5.2 Triangulation

Landmarke

- auffälliges, (relativ) weit sichtbares Objekt zur Orientierung
- aus allen Richtungen eindeutig anzupeilen
- in der Karte eingetragen
- Koordinaten jeder Landmarke bekannt

Geometrische Triangulation

- Peile Landmarken (mind. 2) an: ermittle präzise Richtungen
- Ermittle Richtungen, ggf. Entfernungen der Landmarken untereinander
- Errechne Position des Roboters

Vorwärtsschnitt bzw. Kreuzpeilung

Rekonstruiere Dreieck aus zwei bekannten Punkten im Referenzsystem und zwei gemessenen Richtungen (= Winkeln im Referenzsystem):

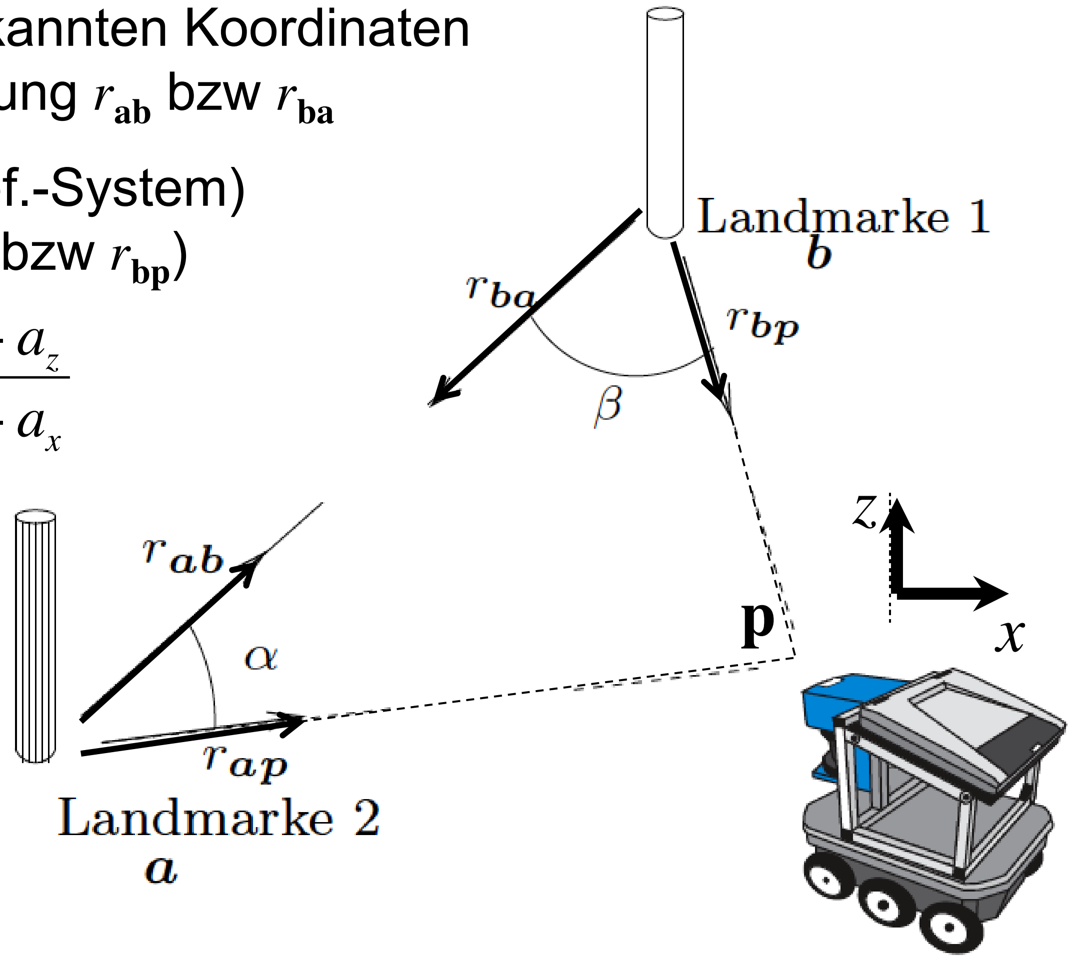
- Strecke **ab** ist gegeben aus bekannten Koordinaten der Landmarken, ebenso Richtung $r_{\mathbf{ab}}$ bzw $r_{\mathbf{ba}}$
- Miss Richtungen (Winkel im Ref.-System) zu **a** und **b** (ausgedrückt als $r_{\mathbf{ap}}$ bzw $r_{\mathbf{bp}}$)

$$\alpha = r_{\mathbf{ap}} - r_{\mathbf{ab}} \qquad \theta_{\mathbf{a},\mathbf{b}} = \arctan\frac{b_z - a_z}{b_x - a_x}$$

$$\beta = r_{\mathbf{ba}} - r_{\mathbf{bp}} \qquad \theta_{\mathbf{a},\mathbf{p}} = \theta_{\mathbf{ab}} + \alpha$$

$$\overline{\mathbf{ap}} = \overline{\mathbf{ab}} \cdot \frac{\sin\beta}{\sin(\alpha+\beta)}$$

$$\mathbf{p} = \begin{pmatrix} p_x \\ p_z \end{pmatrix} = \begin{pmatrix} a_x + \overline{\mathbf{ap}} \cdot \cos\theta_{\mathbf{a},\mathbf{p}} \\ a_z + \overline{\mathbf{ap}} \cdot \sin\theta_{\mathbf{a},\mathbf{p}} \end{pmatrix}$$



Weitere Triangulationsmethoden

… sind im Buch, Abschnitt 5.2 skizziert

… lassen wir hier aus, denn sie sind in der Robotik relativ unüblich

… funktionieren auch auf Robotern, vorausgesetzt man hat Karten mit präzise eingemessenen und eindeutig erkennbaren Landmarken

5.3 Lokalisierungs-Algorithmen

… ist ein Mega-Thema für mobile Robotik
(gewesen – inzwischen gut verstanden),
daher viele Lösungsansätze! Im Folgenden besprochen:

1. Lokalisierung an Scanpunkten (Punktkarte vs. Scanner)
2. Lokalisierung an Linien (Linienkarte vs. Scanner)
3. Lokalisierung an (visuellen) Merkmalen
4. Unimodale probabilistische Lokalisierung
5. Probabilisische Lokalisierung in Rasterkarten
6. wie 5, aber mit Sampling

Nummern 1.-4. beruhen auf inkrementeller Lokalisierung!

5.3.1 Lokalisierung durch 2D-Scanmatching

Gegeben:

- Karte aus Messpunkten
- a priori Poseschätzung (z.B. durch (Gyr-)Odometrie oder GPS)
- aktueller Laserscan

Finde:

Pose in der Karte

Bemerkung: Ist die „Karte“ der vorige Laserscan, bekommt man hier ein Verfahren zur inkrementellen Lokalisierung!

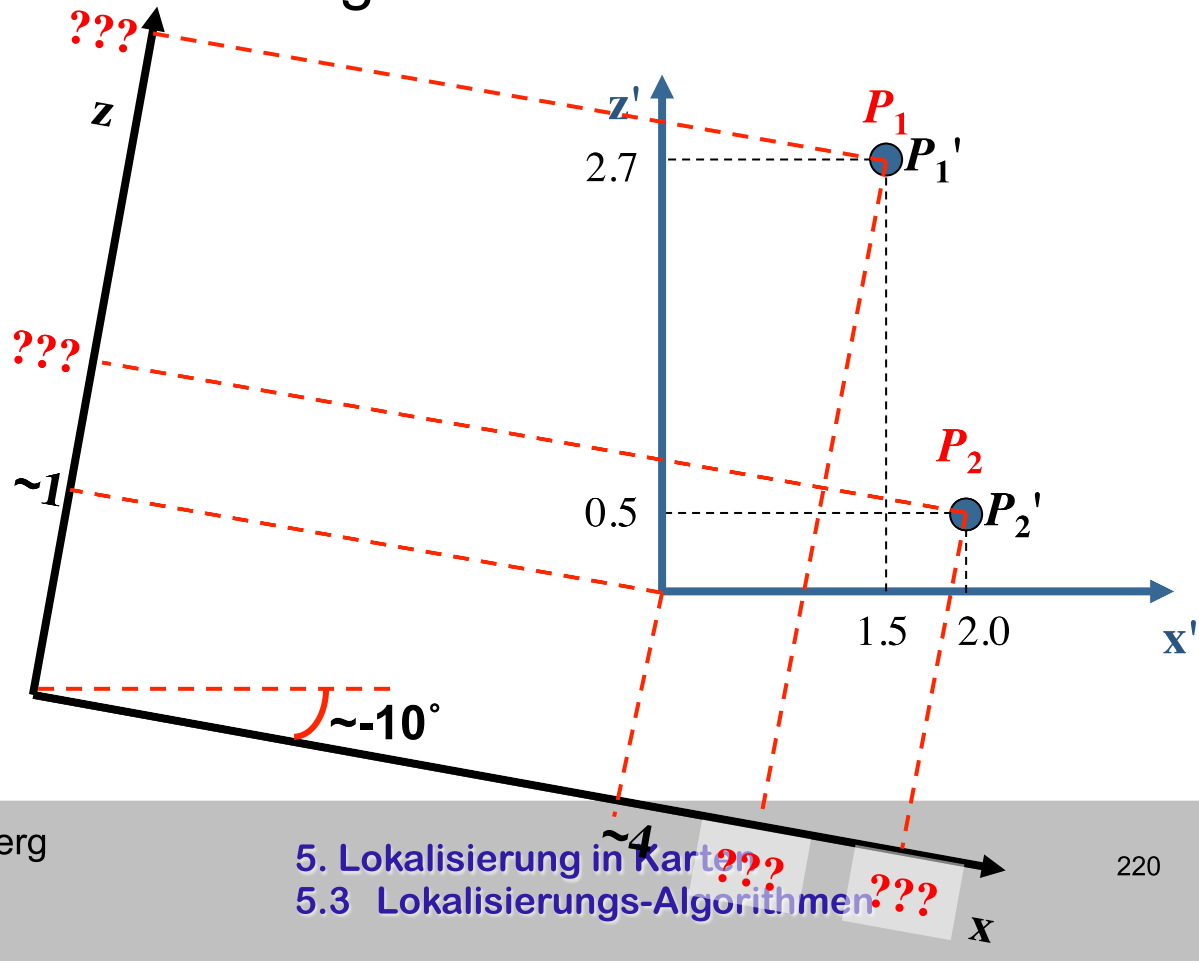
Registrierung von Modell-Punktwolke M und Daten-Punktwolke D (z.B. Punkt-Karte und Laserscan):
Berechnen einer Transformation (Translation & Rotation) von D, dass sie M optimal passend überlagert wird

Lokalisierung durch 2D-Scanmatching

Beispiel:

Nach Poseänderung von ca. $[4,1,-10°]^T$ (Odometrie-Schätzung) finden sich im Scan zwei Punkte P_1' und P_2' (lokales System!). Gäbe es aus voriger Pose im Scan die selben <u>Raumpunkte</u> als P_1, P_2, so könnte die Poseschätzung verfeinert werden!

- Finde/rate korrespondierende Raumpunkte unter den Scanpunkten beider Scans
- Errechne Poseänderung aus Translation& Rotation der entsprechenden Scanpunkte



Zur Erinnerung: Rotations/Drehmatrizen

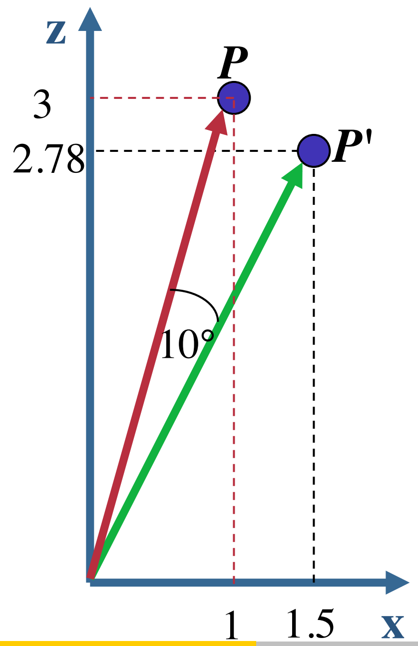
… aus der analytischen Geometrie: Rotation im Bezugssystem um ω (im Uhrzeigersinn!):

$$R_\omega = \begin{bmatrix} \cos\omega & \sin\omega \\ -\sin\omega & \cos\omega \end{bmatrix}$$

$$P' = R_\omega \cdot P$$

Beispiel Drehung von *P* um ω=10°

$$P' = \begin{bmatrix} \cos 10° & \sin 10° \\ -\sin 10° & \cos 10° \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 0.9848 & 0.1736 \\ -0.1736 & 0.9848 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 1.5056 \\ 2.7808 \end{bmatrix}$$

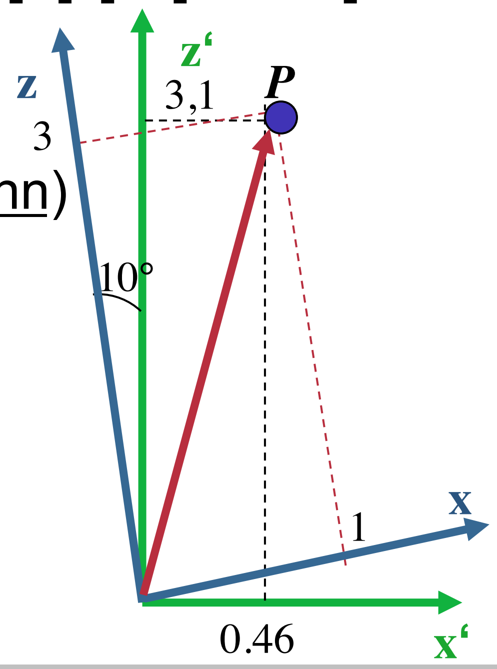


Rotation des Bezugssystems um ω (=Dreh. im Gegenuhrzeigersinn)

$$R_\omega^{-1} = \begin{bmatrix} \cos\omega & -\sin\omega \\ \sin\omega & \cos\omega \end{bmatrix}$$

$$P' = R_\omega^{-1} \cdot P$$

Bspl. $P' = R_{10°}^{-1} \cdot \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 0,4641 \\ 3,1280 \end{bmatrix}$



Scanmatching als Fehlergradientenabstieg

F. Lu, E. Milios: Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans. CVPR94, S.935-938, 1994

Wenn

- wahre N Paare (P_i, P_i') korrespondierender Raumpunkte als Scanpunkte vorkommen (P_i Ref.-/Modellpunkt, P_i' Datenpunkt) und
- alle P_i, P_i' jeweils präzise korrekte Scanwerte haben,

dann

- charakterisiert das Minimum (= Nullstelle) der folgenden Fehlerfunktion die Poseänderung um $[t_x, t_z, \theta]^T$ präzise:

$$E(\theta,T) = \sum_{i=1}^{N} \mathrm{dist}\left(P_i, \mathrm{transf}(P_i')\right) = \sum_{i=1}^{N} \left\| P_i - \left(R_\theta \cdot P_i' + T\right)\right\|^2$$

(Transformation=Rotation+Translation <u>im</u> Referenz-Koord.-System!)

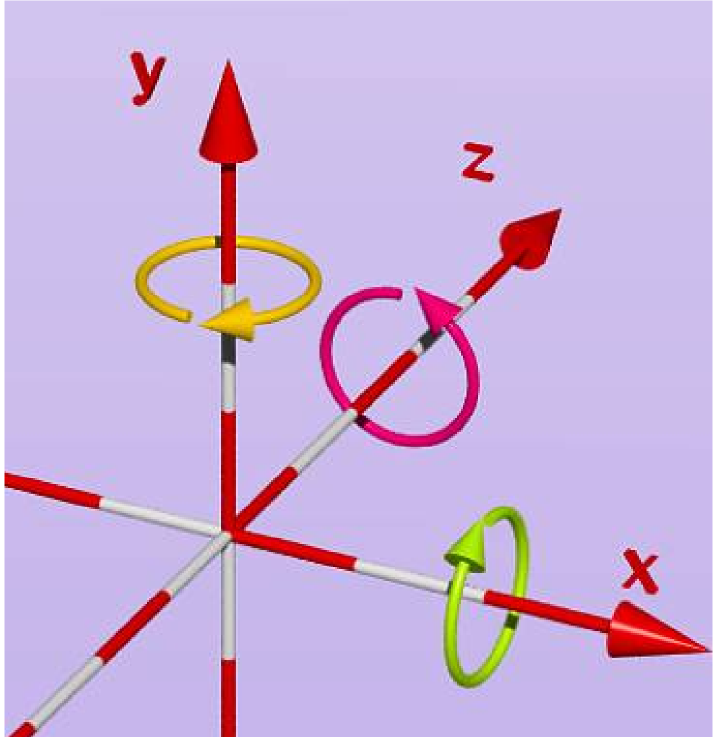
Aber…

Beide Voraussetzungen praktisch nicht erfüllbar!

➥ Minimum ≠0 von E <u>schätzt</u> Poseänderung

➥ Und wie findet man das/ein Minimum?

Erinnerung an Drehrichtungen



- Drehrichtungen des Roboters in dieser Vorlesung bekanntlich entsprechend linkshändigem Koordinatensystem!
- Das heißt: Zwei Rotationen (um x,z-Achsen) im Uhrzeigersinn; Rotation um y-Achse dagegen
- Rotationen im Rahmen der Scantransformation sind hier **durchgängig im math. positiven Sinn**/Gegenuhrzeigersinn definiert!
- Falls e. Rotation aus Scanmatching ermittelt wurde, ist sie entsprechend auf Roboterrotation zurückzurechnen!

Aber...

Fehlerminimierung für (θ, T)

Lemma (Lu/Milios): Die folgende Transformation $(t_x, t_z, \theta)^{\mathrm{T}}$ minimiert die Fehlerfunktion $E(\theta, T)$ bei gegebenen Punkt-korrespondenzen

$$\theta = \arctan \frac{S_{zx'} - S_{xz'}}{S_{xx'} + S_{zz'}}$$

$$t_x = c_x - \left(c'_x \cos\theta - c'_z \sin\theta\right)$$

$$t_z = c_z - \left(c'_x \sin\theta + c'_z \cos\theta\right)$$

Dabei sind

(s. Folie 91 – Linienfinden nach Lu/Milios):

$$c_x = \tfrac{1}{N} \sum_i p_{x,i} \qquad S_{xx'} = \sum_i \left(p_{x,i} - c_x\right)\left(p'_{x,i} - c'_x\right)$$

$$c'_x = \tfrac{1}{N} \sum_i p'_{x.i} \qquad S_{xz'} = \sum_i \left(p_{x,i} - c_x\right)\left(p'_{z,i} - c'_z\right)$$

$$c_z = \tfrac{1}{N} \sum_i p_{z,i} \qquad S_{zx'} = \sum_i \left(p_{z,i} - c_z\right)\left(p'_{x,i} - c'_x\right)$$

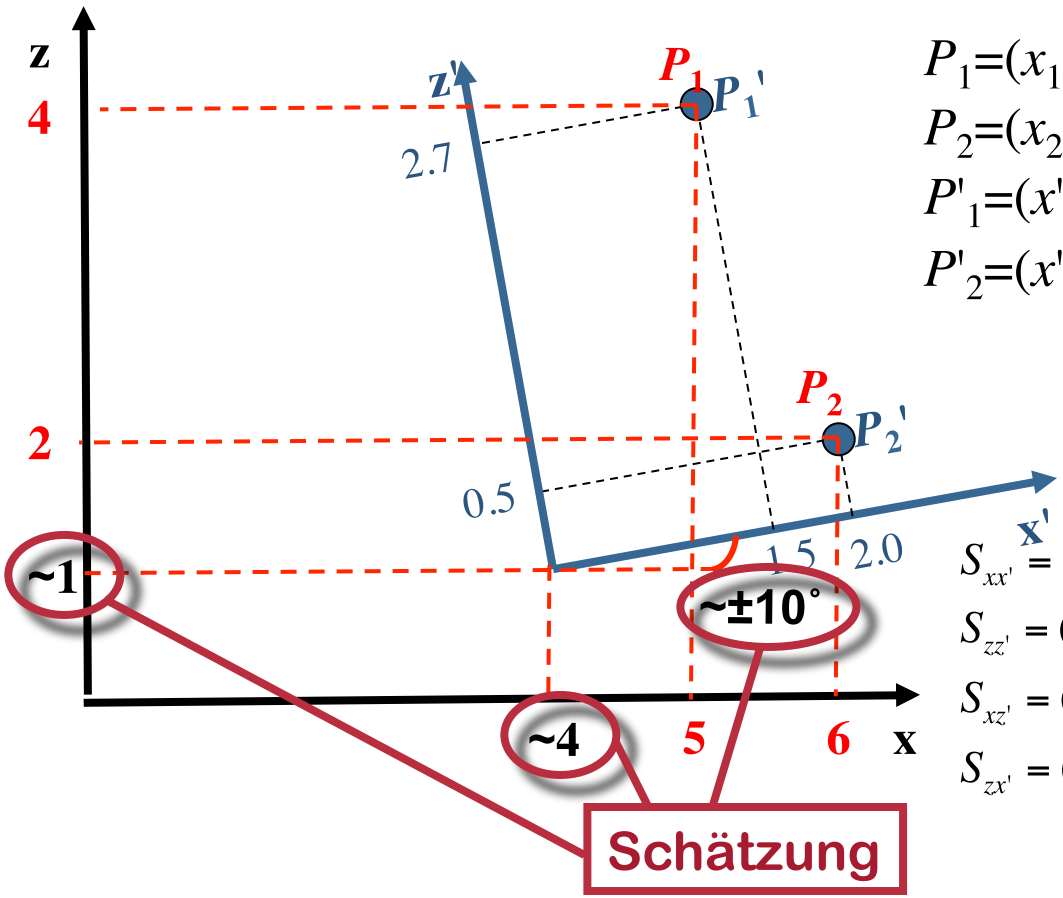
$$c'_z = \tfrac{1}{N} \sum_i p'_{z.i} \qquad S_{zz'} = \sum_i \left(p_{z,i} - c_z\right)\left(p'_{z,i} - c'_z\right)$$

Beweis des Lu/Milios-Lemmas

… s. Buch Kap. 5.3 (S. 180ff)

… und jetzt hier an der Tafel

Beispiel



$P_1=(x_1, z_1) = (5,4)$

$P_2=(x_2, z_2) = (6,2)$

$P'_1=(x'_1, z'_1) = (1.5, 2.7)$

$P'_2=(x'_2, z'_2) = (2, 0.5)$

$c_x = \frac{1}{2}(5+6) = 5.5$

$c_z = 3.0$

$c'_x = 1.75$

$c'_z = 1.6$

$S_{xx'} = (5-5.5)(1.5-1.75)+(6-5.5)(2-1.75) = 0.25$

$S_{zz'} = (4-3)(2.7-1.6)+(2-3)(0.5-1.6) = 2.2$

$S_{xz'} = (5-5.5)(2.7-1.6)+(6-5.5)(0.5-1.6) = -1.1$

$S_{zx'} = (4-3)(1.5-1.75)+(2-3)(2-1.75) = -0.5$

$$\theta = \arctan\frac{S_{zx'} - S_{xz'}}{S_{xx'} + S_{zz'}} = \arctan\frac{0.6}{2.45} = 13{,}7608°$$

$$t_x = c_x - (c'_x \cos\theta - c'_z \sin\theta) = 5.5 - (1.6998 - 0.3806) = 4.1808$$

$$t_z = c_z - (c'_x \sin\theta + c'_z \cos\theta) = 3 - (0.4163 + 1.5541) = 1.0396$$

➥ $\Delta\theta_{robot} = \underline{-3.76°}$

$\theta_{robot} = \underline{-13.76°}$

➥ $\Delta T = (0.18, 0.04)$

Schätzung hier nicht gebraucht. Aber …

Grundidee 2D-Scanmatching-Algorithmus

1. Nimm geschätzten Roboterposeversatz (θ, T) (mit Odometrie etc.) als Start-Schätzung des Scanposeversatzes
2. Finde <u>auf Basis der aktuellen Schätzung</u> des Scanposeversatzes (θ, T) korrespondierende Punkte in Modellscan und Datenscan
3. Aktualisiere (θ, T) durch Minimierung von $E(\theta, T)$ wie gehabt; bestimme dabei $(\Delta\theta, \Delta T)$ (Änderung von (θ, T) gegenüber vorher)
4. Ist $(\Delta\theta, \Delta T)$ kleiner als vorgegebene Schranke, terminiere mit aktuellem (θ, T) als ermitteltem Scanposeversatz; ansonsten gehe zu 2 mit (θ, T) als aktueller Schätzung

Finden/Raten der „richtigen" Korrespondenzen spart Laufzeit; Neuberechnen von Korrespondenzen in #2 kostet Laufzeit!