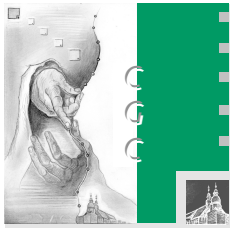
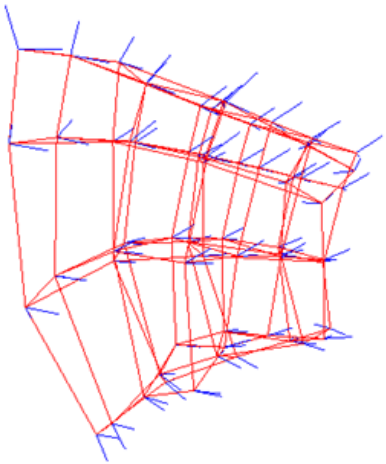


Virtuelle Realität
Darstellungsfehler und Korrektur



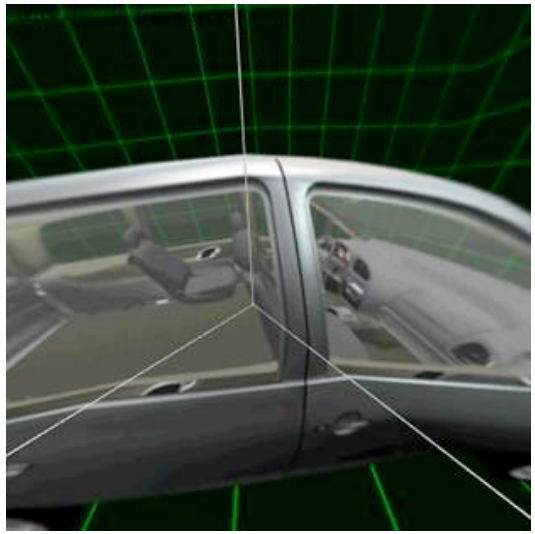
G. Zachmann

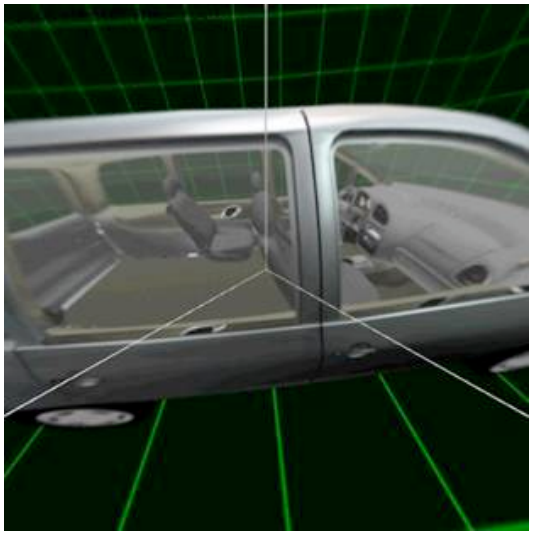
Clausthal University, Germany

cg.in.tu-clausthal.de

Auswirkungen falscher Kamera-/Handpos.

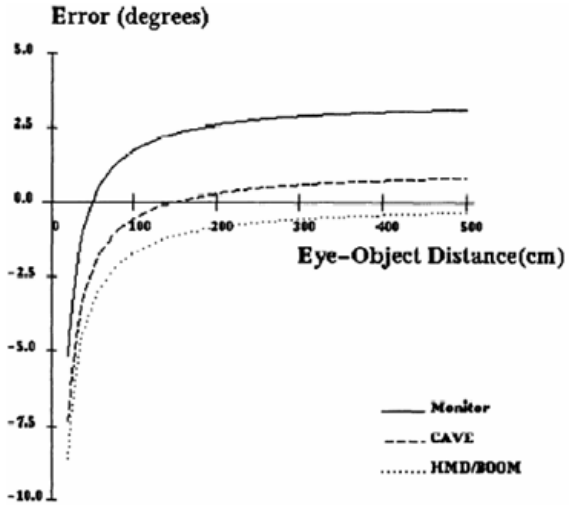
- Cave: Bildverzerrung , "Schwanken" (wg. Latenz)
- HMD: präzise Manipulation/Positionierung wird erschwert
- Anwendungsbeispiele, wo das wichtig ist:
  - Styling review, ...
  - Einbausimulation, Ergonomieuntersuchung, ...



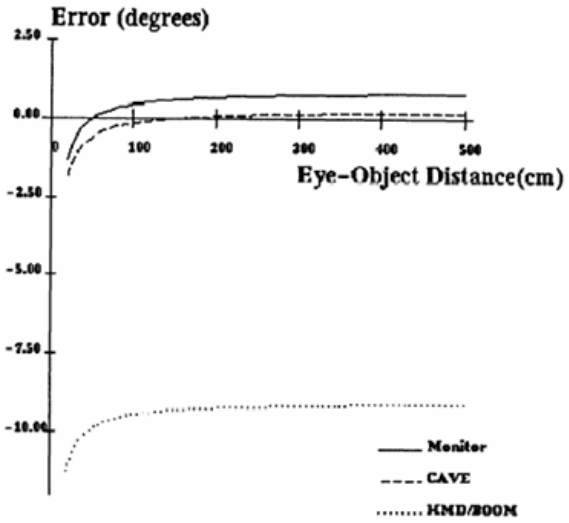


Verschiedene Auswirkung bei versch. Displays

- *Translational camera displacement*: HMD normalerweise besser
- *Rotational camera displacement*: Cave besser
- Problem in der Cave: statische und dynamische Bildverzerrung
- Problem im HMD: Propriozeption widerspricht visueller Wahrnehmung



Angular error, head displacement



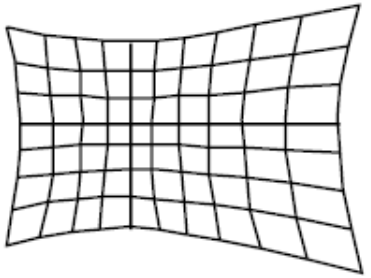
Angular error, head rotation

Fehlerquellen

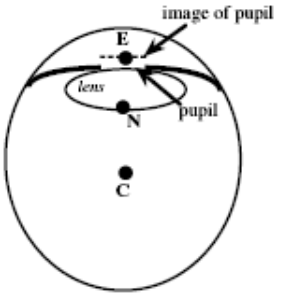
1. Physikalische Fehlerquellen:
   1. Verzögerung (*Delay*, Latenz, Lag)
   2. *Tracking-System*
   3. *Transformations-Pipeline*
   4. Sonstige kleinere Quellen
2. Psychophysiologische Quellen:
   1. Berichte von Usern in der Cave (*Daimler*)
   2. Unerforscht! ...

Kleinere Fehlerquellen

- Optische Verzerrung durch das Display
  - Mögliche Lösung:
    2x rendern mit Texturverzerrung

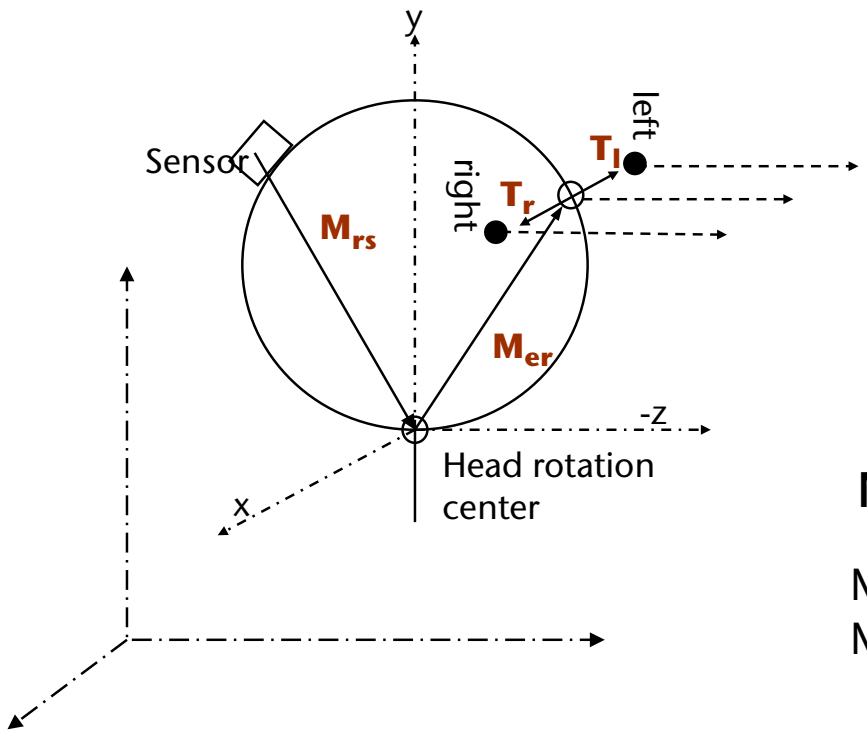


- Augen-Tracking?
  - Fehler ist vernachlässigbar, falls Projektionszentrum = Augenzentrum gewählt wird



Transformationsfehler

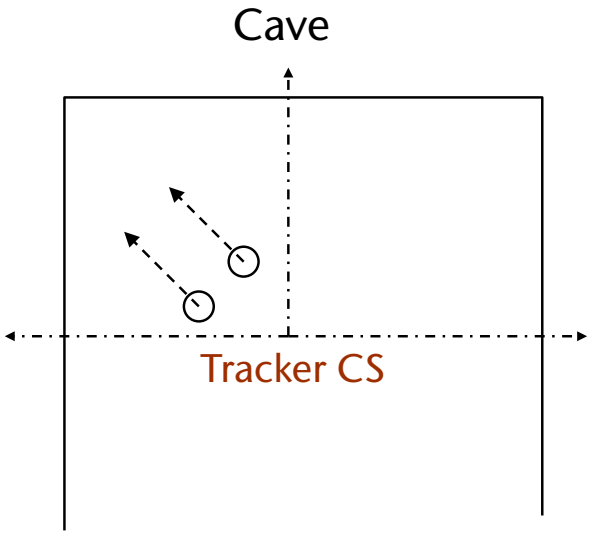
- Fehler in der Bestimmung des User-Modells

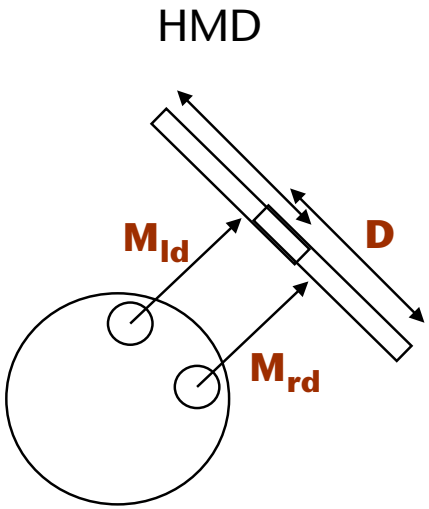


$$M_e = T_r M_{er} M_{rs} M_s$$

M_s = aktuelle Sensorposition
M_e = Viewpoint-Transf.

- Wo ist das Display / die Projektionsebene?



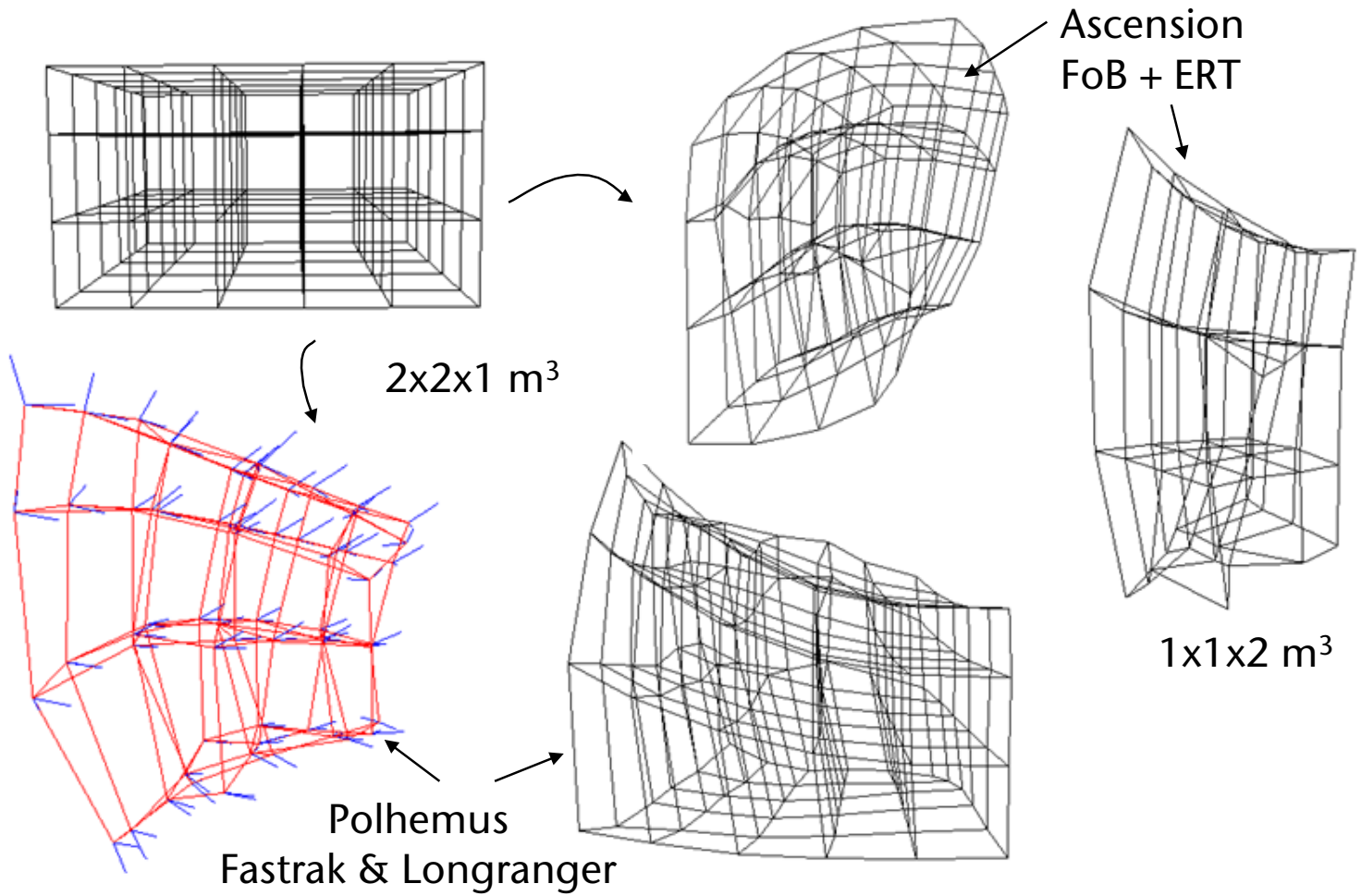


M_{*d} = Transf. vom linken/rechten Auge zum Display
D = Display-Geometrie

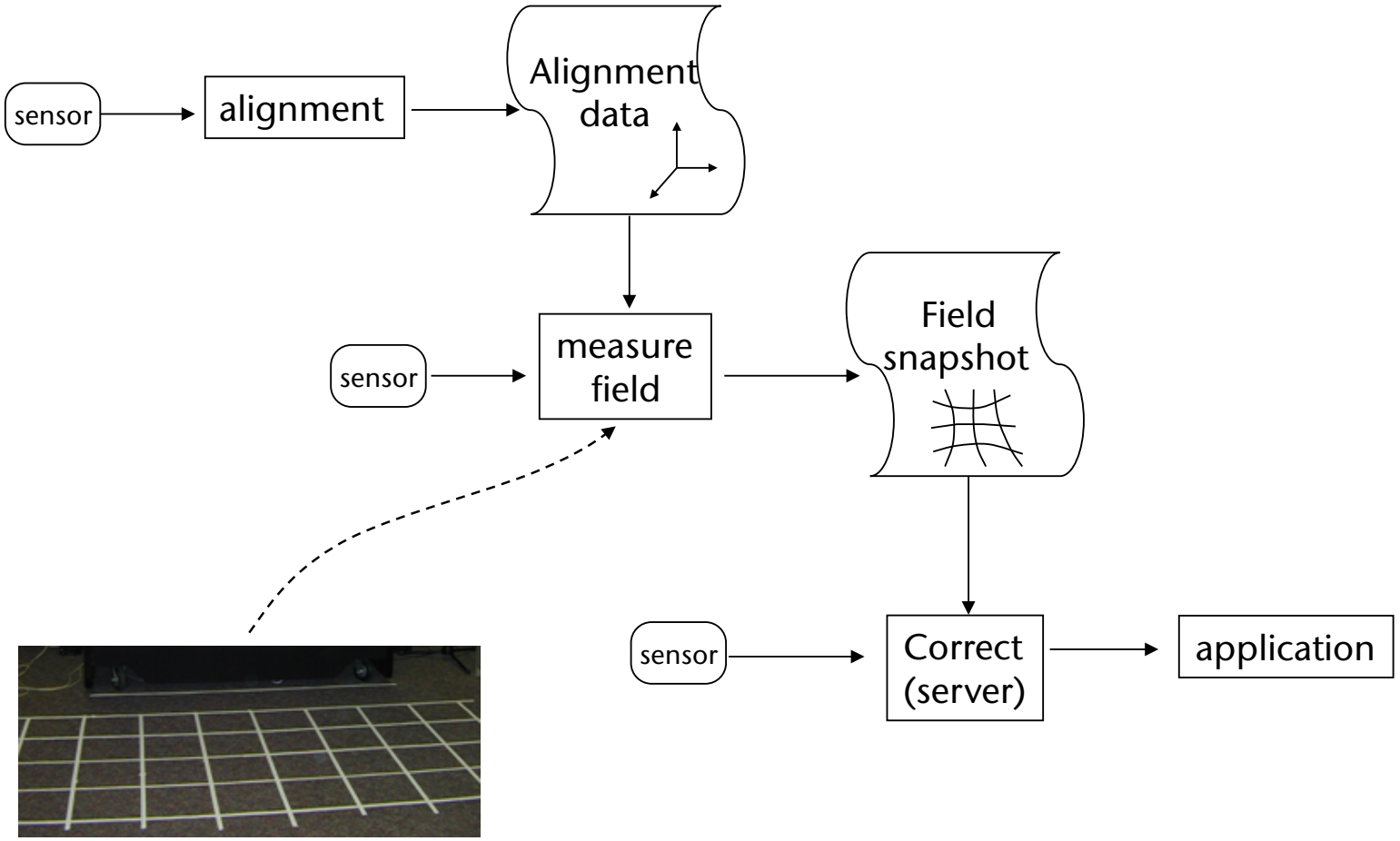
Tracking-Korrektur

- Static:
  - Falsches Alignment des Koord.systems
  - Verzerrungen
- Dynamic:
  - Rauschen
  - Drift
  - Drop-outs

Verzerrung (z.B. durch Fremdmetall beim elektrom. Tr'ing)



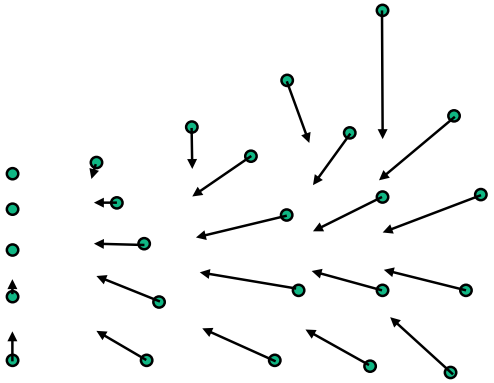
Allgemeine Prozedur



- Problem: Interpolation
- Ansätze:
  - Polynomiale Interpolation/Approximation (normal, Lagrange, ..)
  - B-Spline-Volumen
  - Shape functions (*scientific visualization*)
  - *Look-up tables*
  - Radiale Basisfunktionen (*Hardy's Multi-Quadric*)

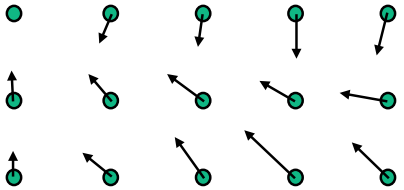
Lookup table

- Gegeben: Meßwerte und Sollwerte

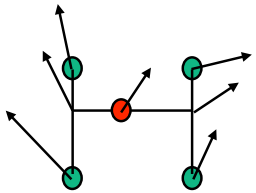


- *Resampling* der Korrekturvektoren in reguläres Gitter mit Gauß-Kernel:

$$v_Q = \sum_p v_P e^{-\frac{|P-Q|^2}{h^2}}$$



- Korrektur zur Laufzeit: trilineare Interpolation der Korrekturvektoren



- Test, ob Gitter fein genug: berechne mittels *Lookup-Table* Korrekturvektoren für echte Messwerte, vergleiche mit echten Korrekturvektoren

Hardy's Multi-Quadric (HMQ)

- Zunächst Korrektur der Translation mit $f : \mathbb{R}^3 \rightarrow \mathbb{R}^3$
- Ansatz:

$$f(P) = \sum A_i \omega_i(P), \quad A_i \in \mathbb{R}^3$$

$$\omega_i(P) = \left[(P - P_i)^2 + R^2 \right]^{\mu}$$

radiale Basisfunktionen

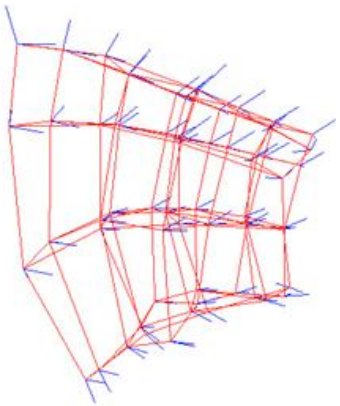
$$\mu = \frac{1}{2}, \frac{1}{4}, 2, -\frac{1}{2}, ||$$

- Einsetzen:

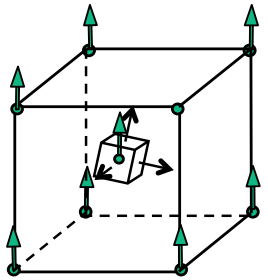
$$\mathrm{f}(\mathrm{P}_j) = Q_j, \quad j = 1, ..., N$$

- liefert 3 LGS:

$$\begin{pmatrix} \omega_i(P_1) & \cdots & \omega_N(P_1) \\ \vdots & \ddots & \vdots \\ \omega_i(P_N) & \cdots & \omega_N(P_N) \end{pmatrix} \begin{pmatrix} A_1 \\ \vdots \\ A_N \end{pmatrix} = \begin{pmatrix} Q_1 \\ \vdots \\ Q_N \end{pmatrix}$$

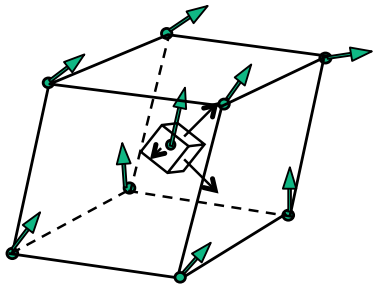


- Orientierung:
  - Effekt auf *Stereo*- und *Hand-Rendering*
  - Beobachtung: g: $\mathbb{R}^3 \rightarrow \mathbb{R}^6$ genügt
  - Theoretische Korrektur:
    - Sensor in definierter "Null-Lage" an jeden Punkt P im Raum halten
    - Orientierung M_P^0 messen
    - Zur Laufzeit: $\mathrm{M}_P^{\text{correct}} = (M_P^0)^{-1} \cdot M_P^{\text{measured}}$



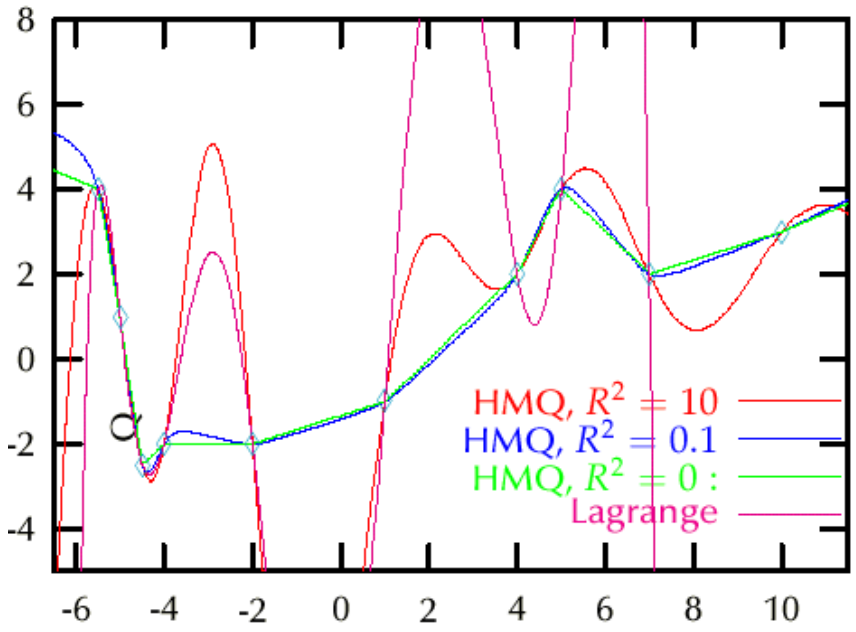
  - Praktisch:
    - Repräsentiere Ori durch 2 Zeilenvektoren

    1. g interpoliert/approximiert M_P^0 "dazwischen" wie bei Translation
    2. g(P) = M anschließend orthonormal machen
    3. Dieses M_P^0 für Korrektur der gemessenen Ori verwenden wie oben

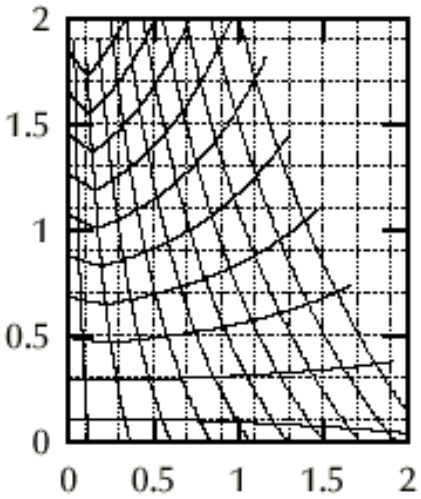


Das optimale R^2

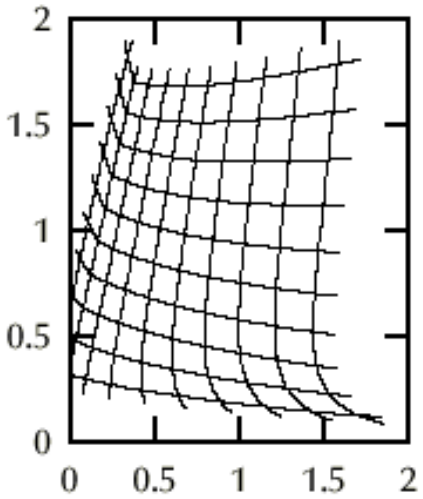
- Nur empirisch
- Meine Erfahrung:
  unkritisch im 3D mit genügend Punkten
  ($R^2 = 0.1 .. 10$)



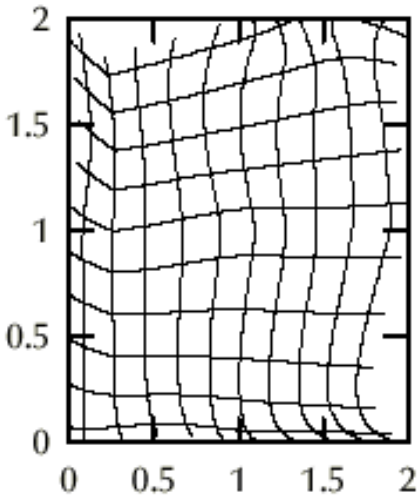
Vergleich HMQ und Polynom-Interpolation



Synth. Feld

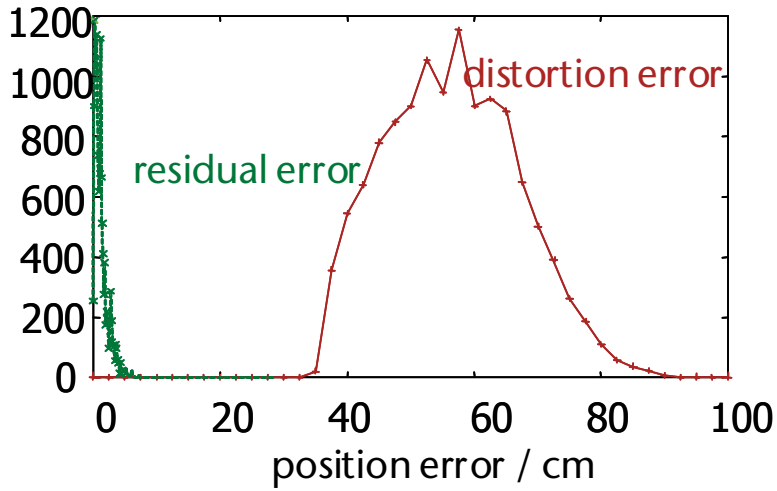


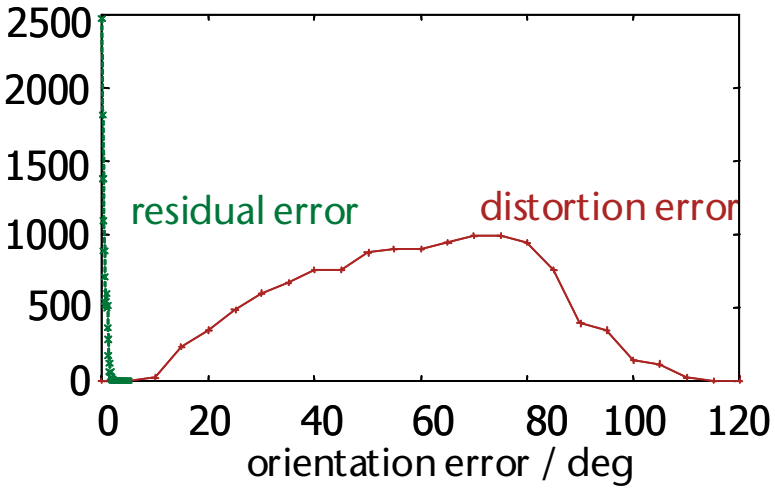
Lagrange



HMQ

- Vorteile:
  - Kein zusätzlicher Lag (200 Punkte < 1 Millisek.)
  - Beliebige Punktwolken
  - Man kann leicht Punkte zur Laufzeit hinzufügen
  - Stetigkeit, Stabilität, Güte
  - Interpolierend





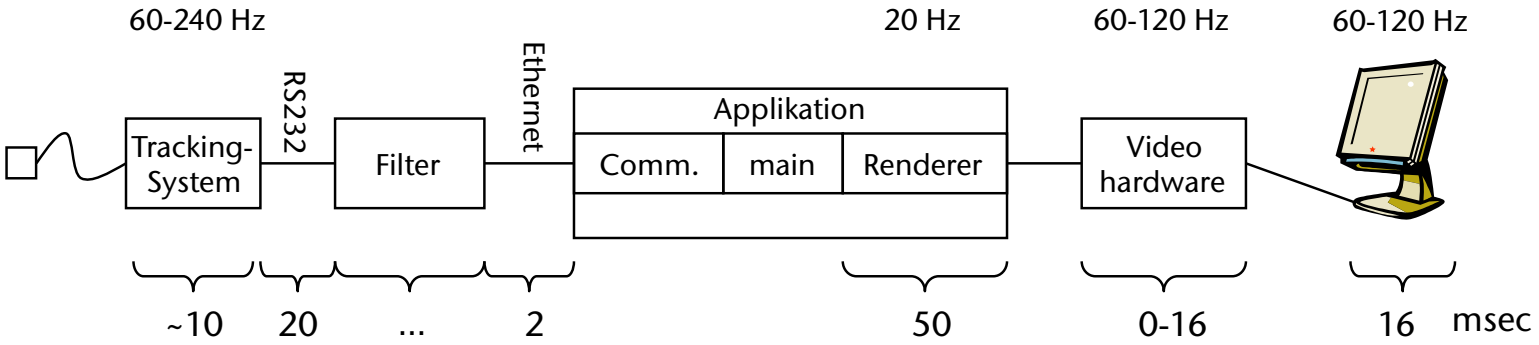
Latenz (*latency, lag*)

- Def.: Zeitdauer von der User-Aktion bis zur Änderung des Displays
- *Human factors*:

Latenz/msec	**Effekt auf User**
5	beobachtbar
30	*User-Performance* sinkt (evtl. "*simulator sickness*")
500	Immersion verschwindet

  - NB: Kopf kann mit 1000 Grad/Sek. rotieren

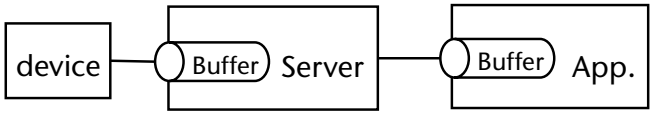
- Latenz-Pipeline:



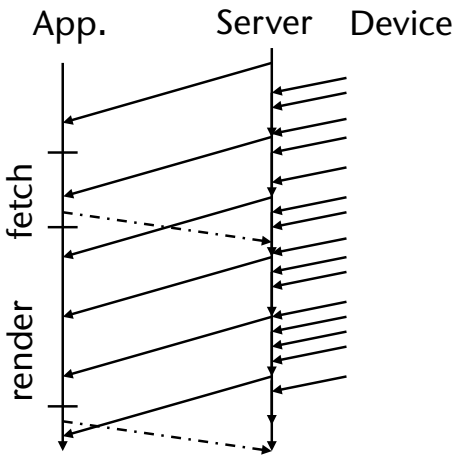
- Arten von Lag:
  - Geräte
  - Transport
  - Software
  - Synchronisation

Was kann man generell tun?

- Gerät-Server-App-Kommunikation:
  - "*continuous mode*"
  - "*keep alive*" message
  - Evtl. 2 Prozesse im Server



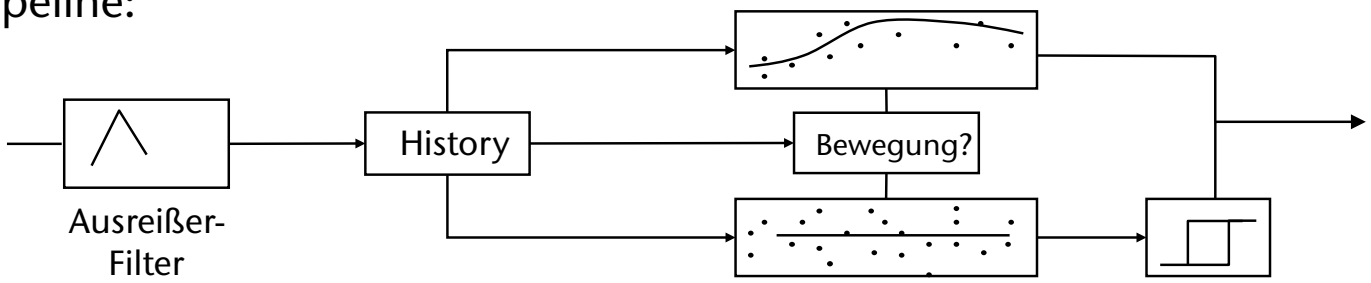
- "*Time-critical computing*":
  - Jeder Teil der App bekommt bestimmte Zeit zugeteilt
  - Abbrechen, wenn Zeit verbraucht



- Prädiktion der Geräte-Daten

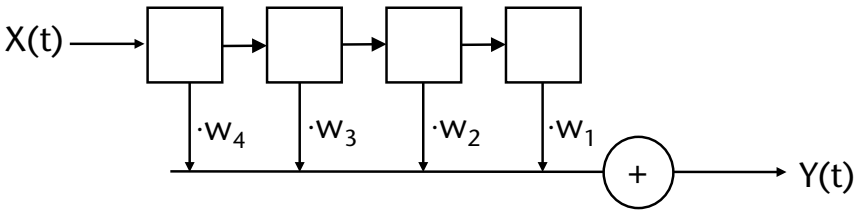
Filterung / Prädiktion

- Latenz bei Kopf- und *Hand-Tracking* besonders stark sichtbar
- Rauschen in *Tracking-Daten*
- Idee: prädiktive Filter:
  Liefere am Ausgang des Filters geglättete Daten in der "Zukunft"
- Erwünschte Eigenschaften:
  1. Rauschen löschen
  2. Stationäres Signal erkennen
  3. Ausreißer löschen
  4. Das Signal selbst sollte nicht gedämpft / verzögert werden
- Pipeline:

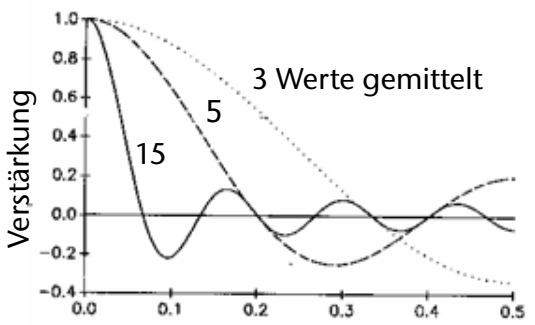


- FIR-Filter (*finite impulse response*):

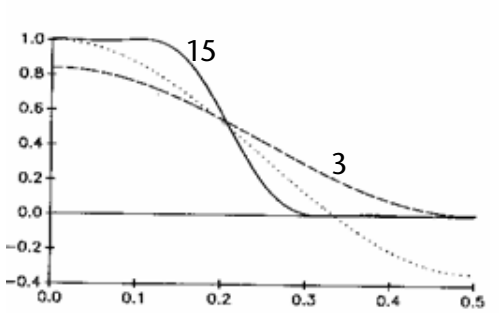
$$y_t = \sum_{i=-k}^{k} w_i x_{t+i}$$



- Charakteristik:
  - Zeigt, wie Filter Signal dämpft, abhängig von Abtastfrequenz
  - Mittelwert-Filter:

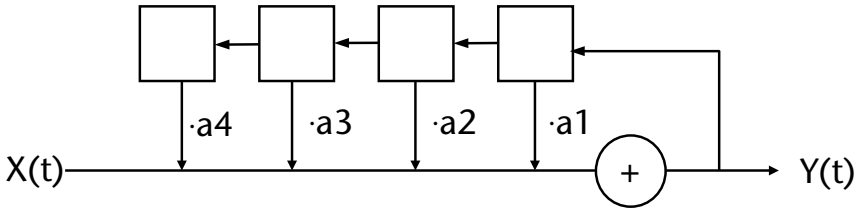


  - Verschiedene Gewichte (z.B. ¼, ½, ¼):



- IIR (*infinite impulse response*):

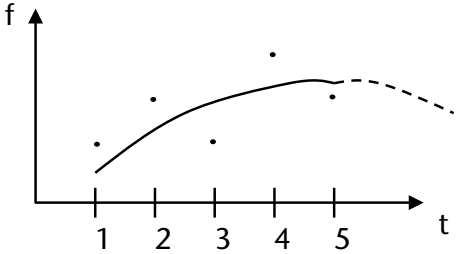
$$y_t = x_t + \sum_{i=1}^{k} a_i y_{t-i}$$



- Autoregressions-Modell:

$$y_t = \sum_{i=1}^{k} a_i y_{t-i} + \epsilon_t$$

- Polynomiale Filterung/Prädiktion:
  - Ansatz: $f(i) = a_0 + a_1 i + \ldots + a_n i^n$
  - Löse $A^T A\mathbf{a} = A^T\mathbf{f}$



$$A_{ij} = i^j, \quad \mathbf{a} = (a_0, \ldots, a_n), \mathbf{f} = (f(0), \ldots, f(n))$$

  - Werte f danach in der "Zukunft" aus

- Kalman-Filter:
  - Gegeben: linearer Prozeß mit diskreten Zeit-Schritten

$$x_{k+1} = A_k x_k + B u_k + w_k$$

  - Messprozeß

$$z_k = H_k x_k + v_k$$

  - Man kennt nur z_k und Schätzung für x_k, will aber x_{k+1} vorhersagen
    1. Schätze Vorhersage $\hat{x}_{k+1} = \hat{A}_k \hat{x}_k + \hat{B} u_k$
    2. Messe z_{k+1}
    3. Korrigiere $\hat{x}_{k+1}$
  - Optimal für lineare Prozesse
  - Parameter einstellen ist Magie