TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik | Institut für Software- und Multimediatechnik | Lehrstuhl für Softwaretechnologie

Model-driven Multi-Quality Auto-Tuning of Robotic Applications

MORSE 2015

Christian Piechnick, Sebastian Götz,
René Schöne and **Uwe Aßmann**

Technische Universität Dresden
Software Engineering Group

Frank Bahrmann and
Hans-Joachim Böhme

HTW Dresden
Artificial Intelligence Group

HAEC

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Outline

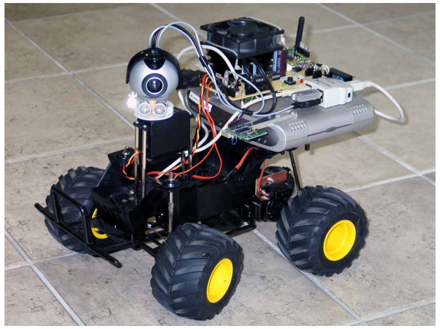
1. Motivation and Background
2. MQuAT for Simultanous Localization and Mapping (SLAM)
3. Evaluation of SLAM as a Service
4. Summary and Future Work

Model-driven Multi-Quality Auto-Tuning of Robotic Applications

MOTIVATION AND BACKGROUND

Simultaneous Localization and Mapping (SLAM)

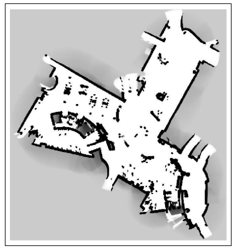

2D Laser Scanner


RGB Camera


Stereo Camera + Ultra-Sonic Sensor

Many mobile Robots must operate in varying or unkown environments.

- **No static map feasible** (changing layouts or unkown enviroments)
- **Dynamic creation of a map**
- **Dynamic localization within the dynamically created map**



SLAM algorithms create a map by interpreting sensor data and localize the position of the corresponding entity simultanously.

SLAM Variability

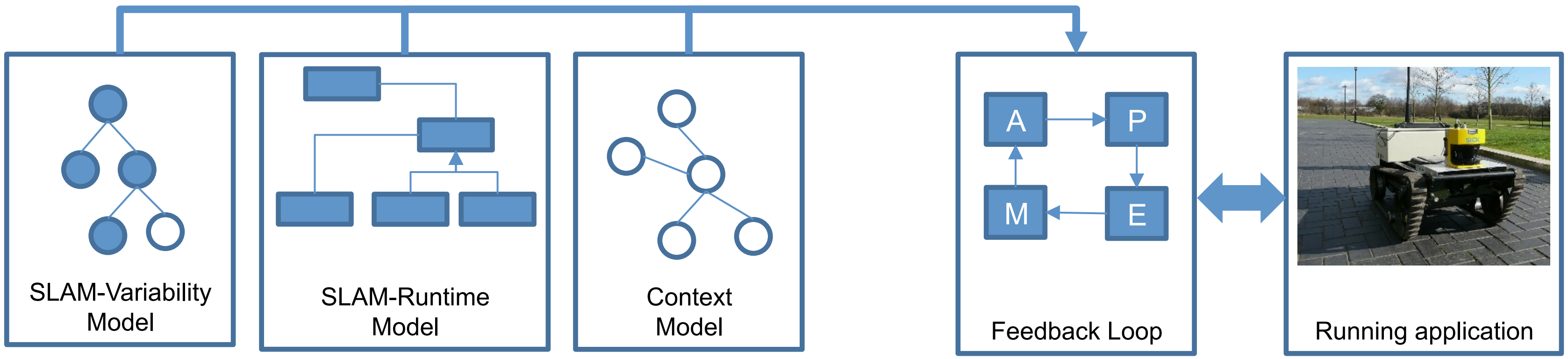
- **60+ different implementations** found in a online search
- **Different requirements w.r.t.**
  - Resource consumption (e.g., cpu, main memory)
  - Performance
  - Precision of the algorithm
  - Context dependencies (e.g., outdoor, indoor, available hardware etc.)
  - Software platform (e.g., programming language, robotic framework etc.)
- **Very poor reuse**
  - No standardization of the used data types (e.g., grid maps, feature maps, laser scanner data etc.)
  - No modularization
  - Complete re-implementation on changed requirements
- **Requirements may change during runtime**
  - Runtime adaptivity needed

Strategic Goal 1: Modularization of SLAM to increase reuse

Strategic Goal 2: Self-Adaptive SLAM for enhancing robotic applications

What we need

- PIM for SLAM process
- PIM for data-representations
- PSM for SLAM modules (with requirements and NFPs)
- Models for variability



Framework GeneralRobot

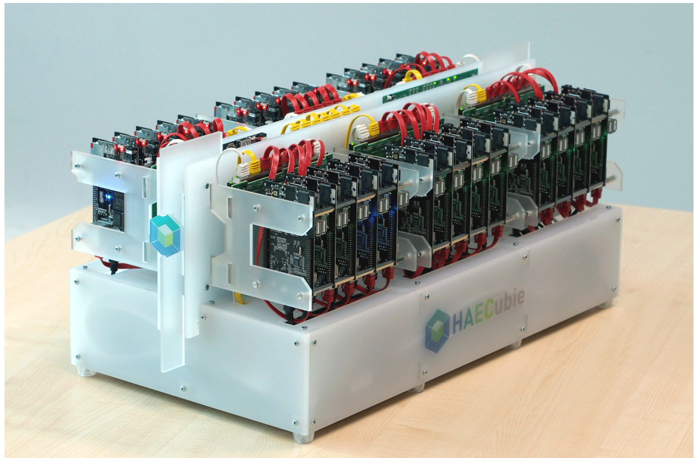
- Component-based Middleware for Robotic Applications
- Modules for map creation, localization, navigation etc.
- Static variability for SLAM (configuration file)
- High-level modules (i.e., non-hierarchical components)
  - Variabilty managed manually within Java-Code
  - Scattering and Tangling of variability management code
  - No focus on maintainability and reusability

Stable running robotic applications

- „August der Smarte“ – Tour Guide Robot in the museum „Technische Sammlungen Dresden“
- AAL Robot in a elderly care institution in Dresden

CRC 912 - Highly Adaptive Energy-Efficient Computing

- New hardware- and software-architectures for **energy proportional solutions**
- Domain: Server Applications
- HAEC Box as prototypical hardware platform
  - Cluster of *Cubieboards* as single-board computers
  - Boards can be switched-off on demand to reduce energy consumption



Multi-Quality Auto-Tuning (MQuAT) for the runtime optimization of software architectures

Collaboration

Software Technology Group

- MDSD Experts
- Abstract SLAM process
- SLAM variability models

HAEC

- Model-Driven Optimization
- Benchmarking Framework
- Framework for Feedback Loops

Hochschule für Technik und Wirtschaft Dresden
University of Applied Sciences

- Robotic Experts
- SLAM implementation artifacts
- Simulation environment

Model-driven Multi-Quality Auto-Tuning of Robotic Applications

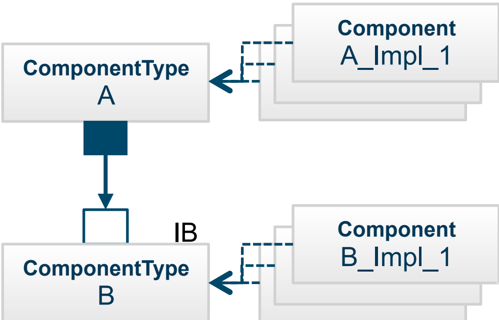
TECHNISCHE UNIVERSITÄT DRESDEN

Model-driven Multi-Quality Auto-Tuning of Robotic Applications

MQUAT FOR SIMULTANOUS LOCALIZATION AND MAPPING (MQUAT-SLAM)

Multi-Quality Auto-Tuning (MQuAT)

- **Structural Model:** SW/HW Description Language for architectures
  - Each component type can have **multiple implementations** (SW variation points)
- **Variant Model:** State of HW/SW components (e.g., current SW architecture, CPU load etc.)
- **Non-functional properties** of provided/required ports described with contracts (QCL)
- Component-stub code + ILP generation
- Benchmarking framework + THEATRE runtime environment (implementation of feedback loop)




```
contract B for port type IB {
    requires resource CPU {
      min frequency: 2 GHz
    }
    requires resource Net {
      min bandwidth: 10 MBit/s
    }

    provides min resolution: 1 ppmm
    provides min responseTime: 2ms
}
```

Current State of SLAM algorithms

- Almost no reuse of SLAM code (Re-Implementation for varying requirements)
- Almost no reuse in adaptivity-handling code (Re-Implementation for each solution)
- Variability handling within business logic

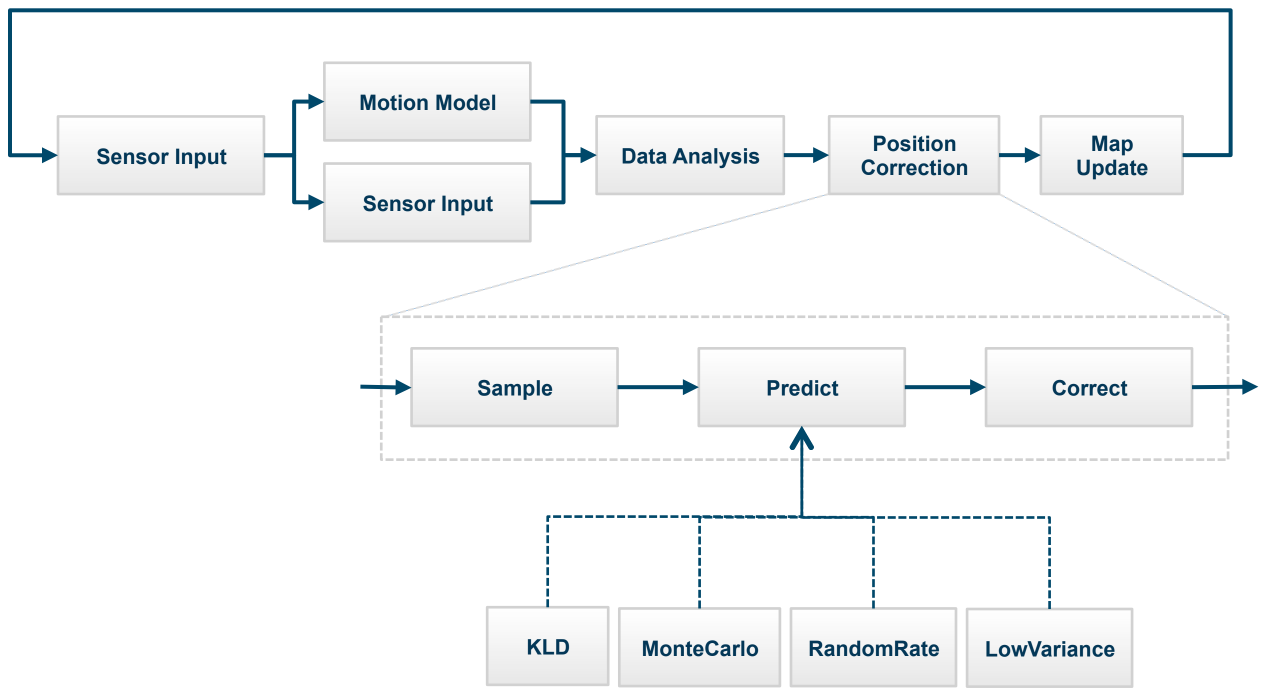
Desired State

- SLAM-Framework with all alternative implementation variants
- Automatic generation of adaptivity-handling code
- External feedback loop to resolve scattering and tangling
- Change of objective function changes energy consumption, performance, and precision

Contribution

- MQuAT for SLAM process (SLAM modularization, Code generation, ILP generation, Feedback Loop)
- Optimizer follows changes of objective function
- Case study to show feasability

SLAM Process Model

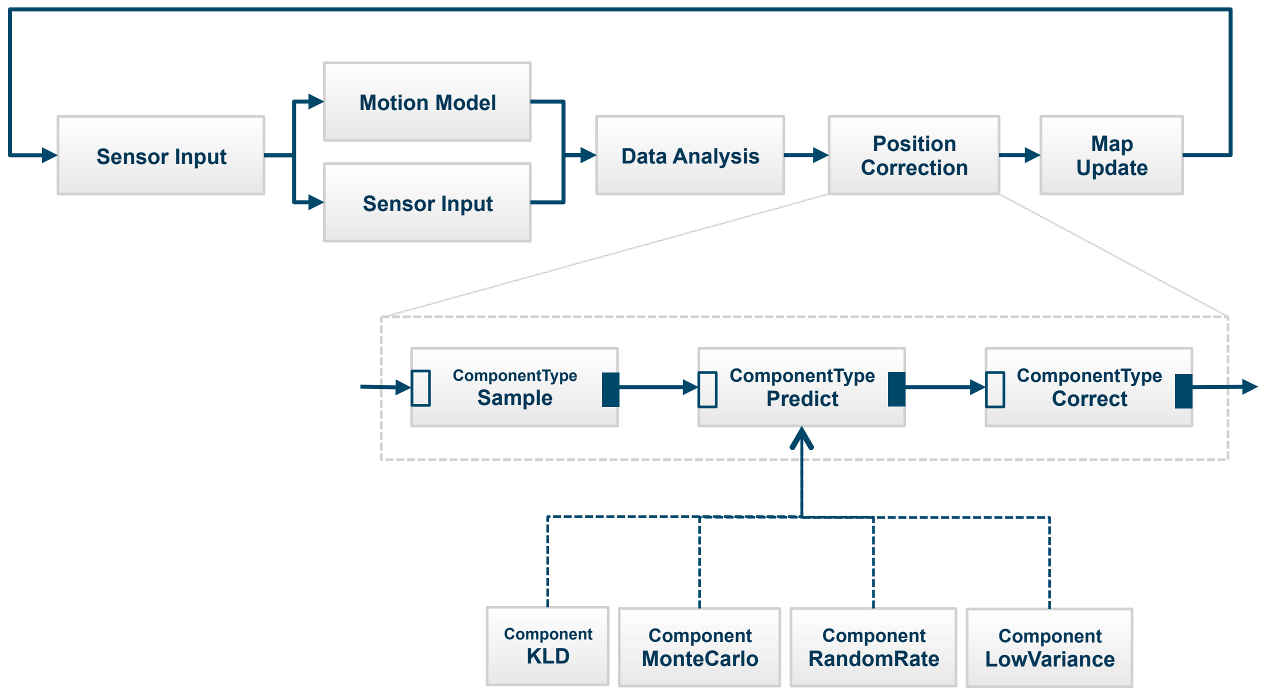


Abstract Process

Variant with Particle Filtering

Alternative algorithms for prediction

MQuAT Modeling of the SLAM Process

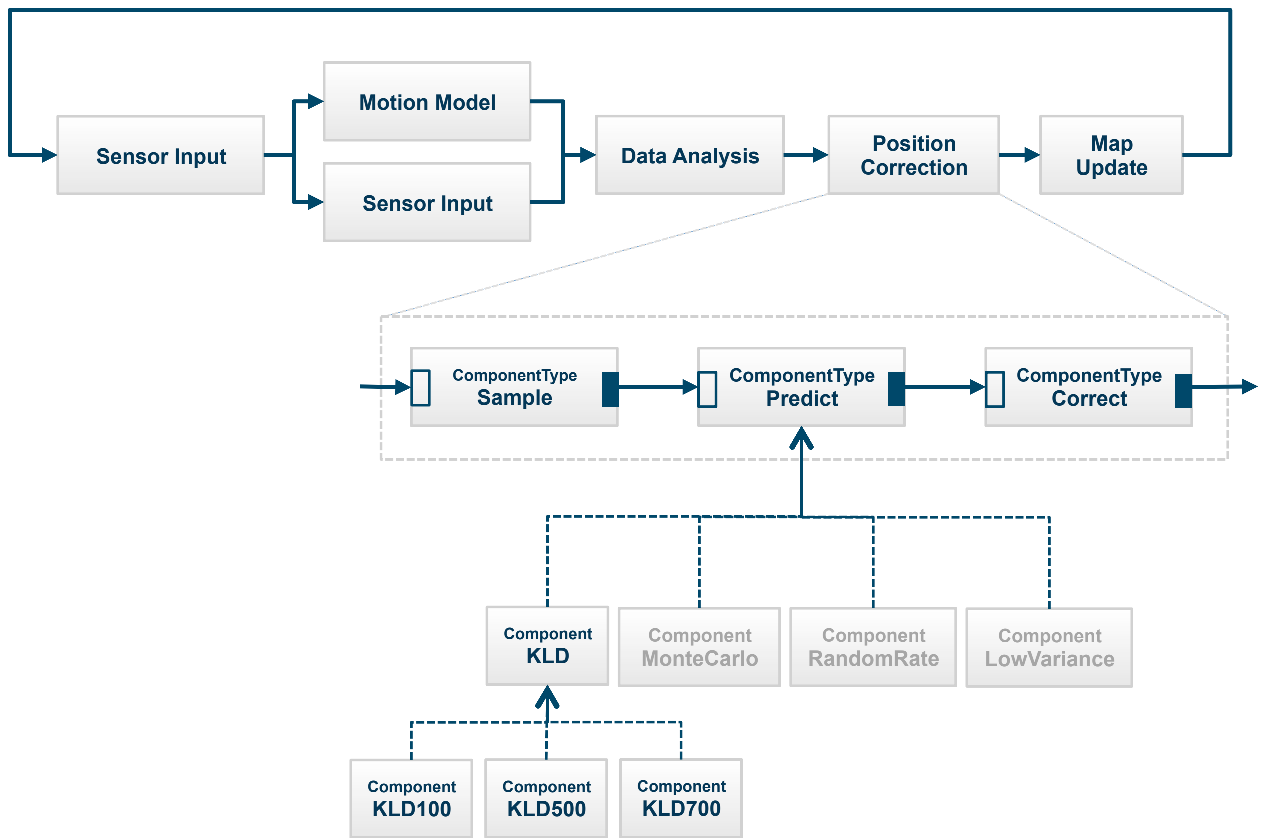


Abstract Process

Variant with Particle Filtering

Alternative algorithms for prediction

MQuAT Modeling of the SLAM Process



Abstract Process

Variant with Particle Filtering

Alternative algorithms for prediction

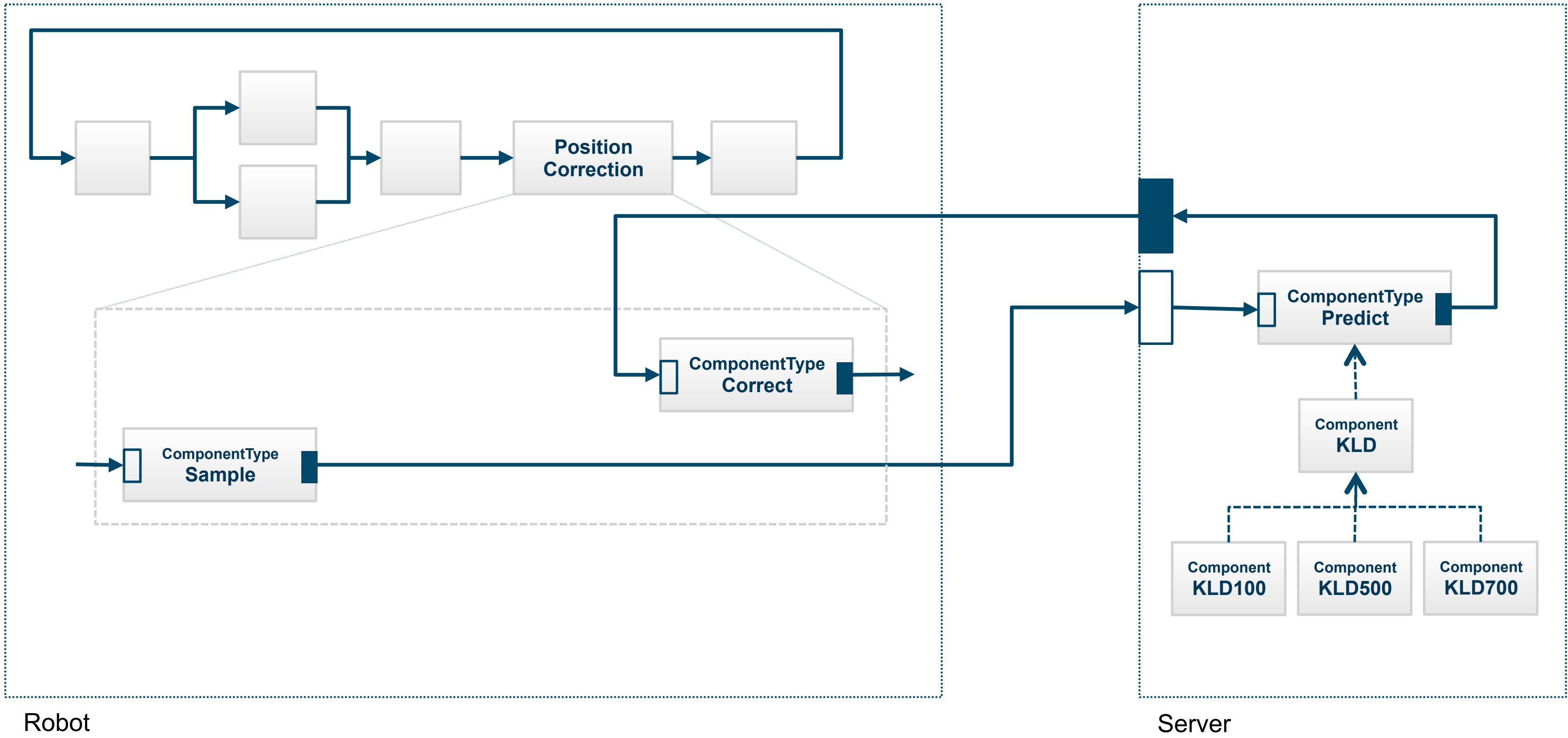
Variants by parameterization
(e.g., number of particles)

SLAM as a Service

- Battery is a very limited resource in mobile robotic systems
  - Predicition of particles is a computation intensive task

    → Prediction consumes much energy
- Outsourcing of the prediction logic
- **Hosting the prediction calculation on a server as a service**

SLAM as a Service

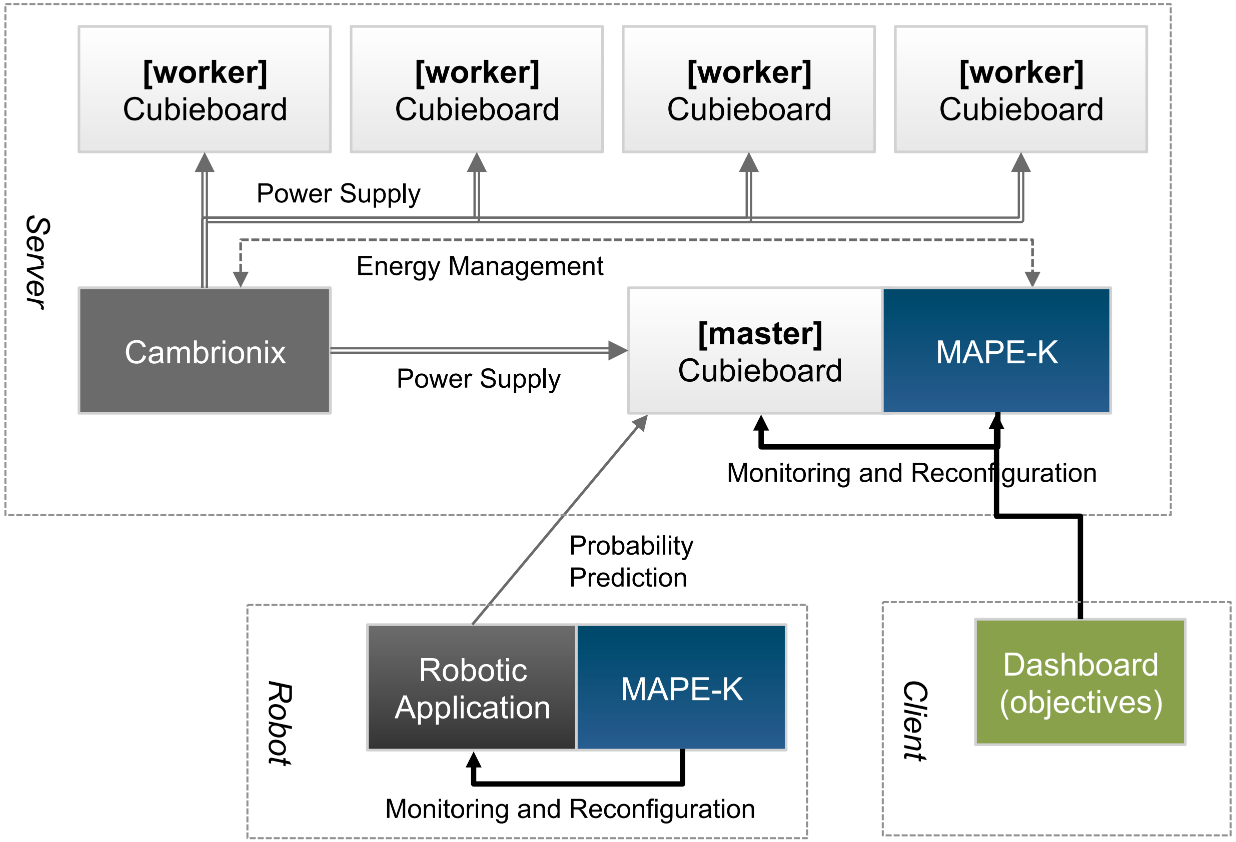


Model-driven Multi-Quality Auto-Tuning of Robotic Applications

EVALUATION

SLAM PARTICLE PREDICTION AS A SERVICE

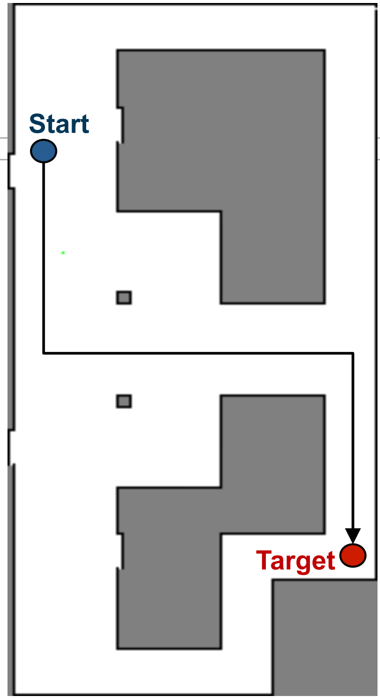
Hardware setup



Cubieboard

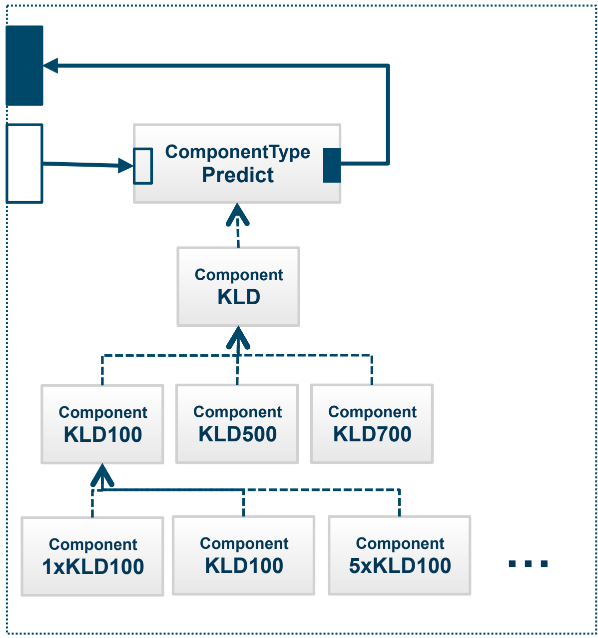
Cambrionix USB Power Supply

SLAM as a Service with Parallelization



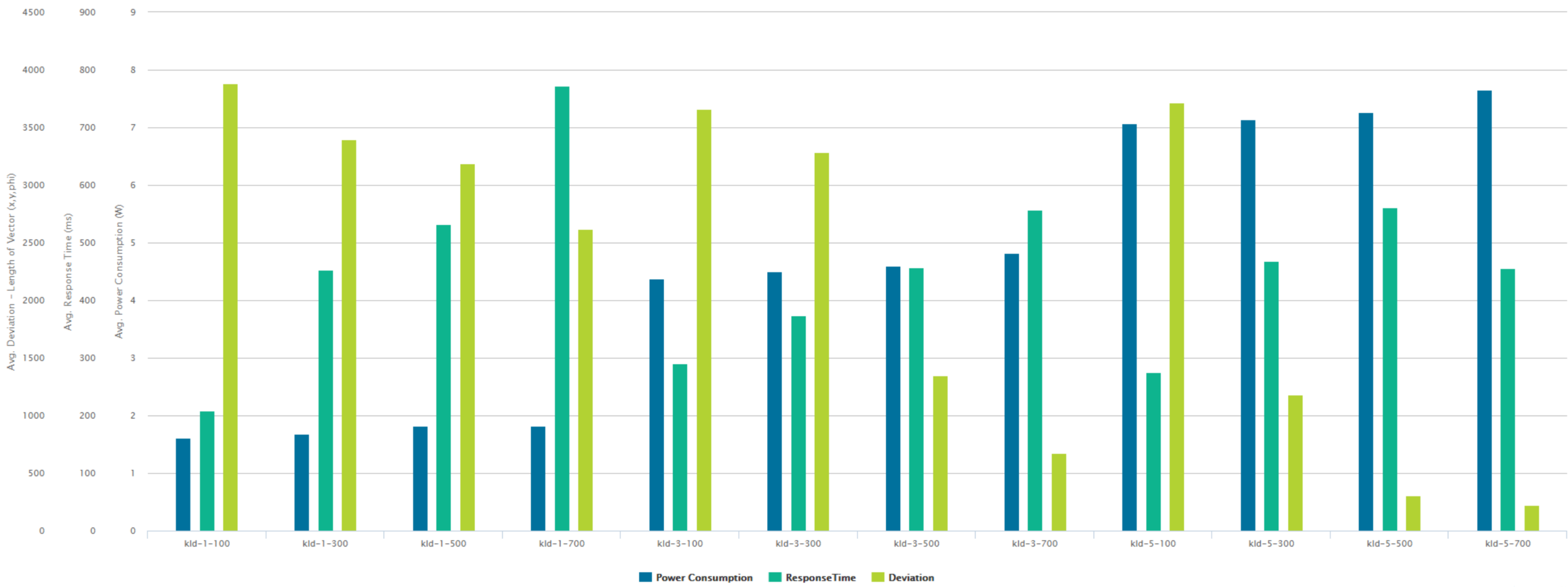
- Robot driving from the start to the target position
- **Simbad** simulation environment
- **GeneralRobot** target framework
- **MQuAT SLAM optimizer**
  - Prediction is done for each particle in isolation → Can calculated in parallel
  - 1-5 boards with 2 cores, max. 10 parallel threads
  - Kullback-Leibler Divergence with n*100 particles
- For each variant, measure:
  - **PC:** Server power consumption in ms
  - **T:** Response time of the service in Watt
  - **D:** Deviation between real and estimated position as length of the vector (Δx; Δy; ΔΦ)

    x,y = Position, Φ = rotation



Server

Result



Avg. Deviation - Length of Vector (x,y,phi)
Avg. Response Time (ms)
Avg. Power Consumption (W)
4500 4000 3500 3000 2500 2000 1500 1000 500 0
900 800 700 600 500 400 300 200 100 0
9 8 7 6 5 4 3 2 1 0
kld-1-100
kld-1-300
kld-1-500
kld-1-700
kld-3-100
kld-3-300
kld-3-500
kld-3-700
kld-5-100
kld-5-300
kld-5-500
kld-5-700
Power Consumption
ResponseTime
Deviation

Result

particles	1 cubie 100	1 cubie 500	1 cubie 700	3 cubies 100	3 cubies 500	3 cubies 700	7 cubies 100	7 cubies 300	7 cubies 700
T (ms)	208.6	532.2	772.7	290.2	457.2	557.4	275.3	361.6	456.2
PC (W)	1.6	1.8	1.8	4.4	4.6	4.8	7.1	7.3	7.7
D	338.4	318.7	261.9	365.9	134.6	67.6	371.3	30.6	22.4

particles	1 cubie 100	1 cubie 500	1 cubie 700	3 cubies 100	3 cubies 500	3 cubies 700	7 cubies 100	7 cubies 300	7 cubies 700
T (ms)	**208.6**	**532.2**	**772.7**	**290.2**	**457.2**	**557.4**	**275.3**	**361.6**	**456.2**
PC (W)	1.6	1.8	1.8	4.4	4.6	4.8	7.1	7.3	7.7
D	338.4	318.7	261.9	365.9	134.6	67.6	371.3	30.6	22.4

- **Response time depends on both parameters**

  → More boards = lower response time

  → More particles = higher response time

particles	100	500	700	100	500	700	100	300	700
	1 cubie			3 cubies			7 cubies		
T (ms)	208.6	532.2	772.7	290.2	457.2	557.4	275.3	361.6	456.2
PC (W)	**1.6**	**1.8**	**1.8**	**4.4**	**4.6**	**4.8**	**7.1**	**7.3**	**7.7**
D	338.4	318.7	261.9	365.9	134.6	67.6	371.3	30.6	22.4

- **Power consumption** mainly depends on number of boards

  → More boards = higher power consumption

  → More particles = slightly higher power consumption

	1 cubie			3 cubies			7 cubies		
particles	**100**	**500**	**700**	**100**	**500**	**700**	**100**	**300**	**700**
T (ms)	208.6	532.2	772.7	290.2	457.2	557.4	275.3	361.6	456.2
PC (W)	1.6	1.8	1.8	4.4	4.6	4.8	7.1	7.3	7.7
D	**338.4**	**318.7**	**261.9**	**365.9**	**134.6**	**67.6**	**371.3**	**30.6**	**22.4**

- **Deviation** depends on both parameters

  → More boards = lower deviation

  → More particles = lower deviation

Result

	1 cubie			3 cubies			7 cubies		
particles	**100**	**500**	**700**	**100**	**500**	**700**	**100**	**300**	**700**
T (ms)	**208.6**	532.2	772.7	290.2	457.2	557.4	275.3	361.6	**456.2**
PC (W)	**1.6**	1.8	1.8	4.4	4.6	4.8	7.1	7.3	**7.7**
D	**338.4**	318.7	261.9	365.9	134.6	67.6	371.3	30.6	**22.4**

- **Real trade-off between response time, power consumption and deviation**
- Lower response time leads to high deviation
- Lower power consumption leads to high deviation
- Lower deviation leads to:

    higher power consumption (with low response time)

    higher response time (with low power consumption)

Objective function of ILP important

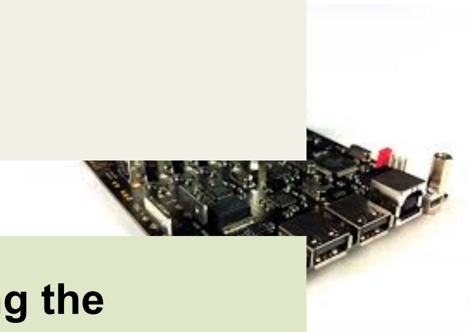
MQuAT Optimizer Follows Change of Objective Function of ILP

[worker] [worker] [worker] [worker]

Server

- **Real trade-off between response time, power consumption and deviation**
- Lower response time leads to high deviation
- Lower power consumption leads to high deviation
- Lower deviation leads to:
    - higher power consumption (with low response time)
    - higher response time (with low power consumption)

Probability Prediction



er Supply

- **MQuAT Optimizer dynamically adapts SLAM by following the changes of Objective Functions**

TECHNISCHE UNIVERSITÄT DRESDEN

Model-driven Multi-Quality Auto-Tuning of Robotic Applications

CONCLUSION AND FUTURE WORK

Conclusion

- SLAM has a high degree of variation based on varying requirements (also @run.time)
- **State**: Poor reuse of SLAM-code and adaptation logic
- **Assumption**: Component Modeling + Code Generation decreases development time and increases maintainability
- MQuAT for runtime optimization of architectures with Quality Contracts
  - Applicable for SLAM processes

- Benchmarks show that trade-offs exist **(only for one small step within a complex process)**
- **Energy-consumption can be decreased, when lower response time or lower quality is accapable**
- **MQuAT optimizer follows changes of objectives**

Future Work

- Include benchmarks of the other variants of the prediction algorithm
- Model and migrate existing implementations for whole SLAM process
- Develop SLAM-Toolbox for static and dynamic variant generation
- Integration in standard-platforms (e.g., ROS)

Model-driven Multi-Quality Auto-Tuning of Robotic Applications

MORSE 2015

Christian Piechnick, Sebastian Götz,
René Schöne and **Uwe Aßmann**

Technische Universität Dresden
Software Engineering Group

Frank Bahrmann and
Hans-Joachim Böhme

HTW Dresden
Artificial Intelligence Group