

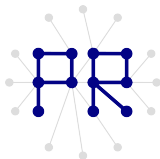
# Einführung in die Informatik II

## III.4 Visualisierung von 3D-Daten

Prof. Dr.-Ing. Marcin Grzegorzek<sup>1</sup>

Forschungsgruppe für Mustererkennung  
[www.pr.informatik.uni-siegen.de](http://www.pr.informatik.uni-siegen.de)

Institut für Bildinformatik  
Universität Siegen



---

<sup>1</sup>Die im Rahmen dieser Lehrveranstaltung verwendeten Lernmaterialien wurden uns zum Großteil von Herrn Prof. Dr. Wolfgang Wiechert und Herrn Prof. Dr. Roland Reichardt zur Verfügung gestellt.

# Inhaltsverzeichnis

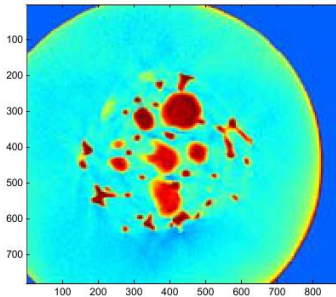
- I. MATLAB-Einführung
- II. Algorithmen
- III. MATLAB-Fortsetzung
  - 1. Internet und Werkzeuge
  - 2. Dateien
  - 3. Visualisierung
  - 4. **Visualisierung von 3D-Daten**
  - 5. Optimierung

# Was sind 3D-Daten?

- Typische 3D-Daten im Maschinenbau:
  - Konstruiertes Bauteil
  - Gemessene Werte (z.B. Qualitätssicherung)
  - Berechnete Werte (z.B. Computersimulationen)
- 3D-Daten
  - Wo:
    - Drei Koordinaten (x,y und z)
    - Typischerweise kartesisches Koordinatensystem.
  - Was:
    - Ein Wert an dieser Position (Skalar)
    - Ein Vektor an dieser Position

# Beispiel Werkstofftechnik

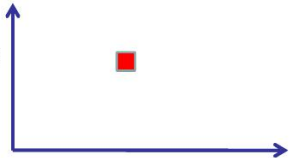
- Computertomogramm (ct) von Metallschäumen
  - ct Daten werden analysiert und 3D-Struktur des Metallschaums zu berechnen
  - 3D-Daten → FEM Analyse



# Beispiel

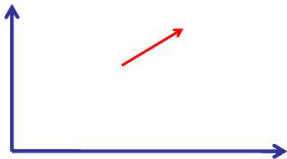
- Skalar:

- Luftdruck in bestimmter Höhe
- Festigkeit in Bauteil
- Computertomographie
- Also:  $P(x,y,z)$ , Wert



- Vektor:

- Windrichtung in bestimmter Höhe
- Spannung im Bauteil
- Also:  $P(x,y,z)$ ,  $W(x,y,z)$



# Beispieldaten in Matlab: "Wind"

- In Matlab werden Beispieldaten zu Verfügung gestellt:
  - Wind über Nordamerika
  - `load wind`
- Variablen:
  - Positionen:
    - x,y und z
  - Windgeschwindigkeitsvektoren an diesen Positionen:
    - u,v und w

# Matrix in Vektor

- Eine Matrix in einen Vektor konvertieren:

□ `a=[1 4 7;2 5 8; 3 6 9];`

1	4	7
2	5	8
3	6	9

□ `b=a (:);`

1
2
3
4
5
6
7
8
9

Spalten von links nach rechts  
werden untereinander gesetzt.

# Maximum/Minimum einer Matrix

- `xmin = min(x(:));`
- `xmax = max(x(:));`
  
- `ymin = min(y(:));`
- `ymax = max(y(:));`
  
- `zmin = min(z(:));`



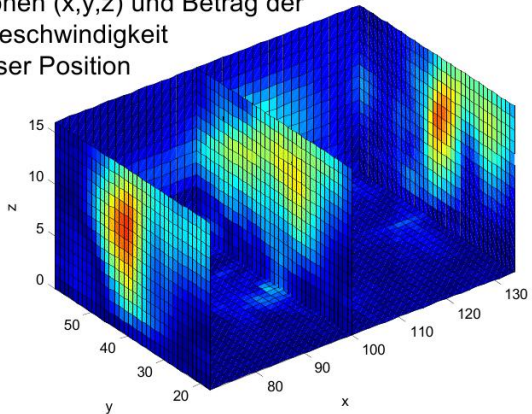
# Windgeschwindigkeit

- In den ersten Beispielen wird die Windrichtung an den verschiedenen Positionen nicht benötigt.
- Betrag der Windgeschwindigkeit:
  - Euklidische Norm: Wurzel aus der Quadratsumme
- Achtung: Elementweise soll dies geschehen!
  - `wind_speed = sqrt(u.^2 + v.^2 + w.^2);`

# Schnittbilder

- Stand der Berechnungen:
  - Positionen  $(x,y,z)$  und Betrag der Windgeschwindigkeit an dieser Position

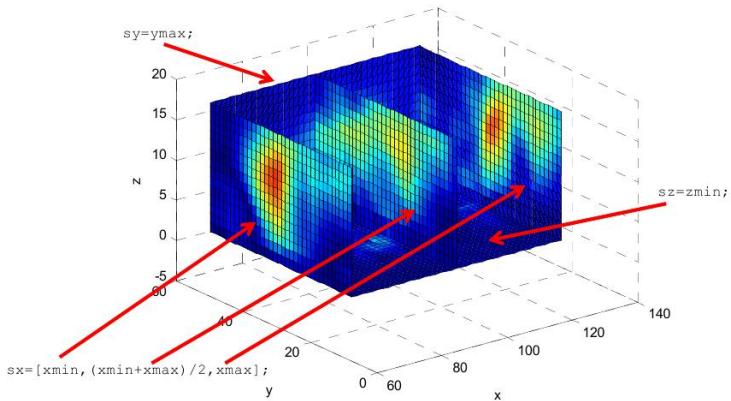
- Ziel:



# slice-Befehl

- `slice(x, y, z, wind_speed, sx, sy, sz);`
- `x, y, z` und `wind_speed` stehen schon zu Verfügung
- `sx, sy` und `sz` Position der Schnittebene entlang der `x, y` und `z`-Achse
  - Kann auch Vektor sein!
- Beispiel
  - `sx=[xmin, (xmin+xmax)/2, xmax];`
  - `sy=ymax;`
  - `sz=zmin;`

# Ergebnis

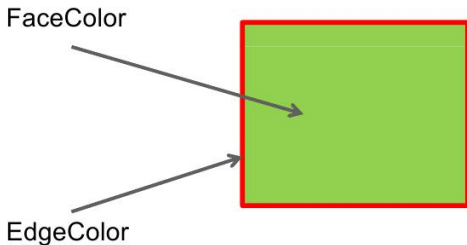


# Eigenschaften ändern

- Um das Aussehen von Grafikobjekten (nachträglich) zu ändern, muss man auf diese gezielt zugreifen können.
- Jedes Grafikobjekt gibt eine eindeutige Kennung (**handle**) als Funktionswert zurück.
- Beispiel:  
`hsurfaces=slice(x,y,z,wind_speed,sx,sy,sz)`
- Mit der `get(<handle>)`-Methode können Eigenschaften **abgerufen** werden.
- Mit der `set(<handle>)`-Methode können Eigenschaften **gesetzt** werden.

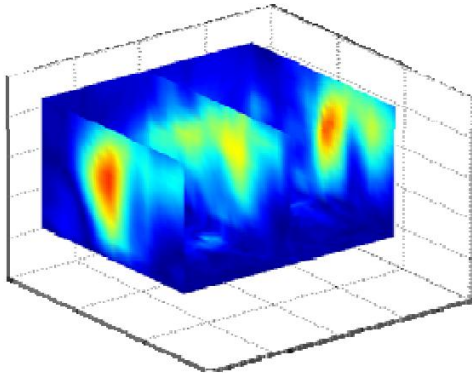
# Farben ändern

- `set(hsurfaces, 'FaceColor', 'interp', ,  
'EdgeColor', 'none')`



# Ergebnis

- Schwarzes Gitter ist weg.
- Farben auf den Oberflächen werden interpoliert.



## contourslice-Befehl

- Höhenringe auf die Schnitteben:

```
hcont =  
contourslice(x, y, z, wind_speed, sx, sy, sz);
```

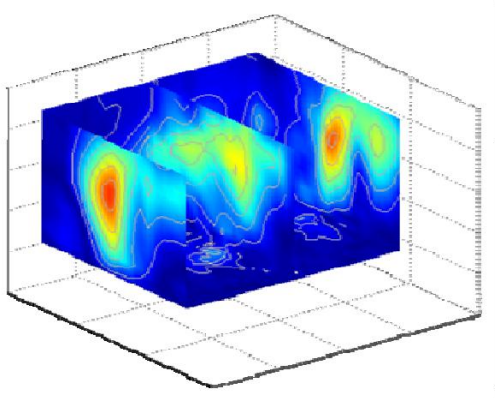
- Parameterfolge identisch mit dem slice-Befehl.
- Farbe und Strichdicke der Linien ändern  

```
set(hcont, 'EdgeColor', [.7, .7, .7],  
      'LineWidth', .5)
```
- `[.7, .7, .7]`: [rot, grün, blau]
  - Alle Werte identisch = Grauwert



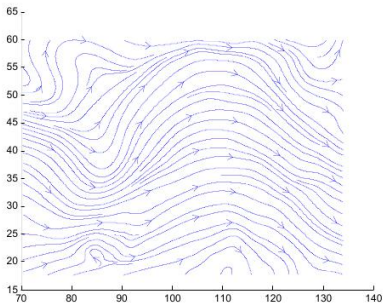
# Ergebnis

- Graue Höhenlinien (gleiche Windstärke)



# Vektorfelder

- In Matlab gibt es diverse Funktionen, um Vektorfelder zu visualisieren.
- Aus der Wettervorhersage bekannte Darstellung:



## streamslice-Befehl

- Der `streamslice`-Befehl zeichnet auf Schnittebenen die Strömungsfelder.
- Parameterfolge identisch mit dem `slice`-Befehl.

- Beispiel:

```
streamslice(x, y, z, u, v, w, [], [],  
            (zmax+zmin)/2)
```

- Keine Schnittebenen in x und y,  
aber eine in der Mitte von z.

# Datenmatrix

- Der Befehl `meshgrid` erzeugt Datenmatrizen für die Berechnung und Visualisierung von 3D-Feldern.
- Manchmal müssen die Daten mit diesem Befehl aufbereitet werden.
- Beispiel:
  - Man möchte innerhalb der Wind-Gebietes Datenpunkte bestimmen, an denen Berechnungen durchgeführt werden:
  - `[sx, sy, sz] = meshgrid(80, 20:5:50, 1:5:15);`
    - X-Position: 80
    - Y-Position: 20 bis 50 in 5er Schritten
    - Z-Position: 1 bis 15 in 5er Schritten

# Stromlinien

- Die für die Visualisierung von 3D-Stromlinien werden 3D-Vektorfelder benötigt.

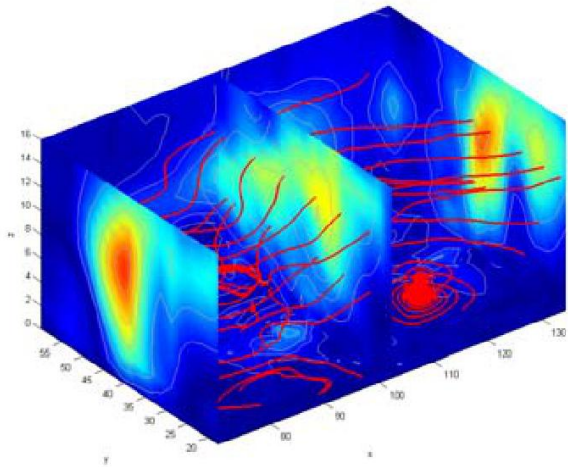
- Beispiel:

```
hlines = streamline(x, y, z, u, v, w, sx, sy, sz);
```

- $x, y, z$ : Position
  - $u, v, w$ : Geschwindigkeit
  - $sx, sy, sz$ : Startpositionen der Stromlinien
- 
- Farbe und Dicke der Stromlinien setzen:  

```
set(hlines, 'LineWidth', 2, 'Color', 'r')
```

# Ergebnis



# Strombänder mit streamribbon

```
load wind
[sx sy sz] = meshgrid(80,20:10:50,0:5:15);
hribbon=streamribbon(x,y,z,u,v,w,sx,sy,sz);

set(hribbon,'FaceColor','interp','EdgeColor',
    'none')
```

## Ansicht und Beleuchtung

```
axis tight
view(3);
daspect([2,2,1])
camlight; lighting gouraud
```

## Strombänder mit streamtube

- `htubes = streamtube(x,y,z,u,v,w,sx,sy,sz,[2.5 20]);`
- `[2.5 20]`:
  - 20 = Anzahl der „Ecken“ der Röhre
  - 2.5 = Verstärkungsfaktor für den Durchmesser
- `set(htubes, 'EdgeColor', 'none', 'FaceColor', 'r', 'AmbientStrength', .5)`



# Ergebnis

