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
1. Motivation and Background
2. MQuAT for Simultaneous 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)

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

Many mobile Robots must operate in varying or unkown environments.

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

2D Laser Scanner  RGB Camera  Stereo Camera + Ultra-Sonic Sensor
- **60+ different implementations** found in an 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
Model-driven Multi-Quality Auto-Tuning of Robotic Applications

- MDSD Experts
- Abstract SLAM process
- SLAM variability models

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

- Robotic Experts
- SLAM implementation artifacts
- Simulation environment

Collaboration
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)

```plaintext
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 ppm
  provides min responseTime: 2 ms
}
```
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 feasibility
Model-driven Multi-Quality Auto-Tuning of Robotic Applications

**SLAM Process Model**

**Abstract Process**
- Variant with Particle Filtering

**Alternative algorithms for prediction**
- KLD
- MonteCarlo
- RandomRate
- LowVariance

---

**Motion Model**
- Sensor Input
- Motion Model
- Sensor Input
- Data Analysis
- Position Correction
- Map Update
- Sample
- Predict
- Correct
MQuAT Modeling of the SLAM Process

Abstract Process

Variant with Particle Filtering

Alternative algorithms for prediction

Model-driven Multi-Quality Auto-Tuning of Robotic Applications
MQuAT Modeling of the SLAM Process

Abstract Process

Variant with Particle Filtering

Alternative algorithms for prediction

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

Model-driven Multi-Quality Auto-Tuning of Robotic Applications
Battery is a very limited resource in mobile robotic systems
  - Prediction 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**
EVALUATION
SLAM PARTICLE PREDICTION AS A SERVICE
Hardware setup

Model-driven Multi-Quality Auto-Tuning of Robotic Applications
- 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 \( n \times 100 \) 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 \( (\Delta x; \Delta y; \Delta \Phi) \)
    \( x, y = \) Position, \( \Phi = \) rotation
Model-driven Multi-Quality Auto-Tuning of Robotic Applications
# Result

<table>
<thead>
<tr>
<th>particles</th>
<th>100</th>
<th>500</th>
<th>700</th>
<th>100</th>
<th>500</th>
<th>700</th>
<th>100</th>
<th>300</th>
<th>700</th>
</tr>
</thead>
<tbody>
<tr>
<td>T (ms)</td>
<td>208.6</td>
<td>532.2</td>
<td>772.7</td>
<td>290.2</td>
<td>457.2</td>
<td>557.4</td>
<td>275.3</td>
<td>361.6</td>
<td>456.2</td>
</tr>
<tr>
<td>PC (W)</td>
<td>1.6</td>
<td>1.8</td>
<td>1.8</td>
<td>4.4</td>
<td>4.6</td>
<td>4.8</td>
<td>7.1</td>
<td>7.3</td>
<td>7.7</td>
</tr>
<tr>
<td>D</td>
<td>338.4</td>
<td>318.7</td>
<td>261.9</td>
<td>365.9</td>
<td>134.6</td>
<td>67.6</td>
<td>371.3</td>
<td>30.6</td>
<td>22.4</td>
</tr>
</tbody>
</table>
### Result

#### Response time depends on both parameters

- More boards = lower response time
- More particles = higher response time

<table>
<thead>
<tr>
<th>particles</th>
<th>100</th>
<th>500</th>
<th>700</th>
<th>100</th>
<th>500</th>
<th>700</th>
<th>100</th>
<th>300</th>
<th>700</th>
</tr>
</thead>
<tbody>
<tr>
<td>T (ms)</td>
<td>208.6</td>
<td>532.2</td>
<td>772.7</td>
<td>290.2</td>
<td>457.2</td>
<td>557.4</td>
<td>275.3</td>
<td>361.6</td>
<td>456.2</td>
</tr>
<tr>
<td>PC (W)</td>
<td>1.6</td>
<td>1.8</td>
<td>1.8</td>
<td>4.4</td>
<td>4.6</td>
<td>4.8</td>
<td>7.1</td>
<td>7.3</td>
<td>7.7</td>
</tr>
<tr>
<td>D</td>
<td>338.4</td>
<td>318.7</td>
<td>261.9</td>
<td>365.9</td>
<td>134.6</td>
<td>67.6</td>
<td>371.3</td>
<td>30.6</td>
<td>22.4</td>
</tr>
</tbody>
</table>
### Result

#### Power consumption
- mainly depends on number of boards
  - More boards = higher power consumption
  - More particles = slightly higher power consumption

<table>
<thead>
<tr>
<th>particles</th>
<th>100</th>
<th>500</th>
<th>700</th>
<th>100</th>
<th>500</th>
<th>700</th>
<th>100</th>
<th>300</th>
<th>700</th>
</tr>
</thead>
<tbody>
<tr>
<td>T (ms)</td>
<td>208.6</td>
<td>532.2</td>
<td>772.7</td>
<td>290.2</td>
<td>457.2</td>
<td>557.4</td>
<td>275.3</td>
<td>361.6</td>
<td>456.2</td>
</tr>
<tr>
<td>PC (W)</td>
<td><strong>1.6</strong></td>
<td><strong>1.8</strong></td>
<td><strong>1.8</strong></td>
<td><strong>4.4</strong></td>
<td><strong>4.6</strong></td>
<td><strong>4.8</strong></td>
<td><strong>7.1</strong></td>
<td><strong>7.3</strong></td>
<td><strong>7.7</strong></td>
</tr>
<tr>
<td>D</td>
<td>338.4</td>
<td>318.7</td>
<td>261.9</td>
<td>365.9</td>
<td>134.6</td>
<td>67.6</td>
<td>371.3</td>
<td>30.6</td>
<td>22.4</td>
</tr>
</tbody>
</table>
### Model-driven Multi-Quality Auto-Tuning of Robotic Applications

#### Result

- **Deviation** depends on both parameters
  - More boards = lower deviation
  - More particles = lower deviation

<table>
<thead>
<tr>
<th>particles</th>
<th>100</th>
<th>500</th>
<th>700</th>
<th>100</th>
<th>500</th>
<th>700</th>
<th>100</th>
<th>300</th>
<th>700</th>
</tr>
</thead>
<tbody>
<tr>
<td>T (ms)</td>
<td>208.6</td>
<td>532.2</td>
<td>772.7</td>
<td>290.2</td>
<td>457.2</td>
<td>557.4</td>
<td>275.3</td>
<td>361.6</td>
<td>456.2</td>
</tr>
<tr>
<td>PC (W)</td>
<td>1.6</td>
<td>1.8</td>
<td>1.8</td>
<td>4.4</td>
<td>4.6</td>
<td>4.8</td>
<td>7.1</td>
<td>7.3</td>
<td>7.7</td>
</tr>
<tr>
<td>D</td>
<td>338.4</td>
<td>318.7</td>
<td>261.9</td>
<td>365.9</td>
<td>134.6</td>
<td>67.6</td>
<td>371.3</td>
<td>30.6</td>
<td>22.4</td>
</tr>
</tbody>
</table>
### Result

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)

<table>
<thead>
<tr>
<th>particles</th>
<th>100</th>
<th>500</th>
<th>700</th>
<th>100</th>
<th>500</th>
<th>700</th>
<th>100</th>
<th>300</th>
<th>700</th>
</tr>
</thead>
<tbody>
<tr>
<td>T (ms)</td>
<td><strong>208.6</strong></td>
<td>532.2</td>
<td>772.7</td>
<td>290.2</td>
<td>457.2</td>
<td>557.4</td>
<td>275.3</td>
<td>361.6</td>
<td><strong>456.2</strong></td>
</tr>
<tr>
<td>PC (W)</td>
<td><strong>1.6</strong></td>
<td>1.8</td>
<td>1.8</td>
<td>4.4</td>
<td>4.6</td>
<td>4.8</td>
<td>7.1</td>
<td>7.3</td>
<td><strong>7.7</strong></td>
</tr>
<tr>
<td>D</td>
<td><strong>338.4</strong></td>
<td>318.7</td>
<td>261.9</td>
<td>365.9</td>
<td>134.6</td>
<td>67.6</td>
<td>371.3</td>
<td>30.6</td>
<td><strong>22.4</strong></td>
</tr>
</tbody>
</table>
MQuAT Optimizer Follows Change of Objective Function of ILP

- **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)

- **MQuAT Optimizer dynamically adapts SLAM by following the changes of Objective Functions**
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 acceptable**
- MQuAT optimizer follows changes of objectives
- 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