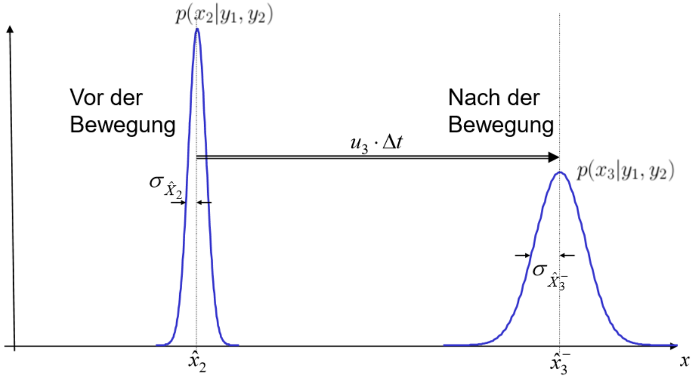
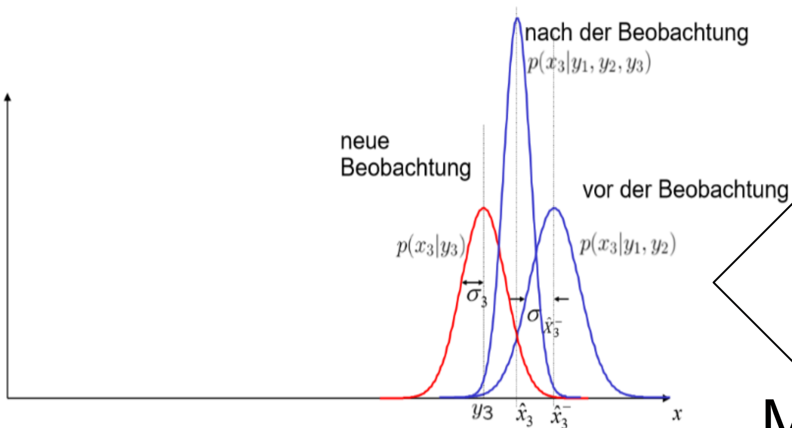
Roboterlokalisierung in 2D mittels SE(2) Filterung

Martin Miller, He Jingyu

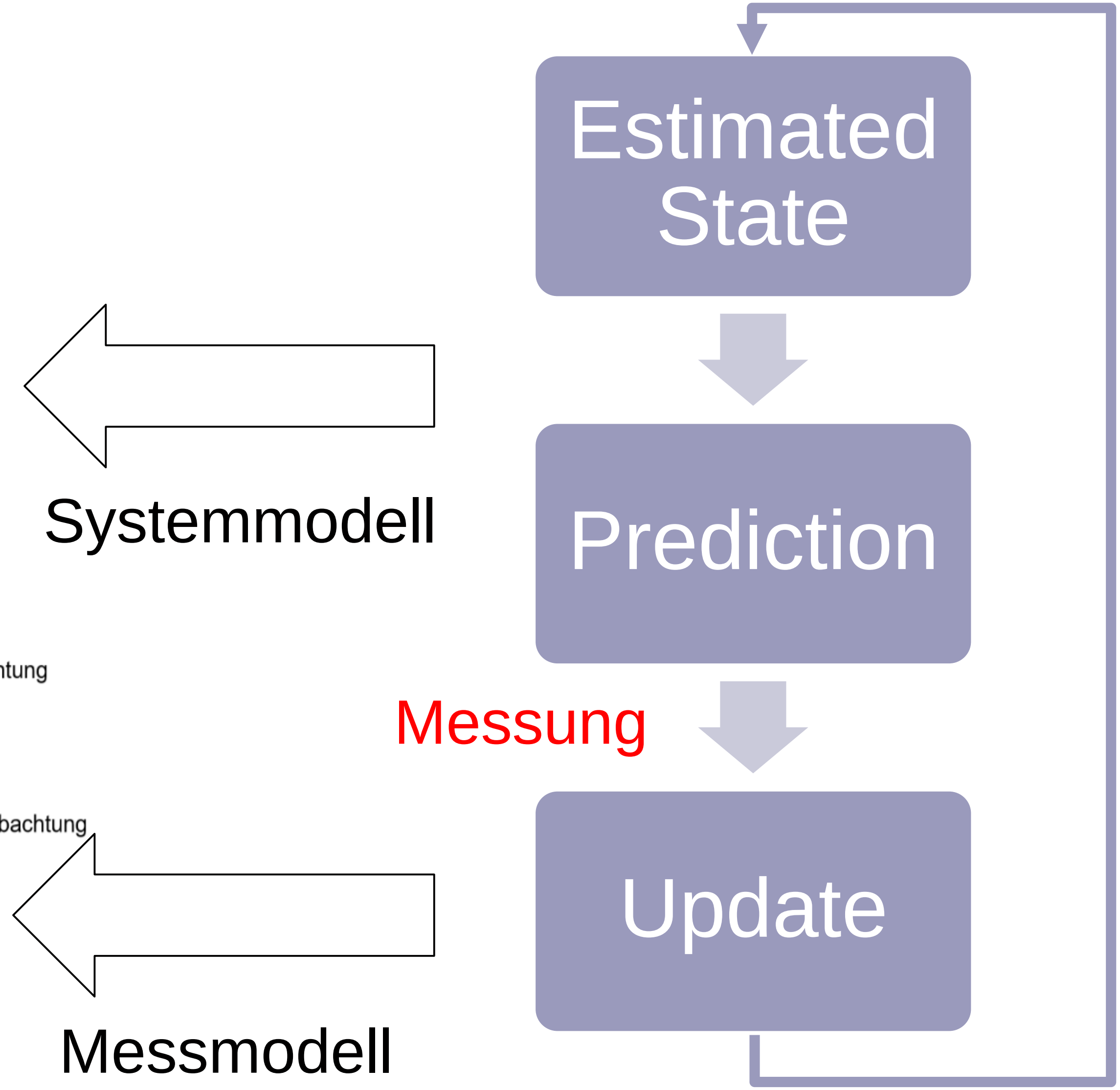
Kalman Filter



Propagation der Verteilung im Prädiktionsschritt



Verbesserung der Verteilung im Innovationsschritt

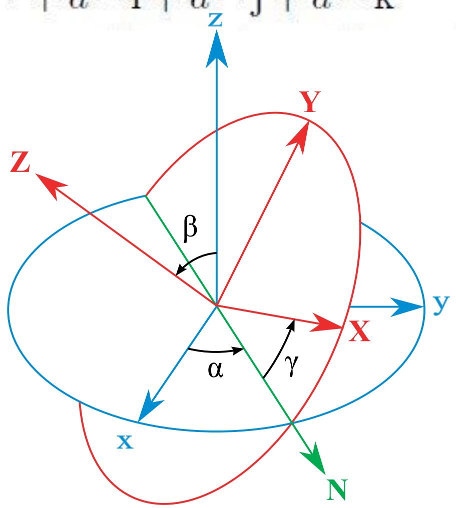


Duale Quaternionen

Einfache Quaternionen:
- 4 Komponenten
- Rotationen als Quaternion
- Anwendung auf Vector v:

qvq^*

$$a^{(1)} + a^{(2)}\mathrm{i} + a^{(3)}\mathrm{j} + a^{(4)}\mathrm{k}$$



Duale Quaternionen:
- 8 Komponenten
- Rotationen mit Translation

p + ε q

$$a^{(1)} + a^{(2)}\mathrm{i} + a^{(3)}\mathrm{j} + a^{(4)}\mathrm{k} \\ + \varepsilon\left(a^{(5)} + a^{(6)}\mathrm{i} + a^{(7)}\mathrm{j} + a^{(8)}\mathrm{k}\right)$$

$$\boxed{\left[1 + \varepsilon\frac{1}{2}(t_x\mathrm{i} + t_y\mathrm{j})\right]} \cdot \boxed{\left[\cos\left(\frac{\alpha}{2}\right) + \sin\left(\frac{\alpha}{2}\right)\mathrm{k}\right]}$$

$$= \cos\left(\frac{\alpha}{2}\right) + \sin\left(\frac{\alpha}{2}\right)\mathrm{k} \\ + \frac{\varepsilon}{2}\left[\left(\cos\left(\frac{\alpha}{2}\right)t_x + \sin\left(\frac{\alpha}{2}\right)t_y\right)\mathrm{i} \right. \\ \left. + \left(\cos\left(\frac{\alpha}{2}\right)t_y - \sin\left(\frac{\alpha}{2}\right)t_x\right)\mathrm{j}\right] .$$

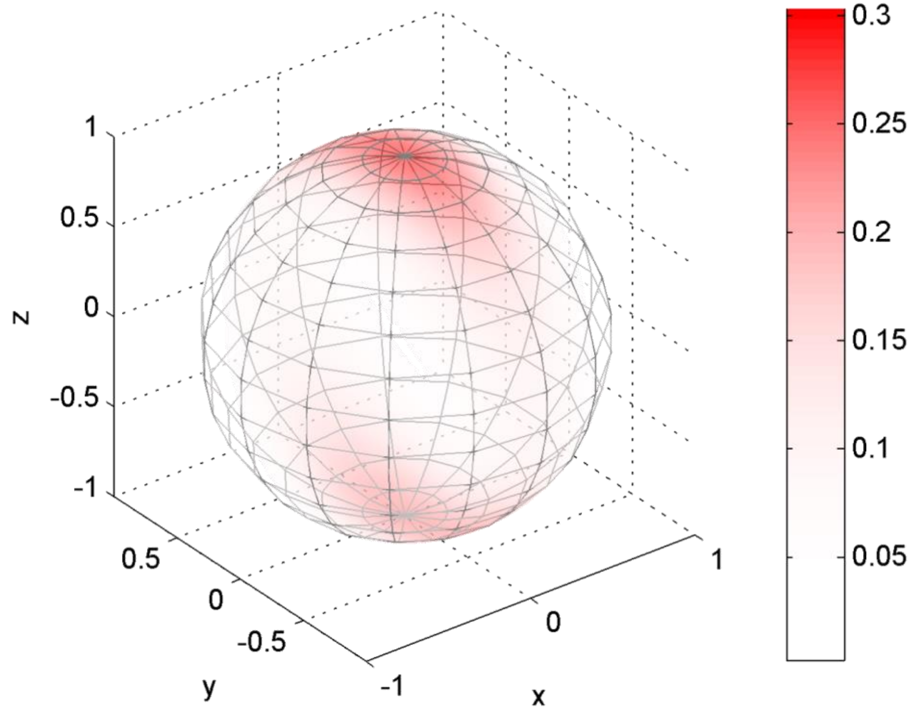
SE2-Filter

Parameter Estimation

Deterministic Sampling

Prediction

Update



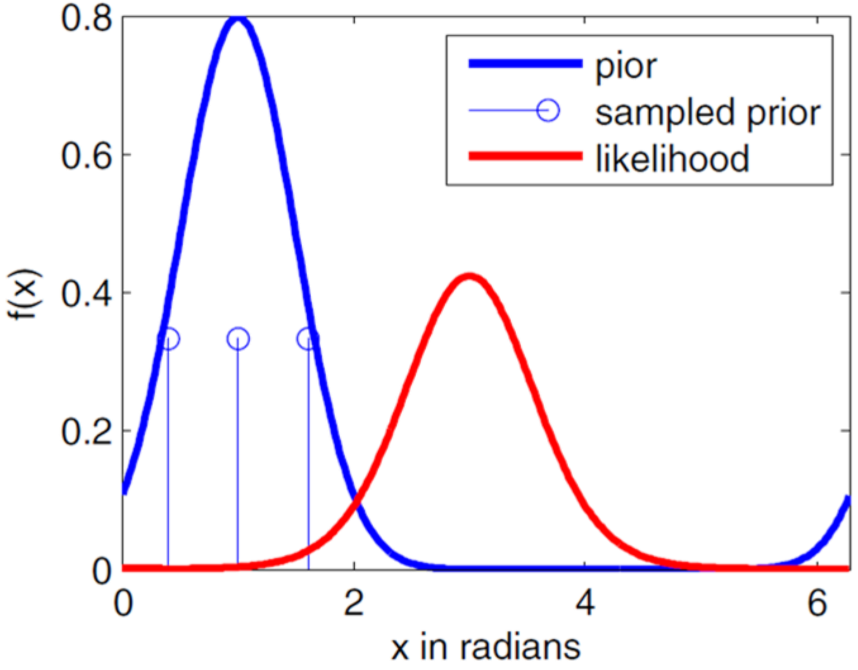
SE2-Filter

Parameter Estimation

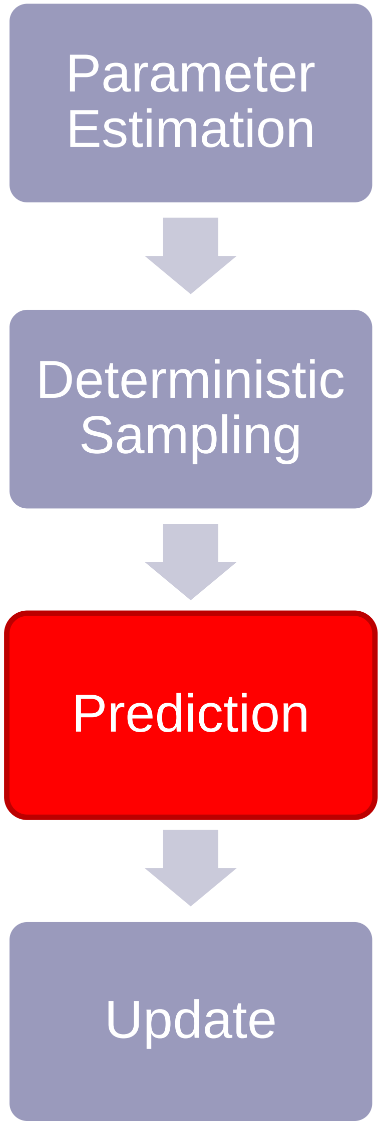
Deterministic Sampling

Prediction

Update



SE2-Filter



- Anwendung des Systemmodells auf Samples
- Bestimme Parameter des Systemzustands

SE2-Filter

Parameter Estimation

↓

Deterministic Sampling

↓

Prediction

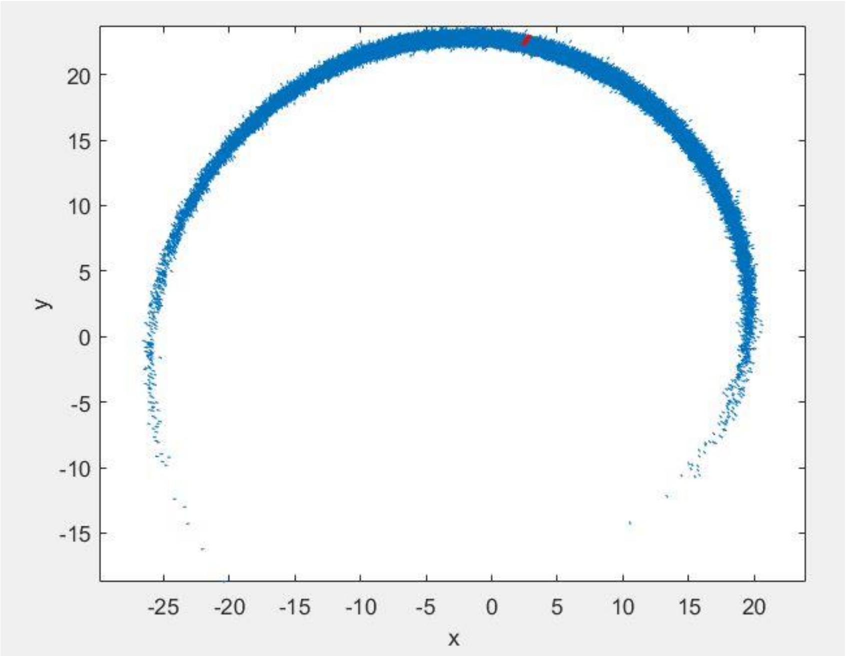
↓

Update

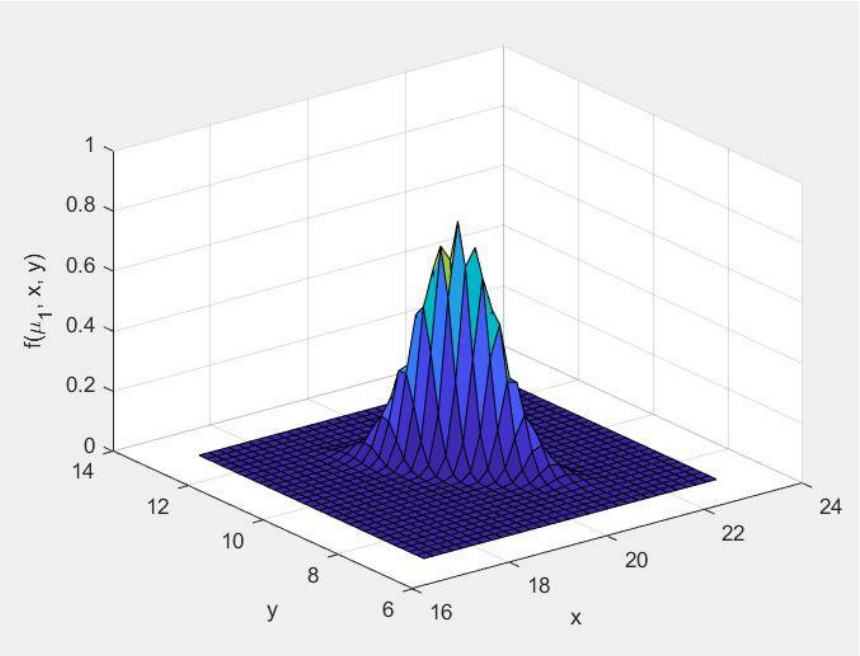
Messung

Fusionierung mit Messdaten

Einmalige Ergebnis der SE2Filter

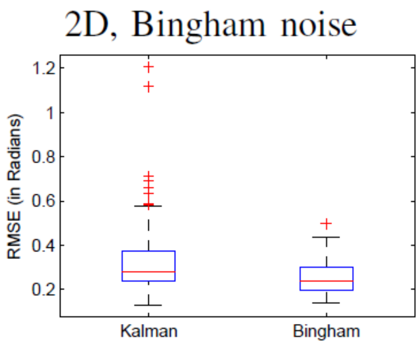


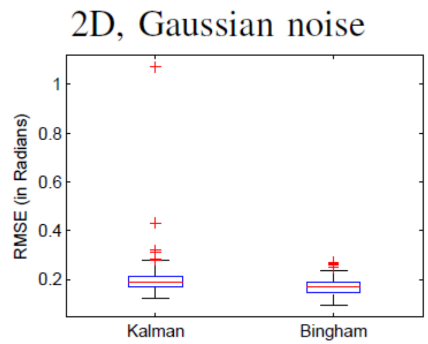
Rotation

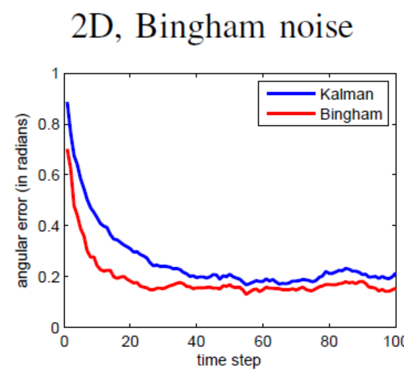


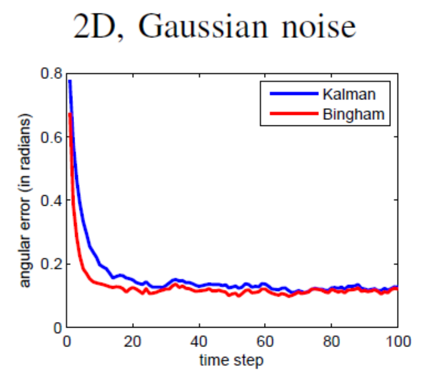
Translation

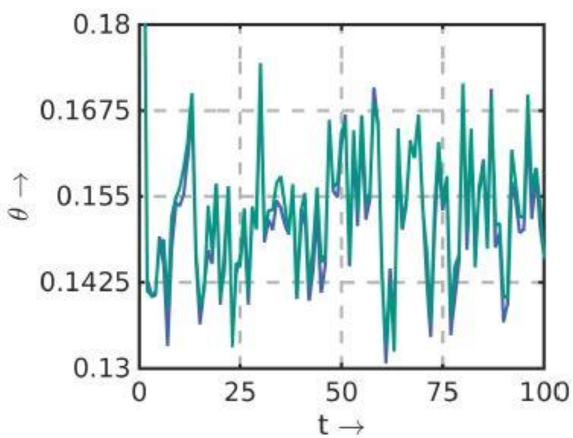
Vergleich SE2 und Kalman von vorheriger Versuch



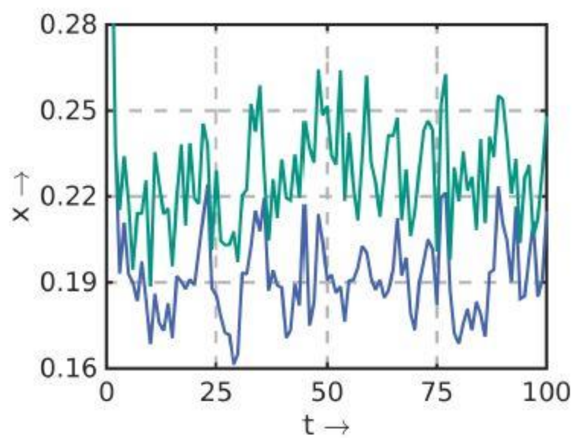




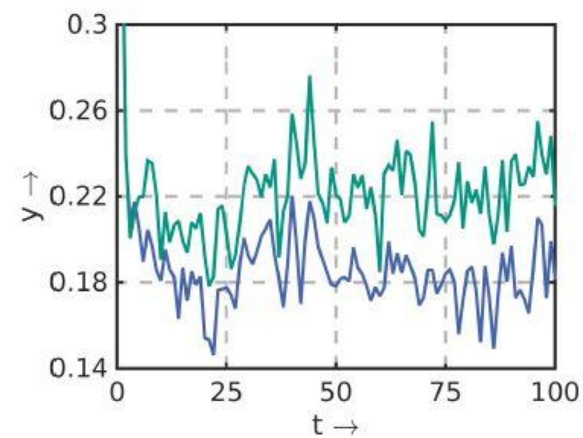




(a) Orientation



(b) x-Position



(c) y-Position

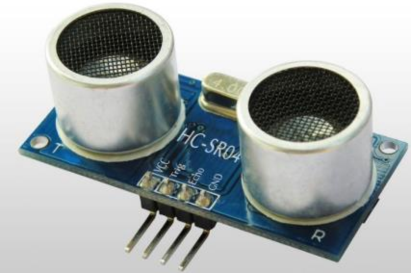
Crawler Framework

Crawler

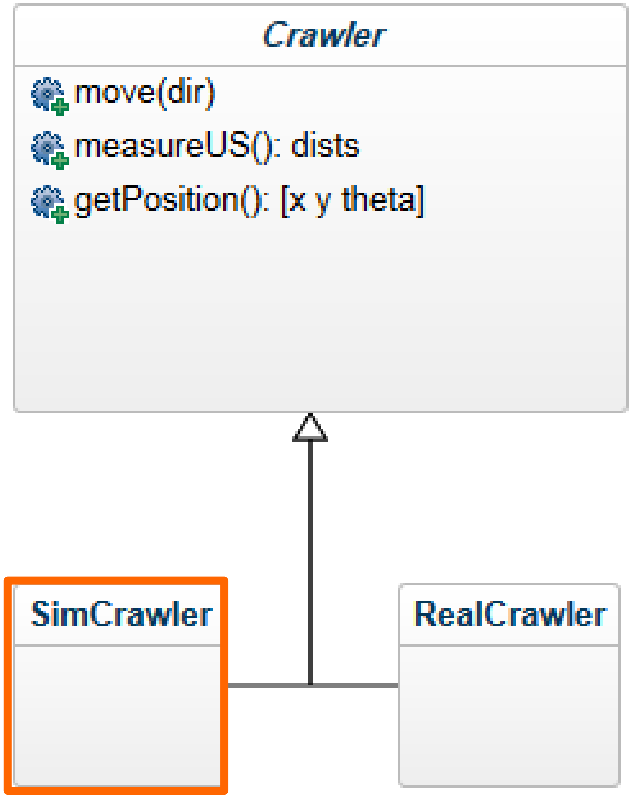
- Ultraschall-Sensor
- Aktuatorik

Matlab

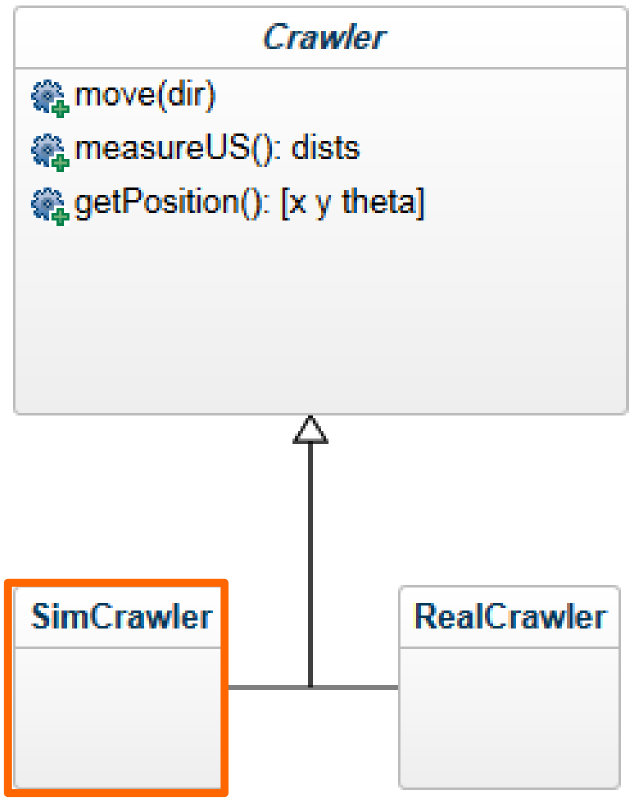
- RealCrawler
- Partikelfilter
- SE2-Filter



Simulations-Framework



Simulations-Framework

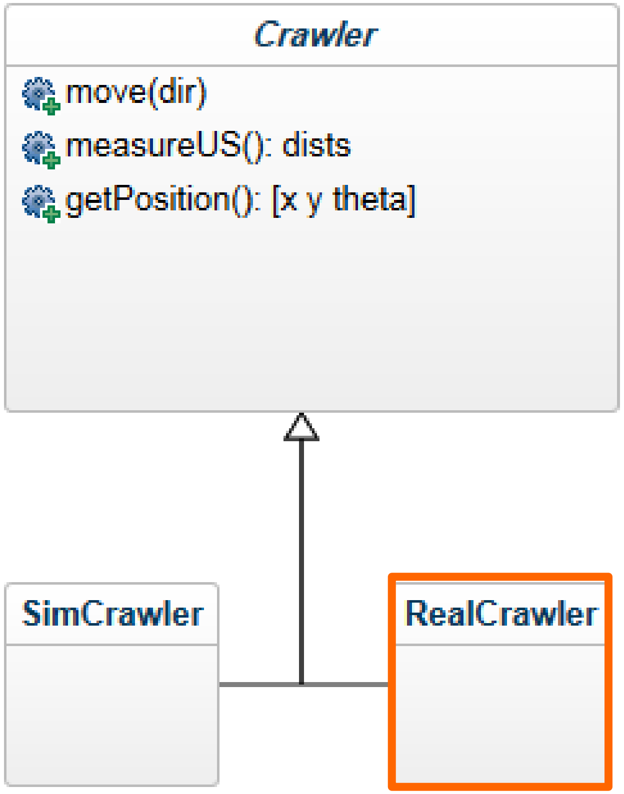


SimCrawler

```
function move(obj, dir)
   obj.position = obj.position + STEP + NOISE;
end


function dists = measureUS(obj)
   intersects = obj.map.getIntersections();
   dists = vecnorm(inter - pos);
end
```

Simulations-Framework



RealCrawler

```
function move(obj, dir)
   tcpMove(dir, 100, 1);
end


function dists = measureUS(obj)
   for i = 1:4
       dists(i)= tcpUS(i, 1);
   end
end

function [x y theta] = getPosition(obj)
   img = getCameraImage();
   [red yellow] = findLEDs(img);
   [x y theta] = estimatePosition(red, yellow);
end
```

Weiteres Vorgehen

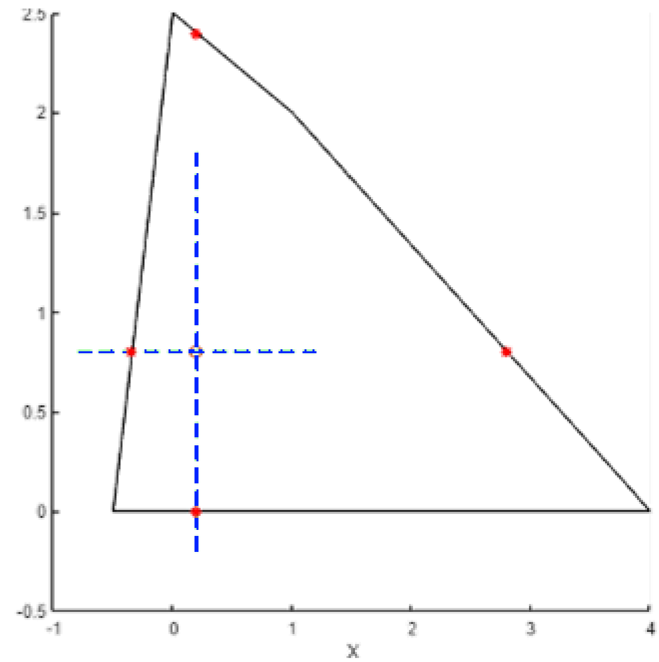
SE(2)-Filter ⟷ Ground Truth (x, y, **φ**) ⟷ Vergleichsalgorithmus

- Partikelfilter
- Kalman-Filter

Schwierigkeiten

Positionsbestimmung
anhand der Ultraschall-Abstände:

USdists → ? → [x y theta]



- Erster Ansatz:
  - Löse Optimierungsproblem
  - **Problem**: Uneindeutigkeit
  - + lokale Minima
  - => Asymmetrische Karte + Startwert
- Progressive Updates

Vielen Dank für Ihre Aufmerksamkeit