03-05-H
-709.53

Echtzeitbildverarbeitung (11)

Prof. Dr. Udo Frese

Auffrischung Stochastik
Quadratische Ausgleichsrechnung
Downhill Simplex Algorithmus
Parametrisierung von Drehungen

Was bisher geschah

- **Geometrische Rekonstruktion**
  - Kalibrierte Kamera ist ein Winkelmessgerät, eine Länge benötigt
  - Meist ist es so, wie die Zahl der Freiheitsgrade suggeriert.
  - Fluchtpunkt von parallelen Geraden hängt nur von Richtung ab.
  - Zwei Winkel zu Landmarken beschränkt die Position auf einen Kreis.
- **Kameragleichung in 3D (Parameter DOF)**
  - Transformation in Kamerakoordinaten $C2W^{-1}$ (6DOF)
  - Perspektive p (0 DOF)
  - Verzerrung d_κ (üblich: 0/1DOF)
  - Skalierung/Offset (1/3DOF)

$$\begin{pmatrix} x_I \\ y_I \end{pmatrix} = \begin{pmatrix} x_{center} \\ y_{center} \end{pmatrix} + f_{eff} \cdot d_\kappa\left(p\left(C2W^{-1} \cdot p_W\right)\right)$$

- **RANSAC**
  - finde passende Zuordnungen aus einer Menge von Zuordnungen
  - generiere Hypothese aus zufällig gezogenen Zuordnungen
  - zähle, wie viele Zuordnungen dazu passen

Auffrischung Stochastik

Auffrischung Stochastik

- **Zufallsexperiment: Ein Vorgang der einen nicht vorhersagbaren beobachtbaren Ausgang hat. Hier pragmatisch aufgefasst. Z.B. das Werfen zweier Würfels.**
- **Zufallsvariablen X, Y sind Größen, die von dem Ausgang eines Zufallsexperimentes abhängen. Z.B. Augensumme.**
- **p(X=x) ist die Wahrscheinlichkeit(-sdichte) dafür, dass die Zufallsvariable X, den Wert x hat. Es ist also eine Funktion von x.**
- **Wichtige Unterscheidung: X ist eine Zufallsvariable, also eine Größe, die vom Zufallsexperiment abhängt. x ist einfach nur eine Zahl. So wie p(X=0), p(X=7)**

Auffrischung Stochastik

- p(X=x, Y=y) ist die Wahrscheinlichkeit(-sdichte), dass X den Wert x hat und Y den Wert y hat.
- p(X=x|Y=y) = p(X=x, Y=y) / p(Y=y) ist die Wahrscheinlichkeit(-sdichte), dass X den Wert x hat, wenn Y schon den Wert y hat.
- Zwei Zufallsvariablen sind stochastisch unabhängig, wenn p(X=x, Y=y) = p(X=x) p(Y=y)
- Kurzschreibweise p(x) statt p(X=x), etc.
- Schreibweise: Bei kontinuierlichen Zufallsexperimenten p(…) für Wahrscheinlichkeitsdichten, bei diskreten P() für Wahrscheinlichkeiten.

Auffrischung Stochastik

- **Zufallsexperiment: Werfen zweier Würfel (einer weiß, einer rot)**
- **Zufallsvariable: X Augen des roten Würfel, Y Augen des weißen Würfel, Z=X+Y: Augensumme**

<u>Frage an das Audi-torium</u>: Welche Wahrscheinlichkeiten haben die verschiedenen Ereignisse?

P(X=x, Y=y)=

	P(X=_)	P(Y=_)	P(Z=_)	P(Z=_ |X=3)	P(Z=_ |Y=4)	P(Z=_ |X=4,Y=3)
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						

Auffrischung Stochastik

- **Zufallsexperiment: Werfen zweier Würfel (einer weiß, einer rot)**
- **Zufallsvariable: X Augen des roten Würfel, Y Augen des weißen Würfel, Z=X+Y: Augensumme**

	P(X=_)	P(Y=_)	P(Z=_)	P(Z=_ |X=3)	P(Z=_ |Y=4)	P(Z=_ |X=4,y=3)
1	1/6	1/6	0	0	0	0
2	1/6	1/6	1/36	0	0	0
3	1/6	1/6	2/36	0	0	0
4	1/6	1/6	3/36	1/6	0	0
5	1/6	1/6	4/36	1/6	1/6	0
6	1/6	1/6	5/36	1/6	1/6	0
7	0	0	6/36	1/6	1/6	1
8	0	0	5/36	1/6	1/6	0
9	0	0	4/36	1/6	1/6	0
10	0	0	3/36	0	1/6	0
11	0	0	2/36	0	0	0
12	0	0	1/36	0	0	0

P(X=x, Y=y)=1/36, für alle x,y $\in$[1..6] sonst 0

Quadratische Ausgleichsrechnung

Quadratische Ausgleichsrechnung

<u>Aufgabe</u>

- **bestimme Parameter eines Messmodells so, dass Vorhersagen möglichst gut mit Messungen übereinstimmen.**
- **Ziel: Mehr Messungen sollten die Genauigkeit verbessern (Ausgleichsrechnung).**
- **Anwendung Kalibrierung:**
  - bestimme Kamerapose *C2W*, Verzerrung κ und Brennweite f_{eff} so, dass Projektionen bekannter Punkte möglichst gut mit Bildpunkten übereinstimmen.
- **Anwendung Kamera / Objektlage:**
  - wie oben, aber κ und f sind fest.
- **Anwendung 3D Rekonstruktion :**
  - bestimme Kameraposen *C2W* und Punkte, so dass deren Projektion möglichst gut mit Bildpunkten übereinstimmen.
  - heißt auch Simultaneous Localization and Mapping (SLAM)
  - heißt auch Structure from Motion (SfM)

Quadratische Ausgleichsrechnung

Praxis

- die Theorie ist hilfreich, aber nicht ganz leicht zu verstehen.
- deshalb zuerst: Was rechnen wir am Ende praktisch?

$$\hat{x} = \arg\min_x \sum_i \frac{\left(z_i - f_i(x)\right)^2}{\sigma_i^{\;2}}$$

- wahre Parameter X, gesucht
- Messungen z_i
- Messfunktion(-en) f_i, d.h. „wenn der Zustand x wäre, müssten wir eigentlich $Z_i=f_i(x)$ messen“
- σ_i Unsicherheit der Messung Z_i
- Frage an das Autitorium: Könnt Ihr die Formel in Worte fassen?

Quadratische Ausgleichsrechnung

Praxis

$$\hat{x} = \arg\min_x \sum_i \frac{(z_i - f_i(x))^2}{\sigma_i^2}$$

- **wahre Parameter X, gesucht**
- **Messungen Z_i**
- **Messfunktion(-en) f_i, d.h. „wenn der Zustand x wäre, müssten wir eigentlich $Z_i=f_i(x)$ messen“**
- **σ_i Unsicherheit der Messung Z_i**
- **Formel in Worten**
  - Klammer ist Differenz zwischen, was wir im Zustand x hätten messen sollen und was wir gemessen haben.
  - Division durch σ_i^2 macht die Differenz relativ zur Ungenauigkeit des Sensors.
  - dadurch Maß für Plausibilität des Zustandes x unter der Messung Z_i.
  - Summe ist Gesamtplausibilität des Zustandes x unter allen Messungen.
  - wir suchen den plausibelsten Zustand.

Quadratische Ausgleichsrechnung

Ansatz

- **Modelliere Messvorgang als Zufallsexperiment, bei dem auf den „wahren“ Messwert der sich aus den „wahren“ Parametern ergibt eine zufällige Störung addiert wird. Frage dann nach den *wahrscheinlichsten* Parametern gegeben die Messungen.**

$$Z_i = f_i(X) + N_i$$

- **wahre Parameter X, Messungen Z_i, Messfunktion(-en) f_i, Störung N_i**

Quadratische Ausgleichsrechnung

Herleitung

$$Z_i = f_i(X) + N_i$$

- **bekannt: $Z_i = z_i$, f_i Unbekannt: X, N_i**
- **verschiedene f_i für verschiedene Messwerte möglich.**
  - verschiedene Teile einer Messung, z.B. X- und Y- Koordinate eines Bildpunktes
  - verschiedene Sensoren, z.B. Winkelpeilung und Radarentfernung
  - abhängig von Parameter $f_i(X) = f(p_i, X)$, z.B. Messung von Bildpunkten verschiedener Raumpunkte p_i.
- **Annahme: Störungen (Messungen) stochastisch unabhängig.**

Quadratische Ausgleichsrechnung

Herleitung

- **Gesucht: arg max$_x$ p(X=x|Z=z),**
- **also der Parametersatz, mit der höchsten Wahrscheinlichkeit unter der Bedingung dass das gemessen wurde, was gemessen wurde.**
- **Berechnung über Bayes Theorem:**

$$p(X=x|Z=z)\,p(Z=z) = p(X=x, Z=z) = p(Z=z|X=x)\,p(X=x)$$

Wahrscheinlichkeit, dass X den Wert x hat, gegeben, dass Z den Wert z hat

Wahrscheinlichkeit, dass Z den Wert z hat

Wahrscheinlichkeit, dass X den Wert x hat und dass Z den Wert z hat

Quadratische Ausgleichsrechnung

A-posteriori Wahrscheinlichkeit

$$\arg\max_x p(X = x | Z = z) = \arg\max_x \frac{p(Z = z | X = x) p(X = x)}{p(Z = z)}$$

$$= \arg\max_x p(Z = z | X = x) p(X = x) = \arg\max_x p(Z = z | X = x)$$

Keine a-priori Info

P(Z=z) ist gleich für alle *x*

Wie wahrscheinlich ist es, *Z*=*z* zu messen, wenn *X*=*x* wäre? (likelihood von x)

Wie wahrscheinlich ist *x* grundsätzlich?

Quadratische Ausgleichsrechnung

$$\arg\max_x p(Z = z \mid X = x) = \arg\max_x p(f(x) + N = z \mid X = x)$$

$$= \arg\max_x p(N = z - f(x) \mid X = x) = \arg\max_x p(N = z - f(x))$$

- **für ein hypothetisches *x* wissen wir, was bei den Messung hätte herauskommen sollen, nämlich *f*(*x*).**
- **wir wissen, was herausgekommen ist, nämlich *Z*=z.**
- **wir kennen den Meßfehler: *z*- *f*(x).**
- **wie wahrscheinlich ist solch ein Meßfehler?**
- **Messungen stochastisch unabhängig:**

$$= \arg\max_x \prod_i p(N_i = z_i - f_i(x))$$

Quadratische Ausgleichsrechnung

- **Modell des Meßrauschens:**
- **n_i hat Gaußverteilung mit Mittelwert μ_i=0 und Standardabweichung σ_i.**

$$p(N_i = n_i) = \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{(n_i - \mu_i)^2}{2\sigma_i^2}}$$

$$= \arg\max_x \prod_i e^{-\frac{(z_i - f_i(x))^2}{2\sigma_i^2}} = \arg\min_x \sum_i \frac{(z_i - f_i(x))^2}{\sigma_i^2}$$

Quadratische Ausgleichsrechnung

Anwendung Kamerakalibrierung aus Geraden

- **Parametervektor x = (t_x, t_y, t_z, α, β, γ, κ, f_{eff}) (kein κ in Übungen)**
- **Messfunktion Kameragleichung auf Geraden (2 Punkte abbilden)**
- **Meßwert: Im Bild erkannte Geraden**
- **Funktion *diff* Unterschied zweier Geraden (Winkelperiodizität)**
- **statt $(z_i-f(x))^2$ nutze $diff^2(z_i, f(x))$ für Unterschied zweier Geraden**
- **T_0 in jeden Schritt festgehalten und durch $T_0 \cdot T(t_x, t_y, t_z, \alpha, \beta, \gamma)$ ersetzen (siehe Parametrisierung von Drehungen)**

Downhill Simplex Algorithmus

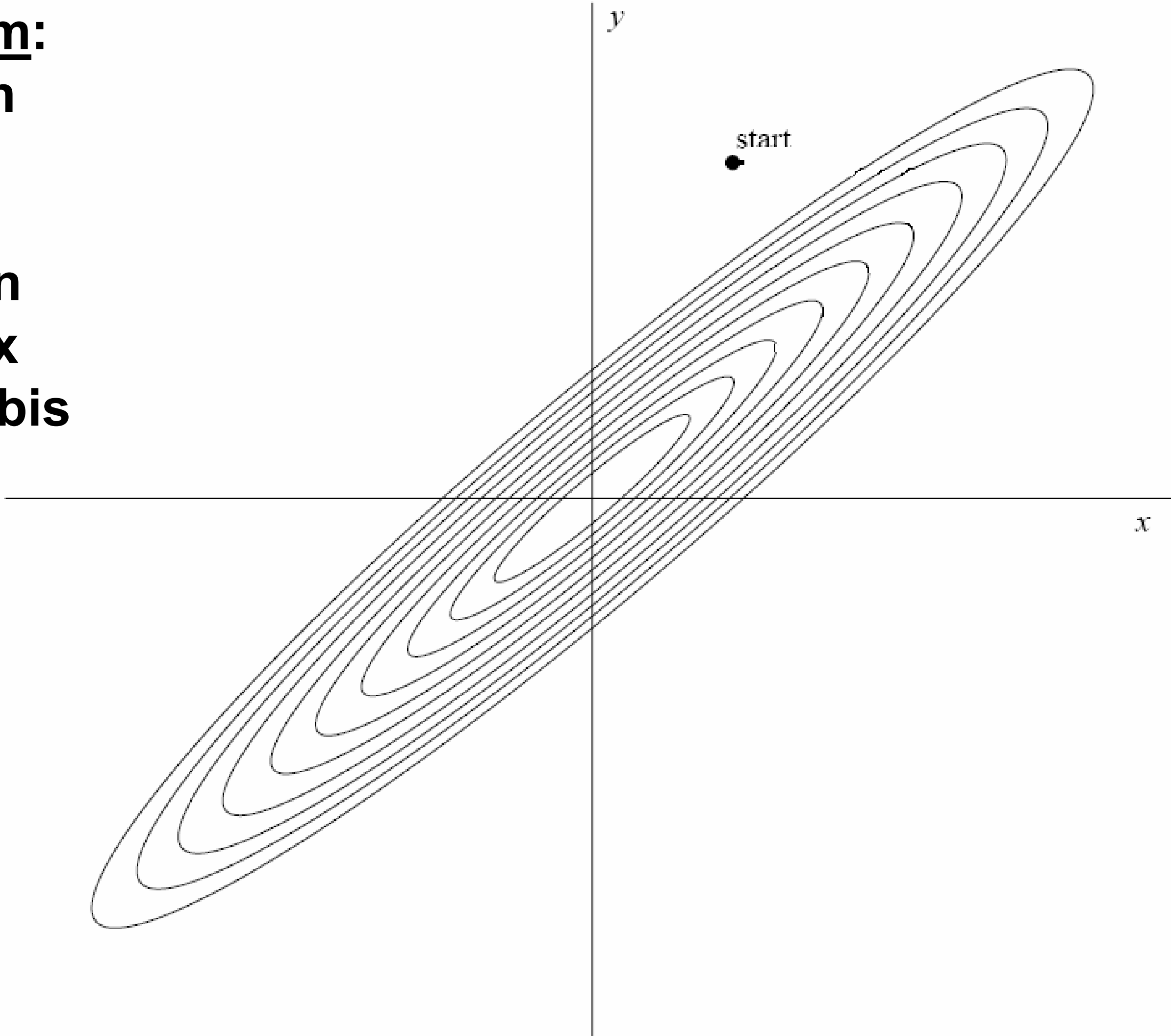
Downhill Simplex Algorithmus

Aufgabe

- **Finde Minimum einer Funktion**
- **für allgemeine Funktionen**
  - vom Startwert, schrittweise abwärts, bis es nicht mehr weitergeht.
  - Problem: vernünftiger Startwert nötig.
  - Problem: lokale Minima.
- **Vorteil: einfach, ohne Ableitungen.**
- **Quelle: Press, Teukolsky, Vetterling, Flannery, Numerical Recipes in C, Chapter, Chapter 10.4 (sehr gut für praktische Numerik)**
- **Nicht verwechseln mit Simplex aus Operations Research.**

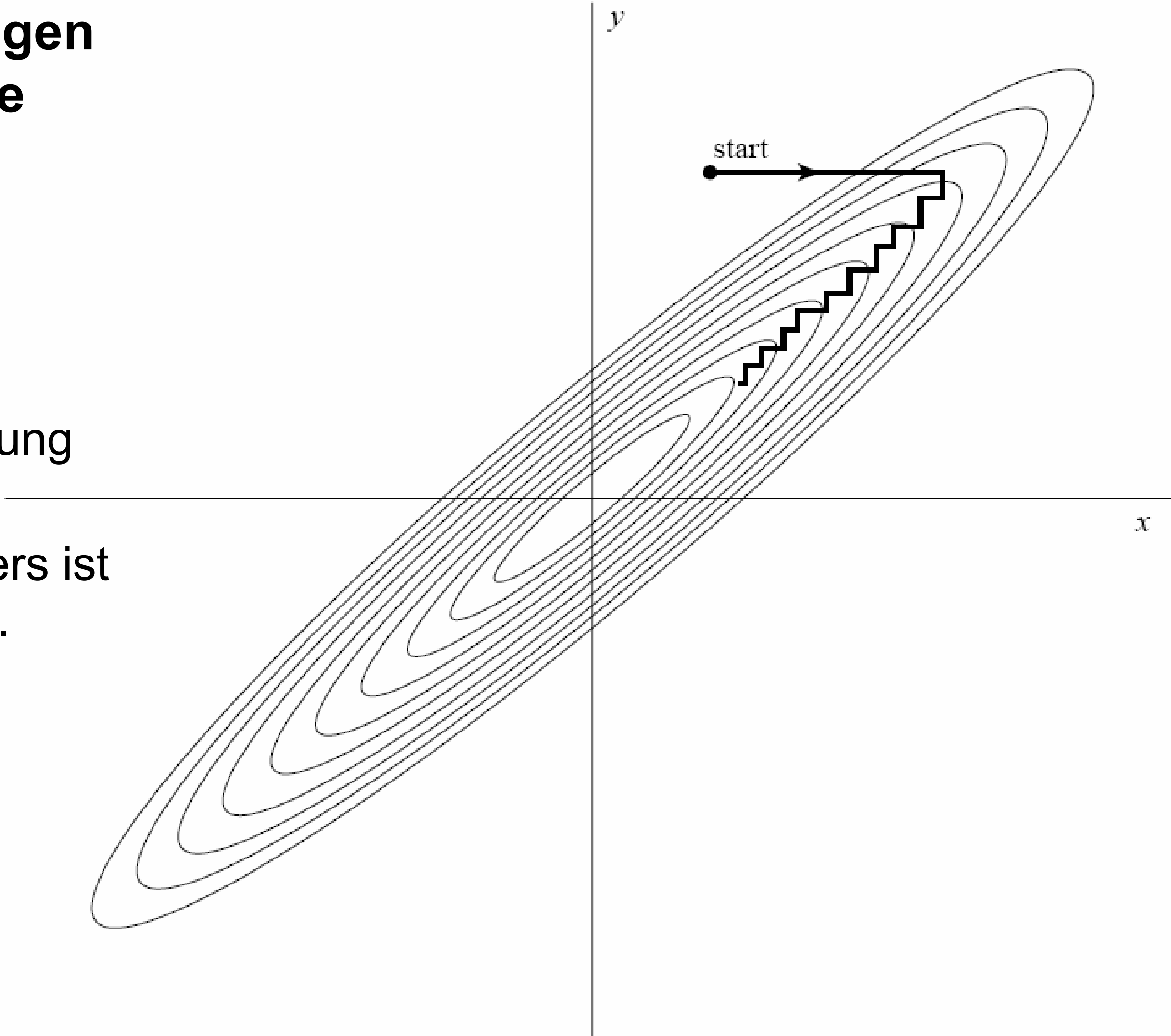
Downhill Simplex Algorithmus

- **<u>Frage an das Auditorium</u>:**
  Rechts sieht man Linien gleicher Werte für $x^2+y^2-1.5xy$.
  Was passiert, wenn man immer abwechselnd in x und y Richtung jeweils bis zum Minimum in dieser Richtung absteigt?



Downhill Simplex Algorithmus

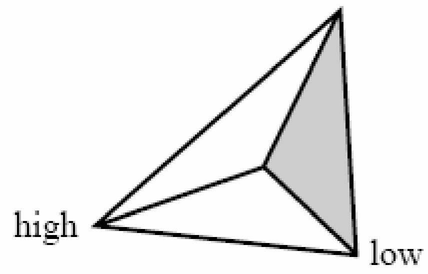
- **treppenförmiges Absteigen durch das langes, flache Tal.**
  - sehr ineffizient.
  - häufiges Phänomen
- **Gegenmassnahme:**
  - Minimierer soll sich Richtung des Tals anpassen.
  - $\Rightarrow$ Zustand des Minimierers ist mehr als nur einen Punkt.



Downhill Simplex Algorithmus

Simplex als Minimiererzustand

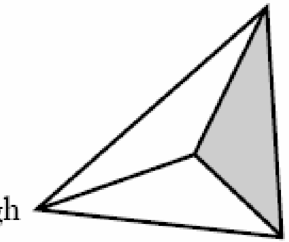
- ***n*** **Parameter (7 bzw. 8 bei Kalibrierung).**
- **Zustand des Minimierers ist ein Simplex im Parameterraum R^n:**
  - konvexe Hülle aus n+1 Vektoren.
  - kleinste Anzahl Vektoren, die alle Dimensionen des Raums aufspannen.
  - 1D: Interval, 2D: Dreieck, 3D: Tetraeder, …



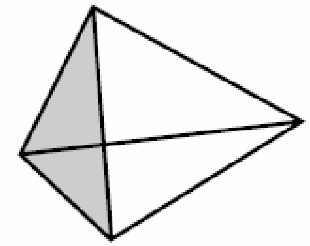
- **Simplex passt sich „wie eine Amöbe“ dem Tal an.**
- **vier Operationen: reflection, expansion , contraction, multiple contraction**
- **Versuch, den Parametervektor mit höchstem Funktionswert (p_n) zu beseitigen, d.h. kleiner als den zweithöchsten zu machen.**

Ein Schritt im Downhill Simplex

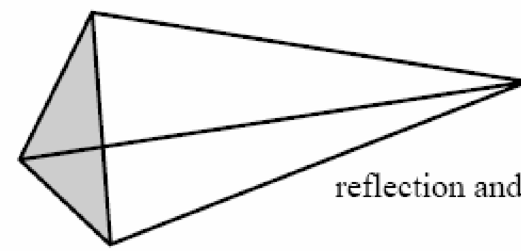
$$p^{neu} = \frac{1-\lambda}{n}\sum_{i=0}^{n-1} p_i + \lambda p_n$$



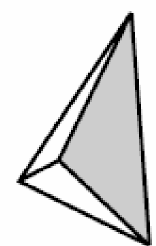
- **reflection:**
  - schlechtesten Parametervektor (p_n) über Schwerpunkt der anderen (λ=-1) spiegeln.
  - wenn besser, ersetzen.



- **expansion: wenn besser als bester**
  - λ=2 auf *neuem* p_n probieren
  - falls wieder besser ersetzen
  - nicht λ=-2, weil schon gespiegelt

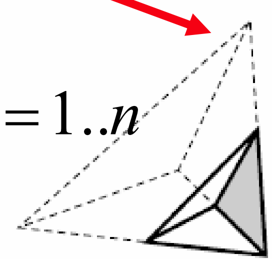


- **contraction: wenn schlechter als zweitschlechtester**
  - λ=1/2 probieren.



- **multiple contraction: wenn schlechter als alter schlechtester**
  - jeden Parametervektor durch die Mitte zum Besten ersetzen.

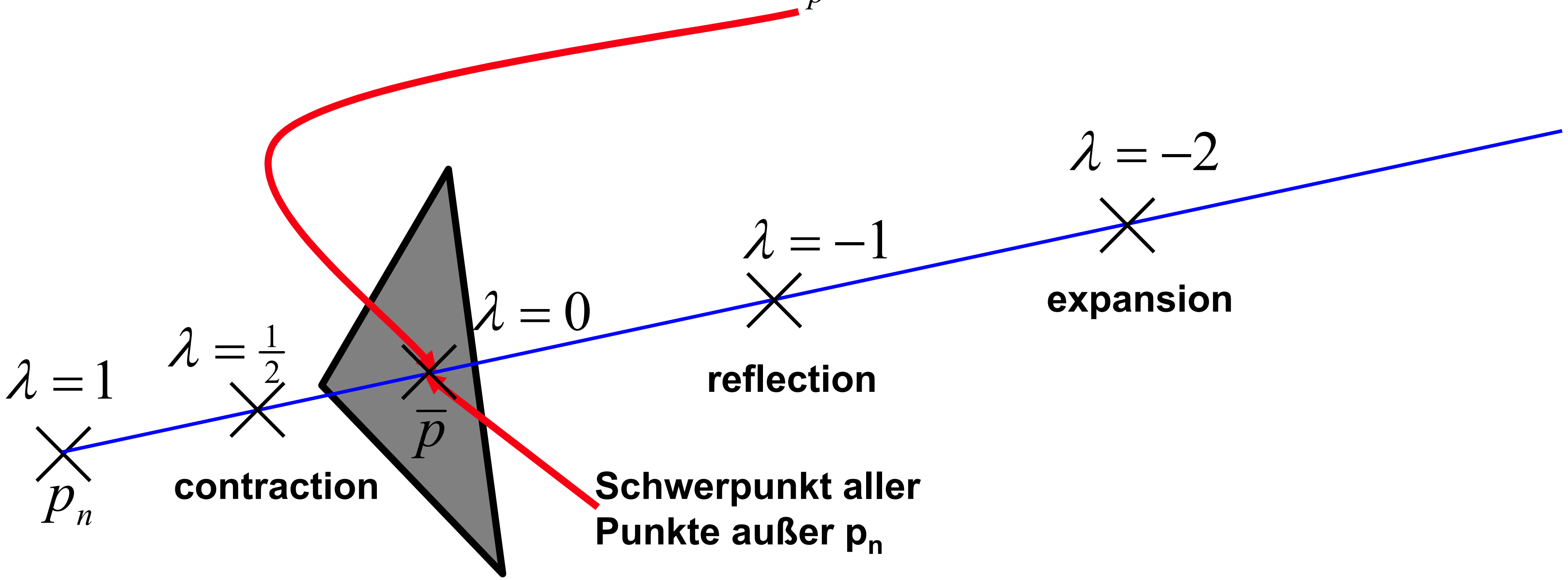
$$p_i^{neu} = \frac{p_i + p_0}{2} \quad \forall i = 1..n$$



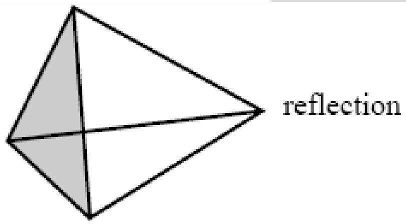
Quelle: Numerical Recipes, §10.4

Downhill Simplex Algorithmus

$$p^{neu} = \frac{1-\lambda}{n}\sum_{i=0}^{n-1} p_i + \lambda p_n = (1-\lambda)\underbrace{\left(\frac{1}{n}\sum_{i=0}^{n-1} p_i\right)}_{\bar{p}} + \lambda p_n = (1-\lambda)\bar{p} + \lambda p_n$$

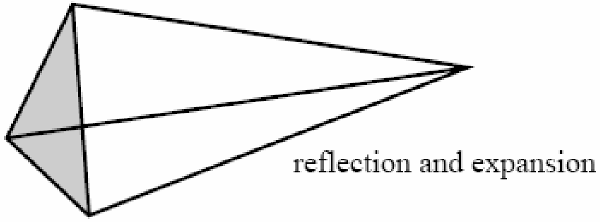


Downhill Simplex Algorithmus

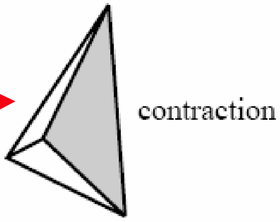


Analogie:

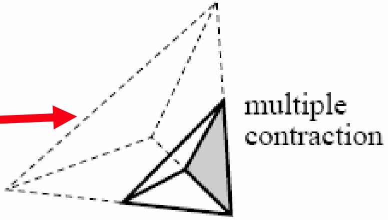
- **Weiterfahren**



- **Gas geben**

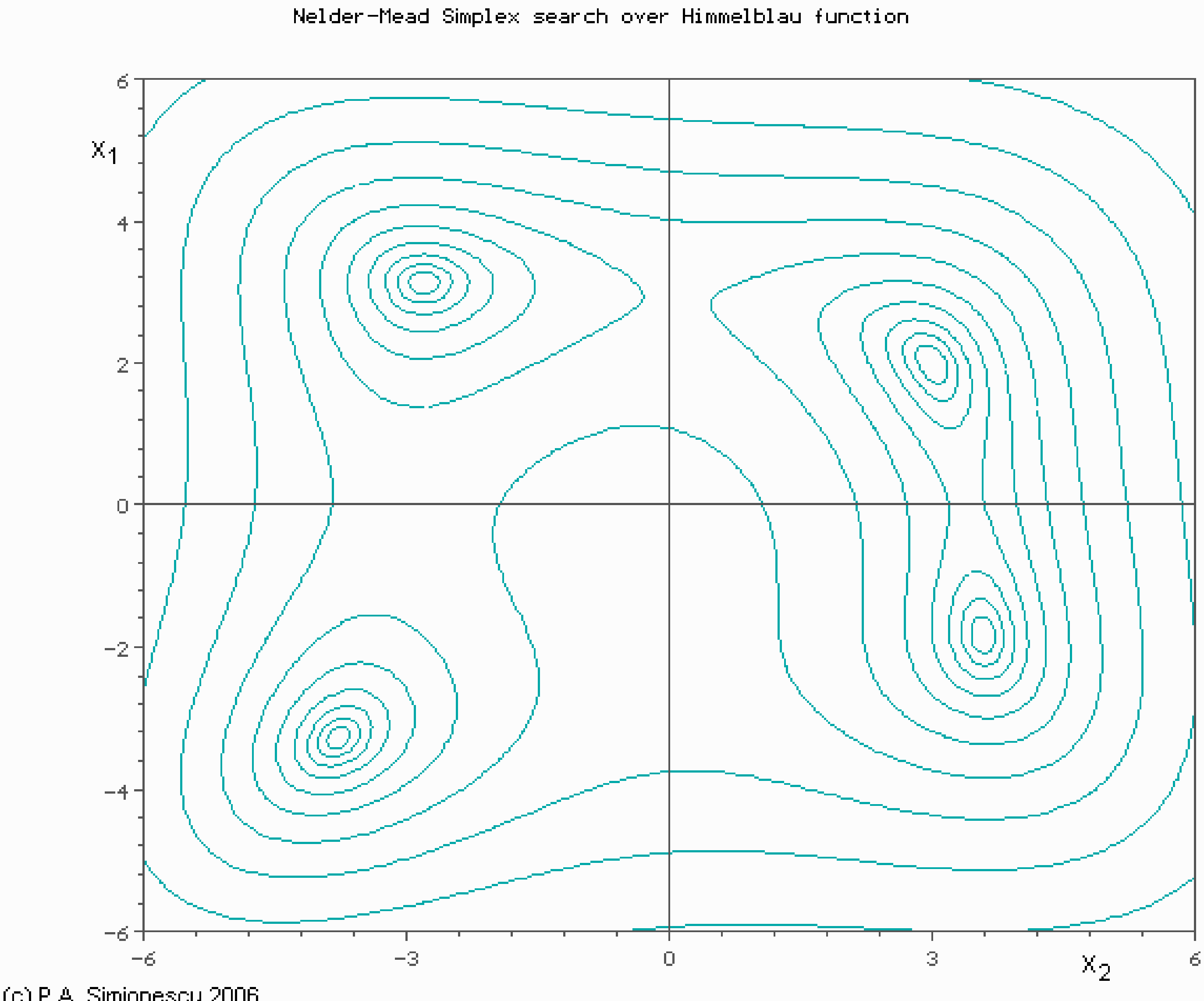


- **Bremsen**



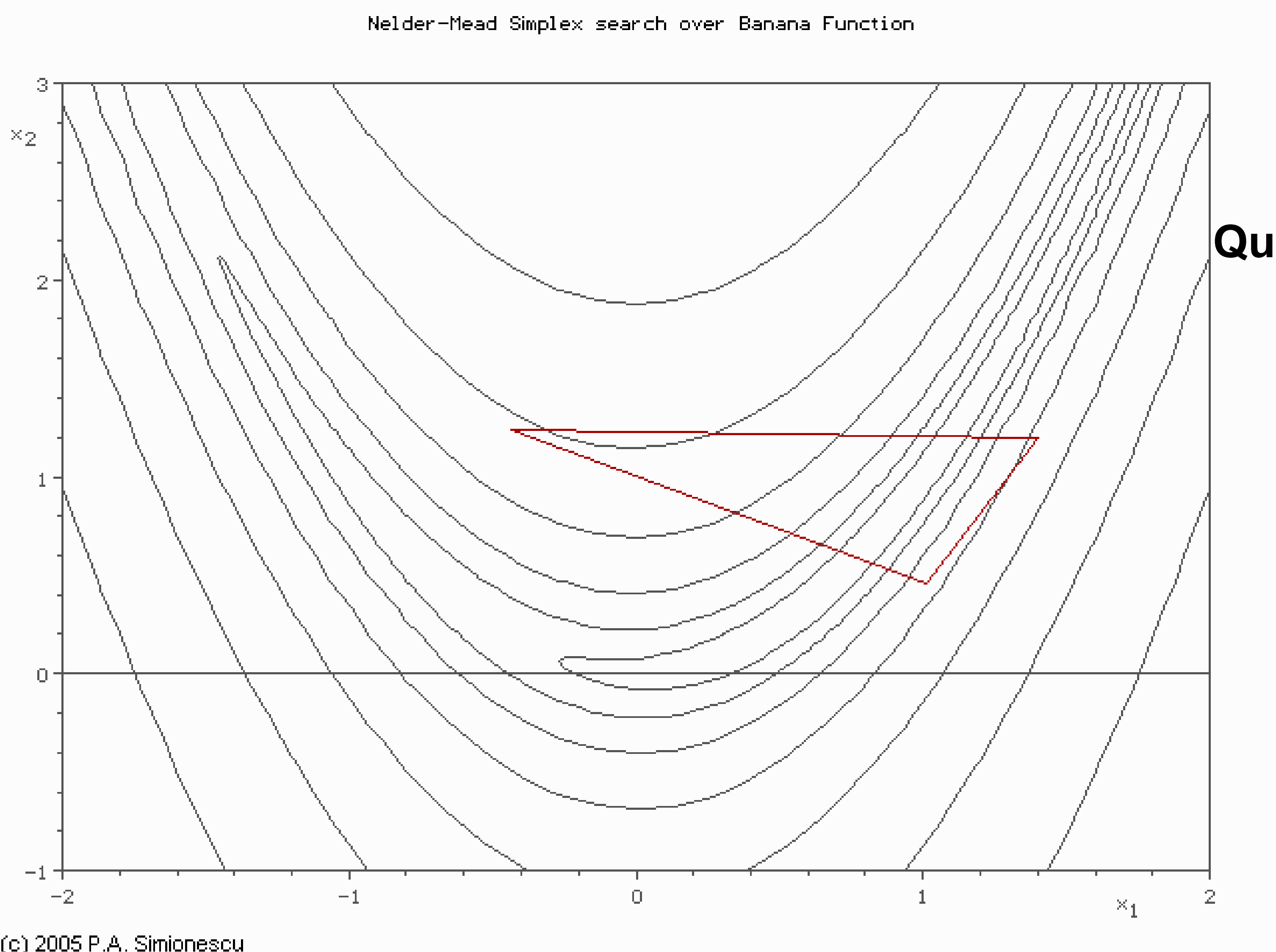
- **Vollbremsung**

Downhill Simplex Algorithmus



Quelle: wikipedia

Downhill Simplex Algorithmus



Quelle: wikipedia

```
downhillSimplexStep () { //p Liste von Parametervektoren, Fehler in .error
   renormalize rotation representation // speziell für Drehungen
   sort p by ascending .error value
   if (p[n].error-p[0].error<tolerance) {
      if (p[0].error<lastRestart) {
         lastRestart = p[0].error;
         restart();
      }
      else return true; // simplex converged
   }
   compute pNew with λ=-1 from p[n], evaluate pNew.error // reflection
   if (pNew.error<=p[n].error) p[n] = pNew;
   if (pNew.error<=p[0].error) { // gut!
      compute pNew with λ=2 from p[n], evaluate pNew.error // and expansion
      if (pNew.error<=p[n].error) p[n] = pNew; // besser!
   }
   else if (pNew.error>=p[n-1].error) { // mäßig
      compute pNew with λ=0.5 from p[n], evaluate pNew.error // contraction
      if (pNew.error<=p[n].error) p[n] = pNew;
      else { // besch…
         for (int i=1;i<=n;i++) // multiple contraction
            compute p[i] as (p[i]+p[0])/2, evaluate p[i].error
      }
   }
   return false; // weitere Iterationen nötig
}
```

Downhill Simplex Algorithmus

Initialisierung, restart und Renormalisierung

- **Man benötigt einen Startparametervektor p_{start}.**
  - Startwert von Hand wählen (oft mühsam).
  - RANSAC (gewählte Lösung mit meisten passenden Daten)
  - bei restart besten Parametervektor p[0] als p_{start} verwenden.
- **Startsimplex**
  - p_{start}
  - n mal p_{start} in je einer Komponente um ε erhöht.
  - ε lieber klein wählen
- **Renormalisierung**
  - wenn Orientierungen Teil der Parameter sind
  - su.

Parametrisierung von Drehungen

Parametrisierung von Drehungen

Minimalparametrisierung einer Transformation

- **Transformation hat 12 Zahlen, aber nur 6 Freiheitsgrade und 6 Zwangsbedingungen.**
- **Parametrisierung mit 6 Parametern durch Eulerwinkel**
  - Translation als Vektor t_x, t_y, t_z.
  - 3 aufeinanderfolgende Drehungen α, β, γ (Eulerwinkel)
- **Jede Transformation lässt sich durch diese 6 Parameter darstellen.**

$$T(t,\alpha,\beta,\gamma)=$$

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Parametrisierung von Drehungen

- **Frage an das Auditorium**: Was machen die einzelnen Matrizen?

$$T(t,\alpha,\beta,\gamma)=$$

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

? ? ? ?

Parametrisierung von Drehungen

- **Frage an das Auditorium: Was machen die einzelnen Matrizen?**

$$T(t,\alpha,\beta,\gamma) =$$

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Verschieben **Drehen um Z** **Drehen um Y** **Drehen um X**

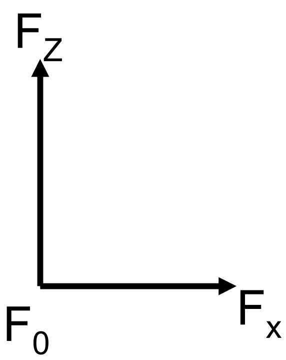
Parametrisierung von Drehungen

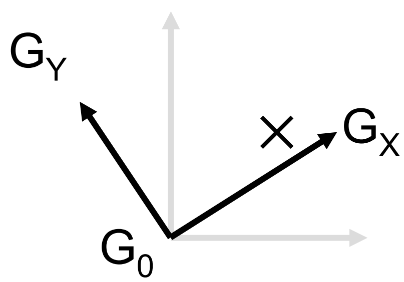
$$T(t,\alpha)=\begin{pmatrix}1 & & t_x\\ & 1 & t_y\\ & & 1\end{pmatrix}\cdot\begin{pmatrix}\cos\alpha & -\sin\alpha & 0\\ \sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{pmatrix}$$

F

G

- **Transformation des Punktes von G-Koordinaten nach F-Koordinaten von (rechts nach links)**
- **Bewegung des F Koordinatensystems nach G von (links nach rechts)**





$$t_x=1, t_y=0, \alpha=45°$$

Parametrisierung von Drehungen

- **<u>Frage an das Auditorium</u>: Wo liegen die Zwischensysteme (rote Pfeile) des Matrixproduktes für das Beispiel `Table2World`? Welche Parameter hätte die Transformation?**

$$T(t,\alpha,\beta,\gamma)=$$

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

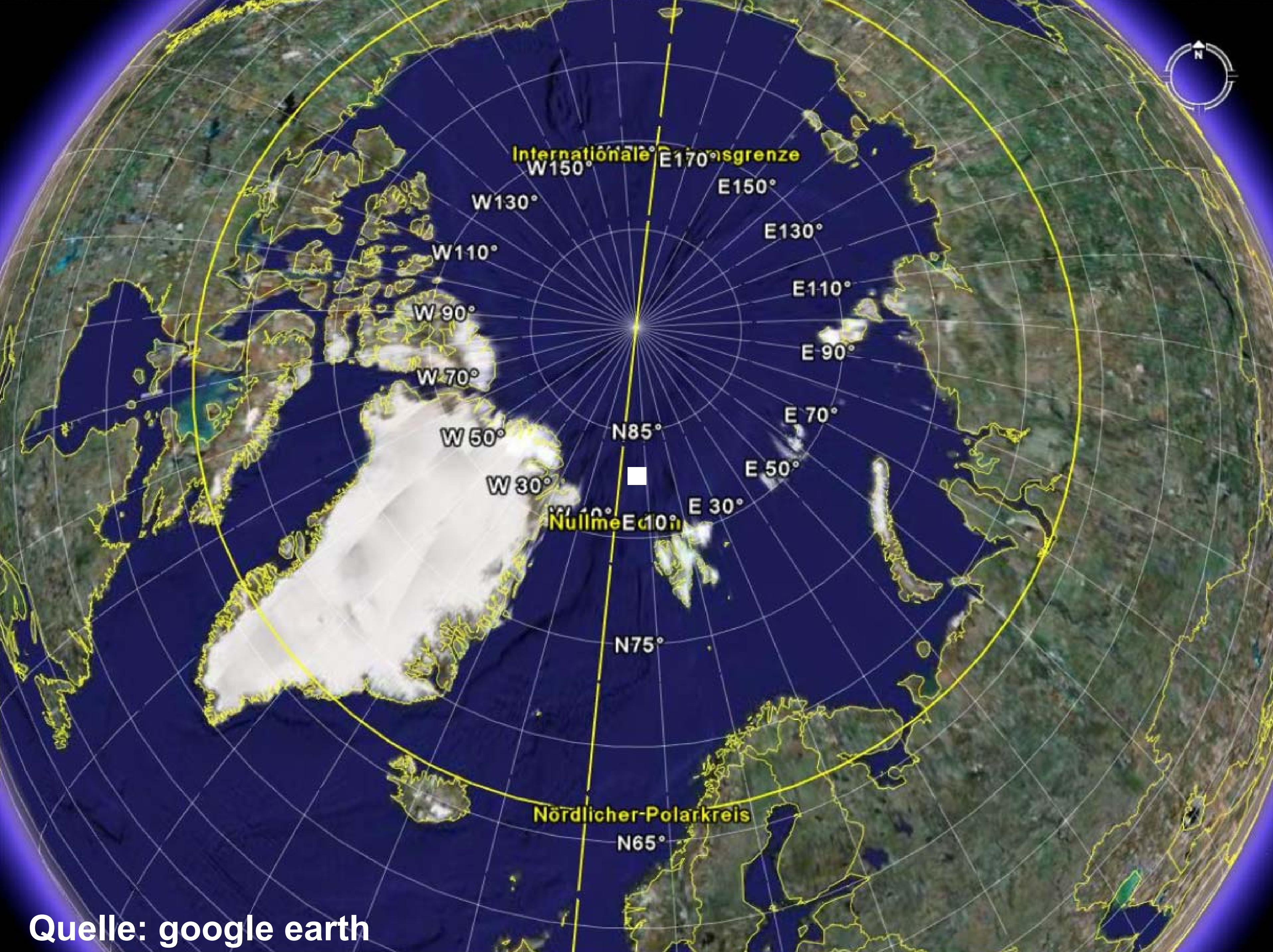
Welt **?** **?** **?** **Tisch**

t_x = ? **t_y = ?** **t_z = ?** **α = ?** **β = ?** **γ = ?**

Parametrisierung von Drehungen

Singularitäten

- **Eulerwinkel wenig intuitiv.**
- **jeder benutzt eine andere Definition.**
- **sogenannte Singularitäten**
  - es gibt Transformationen bei denen eine geringe Änderung der Transformation eine große Änderung der Parameter α, β, γ erfordert.
- **betrachte die X-Achse der Transformation und t=0**
  - wie 3D Vektor aus geographischer Länge (γ) und Breite (β), wenn Z die Erdachse ist und X, Y in der Äquatorebene auf Greenwich Länge liegt.
- **um den Pol befinden sich Punkte aller Längen.**
  - kleiner Schritt Richtung Greenwich $\Rightarrow$ Länge 0
  - kleinen Schritt Richtung Chicago $\Rightarrow$ Länge 90°W.



Quelle: google earth

Parametrisierung von Drehungen

Singularitäten

- **bei β=+/-90° (entspricht Pol)**
- **Singularitäten sind ein großes Problem für viele Algorithmen**
  - Gleichungslöser für nichtlineare Gleichungen
  - Schätzalgorithmen (Least Square)
  - Differentialgleichungslöser
- **„In der Umgebung einer theoretischen Singularität ist eine praktische Todeszone.“**
- **Informatiker in der Yucca-Palme: „Der Prototyp eines Kampfflug-zeugs soll sich einmal beim Überfliegen des Äquators auf den Rücken gedreht haben wegen einer Singularität in der Darstellung.“**

Parametrisierung von Drehungen

Vermeidung von Singularitäten im Downhill Simplex

- **parametrisiere relativ zu Referenztransformation**
  $T_i = T_0 \cdot T(\delta \boldsymbol{t_i}, \delta\alpha_i, \delta\beta_i, \delta\gamma_i)$
- **$\delta \boldsymbol{t_i}, \delta\alpha_i, \delta\beta_i, \delta\gamma_i$ in Parametervektor p_i**
- **T_0 in jeden Schritt und alle Parameter (p_i) fest**
- **Downhill Simplex arbeitet auf $\delta \boldsymbol{t}, \delta\alpha, \delta\beta, \delta\gamma$**
- **gelegentlich renormalisieren:**
  - schiebe beste Transformation von $\delta t, \delta\alpha, \delta\beta, \delta\gamma$ nach T_0
  - $T_0 = T_0 \cdot T(\delta t_0, \delta\alpha_0, \delta\beta_0, \delta\gamma_0)$
  - $(\delta t_i, \delta\alpha_i, \delta\beta_i, \delta\gamma_i) = (\delta t_i, \delta\alpha_i, \delta\beta_i, \delta\gamma_i) - (\delta t_0, \delta\alpha_0, \delta\beta_0, \delta\gamma_0)$
  - dadurch $(\delta t_i, \delta\alpha_i, \delta\beta_i, \delta\gamma_i)$ immer klein
  - weit weg von Singularitäten
  - untere Gleichung ist Näherung

Zusammenfassung

- **Quadratische Ausgleichsrechnung**
  - Berechnung der wahrscheinlichsten Parameter
    x = argmax_x P(X=x|Z=z) als = argmax_x P(Z=z | X=x).
  - „*Die wahrscheinlichsten Parameter, gegeben, dass ich gesehen habe, was ich gesehen habe sind die, bei denen es am wahrscheinlichsten ist, dass zu sehen, was ich gesehen habe.*“
  - Gauß´sches Messrauschen führt zu quadratischer Minimierung.

$$\hat{x} = \arg\min_x \sum_i \frac{\left(z_i - f_i(x)\right)^2}{\sigma_i^2}$$

- **Downhill Simplex Algorithmus**
  - Minimiert allgemeine Funktionen, einfach, ableitungsfrei aber langsam.
  - Zustand durch Simplex von n+1 Parametervektoren.
  - vier Operationen: reflection, expansion, contraction, multiple-contraction
- **Parametrisierung von Rotationen**
  - 6 Parameter, 3D Translation + 3 Drehungen hintereinander T(δt, $\delta\alpha$, $\delta\beta$, $\delta\gamma$)
  - Singularität bei $\delta\beta$=90°
  - besser: relativ zu Referenztransformation $T_0 \cdot T(\delta t, \delta\alpha, \delta\beta, \delta\gamma)$
  - gelegentlich von T(…) nach T_0 schieben