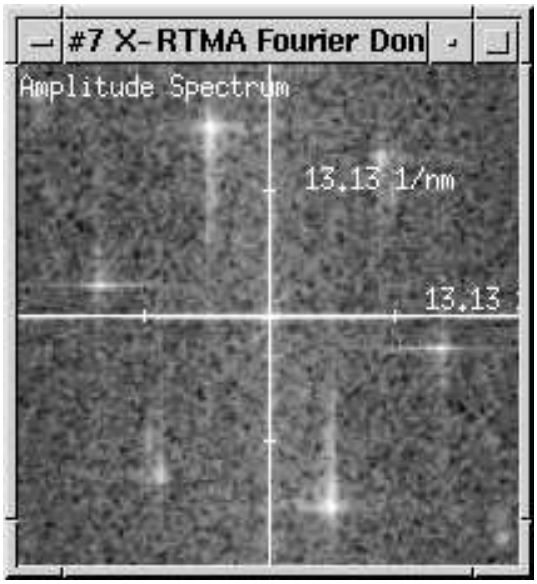

Abb. 7.: Fourier-Fenster

Klicken mit der linken Maustaste zeigt im Script-Fenster (→ 4.2.2.) Informationen zur selektierten Stelle im Frequenzbereich. Durch Klicken mit der mittleren Maustaste kann zwischen Amplituden- und Phasenspektrum gewechselt werden.

4.1.5. Histogramm-Fenster

Im Histogramm-Fenster werden die relative Häufigkeiten einer diskreten Menge von Werten angezeigt. Das H.-Fenster erlaubt keine Manipulation des Histogrammes.

4.2. Controls

4.2.1. Menüfunktionen

Mittels der Menüzeile wird der Ablauf des Programmes interaktiv gesteuert. Folgende Funktionen stehen zur Wahl:

File

New	Öffnet ein neues Bild-Fenster der Größe 300*300, *New* kann nur gewählt werden, wenn schon mindestens ein Bild geladen oder generiert wurde.
Open	Öffnen einer Datei; Akzeptiert werden Dateien im IFW-Format, im NanoScope-III-Format und im DFM-Format. Die Auswahl der Datei erfolgt in der Standard-File Dialogbox.
Lookup Table	Laden einer Farbtabelle; mögliche Tabellen sind: *standard*, *ave*, *fft*, *gold*, *grey*, *light*, *red*, *reverse*, *yellow* und *zizzag*.
Save	Speichern der Bilddaten aus dem obersten Bildfenster in der durch die Titelleiste bezeichneten Datei. Hat das oberste Bildfenster noch keinen Namen, so wird die Standard-File Dialogbox geöffnet. Die Daten werden grundsätzlich im IFW-I-Format abgelegt.
Save As ...	Speichern der Bilddaten aus dem obersten Bildfenster, es wird ein Dateiname abgefragt, die Speicherung erfolgt grundsätzlich im IFW-I-Format.
Export	Exportiert die Bilddaten in das angegebene Format, soll der Inhalt des obersten Bild-Fensters gespeichert werden, so muß das Basic-Window-Pulldown Menü verwendet werden (→ 4.1.1.).

NA3 Nanoscope 3	als NanoScope III Format
PS PostScript	als PostScript
BMP Bitmap	als MS-Windows Bitmap

Die Dateiauswahl erfolgt wieder in der File-Dialogbox.

Print	Drucken der Bilddaten auf einem PostScript-fähigen Drucker. Es werden das Bild und einige Informationen auf einer A4-Seite gedruckt.
Exit	Beenden des Programmes, es erfolgt **keine Abfrage**, ob geänderte Daten gespeichert werden sollen.
View	Setzen des Ansichtsmodus für das oberste Bild-Fenster.
Topview	Draufsicht, die Werte sind durch die Farbgebung kodiert.
Wireframe	Die Bildpunkte werden zeilenweise als Polygonzug angezeigt. Die Darstellung erfolgt 3-dimensional bei freier Wahl des Beobachterstandortes (→ 4.1.1.). Auch hier sind die Wertinformationen zusätzlich durch die Farbgebung kodiert.

Solid Geschlossene Oberflächendarstellung.

Light Darstellung als beleuchtete Fläche. Die Farbinformationen geben hier die Intensität der Beleuchtung an einem Punkt wieder. Die Wertinformationen sind nur in der "Höheninformation" enthalten.

Color overload Sind zwei Fenster miteinander verbunden, so werden, wenn dieser Optionsknopf aktiviert ist, die Farbinformation einer 3D-Ansicht aus den zugeordneten Daten bezogen.

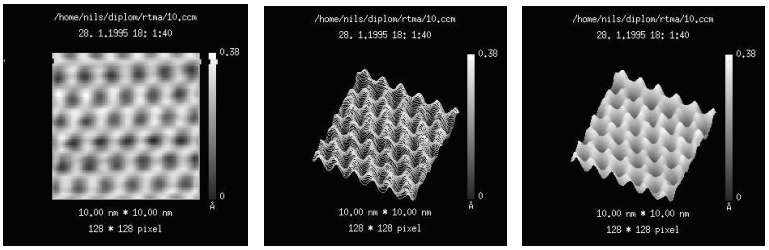

Abb. 8.: Topview, Wireframe und Solid

Filter Zur Verbesserung der Bildqualität können auf die Daten verschiedene Filterfunktionen angewendet werden. Dabei gibt es zwei unterschiedliche Wege: den Filter im Frequenzbereich und den Filter im Ortsbereich. Ersterer wird immer nach dem Schema Bild→Spektrum→manipuliertes Spektrum→Bild ausgeführt. Für diese Filter erscheint es mitunter sinnvoll, erst die Fouriertransformation berechnen zu lassen (→*Analysis*) und mit den dadurch gewonnenen Informationen den Filter bzw. die Parameter des Filters auszuwählen.

Low Pass interaktiver Tiefpaß, nur Frequenzanteile kleiner als die angegebenen Grenzfrequenzen werden übernommen. Zur Eingabe der Filterdaten erscheint eine Dialogbox.

High Pass interaktiver Hochpaß, nur Frequenzanteile größer als die angegebenen Grenzfrequenzen werden übernommen

.

Band Pass interaktiver Bandpaß, nur Frequenzanteile innerhalb der angegebenen Grenzfrequenzen werden übernommen.

Energy Filter interaktiver Energie-Filter, nur Anteile deren Energiedichte größer als das angegebene Niveau (in % vom Maximalwert) ist werden übernommen.

MELP Matched Energy Lowpass; durch Auswertung des Spektrums generierter Filter mit Tiefpaß Charakteristik, garantiert in vielen Fällen den Erhalt der inneren Struktur des Bildes.

SDF Signal Driven Filter; durch die Auswertung des Spektrums erzeugter Filter mit zusammengesetzter Bandpaß Charakteristik. Garantiert den Erhalt der Struktur des Bildes. Dämpft zum ungestörten Signal gehörende Nebengipfel des Spektrums nicht.

Convolution Filter im Ortsbereich, wird durch eine Filtermatrix mit ungerader Dimension bestimmt. Spielt die Genauigkeit der gefilterten Daten eine Rolle, sollten diese Filter nicht verwendet werden.

Free Ermöglicht die freie Eingabe der Filtermatrix, sollte nur von Experten benutzt werden. In der Dialogbox kann die Dimension des Filters in den Grenzen $1 \le n,m \le 13$ gewählt werden. Die Dimensionsänderung wird durch *"Apply dimension"* aktiviert.

CLP Convolution Low Pass, Standardfilter mit Tiefpaß Charakteristik. Dieser Filter glättet das Bild.

Laplacian Filter Gibt das Ergebnis der Anwendung des Laplace-Operators auf das Bild zurück.

Edge Detection Dient der Erkennung von Kanten. Dieser Filter ist aus der Literatur übernommen und könnte besser durch den Betrag des Gradienten (*Analysis→ Absolute of Gradient*) ersetzt werden.

SobelX operator $\partial / \partial x$ auf das Bild angewandt.

SobelY operator $\partial / \partial y$ auf das Bild angewandt.

Correction

Slope Polynomiale Ausgleichskorrektur des Bildes mit Grad $0 \le n \le 20$. Es besteht die Möglichkeit, zur Berechnung des Ausgleichpolynoms nur einen bestimmten Bereich des Bildes zu verwenden. Dazu muß ein Bereich markiert werden (→ 4.1.1.) und der Optionsknopf "*use rectangle*" im Slope-Dialog muß aktiviert werden.

Dropout Elimination Entfernt Punktstörungen (Dropouts) bei vorgegebener Fehlerwahrscheinlichkeit. Die Fehlerwahrscheinlichkeit läßt sich mit Hilfe eines Histogrammes (*Analysis→ Histogram*) sehr gut bestimmen. Es können verschiedene Fehlerwahrscheinlichkeiten für Ausreißer am oberen und unteren Rand angegeben werden.

Line Dropout Elimination Entfernt gestörte Zeilen. Das Verfahren zur Detektion von Zeilenstörungen beruht auf der Untersuchung des Korrelationskoeffizienten zweier aufeinanderfolgender Bildzeilen.

Operations Verknüpft die Operanden und erzeugt ein neues Bild mit Bild-Fenster, welches dem Ergebnis der Operation entspricht. Die Wahl der Operanden erfolgt nach Abfrage durch Klicken.

. + . Summe
. - . Differenz

Analysis

FFT Fast Fourier Transformation; Schnelle Fouriertransformation eines Bildes. Diese Funktion liefert nur Informationen. Sie kann nicht benutzt werden, um das Spektrum des Bildes interaktiv zu ändern. Standardmäßig wird das Amplitudenspektrum mit logarithmischer Wertskala dargestellt.

Cross Correlation Kreuzkorrelation zweier Bilder. Die Wahl der Bilder erfolgt wieder wie bei *Operations*.

Cross Covariance Kreuzkovarianzfunktion ($C_{AB}(x,y) = \big(R_{AB} - E(A)\cdot E(B)\big)(x,y)$).

Histogram Histogramm mit vorgegebener Klassenanzahl. Ein Histogramm kann auch von einem Profil ermittelt werden. Dazu muß das oberste Fenster das entsprechende Profil-Fenster sein

.

Abs. of Gradient Bestimmung des Absolutbetrages des Gradienten.

Options

Info Öffnet einen Informationsdialog, der die wichtigsten Daten des Bildes im obersten Bild-Fenster anzeigt (Abb. 9). Die Informationen können gedruckt oder als ASCII-Text exportiert werden.

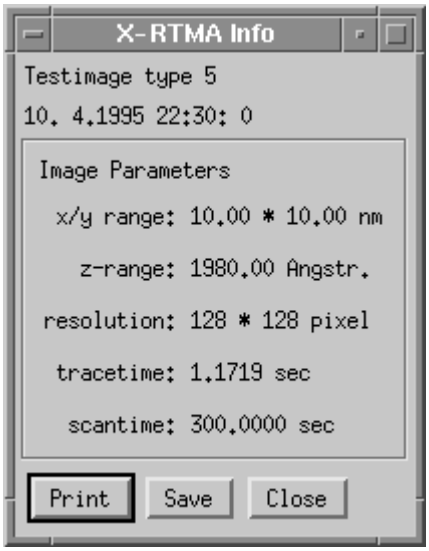

Abb. 9.: Informations-Dialog

Interpolation Festlegen des Interpolationsmodus für die Topview-Darstellung und die Meßwertinterpolation. Möglich ist eine bilineare Interpolation in einer 1-Punkt Umgebung und die C^2-Spline Interpolation. Letztere liefert auf jeden Fall visuell bessere Ergebnisse.

Trace Ein- und Ausschalten des Protokolls. Standard ist ein, d.h. die letzten 30 gerufenen Funktionen werden protokolliert. Das Protokoll kann bei Bedarf in eine Textdatei exportiert werden (→ 4.2.2.).

Set Light Setzten der Position der Lichtquelle für die beleuchtete Darstellung. Das Fenster zeigt eine Draufsicht auf eine Sphäre über der positiven Halbebene, auf der sich die Lichtquelle befindet. Klicken und Halten mit der linke Maustaste verändert die Position, Klicken mit der rechten Taste übernimmt die Änderung.

Display Info Optionsknopf, der die Anzeige von Informationen im Bildfenster ein- oder ausschaltet (Standard: ein).

Font	Setzen der Schriftart für Fenstertexte, die Änderung wird erst nach Neuzeichnen sichtbar. Verwendbare Schriften, Breite x Höhe in Punkten: 6x13, 7x14 (Standard), 9x15, 8x16.
Help	Das Programm verfügt über ein kontextsensitives Hilfesystem, d.h. wird die Taste F1 gedrückt, so erhält man zur jeweils aktiven Funktion den zugehörigen Hilfetext. Zur näheren Erläuterung siehe 4.2.3.
Index	Ruft die Index-Seite des Hilfesystems.
Search	Ermöglicht die Suche nach einem Stichwort.
About	Copyright Meldung

.

4.2.2. Script-Fenster

Das Script-Fenster dient der Eingabe von Befehlen und der Ausgabe von Textinformationen. Nach der Eingabeaufforderung (Prompt)

<Zeilennummer> '> _'

kann ein Befehl eingegeben werden (→ 5.). Informationen aus dem Script-Fenster können mit dem unter X-Windows üblichen Kopiermechanismus sofort in eine andere Anwendung übernommen werden.

4.2.3. Hilfe-System

Das Hilfesystem bietet eine vollständige Dokumentation aller Funktionen der graphischen Oberfläche. Um Hilfe für eine Funktion zu bekommen, hat man drei Möglichkeiten:

1. Drücken der Taste *F1*

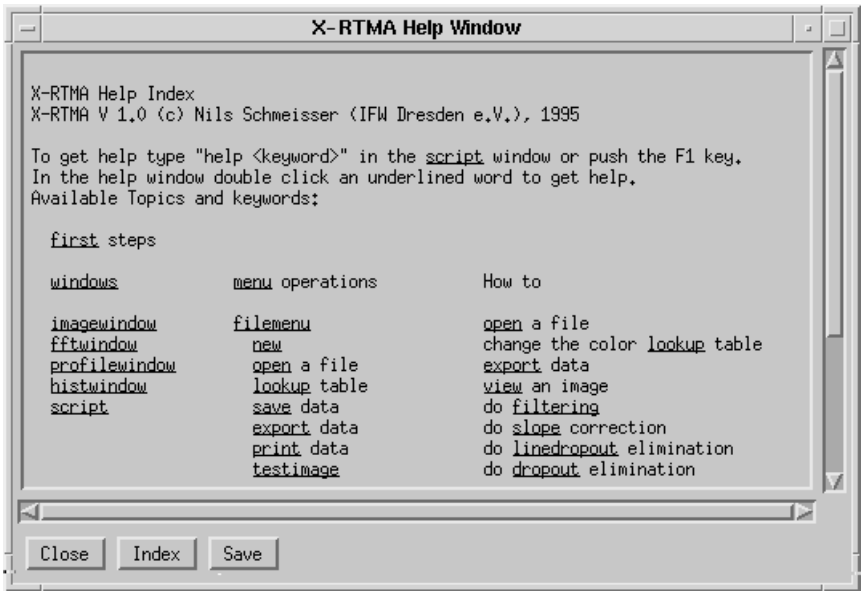

Abb. 10.: Hilfe-Index

2. Auswahl von *Help→ Index* für den Hilfe-Index (Abb. 10)

3. Auswahl von *Help→ Search* um nach einem Stichwort zu suchen

Der Hilfe-Text enthält unter Umständen durch Unterstreichung gekennzeichnete Querverweise. Zu den entsprechenden Seiten gelangt man durch Doppelklicken auf ein Wort. Auch die, nicht als Querverweise gekennzeichneten Worte im Text können verwendet werden, führen aber nicht notwendig zu neuen Informationen. Die aktuelle Seite kann mit *"Save"* als ASCII-Text exportiert werden. *"Index"* liefert den Hilfe-Index.

4.2.4. Der Editor

Der Editor erlaubt die Bearbeitung von Textdateien im ASCII-Format und die Erstellung von Scripts (→6.). Ein Script kann mit dem *[RunScript]* Knopf zur Ausführung gebracht werden.

5. Kommandosprache

Die Kommandosprache ist kontextsensitiv, d.h. Groß- und Kleinschreibung sind wesentlich. Die Kommandos werden im Script-Fenster eingegeben.
Zur Notation: - durch ' eingeschlossene Symbole müssen so stehen
- durch < > eingeschlossene Identifikatoren werden durch ihre Werte ersetzt
- durch [ ] eingeschlossene Symbole sind optional und müssen nicht stehen
- durch { } eingeschlossene und mit | voneinander getrennte Symbole sind Alternativen, d.h. genau ein Symbol aus { } wird verwendet

Datei-Kommandos:

'load' <fname> Öffnen der Datei mit dem Namen <fname>

'export' <type> <fname> Exportieren des Bildes des obersten Bild-Fensters in eine Datei vom Typ <type> mit dem Namen <fname>. Mögliche Typen sind:

BMP	MS-Windows Bitmap
PS	PostScript
NA3	NanoScope III-Format
IFW	IFW-Format

'exit' beendet das Programm

'edit' [<fname>] Öffnen des Editors und laden der Datei *<fname>*, falls angegeben..

'run' <sname> Startet das Script in der Datei *<sname>*

Hilfe-Kommandos:

'help' [<keyword>] ruft die Hilfe zum Stichwort <keyword>, falls kein Stichwort angegeben wird, wird der Hilfe-Index gerufen

'about' gibt die Copyright-Meldung aus

Protokoll-Kommandos:

'trace' gibt das Protokoll aus

'trace' { 'on' | 'off' } schaltet das Protokoll an bzw. aus, das Protokoll ist per Voreinstellung aktiviert.

'trace clear' löscht den Inhalt des Protokolls

'trace' <fname> schreibt das Protokoll als Text in die Datei mit dem Namen *<fname>*

'info' [<fname>] gibt die Informationen zum Bild im obersten Bild-Fenster aus und speichert sie, falls ein Dateiname angegeben wurde, als ASCII-Text.

Meß-Funktionen:

Die Meßfunktionen beziehen sich jeweils auf das Bild im obersten Bildfenster.

'LINES'	Anzahl Zeilen
'POINTS'	Anzahl Meßwerte pro Zeile
'DATE'	Datum und Uhrzeit des Starts der Messung
'SCANTIME'	Dauer der Messung in Sekunden
'TRACETIME'	Zeit pro Zeile in Sekunden
'SIZE'	physische Größe der Bildes
'OFFSET'	Offset
'BIAS'	Tunnelspannung
'TUNNELCURRENT'	Tunnelstrom
'TEMPERATURE'	Temperatur während der Messung
'PRESSURE'	Druck im Gerät während der Messung
'POTVOLTAGE'	Falls das oberste Bild eine Potentialverteilung repräsentiert, liefert dieses Kommando die Größen der angelegten Potentiometriespannung.
'SCANMODE'	Abtastmodus
'TYPE'	Typ des Bildes im obersten Bild-Fenster
'NAME'	Dateiname des Bildes im obersten Bild-Fenster
'PEAKTOPEAK'	Maximum-Minimum, dieser Wert entspricht dem, im Bild-Fenster angezeigten Maximalwert
'MEANSQUARE'	Standardabweichung der Meßwerte
'SKEWNESS'	Surface Skewness
'KURTOSIS'	Surface Kurtosis
'ROUGHNESS'	Average Surface Roughness

6. XRTMA-library

Die Funktionalität des Programms kann auch ohne graphische Oberfläche genutzt werden. Dazu ist ein gewisses Grundverständnis der Sprache C/C++ notwendig. Ein Programm oder Script besteht aus zwei Dateien: dem nicht zu verändernden Rahmen und der Script-Datei, welche eine Folge von Deklarationen und Anweisungen enthält.

Vorgehen: 1. Erstellen der Script-Datei mit dem Namen <*sname*>
2. Aufruf von *runscript* <*sname*>

<*sname*> darf nicht den Namen *"/tmp/rtma/xrtma.scr"* haben.
Ein Script besteht aus einem Funktionsdefinitionsteil und einem Hauptprogramm. Der Funktionsdefinitionsteil enthält Definitionen im C/C++ Stil und muß sich am Anfang der Script-Datei befinden. Das Hauptprogramm wird durch das Schlüsselwort *'BEGIN'* eingeleitet und durch den Bezeichner *'END'* abgeschlossen. Variablendeklarationen dürfen an jeder Stelle stehen. Beispiel:

```
void Hallo() {
   printf( "Hallo user.\n");
}

BEGIN
   Hallo();
END
```

6.1. Deklarationen

Eine Deklaration besteht aus einem Typbezeichner gefolgt von einer Liste von Identifikatoren. Es können alle C Standardtypen verwendet werden. Zusätzlich stehen die Typen *complex*, *fftstruct* und *Image* zur Verfügung.

complex = komplexe Zahlen mit double Komponenten
Konstruktoren: *complex()* erzeugt die komplexe Zahl *0+i*0*
complex(a) erzeugt die Zahl *a+i*0*
complex(a,b) erzeugt die Zahl *a+i*b*

Sei *c=a+i*b* eine Instanz vom Typ *complex*, dann liefert

real(c)	den Realteil *a*
imag(c)	den Imaginärteil *b*
abs(c)	den Absolutbetrag $\|c\| = \sqrt{a^2 + b^2}$
arg(c)	das Argument φ von $c = \|c\| \cdot e^{i \cdot \varphi}$
conj(c)	die konjugiert komplexe Zahl $\bar{c} = a - i \cdot b$
exp(c)	e^c

fftstruct = Ergebnis der Fast Fourier Transformation und Eingabe für die inverse Transformation. Den zum Frequenzpaar *(u,v)* gehörenden Wert $F_{i,j}$ erhält man aus der Instanz *fftstruct fft* durch:

```
i=(int)(v/VMAX(fft)*DIMY(fft)/2);
if (i<0) i=ny+i;
j=(int)(u/UMAX(fft)*DIMX(fft)/2);
if (j<0) j=nx+j;
F_{i,j}=fft.data[i*nx+j];.
```

Vorsicht: $(u,v) \in [-UMAX(fft), UMAX(fft)] \times [-VMAX(fft), VMAX(fft)]$!

Image = Bildtyp, eigentlicher Gegenstand der Bearbeitung

Konstruktoren:

Image()	erzeugt eine neue Instanz vom Typ *Image*
Image(fname)	erzeugt aus dem Inhalt der Datei *fname* eine Image-Instanz
Image(fname,k)	erzeugt aus dem *k*-ten Bild der Datei *fname* eine Image-Instanz

Image(fname,&imgp) erzeugt eine Instanz aus dem Inhalt der Datei *fname* und erzeugt eine durch *Image *imgp* referenzierte Instanz, falls ein zweites Bild in der Datei vorhanden ist. Falls kein zweites Bild vorhanden ist, hält *imgp* den Wert *NULL*.

Zwei Instanzen vom Typ *Image* können mit den Infix-Operatoren +, -, * verknüpft werden. + und - sind die im üblichen Sinne punktweise Addition bzw. Subtraktion, * liefert die Faltung. Die Anwendung der Operatoren * und / auf *Image <op> float* ist ebenfalls zulässig und repräsentiert Verstärkung beziehungsweise Dämpfung der Werte.

Beispiel: *int i,j;* deklariert die Variablen *i* und *j* als Variablen vom Typ Integer
Image I1; deklariert *I1* als Variable vom Typ *Image*

6.2. Anweisungen

Sämtliche C/C++ Konstrukte sind Anweisungen, d.h. die Zuweisung

```
I1=Image("testbild.ccm");
```

ist eine gültige Anweisung (hier Zuweisung der durch den Aufruf *Image("testbild.ccm")* erzeugten *Image*-Instanz an die Variable *I1*, "=" heißt hier auch Copy-Konstruktor).

Beispiel:

```
BEGIN
   Image I1,I2,I3;            // Deklaration von I1, I2 und I3

   I1=Image("bild1.ccm");  // Laden der Datei bild1.ccm
   I2=Image("bild2.ccm");  // Laden der Datei bild2.ccm
   I3=I1*I2;                  // I3=Faltung von I1 mit i2
   EXPORT(I3,BMP,"bild3.bmp");      // Speichern von I3 als Windows-Bitmap
END
```

7. Scriptfunktionen

7.1. Steuerfunktionen

SETIPMODE(MODE)
Setzen des Interpolationsmodus auf *SIMPLEIP* - bilineare Interpolation in 1-Punkt Umgebung *SPLINEIP* - C^2-Spline Interpolation. Standardeinstellung ist SIMPLEIP.

int GETIPMODE
Gibt den derzeitigen Interpolationsmodus zurück.

7.2. Dateifunktionen

Image Image(FNAME)
Gibt den Inhalt der Bilddatei mit dem Namen *FNAME* zurück. Akzeptiert werden Dateien, die im IFW-, NanoScope III- und DFM-Format vorliegen.

Image Image(FNAME,K)
Gibt das *k*-te Teilbild der Datei zurück.

*char **BROWSE(FNAME)*
Gibt die Bezeichnung der Teilbilder der Datei *FNAME* als Feld von *0*-terminierten Strings zurück. Das Feld wird durch den Eintrag *charvar[i][0]=0* abgeschlossen.

EXPORT(IMG,TARGET,FNAME)
Exportiert das Bild *IMG* in die Datei mit dem Namen *FNAME* und dem Typ:

BMP	Windows-Bitmap
POSTSCRIPT	PostScript-Datei
IFW	IFW-Format
NA3	NanoScope III-Format

SETLUT(LUTNAME)
Setzen der Farbtabelle (für den Export in eine Windows-Bitmap Datei). Zur Verfügung stehen "*standard*", "*ave*", "*fft*", "*gold*", "*grey*", "*light*", "*red*", "*reverse*", "*yellow*" und "*zizzag*". "*standard*" ist voreingestellt.

7.3. Messfunktionen

Absolute Meßwerte werden in der durch *UNITZ(IMG)* bestimmten Einheit zurückgegeben. *IMG* ist eine Instanz vom Typ *Image*, also z.B. eine Variable *I*, die als *Image I;* deklariert wurde.

float PEAKTOPEAK(IMG)
Peak-To-Peak Value nach ISO 4287/1

float MEANSQUARE(IMG)
Standardabweichung der Meßwerte nach ISO 4287/1

float SKEWNESS(IMG)
Surface Skewness nach ISO 4287/1

float KURTOSIS(IMG)
Surface Kurtosis nach ANSI B.46.1

float ROUGHNESS(IMG)
Average Surface Roughness nach DIN 4768

float GET(IMG,X,Y)
Gibt den Wert an der Stelle *(X,Y)* zurück. Die Koordinaten werden in relativen Koordinaten zwischen 0 und 1 angegeben. Nullpunkt ist die linke obere Bildecke.

float[] PROFILE(IMG,X0,Y0,X1,Y1,L)
Gibt ein 1-dimensionales Feld mit *L* Einträgen zurück, welches das Profil zwischen den Punkten *(X0,Y0)* und *(X1,Y1)* enthält.

float VOLUME(IMG,X0,Y0,X1,Y1)
Gibt das Volumen des durch *(X0,Y0)* und *(X1,Y1)* bestimmten rechteckigen Bildausschnittes bezüglich des Minimums im Bild an.

float LINEMEANVALUE(IMG,LINE)
Gibt den Mittelwert von Zeile *LINE* zurück.

float LINESTANDARDDEVIATION(IMG,LINE)
Standardabweichung der Meßwerte in Zeile *LINE*.

float LINECOVARIANCE(IMG,LINE1,LINE2)
Kovarianz der Meßwerte in den Zeilen *LINE1* und *LINE2*.

float LINECORRELATION(IMG,LINE1,LINE2)
Korrelationskoeffizient der Meßwerte in den Zeilen *LINE1* und *LINE2*.

float PROFILECOVARIANCE(IMG1,IMG2,XA,YA,XB,YB)
Kovarianz entlang der Profillinie *(XA,YA),(XB,YB)*.

float PROFILECORRELATION(IMG1,IMG2,XA,YA,XB,YB)
Korrelationskoeffizienten entlang der Profillinie *(XA,YA),(XB,YB)*.

Image CORRELATION(IMG1,IMG2)
Liefert die Kreuzkorrelation der Bilder *IMG1* und *IMG2*.

int[] HISTOGRAM(IMG,NC)
Liefert das Histogramm des Bildes *IMG* bei *NC* Klassen.

int LINES(IMG)
Liefert die Anzahl der Bildzeilen.

int POINTS(IMG)
Liefert die Anzahl der Meßwerte pro Zeile.

STARTTIME(IMG,H,M,S)
Gibt die Startzeit des Scans in den Integer Variablen *H, M* und *S* zurück.

DATE(IMG,D,M,Y)
Gibt das Datum der Messung zurück.

float SCANTIME(IMG)
Liefert die Gesamtdauer des Scans in Sekunden als Gleitkommazahl.

float TRACETIME(IMG)
Liefert die für eine Zeile benötigte Zeit in Sekunden.

float SIZEX(IMG)
Liefert die X-Ausdehnung in der Einheit *UNITX(IMG)*.

float SIZEY(IMG)
Liefert die Y-Ausdehnung in der Einheit *UNITY(IMG)*.

float SIZEZ(IMG)
Liefert die Z-Ausdehnung in der Einheit *UNITZ(IMG)*, *SIZEZ=Maximum-Minimum*.

float SCALEZ(IMG)
Liefert den Skalierungsfaktor, mit dem aus den Integer-Meßwerten der einheitenbehaftete Meßwert berechnet wird, d.h. *Z=SCALEZ(IMG)*IMG(I,J)*. Dies ist dasselbe wie der Aufruf *Z=GET(IMG,J/(nx-1),I/(ny-1))*.

char[] UNITX(IMG)
Liefert die Einheit der X-Ausdehnung als 0-terminierten String.

char[] UNITY(IMG)
Liefert die Einheit der Y-Ausdehnung als 0-terminierten String.

char[] UNITZ(IMG)
Liefert die Einheit der Z-Ausdehnung als 0-terminierten String.

float OFFSETX(IMG)
Gibt das X-Offset der Messung in nm zurück.

float OFFSETY(IMG)
Gibt das Y-Offset der Messung in nm zurück.

float FILTER(IMG)
Gibt die Grenzfrequenz eines, bei der Messung vorgeschalteten Tiefpaßfilters in Hz zurück.

float BIAS(IMG)
Liefert die Tunnelspannung in mV.

float BIASFREQ(IMG)
Liefert die Frequenz der Tunnelspannung in Hz. Falls mit Gleichspannung getunnelt wurde, ist *BIASFREQ*=0.

float TEMPERATURE(IMG)
Gibt die Temperatur während der Messung in K zurück. *TEMPERATURE=288.15 K*, falls keine Temperaturmessung erfolgte.

float PRESSURE(IMG)
Liefert den Druck während der Messung in mBar. *PRESSURE= 1013.25 mbar*, falls keine Druckmessung erfolgte.

float ITUNNEL(IMG)
Liefert den Tunnelstrom in nA. ! Nicht im Constant Height Mode.

float UPOT(IMG)
Liefert die Größe der angelegten Potentiometriespannung in mV im Potentiometrie-Modus.

float UPOTFREQ(IMG)
Liefert die Frequenz der Potentiometriespannung.

float SPEUMIN(IMG)
Minimale Spektroskopiespannung in *mV*.

float SPEUMAX(IMG)
Maximale Spektroskopiespannung in *mV*.

float SPEU0(IMG)
Stelle, an der der Meßwertabgriff erfolgte (*SPETYPE=U0* oder *=DER*).

float SPEI0(IMG)
Stromwert, für den die Spannung bestimmt wurde (*SPETYPE=ISO*).

int MICTYPE(IMG)
Liefert den Typ des Mikroskopes, mit dem das Bild aufgezeichnet wurde:
STM: Rastertunnelmikroskop
AFM: Rasterkraftmikroskop

int SCANMODE(IMG)
Gibt den Meßmodus zurück:
CCM: Constant Current Mode
CHM: Constant Height Mode
CON: Contact Mode

int IMGTYPE(IMG)
Gibt den Bildtyp zurück:
TOP: Bild repräsentiert Topographiedaten
POT: Bild repräsentiert eine Potentialverteilung
SPE: Bild repräsentiert Spektroskopiedaten
HAR: Harmonische

int SPETYPE(IMG)
Spektroskopietyp
U0: Strom an einer Stelle U_0
DER: Ableitung an der Stelle U_0
ISO: Isofläche, Spannung für vorgegebenen Strom
I_U: Strom-Spannungs-Kennlinie

char[] IMGNAME(IMG)
Liefert den Namen (=Dateinamen) des Bildes als 0-terminierten String.

7.4. Filterfunktionen

LOWPASSF(IMG,FA,FB)
Tiefpaßfilterung mit den Grenzfrequenzen *FA* und *FB*, angegeben in *1/Länge* in X- bzw. Y-Richtung. Das Argument *IMG* wird verändert.

HIGHPASSF(IMG,FA,FB)
Hochpaßfilterung mit den Grenzfrequenzen *FA* und *FB* in X- bzw. Y-Richtung. Das Argument *IMG* wird verändert.

BANDPASSF(IMG,FXMIN,FXMAX,FYMIN,FYMAX)
Bandpaßfilterung mit *(FXMIN,FXMAX),(FYMIN,FYMAX)* als Durchlaßbereich. Das Argument *IMG* wird verändert.

ENERGIEF(IMG,ERG)
Energiefilter des Bildes *IMG*. Nur Frequenzanteile deren Energiedichte größer oder gleich *ERG/100*<Maximale Energiedichte>* ist, werden durchgelassen.

MATCHEDLPF(IMG)
Signalangepaßter Tiefpaß.

SIGNALSENSF(IMG)
Signalgesteuerter Bandpaß.

HISTOGRAMF(IMG,A,B)
Histogrammfilter, die Randwerte des Histogrammes werden als Fehler betrachtet und entsprechend den vorgegebenen Fehlerwahrscheinlichkeiten *A* und *B* am oberen und unteren Rand des Histogrammes abgetrennt. *A* und *B* werden im Bereich *0..1* angegeben.

CLOWPASSF(IMG)
Convolution Lopass des Bildes *IMG*.

LAPLACE(IMG,N)
Anwendung des Laplace-Operators Δ auf das Bild *IMG*.

EDGEDETECTION(IMG)
Edge Detection Filter, liefert eine Betonung der Kanten des Bildes. Zur Erkennung von Kanten sollte der Betrag des Gradienten verwendet werden (*GRADIENT(IMG)*).

SOBELX(IMG)
Sobel Operator in X-Richtung, entspricht der Anwendung von $\partial / \partial x$.

SOBELY(IMG)
Sobel Operator in Y-Richtung, entspricht der Anwendung von $\partial / \partial y$.

GRADIENT(IMG)

Betrag des Gradienten des Bildes ($\sqrt{\left(\frac{\partial}{\partial x}\right)^2 + \left(\frac{\partial}{\partial y}\right)^2}$).

7.5. Korrekturfunktionen

SLOPECORRECTION(IMG,DEGREE)

Ausgleichskorrektur des Bildes mit Polynom vom Grad *DEGREE*.

LINEDROPOUT(IMG)

Eliminiert gestörte Zeilen aus dem Bild *IMG*.

Zur Elimination von Dropouts (Punktstörungen) ist der Histogrammfilter *HISTOGRAMF* zu verwenden.

7.6. Fourier-Tools

FDATA: Datentyp für die FFT (=*fftstruct*).
DATA: Datentyp des Ergebnisses der IFFT (=*short[][]*).

FDATA FFT(IMG)

Schnelle Fouriertransformation des Bildes *IMG*.

DATA IFFT(FFTDAT)

Inverse Schnelle Fouriertransformation des Frequenzbereiches *FFTDAT* (Instanz vom Typ *FDATA*).

float UMAX(FFTDAT)

Liefert die maximale Frequenz in X-Richtung in *1/UNITX*.

float VMAX(FFTDAT)

Liefert die maximale Frequenz in Y-Richtung in *1/UNITY*.

8. Bibliotheksfunktionen

☞ Die folgenden Ausführungen und Übersichten sind nur für den erfahrenen C/C++ Programmierer gedacht.
Für diesen sollte es leicht sein, die Funktionalität der RTMA-Bibliothek zu erweitern.

8.1. Typedefinitionen

```
typedef struct timerec {
        unsigned short int year,month,day,wday;
        unsigned short int hour,minute,second,second100;
       };
```

Die Information in der ***timerec*** Struktur ist nur gültig, wenn die Komponente ***timrec***.*day* ungleich *0* ist.

```
typedef struct infostruct {
        int min,max;            // Minimum und Maximum der Bildwerte
        int mictype;            // Mikroskoptyp: STM, AFM
        int mtype;              // Meßmodus: CCM, CHM, CON
        int type;               // Bildtyp: TOP, POT, SPE, HAR
        int stype;              // Spektroskopieart: U0, DER, I_U, ISO

        int nx,ny,numpoints;    // Anzahl Zeilen,Spalten, Anzahl Stützstellen bei Kurven
        timerec start,stop;     // Start und Stopzeit
        float tracetime,scantime; // Zeit pro Zeile, Gesamtzeit
        float sizex,sizey,sizez;  // physische Größe in X/Y und Z-Skalierung
        float offsetx,offsety,offsetz; // Offset in X, Y und Z
        char unitx[6],unity[6],unitz[6]; // Einheiten für diese Daten

        float filter;           // Grenzfrequenz eines vorgeschalteten Teifpaßfilters in Hz
        float bias;             // Tunnelspannung in mV
        float biasfreq;         // Frequenz der Tunnelspannung in Hz
        float temperature;      // Temperatur in K
        float pressure;         // (Gas)Druck in mbar
        float tunnel_current;   // Tunnelstrom in pA, nur CCM
        float U_pot;            // Potentiometriespannung in mV
        float potfreq;          // Frequenz dazu in Hz
        int   potgain;          // internal POT gain
        int   hdegree;          // Grad der Harmonischen
        float hphase;           // Phase der Harmonischen
        float usmin,usmax;      // Spektroskopiespannungsintervall
        float us0,is0;          // Stellen U0  bzw. I0
        int repeat;             // Anzahl Meßwertabgriffe pro Punkt (nur SPE)

       char* name;              // Bildname=Dateibame
        char ctx[256];          // Mikroskop Kontext: UHV, LC, ITC4, ...
        char stext[256];        // Geräte Kommentarfeld
        char history[256];      // Umgebungs Kommentarfeld
        char text[256];         // Bild Kommentarfeld
};
```

```
class Image {
  public:
    short int **data;                    // Bilddaten
    infostruct info;                     // Bildinformationen

    Image();                             // Standardkonstruktor
    Image(char *fname);                  // Lädt das erste Bild der Datei fname
    Image(char *fname,int cnt);          // Lädt das cnt-te Bild
    Image(char *fname,Image** img2);     // Lädt die ersten beiden Bilder von fname
    Image(short **data_,infostruct inf); // Erzeugen eines Bildes aus Daten und
                                         // Informationen
    Image(int type,int nx_,int ny_);     // Testbild

    void SlopeCorrection(int degree,float x0=0.0,float y0=0.0,float x1=1.0,float y1=1.0);
                                         // Ausgleichskorrektur vom Grad degree über
                                         // dem Fenster (x0,y0),(x1,y1)
    void MinMax();                       // Minimum und Maximum neu bestimmen
    short Get(int i,int j);              // Wert in Zeile i, Spalte j
    short Get(float x,float y);          // Wert an der Stelle (x,y) aus [0,1]x[0,1]
    Image Zoom(float X0,float Y0,float X1,float Y1);      // Vergrößern des Ausschnittes

    Image &operator =(const Image &image);    // Copy-Konstruktor
    Image operator +(Image &b);          // Addition
    Image operator -(Image &b);          // Subtraktion
    Image operator *(Image &b);          // Kreuzkorrelation
    Image operator *(float x);           // Verstärkung (x>1)
    Image operator /(float &x);          // Dämpfung (x>1)

    virtual ~Image();                    // Destruktor
};
```

Die Semantik der Konstruktoren wurde schon unter 6.1 erläutert. *Image(int type,int nx_,int ny_)* erzeugt ein Testbild. *Save(char *fname)* speichert die Instanz als IFW-Datei mit dem Namen *fname*. *MinMax()* bestimmt Minimum und Maximum der Daten *data[][]* und schreibt sie in *info.min* und *info.max*. *Get(float,float)* liefert einen gemäß dem gesetzten Interpolationsmodus bestimmten Wert. *Zoom(float,float,float,float)* liefert eine neue Instanz, die den angegebenen Ausschnitt enthält. Die Punktauflösung bleibt erhalten, die Werte von *info* werden angepaßt.

8.2. Funktionsreferenz

8.2.1. Dateifunktionen

#define BMP 0
#define TIF 1
#define POSTSCRIPT 2
#define IFW 3
#define DFM 4
#define DME 5
#define NA2 6
#define NA3 7
#define ASCII 8

Dateitypkonstanten, *TIF*, *DME* und *NA2* werden nicht benutzt.

char Copyright*

Copyright String

*int Export(Image *image,int target,char *fname)*

Exportieren eines Bildes *image* in die Datei *fname* vom Typ *target*. Liefert im Fehlerfall einen Wert ungleich *0*.

*void Append(Image *image,int target,char *fname)*

Anhängen von der Daten des Bildes *image* an die Datei *fname* vom Typ *target*. Typ der bestehenden Datei und *target* müssen übereinstimmen.

*short **loadfile(char *fname,infostruct *info)*

Liefert die Rohdaten aus der Datei *fname* und gibt die zugehörigen Informationen in *info* zurück. Im Fehlerfall liefert *loadfile* den Wert *NULL*.

*short **load2ndfile(char *fname,infostruct *info)*

Lädt ein eventuell vorhandenes zweites Teilbild aus der Datei *fname*. Im Fehlerfall wieder *NULL*.

*char **IFW1Browser(char *fname)*

Inhaltsverzeichnis eine IFW-I Datei laden und als *0*-terminiertes Feld zurückgeben, d.h. der erste Eintrage char[k][0]=0 terminiert das Feld.

*short **LoadIFW1(char *fname,infostruct *info,int part)*

Laden der *part*-ten Komponente einer IFW-I Datei.

*void ExportToPostScript(Image *image,char* Filename,float scale,char* lutname)*

Exportiert die Daten von *image* in eine PostScript-Datei. Falls *lutname* ungleich *NULL* ist, erfolgt die Ausgabe farbig gemäß der angegebenen Farbtabelle. Zusätzlich zum Bild werden Informationen über Größe, Meßpunktzahl u.ä. mit ausgegeben. Mittels der Skalierung *scale* kann die Größe der Grafik beeinflußt werden.

8.2.2. Filterfunktionen

#define LOWPASS 0
#define HIGHPASS 1
#define BANDPASS 2
#define CONVOLUTION 3

*void lowpass_Filter(Image *image,float fx,float fy)*
Tiefpaßfilter mit Grenzfrequenzen *fx* und *fy*.

*void highpass_Filter(Image *image,float fx,float fy)*
Hochpaßfilter mit Grenzfrequenzen *fx* und *fy*.

*void bandpass_Filter(Image *image,float fxmin,float fxmax,float fymin,float fymax)*
Bandpaß mit Durchlaßbereich *(fxmin,fxmax),(fymin,fymax)*.

*void energy_Filter(Image *image,float minerg)*
Energiefilter mit Energiesperre bei *(minerg*100) %* der maximalen Energiedichte.

*void MLowpassFilter(Image *image)*
Signalangepaßter Tiefpaß.

*void sigsens_Filter(Image *image)*
Signalsensitiver Filter.

*void histogram_Filter(Image *image,float A,float B)*
Histogrammfilter mit Fehlerwahrscheinlichkeiten *A* und *B*.

*void convolution_lowpass_Filter(Image *image)*
Convolution Lowpass.

*void Laplace(Image *image,int n)*
Laplacefilter mit Parameter *n*.

*void EdgeDetection(Image *image)*
Edge detection Filter.

*void convolution_Filter(short **data,int nx,int ny,convdata cd)*
Convolution Lowpass mit freier Eingabe der Filtermatrix.
Struktur für den Convolution Free-Filter:

```
typedef struct convdata {
          int n,m;
          int **w;
        };
```

n und *m* müssen ungeradzahlig sein. Der Filter wird nach der Formel

$\tilde{s}_{i,j} = \frac{1}{|F|} \sum_{k=-n}^{n} \sum_{l=-m}^{m} F_{n,m} \cdot s_{i-k,j-l}$; berechnet. |*F*| ist die Summe der Werte der Filtermatrix.

Für Randpunkte von *s* wird die Filtermatrix intern so geändert, daß Punkte außerhalb von *s* nicht benutzt werden.

*void SobelX(Image *image)*

Sobel Operator in X-Richtung. $F = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$

*void SobelY(Image *image)*

Sobel Operator in Y-Richtung. $F = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

*void Gradient(Image *image)*

Betrag des Gradienten von *image*.

8.2.3. Interpolationsfunktionen

```
#define SIMPLEIP  0
#define SPLINEIP  1
```

void SetIPMode(int mode)

Setzen des Interpolationsmodus, *SIMPLEIP*: bilineare Interpolation in 1-Punkt Umgebung, *SPLINEIP*: C^2-Spline Interpolation.

int GetIPMode()

Gibt den aktuellen Interpolationsmodus zurück.

*int Interpolate(short **data,int nx,int ny,float x,float y)*

Liefert den interpolierten Wert auf dem Platz *(x,y)* über dem *nx*ny* Gitter *data[][]*. $x, y \in [0,1]$

float c2spline(float x,int n,float y)*

Liefert den interpolierten Wert an der Stelle *x* auf dem *n*-Gitter *y[]*.

void c2spline(int n,float y,float* m,float* a,float *b)*

Gibt die zur C^2-Spline Interpolation nötigen Daten zurück. *n* und *y[]* sind vorgegeben. Für die Felder *m*, *a* und *b* müssen vor dem Aufruf *n+1* Speicherplätze vom Typ *float* allokiert werden.

*float c2spline(float x,int n,float*m,float* a,float* b)*

Erzeugt aus den vorgegebenen Daten der C^2-Spline Interpolation den approximierten Wert an der Stelle *x*. Die Aufruffolge:

```
float *a,*b,*m;
a=new float[n+1];
b=new float[n+1];
m=new float[n+1];
c2spline(n,y,m,a,b);
z=c2spline(x,n,m,a,b);
delete a;
delete b;
delete m;
```

ist äquivalent mit dem Aufruf

z=c2spline(x,n,y);

*short int **fastc2spline(float X0,float Y0,float X1,float Y1, short **data,in nx,int ny)*
Optimierter C^2-Spline-Zoom des Ausschnittes *(X0,Y0),(X1,Y1)* des Feldes *data[][]* in ein Feld der gleichen Dimension.

*short int **interpolate2d(float X0,float Y0,float X1,float Y1,short **data,int nx,int ny)*
Zoom unter Verwendung des gesetzten Interpolationsmodus.

8.2.4. Korrektur

*void SlopeCorr(float *image,int n,int m, int degree)*
Polynomiale **2-dimensionale** Ausgleichskorrektur vom Grad *degree*. Das 2-dimensionale Feld muß vor dem Aufruf in ein 1-dimensionales Feld durch Hintereinanderschreiben der *n* Zeilen überführt werden.

*void SlopeCorr(float *image,int n,int m, int degree,*
float x0,float y0,float x1,float y1,int inside)
Polynomiale Ausgleichskorrektur vom Grad *degree*, unter Benutzung der Werte innerhalb des Rechteckes *(x0,y0),(x1,y1)*, falls *inside=1*, der Werte außerhalb, falls *inside=0*.

*void LineDropoutCorrection(Image *image)*
Eliminiert gestörte Zeilen aus dem Bild *image*.

8.2.5. Statistikfunktionen

#define HISTSIZE 0x1000
Größe des anzulegenden Feldes für ein Histogramm, =größte Anzahl Klassen.

Im folgenden gibt, wo nicht anders erläutert, *int n* die Anzahl der Wert an.

*unsigned int *Histogram(short *data,int n,int nclass)*
*unsigned int *Histogram(float *data,int n,int nclass)*
*unsigned int *Histogram(short **data,int nx,int ny,int nclass)*
*unsigned int *Histogram(Image *image,int nclass)*
*unsigned int *Histogram(complex *data,int n,int nclass)*
Histogramm von *data* mit *nclass* Klassen.

*float MeanValue(short *data,int n)*
*float MeanValue(float *data,int n)*
Statistisches Mittel der Werte *data*.

*float StandardDeviation(short *data,int n)*
*float StandardDeviation(float *data,int n)*
Standardabweichung der Werte *data*.

*float Covariance(short *data1,short *data2,int n)*
*float Covariance(float *data1,float *data2,int n)*
Kovarianz der Wertfolgen *data1* und *data2*.

*float Correlation(short *data1,short *data2,int n)*
*float Correlation(float *data1,float *data2,int n)*
Korrelationskoeffizient der Wertfolgen *data1* und *data2*.

*float Modal(float *data,int n,int nclass)*
*float Modal(complex *data,int n,int nclass)*
Modalwert der Werte von *data* bei *nclass* Klassen.

*float Modal(float *data,int n)*
*float Modal(complex *data,int n)*
Modalwert der Werte von *data* bei *n/4* Klassen.

*float LineMeanValue(Image *image,int line)*
Statistisches Mittel der Werte in Zeile *line* des Bildes *image*.

*float LineStandardDeviation(Image *image,int line)*
Standardabweichung der Werte in Zeile *line* des Bildes *image*.

*float LineCovariance(Image *image,int line1,int line2)*
Kovarianz der Zeilen *line1* und *line2* des Bildes *image*.

*float LineCorrelation(Image *image,int line1,int line2)*
Korrelationskoeffizient der Zeilen *line1* und *line2* des Bildes *image*.

Image Correlation(Image *I1,Image *I2)*
Kreuzkorrelation von *I1* und *I2*.

8.2.6. Fourier-Tools

```
#define FEPS 1e-5       //
#define AMPLITUDE 1 // Amplitudenspektrum
#define PHASE     2     // Phasenspektrum

typedef struct fftstruct {
        float umax,vmax;
        int nx,ny;
        complex *data;
};
```

*fftstruct FFT(int n,short *data)*
1-dimensionale Schnelle Fouriertransformation der *n* Werte im Feld *data*. *fftstruct.umax=1*, *fftstruct.vmax* wird nicht benutzt. *fftstruct.nx=n*.

fftstruct FFT(Image &image)
2-dimensionale Schnelle Fouriertransformation des Bildes *image*. Die Werte fftstruct.umax und fftstruct.vmax werden gemäß

$$u_{\max} = \frac{POINTS(image)}{SIZEX(image)} \text{ und } v_{\max} = \frac{LINES(image)}{SIZEY(image)}$$

bestimmt.

*short **IFFT(fftstruct &fft)*
: Inverse 2-dimensionale Fouriertransformation des Frequenzbereiches *fft* (siehe auch *class Image*).

*void Get(fftstruct f,float x,float y,float *fx,float *fy,float *phase,float *amp)*
: Liefert die Frequenzen *fx* und *fy*, die Phase und die Amplitude an der, durch *(x,y)* relativ adressierten Stelle im Frequenzbereich *f*.

A.1. Dateiindex

Die Installationsdiskette hat ein MS/DOS-Filesystem und enthält folgende Dateien:

xrtma.tar	XRTMA-Archiv, enthält das vorcompilierte Executable, die Bibliotheken librtma.a und libxapp.a, Farbtabellen, Hilfedateien, Dokumentationen und die Quelltexte.
rtmainst	XRTMA Installationsscript
version	Textdatei mit Versionsinformation

A.2. Referenz

IFW	Institut für Festkörperphysik und Werkstofforschung Dresden e.V.
IBM	ist ein Warenzeichen der International Business Machine Corp.
AIX	ist ein eingetragenes Warenzeichen von IBM
UNIX	ist ein eingetragenes Warenzeichen von AT&T
X-Windows	ist ein Warenzeichen des M.I.T.
MS-DOS	ist ein eingetragenes Warenzeichen der Microsoft Corp.
PostScript	ist ein Warenzeichen von Adobe Systems Inc.
NanoScope	ist ein eingetragenes Warenzeichen von NanoScope
DFM	Danish Institute of Fundamental Metrology
DME	ist ein Warenzeichen von Danish Micro Engineering A/S