
# 3D-Bildabgleich

Seminar: Mustererkennung in Bildern und 3D-Daten

SoSe 2004

Christian Gleichner

# Gliederung:

- Einführung
- Grundtechniken
  - Abgleich mit Punktentsprechungen
    - Quaternion Methode, Singulärwertzerlegung, Duale Quaternionen, Orthogonale Matrizen
  - Abgleich ohne Punktentsprechungen
    - ICP, CM
- Ungenauigkeiten im Abgleich

# Einführung

- Abgleich von Objekten dreidimensionaler Bilder über ihre Datensätze von 3D-Koordinaten

- Matchen zweier 3D-Punktmengen $X$, $Y$ durch ermitteln sich entsprechender Punktpaare $(\boldsymbol{x_i}, \boldsymbol{y_i})$ und anschließender Transformation

$$\boldsymbol{x}_i = \boldsymbol{R}\boldsymbol{y}_i + \boldsymbol{t}$$

# Abgleich mit Punktentsprechungen

- Entsprechung aller Punkte mit selben Index zweier 3D-Punktmengen (Registration with Point Correspondences)

$$X = \{x_1, x_2, \mathrm{K}\ , x_N\}$$
$$Y = \{y_1, y_2, \mathrm{K}\ , y_N\} \qquad x_i, y_i \in \mathbf{R}^3$$

- 3D-Objekte deckungsgleich, wenn jeder Punkt $\boldsymbol{x}_i$ über eine Transformation des entsprechenden Punktes $\boldsymbol{y}_i$ ermittelt werden kann
  - d.h. jeweils Punkt $\boldsymbol{x}_i$ auf Punkt $\boldsymbol{y}_i$ matchen

- Algorithmen, die den Abgleich zweier übereinstimmender Punktmengen möglich machen, werden benötigt
- Ermittlung der Transformation zwischen den Punktmengen, d.h. der Rotationsmatrix $\boldsymbol{R}$ und des Translationsvektors $\boldsymbol{t}$
  - in jedem Punkt $\boldsymbol{x}_i$ tritt ein Fehler $e_i$ in ungenauen Datensätzen auf

$$\boldsymbol{x}_i = \boldsymbol{R}\boldsymbol{y}_i + \boldsymbol{t} + e_i$$

- Ziel der Transformations-Algorithmen ist es, die Funktion der Summe der Fehlerquadrate zu minimieren

$$\sum^2 = \sum_{i=1}^{N} \left|\left| \boldsymbol{x}_i - \boldsymbol{R}y_i - \boldsymbol{t} \right|\right|^2$$

# Quaternion Rotation

- Quaternionen sind Vektoren komplexer Zahlen, die 2-D und 3-D Rotationen repräsentieren

$$\boldsymbol{q}_R = (q_0 \quad q_1 \quad q_2 \quad q_3) \quad mit\ q_0 \geq 0\ und\ q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$$

- Vor dem Algorithmus werden die Objekte in den Koordinatenursprung bewegt
  - Schwerpunkte der Objekte ergeben sich dabei aus dem arithmetischen Mittel aller Punkte

$$\mu_X = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i \qquad \mu_Y = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{y}_i$$

  - Berechnung der neuen Punktmengen um den Koordinatenursprung

$$X' : \qquad \boldsymbol{x}_i' = \boldsymbol{x}_i - \mu_X$$

$$Y' : \qquad \boldsymbol{y}_i' = \boldsymbol{y}_i - \mu_Y$$

# Quaternion Methode:

1. Berechnung der Kovarianzmatrix $\sum_{XY} = \sum_{i=1}^{N} \boldsymbol{x}'_i \boldsymbol{y}'^{T}_i$

2. Berechnung der Matrix $\boldsymbol{A} = \sum_{XY} - \sum_{XY}^{T}$ $\quad \boldsymbol{A} = \begin{pmatrix} a_{11} & \boldsymbol{a}_{12} & a_{13} \\ a_{21} & a_{22} & \boldsymbol{a}_{23} \\ \boldsymbol{a}_{31} & a_{32} & a_{33} \end{pmatrix}$

3. Konstruktion des symmetrischen Matrix

$$\boldsymbol{Q} = \begin{pmatrix} sp(\sum_{XY}) & \Delta^T \\ \Delta & \sum_{XY} + \sum_{XY}^{T} - sp(\sum_{XY})\mathrm{I}_3 \end{pmatrix} \qquad \Delta = \begin{pmatrix} a_{23} \\ a_{31} \\ a_{12} \end{pmatrix}$$

4. Berechnung des normierten Eigenvektors des größten positiven Eigenwertes von $\boldsymbol{Q}$

$$\boldsymbol{q}_R = (q_0 \quad q_1 \quad q_2 \quad q_3)$$

5. Berechnung der orthogonalen Rotationsmatrix und des Translationsvektors

$$\boldsymbol{R} = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{pmatrix}$$

$$\boldsymbol{t} = \mu_X - \boldsymbol{R}\mu_Y$$

# Singulärwertzerlegung

- Auf Singulärwertzerlegung basierende Technik (singular value decomposition, SVD)
- Vor dem Algorithmus werden die Objekte, wie in der Quaternion Methode, in den Koordinatenursprung bewegt

$$\mu_X = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i \qquad \mu_Y = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{y}_i$$

$$X' : \qquad \boldsymbol{x}'_i = \boldsymbol{x}_i - \mu_X$$

$$Y' : \qquad \boldsymbol{y}'_i = \boldsymbol{y}_i - \mu_Y$$

# Singulärwertzerlegung

Sei $\boldsymbol{A}$ $n \times n$ Matrix $\quad \boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^T$

$\boldsymbol{U}$ – $n \times m$ Matrix der Eigenvektoren von $\boldsymbol{A}\boldsymbol{A}^T$

$\boldsymbol{V}$ – $m \times n$ Matrix der Eigenvektoren von $\boldsymbol{A}^T\boldsymbol{A}$

$$\boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{V}^T\boldsymbol{V} = \mathrm{I}$$

$\boldsymbol{\Lambda}$ – Diagonalisierung der Singulärwerte $s_i$ von $\boldsymbol{A}$, die sich aus den Eigenwerten $\lambda_i$ von $\boldsymbol{A}$ ergeben:

$$s_i = \sqrt{\lambda_i} \quad , i = (1 \ldots n)$$

$$\boldsymbol{\Lambda} = diag(s_1 \ldots s_n) = \left(\sqrt{\lambda_1} \ldots \sqrt{\lambda_n}\right) \qquad s_1 \geq \ldots \geq s_n > 0$$

# SVD Methode:

1. Berechnung der Kovarianzmatrix $\sum_{XY} = \sum_{i=1}^{N} \boldsymbol{x}'_i \boldsymbol{y}'^T_i$

2. Finden der Singulärwertzerlegung der Kovarianzmatrix, so das gilt:

$$\sum_{XY} = \boldsymbol{U}\Lambda\boldsymbol{V}^T$$

3. Berechnung der Rotationsmatrix $\hat{\boldsymbol{R}} = \boldsymbol{U}\boldsymbol{V}^T$

4. Prüfen der Determinante um die Gültigkeit des Ergebnisses sicherzustellen

Wenn $\det \left| \hat{\boldsymbol{R}} \right| = 1 \qquad \Rightarrow \boldsymbol{R} = \hat{\boldsymbol{R}}$

Andernfalls, wenn $\det|\hat{\boldsymbol{R}}| = -1$, versagt der Algorithmus, da die berechnete Rotationsmatrix lediglich eine Spieglung darstellt. Man braucht eine Garantie, das nur gültige Rotationsmatrizen berechnet werden.

Bildung der Matrix $\boldsymbol{V}' = \boldsymbol{V} \bullet \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} = \begin{pmatrix} v_{11} & v_{12} & -v_{13} \\ v_{21} & v_{22} & -v_{23} \\ v_{31} & v_{32} & -v_{33} \end{pmatrix}$

Berechnung der Rotationsmatrix $\boldsymbol{R} = \boldsymbol{V}'\boldsymbol{U}^T$

# Orthogonale Matrizen

- Weiterentwicklung der Quaternion Methode

Algorithmus:

1. Berechnung der Kovarianzmatrix $\sum_{XY} = \sum_{i=1}^{N} \boldsymbol{x}'_i \boldsymbol{y}'^{T}_i$

2. Berechnung von Eigenwerten und Eigenvektoren

$$\lambda_{i=1..3}: \; ch(\lambda) = \det\left|\sum_{XY} \sum_{XY}^{T} - \lambda \mathrm{I}\right| = 0$$

$$\hat{\boldsymbol{u}}_{i=1..3}: \; \left(\sum_{XY} \sum_{XY}^{T} - \lambda_i \mathrm{I}\right) \hat{\boldsymbol{u}}_i = 0$$

Zerlegung der symmetrischen Matrix

$$\sum_{XY} \sum_{XY}^{T} = \lambda_1 \hat{\boldsymbol{u}}_1 \hat{\boldsymbol{u}}_1^T + \lambda_2 \hat{\boldsymbol{u}}_2 \hat{\boldsymbol{u}}_2^T + \lambda_3 \hat{\boldsymbol{u}}_3 \hat{\boldsymbol{u}}_3^T$$

3. Wenn $\sum_{XY}\sum_{XY}^{T}$ positiv definit $(\lambda_i > 0\ \forall i$ bzw. $rg\left(\sum_{XY}\sum_{XY}^{T}\right)=3)$

   Berechnung der inversen Matrix

$$\boldsymbol{S}^{-1} = \frac{\hat{\boldsymbol{u}}_1\hat{\boldsymbol{u}}_1^T}{\sqrt{\lambda_1}} + \frac{\hat{\boldsymbol{u}}_2\hat{\boldsymbol{u}}_2^T}{\sqrt{\lambda_2}} + \frac{\hat{\boldsymbol{u}}_3\hat{\boldsymbol{u}}_3^T}{\sqrt{\lambda_3}}$$

   ansonsten, wenn $rg\left(\sum_{XY}\sum_{XY}^{T}\right)=2, \text{d.h. } \lambda_3 = 0 \Rightarrow \hat{\boldsymbol{u}}_3 \text{ linear abhängig}$

   Berechnung der pseudo-inversen Matrix

$$\boldsymbol{S}^{+} = \frac{\hat{\boldsymbol{u}}_1\hat{\boldsymbol{u}}_1^T}{\sqrt{\lambda_1}} + \frac{\hat{\boldsymbol{u}}_2\hat{\boldsymbol{u}}_2^T}{\sqrt{\lambda_2}}$$

4. Berechnung der Rotationsmatrix

   $\boldsymbol{R} = \boldsymbol{\Sigma}_{XY}\boldsymbol{S}^{-1}$ bzw.

   $\boldsymbol{R} = \max\left\{\boldsymbol{\Sigma}_{XY}\boldsymbol{S}^{+} \pm \hat{\boldsymbol{u}}_3\hat{\boldsymbol{u}}_3^T\right\}$ $(\det(\boldsymbol{R})$ positiv $\Leftrightarrow$ gültige Rotation, statt Spieglung$)$

# Duale Quaternion Methode

- Die duale Quaternion Methode unterscheidet sich signifikant von den vorherigen Algorithmen
- Die Methode wurde ursprünglich entwickelt um die folgende Fehlerfunktion zu minimieren

$$\sum^2 = \sum_{i=1}^{L} \alpha_i \left|\left| \boldsymbol{n}_{x_i} - \boldsymbol{R}\boldsymbol{n}_{y_i} \right|\right|^2 + \sum_{i=1}^{N} \beta_i \left|\left| \boldsymbol{x}_i - \boldsymbol{R}\boldsymbol{y}_i - \boldsymbol{t} \right|\right|^2$$

$\boldsymbol{n}_{x_i}, \boldsymbol{n}_{y_i}$ Normalenvektoren der Objektgrenzen $\boldsymbol{x}_i, \boldsymbol{y}_i$ $(i = 1 \ldots L)$

$\alpha_i, \beta_i$ Gewichte der Punkt- und Normalenentsprechungen

- Für die Ermittlung der Rotationsmatrix und des Translationsvektors der Punktentsprechung genügen Gewichte von $\alpha_i = 0, \beta_i = 1$

- Rotation und Translation werden durch Quaternionen, bestehend aus zwei Teilen, dargestellt

$$\boldsymbol{q}_X = \begin{pmatrix} \boldsymbol{r} & \boldsymbol{s} \end{pmatrix}$$

mit den Eigenschaften $\boldsymbol{r}^T \boldsymbol{r} = 1,\ \boldsymbol{s}^T \boldsymbol{r} = 0$

- Die Transformation zwischen den Objekten wird als Rotation $\theta$ um und Translation $\boldsymbol{t}$ entlang einer dreidimensionalen Geraden mit dem Richtungsvektor $\boldsymbol{n}$ durch den Punkt $\boldsymbol{p}$ vollzogen

$$\boldsymbol{r} = \begin{pmatrix} \sin\left(\frac{\theta}{2}\right)\boldsymbol{n} \\ \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \qquad \boldsymbol{s} = \begin{pmatrix} \frac{1}{2}\cos\left(\frac{\theta}{2}\right)\boldsymbol{n} + \sin\left(\frac{\theta}{2}\right)(\boldsymbol{p} \times \boldsymbol{n}) \\ -\frac{t}{2}\sin\left(\frac{\theta}{2}\right) \end{pmatrix}$$

# Algorithmus:

1. Definiere $\boldsymbol{Q}(v) = \begin{pmatrix} v_3 \mathrm{I} + \boldsymbol{K}(\boldsymbol{v}_{0\ldots2}) & \boldsymbol{v}_{0\ldots2} \\ -\boldsymbol{v}_{0\ldots2}^T & v_3 \end{pmatrix}$ $\boldsymbol{W}(v) = \begin{pmatrix} v_3 \mathrm{I} - \boldsymbol{K}(\boldsymbol{v}_{0\ldots2}) & \boldsymbol{v}_{0\ldots2} \\ -\boldsymbol{v}_{0\ldots2}^T & v_3 \end{pmatrix}$

mit $\boldsymbol{K}(v) = \begin{pmatrix} 0 & -v_2 & v_1 \\ v_2 & 0 & -v_0 \\ -v_1 & v_0 & 0 \end{pmatrix}$

Berechnung der Matrizen

$$\boldsymbol{C}_1 = \sum_{i=1}^{N} \boldsymbol{Q}(\tilde{x}_i)\boldsymbol{W}(\tilde{y}_i)$$

$$\boldsymbol{C}_3 = 2\sum_{i=1}^{N} \left(\boldsymbol{W}(\tilde{x}_i) - \boldsymbol{Q}(\tilde{y}_i)\right)$$

$$\boldsymbol{C}_4 = \sum_{i=1}^{N} \left(\tilde{x}_i^T \tilde{x}_i + \tilde{y}_i^T \tilde{y}_i\right)$$

wobei $\tilde{x}_i = \frac{1}{2}\begin{pmatrix} x_i \\ 0 \end{pmatrix}$, $\tilde{y}_i = \frac{1}{2}\begin{pmatrix} y_i \\ 0 \end{pmatrix} \in \mathbf{R}^4$

2. Berechnung der Quaternion Rotation ***r*** als Eigenvektor des größten Eigenwertes $\lambda_{max}$ von

$$\boldsymbol{A} = \frac{1}{4N}\boldsymbol{C}_3^T\boldsymbol{C}_3 - \boldsymbol{C}_1 \qquad \boldsymbol{r}: \ \left(\boldsymbol{A} - \lambda_{\max}\mathrm{I}\right)\boldsymbol{r} = 0$$

3. Berechnung der Rotationsmatrix

$$\boldsymbol{R} = (r_3^2 - \boldsymbol{r}_{0\ldots2}^T\boldsymbol{r}_{0\ldots2})\mathrm{I} + 2\boldsymbol{r}_{0\ldots2}\boldsymbol{r}_{0\ldots2}^T + 2r_3\boldsymbol{K}(\boldsymbol{r}_{0\ldots2})$$

4. Berechnung der Translation

$$\boldsymbol{t} = \boldsymbol{W}(r)^T s \qquad s = -\frac{1}{2N}\boldsymbol{C}_3\boldsymbol{r}$$

# Methoden des Abgleichs mit Entsprechungen

- Jeder der vier Algorithmen hat seine Vorteile bzw. Nachteile in Genauigkeit, Stabilität und Effizienz
- Keine erheblichen Unterschiede bei der Anwendung der Methoden auf begrenzten 3D-Daten
  - Orthogonal Matrix Methode ist die schnellste für kleine Datensätze
  - Dual Quaternion Methode ist aufgrund seltener Speicherzugriffe für große Datensätze am effizientesten
- Effizienz hängt bei allen Methoden von der Größe des Cache und vor allem von der Implementierung der Routinen der linearen Algebra ab

# Abgleich ohne Entsprechungen

- Abgleich ohne Entsprechung zwischen 3D-Punktmengen zu kennen (Registration without Correspondences)
- Entsprechungen zwischen den Punkten $\boldsymbol{x}_i$, $\boldsymbol{y}_i$ und die entsprechende Transformation sind zu berechnen

- Algorithmen des Abgleichs ohne sich entsprechender Punktpaare sollten ein optimales Ergebnis, mit möglichst kleinen Fehler, erzielen

# Iterativer Algorithmus des nahsten Punktes

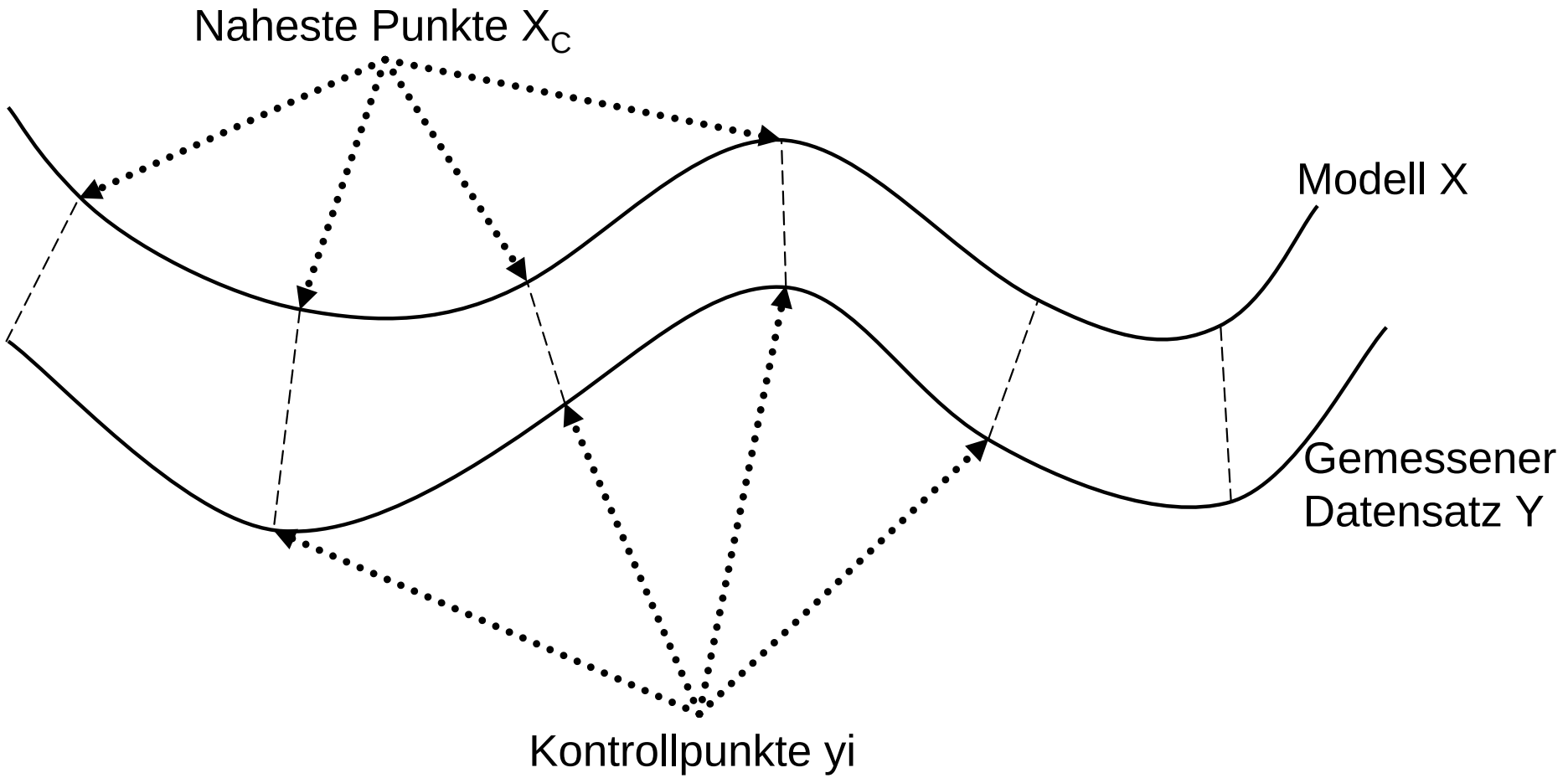
- Besl, McKay, 1992 (Iterative Closest Point Algorithm, ICP)
- Methode, um 3D-Objekte wie Punktmengen, Kurven und Flächen zu erkennen
- Entwickelt, um ein Datensatz eines gemessenen 3D-Objektes $Y$ gegen eine Modell $X$ zu matchen
- Einfacher Algorithmus, da nur zwei Prozeduren
  - Bestimmung des nahsten Punktes des Modells zu einem gegebenen Kontrollpunkt eines geometrischen Objektes
  - Berechnung der Transformation zwischen den zwei entsprechenden (gegebene und ermittelte) Punktmengen

- 3D-Datensatz $Y$ wird in $N_Y$ Punkte und Modell $X$ in $N_X$ Segmente, Punkte oder Dreiecke zerlegt
- Abstandsmetrik zwischen einem gegebenen Punkt $\boldsymbol{y} \in Y$ und Modell $X$

$$d(\boldsymbol{y}, X) = \min_{x \in X} \|\boldsymbol{x} - \boldsymbol{y}\|$$

- Punkt $\boldsymbol{x}_c \in X$ nahster Punkt zu $Y$: $d(\boldsymbol{y}, \boldsymbol{x}_c) = d(\boldsymbol{y}, X)$
- Bilden der Punktmenge $X_C$ aller Punkte $\boldsymbol{x}_c$ der entsprechenden Punkte in $Y$
  - definiere dazu Operator $C$: $X_C = C(Y, X)$
- Ermitteln der Rotationsmatrix $\boldsymbol{R}$, des Translationsvektors $\boldsymbol{t}$, des mittleren quadratischen Abstands $d$, um die zwei Punktmengen $X$, $Y$ zu matchen
  - definiere dazu eine Prozedur $Q$, die z.B. eine der beschriebenen Punktentsprechungs-Methoden (Quaternion Rotation, SVD, orthogonale Matrizen, duale Quaternionen) umsetzt

$$(\boldsymbol{R}, \boldsymbol{t}, d) = Q(X_C, Y)$$


Naheste Punkte $X_C$
Modell X
Gemessener
Datensatz Y
Kontrollpunkte yi

# Algorithmus:

1. Initialisiere $\boldsymbol{R}_0 = \mathrm{I},\ \boldsymbol{t}_0 = 0,\ Y_0 = Y$

2. Wiederhole (für $k = 0,\ k = k+1$) bis der mittlere quadratische Fehler $e$ zwischen zwei Schritten unter einen Schwellwert $\tau > 0$ sinkt

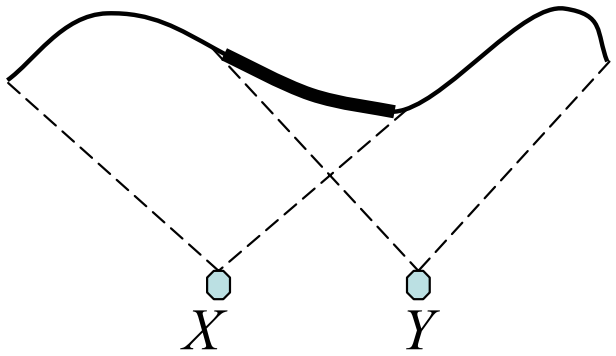
$$e = d_k - d_{k-1} < \tau$$

   a) Ermittle die nahesten Punkte $X_{Ck} = C(Y_k, X)$

   b) Ermittle die Transformation $(\boldsymbol{R}_k, \boldsymbol{t}_k, d_k) = Q(Y_0, X_{Ck})$

   c) Berechne aktuelle Punktmenge $Y_{k+1} = \boldsymbol{R}_k Y_0 + \boldsymbol{t}_k$

- Nutzung eines objektabhängigen Schwellwerts $\tau\sqrt{sp\sum_X}$
- ICP konvergiert für eine optimale Transformation gegen ein lokales Minimum bzgl. des mittleren Abstandes
- ICP stößt bei der Objekt-, Szenenrekonstruktion an seine Grenzen
- Teilweise überlappende Sichten des abzugleichenden Objekts mit dem Modell können zu fehlerhaften Ergebnissen führen
  - es fehlen Punkte, die sich in $X$ und $Y$ entsprechen und Punktpaare bilden können



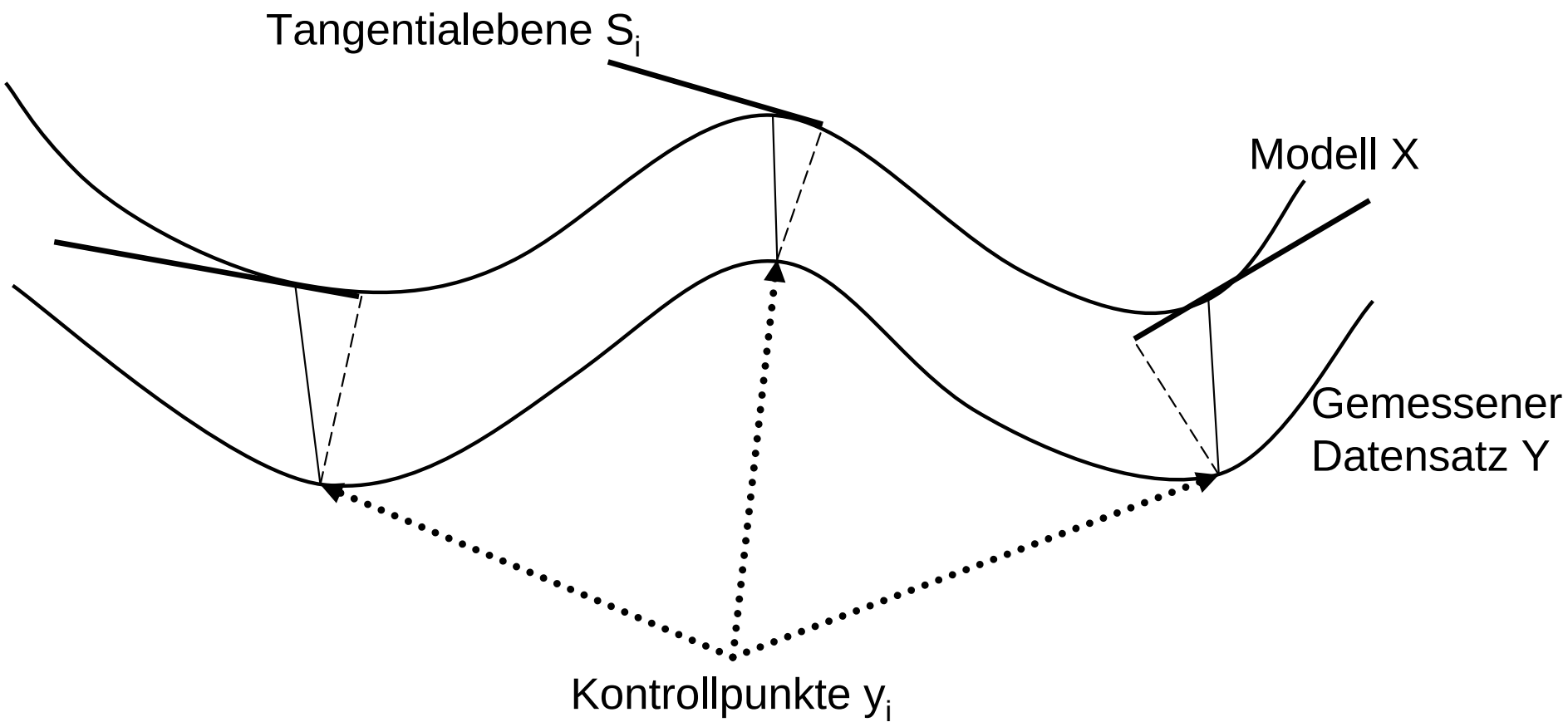
Modifikationen des ICP:

- Verwenden eines Schwellwertes des maximalen Abstandes zwischen den Punktpaaren, die sich in jedem Schritt entsprechen könnten
  - Datensatz $Y$ muss so nicht unbedingt Teil des Modells $X$ sein
- Punkte außerhalb der Flächengrenzen vom Matching ausschließen
  - Entfernen der peripheren Punkte, die signifikante Fehler verursachen
- Nutzen von Gewichten entsprechender Punktpaare

# Iterativer Matching Algorithmus

- Chen, Medioni, 1992 (CM)
- Unterscheidet sich vom ICP nur im Fehlerkriterium
- Minimieren des Abstandes der Punkte des Datensatzes $Y$ zu Tangentenebenen $S_i$ am Modell $X$
  - Normalen an Kontrollpunkten $y_i$ schneiden Modell $X$ in $\boldsymbol{x}_i$
  - Tangentialebenen $S_i$ in $\boldsymbol{x}_i$, um die Transformation, die den Abstand zwischen $S_i$ und $\boldsymbol{y}_i$ minimiert, zu ermitteln
- Fehlerfunktion nach $k$ Schritten

$$\sum^2 = \sum_{i=1}^{N} d_S^2\left(\boldsymbol{T}^k \bullet \boldsymbol{y}_i, S_i^k\right)$$


Tangentialebene $S_i$
Modell X
Gemessener Datensatz Y
Kontrollpunkte $y_i$
Normale auf Y im Punkt $y_i$
Normale auf $S_i$ durch $y_i$

# Algorithmus:

1. Wählen der Kontrollpunkte $\boldsymbol{y}_i \in Y$ $(i=1...N')$ und berechnen der Flächennormalen $n_i$ an diesen Punkten
2. Wiederhole (für $k = 0,\ k = k+1$) bis ein experimentell ermittelter Schwellwert erreicht ist

$$\delta = \frac{\left| e^k - e^{k-1} \right|}{N'} \leq \varepsilon_e \qquad \text{mit } \varepsilon_e > 0$$

   a) Für jeden Kontrollpunkt $\boldsymbol{y}_i$

      i. Berechne mittels der Transformation $T$ die neuen Punkte und Normalen

$$\boldsymbol{y}'_i = \boldsymbol{T}^{k-1} \bullet \boldsymbol{y}_i$$

$$\boldsymbol{n}'_i = \boldsymbol{T}^{k-1} \bullet \boldsymbol{n}_i$$

      ii. Ermitteln der Punkte $\boldsymbol{x}_i$, in denen die Normalen $\boldsymbol{n}'_i$ der $\boldsymbol{y}'_i$ das Modell $X$ schneiden

      iii. Berechnen der Tangentenebenen $S_i$ an den Punkten $\boldsymbol{x}_i$

b) Ermitteln der Transformation $\boldsymbol{T}$, die den Abstand zwischen den Kontrollpunkten $\boldsymbol{y}_i$ und den entsprechenden Ebenen $S_i$ minimiert

- Berechnen des zu $\boldsymbol{y}_i$ nahsten Punktes der Ebene $S_i$
- Berechnen der Transformation mittels der Punktentsprechungs-Methoden (Quaternion Rotation, SVD, orthogonale Matrizen, duale Quaternionen)

c) Aktualisieren der Transformation $\boldsymbol{T}^k = \boldsymbol{T} \circ \boldsymbol{T}^{k-1}$

# Ungenauigkeiten im 3D-Bildabgleich

- Ungenaue Punktdaten in Abgleich-Algorithmen
  - Abweichungen in den ermittelten Punkten werten und nutzen, um den Bildabgleich zu optimieren
- Abgleichsungenauigkeiten bewerten
  - nach der Berechnung der Transformation zwischen den beiden Punktmengen ein Maß der Ungenauigkeit in der aktuellen Transformation ermitteln
- Vereinigen der Punkt- und Abgleichsungenauigkeit
  - kombinieren der individuellen Punktabweichung mit den Abweichungen nach der Transformation, um alle Fehlerquellen vereinigt wiedergeben zu können

- Abweichungen treten in jedem Punkt auf und können in einer Kovarianzmatrix wiedergegeben werden
  - jeder Punkt des ermittelten Objektes mit unterschiedlicher Abweichung
  - Abweichungen unterscheiden sich möglicherweise auch in den Dimensionen des einzelnen Punktes
- Modifizierung der Abgleichsalgorithmen um ermittelte Ungenauigkeiten zu minimieren
  - Punktentsprechungs-Methoden (Quaternion Rotation, SVD, orthogonale Matrizen, duale Quaternionen) durch Techniken, die bekannte Abweichungen in die Berechnung einbeziehen, ergänzen

# Gewichtete Entsprechungen

- Modifizierung des Fehlerkriteriums mit Gewichtung $w_i$ jedes Punktpaares

$$\Sigma^2 = \sum_{i=1}^{N} w_i \left\| \boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t} \right\|^2$$

- Berechnung der Schwerpunkte entsprechend der Gewichte

$$\mu_X = \frac{1}{N} \frac{\sum_{i=1}^{N} w_i \boldsymbol{x}_i}{\sum_{i=1}^{N} w_i} \qquad \mu_Y = \frac{1}{N} \frac{\sum_{i=1}^{N} w_i \boldsymbol{y}_i}{\sum_{i=1}^{N} w_i}$$

- Gewichtete Kovarianzmatrix

$$\Sigma_{XY} = \sum_{i=1}^{N} w_i \boldsymbol{x}'_i \boldsymbol{y}'^T_i$$

$$\boldsymbol{x}'_i = \boldsymbol{x}_i - \mu_X$$
$$\boldsymbol{y}'_i = \boldsymbol{y}_i - \mu_Y$$

# Berechnung der Gewichte:

- In Vereinfachung bezieht sich jedes Gewicht $w_i$ jeweils auf ein Punktpaar $\boldsymbol{x}_i, \boldsymbol{y}_i$
  - nur ein einziger Parameter, der die unterschiedlichen Ungenauigkeiten in beiden Punkten (und ihren Dimensionen) in sich vereinigt

$$w_i = \frac{1}{sp\sum_{X_i} + sp\sum_{Y_i}}$$

  - Spur der Kovarianzmatrizen nur ein grober Wert der absoluten Ungenauigkeit in jedem Punkt
- Keine Quantifizierung des Abgleichfehlers dadurch möglich

# Abgleich mittels EKF

- Abgleichsmethode mit Punktentsprechung von Pennec und Thirion
- Basiert auf den erweiterten Kalman Filter (extended Kalman filter, EKF)
  - rekursive, nichtlineare Technik, die Kovarianzen der individuellen Punkte nutzen und Ungenauigkeiten der Transformation berechnen kann
- Ziel ist die Transformation $\boldsymbol{d}$ mit ihrer Kovarianz $\boldsymbol{\Sigma}_{\boldsymbol{d}}$ zu ermitteln, die die entsprechenden Punktpaare $\boldsymbol{x}_i$, $\boldsymbol{y}_i$ mit jeweiligen Kovarianzen $\boldsymbol{\Sigma}_{xi}$, $\boldsymbol{\Sigma}_{yi}$ optimal abgleicht

# Transformation in Vektorform

- Rotation als Vektor $\boldsymbol{r} \in \boldsymbol{R}^3$ darstellbar
  - Rotation von $\theta$ um ein Gerade mit Richtungsvektor $\boldsymbol{n}$ $(|\boldsymbol{n}|=1)$

$$\boldsymbol{r} = \theta \boldsymbol{n}$$

- Transformationsvektor $\boldsymbol{d}$ gegeben durch Vereinigung des Rotationsvektors $\boldsymbol{r}$ und Translationsvektors $\boldsymbol{t}$

$$\boldsymbol{d} = (\boldsymbol{r}^T \boldsymbol{t}^T)^T$$

- Berechnung der Transformations-Kovarianzmatrix

$$\sum_{\boldsymbol{d}} = E\left[(\boldsymbol{d} - \bar{\boldsymbol{d}})(\boldsymbol{d} - \bar{\boldsymbol{d}})^T\right]$$

mit Mittelwert $\bar{\boldsymbol{d}}$

und Erwartungswert $E(\mathrm{X})$ der Zufallsvariablen $\mathrm{X}$

# Wahl der Punktentsprechungen

- In Matchingalgorithmen wie ICP, sollten optimale Punktentsprechungen in den zwei Datensätzen gewählt werden, um ihn robuster zu gestalten
- Modifizierung des ICP durch den Quadratischern Mahalanobis-Abstand
  - bezieht die Abweichungen in den Punkten und der Transformation mit ein
  - dabei sollte es zusätzlich einen maximal zulässigen Abstand der zu überprüfenden Punktentsprechungen geben
- Schwellwert leicht zu ermitteln, da der Mahalanobis-Abstand auf $\chi^2$-Verteilung mit drei Freiheitsgraden basiert
  - Berechnen des Konfidenzintervalls und ermitteln eines Schwellwert $\varepsilon$ aus den Quantilen der $\chi^2$-Verteilung

- Rotationsmatrix, Translationsvektor sind Funktionen des Transformationsvektors: $\boldsymbol{R} = f(\boldsymbol{d})$ $\boldsymbol{t} = g(\boldsymbol{d})$

- Quadratischer Mahalanobis-Abstand zwischen dem Punkt $\boldsymbol{y}_i$ und dem transformierten Punkt $\boldsymbol{x}_i$

$$d_i^M = (f(\boldsymbol{d})\boldsymbol{x} + g(\boldsymbol{d}) - \boldsymbol{y}_i)^T \sum_i (f(\boldsymbol{d})\boldsymbol{x} + g(\boldsymbol{d}) - \boldsymbol{y}_i)$$

mit der entsprechenden Kovarianzmatrix

$$\sum_i = f(\boldsymbol{d}) \sum_{\boldsymbol{x}} f(\boldsymbol{d})^T + \sigma_{y_i} + \boldsymbol{J}_d \sum_d \boldsymbol{J}_d^T$$

mit $\boldsymbol{J}_d = \dfrac{\partial(f(\boldsymbol{d})\boldsymbol{x} + g(\boldsymbol{d}))}{\partial \boldsymbol{d}}$

# Kombination der Punkt- und Transformationsabweichungen

- Transformation der Punkte $\boldsymbol{x}_i$ und Berechnung der zugehörigen Kovarianz
- Berechnung der neue Punktdaten $(\boldsymbol{x}'_i, \sum'_{x_i})$
  mittels der Transformation $(\boldsymbol{d}, \sum_{\boldsymbol{d}})$

$$\boldsymbol{x}'_i = \boldsymbol{d} \bullet \boldsymbol{x}_i$$

$$\sum'_{x_i} = \boldsymbol{J}_d \sum_d \boldsymbol{J}_d^T$$

# Quellenangaben


- Bennamoun, M.; Mamic, G. J.: 3D object creation for recognition. Fundamental techniques / Uncertainty in 3-D registration. In: Bennamoun, M.; Mamic, G.J.: Object Recognition – Fundamentals and Case Studies. Springer, London 2002, S. 107–122

- Singular Value Decomposition - SVD

  http://www.awi-bremerhaven.de/ GEO/Publ/PhDs/CPorthun/node45.html
- Quaternionen

  http://www.ais.fhg.de/ARC/3D/download/diplom/node75.htm

  http://www.cis.upenn.edu/~kostas/mypub.dir/ijrr99.pdf
- Iterativer Algorithmus des nahsten Punktes

  http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FISHER/ICP/cvoicp.htm