9. 2D-Strömungsvisualisierung

Voraussetzung:

Simulationsdaten

Strömungsfelder (Flow Fields)

üblicherweise sind mehrere Datenfelder gegeben,

- Geschwindigkeit (Vektorfeld)
- Druck (Skalarfeld)
- Dichte (Skalarfeld)
- Temperatur (Skalarfeld)

bzw. können errechnet werden:

- Absolutbetrag der Geschwindigkeit
- Divergenz und Rotation (Skalarfelder)

Übersicht:

- **Direkte Verfahren**
  - Farbcodierung (color coding)
  - Vektorpfeile, Arrow Plots *(hedgehogs)*
  - Glyphen/Icons

- **Integrations-basierte Verfahren**
  - Wahl der Startpunkte
  - Particle Tracing im unstrukturierten Gitter
  - Particle Tracing im curvilinearen Gitter
  - Streamlets
  - Streamlines, Streaklines, Pathlines
  - Texturbasierte Verfahren

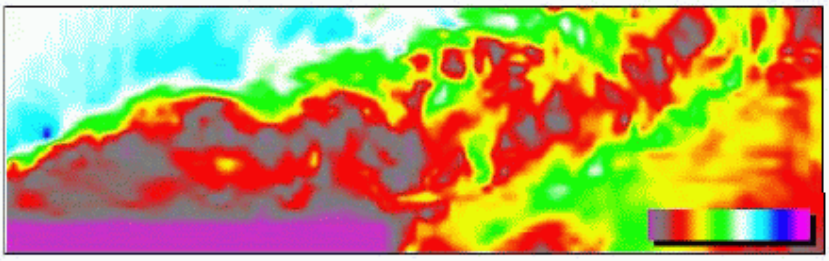
Direkte Verfahren

Verfahren mit möglichst direkter Transformation der Strömungsdaten in visuelle Objekte

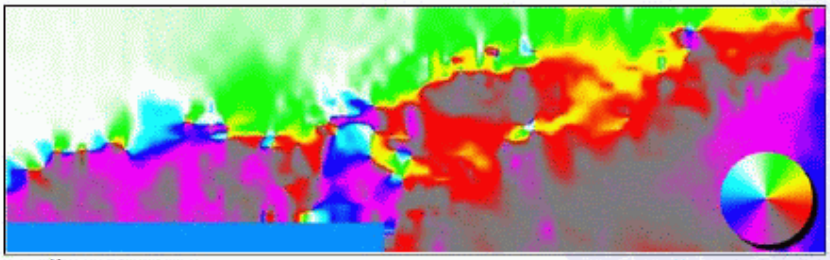
- schnelle, unmittelbare Darstellung ohne aufwändige Vorverarbeitung
- nur lokale Eigenschaften, Zusammenhänge und Langzeit-Verhalten wird nicht explizit dargestellt

Farbkodierung

- *Absolutbetrag der Strömungsgeschwindigkeit*
  Vektorlänge farbkodiert



- *Richtung der Strömung*
  Richtung (Winkel in Polarkoordinaten) farbcodiert



z.B. Hue-Winkel

Quelle: TU Wien

Attribute der Strömung werden farblich dargestellt

- Geschwindigkeitsbetrag,
- Druck, Dichte, Temperatur

Vorteil:

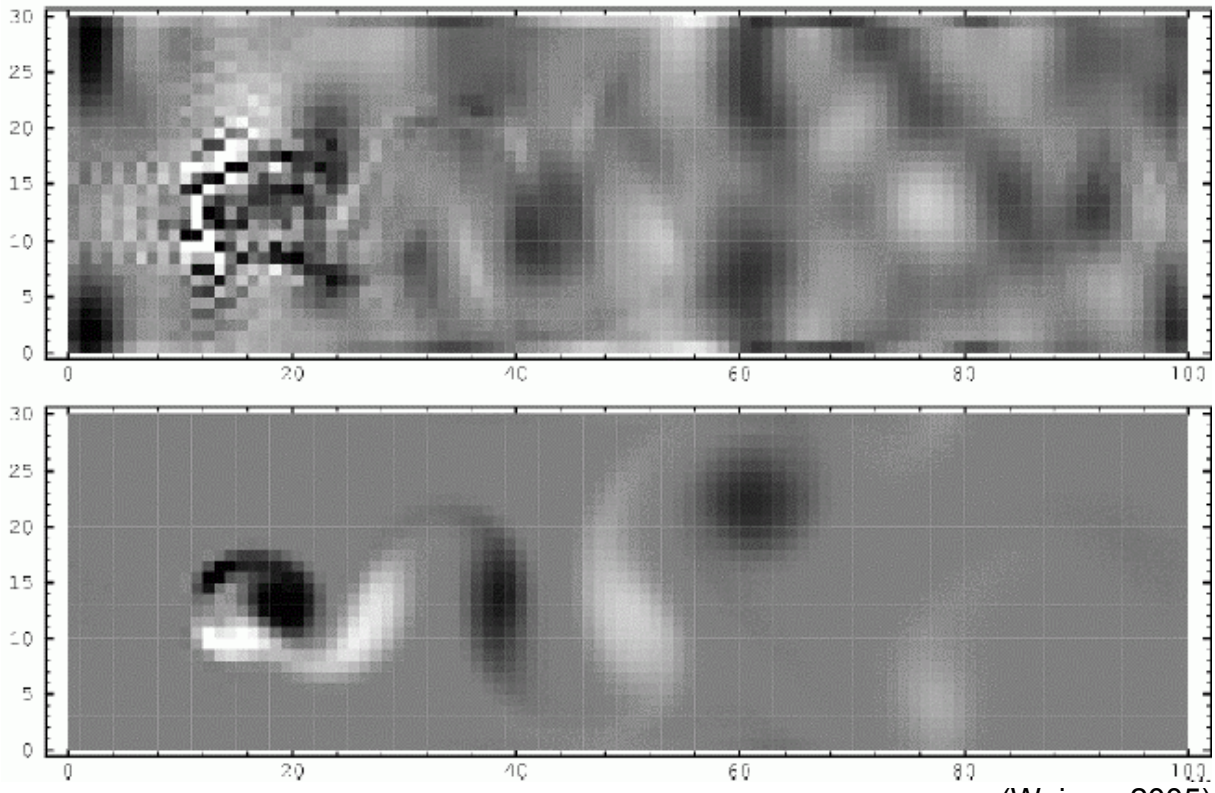
- Sehr einfache Verfahren,
- gut geeignet auch für zeitabhängige Daten

Nachteil:

- nur skalare Größen, keine Orientierung
- keine zeitlichen Zusammenhänge

(Rezk-Salama, o.J.)

Helligkeits- oder Farbcodierung auch für Divergenz und Rotation möglich:

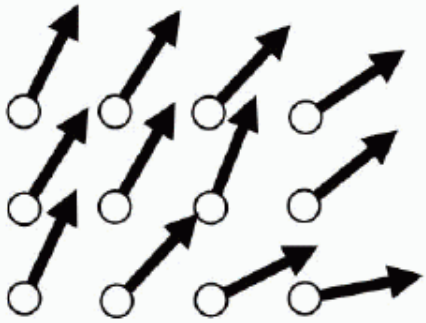


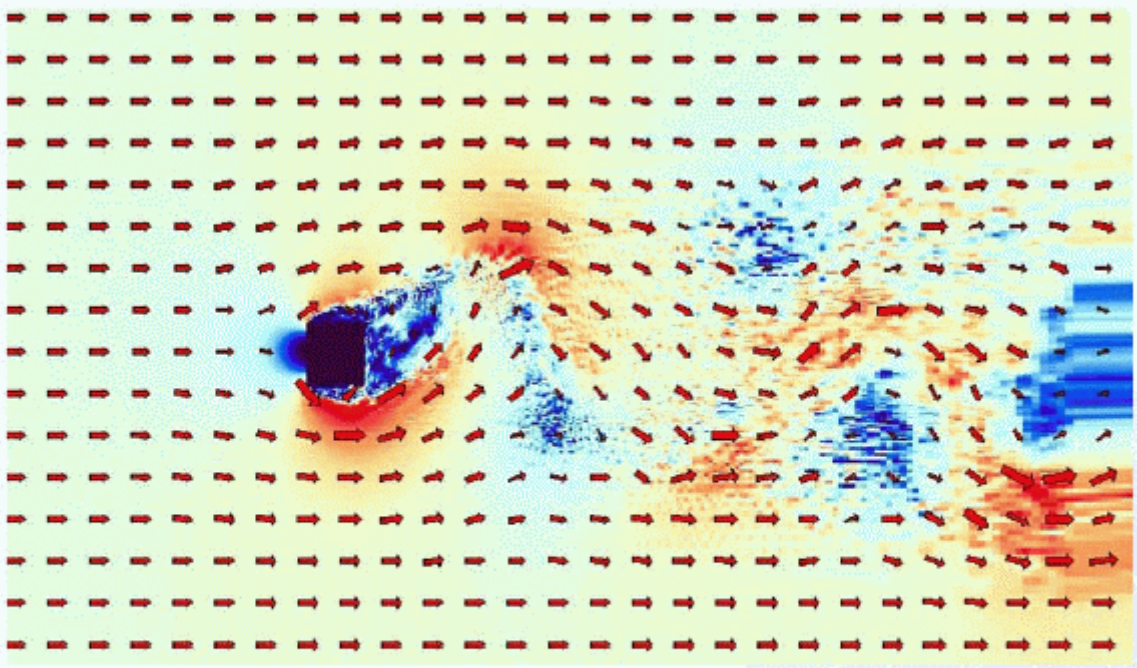
(Weimar 2005)

Vektorpfeile

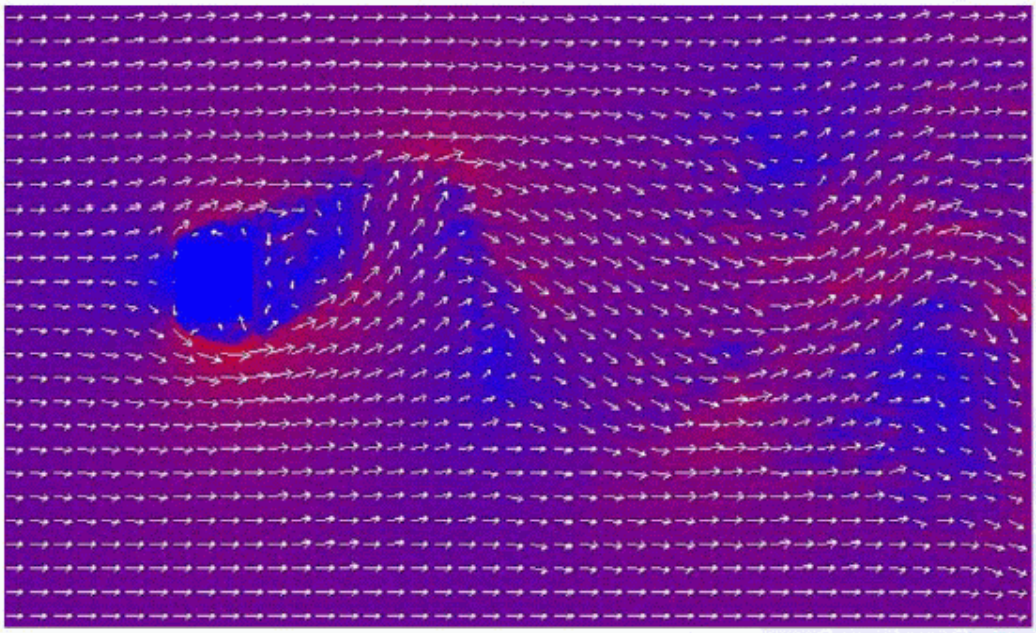
Datenpunkte werden durch kurze Linien oder Vektorpfeile dargestellt

- z.B. gleichmäßig auf kartesischem Gitter
- auch *Arrow Plots* oder *Hedgehogs*.





Bei relativ gleichmäßiger Strömung:
Länge/Größe des Pfeils entspricht Geschwindigkeitsbetrag

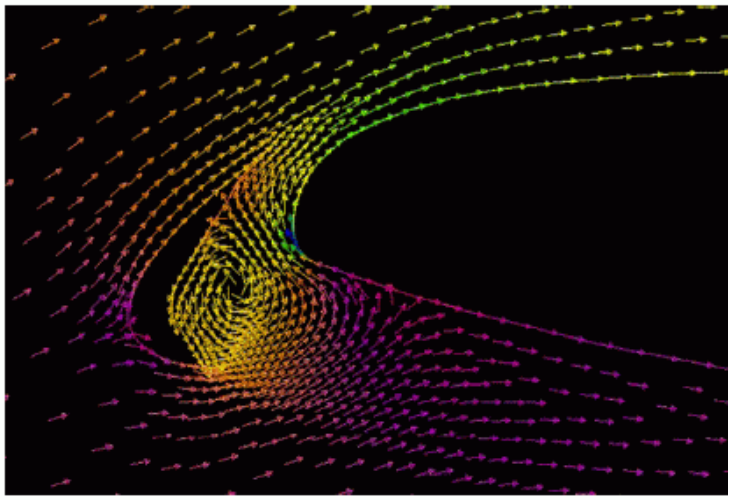


Bei stark variierender Geschwindigkeit:
Länge des Pfeils einheitlich, Absolutbetrag über Farbkodierung

(Rezk-Salama, o.J.)

Pfeildarstellung :

- Richtung im Feld
- Orientierung
- Beträge/Größen



\+ Einfach, Schnell

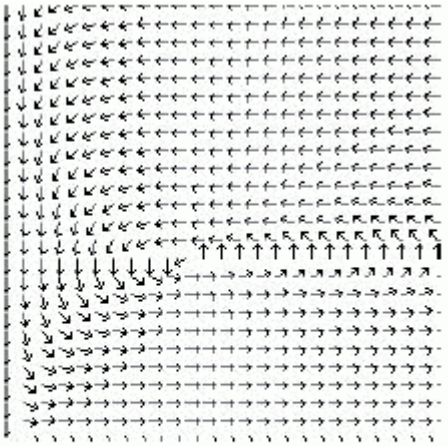
\- Starke Variationen der Beträge sind schwierig

\- Verdeckung (bei 3D)

(Bartz 2005)

Aspekte:

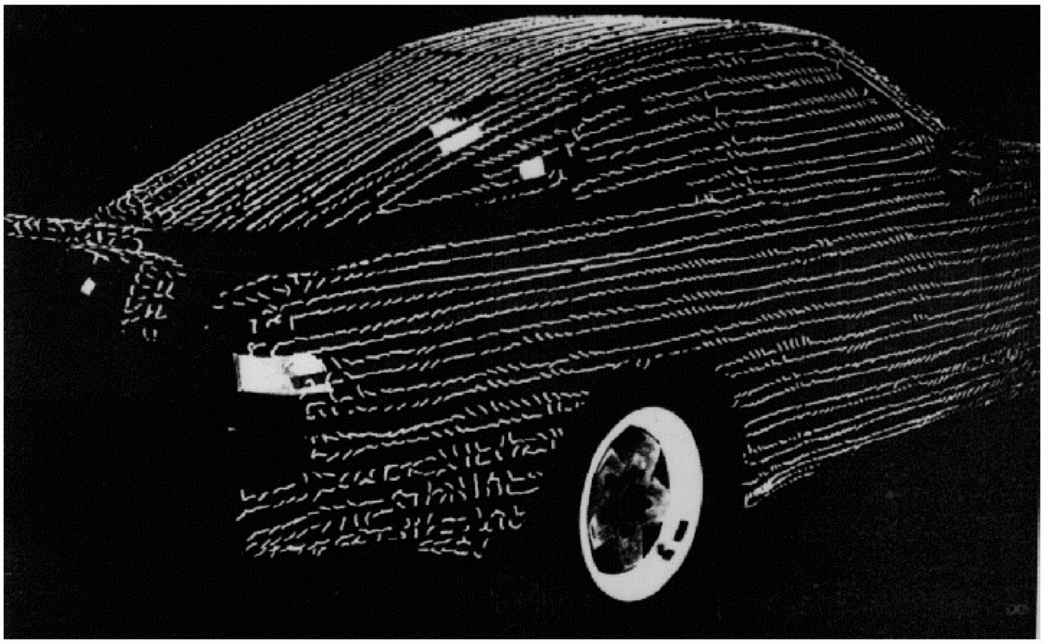
- Direkte FlowVis
- normierte Pfeile vs. Skalierung mit Geschwindigkeit
- 2D: ganz gut brauchbar, 3D: meist problematisch
- oft nur bedingt verständlich (zeitliche Komponente fehlt)
- oft in Verwendung!



(Hauser 2007)

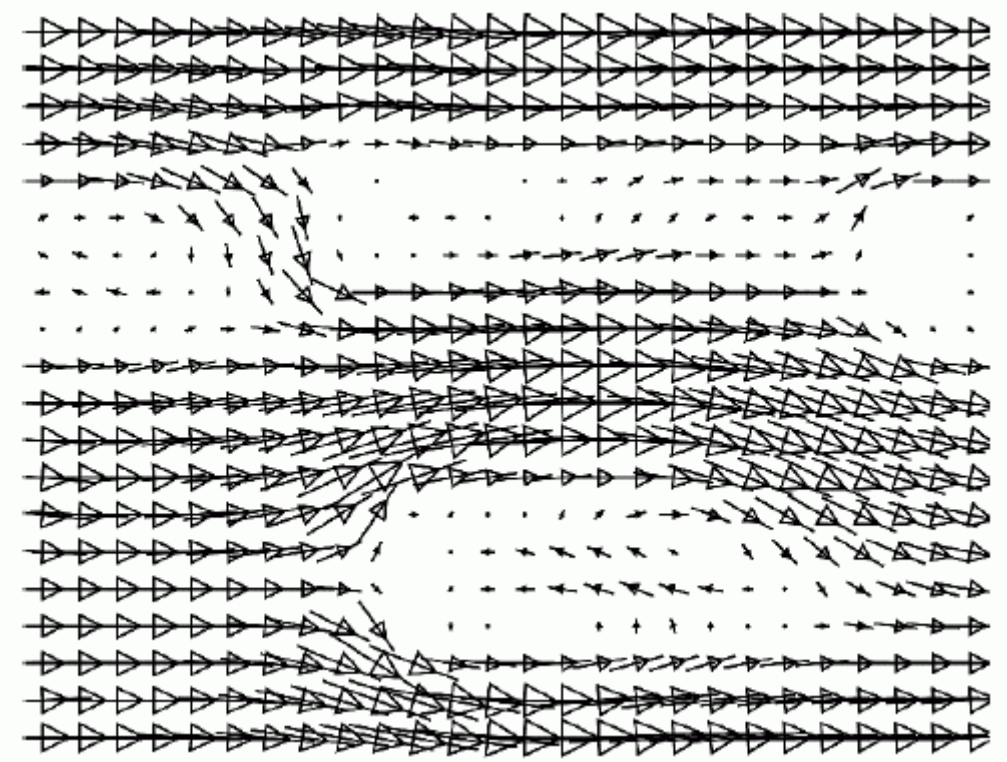
Analogie zum realen Experiment:

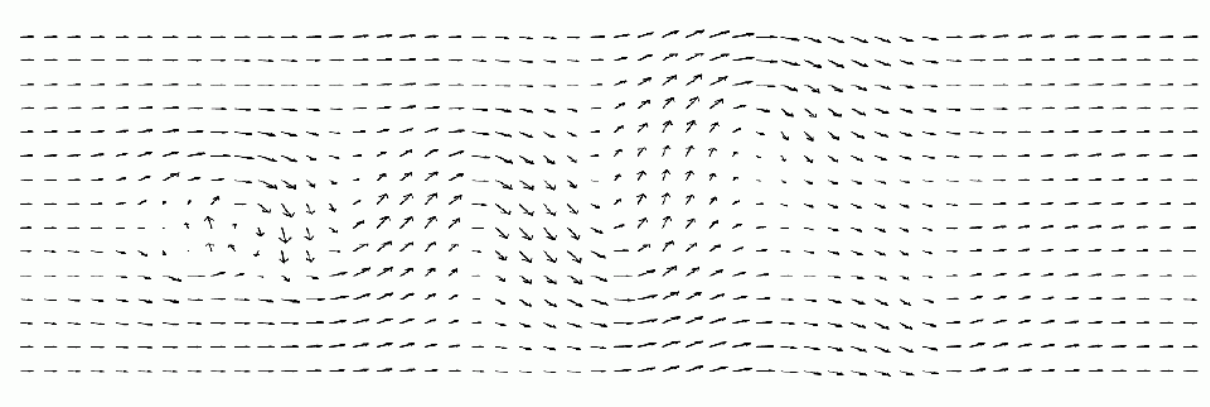
- Tufts (Fäden) → Vektoren, kurze Stromlinien



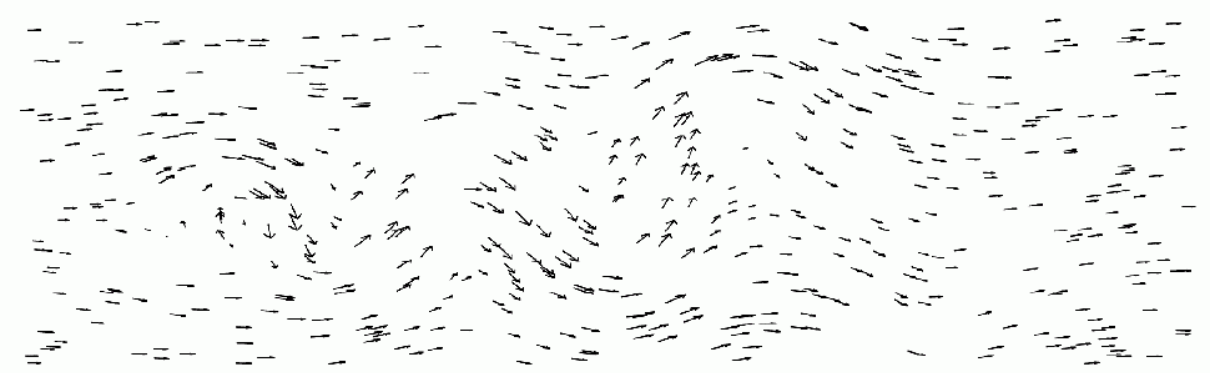
(Weimar 2005)

verschiedene Typen von Vektorplots:

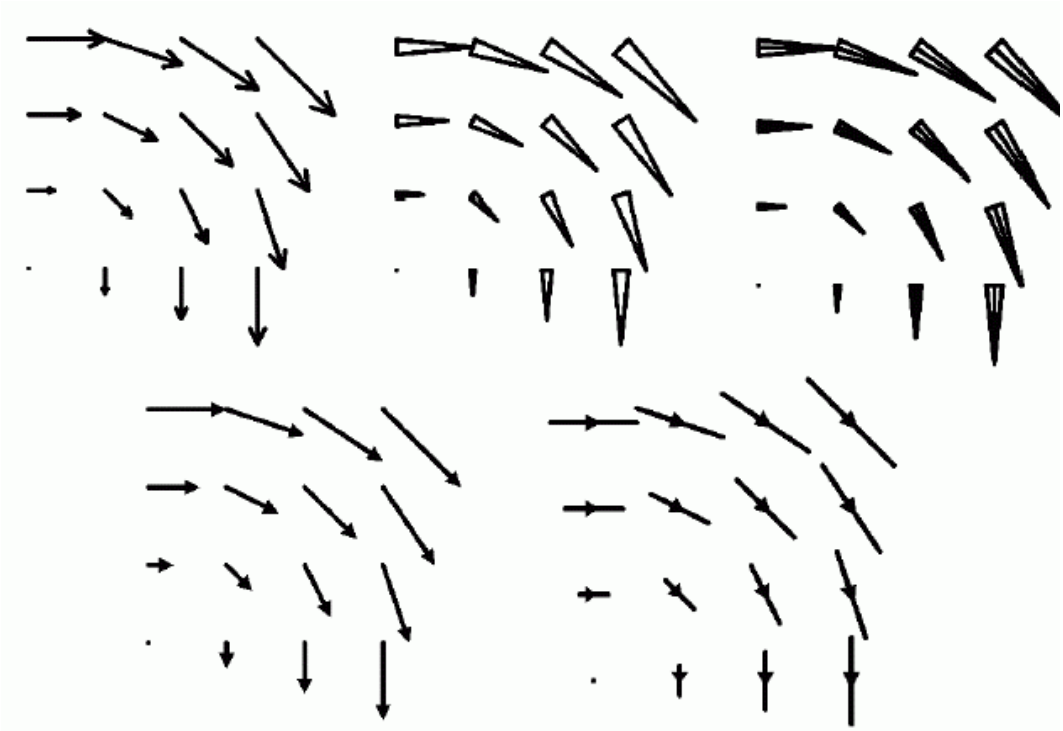




Vektorplot mit zufälligen Positionen:

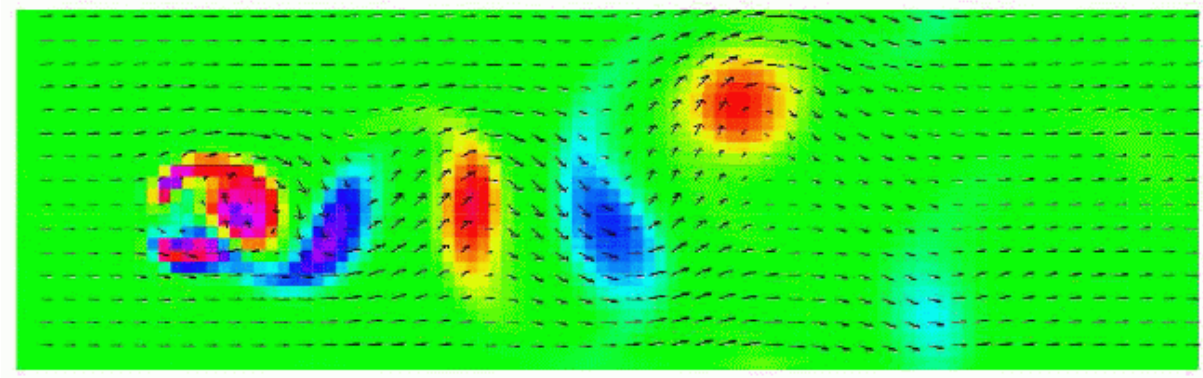


Darstellungsmöglichkeiten für 2D-Vektoren:



(Weimar 2005)

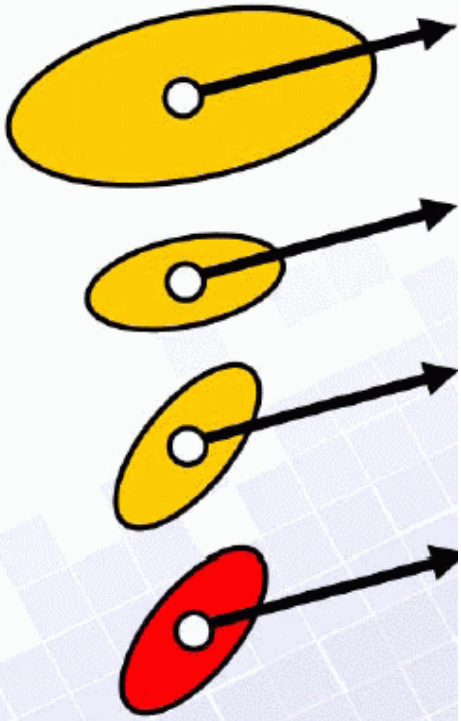
Kombination mit Farbcodierung: Vektoren + Rotation



(Weimar 2005)

Glyphen/Icons

Verallgemeinerung der Vektorpfeile:
Geometrische Objekte, die lokale Eigenschaften der Strömung darstellen.

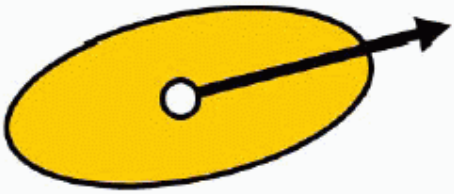


Beispiel: Vektorpfeil mit Ellipse

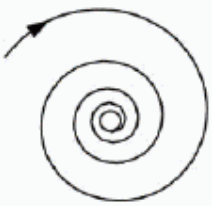
- Pfeil zeigt in *Strömungsrichtung*
- Fläche der Ellipse: *Geschwindigkeitsbetrag*
- Verdrehung der Ellipse : *Rotation*
- Farbcodierung: *Divergenz*

Schwierigkeiten:

Geometrische Eigenschaften sind nicht unabhängig!



Pfeil: Strömungsrichtung
Fläche: Geschwindigkeit
Verdrehung: Rotation
Farbe: Divergenz



Ellipsen-Beispiel: Wirbelsenke (kritischer Punkt)

- *Keine Geschwindigkeit:* Ellipse verschwindet
- *Rotation* nicht mehr erkennbar.
- *Divergenz* nicht mehr erkennbar.

Schwierigkeiten:

Auf *Skalierung* achten!

Länge skaliert linear.
Fläche skaliert quadratisch.
Drehung ist invariant gegenüber Skalierung.

Beispiel:

Visualisierung der Rotation (Vorticity)

Radius und Transparenz:
Betrag der Rotation:

Farbe (rot und grün):
Rotationsrichtung

Quelle: Jason S. Sobel, Brown University, USA

Zusammenfassung

Vektorpfeile/Glyphen/Icons

Vorteile:

- Skalare und vektorielle Information wird dargestellt
- Direkte Darstellung, geringer Rechenaufwand

Nachteile:

- Dichte der Vektorpfeile ist begrenzt (sonst wird das Bild schnell unübersichtlich)
- Nur lokale Information (keine Zusammenhänge)
- Es ist schwierig, intuitiv verständliche Glyphen zu finden

Verfahren mit Integration

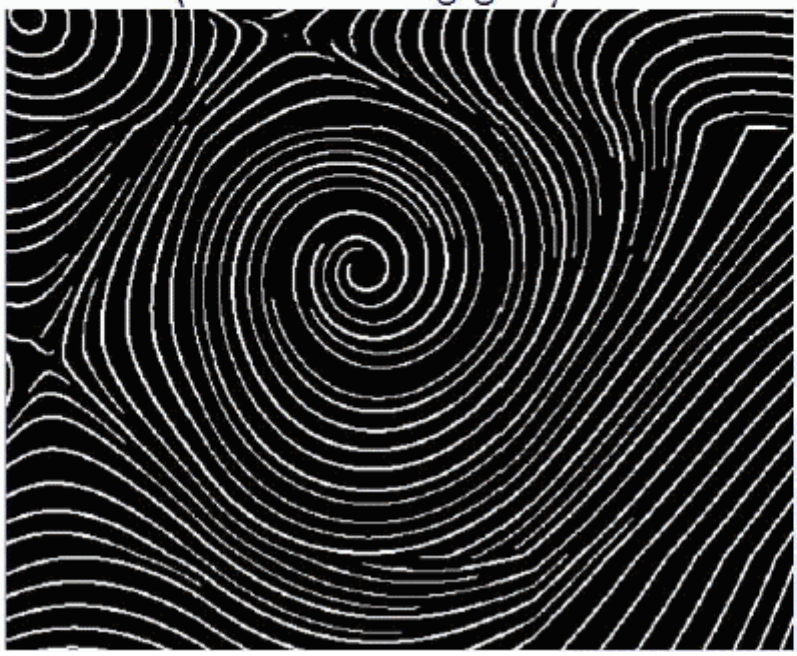
Verfahren, die auf Integration des Vektorfelds (Partikelbahnen) basieren:

- erfordert die Bestimmung von Partikelbahnen.
- Numerisches Integrationsverfahren notwendig *(z.B. Euler, Heun, Runge-Kutta)*
- Vorteil gegenüber direkten Verfahren: Zeitliche Zusammenhänge werden dargestellt

Partikelbahnen

Beispiel:

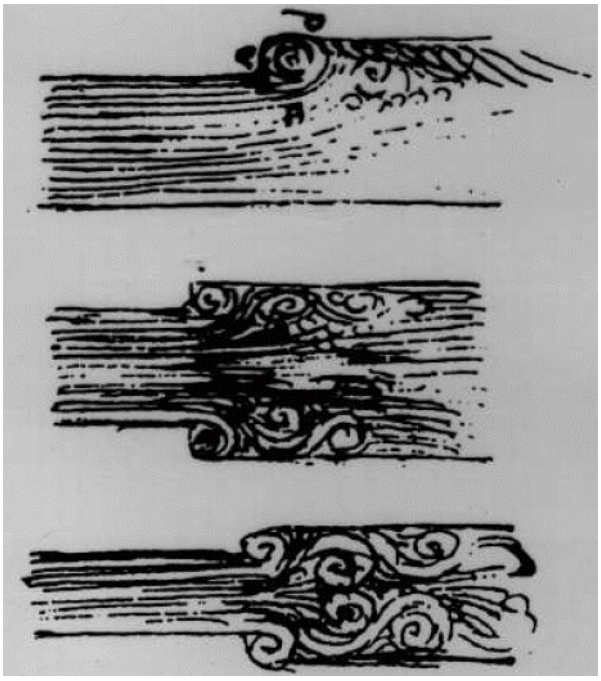
Feldlinien, Strömungslinien (zeitlicher Zusammenhang) im statischen (zeitunabhängigen) Vektorfeld



Quelle: TU Wien

(Rezk-Salama, o.J.)

Analogie zu realen Beobachtungen (hist. Zeichnungen):



(Weimar 2005)

Berechnung der Partikelbahn

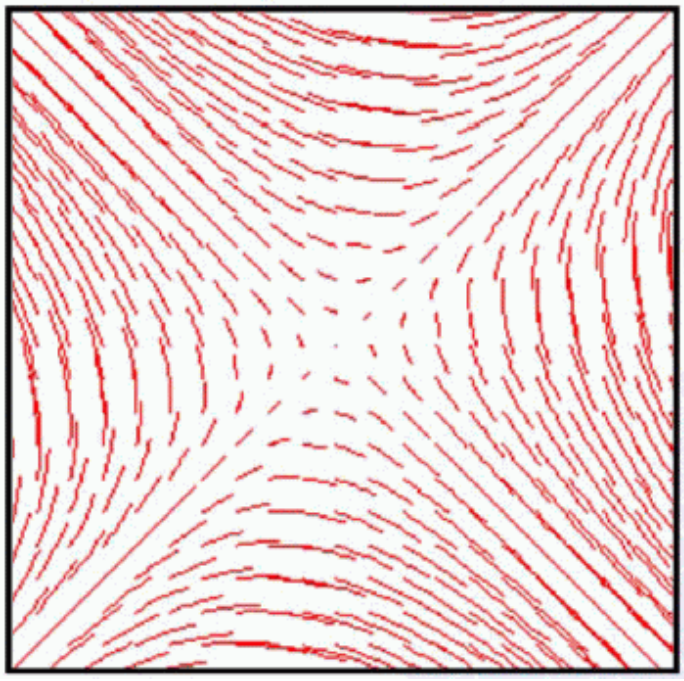
- Schrittweise mit numerischem Integrationsverfahren *(siehe letzte Stunde)*
- Einfach auf uniformem Gitter
- Für curvilineare und unstrukturierte Gitter ist *Zellsuche* erforderlich.

Wahl der Startpunkte (seed points) für Partikelbahnen

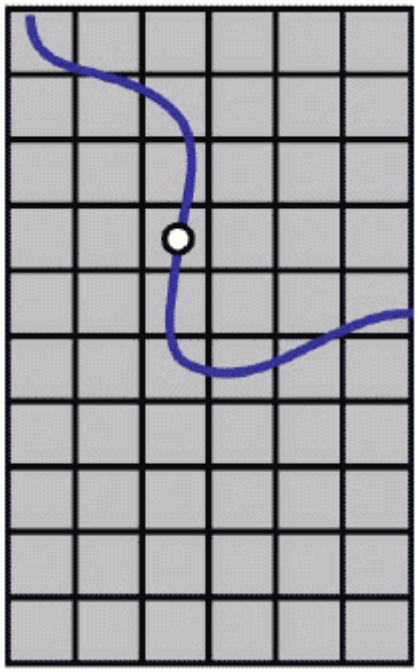
- Benutzerdefiniert *(Flowfield-Probes)*
- Uniformes Gitter von Startpunkten (streamlets) (wird leicht unübersichtlich und redundant)
- Zufällige Verteilung von Startpunkten
- Spezielle Algorithmen

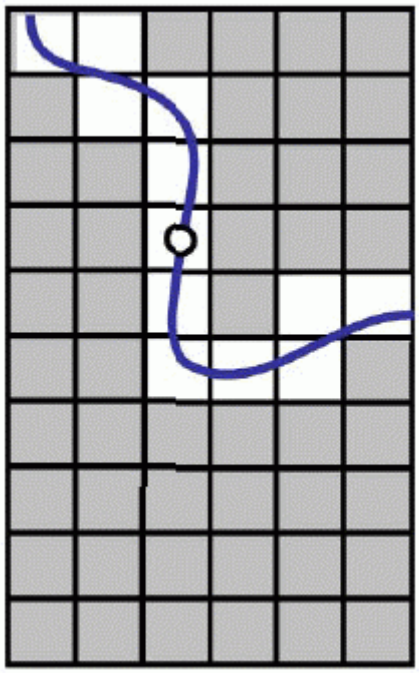
Wahl der Startpunkte

- **Streamlets:** Zeichne für jeden Gitterpunkt eine kurze Partikelbahn (nur wenige Zeitschritte vor- bzw. rückwärts)

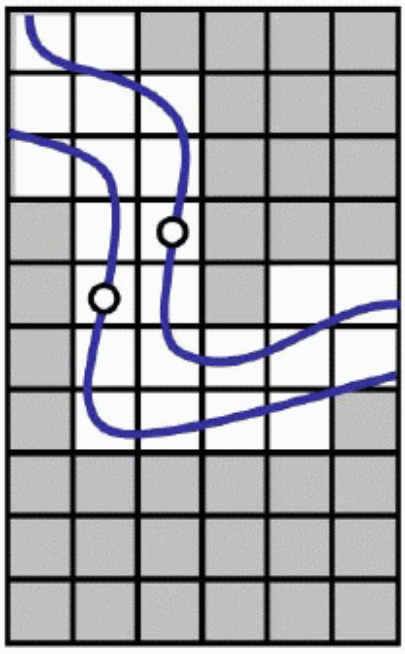


Seed point placement





- *Seed Point Placement Algorithms:* Ziel ist eine möglichst gleichmäßige Verteilung der Partikelbahnen



- Teile die Ebene in Zellen bestimmter Größe (unabhängig vom zugrunde liegenden Gitter).
- Wähle den ersten Startpunkt (beliebig)
- Bestimme die Partikelbahn vor- und rückwärts (solang es geht)
- Markiere alle Zellen, die die Partikelbahn schneidet
- Wähle einen neuen Startpunkt in einer freien (unmarkierten) Zelle
- Breche die Berechnung der neuen Partikelbahnen ab, wenn eine bereits markierte Zelle geschnitten wird

(Rezk-Salama, o.J.)

gitterunabhängige Variante des Algorithmus:

Überblick über Algorithmus

- Idee: Strömungslinien sollen sich nicht zu nahe kommen
- Ansatz:
  - Wähle neuen Startpunkt d_{sep} von einer bestehenden Strömungslinie entfernt
  - Vorwärts- und Rückwärtsintegration bis Abstand d_{test} erreicht.
  - Zwei Parameter:
    - d_{sep} … Startabstand
    - d_{test} … Mindestabstand

Algorithmus – Pseudocode

- Compute initial stream line, put into queue
- Initial stream line gets current stream line
- WHILE not finished DO:

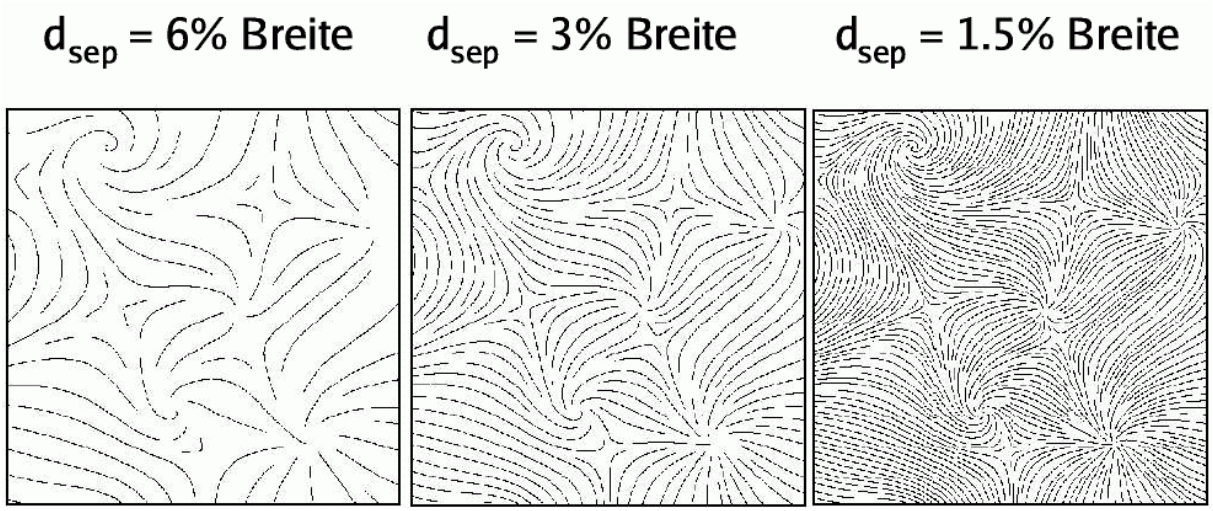
  TRY: get new seed point which is d_{sep} away from current stream line

  IF successful THEN compute new stream line and put to queue

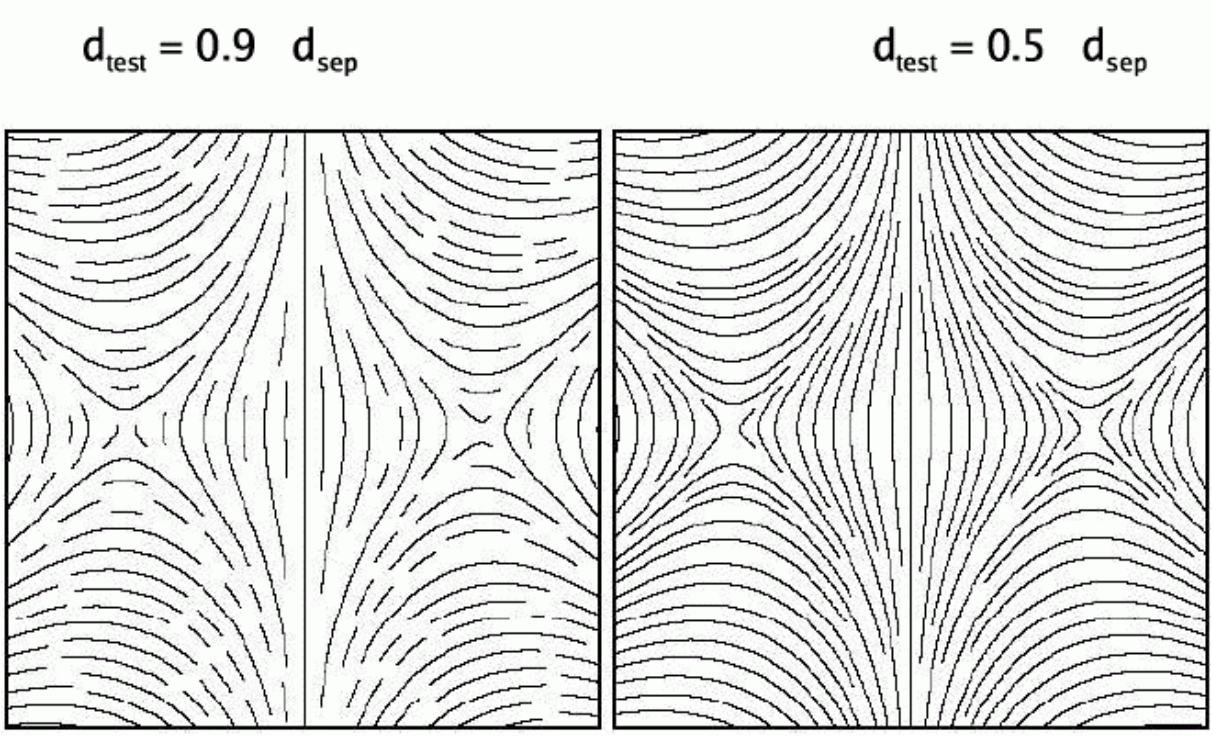
  ELSE IF no more stream line in queue
  THEN exit loop
  ELSE next stream line in queue becomes current stream line

(Hauser 2007)

Auswirkung verschiedener Dichten:



d_{sep} vs. d_{test} :

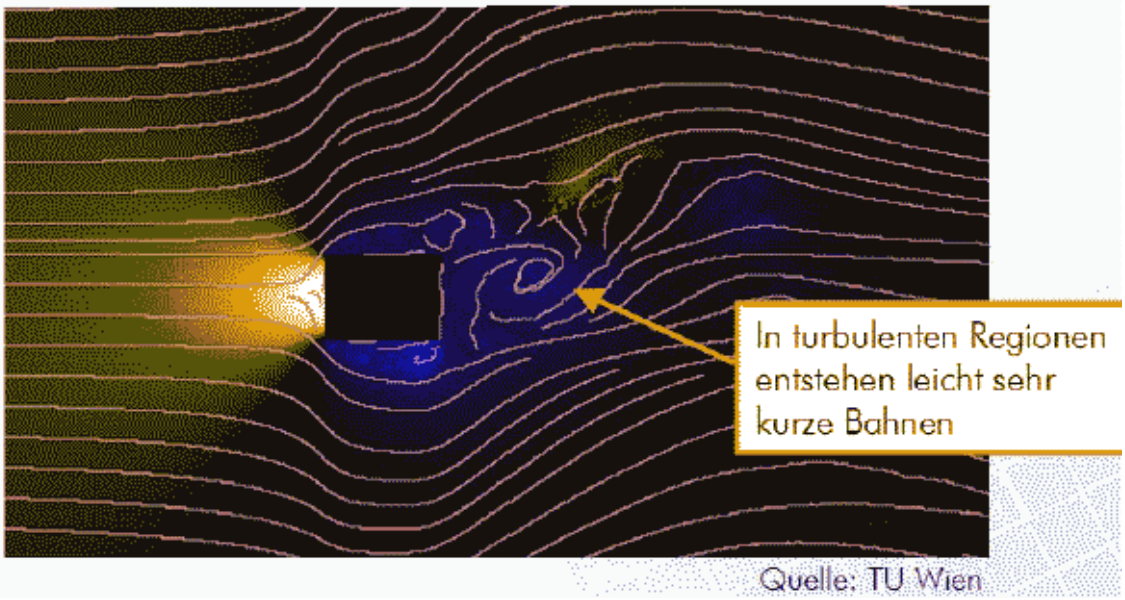


(Weimar 2005)

Partikelbahnen

Beispiel:

Feldlinien, Strömungslinien (zeitlicher Zusammenhang) im statischen (zeitunabhängigen) Vektorfeld



(Rezk-Salama, o.J.)

Partikelbahn / Partikelspur:

- Gegeben Vektorfeld $\underline{v}(\underline{x},t)$
- Partikelspur: Trajektorie eines Teilchens über die Zeit
  - Styroporkugeln & Langzeitbelichtung
  - Gegeben $\underline{x}(0) = \underline{x}_0$

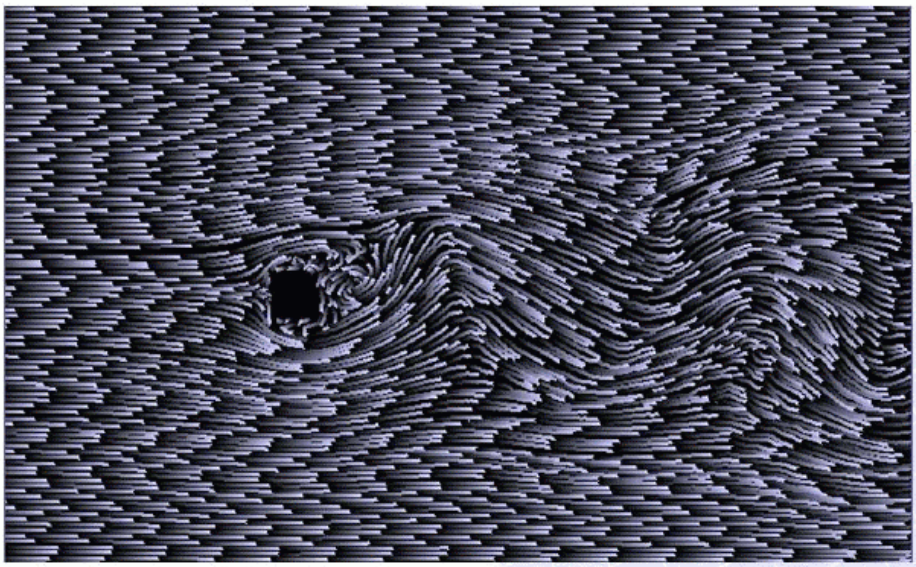
$$\frac{d\underline{x}}{dt} = \underline{v}(\underline{x},t)$$

- Position des Teilchens zum Zeitpunkt $T > t_0$

$$\underline{x}(T) = \underline{x}(0) + \int_{t=0}^{T} \underline{v}(\underline{x}(t),t)$$

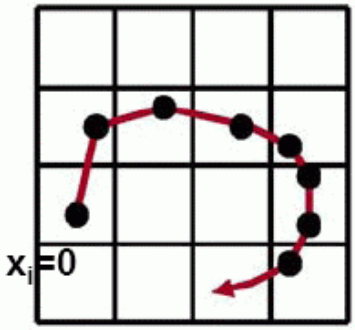
(Weimar 2005)

- *Farbcodierung der Zeitschritte*



(Rezk-Salama, o.J.)

Partikelverfolgung auf Kartesischen Gittern:



Wähle Startpunkt $\boldsymbol{x}_{i=0}$

Solange $\boldsymbol{x}_i$ in gültigem Bereich

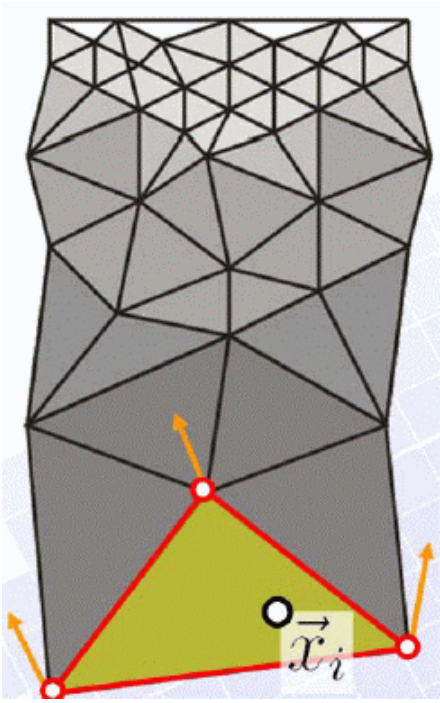
- Finde Zelle, die $\boldsymbol{x}_i$ enthält, und α_i
- Berechne $\boldsymbol{v}(\boldsymbol{x}_i)$
- Integriere zu $\boldsymbol{x}_{i+1}$
- Schrittweitenanpassung
- Zeichne Segment $(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})$ oder Kügelchen
- $i \leftarrow i+1$

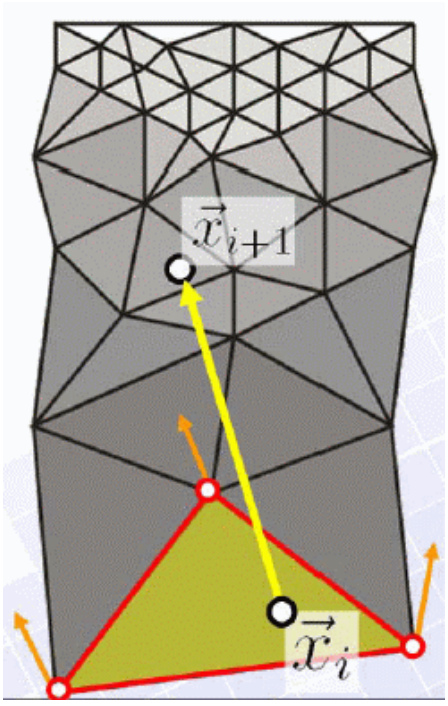
- Lokalisation der Zelle (über Indizes)
- Interpolation ist
  - bilinear (2D)
  - trilinear (3D)

(Bartz 2005)

Partikelverfolgung im Dreiecksgitter

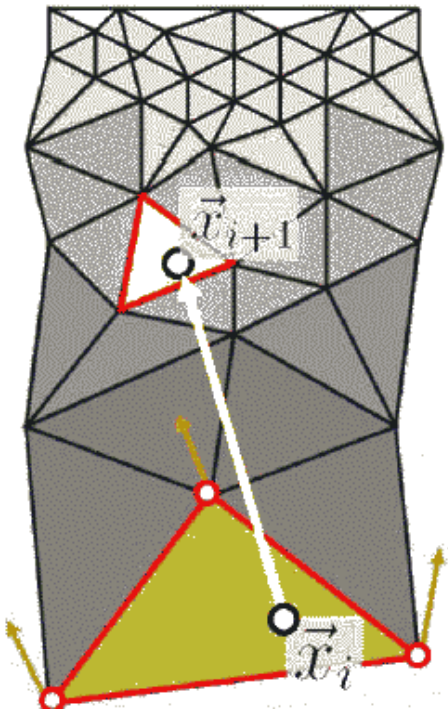
Algorithmus:





- Gegeben sei ein reguläres Dreiecksnetz

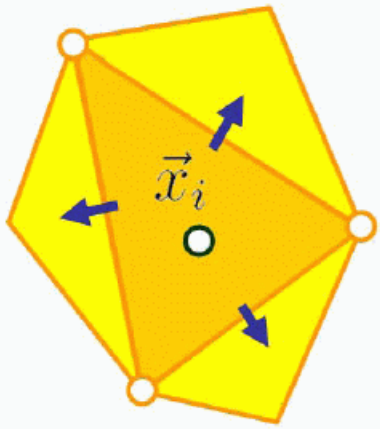
1. Wähle einen Startpunkt $\vec{x}_i$ (die Zelle sei hier bekannt)
2. Interpoliere die Geschwindigkeit (baryzentr. Koord)
3. Bestimme den Folgepunkt $\vec{x}_{i+1}$ durch numerisches Verfahren.
4. Suche die Zelle, in der der Folgepunkt liegt.

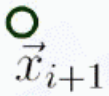


Zellsuche-Algorithmus

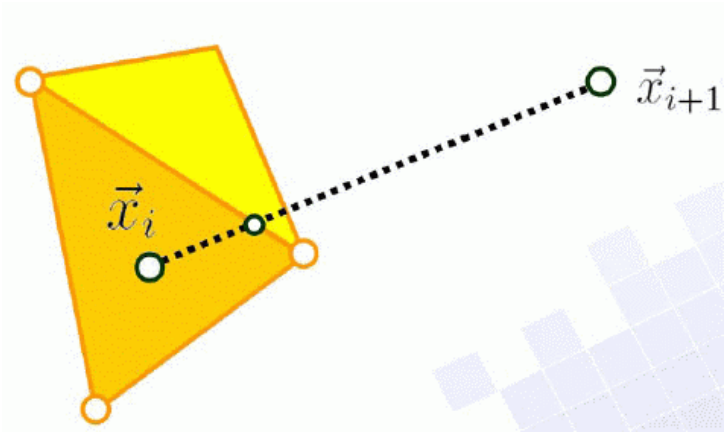
Zellsuche im Dreiecksnetz

- *Idee: Iterativer Ansatz.* Suche eines der benachbarten Dreiecke, das näher an dem Folgepunkt liegt.



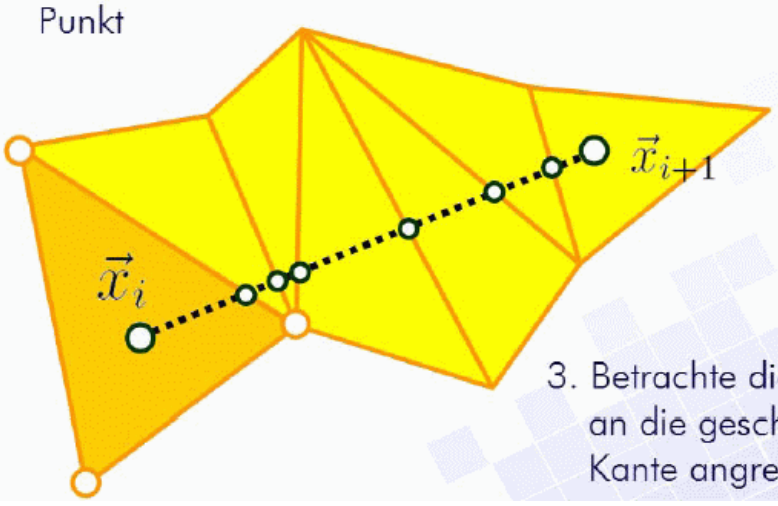


- *Voraussetzung:* Datenstruktur, die für jedes Dreieck Referenzen auf die benachbarte Dreiecke speichert *(z.B. extended winged edge)*



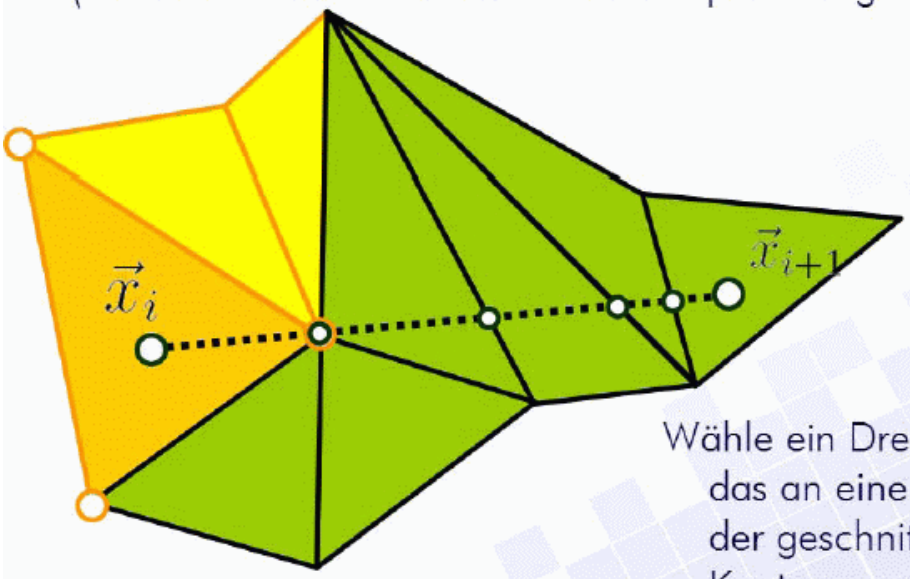
1. Betrachte die (gerichtete) Verbindungslinie zwischen dem alten und dem neuen Punkt

2. Bestimme Schnittpunkte mit den Kanten der aktuellen Zelle



3. Betrachte die Zelle die an die geschnittene Kante angrenzt.

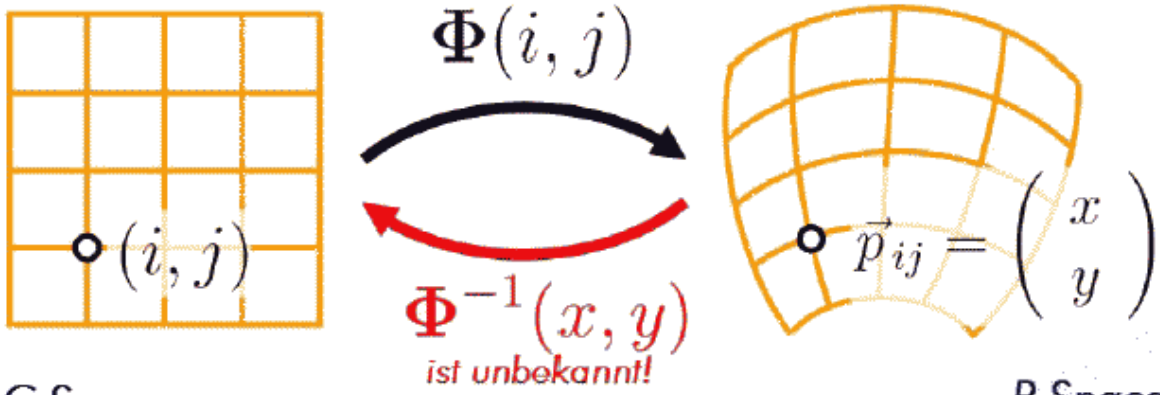
Sonderfall: Verbindungslinie geht durch einen Vertex
(d.h. zwei Kanten werden an den Endpunkten geschnitten)



Wähle ein Dreieck,
das an eine beliebige
der geschnittenen
Kante angrenzt.

Curvilineare Gitter

- Bei curvilinearen Gittern unterscheidet man zwischen *P-Space* und *C-Space*

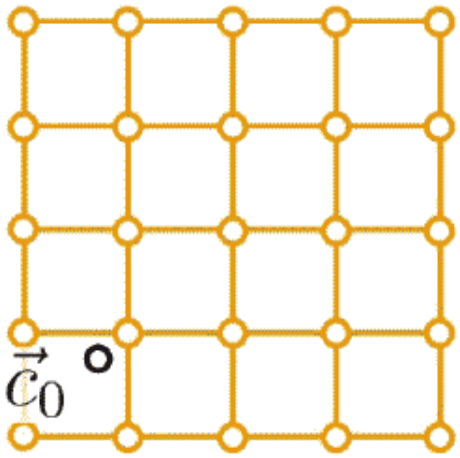


C-Space
computational space
Parameterraum
uniformes Gitter
hier kann ich leicht interpolieren

$$\Phi(i,j) = \vec{p}_{ij}$$

P-Space
physical space
physikalischer Raum
curvilineares Gitter
hier sind die Vektoren gültig

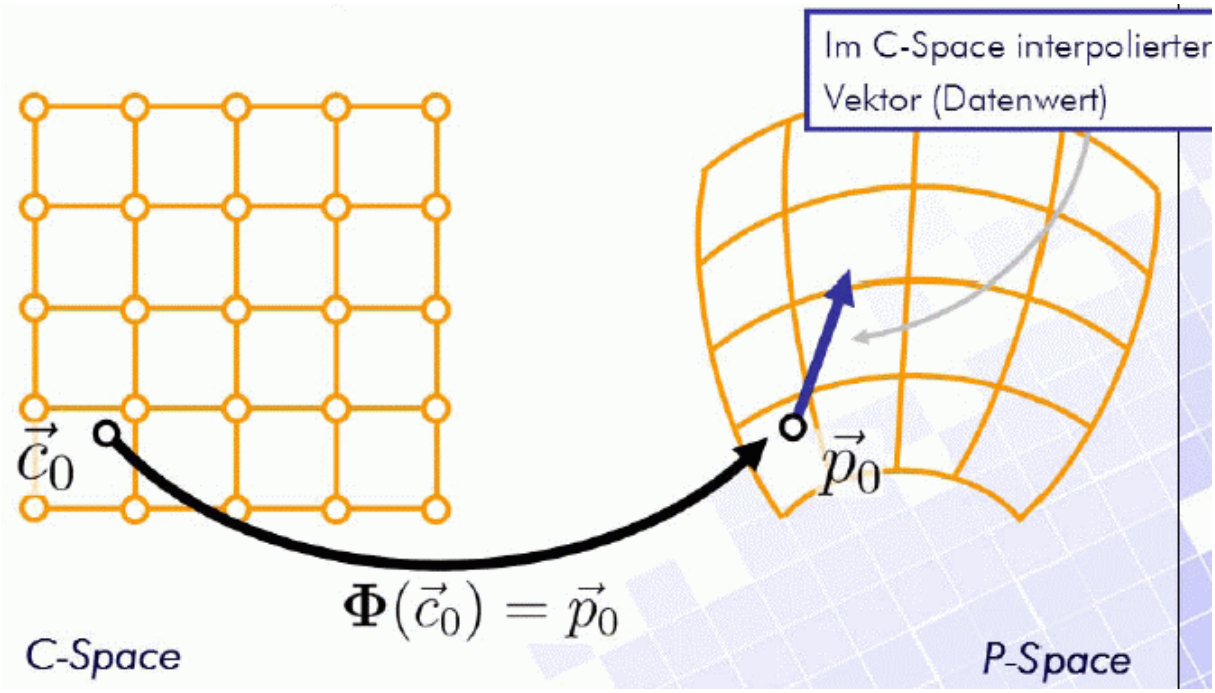
- Particle Tracing auf curvilinearen Gittern

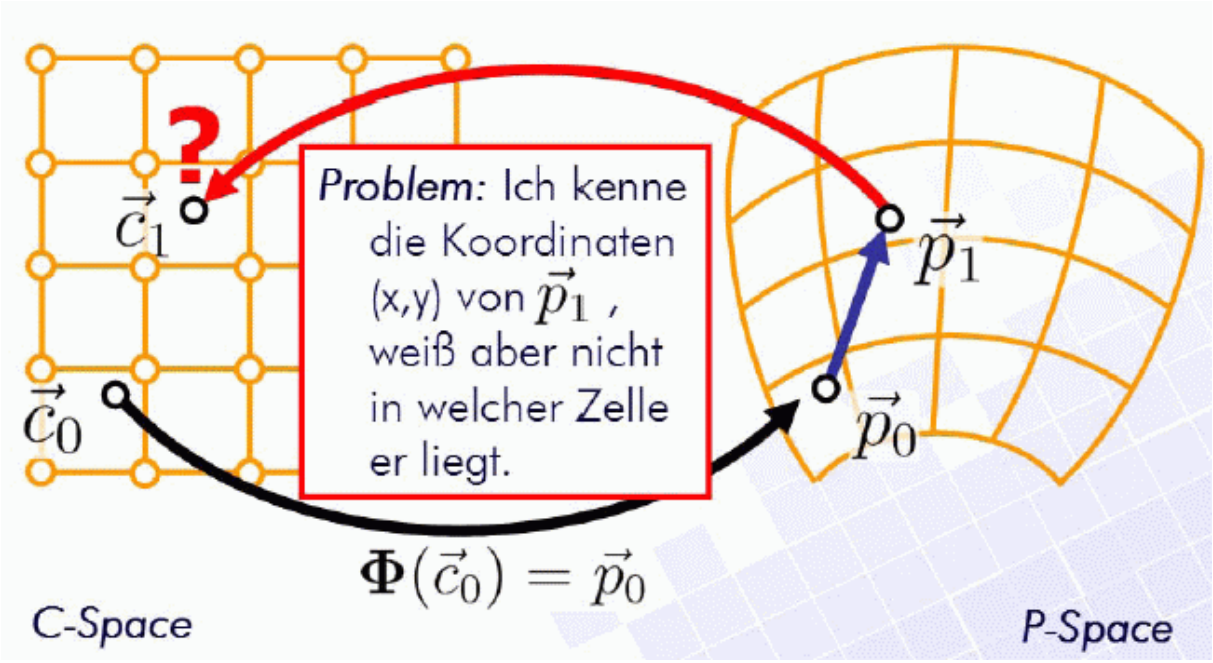


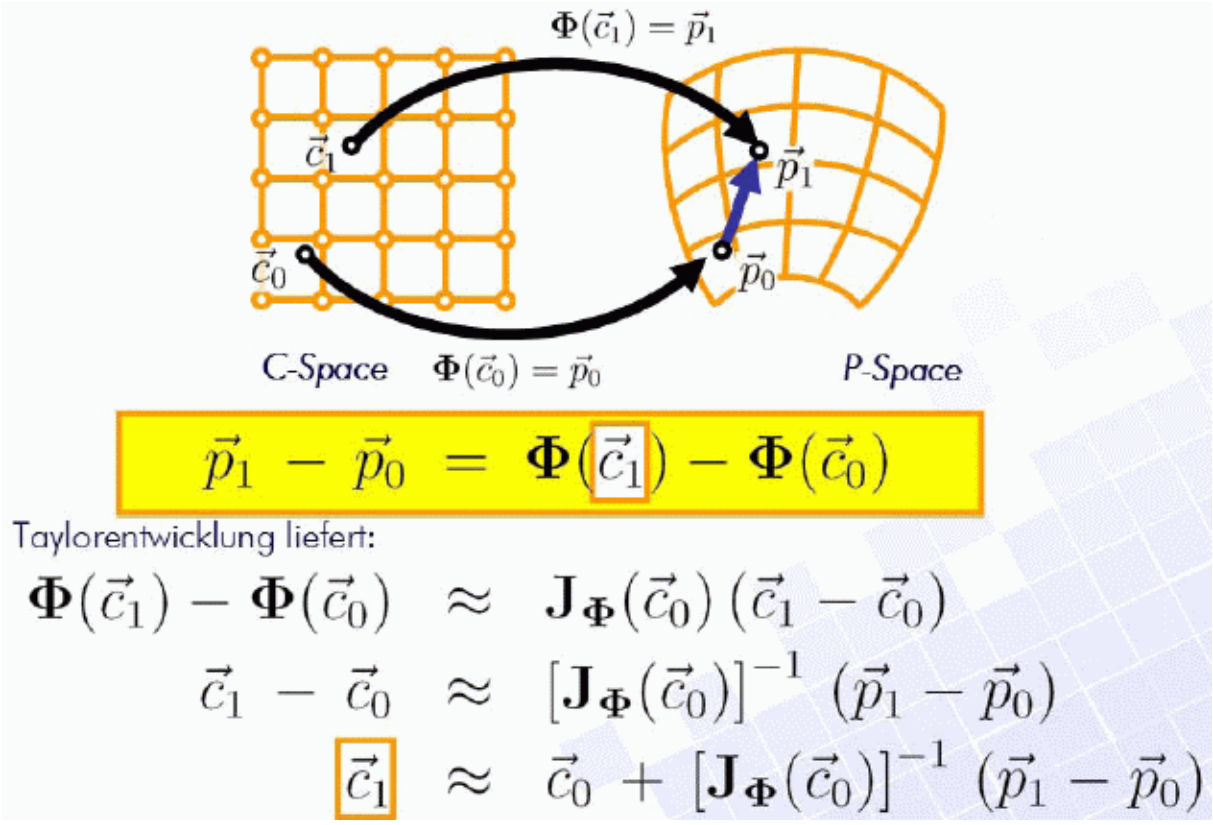
Für jeden Gitterpunkt im C-Space habe ich einen ***Datenwert*** (z.B. Geschwindigkeitsvektor) ***und*** einen dazugehörigen ***Punkt im P-Space*** gegeben.

Für einen ***beliebigen*** Punkt im C-Space kann ich demnach ***einen Datenwert und*** eine dazugehörige ***Position im P-Space*** bilinear interpolieren.

C-Space







$$\vec{p}_1 - \vec{p}_0 = \Phi(\vec{c}_1) - \Phi(\vec{c}_0)$$

Taylorentwicklung liefert:

$$\begin{aligned} \Phi(\vec{c}_1) - \Phi(\vec{c}_0) &\approx \mathbf{J}_\Phi(\vec{c}_0)\,(\vec{c}_1 - \vec{c}_0) \\ \vec{c}_1 - \vec{c}_0 &\approx [\mathbf{J}_\Phi(\vec{c}_0)]^{-1}\,(\vec{p}_1 - \vec{p}_0) \\ \vec{c}_1 &\approx \vec{c}_0 + [\mathbf{J}_\Phi(\vec{c}_0)]^{-1}\,(\vec{p}_1 - \vec{p}_0) \end{aligned}$$

liefert erste Näherung an den Punkt c_1 im C-Space

- in der Regel nicht genau genug

Fixpunkt-Iteration bis zur gewünschten Genauigkeit

$$\vec{c}_1^{(0)} \approx \vec{c}_0 + [\mathbf{J}_\Phi(\vec{c}_0)]^{-1} (\vec{p}_1 - \vec{p}_0)$$

Führe Rekursion so lange aus, bis sich das Ergebnis nicht mehr signifikant verändert:

$$\vec{c}_1^{(k+1)} \approx \vec{c}_1^{(k)} + \left[\mathbf{J}_\Phi(\vec{c}_1^{(k)})\right]^{-1} (\vec{p}_1 - \Phi(\vec{c}_1^{(k)}))$$

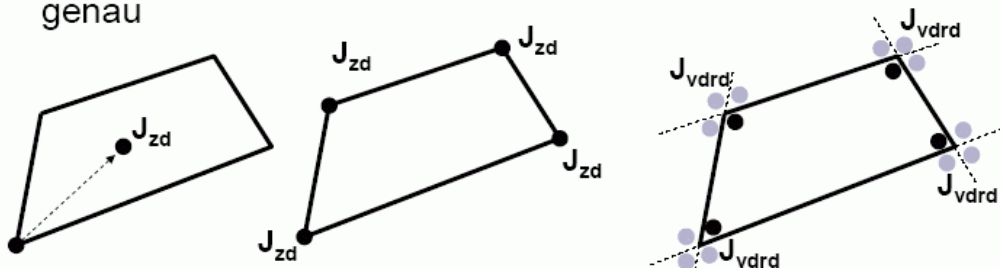
- *Fixpunkt-Iteration (Pseudo-Code):*

```
00  c = c0; p = p0; // initialize
01  do
02     dp = p1 - p;
03     dc = inverse(Jacobi(c)) * dp;
04     c = c + dc;
05     p = Φ(c);
06  until (dc < toleranz)
07  c1 = c;
```

(Rezk-Salama, o.J.)

Alternativen zur Verwendung der Jacobi-Matrix auf gebogenen Gittern:

- Per Zelle (Zentrale Differenzen): Schnell & ungenau
- Per Eckpunkt (Voxel, Zentrale Diff'): Langsam & genauer
- Vor- & Rückwärtsdifferenzen an Eckpunkten: Aufwendig & genau



Zusammenfassung

Partikelverfolgung im P-Space:

Wähle Startpunkt $\boldsymbol{x}_{i=0}$

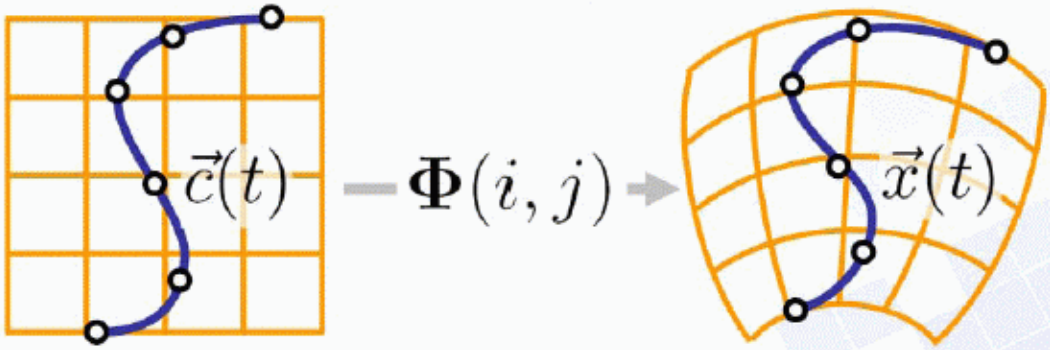
Solange $\boldsymbol{x}_i$ in gültigem Bereich

- Finde Zelle, die $\boldsymbol{x}_i$ enthält, und α_i
- Berechne $\boldsymbol{v}(\boldsymbol{x}_i)$
- Integriere zu $\boldsymbol{x}_{i+1}$
- Schrittweitenanpassung
- Zeichne Segment $(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})$ oder Kügelchen
- $i \leftarrow i+1$

(Bartz 2005)

C-Space Verfahren:

- Zellsuche im curvilinearen Gitter kann ziemlich aufwändig werden
- *Alternative: Particle Tracing im C-Space*



Jeder Partikelbahn $\vec{x}(t)$ im P-Space entspricht eine Partikelbahn $\vec{c}(t)$ im C-Space.

Idee: Bestimme die Partikelbahn im C-Space und transformiere die Punkte zurück in den P-Space

- Wie transformiere ich die Geschwindigkeits-vektoren in den C-Space?

Geschwindigkeitsvektoren sind im P-Space gegeben:

$$\frac{\partial \vec{x}(t)}{\partial t} = \vec{v}(\vec{x}(t))$$

Transformation vom C-Space in den P-Space:

$$\vec{x}(t) = \mathbf{\Phi}(\vec{c}(t))$$

nachdifferenzieren (Kettenregel)

Differenzieren:

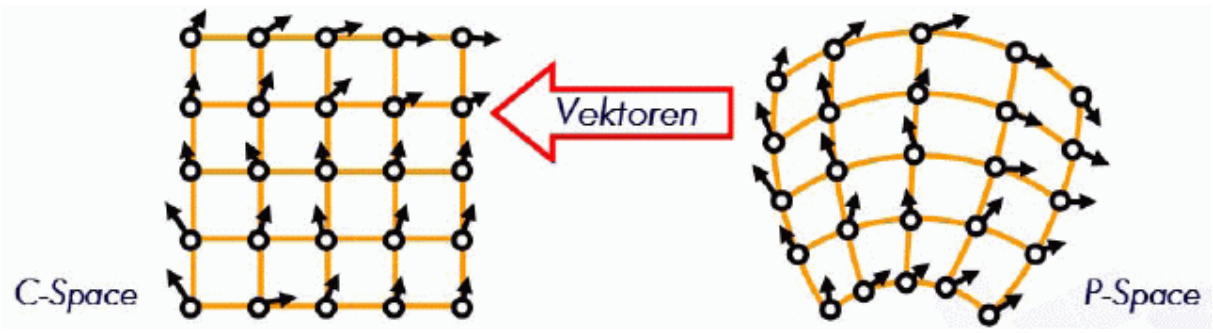
$$\frac{\partial \vec{x}(t)}{\partial t} = \mathbf{J}_{\mathbf{\Phi}}(\vec{c}(t)) \frac{\partial \vec{c}(t)}{\partial t}$$

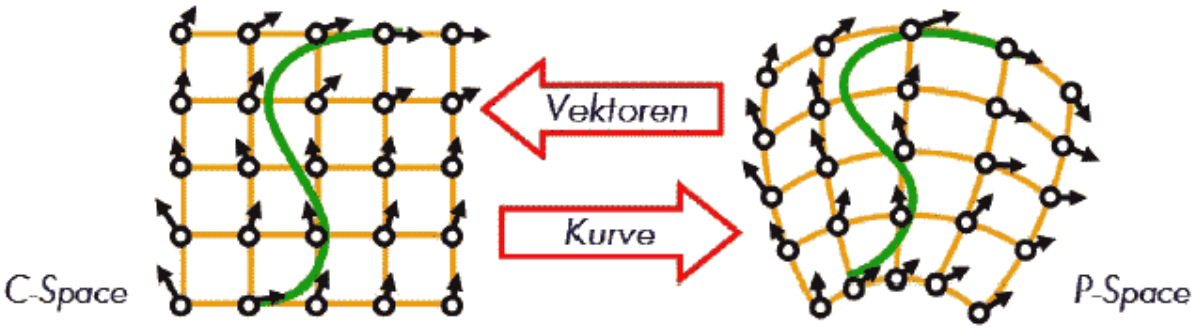
$$[\mathbf{J}_{\mathbf{\Phi}}(\vec{c}(t))]^{-1} \frac{\partial \vec{x}(t)}{\partial t} = \frac{\partial \vec{c}(t)}{\partial t}$$

Differentialgleichung im C-Space:

$$\frac{\partial \vec{c}(t)}{\partial t} = \tilde{\vec{v}}(\vec{c}(t))$$

$$\tilde{\vec{v}}(\vec{c}(t)) = [\mathbf{J}_{\mathbf{\Phi}}(\vec{c}(t))]^{-1} \vec{v}(\mathbf{\Phi}(\vec{c}(t)))$$





Differentialgleichung im C-Space:

$$\frac{\partial \vec{c}(t)}{\partial t} = \tilde{\vec{v}}(\vec{c}(t))$$

$$\tilde{\vec{v}}(\vec{c}(t)) = [\mathbf{J}_{\mathbf{\Phi}}(\vec{c}(t))]^{-1}\, \vec{v}(\mathbf{\Phi}(\vec{c}(t)))$$

Zusammenfassung

P-Space Verfahren

- Interpoliere im C-Space
- Transformiere den Punkt in den P-Space
- Numerischer Integrationsschritt im P-Space
- Zellsuche zur Bestimmung des Folgepunktes im C-Space

C-Space Verfahren

- Transformiere die Vektoren in den C-Space
- Bestimme Partikelbahn durch num. Integration im C-Space
- Transformiere die Ergebnis-Kurve zurück in den P-Space

- Jacobi-Matrix wird nur für Zellsuche verwendet
- **Höhere Genauigkeit**

- Vektoren werden transformiert (ungenau)
- **Einfach und schnell**

Zeitabhängige Daten

- *Bisher:* Particle Tracing in zeitlich konstanten (statischen) Vektorfeldern:

$$\frac{\partial \vec{x}(t)}{\partial t} = \vec{v}(\vec{x}(t))$$

- *Jetzt:* Particle Tracing in zeitabhängigen (dynamischen) Vektorfeldern:

$$\frac{\partial \vec{x}(t)}{\partial t} = \vec{v}(\vec{x}(t), t)$$

Was ändert sich an den numerischen Verfahren?

Zeit muss als zusätzlicher Parameter mitgeführt werden.

z.B. Eulerschritt $t_{i+1} = t_i + \tau$

$$\vec{x}(t_{i+1}) = \vec{x}(t_i) + \tau\,\vec{v}\left(\vec{x}(t_i), t_i + \tau\right)$$

(Rezk-Salama, o.J.)

Charakteristische Linien L(u)

- Stromlinien (Streamlines):
  Tangential zum Vektorfeld
- Bahnlinien (Pathlines):
  Masseloses Partikel im Strom
- Streichlinien (Streaklines):
  Farbausfluss aus fester Position über die Zeit
- Zeitlinien (Timelines):
  Zeitentwicklung einer masselosen Linie

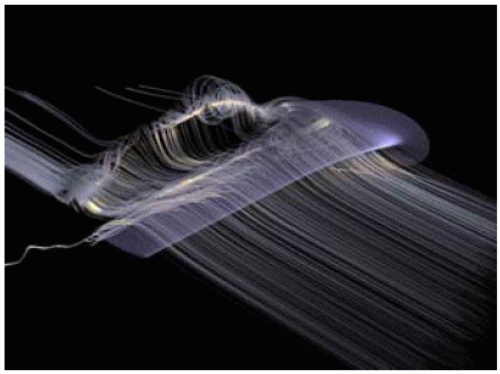
Bahnlinien (Pathlines):

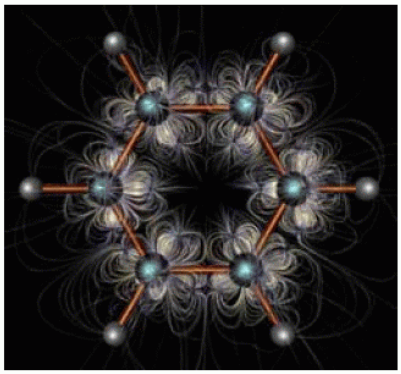
- Bahnen masseloses Partikel im Strom
- Zeitabhängiges (ggf. instabiles) Vektorfeld
- Auch Lösung zum Anfangswertproblem einer gewöhnlichen DGL

$$\boldsymbol{L}(0) = \boldsymbol{x}_0 \qquad \frac{\partial \boldsymbol{L}(u)}{\partial u} = \boldsymbol{v}(\boldsymbol{L}(u), u)$$

Liniendarstellung

Pathline





(Bartz 2005)

auch: "Pfadlinie".

- *Pathlines (Pfadlinie)*: Die Bahn eines Partikels im zeitabhängigen Vektorfeld

$$\frac{\partial \vec{x}(t)}{\partial t} = \vec{v}(\vec{x}(t), t) \qquad \vec{x}_0 = \vec{x}(t_0)$$

Experimentelle Bestimmung: „Injiziere zum Teitpunkt t_0 einen einzigen Tropfen Farbstoff in die Strömung und mache ein Foto mit langer Belichtungszeit."

- *Streamlines (Feldlinie)*: zeitlicher Schnappschuß des Vektorfeldes

$$\frac{\partial \vec{x}(t)}{\partial t} = \vec{v}(\vec{x}(t), T) \qquad T = const.$$

Experimentelle Bestimmung: nicht möglich, da die Zeit nicht abgehalten werden kann.

(Rezk-Salama, o.J.)

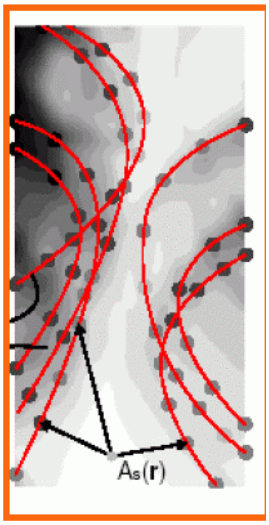
Betrachtungsweisen

Euler

- Auf Gitterknoten liegen Eigenschaften vor
- Keine individuelle Partikel
- Position nur implizit
- Streamlines
- „Boje"

Lagrange

- Betrachte individuelle Partikel
- Haben Eigenschaften wie Geschwindigkeit, Position, etc.
- Particle-Tracing
- „Schiffchen"



(Bartz 2005)

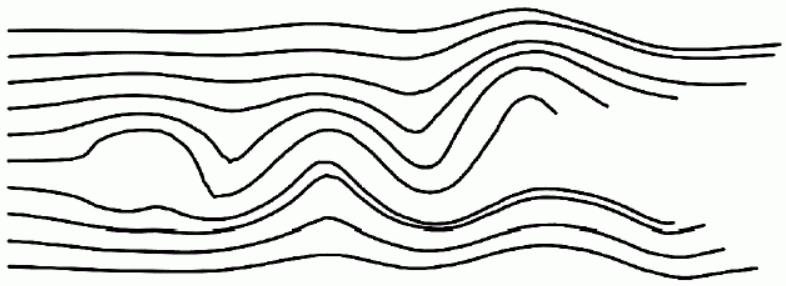
Stream line: dt. "Stromlinie"

- Integration entlang des Vektorfeldes.

$$\underline{x}(T) = \underline{x}(0) + \int_{s=0}^{T} \underline{v}(\underline{x}(s), t)$$

  - Laufvariable s nicht die Zeit!
  - Vektorfeld an konstantem Zeitpunkt!
- Stromlinien sind überall tangential zu $\underline{v}(\underline{x},t)$

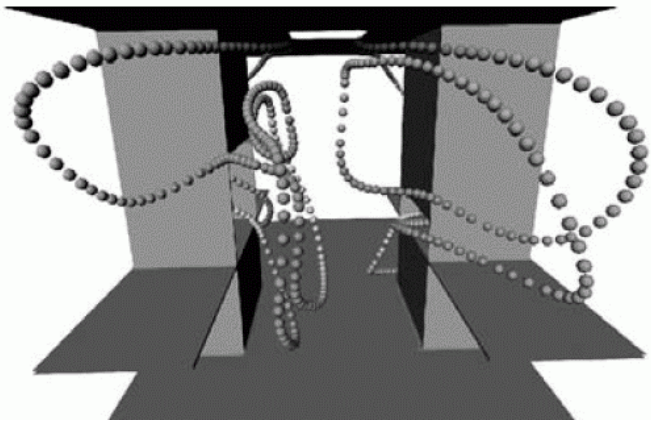
- Bei konstantem Vektorfeld (z.B. stationärer Strömung): alle gleich!



(Weimar 2005)

Streamlines als Kügelchen

- Radius kodiert weiteren skalaren Wert



Stromlinien (Streamlines):

- Tangential zum Vektorfeld
- Vektorfeld an einem beliebigen, aber festen Zeitschritt
- Streamline ist Kurve L(u) mit Parameter u
- Lösung zum Anfangswertproblem einer gewöhnlichen DGL

$$\boldsymbol{L}(0) = \boldsymbol{x}_0 \qquad \frac{\partial \boldsymbol{L}(u)}{\partial u} = \boldsymbol{v}(\boldsymbol{L}(u), t)$$

Anfangswert an Saatpunkt x_0 — **Gewöhnliche DGL**

(Bartz 2005)

- Wenn div(v) = 0 ist
  - Divergenzfreies Feld, inkompressible Strömung
- Dann kann das Feld v aus einem Skalarfeld Ψ berechnet werden:
  - v = rot Ψ
- Die Stromlinien sind dann Konturlinien von Ψ
- Effiziente Algorithmen zur Berechnung von Konturlinien (Marching Squares).
- Nur in 2-D!

(Weimar 2005)

Charakteristische Linienscharen L(u,t)

- **Streichlinien (Streaklines):**
  - Farbausfluss aus fester Position über die Zeit

$$\boldsymbol{L}(0,t) = \boldsymbol{x}_0 \quad \boldsymbol{L}(u+\partial u, t+\partial t) - \boldsymbol{L}(u,t) = \partial t \cdot \boldsymbol{v}(\boldsymbol{L}(u,t),t)$$

- **Zeitlinien (Timelines/Surface):**
  - Zeitentwicklung einer masselosen Linie (Oberfläche)
  - Besteht aus vielen Partikeln, die verfolgt werden

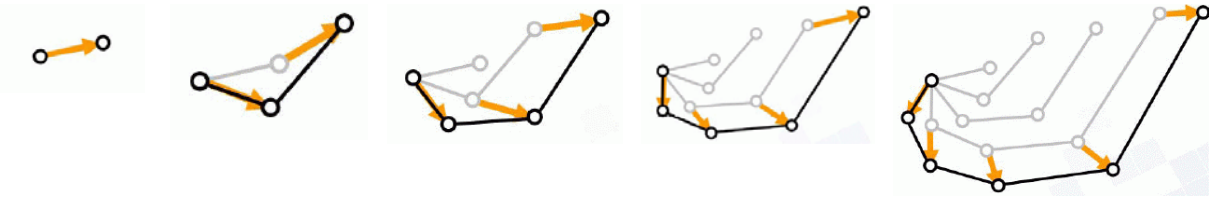
$$\boldsymbol{L}(u,0) = \boldsymbol{L}_0(u) \quad \frac{\partial \boldsymbol{L}(u,t)}{\partial t} = \boldsymbol{v}(\boldsymbol{L}(u,t),t)$$

(Bartz 2005)

- *Streak Line:* Die Positionen von Partikeln, die zu verschiedenen Zeitpunkten am gleichen Ort gestartet sind.

$$\frac{\partial \vec{x}(t)}{\partial t} = \vec{v}(\vec{x}(t),t) \qquad \vec{x}_0 \text{ konstant}, \; t_0 \text{ variabel}$$

Experimentelle Bestimmung: „Injiziere an einem bestimmten Punkt kontinuierlich Farbstoff in die Strömung"



(Rezk-Salama, o.J.)

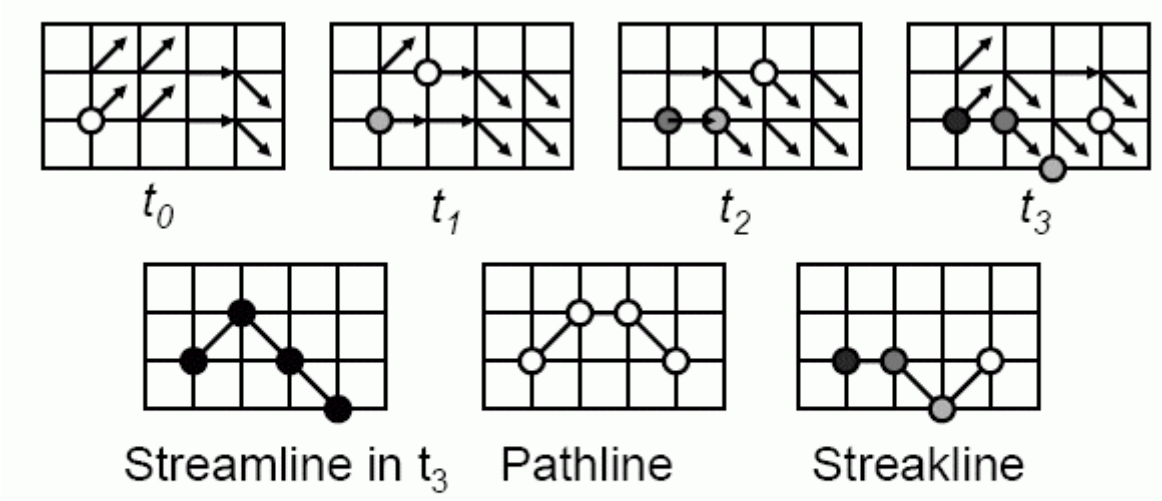
- Streichlinie: Menge der aktuellen Position aller Teilchen, die jemals durch $\underline{x}_0$ geflossen sind.
  - Rauch kontinuierlich eingefügt.
  - ∃ T : Vor T Zeiteinheiten war das Teilchen in $\underline{x}_0$

$$\left\{\underline{x}\in\mathbb{R}^2 \mid \exists T : \underline{x}(T) = \underline{x}(0)+\int_{t=0}^{T} \underline{v}(\underline{x}(t),t)\right\}$$

  - Implementierung: Teilchen einfügen, verfolgen, verbinden.

(Weimar 2005)

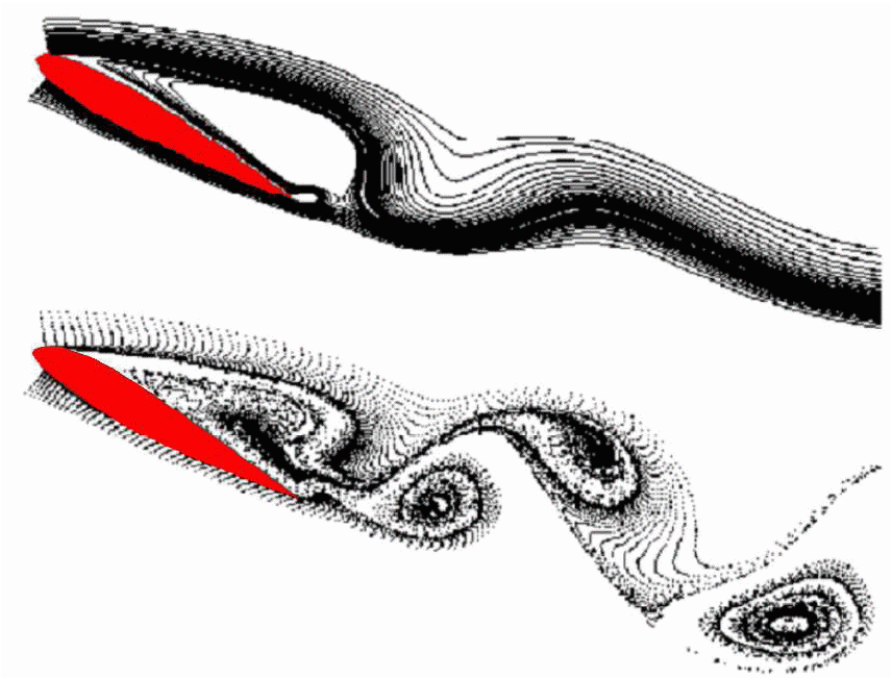
einfaches Beispiel für charakteristische Linien:



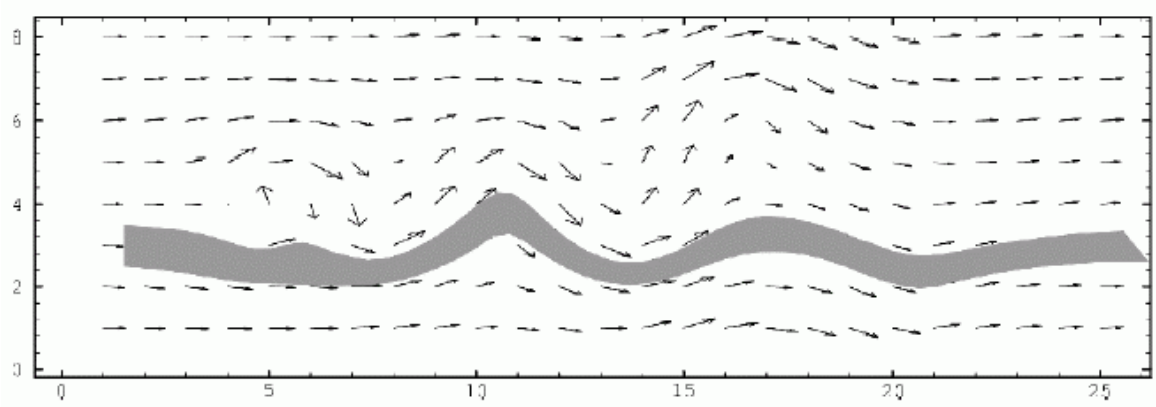
Für stationäre Strömungen sind Streamlines, Pathlines und Streaklines dasselbe.

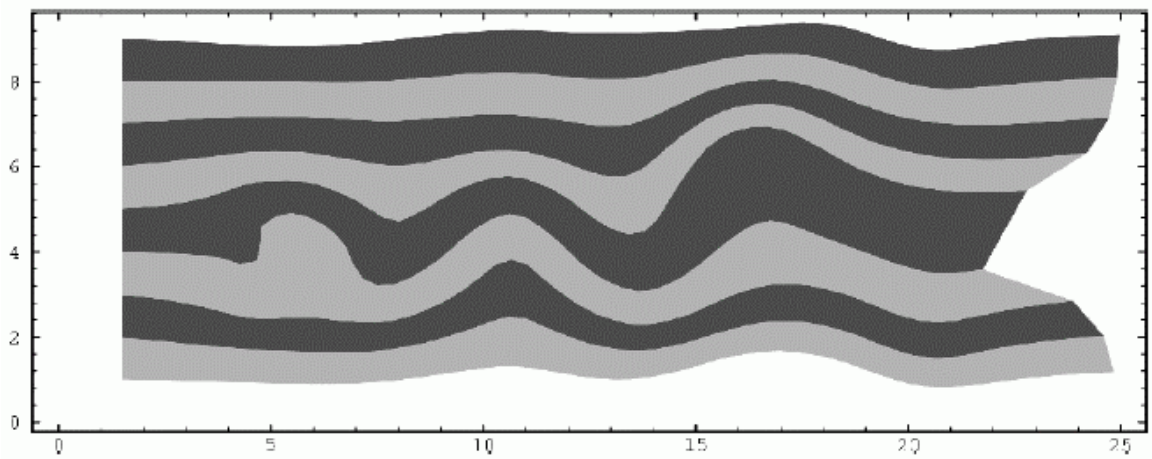
(Bartz 2005)

Stromlinien vs. Streichlinien:



Verallgemeinerung von Stromlinien zu *Strombändern*:





(Weimar 2005)

Zusammenfassung

Partikelbahnen

- *Wahl der Startpunkte* (seed point algorithms)
- *Zellsuche* im unstrukturierten und curvilinearen Gitter
- *C-Space-* und *P-Space* -Verfahren
- Zeitabhängige Daten:
  Streamlines, Streaklines & Pathlines

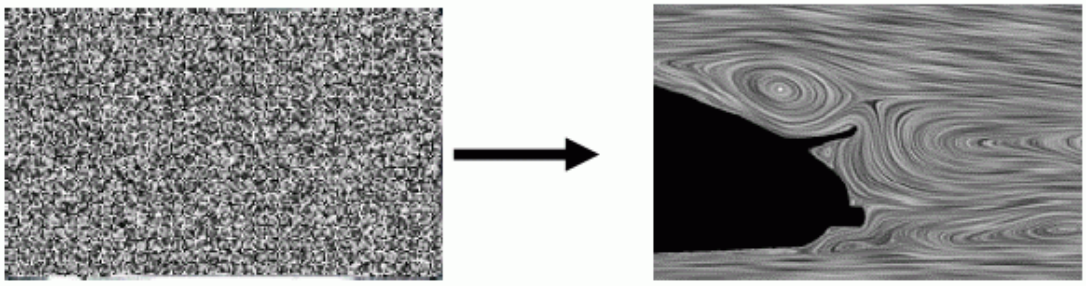
(Rezk-Salama, o.J.)

Line Integral Convolution (LIC):

- Globale Visualisierungsmethode

Idee:

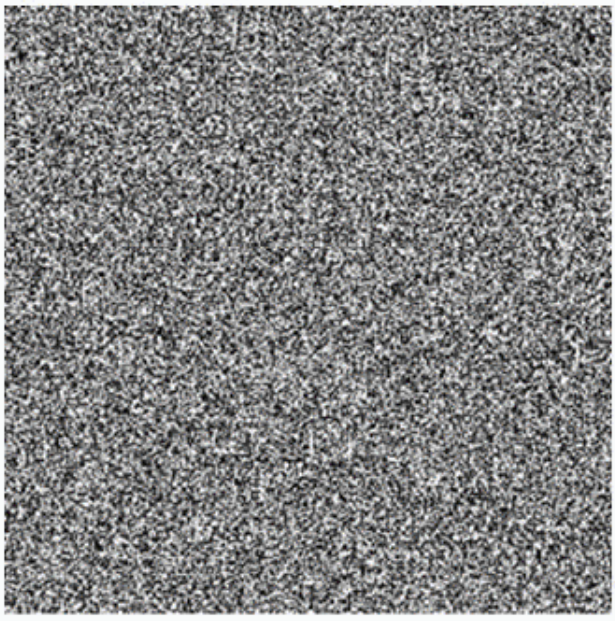
- Starte mit Zufallstextur
- Verschmiere entlang von Streamlines



(Bartz 2005)

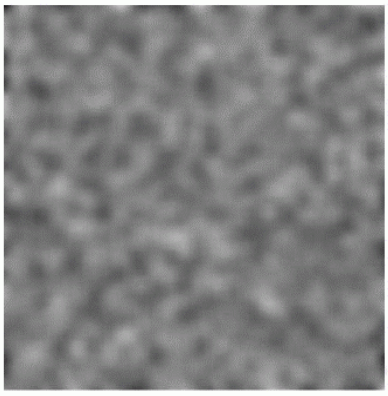
"Zufallstextur" = "Rauschen"

- *Ein Bild (2D Skalarfeld) mit zufälligen (Grau-) Werten bezeichnet man als „Rauschen“*

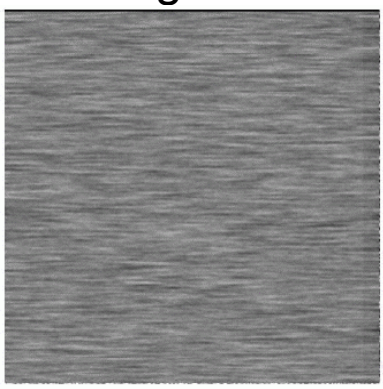


Signalverarbeitung:
In Rauschen sind ***alle*** Frequenzen eines bestimmten Frequenzbandes enthalten

Bearbeitung des Rauschens mit einem Glättungsfilter:

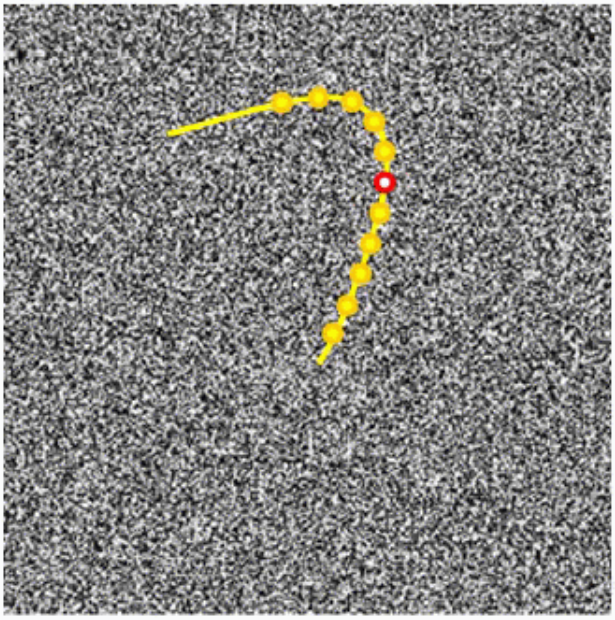


Glättung nur ein eine Richtung (hier: x-Richtung):

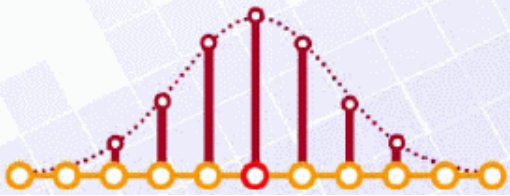


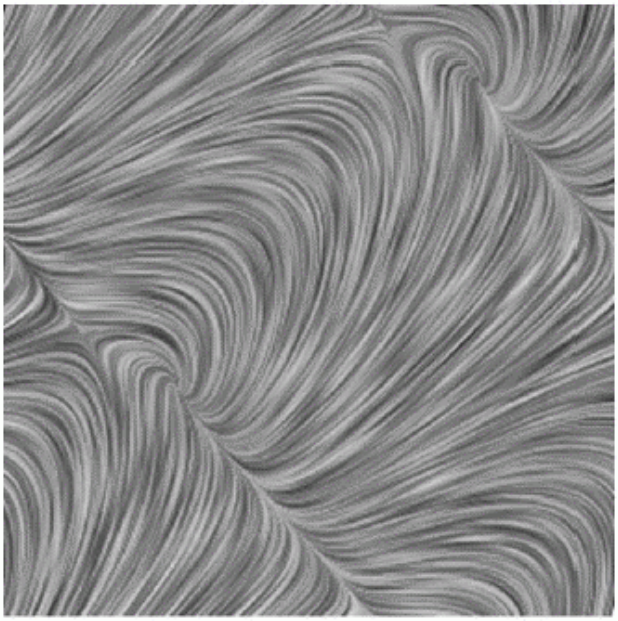
Line Integral Convolution

Idee: Filtere ein 2D Rauschen entlang der Feldlinien eines 2D Vektorfeldes



- Bestimme für jeden Pixel des Ergebnisbildes eine kurze Partikelbahn (vor- und rückwärts)
- Betrachte die Werte des Rauschbildes an den Positionen
- Filtere die Werte mit einem Glättungsfilter

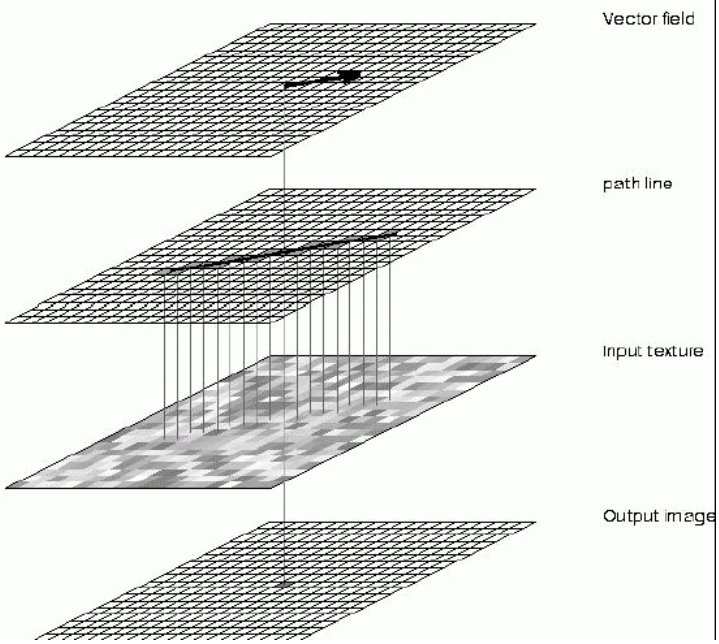




(Rezk-Salama, o.J.)

LIC Algorithmus:

- Falte Texture entlang der Streamlines



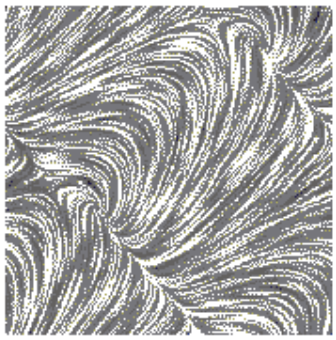
(Bartz 2005)

Line-Integral: Linienintegral = Partikelbahn

Convolution: Faltung = Linearer Filter

„Faltung entlang der Linienintegrale"

In der Praxis: Box-Filter (Mittelung)



Vorteil:

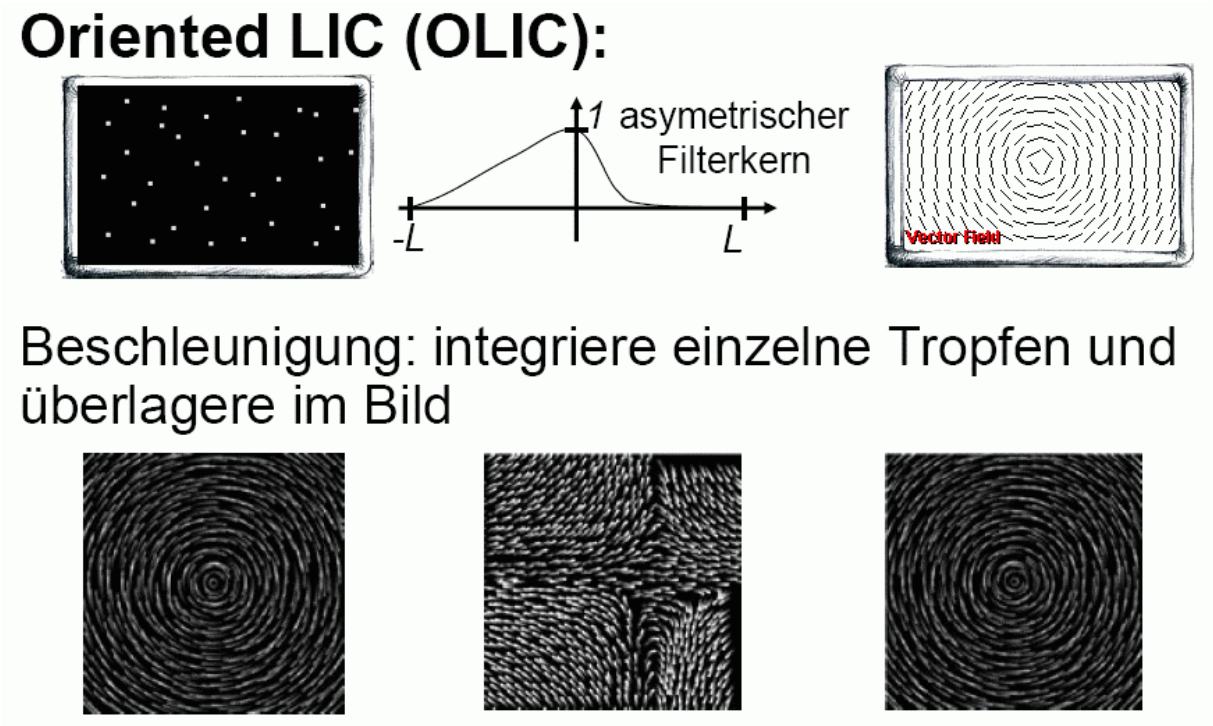
- Hohe Dichte an Partikelbahnen,
- sehr intuitiv verständlich

Nachteil:

- Relativ hoher Rechenaufwand
- Kein Absolutbetrag der Geschwindigkeit!

(Rezk-Salama, o.J.)

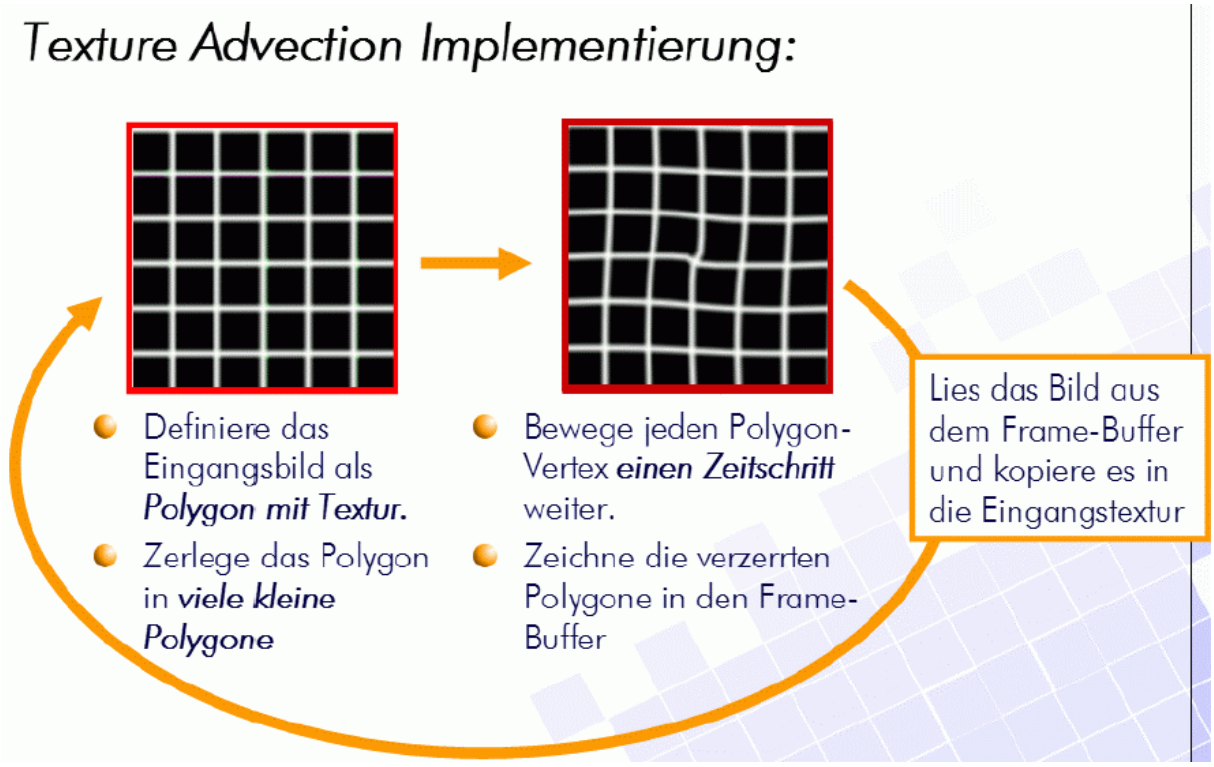
Variante:



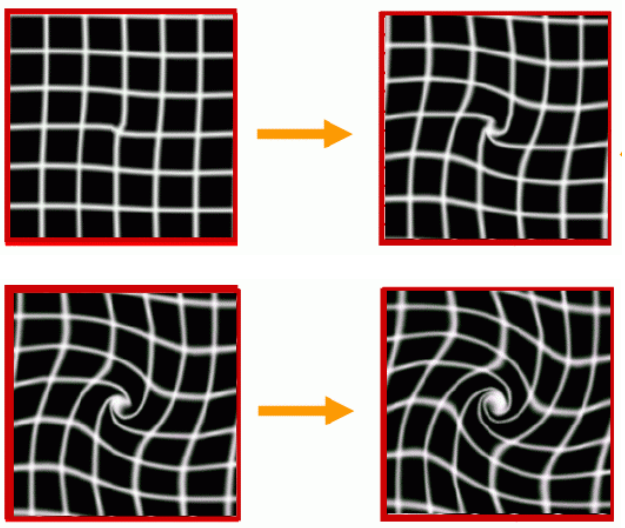
(Bartz 2005)

Texture Advection:

Verzerre ein Eingabebild (Textur) entlang der Strömungslinien des Vektorfeldes.



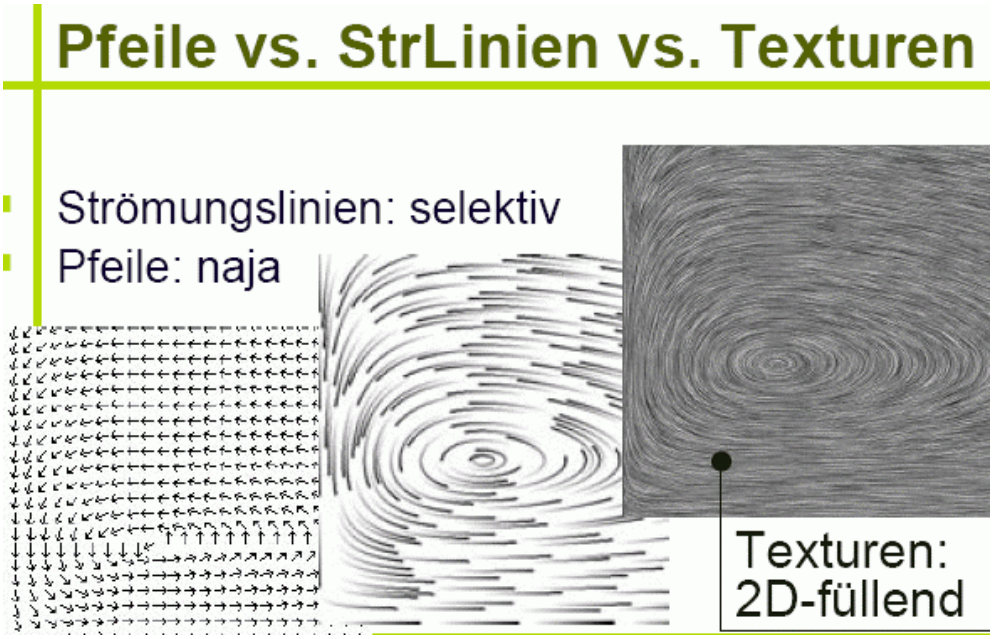
nächste Schritte:



Zusammenfassung:

- *Texturbasierte Verfahren*
  - *Line Integral Convolution* (LIC)
    Faltung eines Rauschfeldes entlang der Strömungslinien eines Vektorfeldes
  - *Texture Advection*
    Bildbasierter Ansatz: Verzerre ein Ergebnisbild schrittweise in Richtung der Strömung

(Rezk-Salama, o.J.)



(Hauser 2007)