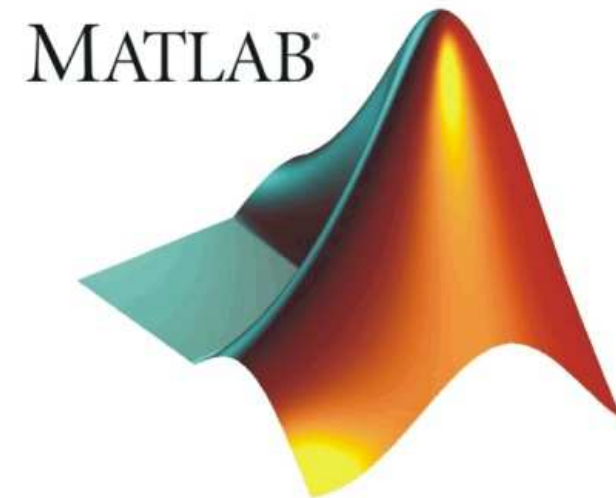




Einführung in MATLAB

Prof. Ingrid Scholl
Bildverarbeitung WS 2013/2014



Entstehung von MATLAB

Aus <http://de.wikipedia.org/wiki/Matlab>:

„Matlab wurde Ende der 1970er Jahre von Cleve Moler an der Universität New Mexico entwickelt, um den Studenten die Fortran-Bibliotheken LINPACK und EISPACK für lineare Algebra von einer Kommandozeile aus ohne Programmierkenntnisse in Fortran zugänglich zu machen.

Zusammen mit Jack Little und Steve Bangert gründete Moler 1984 *The MathWorks* und machte Matlab zu einem kommerziellen Produkt, das zusammen mit einer ersten Funktions-Sammlung, der *Control System Toolbox*, vor allem in der Regelungstechnik viele Anwender fand... “

MathWorks ist heute der führende Entwickler und Vertreiber von fachspezifischer Simulations- und Analyse Software. Aktuell über 1600 Mitarbeiter weltweit, davon über 30% in Produktentwicklungen. Privates Unternehmen, von Beginn profitabel.



Cleve Moler
Chief Scientist



Jack Little
Präsident



Was ist MATLAB?



- MATLAB ist eine hoch-effiziente Umgebung, um wissenschaftlich mathematische Berechnungen durchzuführen und zur grafischen Darstellung der Ergebnisse
- Beinhaltet eine Entwicklungsoberfläche, um Berechnungen, Visualisierungen und Programmierung zu ermöglichen
- MATLAB steht für „*MATrix LABoratory*“, ermöglicht das numerisch effiziente Rechnen mit Matrizen (integriert LINPACK, EISPACK- und BLAS-Bibliotheken)
- Anwendung in Universitäten für *Mathematik, Ingenieurwissenschaften und Informatik*
- Anwendung in der Industrie für *Hochproduktive Forschung, Entwicklung und Analyse*

Warum MATLAB?

- Über 1.000.000 Kunden weltweit aus innovativen technologischen Unternehmen und Forschungseinrichtungen, Finanzinstituten und mehr als 3.500 Universitäten
- MATLAB und Simulink sind derzeit Standardsoftware in der Wissenschaft und der Industrie
- Verfügt über hocheffiziente Algorithmen
- Wird nach neuester Technologie weiter entwickelt
- Verfügt über diverse Funktionsbibliotheken



MATLAB Homepage

www.mathworks.com

The screenshot shows the MathWorks homepage with the following elements:

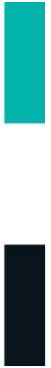
- Header:** MathWorks logo, tagline "Accelerating the pace of engineering and science", language selector (Deutschland), links for Kontakt, Store, Account Anlegen, and Anmelden.
- Navigation Bar:** Links for Produkte & Dienstleistungen, Lösungen, Forschung und Lehre, Support, User Community, Veranstaltungen, and Unternehmen. The first three are circled in red.
- Sub-headers:** MATLAB Produktfamilie, Simulink Produktfamilie, Polyspace Produktfamilie, and Dienstleistungen.
- Main Content Area:**
 - MATLAB:**
 - Parallel Computing:** Parallel Computing Toolbox, MATLAB Distributed Computing Server.
 - Mathematik, Statistik und Optimierung:** Symbolic Math Toolbox, Partial Differential Equation Toolbox, Statistics Toolbox, Curve Fitting Toolbox, Optimization Toolbox, Global Optimization Toolbox, Neural Network Toolbox, Model-Based Calibration Toolbox.
 - Entwurf und Analyse von Steuerungssystemen und Regelungssystemen:** Control System Toolbox, System Identification Toolbox, Fuzzy Logic Toolbox, Robust Control Toolbox, Model Predictive Control Toolbox, Aerospace Toolbox.
 - Signalverarbeitung und Kommunikation:** Signal Processing Toolbox, DSP System Toolbox, Communications System Toolbox, Wavelet Toolbox, Fixed-Point Toolbox, RF Toolbox, Phased Array System Toolbox.
 - Bildverarbeitung und Computer Vision:** Image Processing Toolbox, Computer Vision System Toolbox, Image Acquisition Toolbox, Mapping Toolbox.
 - Test- und Messtechnik:** Data Acquisition Toolbox, Instrument Control Toolbox, Image Acquisition Toolbox, OPC Toolbox, Vehicle Network Toolbox.
 - Computational Finance:** Financial Toolbox, Econometrics Toolbox, Datafeed Toolbox, Fixed-Income Toolbox, Financial Derivatives Toolbox.
 - Computational Biology:** Bioinformatics Toolbox, SimBiology.
 - Codegenerierung:** MATLAB Coder, Filter Design HDL Coder.
 - Erstellung eigenständiger Applikationen:** MATLAB Compiler, MATLAB Builder NE (for Microsoft .NET Framework), MATLAB Builder JA (for Java language), MATLAB Builder EX (for Microsoft Excel), Spreadsheet Link EX (for Microsoft Excel).
 - Datenbankanbindung und Reporting:** Database Toolbox, MATLAB Report Generator.

On the right side, there is a large image of a robotic arm with the text "Hardware Model Robot Simulation" and a smaller image of a bar chart with the text "MATLAB, Simulink weitere Produkte testen".

Das MATLAB System

... besteht aus 5 Bereichen:

1. Entwicklungsumgebung
2. MATLAB Mathematische Funktionsbibliothek
3. MATLAB Programmierung
4. Grafik
5. MATLAB API (Application Program Interface)



Das MATLAB System

... besteht aus 5 Bereichen:

1. Entwicklungsumgebung

MATLAB Desktop, Command Window, Command History, Editor, Debugger, Browser für die Hilfe, Ansichten des Arbeitsspeichers und der Dateien und des aktuellen Arbeitsverzeichnisses

2. MATLAB Mathematical Function Library

3. MATLAB Programmierung

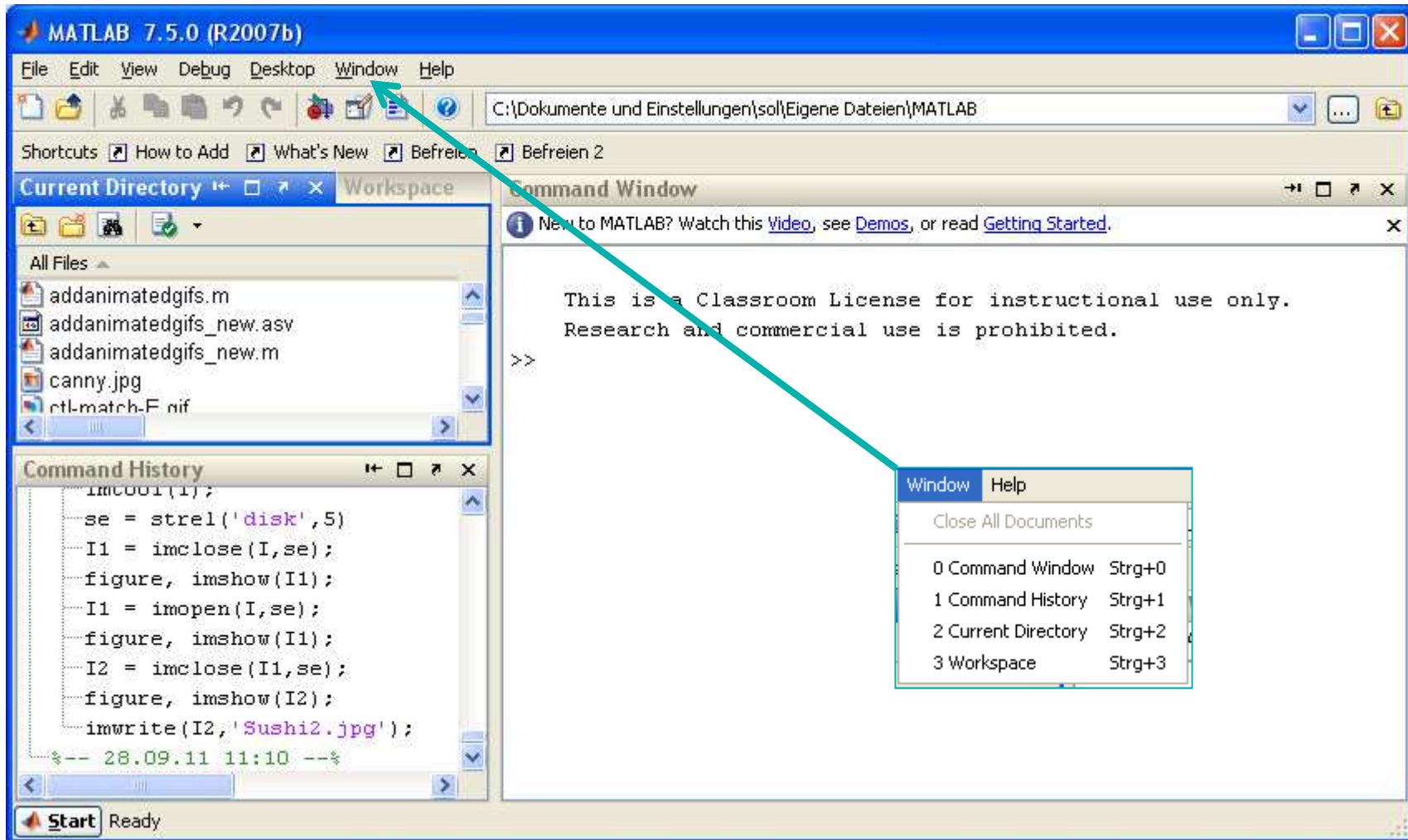
4. Grafik

5. MATLAB API (Application Program Interface)

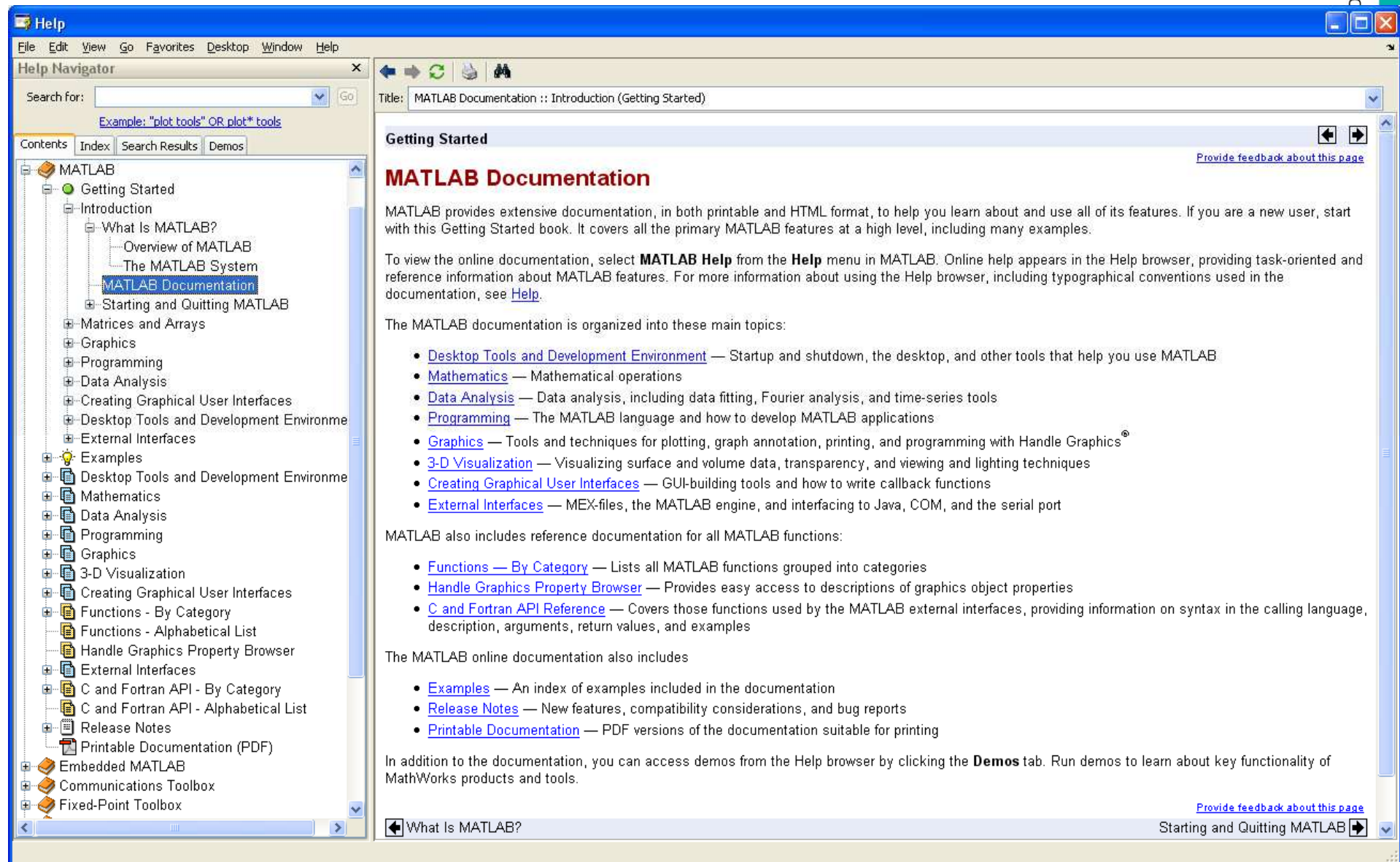


MATLAB System

1. Entwicklungsumgebung



MATLAB Dokumentation



Das MATLAB System

... besteht aus 5 Bereichen:

1. Entwicklungsumgebung

2. **MATLAB Mathematische Funktionsbibliothek:**

Sammlung von mathematischen numerischen Algorithmen wie sum, sine, cosine, complexe Arithmetik, Inverse und Eigenwerte einer Matrix, Fouriertransformation

3. MATLAB Programmiersprache

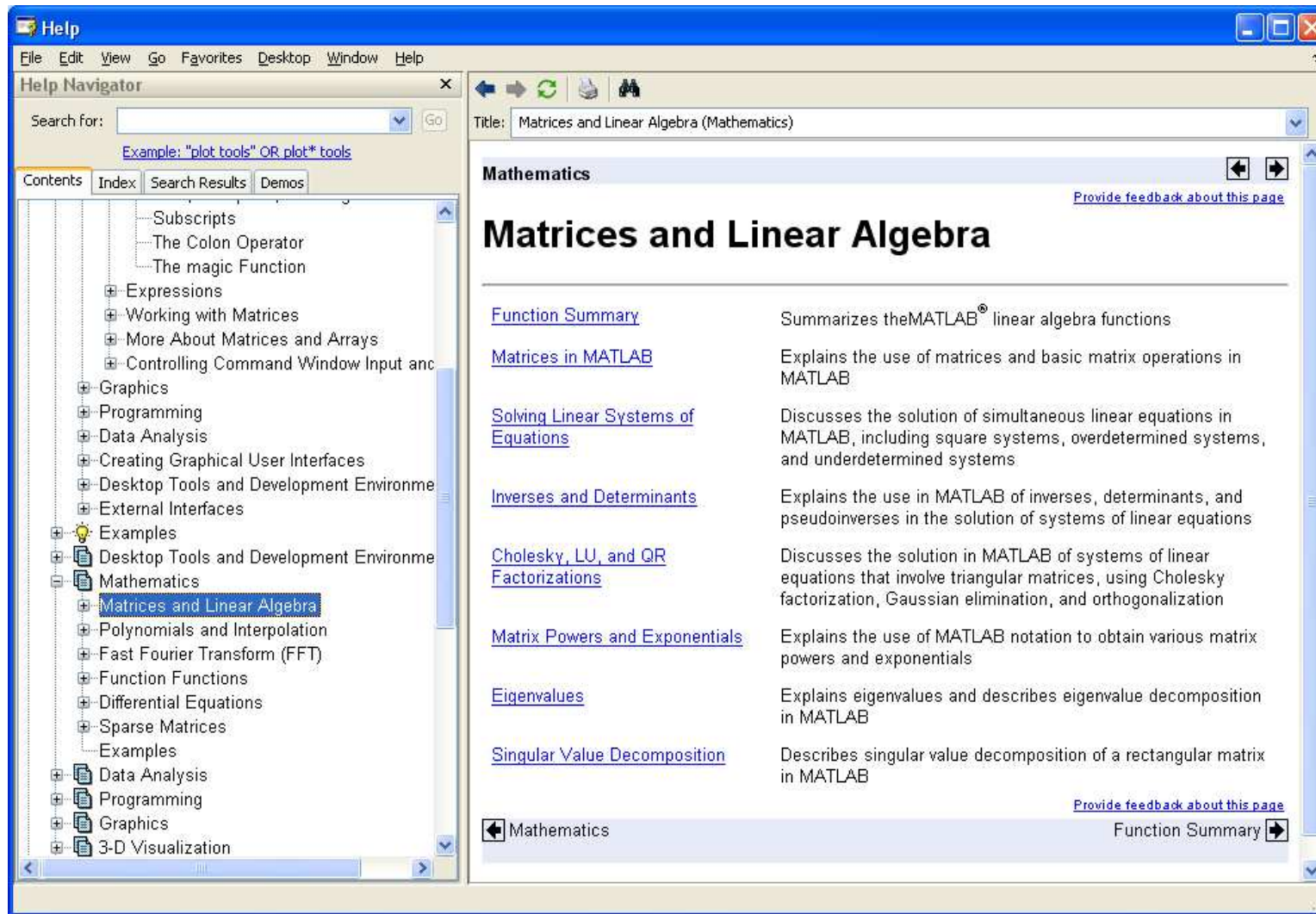
4. Grafik

5. MATLAB API (Application Program Interface)



MATLAB System

2. Mathematische Funktionsbibliothek



Das MATLAB System

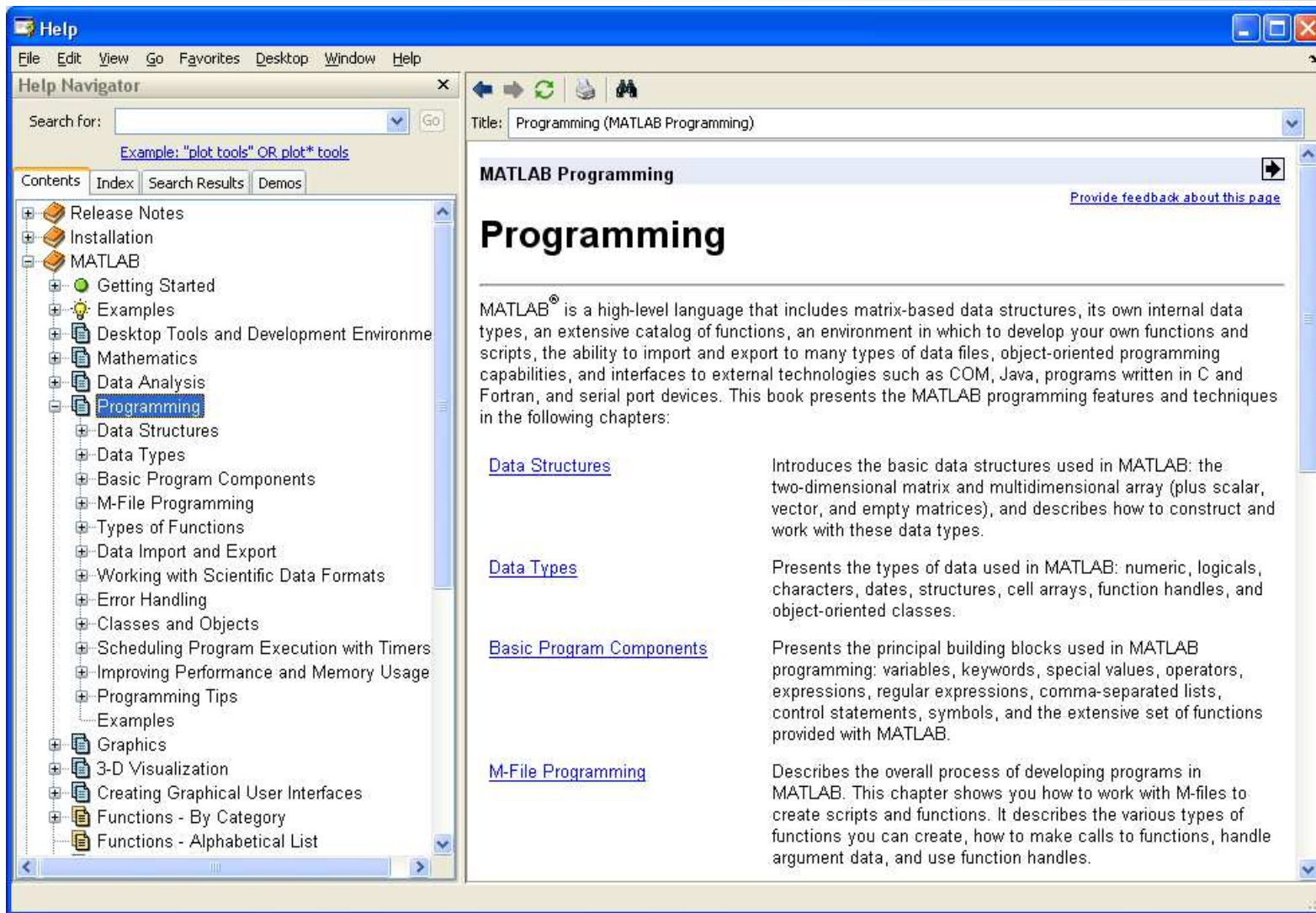
... besteht aus 5 Bereichen:

1. Entwicklungsumgebung
2. MATLAB Mathematical Function Library
3. **MATLAB Programmierung**
Matrix-/Array-basierte Sprache mit Kontrollstrukturen, Funktionen, Datenstrukturen, Ein-/Ausgabe und objektorientierte Programmierung
4. Grafik
5. MATLAB API (Application Program Interface)



MATLAB System

3. Programmierung





... besteht aus 5 Bereichen:

1. Entwicklungsumgebung
2. MATLAB Mathematical Function Library
3. MATLAB Sprache

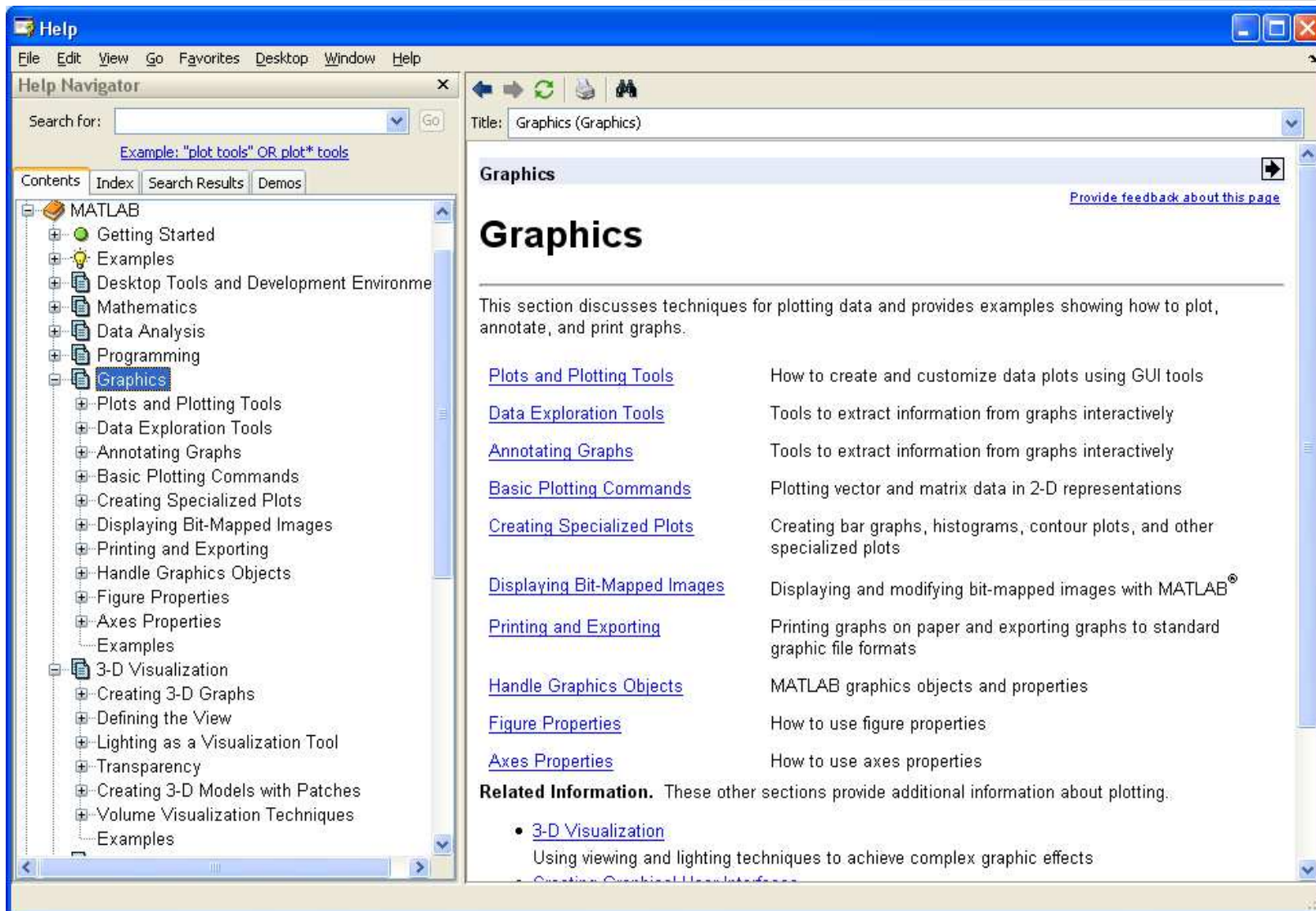
4. Grafik:

Funktionen zur grafischen Anzeige und Ausgabe sowie Annotation von Vektoren und Matrizen, 2D- und 3D-Visualisierung, Bildverarbeitung, Animation und Präsentations-Grafiken. Funktionen zur Bearbeitung von Grafikeigenschaften. Grafischer Editor zur Erstellung von MATLAB-GUIs.

5. MATLAB API (Application Program Interface)

MATLAB System

4. Grafik



Das MATLAB System

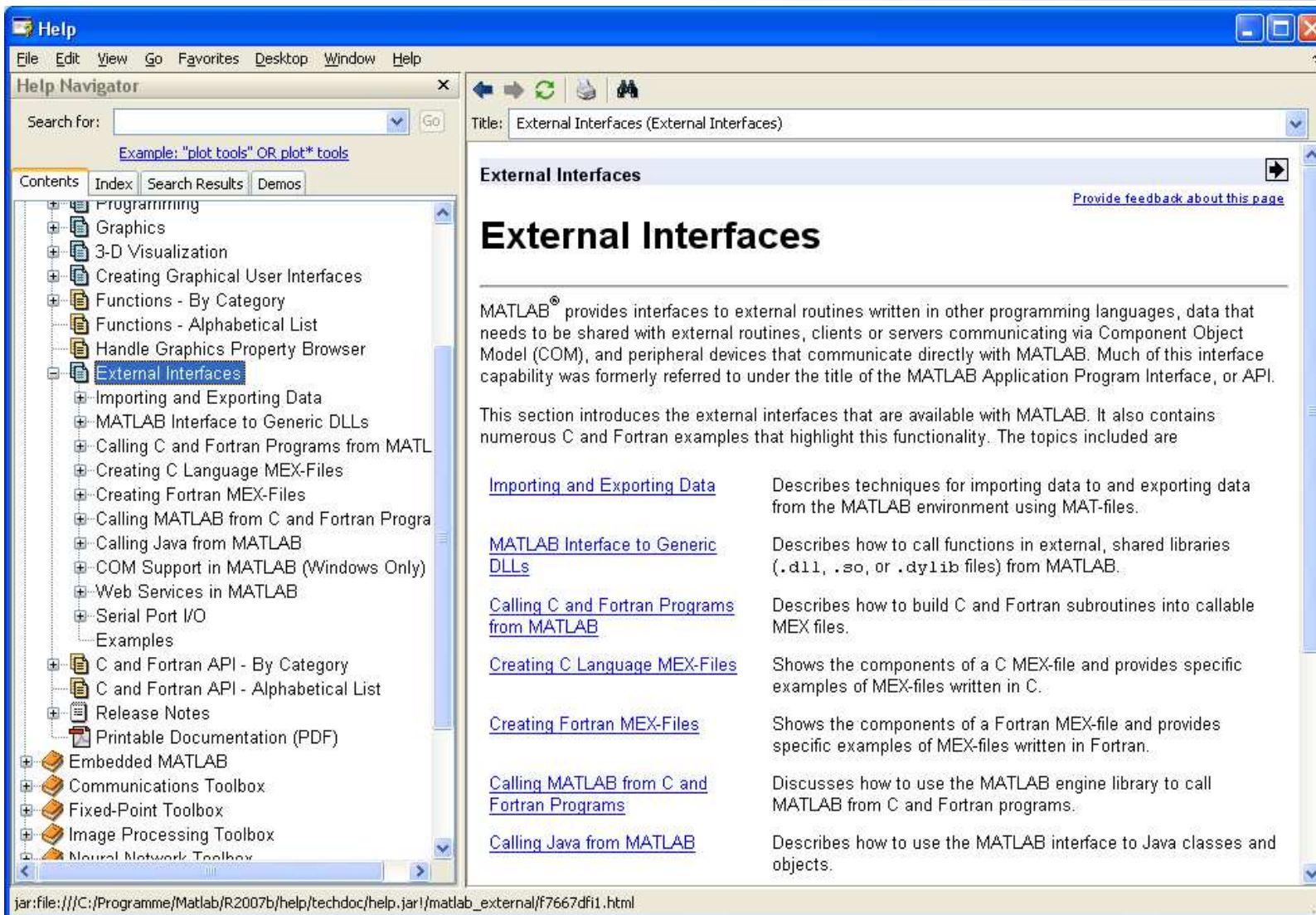
... besteht aus 5 Bereichen:

1. Entwicklungsumgebung
2. MATLAB Mathematical Function Library
3. MATLAB Sprache
4. Grafik
5. **MATLAB API (Application Program Interface):**
C- und Fortran-Programme können über diese Bibliothek MATLAB-Funktionen aufrufen (dynam. Linken), MATLAB für komplexe Berechnungen nutzen sowie MATLAB-Dateien lesen und schreiben.



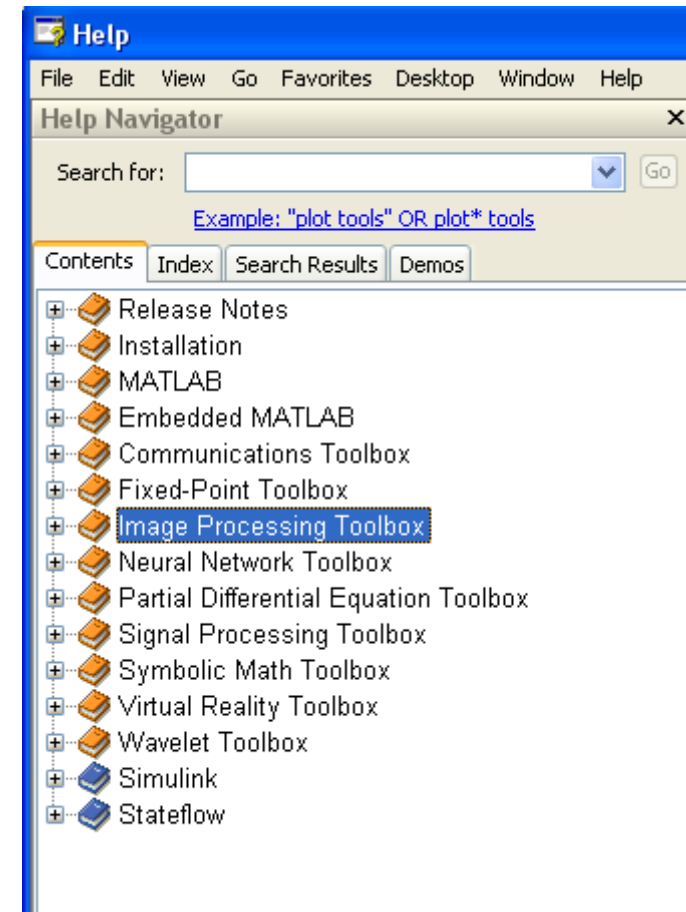
MATLAB System

5. APIs



MATLAB Toolboxes

„*Toolboxes*“ sind Bibliotheken von MATLAB-Funktionen und –Skripten (M-Dateien) zu einer spezifischen Anwendung





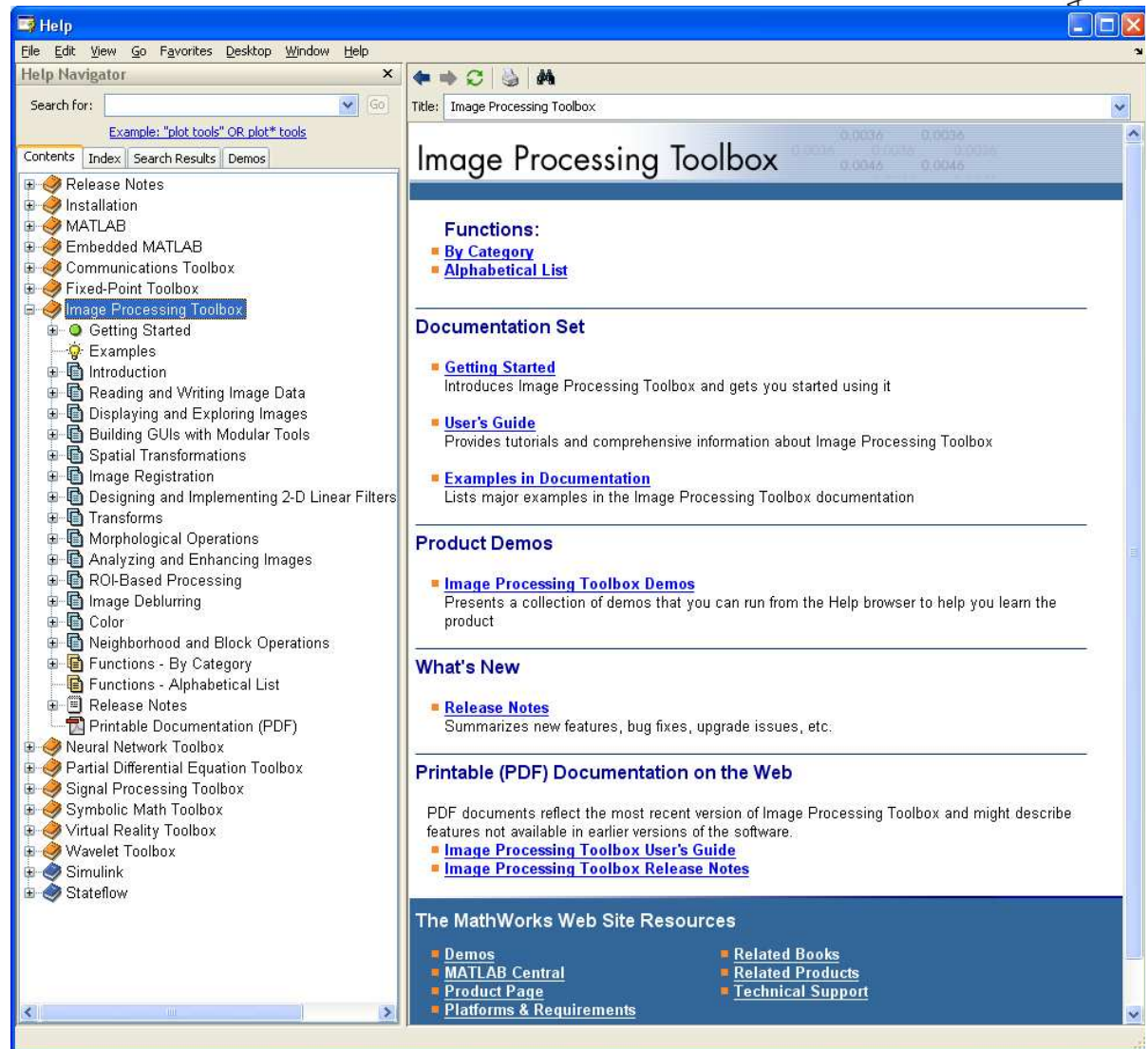
MATLAB – Image Processing Toolbox:

MATLAB – Wavelet Toolbox:

- Tutorials, Demos und Hilfe (mit F1 in MATLAB oder):
Hauptmenü Help in MATLAB und Product Help
- Praktikumsaufgaben werden im Ilias zur Verfügung gestellt
- Ziel: praktische Anwendung der Theorie aus der Vorlesung und Kenntnisse in der MATLAB Programmierung
- 6 Praktika 14-tägig mittwochs von 8:15-11:45 Uhr
- OpenSource Matlab Simulation (GPL Lizenz):
<http://www.gnu.org/software/octave/>

Was leistet die Image Processing Toolbox?

- Lesen/Schreiben von Bildern
- Bildverbesserung durch Bildvorverarbeitung
- 2D-Ortsfilterungen und 2D-Frequenzfilterung
- Isoliert „Region of Interest“ (ROIs) zur Weiterverarbeitung
- Bildregistrierung
- Transformationen
- Rauschunterdrückung
- Farbbildverarbeitung
- Bewegungsverfolgung von Objekten in einer Bildserie
- ...



Matlab – Links zur weiteren Dokumentation

- Webinars: www.mathworks.com/company/events
- Matlab Newsgroups und Dateiaustausch: www.mathworks.com/matlabcentral
- Komplette Produktdokumentation: www.mathworks.com/access/helpdesk/help/helpdesk.html
- User Community mit vielen Beispielprogrammen
- Webinare





MATLAB

Grundlagen



1. Erstellen von Vektoren und Matrizen
2. Doppelpunktoperator, Punktoperator und Indizieren von Matrizen
3. Boolesche Indizierung
4. Plots und Subplots
5. Bilder und Farbtabellen (Pseudofarb-Tabelle)
6. M-Files: Skripte und Funktionen
7. Datenstrukturen in Matlab

1. Erstellen von Vektoren und Matrizen

```
>> a = [1 2 3 4]
a =
    1     2     3     4
```

← Zeilenvektor

```
>> b = [1; 2; 3]
b =
     1
     2
     3
```

← Spaltenvektor

```
>> c = [1 2 ; 3 4]
c =
     1     2
     3     4
```

← Matrix





```
>> a = [1 2 3 4];  
>> d = a'
```

```
d =
```

```
    1  
    2  
    3  
    4
```

```
>> a = [1 2 ...  
3 4]
```

```
a =
```

```
    1    2    3    4
```

- Semikolon am Ende einer Zeile unterdrückt die Ausgabe
- Semikolon innerhalb einer Vektordefinition erzeugt Spaltenvektoren
- Die Transponierte eines Vektors v erhält man durch v'
- Ein Zeilenumbruch wird durch ... gekennzeichnet und ist erforderlich

2.1 Doppelpunkt-Operator

```
>> a = [2 3 4 5]
```

```
a =
```

```
     2     3     4     5
```

```
>> b = 2:5
```

```
b =
```

```
     2     3     4     5
```

```
>> c = 10:-2:4
```

```
c =
```

```
    10     8     6     4
```

➤ Der : Operator wie 2:5 wird zum Erstellen von und Indizieren innerhalb von Vektoren und von Matrizen verwendet.

➤ Der mittlere Wert -2 gibt eine Schrittweite an.



2.2 Punkt-Operator



```
>> a = [1 2; 3 4]
```

```
a =
```

```
     1     2
     3     4
```

```
>> b = [5 6; 7 8]
```

```
b =
```

```
     5     6
     7     8
```

```
>> a*b
```

```
ans =
```

```
    19    22
    43    50
```

```
>> a.*b
```

```
ans =
```

```
     5    12
    21    32
```



Der . Operator führt die nachfolgende mathematische Multiplikation oder Division **elementweise** durch:
 $a.*b = a_{ij} * b_{ij}$ für alle Matrixelemente

```
>> a./b
```

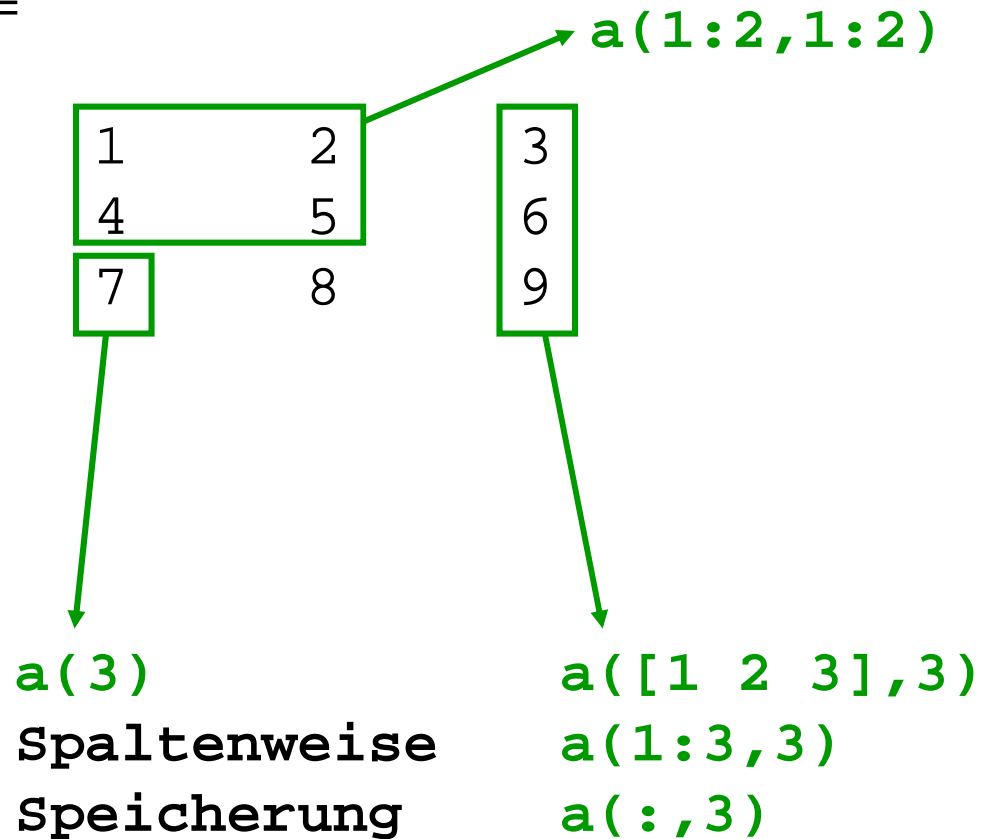
```
ans =
```

```
    0.2000    0.3333
    0.4286    0.5000
```

2.3 Matrizen indizieren

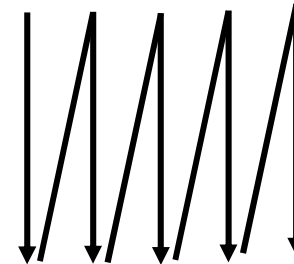
```
>> a = [1 2 3; 4 5 6; 7 8 9]
```

a =



Matrixindizierung:
„Zuerst Zeilenindex,
dann Spaltenindex“

Speicherung:
spaltenweise



3. Boolesche Indizierung

Boolesche

Operatoren:

== equal

> greater

< less

~ not

& and

| or

isempty()

isfinite()

any()

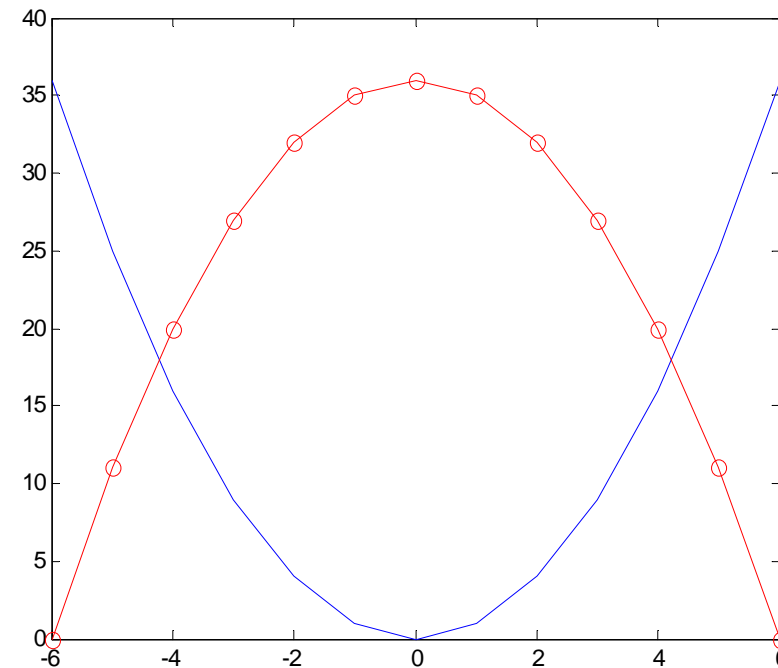
all(), ...

```
>> values = [-4 10 20 NaN -8 Inf 50]
values =
    -4    10    20   NaN    -8   Inf    50
>> positives = values>=0
positives =
     0     1     1     0     0     1     1
>> all_positives = all(values>=0)
all_positives =
     0
>> all_positives = all(values([2 3 6 7]) >=0)
all_positives =
     1
>> positives_finite = (values>0)&(isfinite(values))
positives_finite =
     0     1     1     0     0     0     1
>> pos_fin_values = values(positives_finite)
pos_fin_values =
    10    20    50
```



4. Erstellen von Graphen mit plot

```
>> x = -6:6;  
>> y = x.^2;  
>> plot(x,y,'-')  
>> hold on  
>> plot(x,36-y,'r-o');  
>> hold off  
>>  
>> plot(x, y, x, 36-y, 'r-o');
```



- Der `.` Operator bei `x.^2` signalisiert eine elementweise Operation, d.h. jedes Element von `x` wird quadriert.
- **hold on** friert die Ausgabe bis **hold off** ein, so daß mehrere Kurven in einer Ausgabe (figure) definiert werden können.



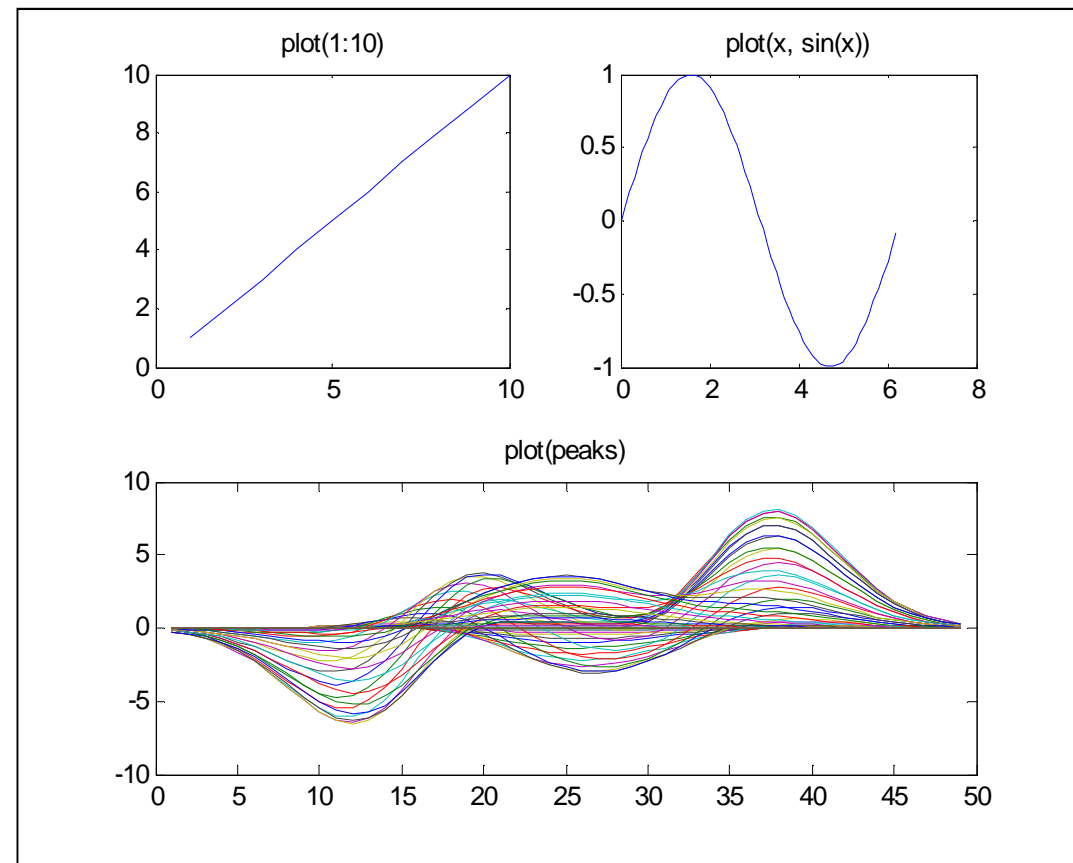
4. Erstellen von mehreren Graphen in Subplots

Ausgabe von mehreren Ausgabeplots in einem Fenster:

`subplot(#rows, #cols, index)`



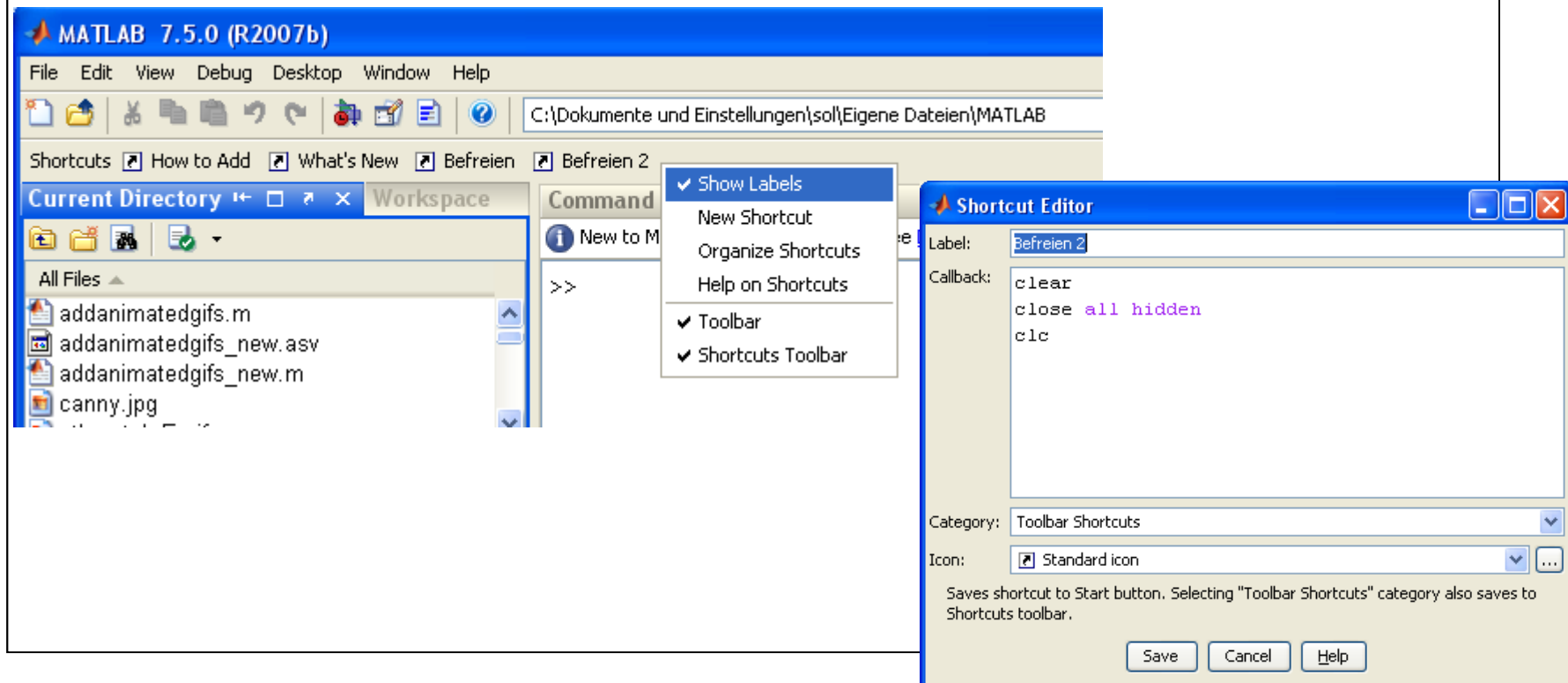
```
>> subplot(2,2,1);  
>> plot(1:10);  
>> title('plot(1:10)');  
>> subplot(2,2,2);  
>> x = 0:0.1:2*pi;  
>> plot(x, sin(x));  
>> title('plot(x, sin(x))');  
>> subplot(2,2,3:4);  
>> plot(peaks);  
>> title('plot(peaks)');
```



Startposition einstellen - Aufräumen

- ```
>> clear
>> close all hidden
>> clc

>> clear; close; clc;
```
- clear – Löschen aller lokalen Variablen
  - close – Schließen aller offenen Ausgabefenster (figures)
  - clc – Löschen der Befehle im Befehlsfenster



## 5. Bilder und Farbtabelle erzeugen



```
>> I = magic(5);
```

```
>> map = hsv(16)
```

```
map =
```

|        |        |        |    |
|--------|--------|--------|----|
| 1.0000 | 0      | 0      | 1  |
| 1.0000 | 0.3750 | 0      | 2  |
| 1.0000 | 0.7500 | 0      | 3  |
| 0.8750 | 1.0000 | 0      | 4  |
| 0.5000 | 1.0000 | 0      | 5  |
| 0.1250 | 1.0000 | 0      | 6  |
| 0      | 1.0000 | 0.2500 | 7  |
| 0      | 1.0000 | 0.6250 | 8  |
| 0      | 1.0000 | 1.0000 | 9  |
| 0      | 0.6250 | 1.0000 | 10 |
| 0      | 0.2500 | 1.0000 | 11 |
| 0.1250 | 0      | 1.0000 | 12 |
| 0.5000 | 0      | 1.0000 | 13 |
| 0.8750 | 0      | 1.0000 | 14 |
| 1.0000 | 0      | 0.7500 | 15 |
| 1.0000 | 0      | 0.3750 | 16 |

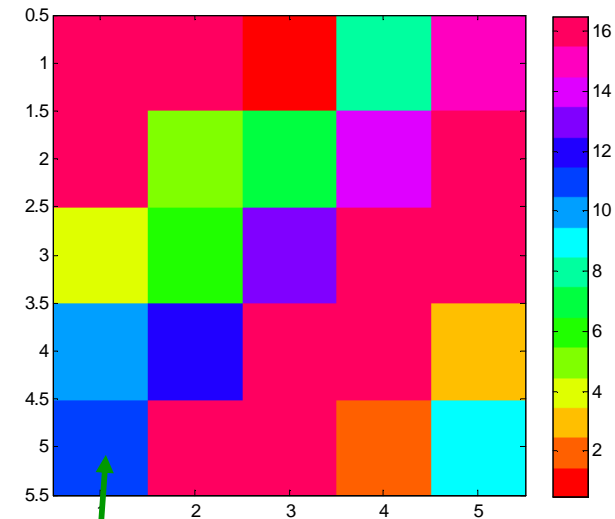
```
>> image(I)
```

```
>> map = hsv(16);
```

```
>> colormap(map)
```

```
>> colorbar;
```

map =  
Farbtabelle



```
>> I = magic(5)
```

```
I =
```

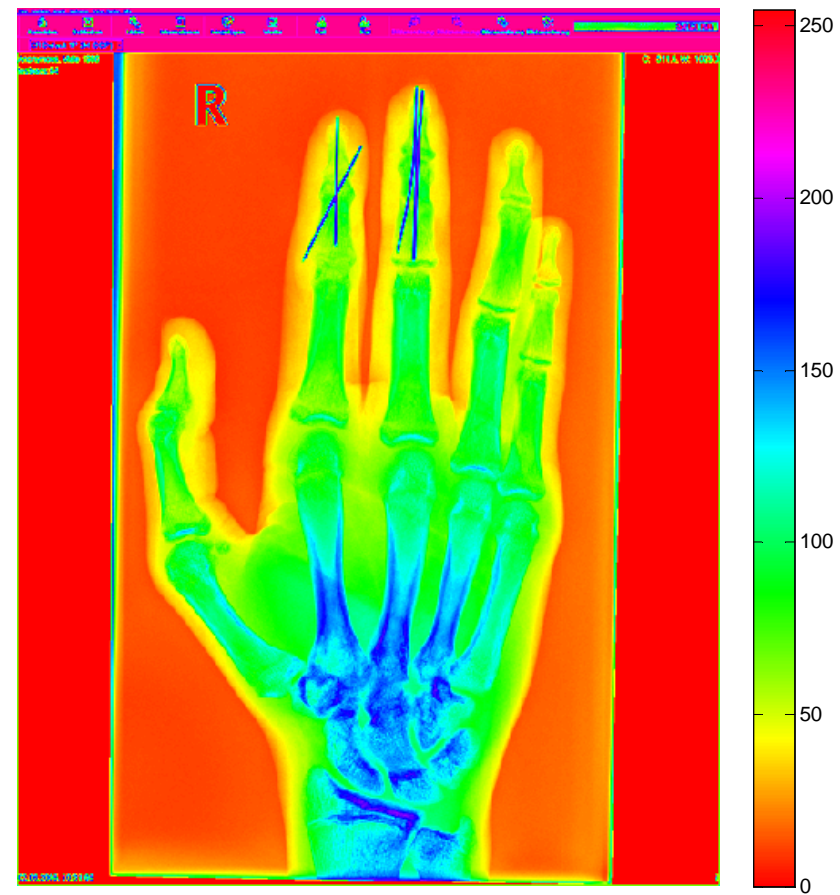
|    |    |    |    |    |
|----|----|----|----|----|
| 17 | 24 | 1  | 8  | 15 |
| 23 | 5  | 7  | 14 | 16 |
| 4  | 6  | 13 | 20 | 22 |
| 10 | 12 | 19 | 21 | 3  |
| 11 | 18 | 25 | 2  | 9  |

Farbindex 11

# Beispiel: Röntgenbild mit verschiedenen Farbtabellen



Farbtabelle: gray



Farbtabelle: hsv



## 6. M-Dateien: Skripte und Funktionen

- M-Dateien: Ablauf einer Folge von Matlab-Befehlen im Hintergrundspeicher
- 2 verschiedene Arten von M-Dateien: Skript oder Funktion
- M-Datei ausführen durch Eingabe des Filenamens ohne .m im Command-Window

### Skript:

Zeile-für-Zeile-Ausführung

Keine Ein- und keine  
Ausgabeparameter

Verwendet Variablen aus dem  
aktuellen Arbeitsbereich

### Funktionen:

Startet mit dem Schlüsselwort  
**function**

Ermöglicht Ein- und  
Ausgabeparameter

Erstellt lokale Variablen im  
eigenen Funktions-  
Arbeitsbereich



## 6.1 Aufbau einer M-Datei Funktion

```
function y = myfunc(a,b)
% Diese Funktion addiert die Eingabeparameter a und b
% und liefert y als Rückgabewert zurück
y = a + b;
end
```

|                   |                    |
|-------------------|--------------------|
| Rückgabeparameter | y                  |
| Eingabeparameter  | a und b            |
| Kommentar         | % oder %%          |
| Funktionsname     | myfunc             |
| Dateiname         | myfunc.m           |
| Aufruf            | >> y = myfunc(a,b) |





## 6.2 Mehrere Rückgabewerte

```
function [y1, y2] = myfunc(a,b)
%% Hier beginnt ein neuer Block
% Funktion mit mehreren Rückgabeparametern
y1 = a + b;
y2 = a*b;
```

```
%% Neuer Block
```

- Gespeichert in myfunc.m
- Mehrere Rückgabewerte werden als Vektor zurückgegeben

Aufruf:

```
>> [a1, a2] = myfunc(4,5)
a1 =
 9
a2 =
 20
```



# Wichtige Befehle



```
>> dir
. linker Bildschirm.png
.. myfunc.m
Thumbs.db rechter Bildschirm.png
```

➤ dir  
Inhalt des aktuellen  
Arbeitsverzeichnisses

```
>> pwd
ans =
D:\BV\Bilder\Beispiele
```

➤ pwd  
Pfad des aktuellen  
Arbeitsverzeichnisses

```
>> type myfunc

function [y1, y2] = myfunc(a,b)
%% Hier beginnt ein neuer Block
% Funktion mit mehreren Rückgabeparametern
y1 = a + b;
y2 = a*b;
```

➤ type  
Inhalt einer Datei

```
%% Neuer Block
```

```
>> which myfunc
D:\BV\Bilder\Beispiele\myfunc.m
```

➤ which  
Pfad einer Datei

```
>> whos
 Name Size Bytes Class Attributes
 ans 1x22 44 char
```

➤ whos  
listet die lokalen Variablen  
des Matlab-Arbeits-  
speichers

## 6.3 Unterfunktionen definieren

```
%% Funktion mit Unterfunktion myadd
function y = myfunc(a,b)
y = myadd(a,b);

%% Unterfunktion
function a = myadd(c,d)
a = c + d;
```



## 6.4 Eingebettete Funktionen (Nested functions)

```
function A(x, y) % Primary function
B(x, y);
D(y);

 function B(x, y) % Nested in A
 C(x);
 D(y);

 function C(x) % Nested in B
 D(x);
 end
 end

 function D(x) % Nested in A
 E(x);

 function E(x) % Nested in D
 ...
 end
 end
end
```

## 6.4 Beispiel: Eingebettete Funktionen

```
function first()
```

```
c = 5;
```

```
d = 5;
```

```
second()
```

```
 function second()
```

```
 a = c + d;
```

```
 end
```

```
end
```

Spart Übergabeparameter (Bilder)



# 7. Datenstrukturen



```
>> student.name = 'Tim';
>> student.noten = [1.0 1.0];
>> student(2).name = 'Andre';
>> student(2).grades = [1.0 1.0];

>> student
student =
1x2 struct array with fields:
 name
 noten
 grades

>> student(2).name
ans =
Andre

>> student(1).noten(1)
ans =
 1
```

## Syntax:

```
>> struct_name(record#).field_name = data
```

## Datentypen:

string, double, sparse, cell, structures  
scalars, vector, 2-D, N-D

## Dynamische Speicherallokation

# Beispiel: Datentyp cell



```
>> A = {'hallo', 'welt', pi, rand(3)}
A =
 'hallo' 'welt' [3.1416] [3x3 double]
```

```
>> str = A{1, 1}
str =
hallo
```

```
>> str = A{1, 2}
str =
welt
```

```
>> str = A{1, 4}
str =
```

```
 0.8147 0.9134 0.2785
 0.9058 0.6324 0.5469
 0.1270 0.0975 0.9575
```

```
>>
```

cell ist in { } eingebettet und  
kann verschiedene Daten-  
typen zu einem Vektor  
kombinieren

## 7. Beispiel: Datenstrukturen

```
>> prof = struct('name',{'Scholl','Ossmann'}, ...
>> 'courses',{{'GDV' 'BV'},{'ARBK','Datenkompression'}});

>> prof(1).courses(1)
ans =
 'GDV'

>> prof(2).courses(2)
ans =
 'Datenkompression'
```







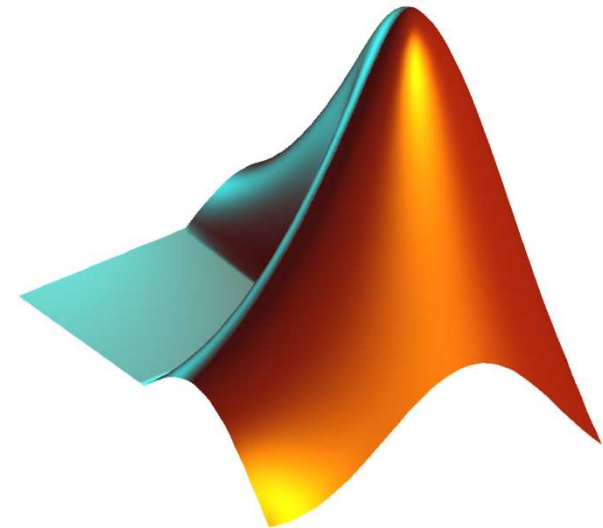
# Vielen Dank für die Aufmerksamkeit !

FH Aachen  
Fachbereich Elektrotechnik und Informationstechnik  
Prof. Ingrid Scholl  
Eupenerstr. 70  
52066 Aachen  
T +49. 241. 6009 52177  
scholl@fh-aachen.de  
www.fh-aachen.de



# Image Processing Toolbox Bilder in MATLAB

Prof. Ingrid Scholl  
Bildverarbeitung WS 2013/2014



# Agenda

---

1. Einlesen und Speichern von Bildern in MATLAB
2. Dateiformate für Bilder
3. Bilder in MATLAB:
  - > Bildkoordinatensystem
  - > Datentypen für Bilder
4. Bildtypen in der Toolbox
5. Bilder anzeigen
6. Konvertierung zwischen den Bildtypen
7. Arbeiten mit Bildsequenzen
8. Mathematische Operationen mit Bildern





- Nennen der 4 Bildtypen aus MATLAB
- Importieren von Bilddaten in MATLAB durch Verwendung von Bildimportierungsfunktionen
- Anzeigen von Bildern in MATLAB
- Konvertieren von Bildern in ein anderes Format
- Bildinformationen zu einem Bild abrufen

# Einlesen und Speichern von Bilddateien



```
>> I = imread('Bild1.jpg');
>> imwrite(I,'NeuesBild.jpg');
>> imwrite(I,'Temp2','jpg');
>> imfinfo('Bild1.jpg')
```

MATLAB verfügt über verschiedene Funktionen, um Bilddaten zu importieren und zu exportieren:

**imfinfo** – liefert Bildeigenschaften

**imread** – liest ein Bild von einer Datei ein

**imwrite** – speichert eine Bildvariable in eine Datei

Innerhalb einer GUI:

**uiimport** oder **uigetfile** – startet ein Dialogfenster zur Eingabe

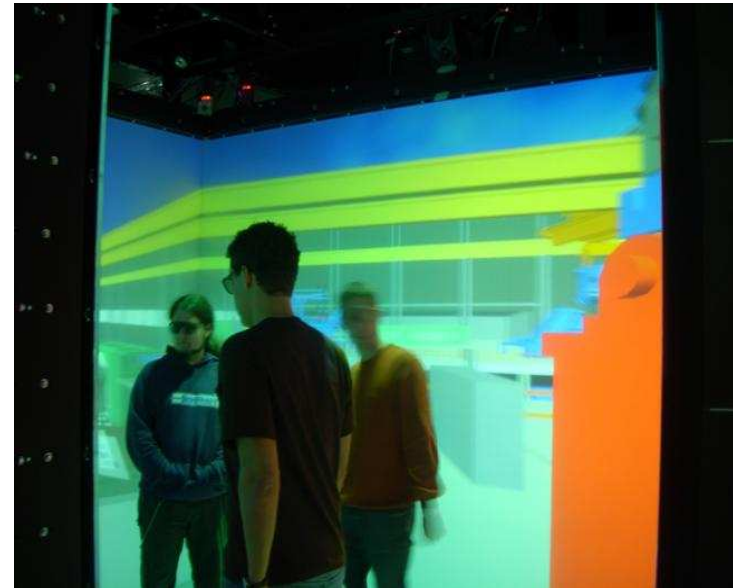
```
>> [filename, pathname] = ...
 uigetfile({'*.jpg'; '*.tif'; '*.gif'; '*.*'}, 'File Selector');
```

# Bildinformationen

```
>> imfinfo('bild1.jpg') Dateiname
```

```
ans =
```

```
 Filename: 'bild1.jpg'
 FileModDate: '07-Apr-2008 22:00:55'
 FileSize: 26894
 Format: 'jpg'
 FormatVersion: ''
 Width: 648
 Height: 486
 BitDepth: 24
 ColorType: 'truecolor'
 FormatSignature: ''
 NumberOfSamples: 3
 CodingMethod: 'Huffman'
 CodingProcess: 'Sequential'
 Comment: {}
```



# Einlesen und Speichern von DICOM-Dateien

DICOM steht für Digital Imaging and Communications in Medicine  
MATLAB verfügt über spez. Funktionen zum Einlesen und Speichern von DICOM-Dateien:

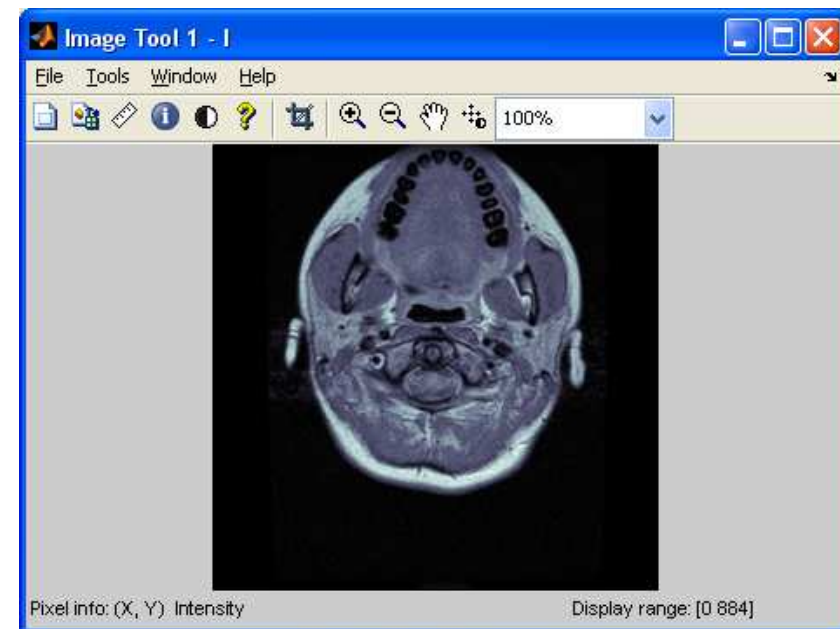
**dicomread** - Einlesen eines Dicom-Bildes  
**dicominfo** - Informationen zum Dicom-Bild  
**dicomwrite** - Schreiben eines Dicom-Bildes  
**dicomdict** - zum Setzen des aktiven Dicom-Dateiverzeichnisses

## Beispiel:

```
>> info = dicominfo('brain_001.dcm')
>> I = dicomread('brain_001.dcm');
>> whos
```

| Name | Size    | Bytes  | Class  |
|------|---------|--------|--------|
| I    | 256x256 | 131072 | int16  |
| info | 1x1     | 16674  | struct |

```
>> imtool(I,[]);
```



# Dateiformate für Bilder

```
>> doc imread
```

| Format      | Formatname                       | Varianten                                                                                                                                                                                   | Bildtyp                          |
|-------------|----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------|
| bmp         | Windows bitmap                   | unkomprimiert Bilder: 1-, 4-, 8-, 16-, 24-, 32-Bits;<br>RLE (run length encoded) Bilder: 4-, 8- Bits                                                                                        | RGB<br>Indexbild                 |
| gif         | Graphics interchange format      | 1- bis 8-Bit Bilder                                                                                                                                                                         | Indexbild                        |
| jpg<br>jpeg | Joint Photographic Experts Group | Grauwertbilder mit 8- oder 12-Bit bei verlustbehafteter Kompression; 8-, 12-, 16-Bit<br>Grauwertbilder mit verlustfreier Kompression                                                        | RGB<br>Grauwertbild              |
| tif<br>tiff | Tagged Image File Format         | 1-, 8-, 24-Bit unkomprimierte Bilder;<br>1-, 8-, 24-Bit Bilder mit „packbits“ Kompression,<br>1-Bit Bilder mit CCITT Kompression, 16-Bit<br>Grauwertbild, 16-Bit Indexbild, 48-Bit RGB-Bild | RGB<br>Indexbild<br>Grauwertbild |
| png         | Portable Network graphics        | 1-, 2-, 4-, 8-, 16-Bit Grauwertbilder, 8- und 16-Bit<br>Indexbilder und 24- oder 48-Bit RGB Bilder                                                                                          | RGB<br>Indexbild<br>Grauwertbild |

```
A = imread(filename, fmt)
[X, map] = imread(...)
[...] = imread(filename)
[...] = imread(URL,...)
```



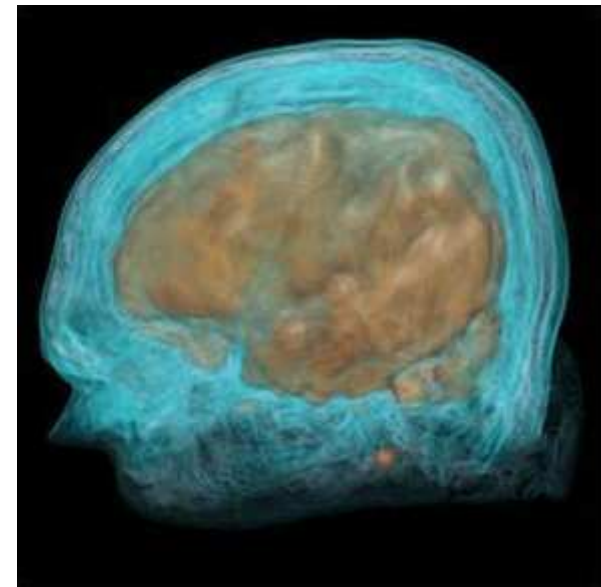
- Basisdatenstruktur für Bilder ist in MATLAB ein 2D Array bzw. eine Matrix
- Jedes Matrixelement entspricht einem Pixel aus dem Bild (*pixel = picture element*)
- Jedes Pixel repräsentiert einen Farbwert bzw. Intensitätswert

```
>> I = imread('gehirn.jpg');
```

```
>> whos
```

| Name | Size      | Bytes  | Class | Attributes |
|------|-----------|--------|-------|------------|
| I    | 295x295x3 | 261075 | uint8 |            |

```
>> figure, imshow(I);
```



# Bildkoordinatensystem in MATLAB



```
>> I(1:5,1:5)
```

ans =

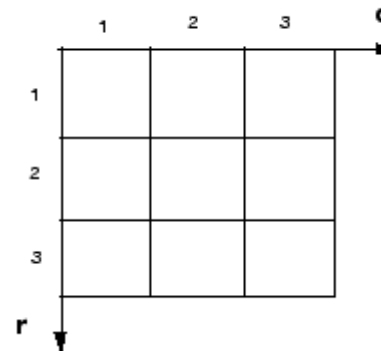
|     |     |     |     |     |
|-----|-----|-----|-----|-----|
| 255 | 239 | 255 | 247 | 255 |
| 233 | 23  | 0   | 0   | 8   |
| 255 | 0   | 8   | 10  | 7   |
| 246 | 0   | 21  | 0   | 0   |
| 255 | 6   | 0   | 0   | 3   |

```
>> I(4,3)
```

ans =

21

The Pixel Coordinate System



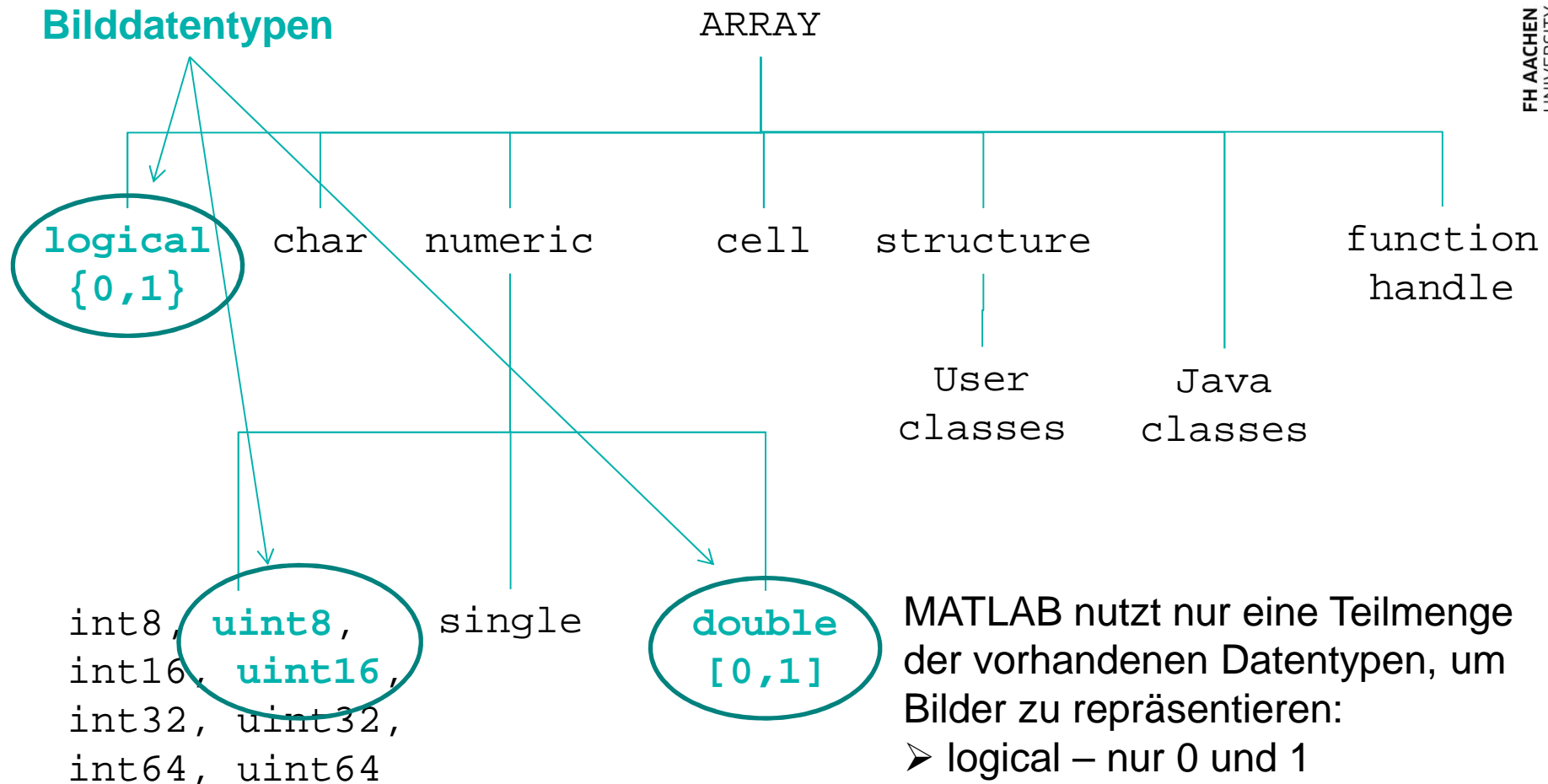
- Bild besteht aus Pixelkoordinaten
- Bild ist ein Gitter von diskreten Pixeln, die von oben nach unten und von links nach rechts angeordnet sind
- Bildkoordinaten sind Integerwerte im Bereich von 1 bis Zeilen- bzw. Spaltenanzahl

# MATLAB Datentypen

```
>> help datatypes
```



## Bilddatentypen



MATLAB nutzt nur eine Teilmenge der vorhandenen Datentypen, um Bilder zu repräsentieren:

- logical – nur 0 und 1
- uint8, uint16 –  $[0, 2^8-1]$  oder  $[0, 2^{16}-1]$
- double – normiert  $[0,1]$

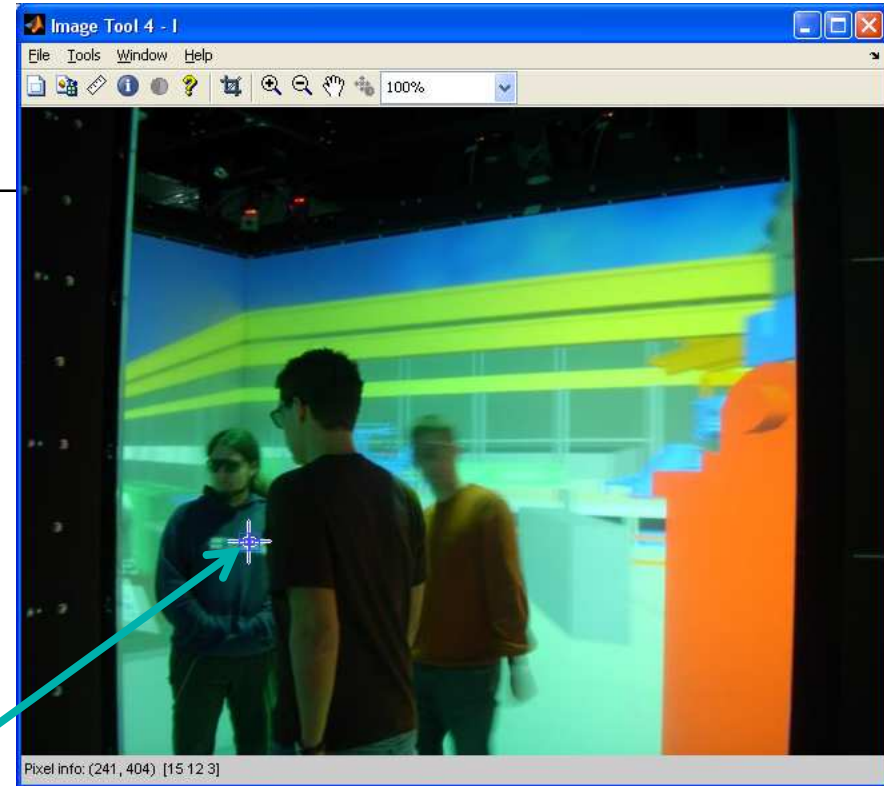
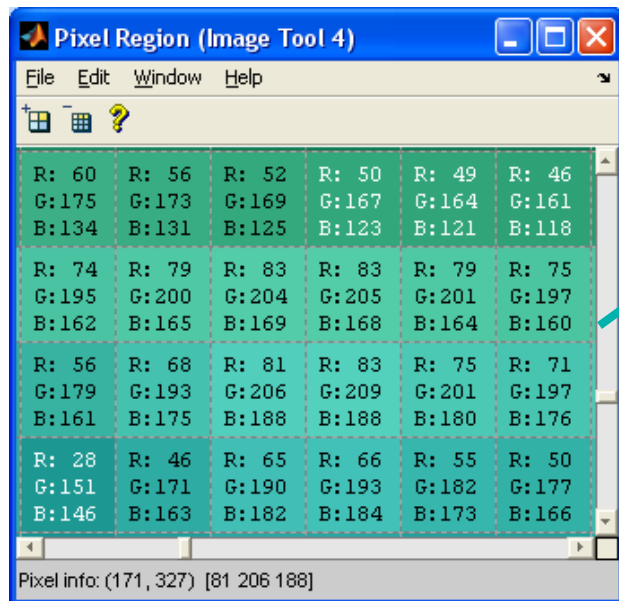


Gegeben: Bild mit m Zeilen und n Spalten

1. Binärbilder  $\{0,1\}$
2. Grauwertbilder  
Intensitätsbilder  $[0,255]$  mit uint8 (256 Grauwerte)  
oder uint16 ( $2^{16}$  Grauwerte)
3. Indexbilder  
Pixelwert im Bild entspricht einem  
Farbindex aus einer Farbtabelle  
 $m \times n$  mit uint8 oder uint16
4. RGB-Bilder  
 $m \times n \times 3$  Kanäle mit je uint8

# RGB Bild

```
% RGB jpg-Bild einlesen
Irgb = imread('bild1.jpg');
% Bild anzeigen
figure; imshow(Irgb);
```



Ein RGB-Bild wird in MATLAB als ein  $m \times n \times 3$  Array gespeichert, d.h. für jede Farbe Rot, Grün und Blau wird ein  $m \times n$  Array definiert.

```
>> Ired = Irgb(:,:,1);
>> Igreen = Irgb(:,:,2);
>> Iblue = Irgb(:,:,3);
```

# Grauwertbild Intensitätsbild



Datentyp uint8  
[0,255]

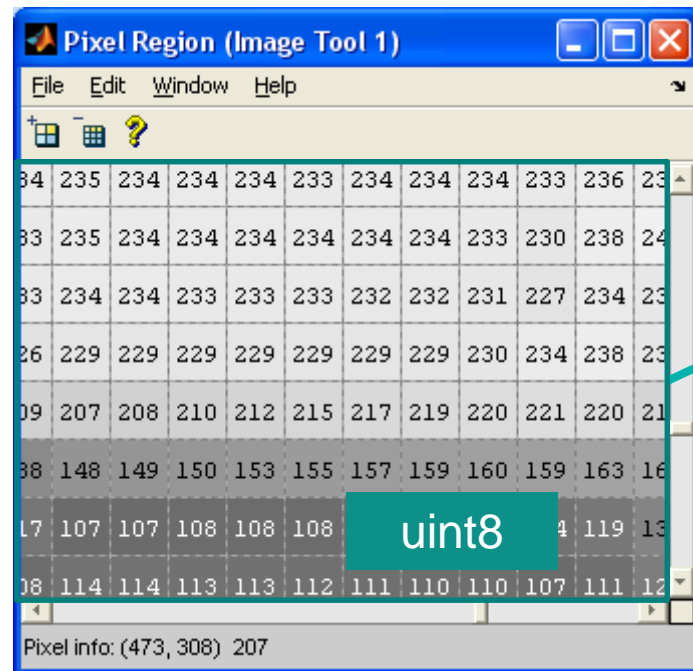


Datentyp double  
[0,1]

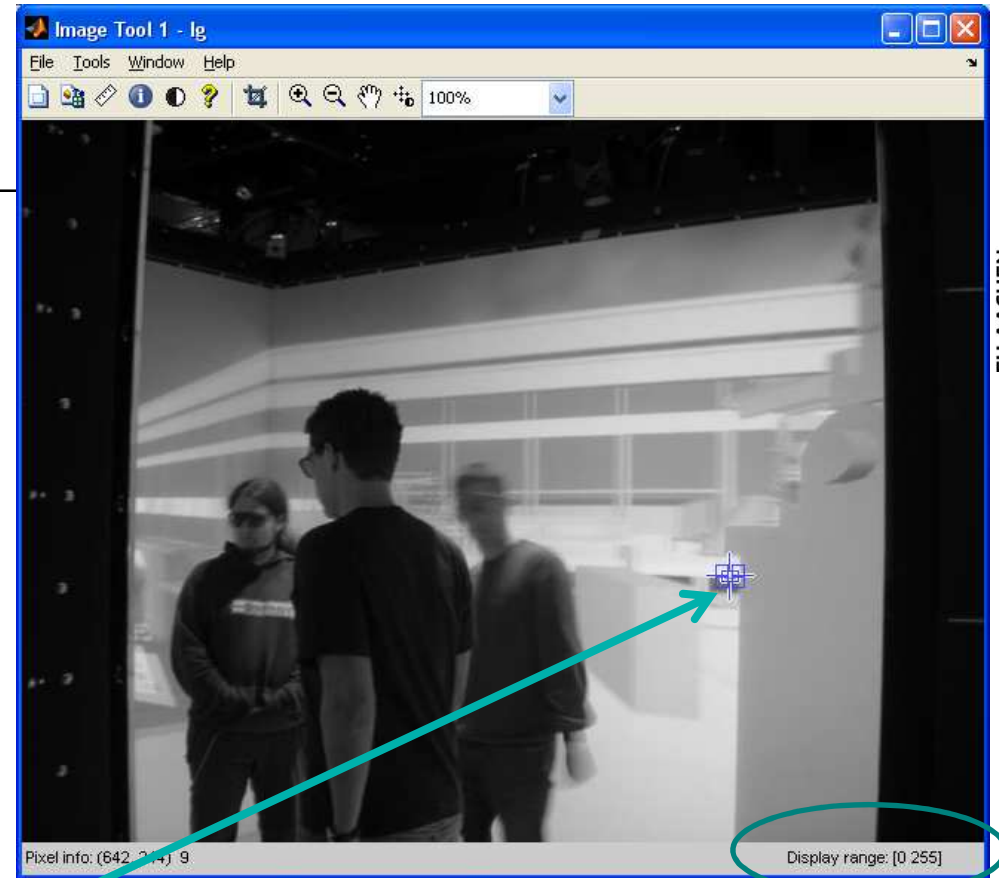


# Grauwertbild Intensitätsbild

```
% Grauwertbild aus
% RGB-Bild erzeugen
Igray = rgb2gray(Irgb);
% Bild anzeigen
imtool(Igray);
```

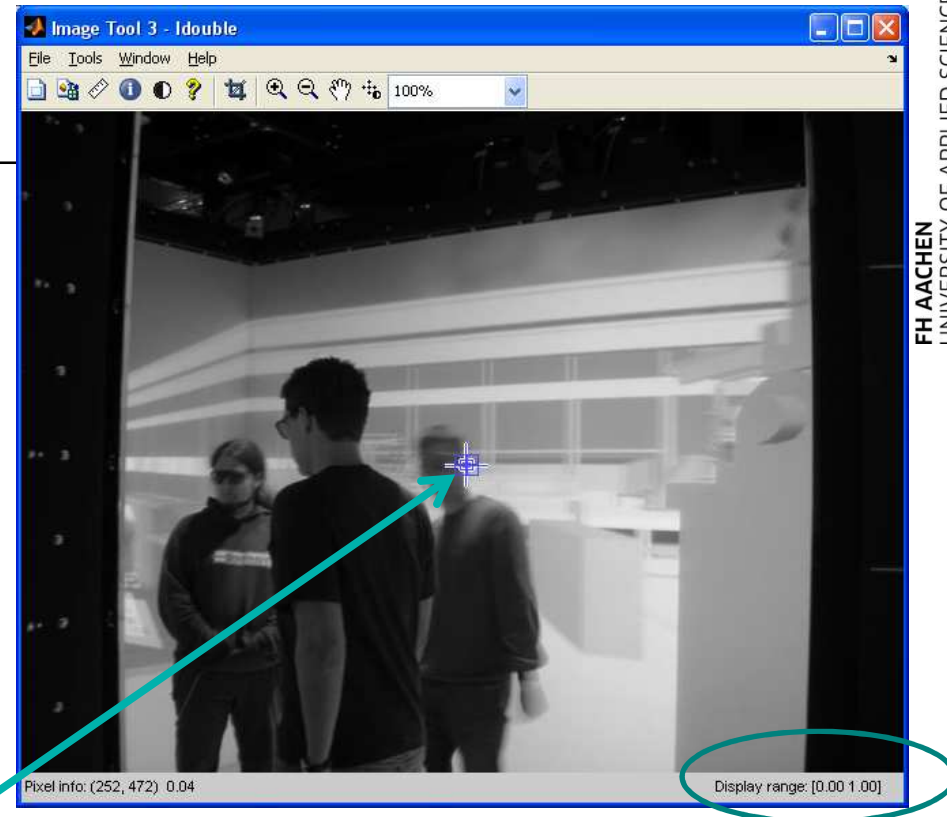
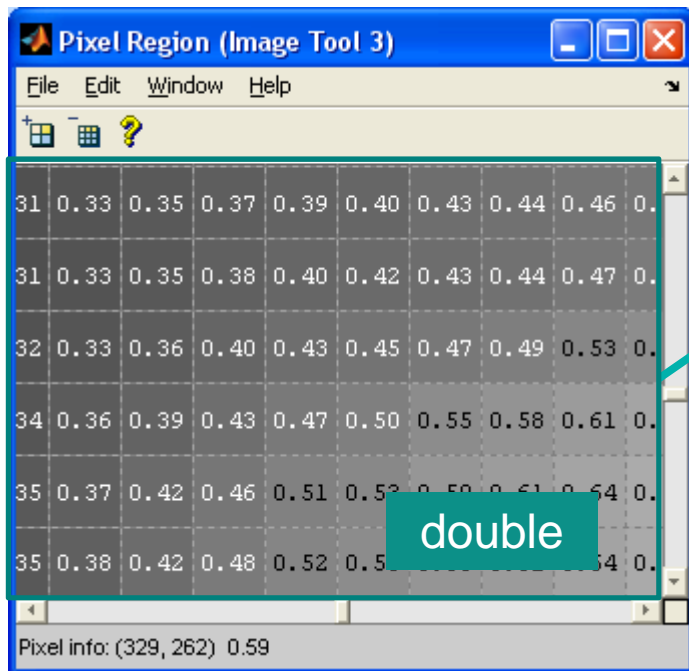


- Besteht aus einer Matrix Igray deren Pixelwerte Intensitäten in einem Wertebereich definieren: normiert zwischen  $[0,1]$  oder diskret zwischen  $[0, 2^8-1]$  bei Datentyp uint8 sind



# Grauwertbild Intensitätsbild

```
% Grauwertbild aus
% RGB-Bild erzeugen
Idouble = im2double(Igray)
% Bild anzeigen
imtool(Idouble);
```



```
>> whos
```

| Name           | Size           | Bytes          | Class         |
|----------------|----------------|----------------|---------------|
| <b>Idouble</b> | <b>486x648</b> | <b>2519424</b> | <b>double</b> |
| Igray          | 486x648        | 314928         | uint8         |
| Irgb           | 486x648x3      | 944784         | uint8         |



# Binärbild

```
% optimaler Schwellwert
T = graythresh(Irgb);
% Binarisierung
Ibinary = im2bw(Irgb, T);
```

- Die Pixelwerte in einem Binärbild bestehen nur aus 2 Werten: {0,1}  
0 für aus und  
1 für an
- RGB-Bild wird zunächst in ein Grauwertbild umgewandelt, dann wird mit dem Alg. von Otsu ein optimaler Schwellwert T bestimmt:

Teste:

```
>> Idouble = rand(100)
>> whos
>> image(Idouble)
>> map = gray(2)
>> colormap(map)
>> colorbar
```



$$f(x, y) \leq T \Rightarrow f_{binary}(x, y) = 0$$

$$f(x, y) > T \Rightarrow f_{binary}(x, y) = 1$$

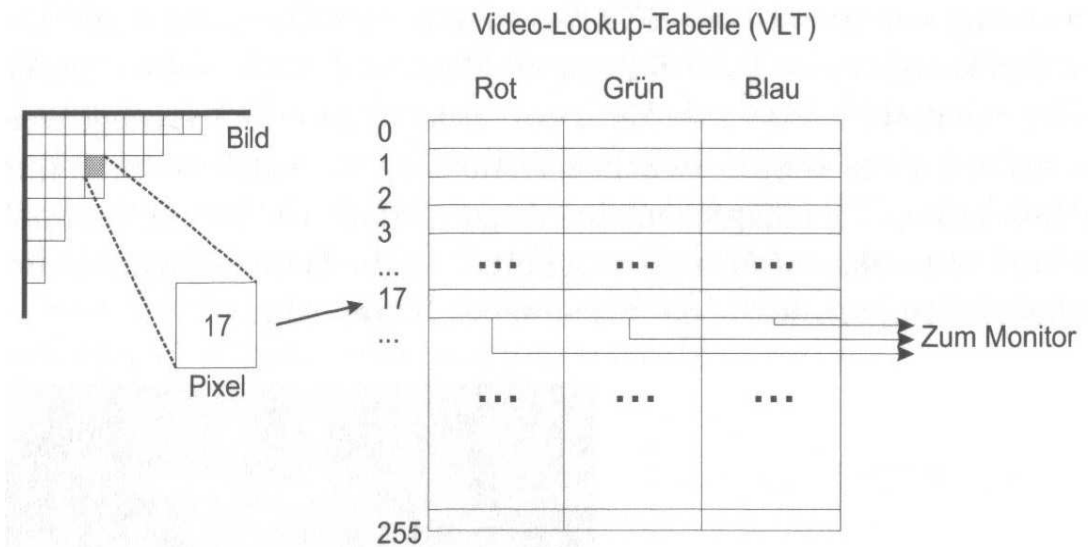
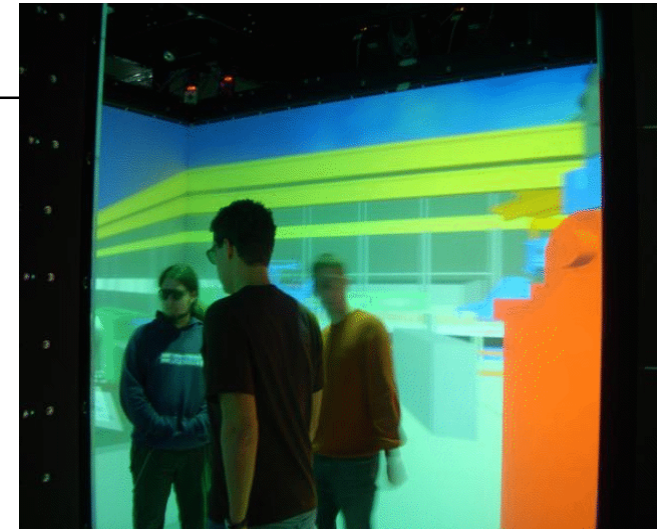


# Index-Bild

```
% n ist Integer, Anzahl der Farben
% und somit Anzahl Zeilen der map
[X,map] = rgb2ind(Irgb, n);

% Rückumwandlung zum RGB-Bild
rgb_image = ind2rgb(X, map);
```

- Ein Indexbild X besteht aus einer Matrix mit den Pixelwerten und einer Farbtabelle (VLT) map mit 3 Vektoren mit n Einträgen für Rot, Grün und Blau
- Die Pixelwerte des Indexbildes X sind Indizes auf die Farbtabelle-einträge



# Indexbild mit 64 Farben

```
>> [Iindex, map] = rgb2ind(Irgb,64);
>> figure, imshow(Iindex, map);
>> colorbar
>> whos
```

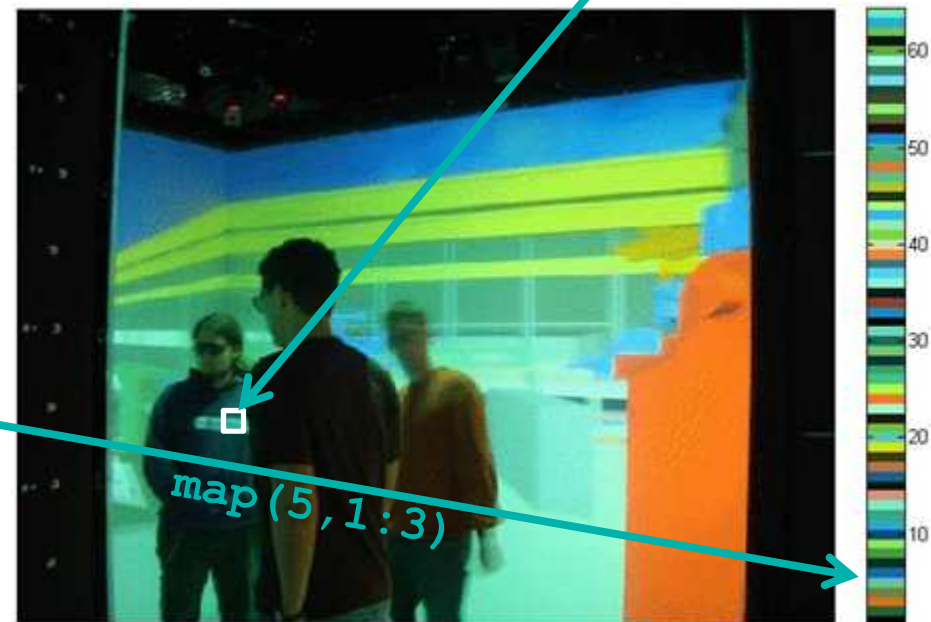
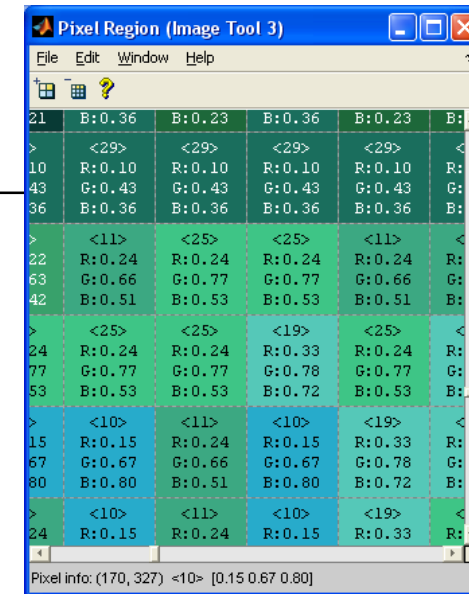
| Name   | Size      | Bytes  | Class  |
|--------|-----------|--------|--------|
| Irgb   | 486x648x3 | 944784 | uint8  |
| Iindex | 486x648   | 314928 | uint8  |
| map    | 64x3      | 1536   | double |

```
>> map
```

← Farbtabelle

```
map =
```

| Rot    | Grün   | Blau   |   |
|--------|--------|--------|---|
| 0.0157 | 0.0157 | 0.0157 | 1 |
| 0.1137 | 0.4078 | 0.2275 | 2 |
| 0.8941 | 0.4235 | 0.0980 | 3 |
| 0.4235 | 0.5333 | 0.2353 | 4 |
| 0.4157 | 0.7569 | 0.6275 | 5 |
| 0.0471 | 0.4235 | 0.7373 |   |
| 0.0314 | 0.2000 | 0.1020 |   |
| 0.2588 | 0.6196 | 0.2745 |   |
| 0.5608 | 0.9059 | 0.2863 |   |
| ...    | ...    | ...    |   |



# Matlab Beispiel: Bildtypen

```
% RGB Bild einlesen
Irgb = imread('bild1.jpg');
% Bild anzeigen
figure; imshow(Irgb);
% optimaler Schwellwert
level = graythresh(Irgb);
% Binarisierung
Ibinary = im2bw(Irgb,level);
figure; imshow(Ibinary);
% Grauwertbild
Igray = rgb2gray(Irgb);
figure; imshow(Igray);
% Indexbild erzeugen mit 256 Farben
[Iindexed, map] = rgb2ind(Irgb, 256);
figure; imshow(Iindexed, map);
```

```
>> whos
```

| Name     | Size      | Bytes  | Class   |
|----------|-----------|--------|---------|
| Ibinary  | 486x648   | 314928 | logical |
| Igray    | 486x648   | 314928 | uint8   |
| Iindexed | 486x648   | 314928 | uint8   |
| level    | 1x1       | 8      | double  |
| map      | 256x3     | 6144   | double  |
| rgb      | 486x648x3 | 944784 | uint8   |

# Übersicht: Darstellung von Bildern in MATLAB



| Bildtyp                       | Indexbild                                                                                                                              | Intensitäts-<br>bild                                                                                                                   | Binärbild                                                        | RGB-Bild                                                                                                       |
|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| double<br>Daten               | (M x N)-Array mit Integerwerten im Bereich von [1,P].<br><br>Farbtabelle ist ein (P x 3)-Array von double-Werten im Bereich von [0,1]. | (M x N)-Array von double Werten im Bereich von [0,1].                                                                                  | (M x N)-Array mit logischen Werten, d.h. nur Werte mit 0 oder 1. | (M x N x 3)-Array von double-Werten im Bereich [0,1].                                                          |
| uint8 oder<br>uint16<br>Daten | (M x N)-Array mit Integerwerten im Bereich von [0,P-1], wobei $P \leq 2^8$ (256) bei uint8 und $P \leq 2^{16}$ (65536) bei uint16.     | (M x N)-Array von „unsigned 8-Bit“ Integerwerten im Bereich von [0,255] bzw. „unsigned 16-Bit“ Integerwerten im Bereich von [0,65535]. | <i>Nicht unterstützt</i>                                         | (M x N x 3)-Array von Integerwerten im Bereich von [0,255] bei uint8 bzw. im Bereich von [0,65535] bei uint16. |

```
>> imshow(X, map)
>> colorbar
```

---

MATLAB verfügt nach dem Einlesen von Bildern über Funktionen zur Anzeige des Bildes:

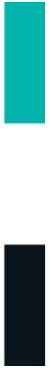
- `imtool`      Anzeige des Bildes in Image Tool
- `imshow`      Anzeige des Bildes in einem Fenster
- `image`        Erstellen und Anzeigen eines image-Objektes
- `imagesc`      Skaliert Bilddaten und zeigt diese an
- `colorbar`     Anzeigen der Farbbalkens
- `colormap`    Setzen der Farbtabelle zum Bild
- `montage`      Anzeigen von mehreren Bildrahmen
- `warp`         Anzeigen von Bildern als texturierte Oberfläche
- `subimage`    Anzeigen von mehreren Bildern in einem einzigen Fenster



# Konvertieren von Bildtypen

>> help im2double

- 
- ind2gray      Indexbild nach Grauwertbild
  - ind2rgb      Indexbild nach Farbbild bzw. RGB
  - gray2ind      Grauwertbild nach Indexbild
  - rgb2gray      RGB-Bild nach Grauwertbild
  - rgb2ind      RGB-Bild nach Indexbild
  
  - mat2gray      Umwandeln einer Matrix zum Grauwertbild
  - im2bw      Umwandeln in ein Binärbild
  - im2double      Bilddaten in double umwandeln
  - im2uint8      Bilddaten in uint8 umwandeln
  - im2uint16      Bilddaten in uint16 umwandeln



# Arbeiten mit Bildsequenzen

Zur Anzeige von mehreren Bildern in einem Fenster dient die **montage**-Funktion der Image Processing Toolbox. Beachte: Die Bilder müssen in einer einzigen Matrix mit 4 Dimensionen kombiniert werden.

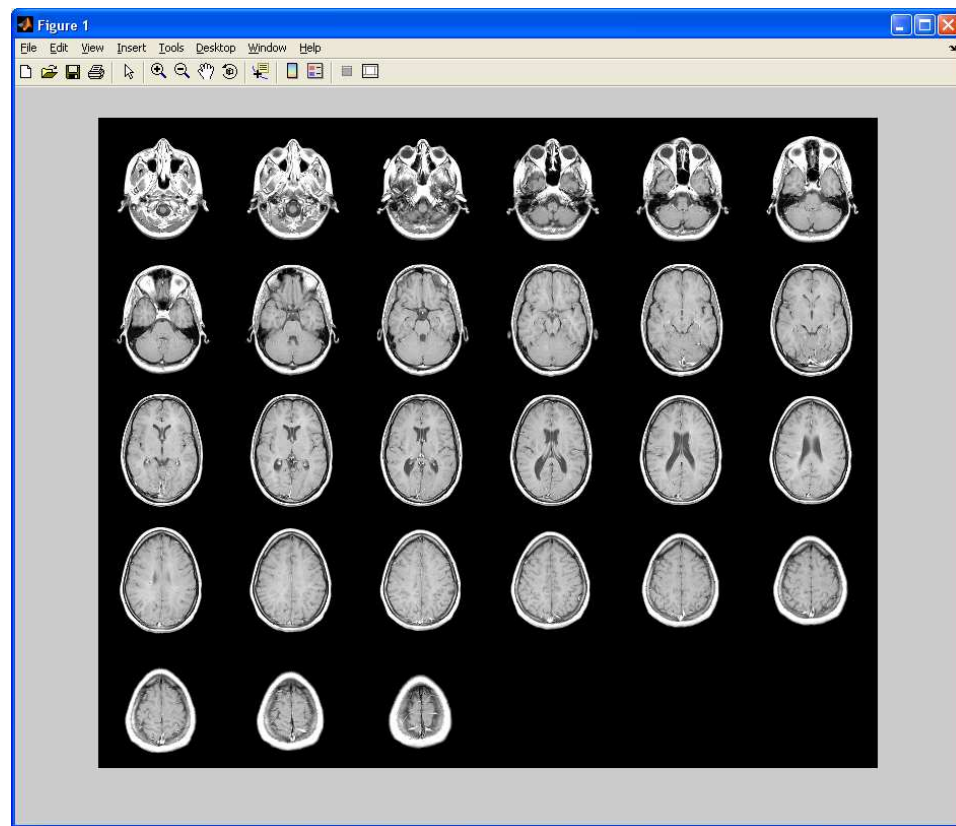
```
>> load mri
>> whos
```

| Name | Size | Bytes  |
|------|------|--------|
| D    | 4-D  | 442368 |
| map  | 89x3 | 2136   |
| siz  | 1x3  | 24     |

```
>> montage(D,map)
```

Zur Animation von mehreren Bildern in einem Fenster verwendet man die Funktionen **immovie** und **movie** wie folgt:

```
>> mov = immovie(D,map);
>> movie(mov,10)
```



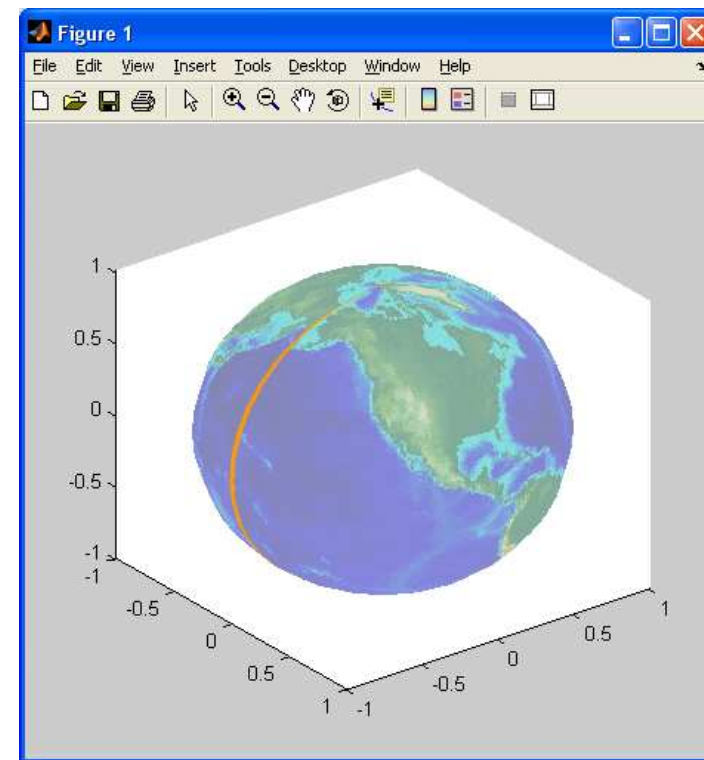
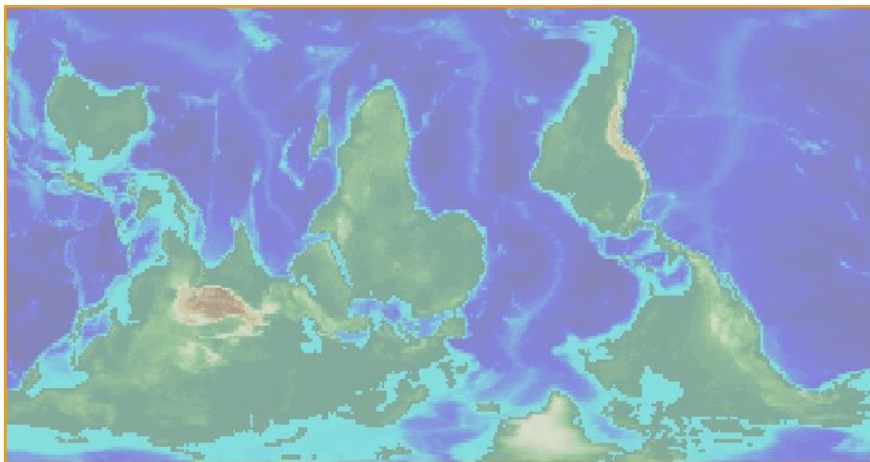


# „Warping Images“

Bilder können auf gekrümmte Oberflächen als Textur „*geklebt*“ werden

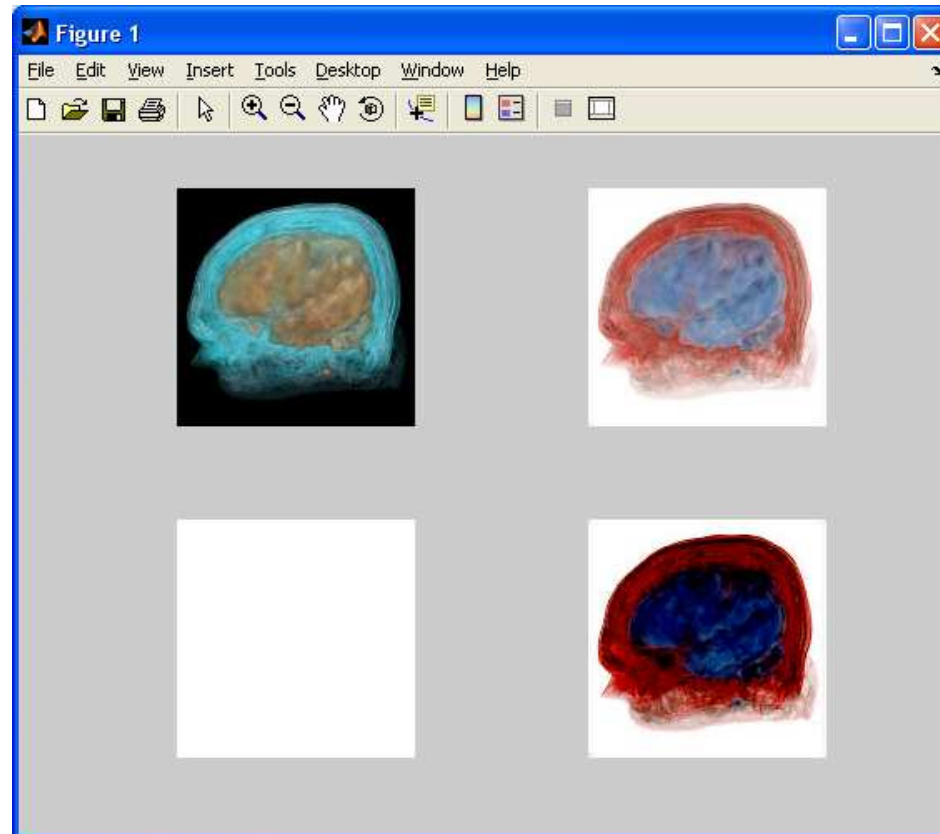
```
>> [E,map] = imread('earth.png');
>> figure, imshow(E,map);
>> [x,y,z] = sphere;
>> warp(x,y,z,E,map)
```

Texturbild: earth.png



# Rechnen mit Bildern

```
Irgb = imread('gehirn.jpg');
Icmy = 255 - Irgb;
Ibinary = (Irgb(:,:,1)>128);
subplot(2,2,1);
imshow(Irgb);
subplot(2,2,2);
imshow(Icmy);
subplot(2,2,3);
imshow(Ibinary);
subplot(2,2,4);
I3 = Icmy + Irgb;
imshow(I3);
I4 = Icmy - Irgb;
subplot(2,2,4);
imshow(I4);
```





- Nennen der 4 Bildtypen aus MATLAB
- Einlesen von Bilddaten in MATLAB durch Verwendung von Bildimportierungsfunktionen
- Anzeigen von Bildern in MATLAB
- Konvertieren von Bildern in ein anderes Format
- Bildinformationen zu einem Bild abrufen

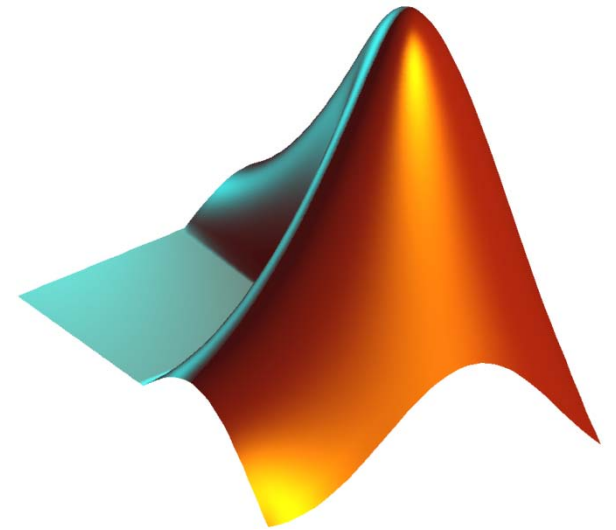


# Vielen Dank für die Aufmerksamkeit !

FH Aachen  
Fachbereich Elektrotechnik und Informationstechnik  
Prof. Ingrid Scholl  
Eupenerstr. 70  
52066 Aachen  
T +49. 241. 6009 52177  
scholl@fh-aachen.de  
www.fh-aachen.de

# Image Processing Toolbox GUI Entwicklung

Prof. Ingrid Scholl  
Bildverarbeitung WS 2011/2012



Mit GUIDE grafische Benutzeroberfläche erzeugen  
(Dateityp: {GUIname}.fig)

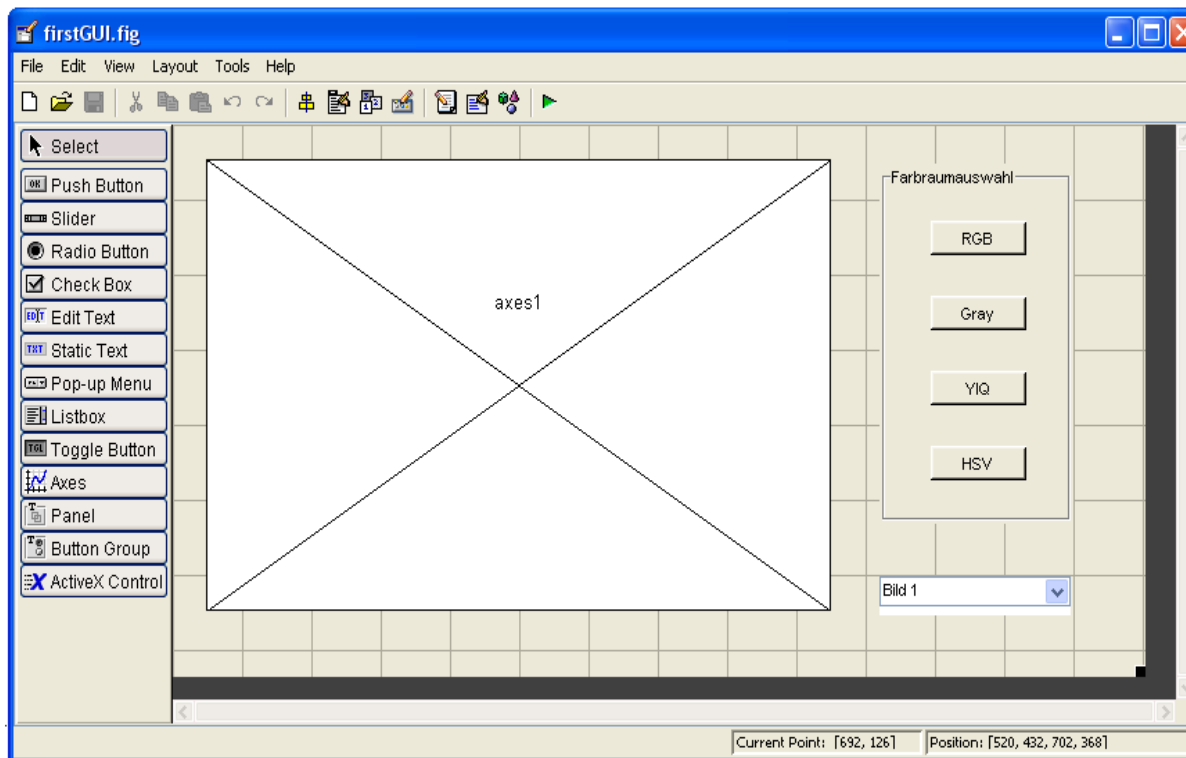
M-Datei wird automatisch erzeugt  
(Dateityp: {GUIname}.m)

Callback Funktionen müssen in der m-Datei  
programmiert werden

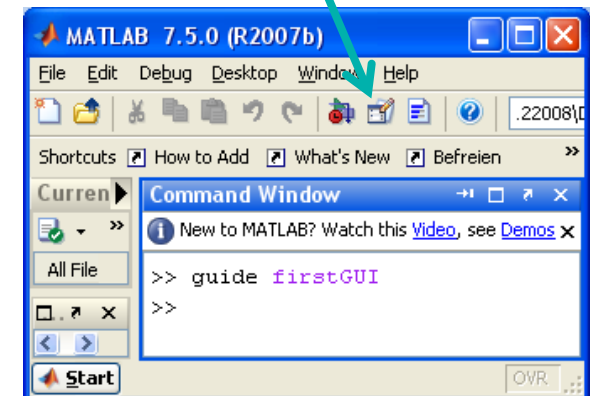
# Beispiel: GUIDE zum Erstellen einer GUI

## *GUIDE -Graphical User Interface Development Environment*

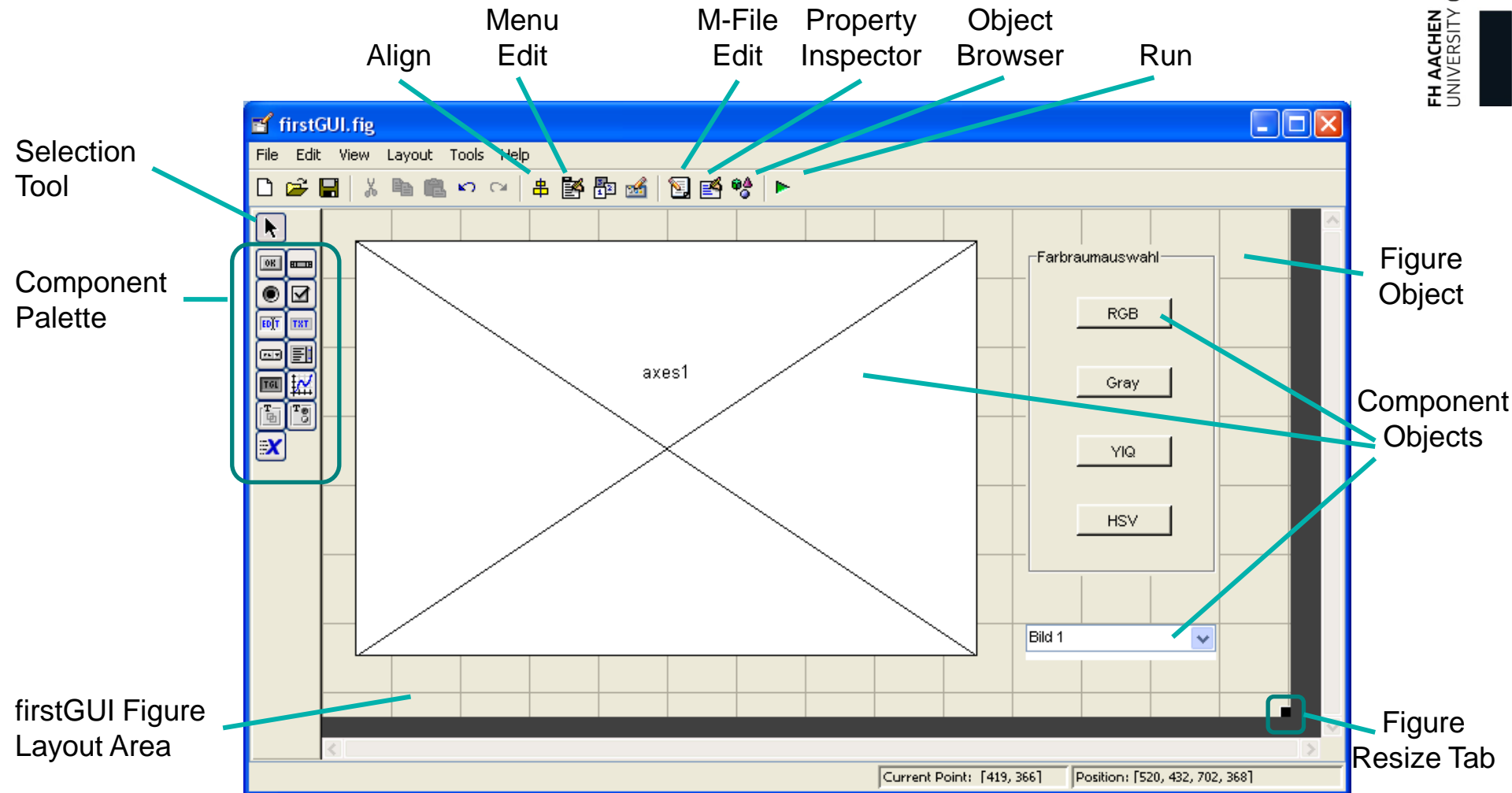
1. Definition eines Layouts für eine GUI mit GUIDE:
  - \*.fig-Datei: beinhaltet alle GUI Objekte sowie deren räumliche Anordnung
2. Programmierung der Methoden der GUI:
  - \*.m-Datei: beinhaltet den Source-Code, der durch die GUI Objekte ausgeführt werden soll (*callback*-Funktionen)



>> guide  
>> guide firstGUI



# GUIDE Layout-Editor



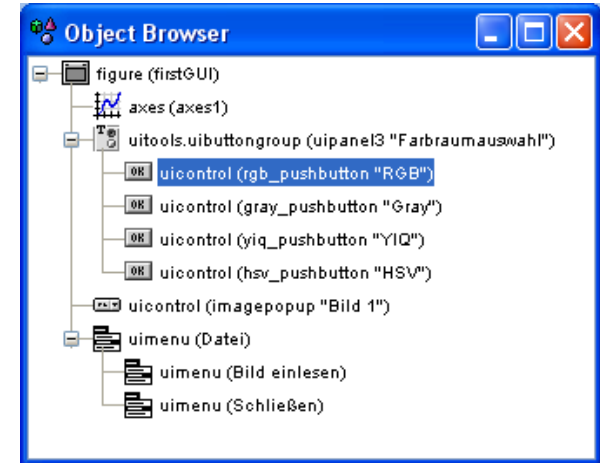


# Object Browser – Property Inspector

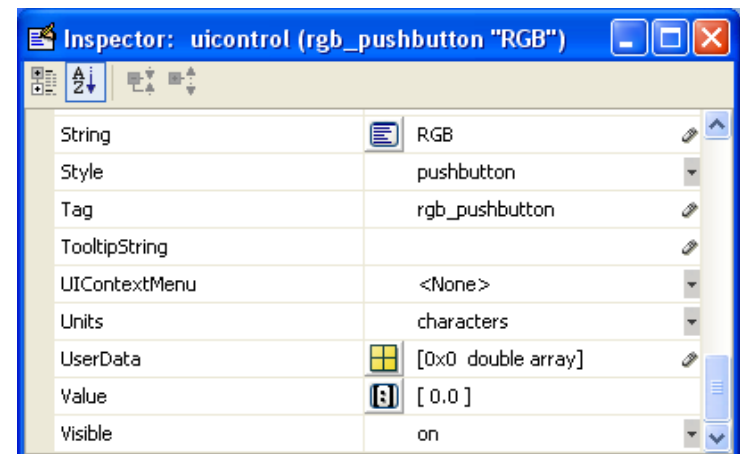
Object Browser: hierarchische Liste aller GUI-Objekte der *figure*

Property-Inspector: Editor für die Attribute eines GUI-Objektes

## GUIDE Object Browser

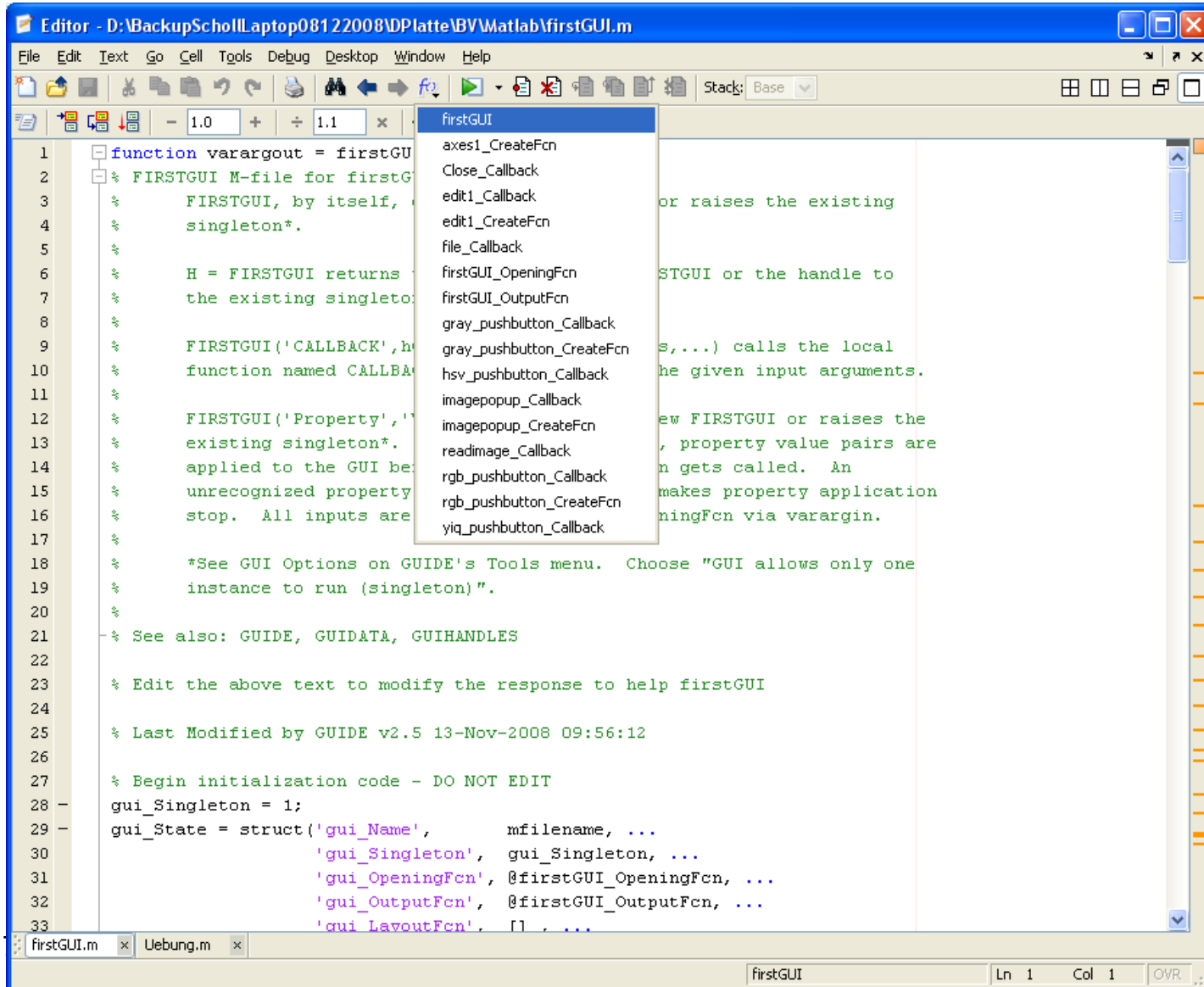


## Property Inspector for the figure-Object



# GUI m-Datei

(wird von GUIDE automatisch aus fig-Datei erzeugt)



```
1 function varargout = firstGUI
2 % FIRSTGUI M-file for firstGUI
3 % FIRSTGUI, by itself, creates a new firstGUI or raises the existing
4 % singleton*.
5 %
6 % H = FIRSTGUI returns the handle to the firstGUI or the handle to
7 % the existing singleton*.
8 %
9 % FIRSTGUI('CALLBACK',hObject,eventData,handles,...) calls the local
10 % function named CALLBACK in FIRSTGUI.m with the given input arguments.
11 %
12 % FIRSTGUI('Property','value',...) creates or raises the firstGUI
13 % existing singleton*. When specified as comma-separated list consisting of
14 % property and value, the property value is applied to the GUI before
15 % FIRSTGUI returns, and the argument is ignored. An unrecognized property
16 % stop. All inputs are unrecognized.
17 %
18 % *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
19 % instance to run (singleton)".
20 %
21 % See also: GUIDE, GUIDATA, GUIHANDLES
22
23 % Edit the above text to modify the response to help firstGUI
24
25 % Last Modified by GUIDE v2.5 13-Nov-2008 09:56:12
26
27 % Begin initialization code - DO NOT EDIT
28 gui_Singleton = 1;
29 gui_State = struct('gui_Name', mfilename, ...
30 'gui_Singleton', gui_Singleton, ...
31 'gui_OpeningFcn', @firstGUI_OpeningFcn, ...
32 'gui_OutputFcn', @firstGUI_OutputFcn, ...
33 'gui_LavoutFcn', [] , ...
```

# GUI m-Datei besteht aus 4 Teilen

---

1. Initialisierungs-Code
2. Opening / Output Funktionen
3. Fenster Callback Funktionen
4. Objekt Callback Funktionen

## 1. Initialisierungs-Code zwischen den „DO NOT EDIT“ Befehlen

```
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
 'gui_Singleton', gui_Singleton, ...
 'gui_OpeningFcn', @firstGUI_OpeningFcn, ...
 'gui_OutputFcn', @firstGUI_OutputFcn, ...
 'gui_LayoutFcn', [] , ...
 'gui_Callback', []);

if nargin && ischar(varargin{1})
 gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
 [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
 gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

# GUI m-Datei: Figure Öffnen- und Ausgabe-Funktionen

---

## 2. Figure-Öffnen- und Ausgabe-Funktionen: wie firstGUI\_OpeningFcn, firstGUI\_OutputFcn

```
% --- Executes just before firstGUI is made visible.
function firstGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% varargin command line arguments to firstGUI (see VARARGIN)

% Create data for the gui
handles.bild1 = imread('bild1.jpg');
handles.bild2 = imread('bild2.jpg');
handles.bild3 = imread('bild3.jpg');
handles.current_data = handles.bild1;
handles.current_image = handles.bild1;
handles.colormodel = 1;
imshow(handles.current_data);

% Choose default command line output for firstGUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

Code wird ausgeführt bevor  
die GUI für den Benutzer  
sichtbar wird bzw. bevor  
die GUI wieder beendet wird.

Parameter:  
hObject, eventdata, handles

handle ist ein struct-Datentyp und dient:

- zum Zugriff und zum Editieren von GUI-Objekteigenschaften

```
% set and get object properties from GUI
set(handles.firstGUI, 'Units', 'pixels');
uisz = get(handles.firstGUI, 'Position');
```

- zum gemeinsamen Datenaustausch innerhalb der Applikation (Initialisierung in der Opening\_Fcn)

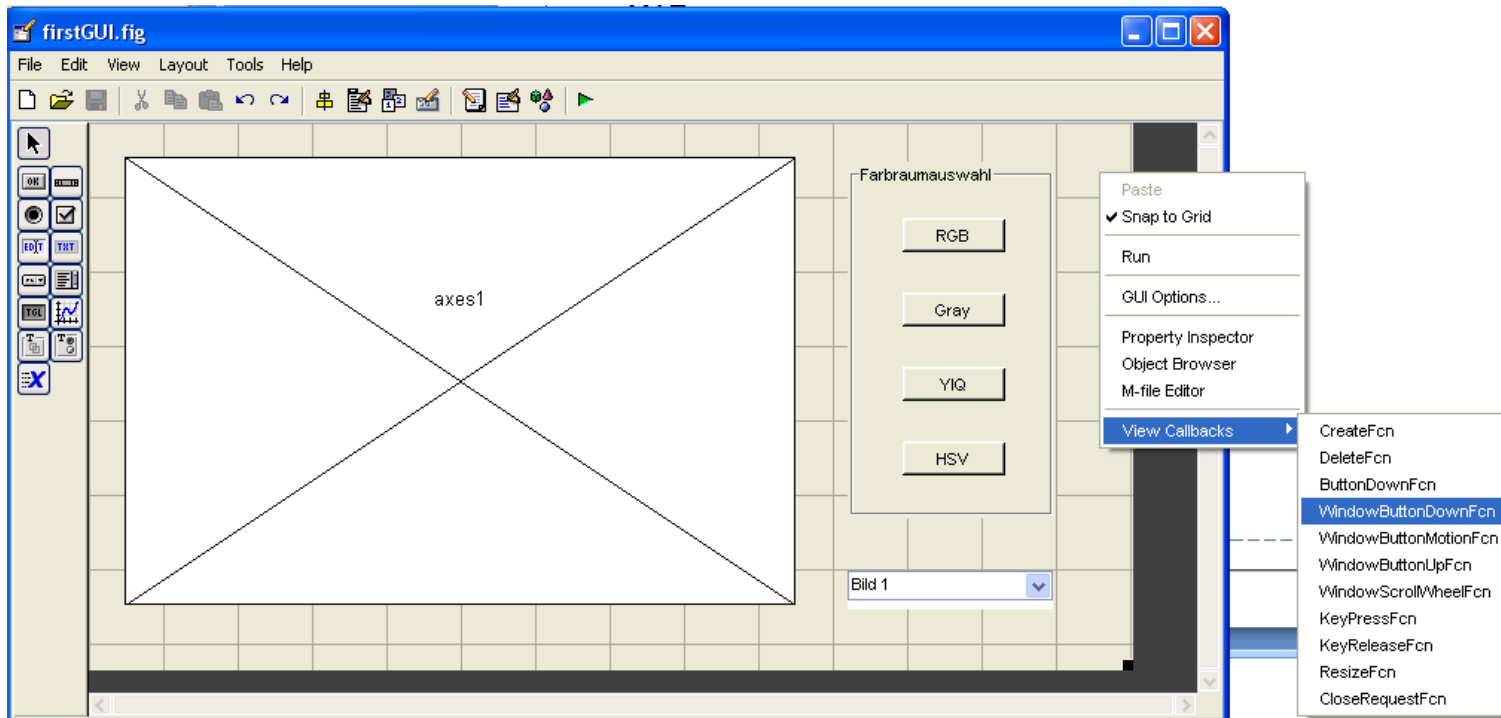
```
% Create data for the gui
handles.bild1 = imread('bild1.jpg');
handles.bild2 = imread('bild2.jpg');
handles.bild3 = imread('bild3.jpg');
handles.current_data = handles.bild1;
handles.current_image = handles.bild1;
handles.colormodel = 1;
```

```
% Update handles structure
guidata(hObject, handles);
```

## ACHTUNG:

Bei Veränderung der handle-Variablen muss diese den figure-Applikationsdaten hObject bekannt gegeben werden mit guidata(hObject, handles)

## 3. Figure-Callback-Funktionen: wie firstGUI\_WindowButtonDownFcn



## 4. Objekt-Callback-Funktionen: wie rgb\_pushbutton\_Callback, gray\_pushbutton\_Callback

```
% --- Executes on button press in gray_pushbutton.
function gray_pushbutton_Callback(hObject, eventdata, handles)
% hObject handle to gray_pushbutton (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
handles.colormodel = 2;
imshow(rgb2gray(handles.current_image));
```

```
% Update handles structure
guidata(hObject, handles);
```

```
function readimage_Callback(hObject, eventdata, handles)
% hObject handle to readimage (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
[filename, pathname] = ...
 uigetfile({'*.jpg'; '*.tif'; '*.gif'; '*..*'}, 'File Selector');
I = imread(filename);
handles.current_data = I;
handles.current_image = I;
handles.colormodel = 1;
imshow(I);
% update the handle
guidata(hObject, handles);
```

Code wird ausgeführt,  
wenn das GUI-Objekt  
vom Benutzer aktiviert  
wird. Hier: Button oder  
Menüpunkt.



# Demo: Erstellen einer GUI und Programmieren der M-Datei



# Vielen Dank für die Aufmerksamkeit !

FH Aachen  
Fachbereich Elektrotechnik und Informationstechnik  
Prof. Ingrid Scholl  
Eupenerstr. 70  
52066 Aachen  
T +49. 241. 6009 52177  
scholl@fh-aachen.de  
www.fh-aachen.de