A Framework for Sparse, Non-Linear Least Squares Problems on Manifolds

Ein Rahmen für dünnbesetzte, nichtlineare quadratische Ausgleichsrechnung auf Mannigfaltigkeiten

Christoph Hertzberg

Universität Bremen
Fachbereich 3 Mathematik und Informatik

Diplomkolloquium, 18.12.2008

Motivation

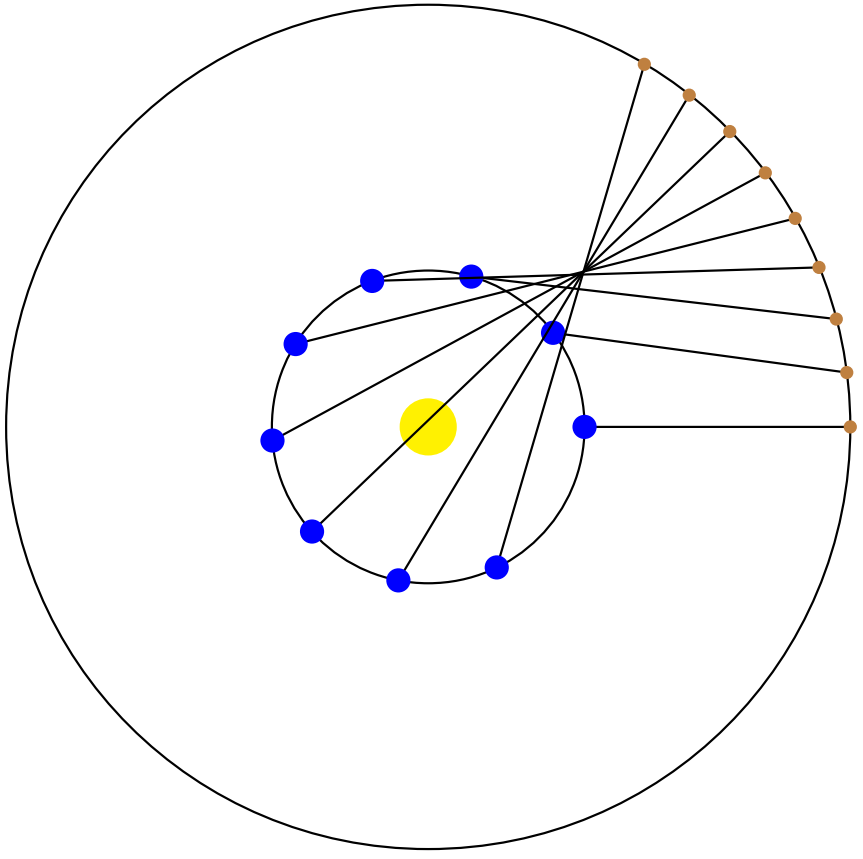
Quadratische Ausgleichrechnung

- Geht zurück auf Carl Friedrich Gauß (1777-1855)
- Taucht (fast) immer auf, wenn aus fehlerbehafteten Messungen Zustände geschätzt werden sollen
- Heute Standardverfahren in experimentellen Naturwissenschaften, Robotik, ...

Entstehung

Entdeckung des Ceres

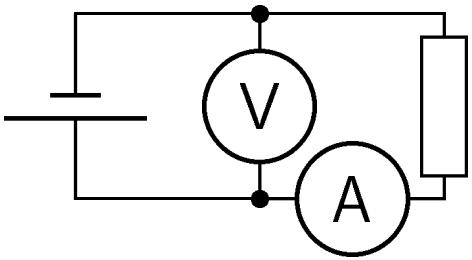
- 1801 entdeckt Giuseppe Piazzi Ceres
- 40 Nächte lang sichtbar, bevor er hinter der Sonne verschwand
- Problem: Wo und wann ist er wieder sichtbar?
- Einzige brauchbare Schätzung von Gauß mittels quadratischer Ausgleichsrechnung

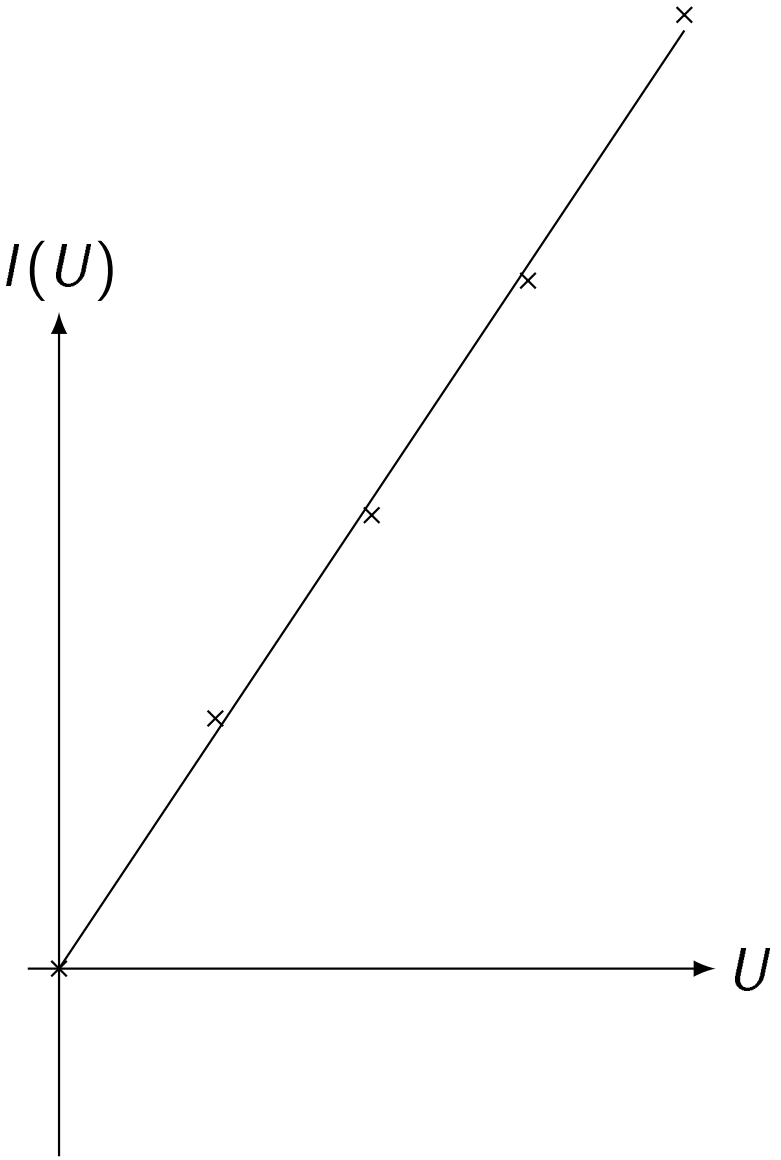


Einführungsbeispiel aus Physikunterricht

Messung eines Widerstands

- Einfacher Stromkreis
- Verschiedene Spannungen anlegen, Stromstärke messen
- Widerstand $R = \frac{U}{I}$, beste Schätzung aus Messwerten?





Verfahren

Gegeben

- (Theoretische) Messfunktion $I(R, U) = \frac{1}{R}U$
- Gemessene Werte $i_{1:n}, u_{1:n}$,
- Gesucht: „beste“ Schätzung für R.

Ansatz

- Minimiere $rss(R) = \sum_{k=1}^{n}(I(R, u_k) - i_k)^2$.

Lösung

- Durch Ableiten nach $G = \frac{1}{R}$ und Nullsetzen erhält man

$$G = \frac{\sum u_k i_k}{\sum u_k^2} \Rightarrow R = \frac{\sum u_k^2}{\sum u_k i_k}$$

Allgemeine Formulierung

Gegeben

- Ein Vektor von *Zufallsvariablen* $[X_1, \ldots, X_n]^\top =: X \in \mathbb{R}^n$
- Eine *Messfunktion* $f : \mathbb{R}^n \to \mathbb{R}^m$
- Eine *Messung* $z \in \mathbb{R}^m$

Gesucht

- Schätzung $\hat{x} = \operatorname{argmin}_x \|f(x) - z\|^2$.
- Wahrscheinlichste Lösung unter der Annahme, dass f_i unabhängig normalverteilt sind mit gleicher Varianz.

Lösung des Problems

Linearer Fall

- Falls $f(x) = Ax + b$ ist das Problem direkt lösbar
- $\hat{x} = (A^\top A)^{-1} A^\top (z - b)$

Nichtlinearer Fall

- Bei den meisten Anwendungen ist f nicht linear
- Keine allgemeine, direkte Lösung möglich
- Iterative Lösung,
  - Ausgangsschätzung x_0
  - f linearisieren, d. h. $f(x_0 + \delta) \approx f(x_0) + J \cdot \delta$
  - linearen Fall lösen daraus $x_1 = x_0 + \delta$ etc.

Simultaneos Localization and Mapping

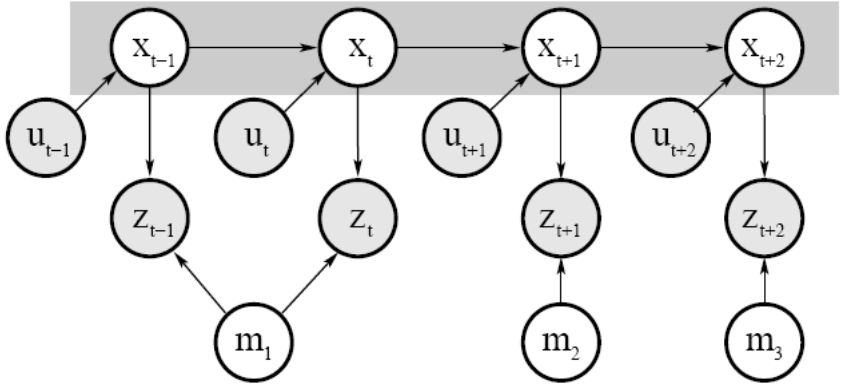
Problemstellung

- SLAM kombiniert zwei Probleme
  - Aus Karte und Sensorinformationen Position bestimmen ⇒Localization
  - Aus Position und Sensorinformation Karte bestimmen ⇒Mapping
- Problem ist als LS Problem interpretierbar. Zu schätzender Zufallsvektor besteht aus
  - Position und Orientierung (=*Pose*) des Roboters zu diskreten Zeitpunkten
  - Einzelne Landmarken der Karte
- Messfunktion besteht aus
  - Zustandsübergangsmessungen (Odometrie), gibt an wie sich Roboter von einem Zeitpunkt zum nächsten bewegt hat
  - Landmarkenmessung, gibt an wo sich Landmarken bzgl. Roboter befinden

Lösungsansätze

Eigenschaften des Problems

- Markov-Eigenschaft, d. h. Pose x_t hängt nur von letzter Pose x_{t-1} und Odometrie u_t ab.
- Wenn die Posen bekannt sind, sind die Landmarken stochastisch unabhängig voneinander



Quelle: `http://robots.stanford.edu/probabilistic-robotics/`

Klassischer Lösungsansatz

Bayes-Filter

- Nutzt die Markov-Eigenschaft aus
- Es werden nur die letzte Pose sowie alle Landmarken gespeichert (inklusive Kovarianz)
- Solange alle Messungen linear, ist optimale Lösung mittels Kalman-Filter berechenbar, andernfalls nur Approximation des bestmöglichen Ergebnisses.
- Onlinefähig (nach jedem Schritt ist aktuell beste Karte verfügbar)
- Problem trotzdem: Kovarianz enthält Einträge für jede Kombination von Landmarken
  $\Rightarrow$Speicherplatz $O(n^2)$, Rechenkosten pro Schritt $\approx O(n^2)$.

Lösung als vollständiges Least Squares Problem

Nachteile klassischer Verfahren

- Durch Linearisieren von Messungen gehen Informationen verloren, die nicht mehr zurückgewonnen werden können.
- Rechenzeit verhältnismäßig hoch

Neue Idee

- Alle Zustände und Messungen behalten
- Klassisches, nichtlineares Least-Square-Verfahren anwenden
- Jetzt nicht mehr onlinefähig, dafür bestmögliche Lösung berechenbar
- Mit etwas Tricks annähernd lineare Speicher- und Rechenkosten ($\Rightarrow$ Sparsity)

Motivation

Problem

- Least Squares Verfahren funktioniert zunächst nur wenn der Zustandsraum einen (euklidischen) Vektorraum bildet.
- Orientierungen im $\mathbb{R}^3$ bilden keinen Vektorraum (lassen sich nicht addieren)

Lösungsmöglichkeit

- Parametrisieren des Zustandsraumes M, z. B. durch Eulerwinkel. Allgemein Abbildung:

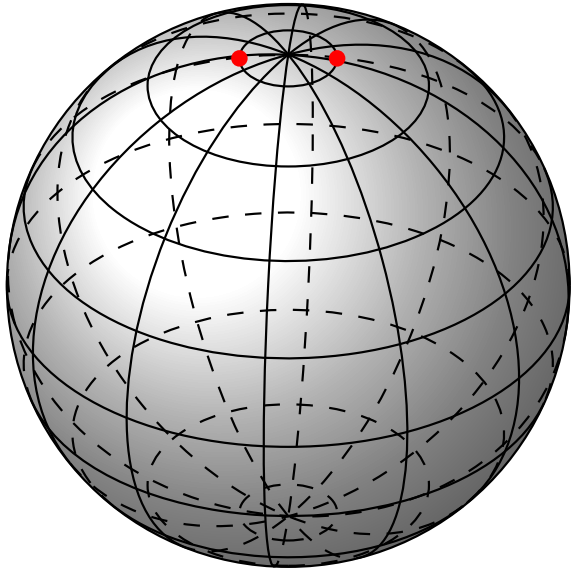
$$\varphi : \mathbb{R}^d \to M$$

- Algorithmus arbeitet auf dem Parameterraum $\mathbb{R}^d$
- Problem: Darstellung nicht singularitätenfrei.
  - Was heißt das?

Singularitäten

Beispiel Kugeloberfläche

- Parametrisierung durch Längen- und Breitengrad
- Problem in der Nähe der Pole $\varphi(-90°, 80°)$ und $\varphi(90°, 80°)$ liegen nah beieinander



Formal

- Parametrisierung $\varphi : \mathbb{R}^d \to M$ hat Singularität, wenn ihre Inverse unstetig ist
- D. h. um kleine Änderungen in M zu bewirken sind große Änderungen in $\mathbb{R}^d$ nötig

Weiterer Lösungsansatz

Überparametrisieren

- Punkte auf Kugeloberfläche als Teilmenge von $\mathbb{R}^3$ auffassen
- LS Algorithmus liefert in jedem Schritt kleine Änderung
- Durch Aufaddieren der Änderung wird im Allgemeinen die Oberfläche verlassen.
  - Lösbar durch Renormalisieren nach jedem Schritt
- Problem jetzt:
  - Algorithmus glaubt im $\mathbb{R}^3$ zu rechnen obwohl der Zustandsraum nur zwei Freiheitsgrade hat.
  - D. h. es müssen mehr Freiheitsgrade als eigentlich nötig berechnet werden
    - Insbesondere ein Problem, wenn z. B. mittels fünf Messungen zwei Punkte auf der Oberfläche geschätzt werden sollen
    - Auch sonst gehen tendenziell Informationen verloren

Mannigfaltigkeiten

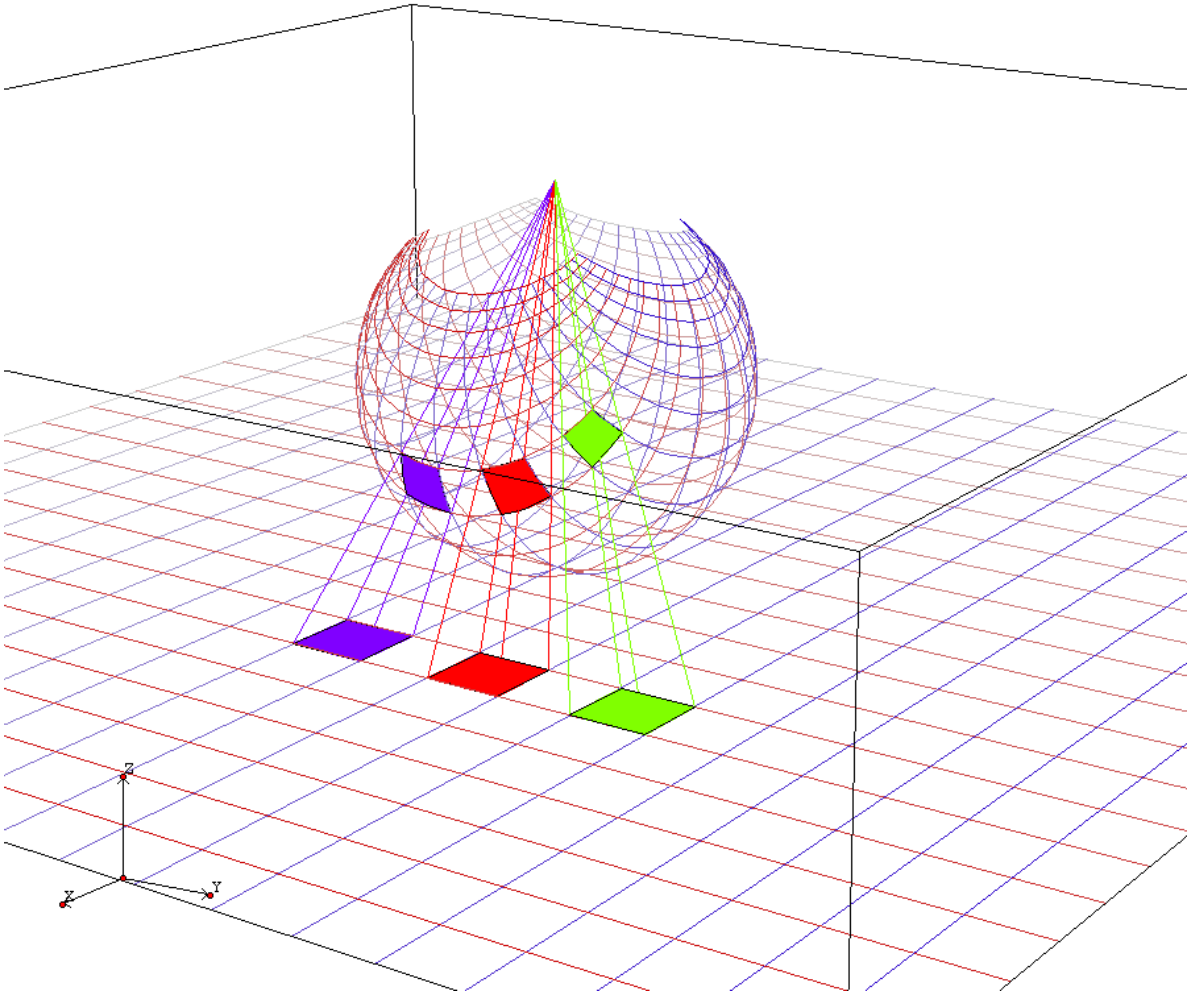
Kombination aus beiden Ansätzen

- Zustandsraum global überparametrisieren
- Lokal minimale Parametrisierung
- Formalisierung führt zu Mannigfaltigkeiten

Vorgehensweise

- Zustandsraum mit Karten (charts) überdecken
- Einzelne Karten bedecken nicht den ganzen Zustandsraum, haben aber Überlappungen mit benachbarten Karten
- LS-Algorithmus rechnet jeweils nur innerhalb einer Karte (minimale Parametrisierung)
- Durch geeignete Wahl der Karte wird vermieden, dass sich der Zustand einer Singularität nähert

Beispiel Kugeloberfläche



Quelle:
`http://xahlee.org/MathGraphicsGallery_dir/sphere_projection/`

Kapselung

Kapselungsoperatoren

- Handhabung der Karten und Koordinatentransformationen sollte nicht dem LS-Algorithmus überlassen werden.
- Umgehung durch zwei Kapselungsoperatoren für Zustandsraum M:

$$\boxplus : M \times \mathbb{R}^d \to M,$$
$$\boxminus : M \times M \to \mathbb{R}^d$$

- $\boxplus$ addiert lokal kleine Änderungen auf einen Zustand
- $\boxminus$ ist sozusagen Inverse von $\boxplus$, d. h. $x \boxplus (y \boxminus x) = y$
- Im Algorithmus $+/-$ durch $\boxplus/\boxminus$ ersetzen
  $\Rightarrow$ Algorithmus arbeitet ohne weitere Anpassungen auf beliebigen Mannigfaltigkeiten

Motivation

Problem

- In vielen LS-Problemen ist die Zahl der Zustände und Messungen sehr hoch
- Bei naiver vorgehensweise reicht heutiger Arbeitsspeicher nicht aus
  $\Rightarrow$Bei 10000 Freiheitsgraden und 50000 Messungen:
  - über 8GB Speicher
  - über 1000s alleine zum Multiplizieren zweier Matrizen

„Dünne“ Abhängigkeiten

Struktur typischer LS-Problem

- Viele einzelne Messungen, die jeweils nur von wenigen (2-3) Variablen abhängen.
- Dadurch entstehen in der Jakobi-Matrix der kombinierten Funktion sehr viele Nullen
- Diese lassen sich sowohl bei Speicherung als auch beim Lösen von Gleichungssystemen ausnutzen
- Laufzeit und Speicherbedarf fast linear zur Zahl der Messungen (Problemabhängig)

Framework

- Es wurde ein C++-Framework erstellt, welches das Formulieren von beliebigen LS-Problemen sehr vereinfacht.
- Anwender muss lediglich
  - seine Zufallsvariablen und Messfunktionen definieren (unter Zuhilfenahme von vordefinierten Makros)
  - Daten für die Messfunktionen einlesen, ggf. zu schätzende Zufallsvariablen geeignet vorinitialisieren
  - Anschließend Optimierungsfunktion aufrufen und die optimierten Daten wieder auslesen.

Vorbereitung

```
// Define 2D-Pose:
MAKE_POSE(Pose, Vect<2>, pos , SO2, orientation , )

// Define Odometry Measurement:
BUILD_MEASUREMENT(Odo, 3, ((Pose, t0)) ((Pose, t1)),
                          ((Pose_T, odo)))
double* Odo::eval(double ret[3]) const
{
  Pose_T diff = t0->world2Local(*t1);
  diff.sub(ret, odo);
  return ret+3;
}
```

Dateneingabe

```
Estimator e;

// Data holding for variables and measurements:
deque<Pose> poses;
deque<Odo> odo;

Pose_T delta;
int from, to;

while(getNextPose(delta, from, to)){
  if(to>=poses.size()){
    poses.push_back(poses[from]->local2World(delta));
    e.insertRV(&poses.back());
  }
  odo.push_back(Odo(poses[from], poses[to], delta));
  e.insertMeasurement(&odo.back());
}
```

Ergebnisse

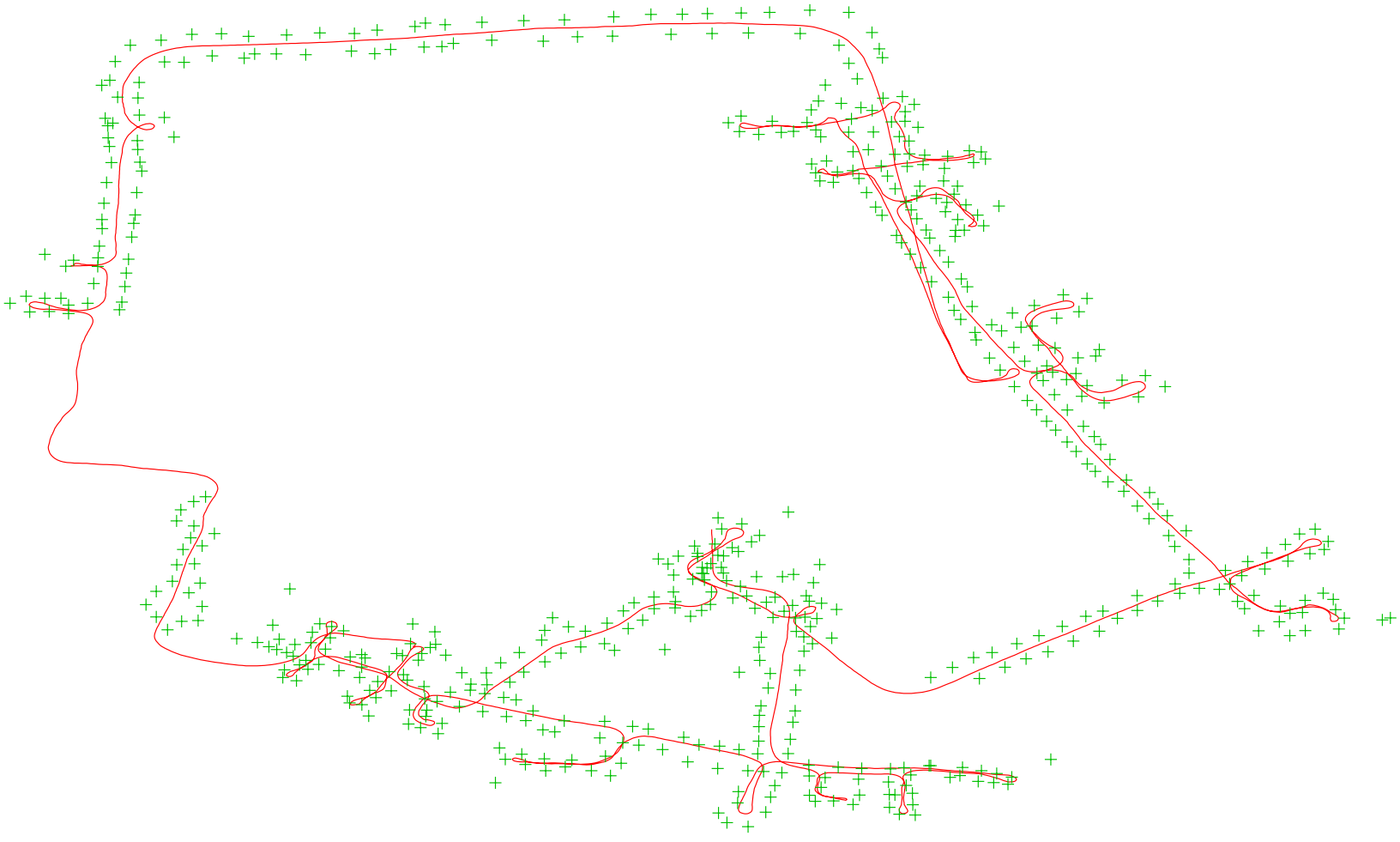
Möglichkeiten des Frameworks

- Mit sehr wenig Aufwand klassische SLAM-Probleme lösbar
- Mit geringem Mehraufwand Kalibrierungsprobleme lösbar
- Es ist möglich einzelne Variablen nicht zu optimieren (z. B. sinnvoll um andere Algorithmen zu analysieren)
- Code veröffentlicht auf `http://openslam.org/slom.html`

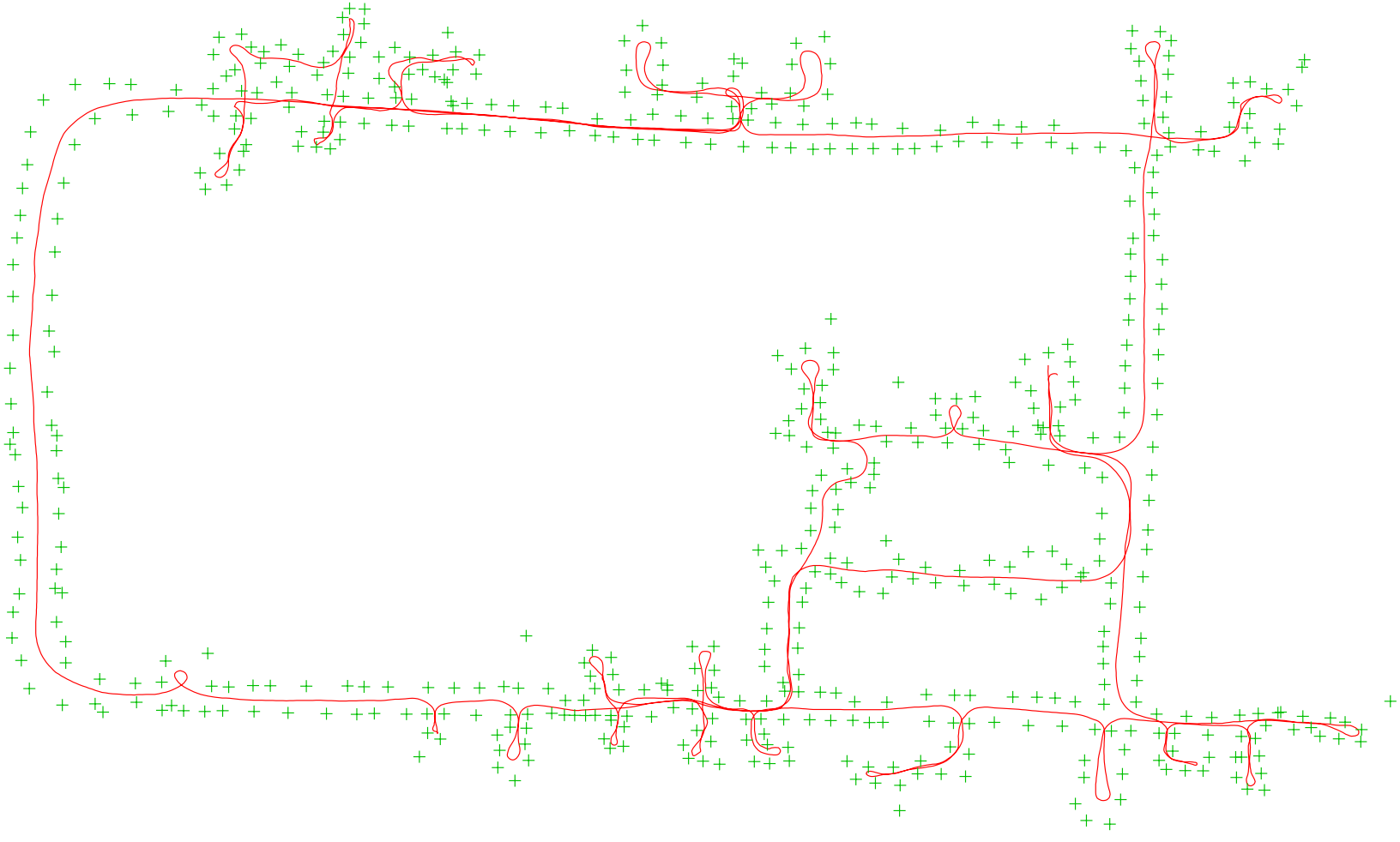
Performanz

- Framework findet in erstaunlich kurzer Zeit optimale Lösung
- Teilweise schneller als auf SLAM-Probleme eingeschränkte Frameworks
  - Für ca 3300 Poses, 580 Landmarken und 14000 Landmarkenmessungen 2.5s bei 1.86GHz.

DLR-Datensatz, Rohwerte

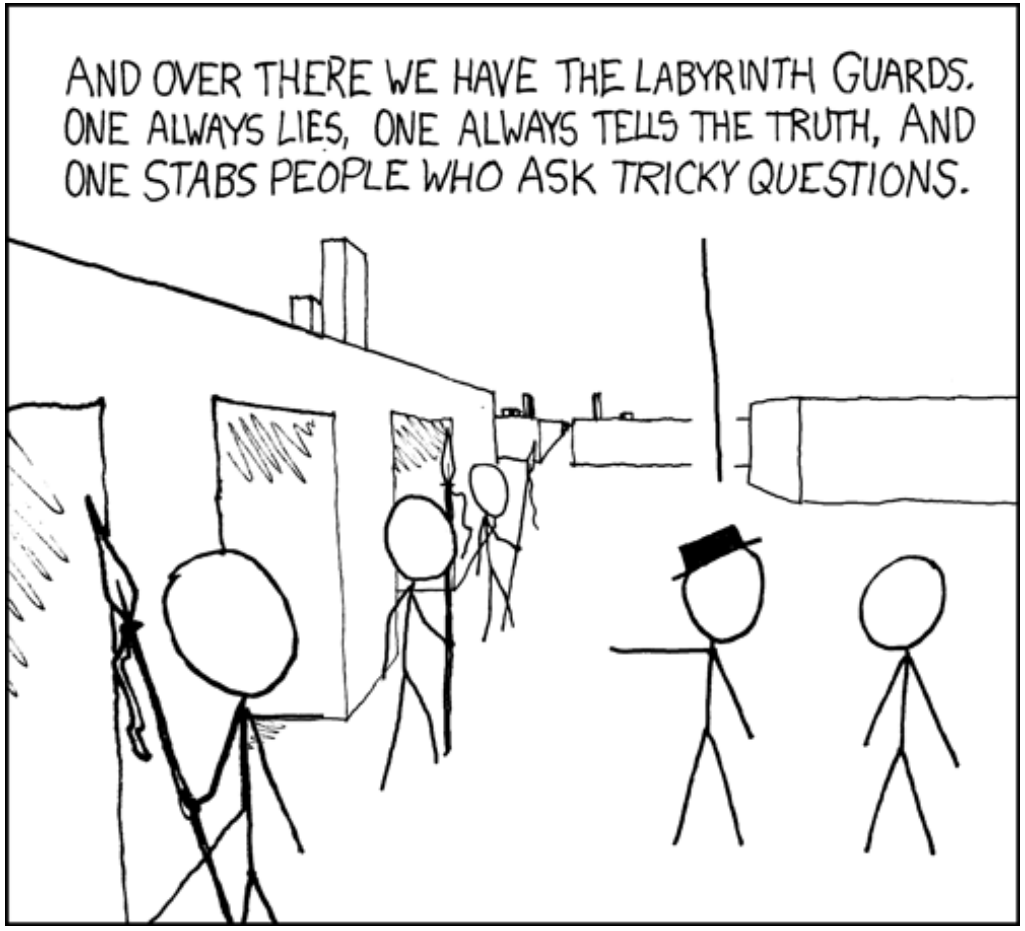


DLR-Datensatz, Optimiert



Danke für die Aufmerksamkeit

Fragen?



Quelle: `http://xkcd.com/246/`