Die ⊞-Methode zur Behandlung von Mannigfaltigkeiten in Sensorfusionsalgorithmen

Udo Frese, Christoph Hertzberg und René Wagner

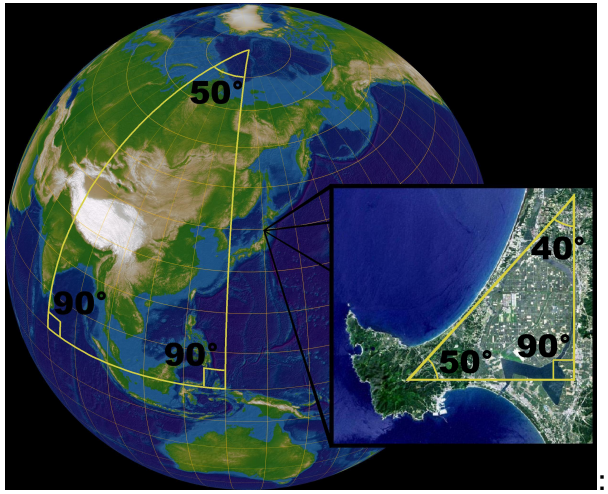
DFKI Sichere Kognitive Systeme

Workshop Lokalisationstechniken im DFKI (6./7.4.2011)

Größen auf Mannigfaltigkeiten

Eine Mannigfaltigkeit ist [...] ein topologischer Raum, der lokal dem euklidischen Raum $\mathbb{R}^n$ gleicht. Im Ganzen muss die Mannigfaltigkeit jedoch nicht einem euklidischen Raum entsprechen [...].

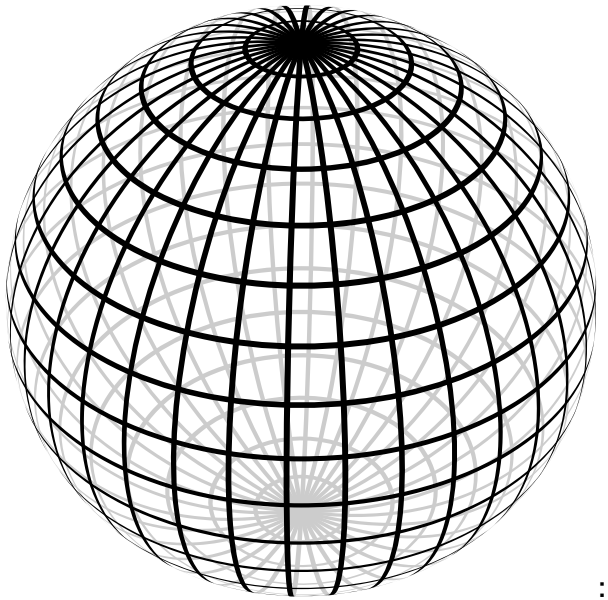
Quelle: Wikipedia



Quelle: wikipedia

Größen auf Mannigfaltigkeiten: Sphäre

- Beispiel: Punkt auf Sphäre (z.B. Erde)
- Beispiel: Richtung im Raum
- $S^2 = \{x \in \mathbb{R}^3 \mid |x| = 1\}$
- 2 Freiheitsgrade
- tradionell: Längen und Breite $x(\lambda, \phi)$
- Singularität an den Polen ($\phi = \pm\pi/2 = \pm 90°$)
- Kleine Änderungen von x erfordern große Änderung von (λ, ϕ)
- Besser: $x \in \mathbb{R}^3$ als Vektor mit Nebenbedingung $|x| = 1$

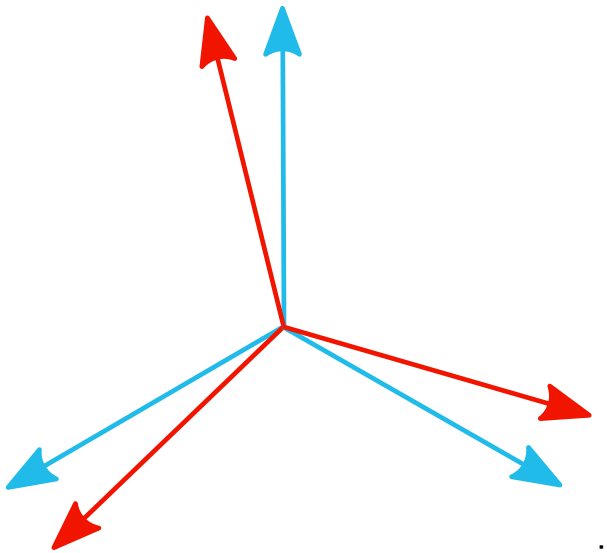


Quelle: wikipedia

Größen auf Mannigfaltigkeiten: 3D-Orientierung

- Beispiel: 3D-Orientierung
- Ursprung, Abstand und Händigkeit erhaltende Abbildung $\mathbb{R}^3 \mapsto \mathbb{R}^3$

$$SO(3) = \{Q : \mathbb{R}^3 \mapsto \mathbb{R}^3 | \\ Q(0) = 0, \\ |Q(u) - Q(v)| = |u - v|, \\ \text{sgn}\,|Q(u)Q(v)Q(w)| = \text{sgn}\,|uvw| \\ \}$$



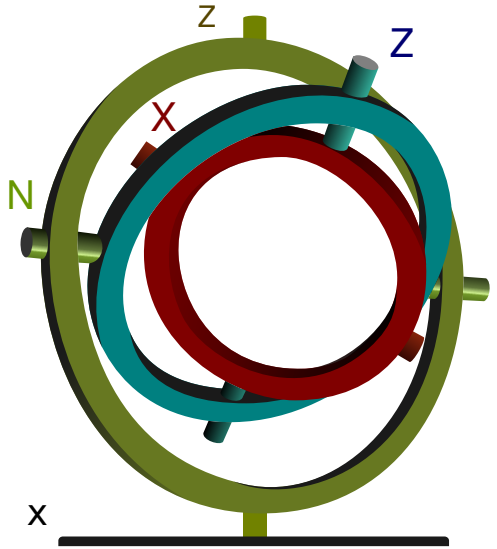
:

Quelle: wikipedia

- 3 Freiheitsgrade
- Rot2Blau: Vektor in Rot-Koordinaten abgebildet auf selben Vektor in Blau-Koordinaten (alias)

Größen auf Mannigfaltigkeiten: 3D-Orientierung

- Beispiel: 3D-Orientierung
- traditionell: Euler-Winkel $Q(\alpha, \beta, \gamma)$
- 3 Zahlen, 3 Freiheitsgrade
- Singularität bei $\alpha = 0$
- alternative bei $\alpha = \pm\pi/2$

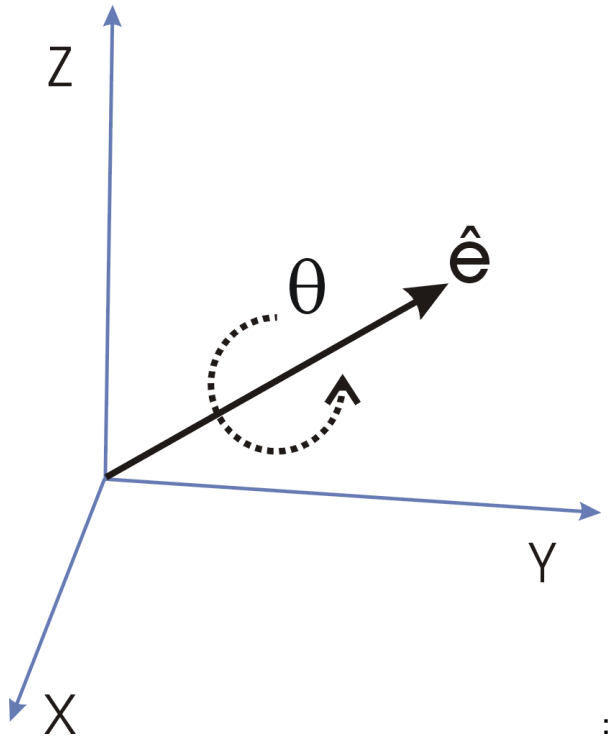


:

Quelle: wikipedia

Größen auf Mannigfaltigkeiten: 3D-Orientierung

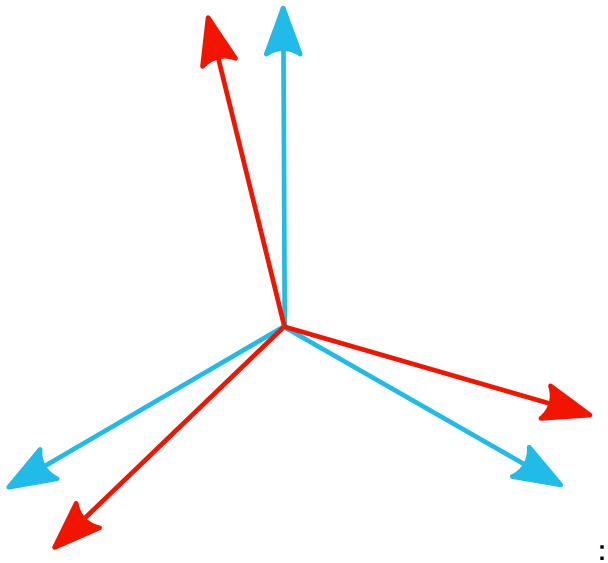
- Beispiel: 3D-Orientierung
- traditionell: angle-axis $Q(e)$
- 3 Zahlen, 3 Freiheitsgrade
- Singularität bei $\theta = |e| = 2\pi$



Quelle: wikipedia

Größen auf Mannigfaltigkeiten: 3D-Orientierung

- Beispiel: 3D-Orientierung
- besser: Orthonormale Matrizen
- $\left\{Q \in \mathbb{R}^{3\times 3} \mid Q^T Q = I, |Q| = 1\right\}$
- 9 Zahlen, 3 Freiheitsgrade
- Nebenbedingung $Q^T Q = I$

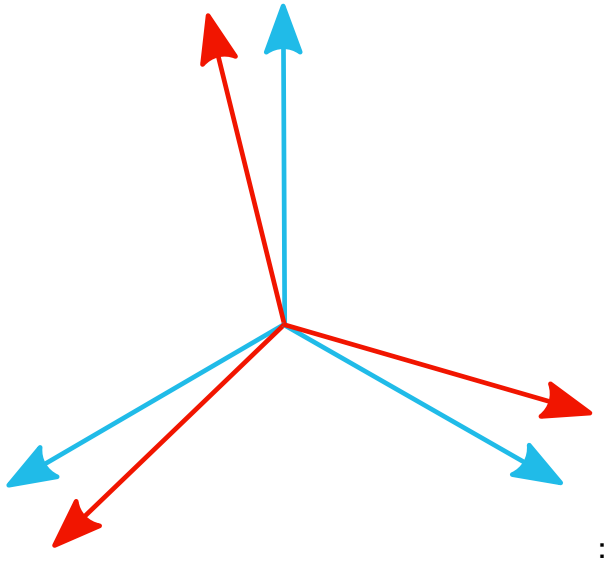


:

Quelle: wikipedia

Größen auf Mannigfaltigkeiten: 3D-Orientierung

- Beispiel: 3D-Orientierung
- besser: Einheitsquaternionen
- $\{q \in \mathbb{R}^4 \mid |q| = 1\}$
- 4 Zahlen, 3 Freiheitsgrade
- Nebenbedingung $|q| = 1$



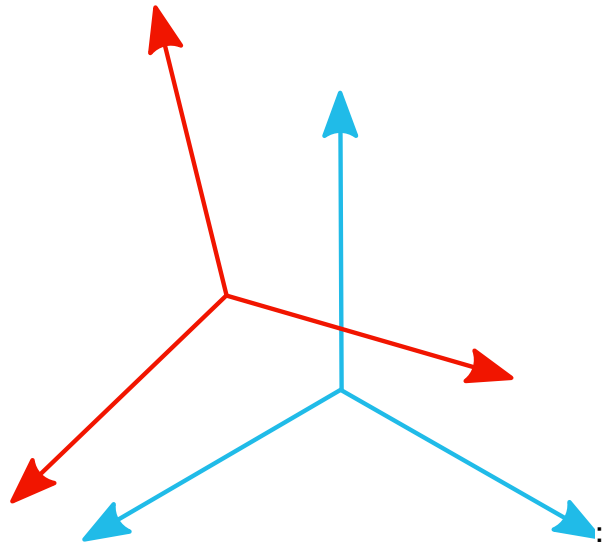
:

Quelle: wikipedia

Größen auf Mannigfaltigkeiten: 3D-Pose

- Beispiel: 3D-Posen
- Abstand und Händigkeit erhaltende Abbildung $\mathbb{R}^3 \mapsto \mathbb{R}^3$

$$SE(3) = \{Q : \mathbb{R}^3 \mapsto \mathbb{R}^3 | \\ |Q(u) - Q(v)| = |u - v|, \\ \operatorname{sgn} |Q(u)Q(v)Q(w)| = \operatorname{sgn} |uvw| \\ \}$$



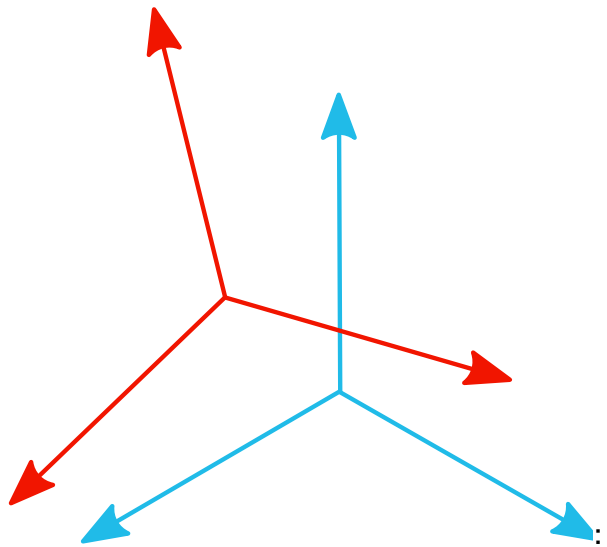
Quelle: wikipedia

- 6 Freiheitsgrade (3 Position, 3 Orientierung)
- Rot2Blau: Punkt in Rot-Koordinaten abgebildet auf selben Punkt in Blau-Koordinaten (alias)

Größen auf Mannigfaltigkeiten: 3D-Pose

- Beispiel: 3D-Posen

$$\left\{ \begin{bmatrix} Q & t \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \middle| \begin{array}{l} Q \in \mathbb{R}^{3\times 3}, t \in \mathbb{R}^3, \\ Q^T Q = I, |Q| = 1, \end{array} \right\}$$



Quelle: wikipedia

Einschub: Namensgebung in Programmen

- Formale Genauigkeit Variablennamen ist *enorm nützlich*
- Bei Punkten Koordinatensystem Im Namen nennen:
  `pInA` ist Punkt `p` in `A`-Koordinaten
- Alle Relativposen `AInB` nennen (4×4 Matrix):
  so dass $\texttt{pInB} = \texttt{AInB} \cdot \texttt{pInA}$
- Multiplikation mit `AInB` von links konvertiert den selben Punkt von `A` nach `B` Koordinaten
- Weltkoordinaten: `pInWorld`, `AInWorld`, ggf. weglassen `p`, `A`
- Posen von Objekt als ObjectInWorld nicht WorldInObject

Einschub: Namensgebung in Programmen

- Inverse von `AInB` ist `BInA`
- Konsistenz von Gleichungen rechts nach links ablesbar: $\texttt{pInCam} = \texttt{CamInWorld}^{-1} \cdot \texttt{ObjectInWorld} \cdot \texttt{pInObject}$
- Koordinatensysteme sinnvoll wählen, z.B. Z immer nach oben
- bei Geräten Koordinatensysteme physisch sichtbar wählen (Kanten) oder anzeichnen
- Bei Quaternionen: Klasse für Kombination von Orientierung q und Translation t, $v \mapsto q \cdot v \cdot q^{-1} + t$ mit Methoden für Komposition und Invertierung, wie bei Matrizen

Schätzalgorithmen auf Mannigfaltigkeiten

- Schätzalgorithmen kombinieren unsichere Messungen zur einer möglichst genauen Schätzung eines unbekannten Zustandes,...
- z.B. quadratische Ausgleichsrechnung (Kalibrierung, Modelfitting, SLAM)
- z.B. (Extended/Unscented) Kalmanfilter (Lokalisation, Tracking)
- ..., indem sie Unsicherheit probabilistisch modellieren,...
- mit *Erwartungswert*, *Kovarianz*, *Gaußverteilung*
- ..., und die *wahrscheinlichste* Lösung suchen.

Schätzalgorithmen auf Mannigfaltigkeiten

- Probleme für Schätzalgorithmen auf Mannigfaltigkeiten
- Algorithmen betrachten Zustand als Vektor
- Singularitäten führen zu Fehlfunktion
- Wie berücksichtigt man die Nebenbedingung beim Suchen der *wahrscheinlichsten* Lösung?
- Wie definiert man *Erwartungswert*, *Kovarianz* und *Gaußverteilung* auf Mannigfaltigkeiten?

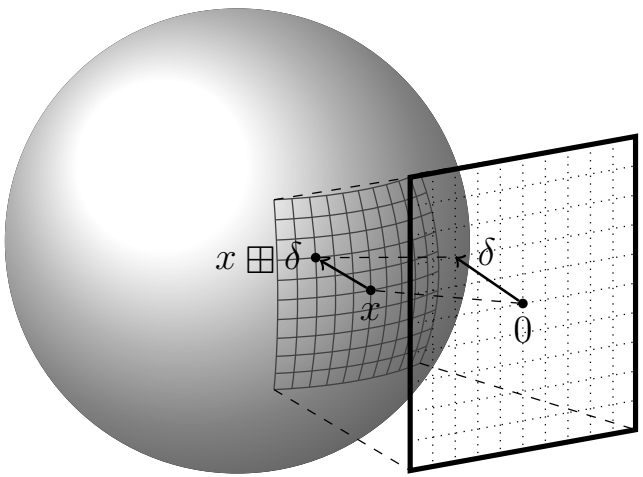
⊞-Mannigfaltigkeiten

- Lösung: ⊞-Methode
- Schätzalgorithmen arbeiten in Schritten
- Betrachte in jedem Schritt Mannigfaltigkeit lokal als Vektorraum,...
- ... aber akkumuliere Schritte topologisch richtig in der Mannigfaltigkeit
- Verkapseln globale topologische Struktur in Operatoren ⊞ und ⊟

⊞-Mannigfaltigkeiten

- ⊞-Mannigfaltigkeit S mit Operatoren

$$\boxplus_{\mathcal{S}} : \mathcal{S} \times \mathbb{R}^n \to \mathcal{S}, \quad (1)$$
$$\boxminus_{\mathcal{S}} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}^n. \quad (2)$$



- $y = x \boxplus \delta$ verändert x um vektorielles δ
- $\delta = y \boxminus x$ berechnet vektorielle Änderung δ von x nach y

⊞-Mannigfaltigkeiten

"We write this operation [the state update] as $x \rightarrow x + \delta x$, even though it may involve considerable more than vector addition." *W. Triggs [TMHF00, p. 7]*

- Beitrag der ⊞-Methode [Her08, HWFS10, WBF11]
- Vage Idee formalisiert, inkl. Erwartungswert, Kovarianz und Gaußverteilung
- Notation und Verkapselung ⊞ und ⊟
- Least-Squares und UKF adaptiert durch, im Wesentlichen, Ersetzen von $+$ durch ⊞ und $-$ durch ⊟

Wahrscheinlichkeitsverteilungen auf ⊞-Mannigfaltigkeiten

- $\mu \in \mathcal{S}$, X Zufallsvariable in R^n
- $Y := \mu \boxplus X$ Zufallsvariable in $\mathcal{S}$
- Gaußverteilung: $\mathcal{N}(\mu, \Sigma) := \mu \boxplus \mathcal{N}(0, \Sigma)$, $\mu \in \mathcal{S}$, $\Sigma \in \mathbb{R}^{n \times n}$

Erwahrtungswert und Kovarianz auf ⊞-Mannigfaltigkeiten

- $\mathrm{E}\, X \stackrel{?}{=} \int_{\mathcal{S}} x \cdot \mathrm{p}(X = x) \mathrm{d}x$. undefiniert
- Gesucht: Äquivalente Formulierung für $\mathbb{R}^n$ die auf ⊞-Mannigfaltigkeit definiert ist
- $\mathrm{E}\, X = \operatorname{argmin}_{x \in \mathcal{S}} \mathrm{E}(\|X \boxminus x\|^2)$
- $\Rightarrow \mathrm{E}(X \boxminus \mathrm{E}\, X) = 0$
- $\operatorname{Cov} X = \mathrm{E}\left((X \boxminus \mathrm{E}\, X)(X \boxminus \mathrm{E}\, X)^\top\right)$
- X Zufallsvariable in $\mathcal{S}$: $\mathrm{E}\, X \in \mathcal{S}$, $\operatorname{Cov} X \in \mathbb{R}^{n \times n}$

Vektorräume

- Beispiel: $\mathbb{R}^n$

$$x \boxplus \delta = x + \delta, \quad y \boxminus x = y - x.$$

3D Orientierungen

- Beispiel: 3D Orientierungen als Matrix
- In $x \boxplus \delta$, Drehung um δ (angle-axis) relativ zu x
- In $y \boxminus x$, Drehung y relativ zu x als angle-axis

$$x \boxplus \delta = x \exp \delta, \quad y \boxminus x = \log \left(x^{-1} y \right),$$

$$\exp \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos\theta + cx^2 & -sz + cxy & sy + cxz \\ sz + cxy & \cos\theta + cy^2 & -sx + cyz \\ -sy + cxz & sx + cyz & \cos\theta + cz^2 \end{bmatrix},$$

$$\theta = \sqrt{x^2 + y^2 + z^2}, s = \operatorname{sinc}\theta, c = \frac{1 - \cos\theta}{\theta^2}$$

$$\log x = \frac{\theta}{2 \sin\theta} \begin{bmatrix} x_{32} - x_{23} \\ x_{13} - x_{31} \\ x_{21} - x_{12} \end{bmatrix}, \theta = \arccos \frac{\operatorname{tr} x - 1}{2}.$$

3D Orientierungen

- Beispiel: 3D Orientierungen als Quaternion
- In $x \boxplus \delta$, Drehung um δ (angle-axis) relativ zu x
- In $y \boxminus x$, Drehung y relativ zu x als angle-axis

$$x \boxplus \delta = x \cdot \exp \frac{\delta}{2}, \quad y \boxminus x = 2\,\overline{\log}(y^{-1} \cdot x),$$

$$\exp \delta = \begin{bmatrix} \cos \|\delta\| \\ \operatorname{sinc} \|\delta\|\, \delta \end{bmatrix},$$

$$\overline{\log} \begin{bmatrix} w \\ v \end{bmatrix} = \begin{cases} 0 & v = 0 \\ \frac{\operatorname{atan}(\|v\|/w)}{\|v\|} v & v \neq 0, w \neq 0 \\ \pm \frac{\pi/2}{\|v\|} v & w = 0. \end{cases}$$

Least-Squares auf ⊞-Mannigfaltigkeiten

Classical Least-Squares

$f : \mathbb{R}^{n} \to \mathbb{R}^{m}$

$f(X) - z \sim \mathcal{N}(0, \Sigma)$

Least-Squares on a ⊞-Manifold

$f : \mathcal{S} \to \mathcal{M}$

$f(X) \boxminus z \sim \mathcal{N}(0, \Sigma)$

Least Squares

$$\hat{x} = \underset{x \in \mathbb{R}^{n}}{\operatorname{argmin}} \|f(x) - z\|_{\Sigma}^{2}$$
$$= \underset{x \in \mathbb{R}^{n}}{\operatorname{argmin}} (f(x) - z)^{T} \Sigma^{-1} (f(x) - z)$$

$$\hat{x} = \underset{x \in \mathcal{S}}{\operatorname{argmin}} \|f(x) \boxminus z\|_{\Sigma}^{2}$$
$$= \underset{x \in \mathcal{S}}{\operatorname{argmin}} (f(x) \boxminus z)^{T} \Sigma^{-1} (f(x) \boxminus z)$$

Gauß-Newton auf ⊞-Mannigfaltigkeiten

Classical Gauss-Newton

$f : \mathbb{R}^{n} \to \mathbb{R}^{m}$

$f(X) - z \sim \mathcal{N}(0, \Sigma)$

Gauss-Newton on a ⊞-Manifold

$f : \mathcal{S} \to \mathcal{M}$

$f(X) \boxminus z \sim \mathcal{N}(0, \Sigma)$

Iterate with initial guess x_0 until x_i converges:

$$J_{\bullet k} := \frac{f(x_i + \varepsilon e_k) - f(x_i - \varepsilon e_k)}{2\varepsilon}$$

$$x_{i+1} := x_i - (J^\top \Sigma^{-1} J)^{-1} J^\top \Sigma^{-1} (f(x_i) - z)$$

$$J_{\bullet k} := \frac{(f(x_i \boxplus \varepsilon e_k) \boxminus z) - (f(x_i \boxplus -\varepsilon e_k) \boxminus z)}{2\varepsilon}$$

$$x_{i+1} := x_i \boxplus -(J^\top \Sigma^{-1} J)^{-1} J^\top \Sigma^{-1} (f(x_i) \boxminus z)$$

MTK, MTKM und SLoM

- Bibliotheken für ⊞-Methode unter C++ (MTK, SLoM) [HWFS10] und MATLAB (MTKM) [WBF11]
- `www.openslam.org`
- **M**anifold **T**ool**k**it (for **M**atlab)
- **S**parse **L**east **S**quares on **M**anifolds
- Vektoren, 2D /3D Orientierung, 2D/ 3D Pose, als ⊞-Mannigfaltigkeit vordefiniert
- Mechanismum um ⊞-Mannigfaltigkeiten als Klasse zusammenzusetzen (mathematisch: $\times$)
- ⊞, ⊟ und Indexverwaltung für Klasse automatisch generiert
- Generische Implementierung von Gauß-Newton, Levenberg-Marquardt, UKF
- Automatische Ausnützung der Dünnbesetztheit des Problems

```
function z= project_point(intrinsic,cam2world,p_world)

p_cam = inv(cam2world.transform()) * [p_world;1]; % Transform to cam coords
z = pinhole(intrinsic.focal_length.vec, intrinsic.offset.vec, intrinsic.
     distortion.vec, p_cam(1:3));
```

```
load_data;
% define custom manifold compound type
intrinsic_t = mtk.make_compound('focal_length', @mtk.Rn,1,
                                'offset', @mtk.Rn,2,
                                'distortion', @mtk.Rn,1);
% compute initial intrinsic and extrinsic parameters
compute_initial_parameters;
% create optimization problem
o = mtk.OptimizationProblem();
% add intrinsic parameters
intrinsic_id = o.add_random_var(intrinsic);

cam2world_id_for_img = cell(num_images);
for i=1:num_images  % for each calibration image
    p_img = Z{i}; p_world = M{i};  % get measurements

    % add extrinsic parameter (camera pose)
    cam2world_id_for_img{i} = o.add_random_var(cam2world_for_img{i,1});

    % add measurements (checkerboard corner points)
    for j=1:size(p_img,2)
        o.add_measurement(p_img(:, j), @project_point, {intrinsic_id,
             cam2world_id_for_img{i}}, {p_world(:, j)}, Sigma);
    end;
end

[X] = nrlm(@o.fun, o.X0);  % solve problem
```

Zusammenfassung

- Größen auf Mannigfaltigkeiten erfordern Sonderbehandlung in Schätzalgorithmen
- ⊞-Methode verkapselt diese in ⊞ und ⊟ Operatoren
- axiomatisiert, mit Erwartungswert, Kovarianz und Gaußverteilung
- Softwareunterstützung durch MTK, MTKM und SLoM
- vermitteln auch zwischen Sicht als flacher Vektor und struktierte Klasse
- erzeugt ⊞ und ⊟ Operatoren für Klassen
- verwaltet Indizes
- Gauß-Newton, Levenberg-Marquardt und UKF
- “Nur das Problem hinschreiben”

Literatur

Hertzberg, C.:

A Framework for Sparse, Non-Linear Least Squares Problems on Manifolds, Universität Bremen, Diplomarbeit, 2008. –
openslam.org/slom

Hertzberg, C. ; Wagner, R. ; Frese, U. ; Schröder, L.:

Integrating Generic Sensor Fusion Algorithms with Sound State Representations through Encapsulation of Manifolds.
In: *Information Fusion* (submitted March 2010). –
(to appear)

Triggs, W. ; McLauchlan, P. ; Hartley, R. ; Fitzgibbon, A.:

Bundle Adjustment – A Modern Synthesis.
In: Triggs, W. (Hrsg.) ; Zisserman, A. (Hrsg.) ; Szeliski, R. (Hrsg.): *Vision Algorithms: Theory and Practice*.
Springer Verlag, 2000 (LNCS), S. 298–375

Wagner, R. ; Birbach, O. ; Frese, U.:

Rapid Development of Manifold-Based Graph Optimization for Multi-Sensor Calibration and SLAM.
In: *Proceedings of the International Conference on Intelligent Robots and Systems*, 2011. –
(under review)