Karlsruhe Institute of Technology

Robotik I: Einführung in die Robotik
Architekturen und Frameworks

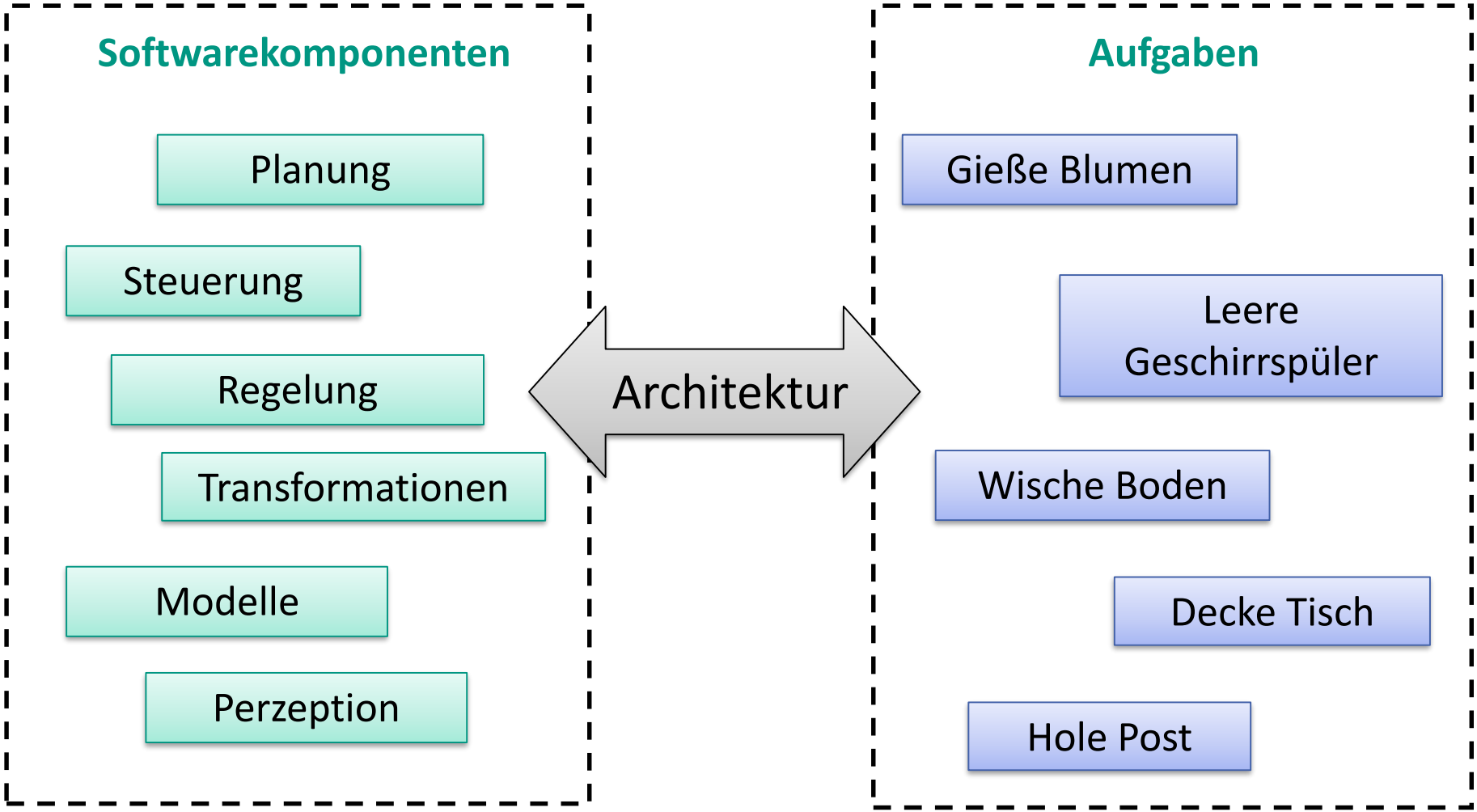
Tamim Asfour

KIT-Fakultät für Informatik, Institut für Anthropomatik und Robotik (IAR)
Hochperformante Humanoide Technologien (H^2T)

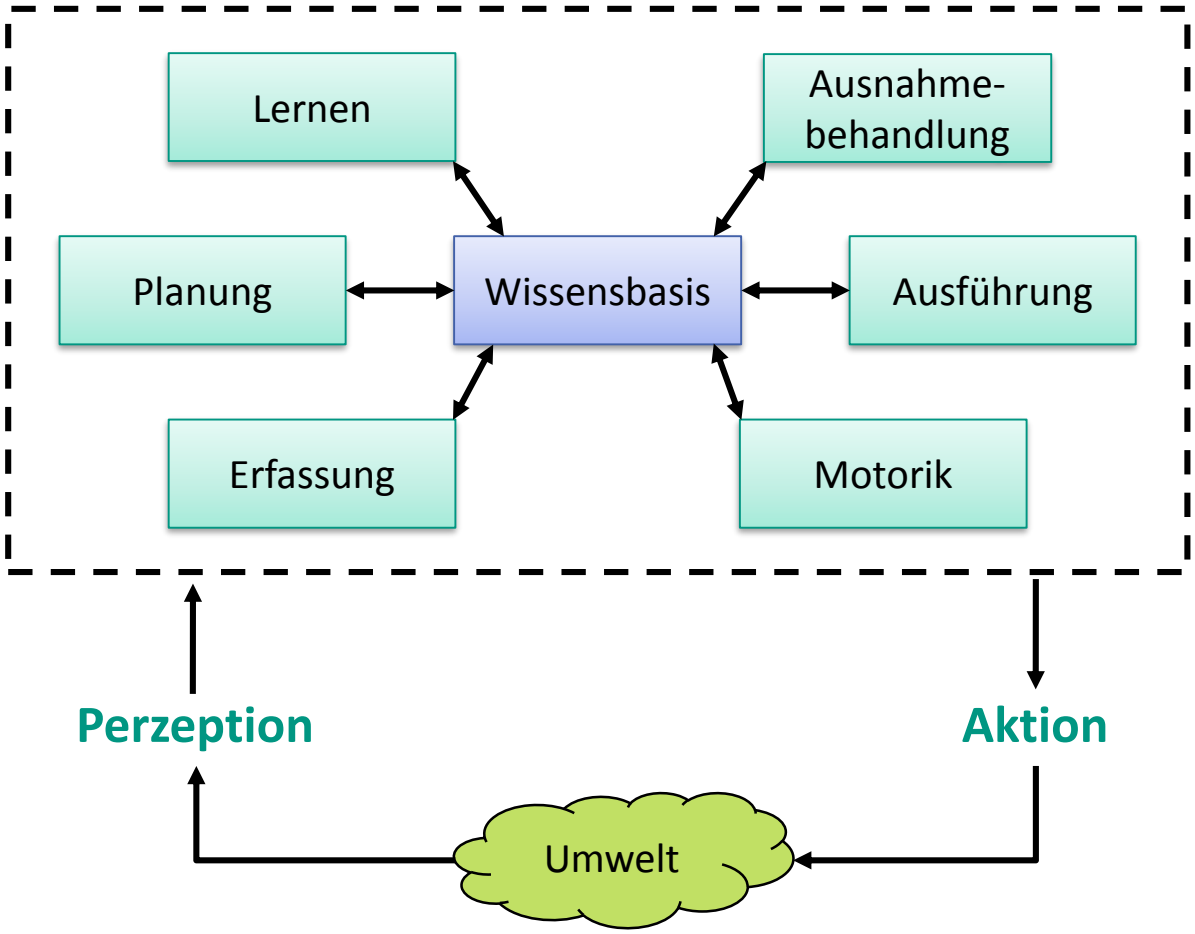
Übersicht

- **Architekturen**
  - **Einführung**
  - Funktionsorientierte Architekturen
  - Verhaltensorientierte Architekturen
  - Hybride Architekturen
  - Kognitive Architekturen
- Frameworks
- Software

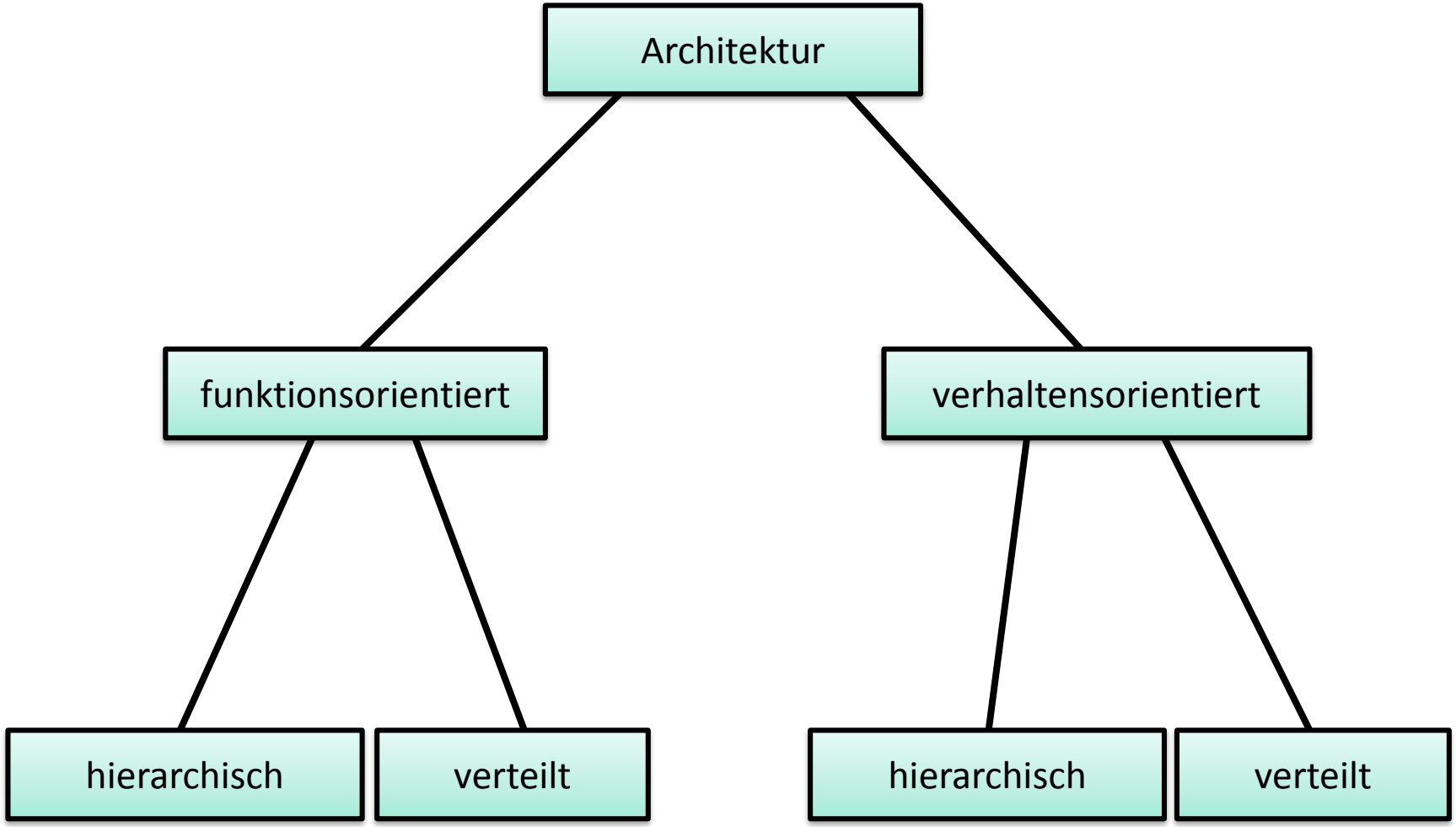
Einführung: Architektur eines Serviceroboters



Einführung: Aktions-Perzeptions-Zyklus



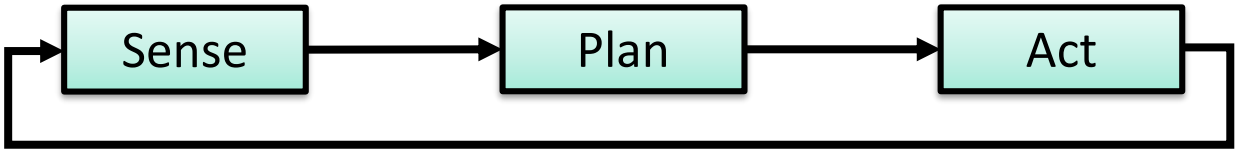
Einführung: Klassifikation von Roboterarchitekturen



Übersicht

- **Architekturen**
  - Einführung
  - **Funktionsorientierte Architekturen**
    - **Hierarchische funktionsorientierte Architekturen**
    - Verteilt funktionsorientierte Architekturen
  - Verhaltensorientierte Architekturen
  - Hybride Architekturen
  - Kognitive Architekturen
- Frameworks
- Software

Funktionsorientierte Architekturen



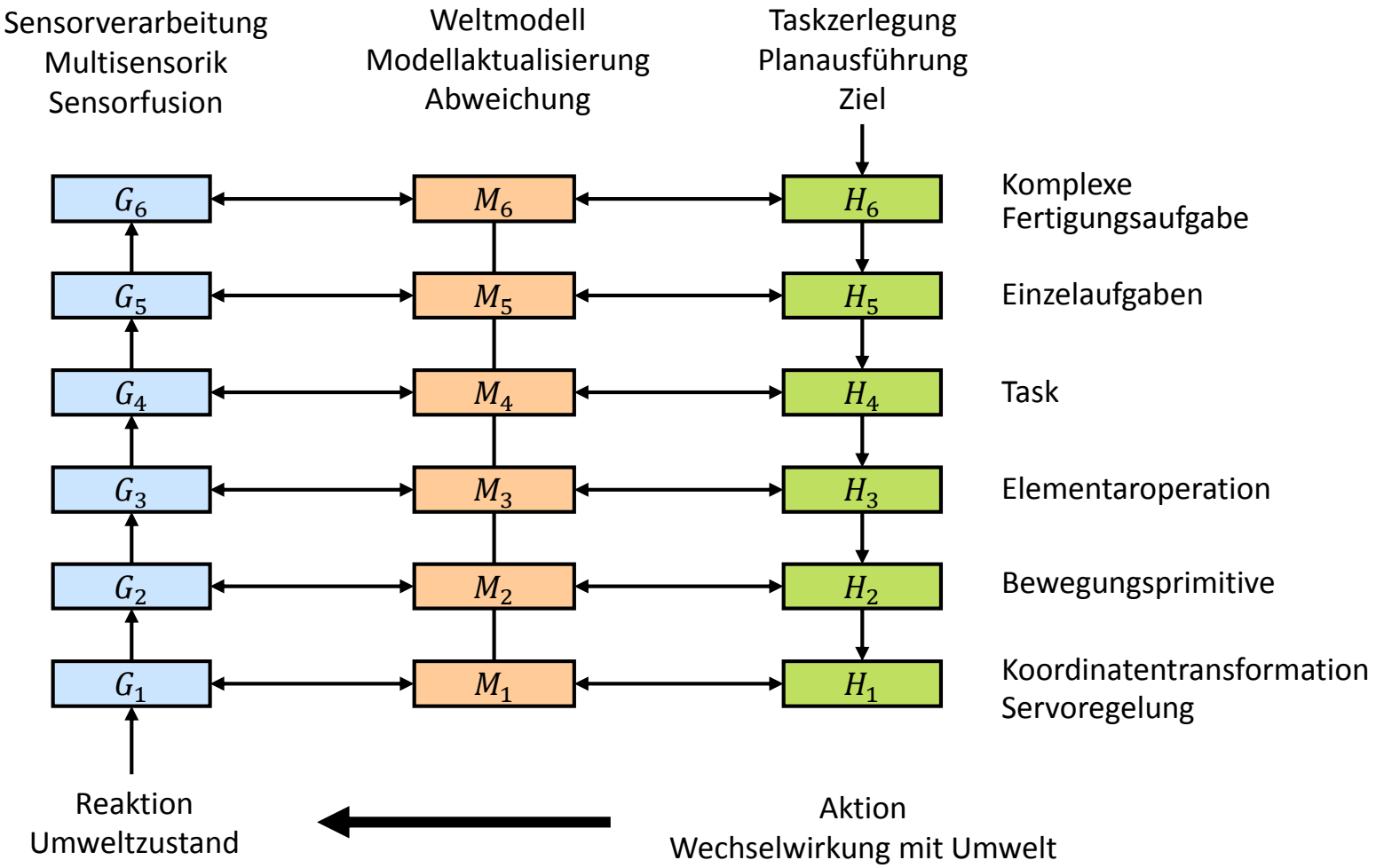
- Eigenschaften
  - Unidirektionaler Informationsfluss
  - Schlecht für dynamische Umgebungen, Interaktion
  - Vorteil: Einfacher Aufbau

NASREM-Modell

- Aufteilung in 4 bis 6 Ebenen
- Drei Module pro Ebene
  - Sensorverarbeitungsmodul G_i
  - Weltmodell- und Referenzdatenmodul M_i
  - Taskzerlegungs-, Planungs- und Ausführungsmodul H_i
- Jede Modul-Art ist durch die Ebeneneinteilung hierarchisch geordnet

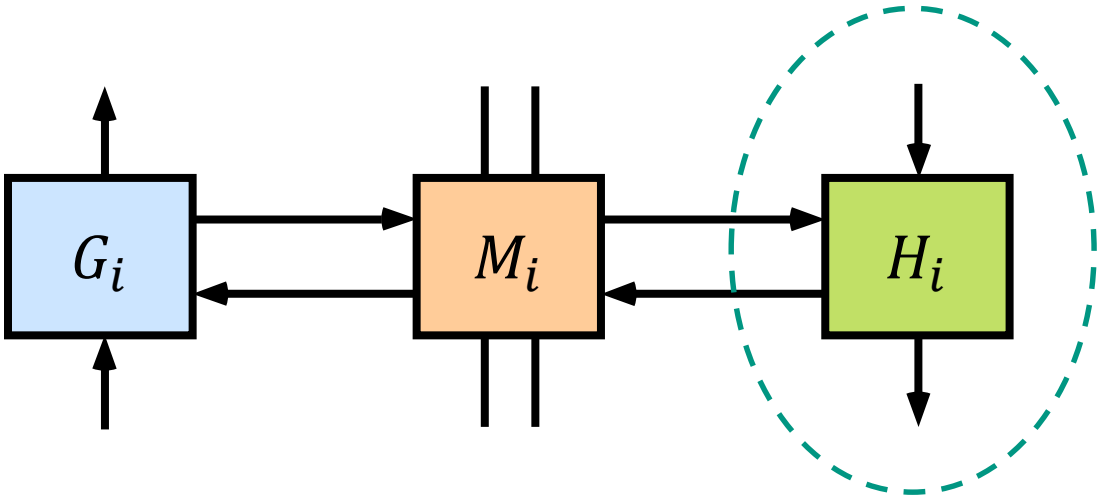
J. S. Albus, H. G. McCain, R. Lumini, *NASA / NBS Standard Reference Model for Telerobot Control System Architecture,* NBS Technical Note 1235, 1987

NASREM-Modell



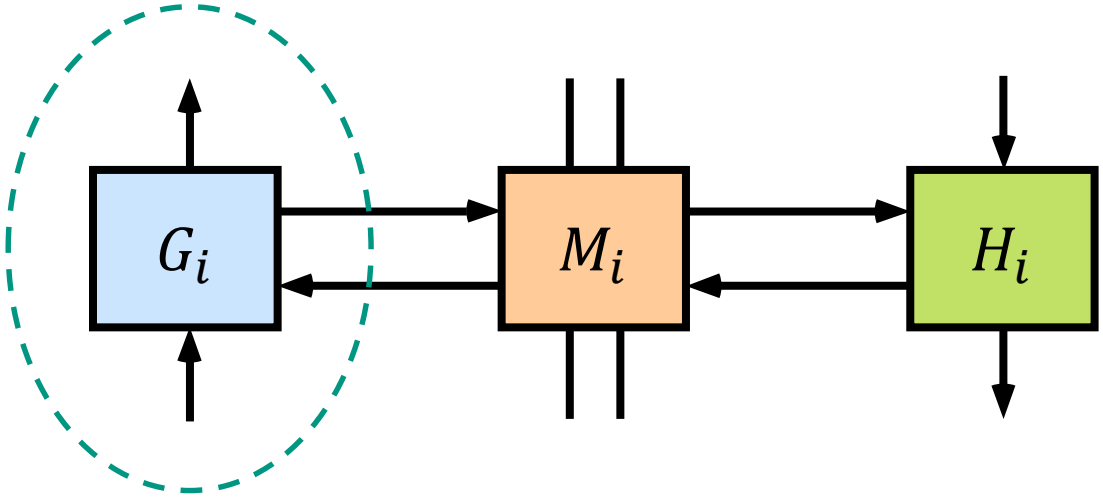
J. S. Albus, H. G. McCain, R. Lumini, *NASA / NBS Standard Reference Model for Telerobot Control System Architecture,* NBS Technical Note 1235, 1987

NASREM-Modell: Das H-Modul



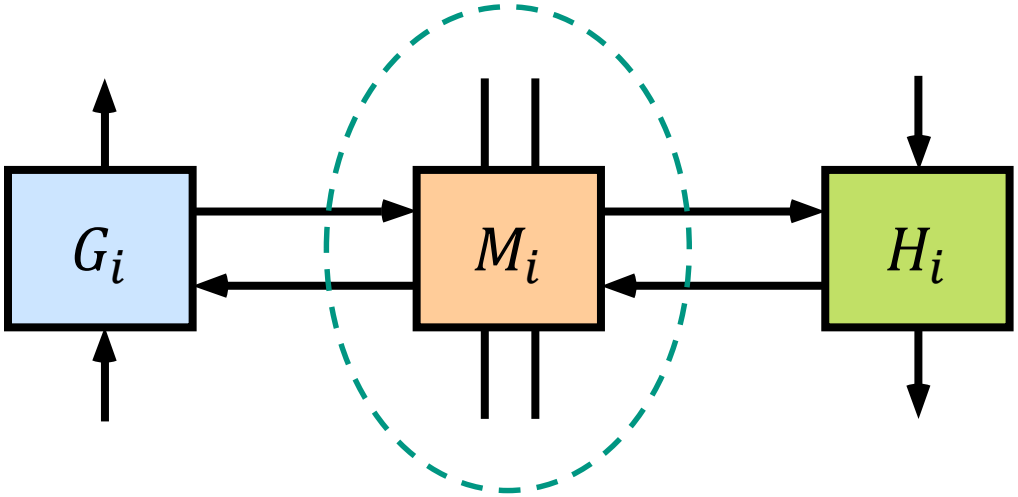
- Das H-Modul erhält eine Aufgabe aus der oberen Ebene.
- Mit den Daten aus dem M-Modul wird die Aufgabe in Teilaufgaben zerlegt
- Das M-Modul wird aktualisiert
- Die Teilaufgaben werden (einzeln) an die untere Ebene weitergereicht

NASREM-Modell: Das G-Modul



- Das G-Modul bekommt Sensor-Informationen aus der unteren Ebene
- Mit dem M-Modul werden diese Daten verarbeitet
- Das M-Modul wird aktualisiert
- Die Informationen werden der oberen Ebene zur Verfügung gestellt

NASREM-Modell: Das M-Modul



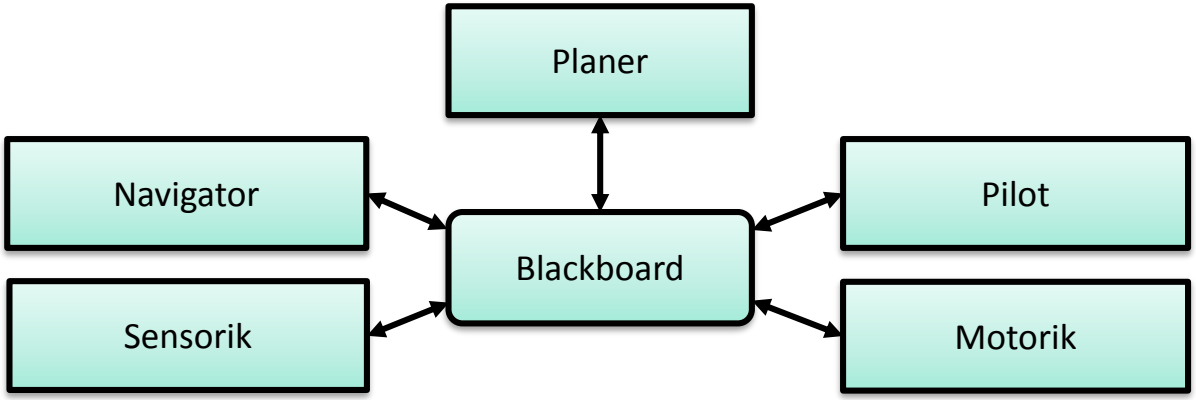
- Das M-Modul besteht aus einer einzigen Komponente, mit welcher die G- und H-Module Daten mit dem Abstraktionsgrad der jeweiligen Ebene austauschen
- Aufgrund einer Differenz zwischen den Solldaten (M-Modul) und den Ist-Daten (G-Modul) wird eine Störung erkannt

Übersicht

- **Architekturen**
  - Einführung
  - **Funktionsorientierte Architekturen**
    - Hierarchische funktionsorientierte Architekturen
    - **Verteilt funktionsorientierte Architekturen**
  - Verhaltensorientierte Architekturen
  - Hybride Architekturen
  - Kognitive Architekturen
- Frameworks
- Software

NAVLAB System (CMU)

- Verteilt funktionsorientiert
- Menge spezialisierter Teilsysteme
- Kommunikation über eine zentrale Kompetenz
- Anwendung (u.a.): Autonome Fahrzeuge



A. J. Stentz, *The NAVLAB system for mobile robot navigation,* 1990

Nav-Lab System (CMU)

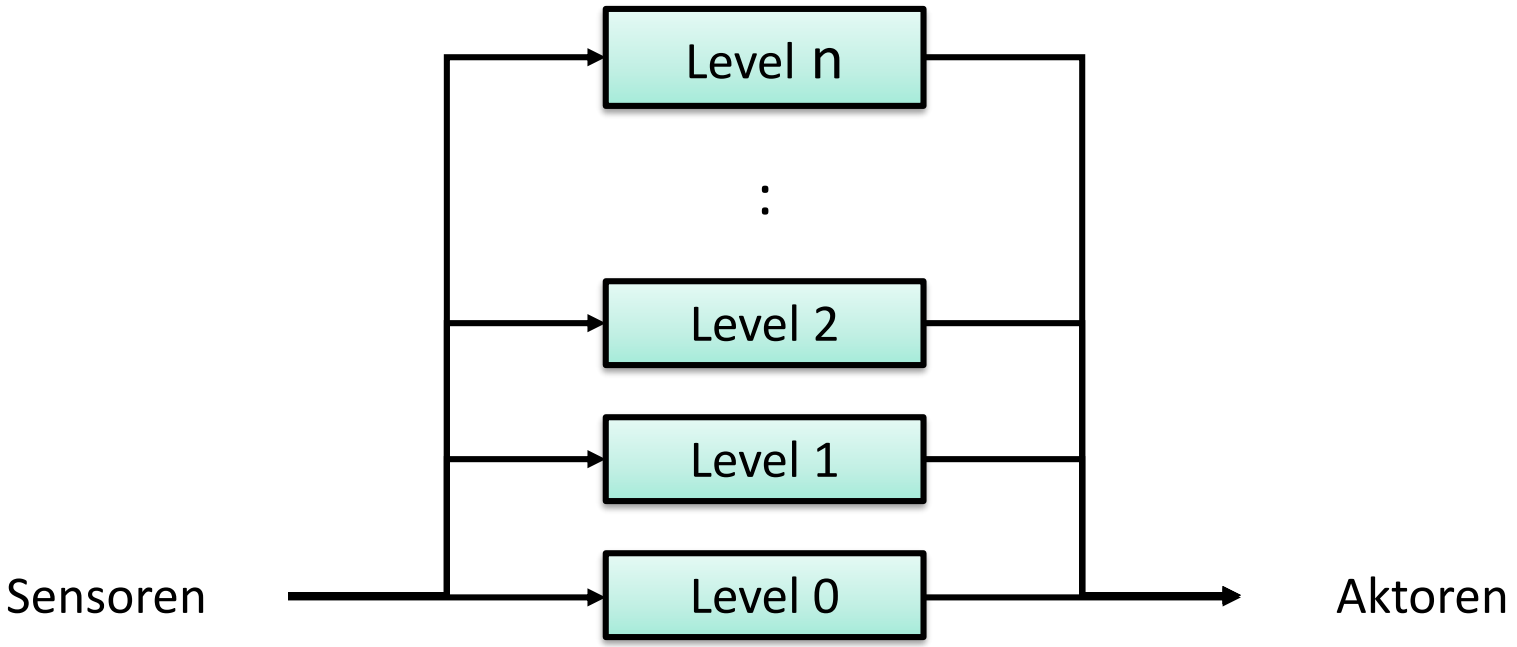
- Ähnliche Architektur eingesetzt vom KIT-Team AnnieWAY
- Teilnehmer bei der DARPA Urban Challenge 2007

Übersicht

- **Architekturen**
  - Einführung
  - Funktionsorientierte Architekturen
  - **Verhaltensorientierte Architekturen**
    - **Hierarchische verhaltensorientierte Architekturen**
    - Verteilte verhaltensorientierte Architekturen
  - Hybride Architekturen
  - Kognitive Architekturen
- Frameworks
- Software

Hierarchisch verhaltensorientierte Architekturen (I)

- Hierarchische Anordnung der Einzelverhalten
- Übergeordnete Verhalten können die Ausgabe darunter liegender Verhalten hemmen



Hierarchisch verhaltensorientierte Architekturen (II)

- Steuerung durch Verhaltensmuster bzw. Reflexe
  - Reaktionen des Systems auf bestimmte Sensorstimuli
- Anordnung in Verhaltensebenen
- Hierarchische Struktur der Kompetenzebenen
- Beispiel: *Subsumption Architecture*
  Verhaltensschemata als endliche Zustandsautomatennetze

R. A. Brooks, *A robust layered control system for a mobile robot,* IEEE Journal on Robotics and Automation 2(1), 1986, pp 14-23.

Übersicht

- **Architekturen**
  - Einführung
  - Funktionsorientierte Architekturen
  - **Verhaltensorientierte Architekturen**
    - Hierarchische verhaltensorientierte Architekturen
    - **Verteilte verhaltensorientierte Architekturen**
  - Hybride Architekturen
  - Kognitive Architekturen
- Frameworks
- Software

Verteilt verhaltensorientierte Architekturen

- Menge von unabhängigen Teilsystemen mit identischen Verhaltensmustern
- Koordination erfolgt über ein Verhaltensmuster
- Beispiel: Eigenschaften von Multi-Agenten-Systemen
  - Selbstorganisation
  - Assoziative Speicherung von Information
  - Erkennung von bestimmten Mustern

Übersicht

- **Architekturen**
  - Einführung
  - Funktionsorientierte Architekturen
  - Verhaltensorientierte Architekturen
  - **Hybride Architekturen**
  - Kognitive Architekturen
- Frameworks
- Software

Hybride Architekturen (I)

- Meistens 3-Schichten Modell
  - Obere Ebene: Planung
  - Mittlere Ebene: Sequentialisierung
    (Ausführen/Abspielen von Plänen)
  - Untere Ebene: Verhaltensebene

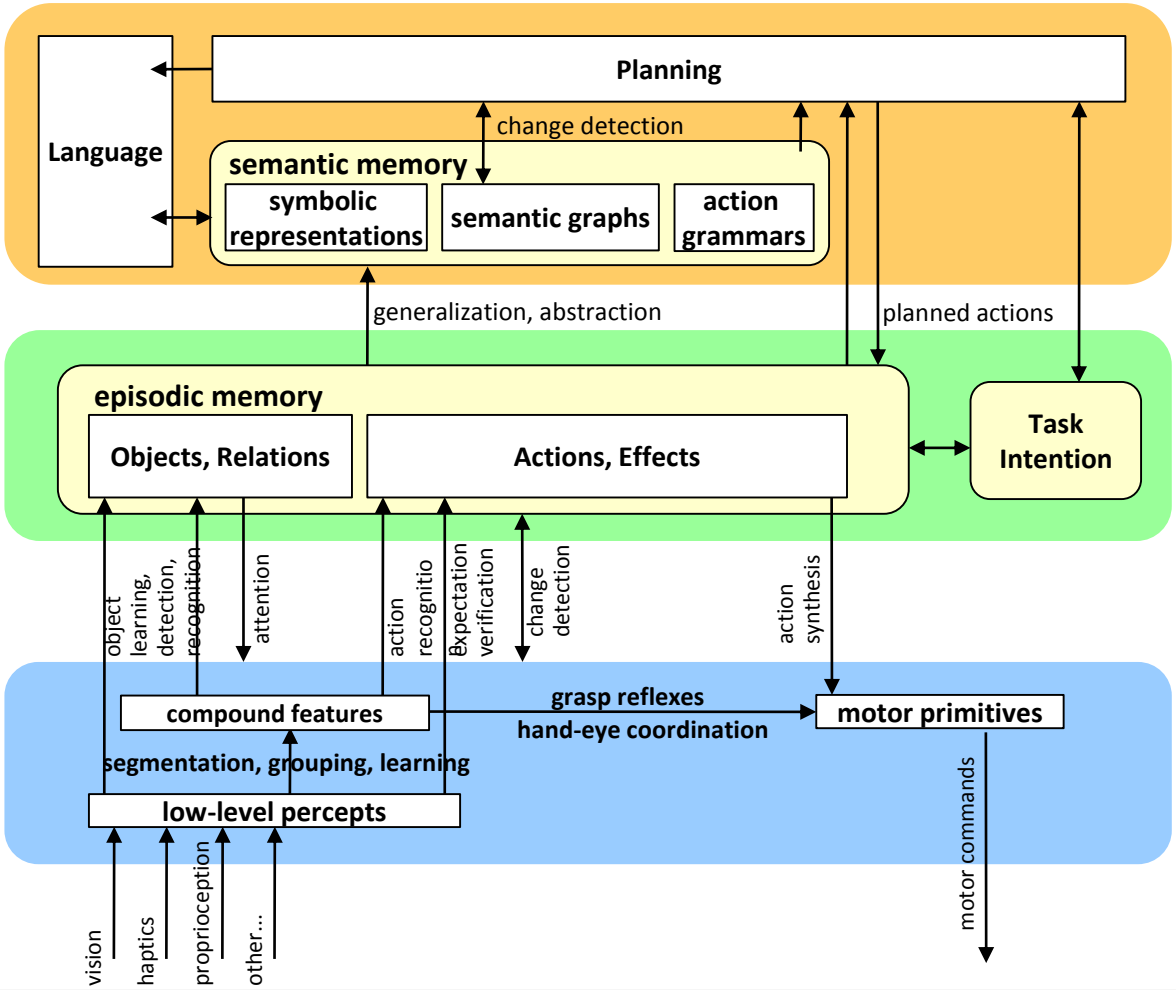
Hybride Architekturen (II)

Beispiel: PACO+ Architektur

High Level
- Reasoning
- Planning
- Language

Mid Level
- Recognition
- Memory
- Consolidation
- Action
- Selection

Low Level
- Online Sensorimotor Processing



Übersicht

- **Architekturen**
  - Einführung
  - Funktionsorientierte Architekturen
  - Verhaltensorientierte Architekturen
  - Hybride Architekturen
  - **Kognitive Architekturen**
- Frameworks
- Software

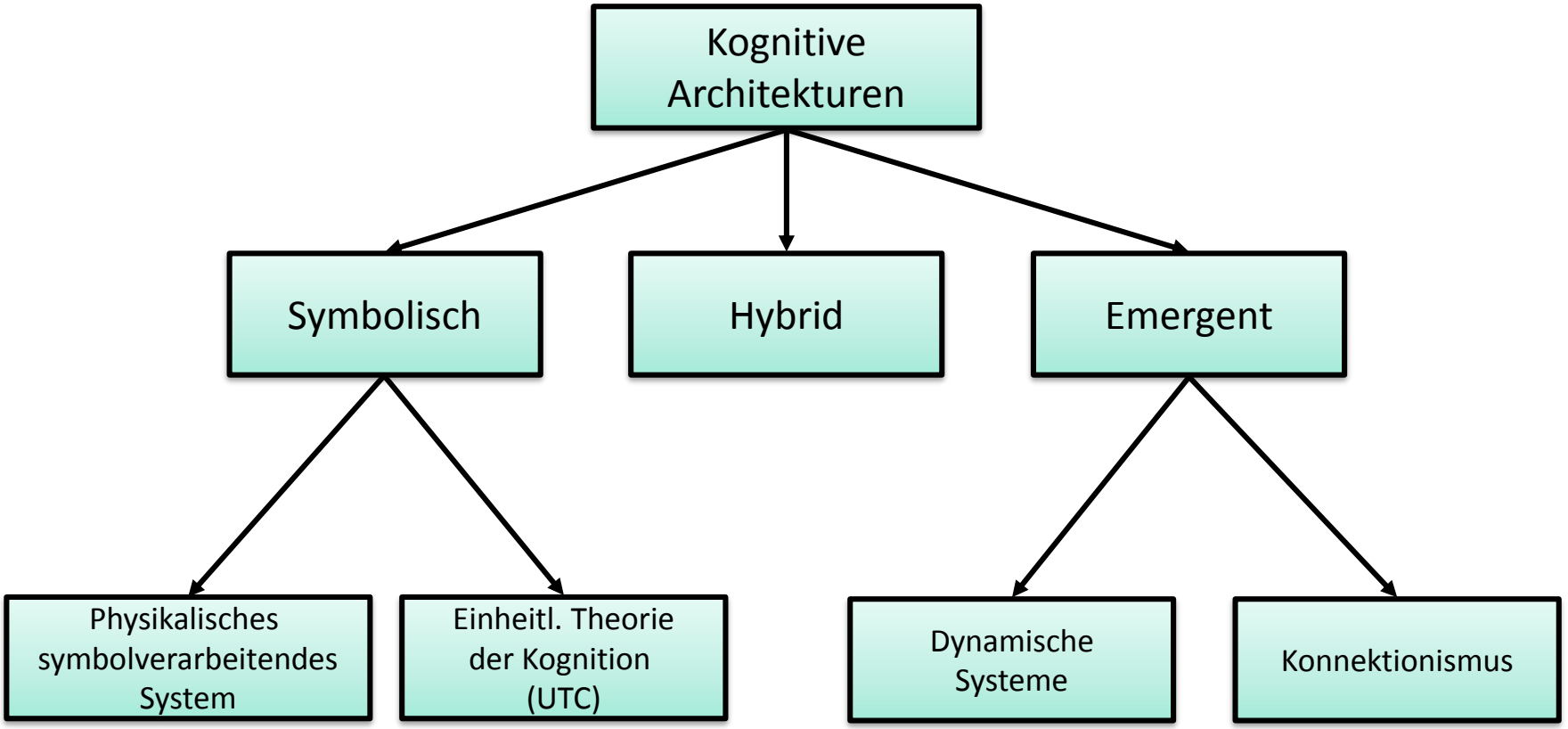
Kognitive Architekturen

- **Definition** (P. Langley, CMU, 2006):
  - „A cognitive architecture specifies the infrastructure for an intelligent system that remains constant across different domains and knowledge bases.
  - This infrastructure includes a commitment to formalisms for representing knowledge, memories for storing this domain content, and processes that utilize and acquire the knowledge.“

- Problem:
  - Lernfähigkeit des Roboters
  - Hinzufügen neuen Handlungswissens

- Keine eindeutige Definition
  - Vorbild: kognitive Prozesse des Menschen
  - Ziel: Eigendynamik, Erweiterbarkeit, Generalisierung des Systems

Kognitive Architekturen



Kognitive Architekturen: Symbolisch

- Basieren auf einer symbolische Repräsentation
- Vorschriften zur Verarbeitung der Symbole, oft in Form von Regeln

- Beispiele:

  - SOAR: *Symbolic cognitive architecture*
    J. Laird, A. Newell , P. Rosenbloom, 1987

  - ACT-R: *Adaptive Control of Thought – Rational*
    J. R. Anderson, 1990

Kognitive Architekturen: Emergent

- Emergenz

  Spontane Herausbildung von Phänomenen oder Strukturen auf der Makroebene eines Systems auf der Grundlage des Zusammenspiels seiner Elemente

- Basierend auf einfachen Grundelementen, komplexe Funktionen durch Verschaltung der Elemente

- Repräsentationen sind subsymbolisch und verteilt

- Beispiel: Neuronale Netze

Übersicht

- Architekturen
- **Frameworks**
  - ROS
  - ArmarX
  - YARP
  - Matlab/Simulink
  - Orocos
  - OpenRTM
- Software

ROS – Robot Operating System

- Robotics middleware
  Not an operating system!
- Very active community
- Launched in 2008 by Willow Garage
- Currently maintained by
  Open Source Robotics Foundation (OSRF)

- Supported OS: Ubuntu Linux
  Experimental: Fedora, macOS, Windows
- Main programming languages:
  C++, Python

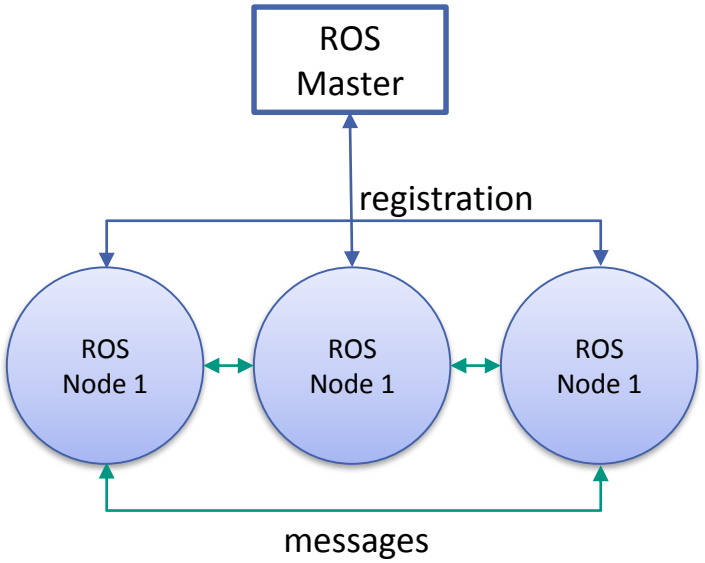
www.ros.org

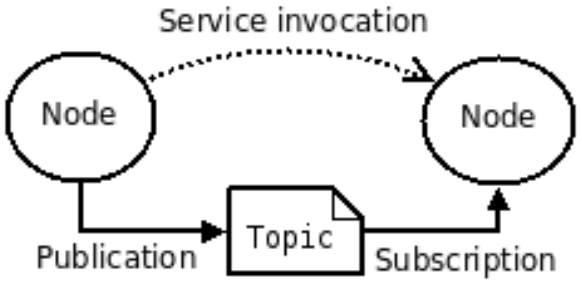
wiki.ros.org

Open Source Robotics Foundation

ROS - Concepts

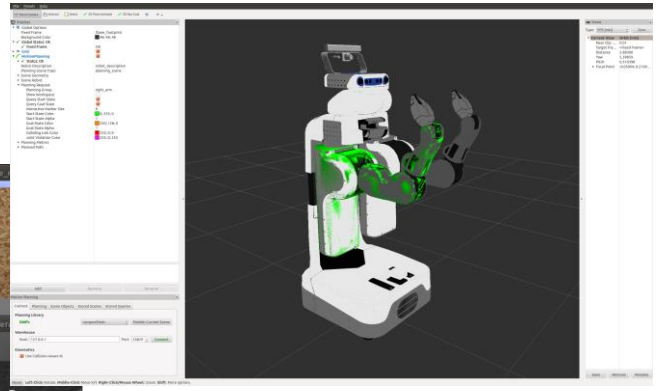
- ROS Master
  - Name registration ad lookup
- Nodes
  - Processes that perform computation
- Communication
  - TCP/IP based
    (no real-time communication)
  - Messages
    basic datatypes
  - Topics
    publisher subscribe mechanism
  - Services
    remote procedure calls (RPC)

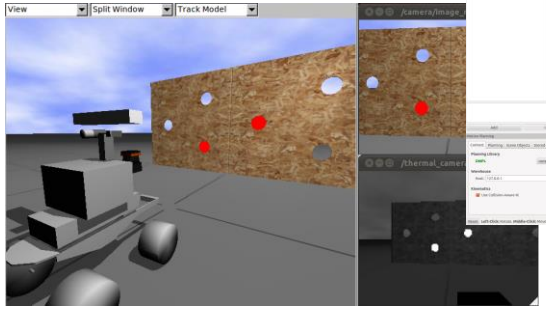


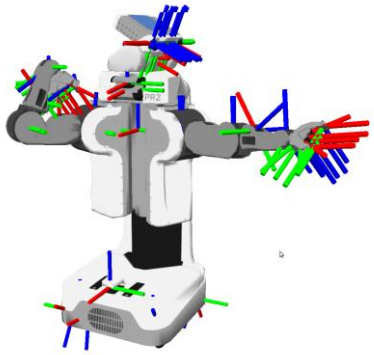


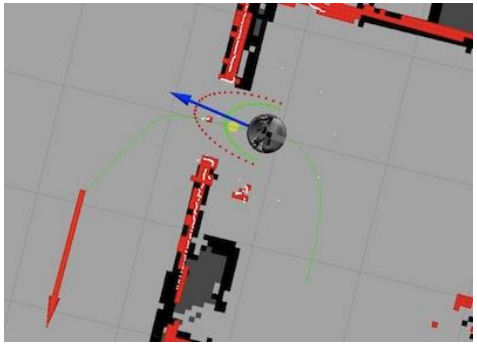
ROS – Main Packages

- Rviz
  - Visualization package
- Gazebo
  - ROS Bindings
  - Dynamics Simulator
- TF
  - Coordinate transformations
- URDF
  - Universal Robot Description Format
  - XML-based robot description
- Navigation
  - 2D navigation
- MoveIt!
  - Motion Planning framework
  - Integrates OMPL (Open Source Motion Planning Library)









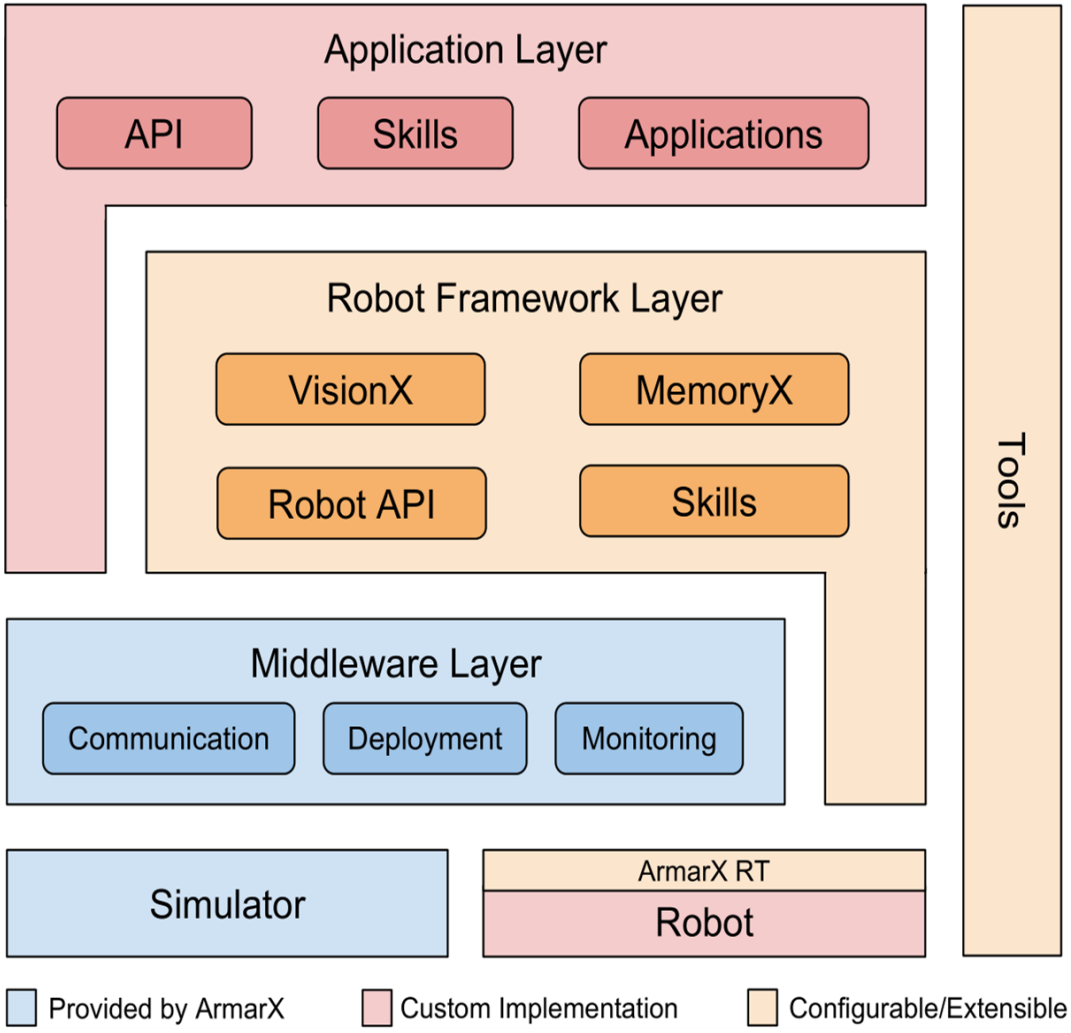
ArmarX

- Distributed Robot Software Framework
  - Separated components (on multiple PCs)
  - Robust to failures / crashes
  - Interaction via well defined interfaces
- Abstracts Hardware Access
  - Interfaces for HW access
  - Simulation
- High Level Robot Programming with Statecharts
  - Graphical Robot Program
  - Access on C++ level
  - Re-usability of robot skills
- Consistent State Disclosure
  - Monitoring on all levels
- Developed at H²T
  - https://armarx.humanoids.kit.edu

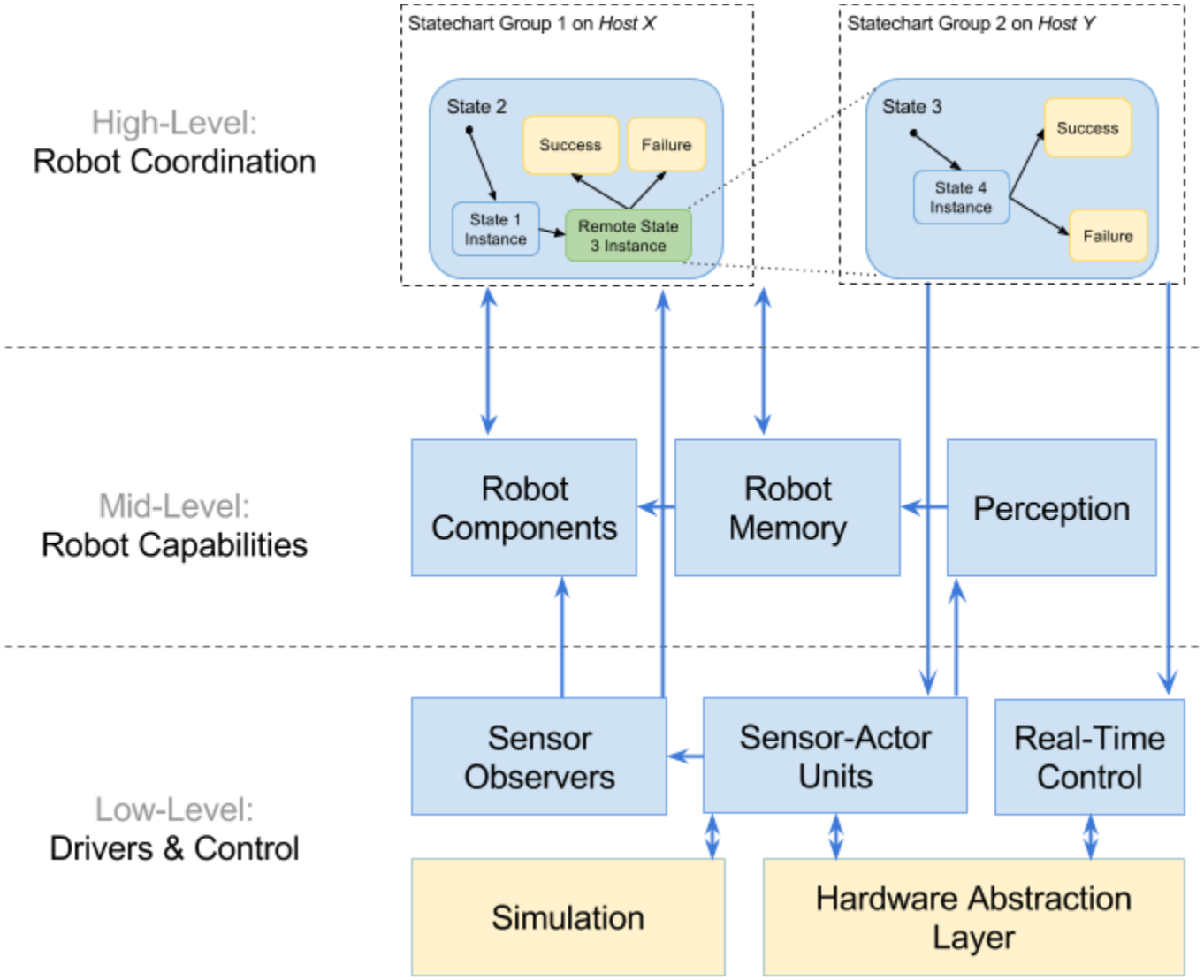
Design Principles of ArmarX

- Disclosure of Internal State
- Extensive Support for Introspection and Debugging
- Graphical editing of Control- and Dataflow
- Interface Definition Language (IDL) for unified interfaces
- Distributed Communication
- Shared, Distributed Resources
- Heterogenous Environments (C++, Java, Python, ...)

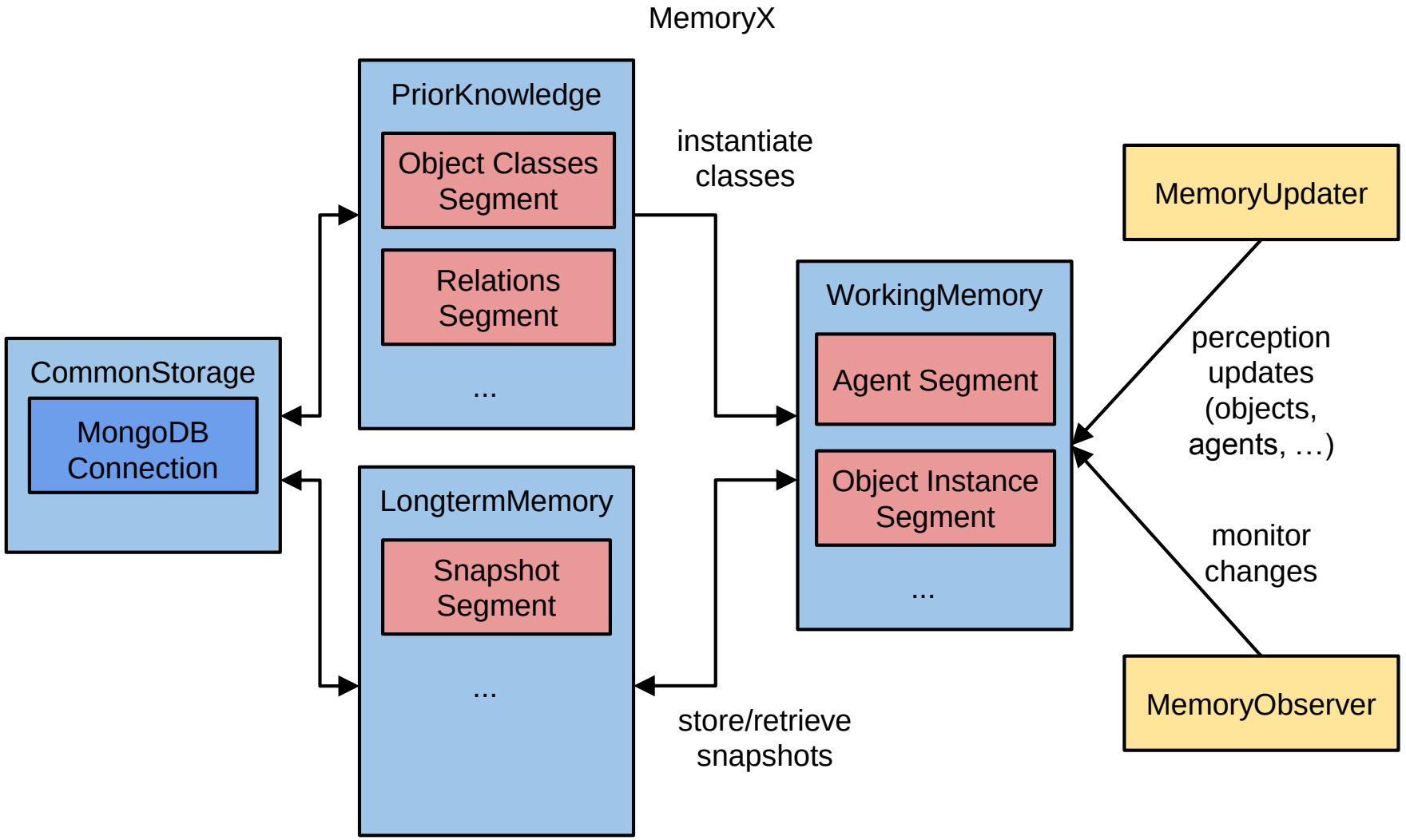
ArmarX Software Structure



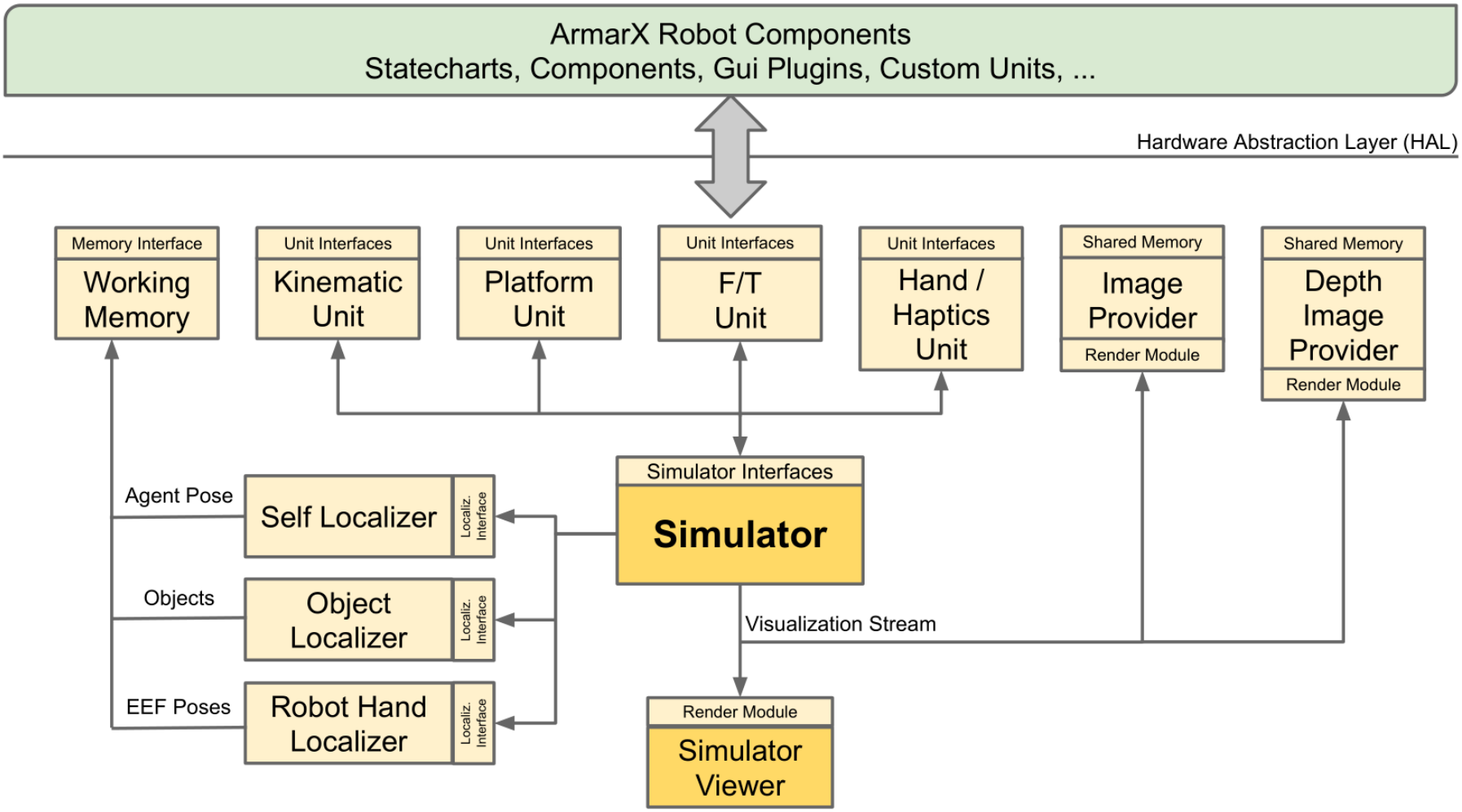
Robot Control Layers



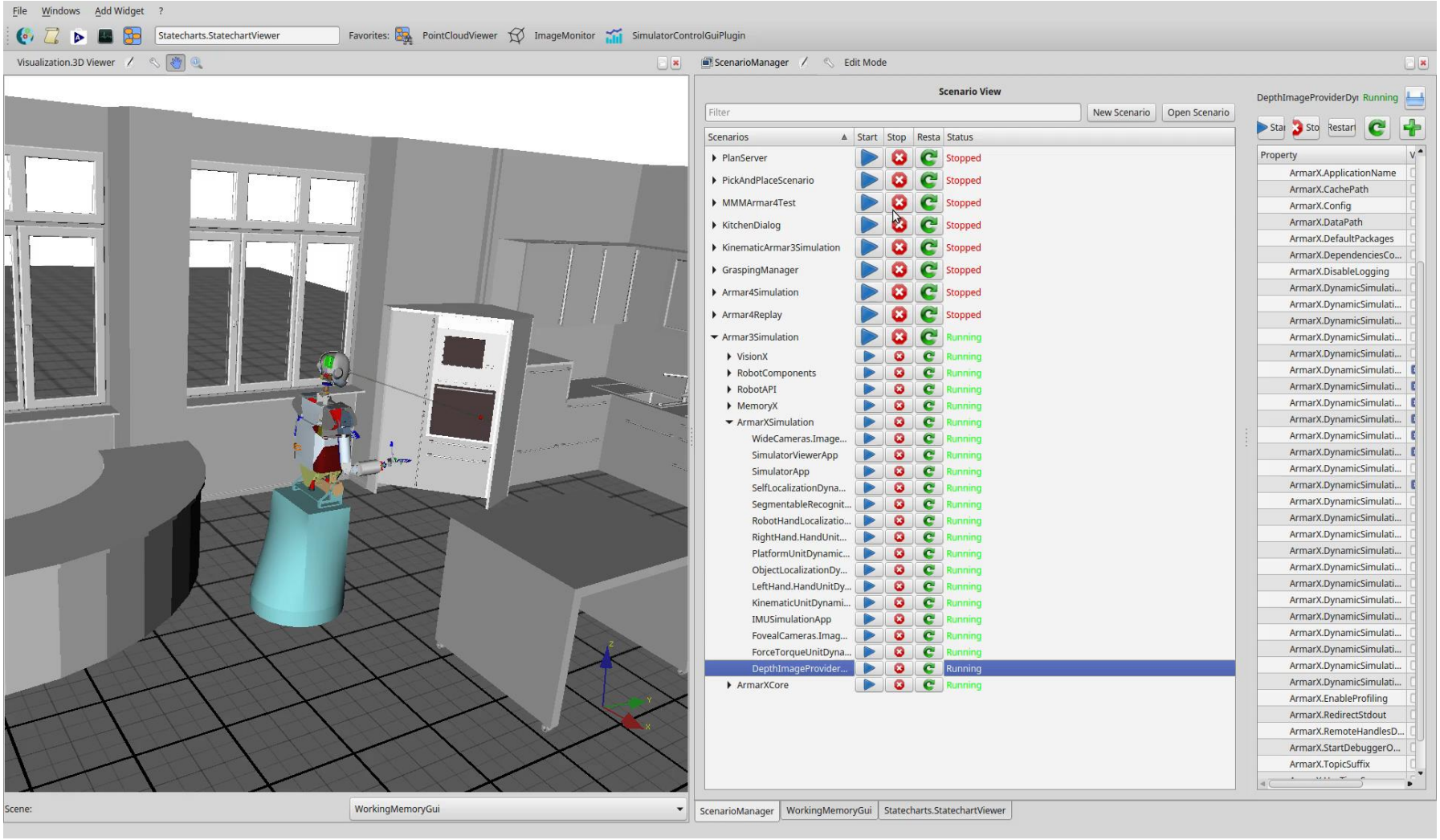
MemoryX: Overview



The ArmarX Simulator

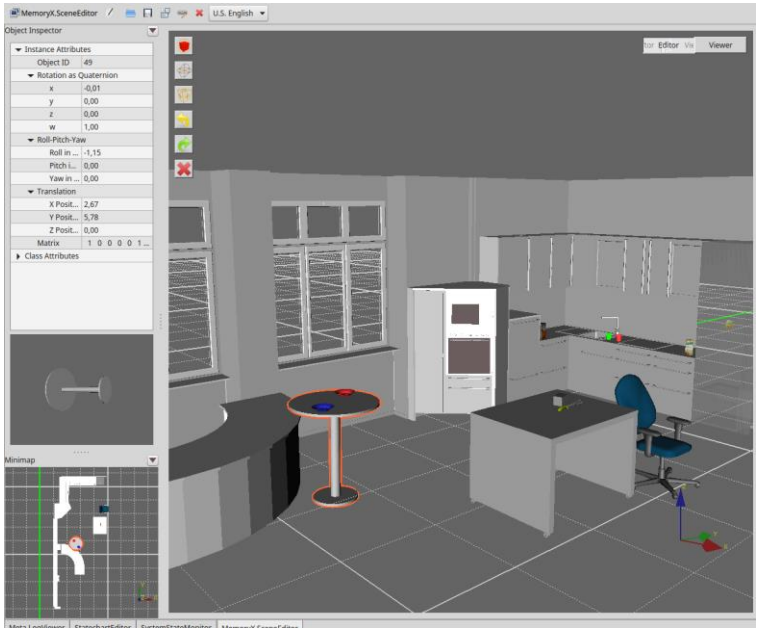


Pick and Place with Statecharts in the ArmarX Simulator



ArmarX Tools

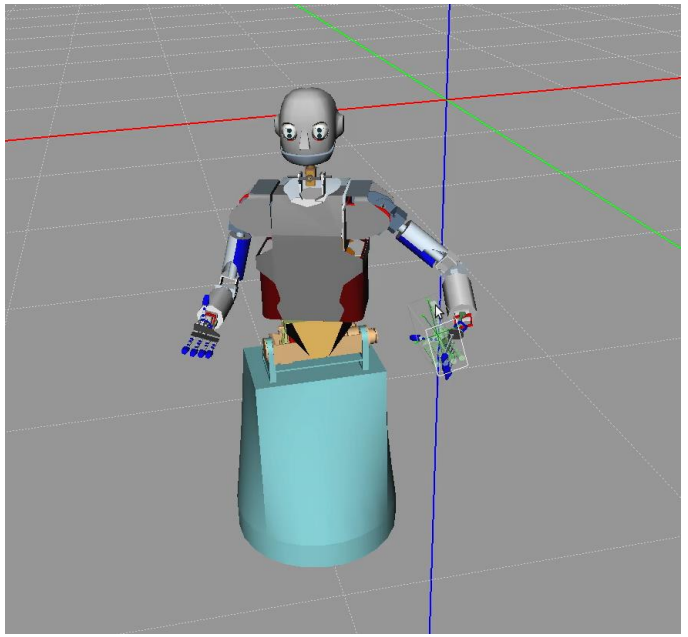
- Statechart Editor
  - Modelling and managing of robot skills
- Statechart Viewer
  - Inspecting and controlling statechart execution
- Working Memory Viewer
  - Inspecting the robot‘s belief about the world state
- Scene Editor
  - Modelling of 3D environments



Scene Editor

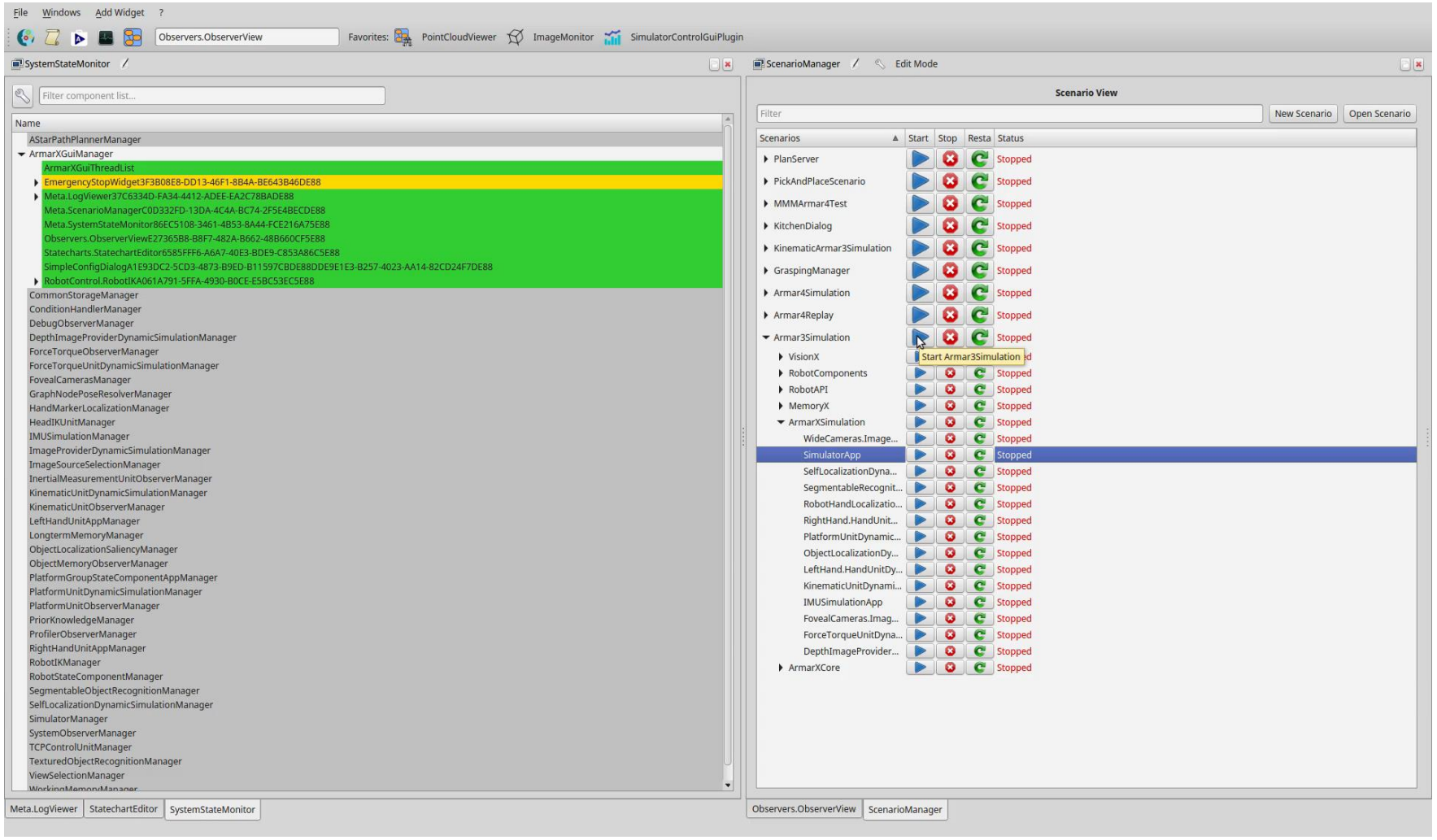
ArmarX Tools

- Interactive Robot IK
  - Online 3D manipulation of all end-effectors
- DebugDrawer
  - Convenient 3D Visualization from all ArmarX components
  - Displayed in ArmarX 3D Viewers (RobotViewer, Working Memory Viewer, Simulator,…)
- Camera and PointCloud Viewer



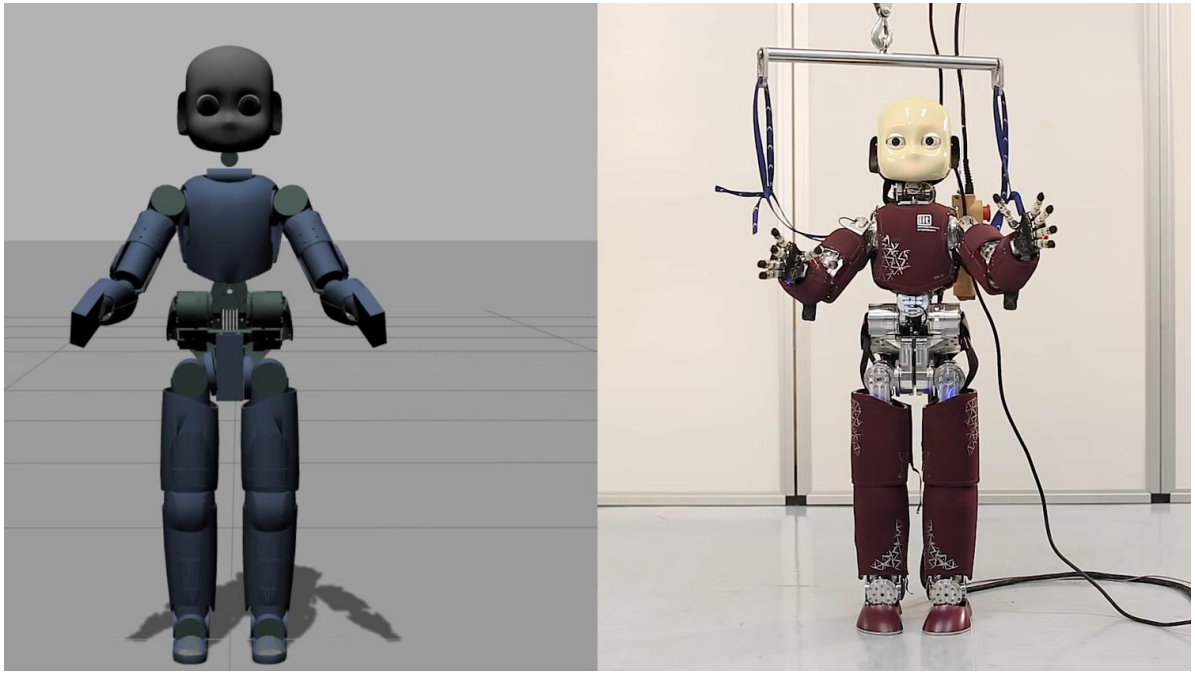
Interactive Robot IK

System State Disclosure on all Abstraction Levels



YARP (Yet Another Robot Platform)

- General
  - Supports building a robot control system as a **collection of programs** communicating in a peer-to-peer way
  - **Channel priorization**: assign priorities to individual connections (e.g. sensors)
  - Used for the iCub Robot



YARP (Yet Another Robot Platform)

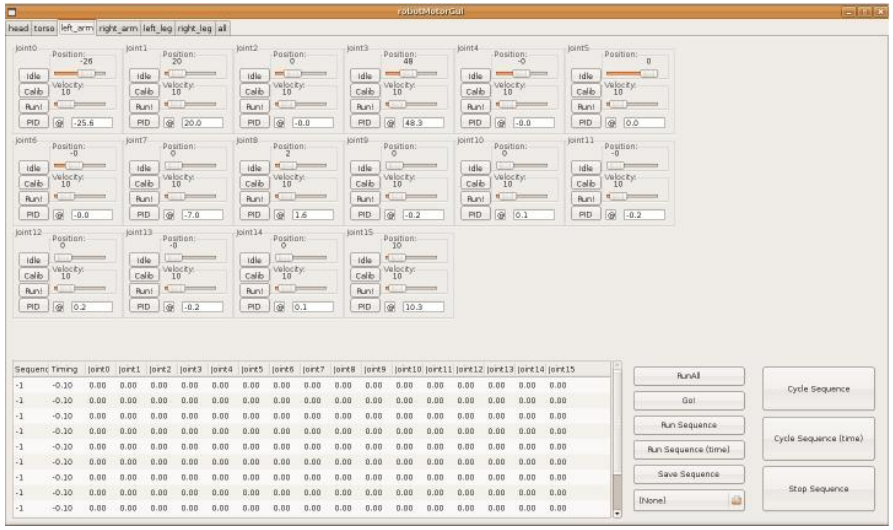
- General
  - No robot operation system
  - No control of the robotic system

Main Components

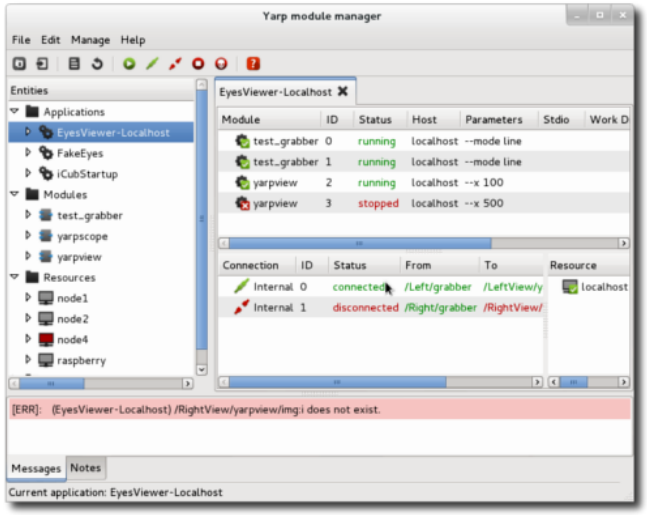
- Yarp_OS:
  - Interfacing with the operating system to support data streaming across threads and machines
- Yarp_sig:
  - Performs signal processing tasks
- Yarp_dev:
  - Interfaces with common robotic devices (framegrabbers, digital cameras, motor controller boards…)

YARP Guis

- Motorgui
  - Running joints (position control)
  - Read joint values
  - Calibration of single joints
  - Check if the robot is in position



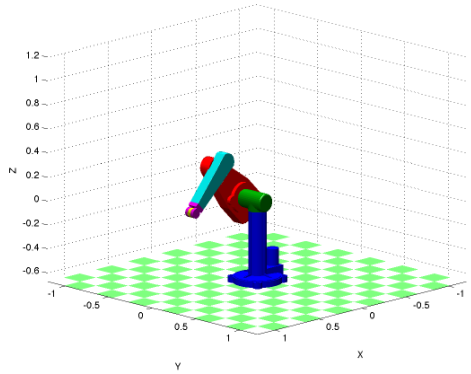
- Yarp Manager
  - Run, stop and kill programs
  - Load balancing
  - Establish port connections between programs

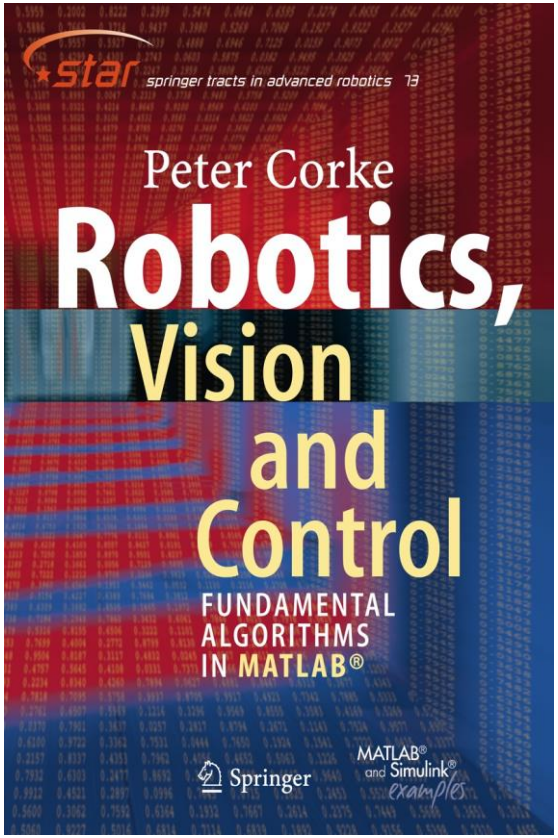


- Guis for image visualization, data recording, battery management…
- http://www.yarp.it

MATLAB / Robotics Toolbox

- Set of tools for MATLAB
  - Inverse / forward kinematics
  - Inverse / forward dynamics
  - Trajectory generation
  - Animation of robots
  - Navigation for mobile robots
    - RRT, probabilistic roadmaps…
  - Localization
  - Computer vision

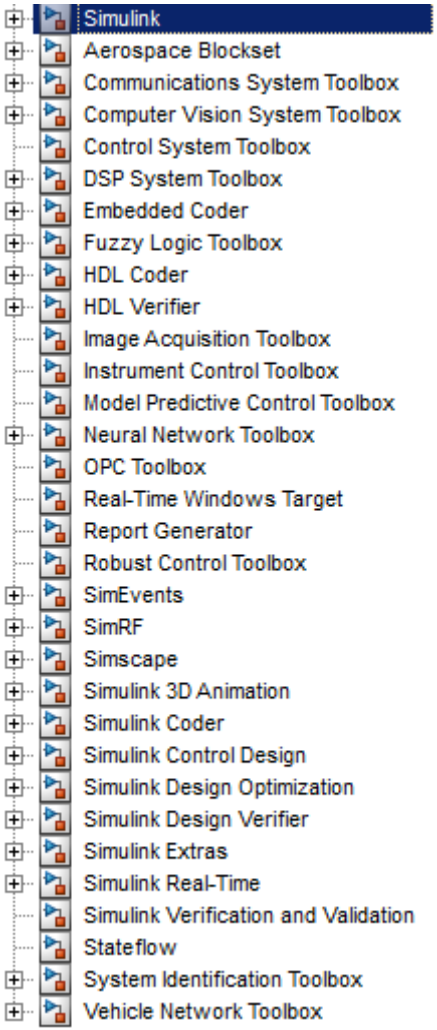




- http://petercorke.com/wordpress/toolboxes/robotics-toolbox

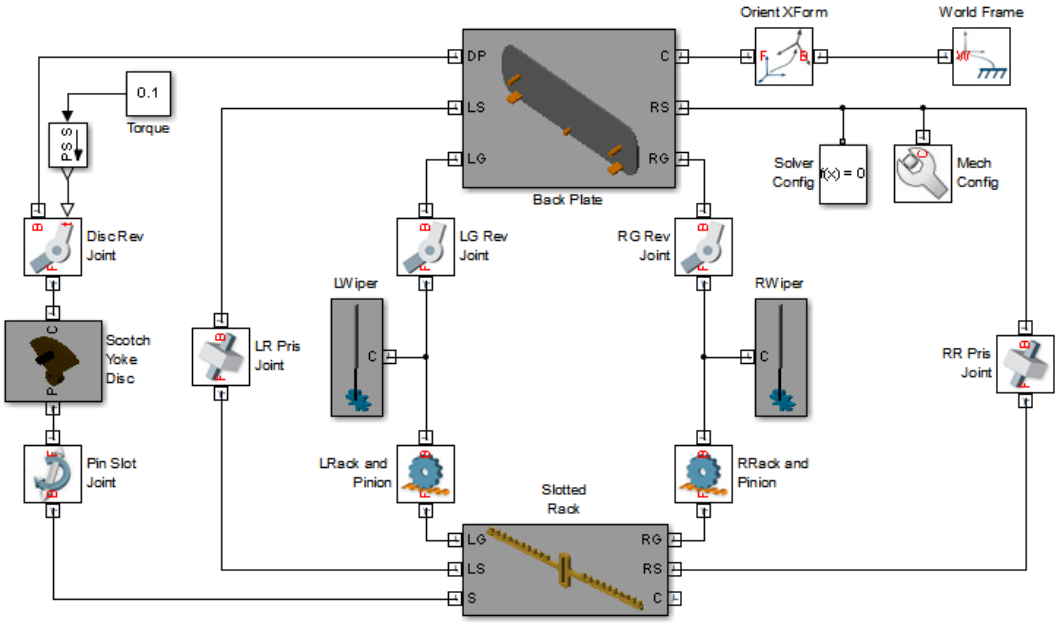
MATLAB / Simulink

- Features
  - Block diagram environment for multidomain simulation
  - Libraries of predefined blocks for modeling
    - Control
    - Mechanic simulation
    - Neural Networks
    - Vision
    - Real-Time
    - Robotics Systems Toolbox
  - Simulation engine with fixed-step and variable-step ODE solvers



MATLAB / Simulink

- Interfaces:
  - ROS
  - Hardware support packages for Arduino, Raspberry Pi, …
  - SDK-based interfaces to Microsoft Kinect, Nao Robot, ...
  - CAN, TCP/IP, RS232, EtherCAT, Bluetooth, …
- Available at KIT Softwareshop



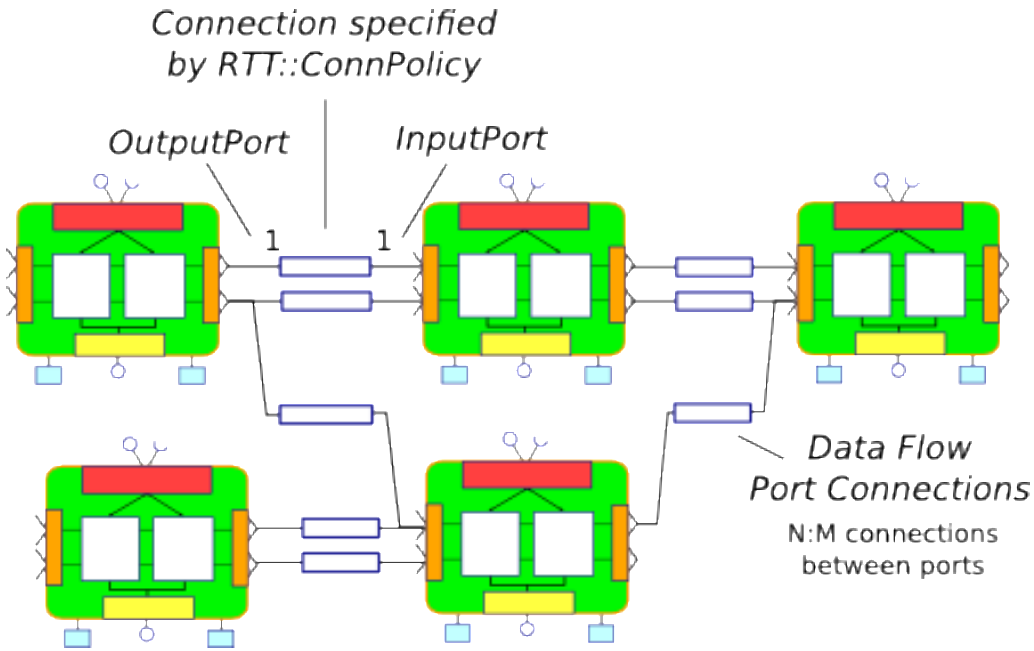
Orocos (Open RObot Control Software)

- Framework für Echtzeitregelung von Robotern und Werkzeugmaschinen
- Komponenten-basiert
- Event-basierte Kommunikation
- Starker Einsatz von Design Pattern
- Plattformunabhängig
- Hauptbestandteile
  - Orocos Toolchain (Orocos Component Library, Real-Time Toolkit)
  - Kinematics-Dynamics Library (KDL)
  - Bayesian Filtering Library (beinhaltet z.B. allgemeine Kalman- oder Partikel-Filter)

- http://www.orocos.org

Orocos (Open RObot Control Software)

- Bestandteile einer Orocos Komponente
  - Ein-/Ausgabe-Ports (orange)
  - Konfigurationsinterface (gelb)
  - Serviceinterface (rot)
  - Verhaltensspezifikation (weiß)
    - C++ Funktionen
    - Zustandsautomaten (rFSM)



Quelle: http://www.orocos.org

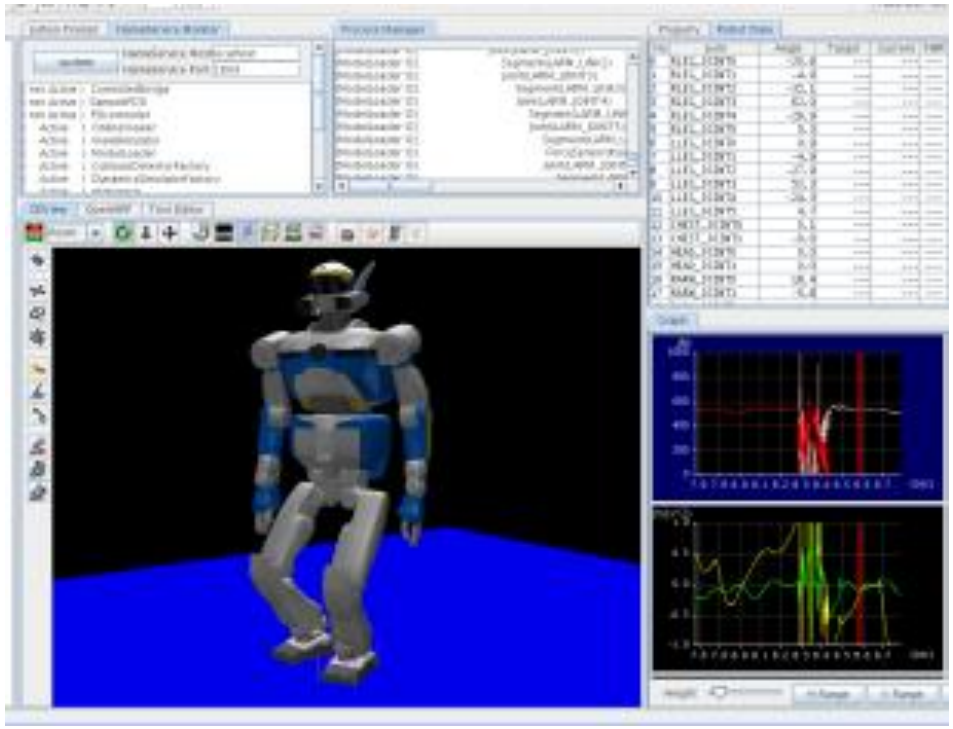
OpenRTM (Open *Robotics Technology Middleware)*

- Verteilte objektorientiert Middleware für Roboter
- Hauptkonzept: RT-Component (RTC)
- Lebenszyklus einer RT-Komponente:
  - CREATED
  - INACTIVE
  - ACTIVE
  - ERROR
- Kommunikation zwischen RT-Komponenten erfolgt über Ports

- http://www.openrtm.org/

OpenHRP3

- OpenHRP3 (Open Architecture Human-centered Robotics Platform version 3)
  - Komponentenbibliothek auf OpenRTM-Basis
  - Sehr präziser Dynamiksimulator
  - Entwicklung und Inspektion von Robotermodellen und –programmen mit Hilfe von Dynamiksimulation

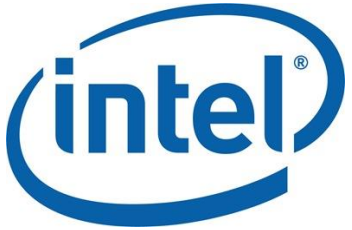


Übersicht

- Architekturen
- Frameworks
- **Software**
  - OpenCV
  - PCL
  - Simox
  - MMM
  - GraspIt!
  - RobotEditor
  - Gazebo Simulator
  - Vrep Simulator

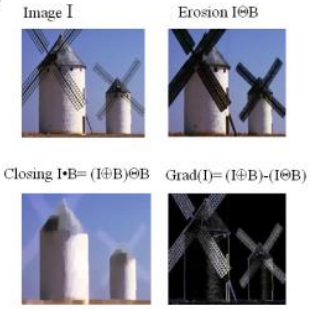
OpenCV

- **OpenCV** stands for **Open** Source **C**omputer **V**ision library
- OpenCV library was created by Intel in 1999 and is now receiving ongoing support from Willow Garage
- Available for C, C++, and Python
- Open Source and free for commercial and research use
- The library supports Linux, Windows and Mac OS
- Easy to use and install



OpenCV

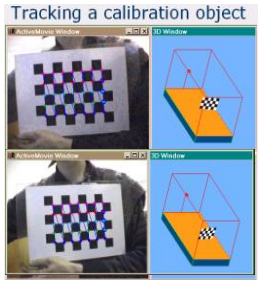
- OpenCV has more than 500 algorithms!



General Image Processing Functions

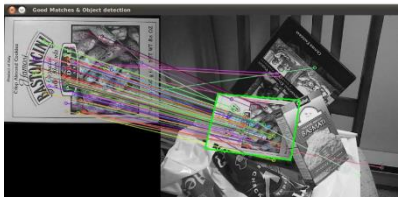
Segmentation

Tracking



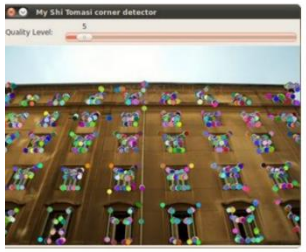
Camera Calibration

Face Detection and Recognition



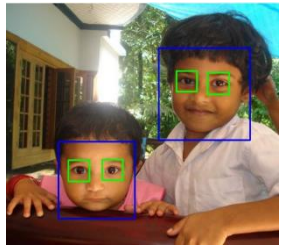
SIFT Feature Detection

Contour Detection



Corner Detection

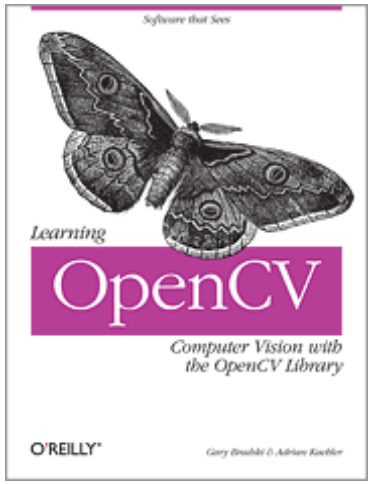
Optical Flow Detection

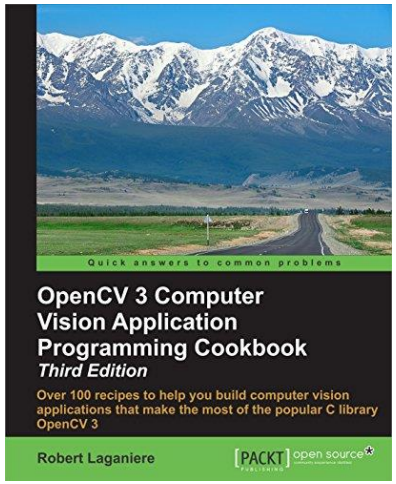


Multi Object Cascade Classifier

OpenCV

- Download OpenCV
  - http://opencv.org
- Online reference
  - http://docs.opencv.org
- Two books

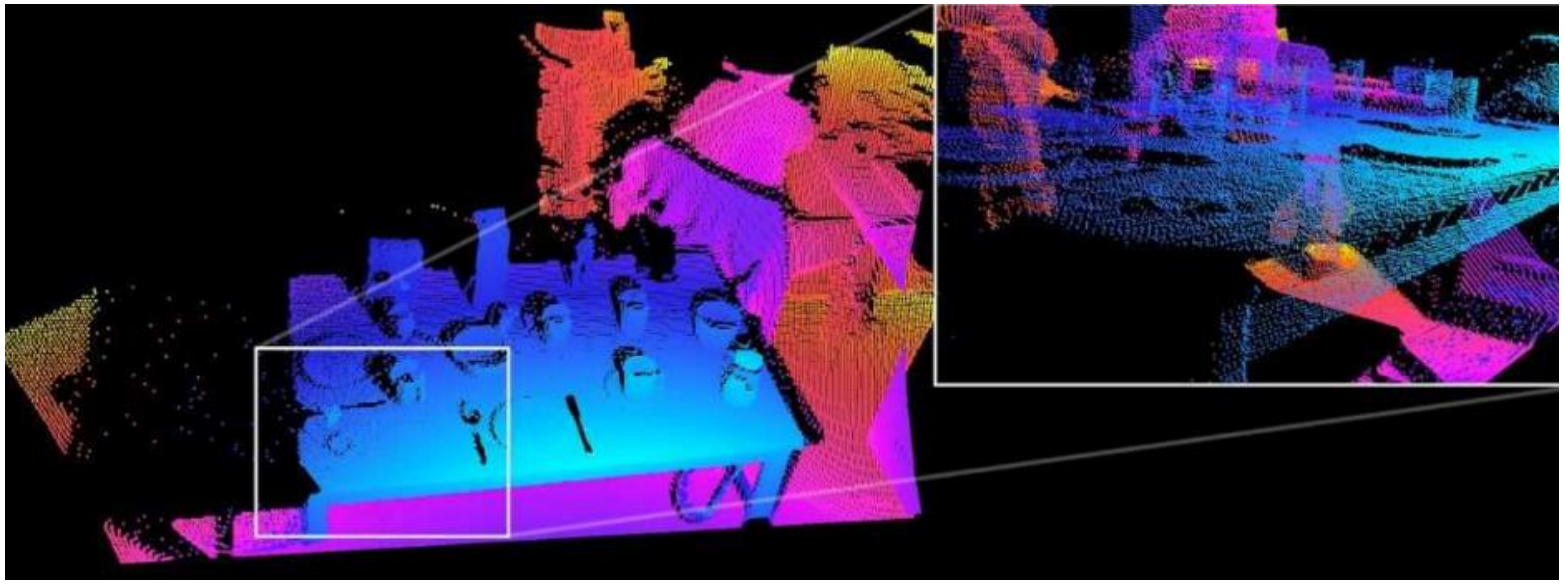




PCL

- **PCL** stands for **P**oint **C**loud **L**ibrary
- PCL is free for commercial and research use.
- PCL is financially supported by Willow Garage, NVidia, and Google.
- Download PCL
  - https://github.com/PointCloudLibrary/pcl
- Online documentation and tutorials
  - http://www.pointclouds.org/documentation/

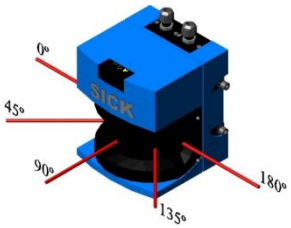
PCL



- Point Cloud = a "*cloud*" (i.e., collection) of nD points (usually $n = 3$)
- $p_i = \{x_i, y_i, z_i\} \rightarrow P = \{p_1, p_2, \dots, p_i, \dots, p_n\}$
- A point cloud P is used to represent 3D information about the world
- Besides XYZ data, each point p can hold additional information, e.g. RGB color

PCL

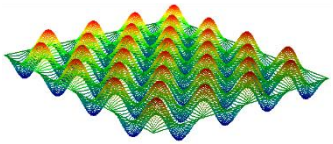
- Where do all cloud points come from?
  - Laser scans (high quality but expensive)



  - Stereo cameras (passive & past but dependent on texture)
  - Time of flight cameras (accurate but have low resolutions)

  - Kinect (cheap!)

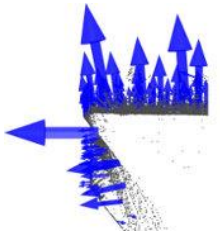
  - Simulation



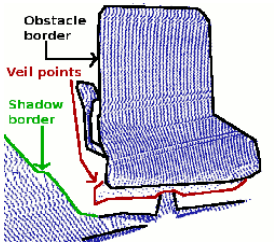
  - …

PCL

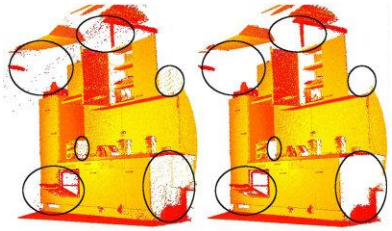
- The PCL framework contains numerous state-of-the art algorithms



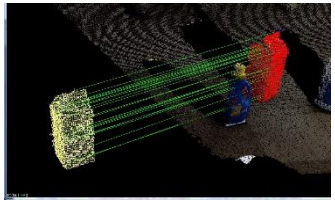
Surface Normal Estimation



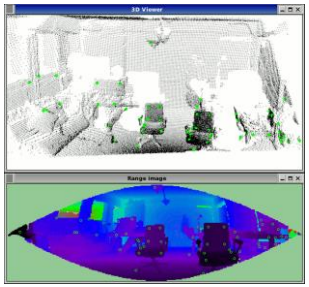
Border Extraction



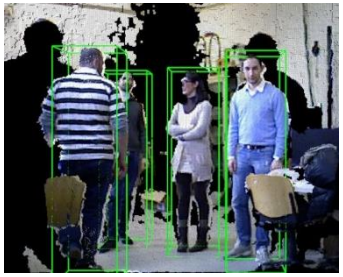
Filtering



Object Segmentation and Recognition

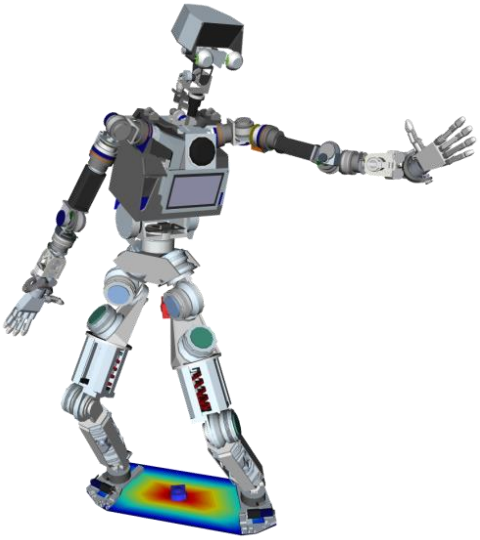


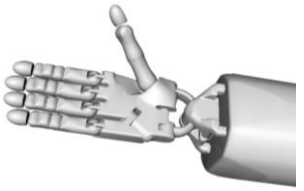
Feature Extraction



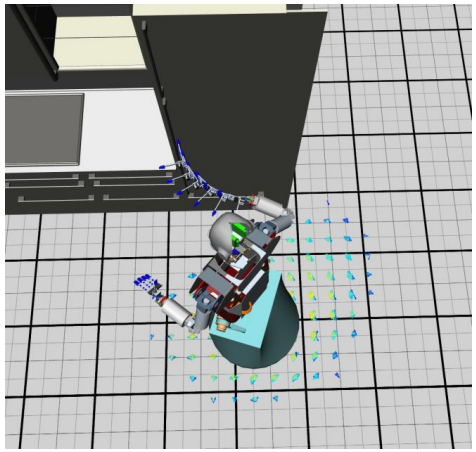
People Detection

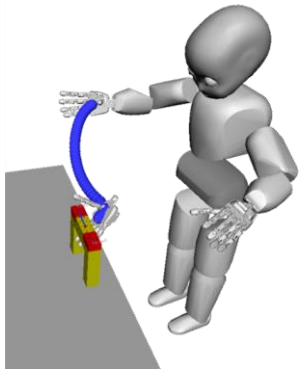
Simox - Overview

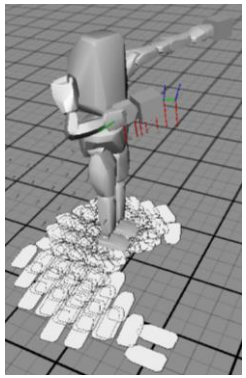




- **Developed at H²T**
  - Open Source (LGPL)
  - C++
  - Robot Independent:
    ARMAR-III, ARMAR-4, iCub, ….
- **Structure**
  - *VirtualRobot*: Kinematic simulation of complex (multi) robot systems
  - *Saba*: Motion Planning
  - *GraspStudio*: Grasp Planning
  - *SimDynamics*: Dynamics Simulation
- **Sources**
  - *https://gitlab.com/Simox/simox*
- **Wiki**
  - *https://gitlab.com/Simox/simox/wikis/home*

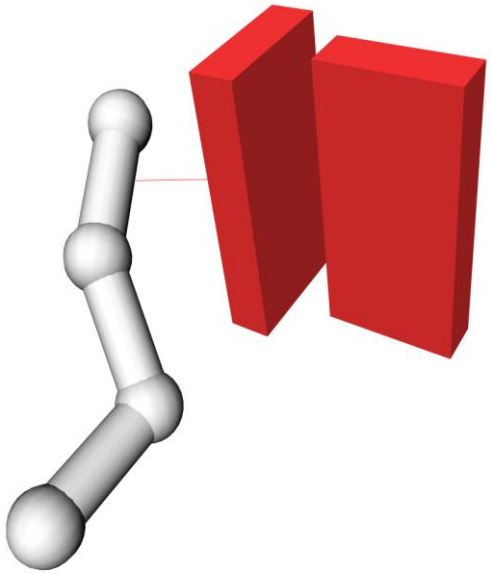




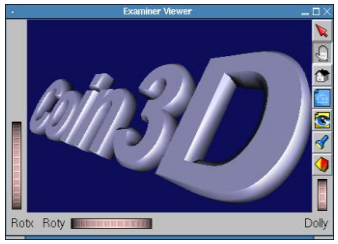


Virtual Robot – Interfaces

- **Collision Engine support**
  - Built-in support for PQP
  - Polygon / Triangle soups
  - Collision & Distance Queries
  - Extendable



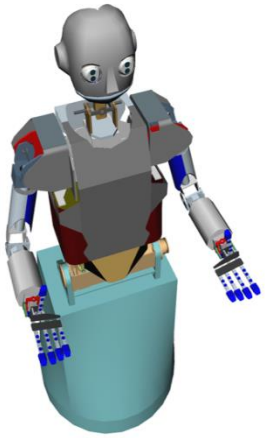
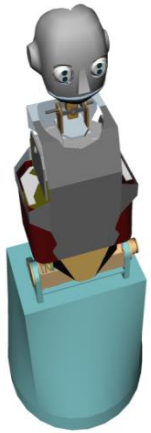
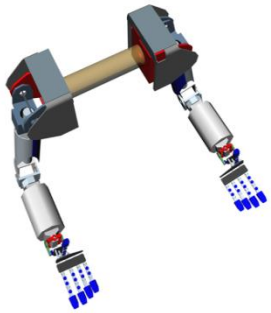
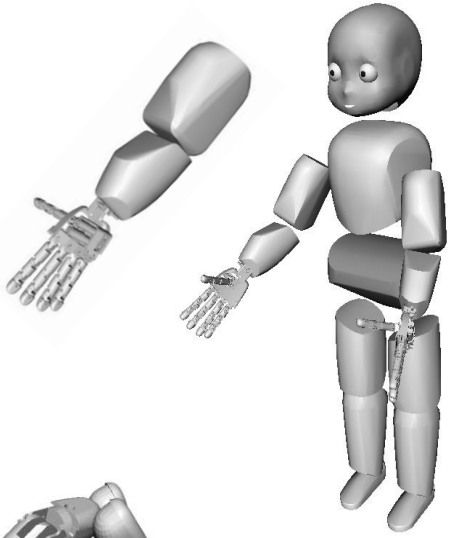
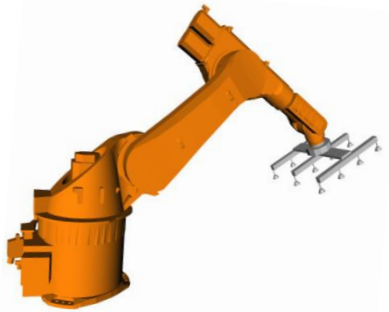
- **Visualization Support**
  - Interfaces for 3D model loading & visualization
  - Support for OpenSceneGraph and Coin3D
    WiP: Ogre engine



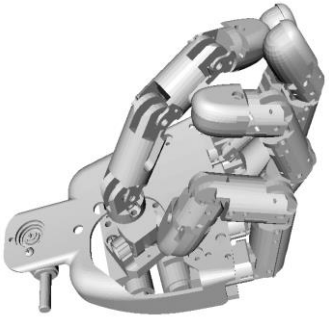
OpenSceneGraph

Virtual Robot – Structured Robots

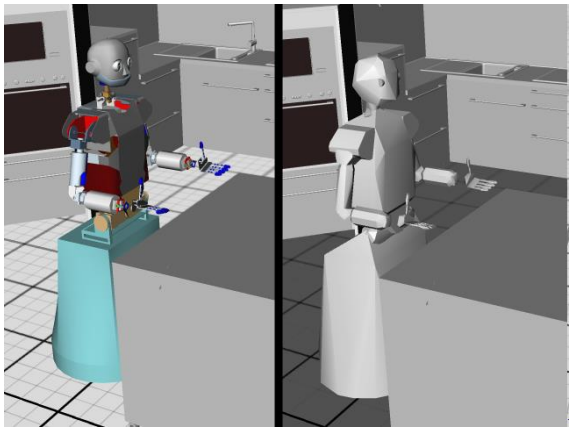
- Logical sets of kinematic sub-parts

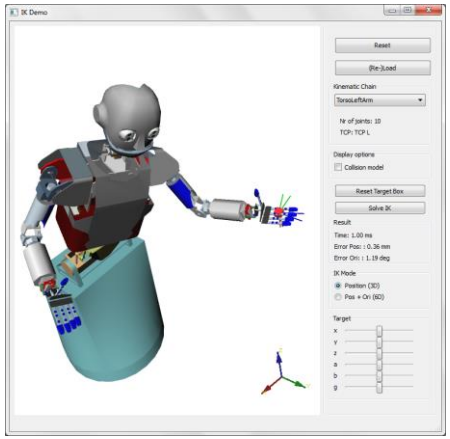
- End-Effector Definitions



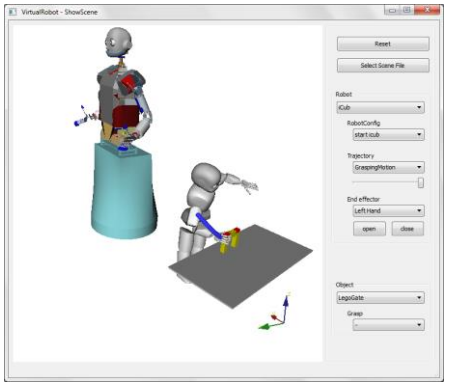
- 3D models
  - HD for visualization
  - simplified versions for collision detection (optional)
- XML/Collada/URDF Import



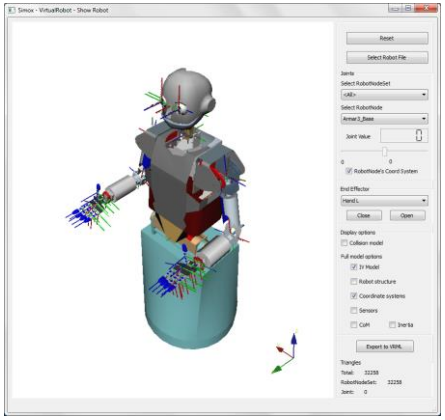
VirtualRobot - Examples



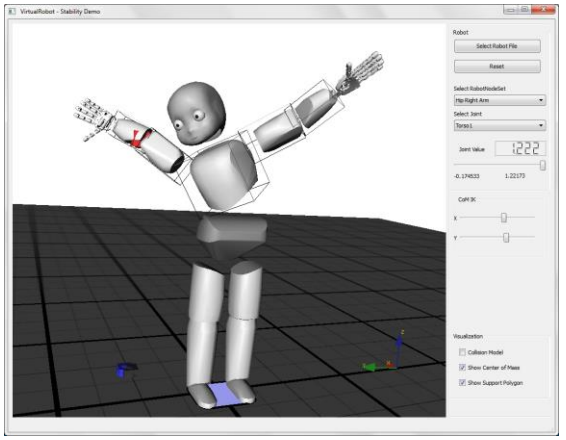
The *GenericIKDemo*



A scene in the *SimoxSceneViewer* tool



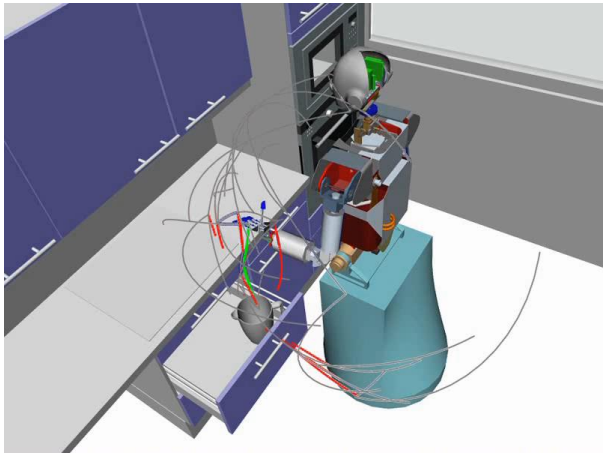
The *RobotViewer* tool can be used to inspect robot models

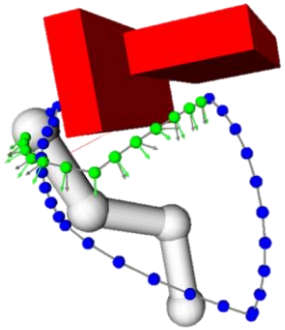


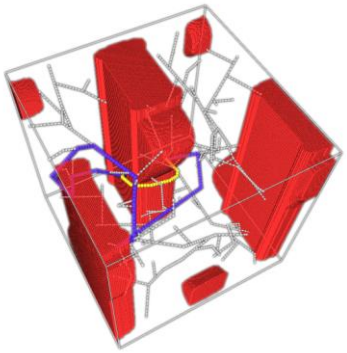
The *StabilityDemo*

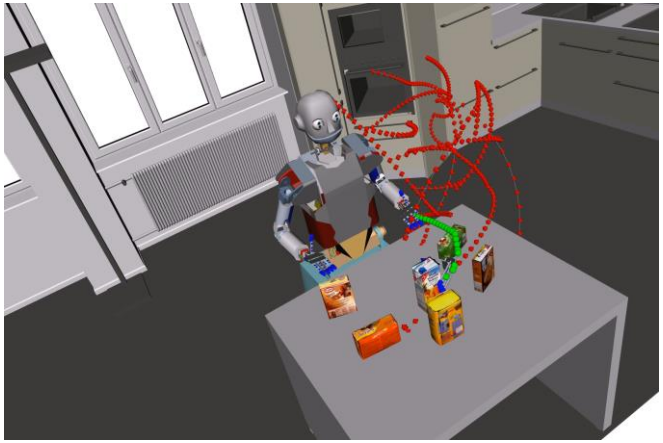
Motion Planning – Basic Elements

- **C-Space**
  - Defined for a *RobotNodeSet* (joints that move)
  - Setup of Collision Detection -> valid/invalid areas
- **Motion Planners**
  - Several RRT variants
- **Multithreading Support**
  - *PlanningThreads* can be used to plan in parallel
- **Path Processing**
  - Post processing of planned trajectories







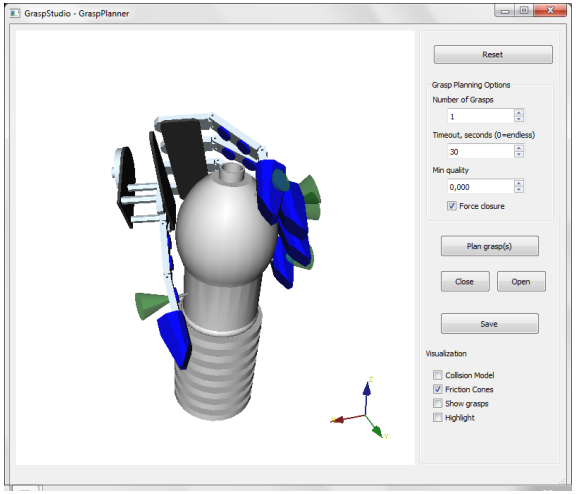


Grasp Planning

- Plan grasping configurations for robot hands / end effectors
- Measure the quality of grasps

  GraspQualityMeasureWrenchSpace
- Approach generation

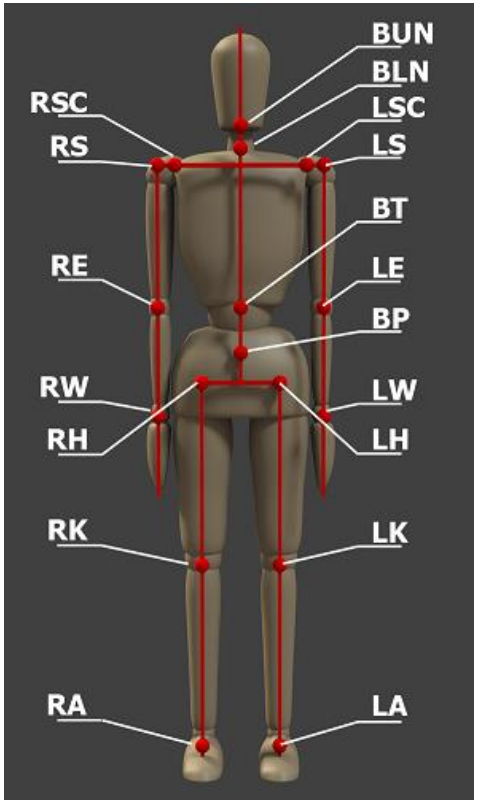
  ApproachMovementSurfaceNormal
- Grasp Planners
  - *GenericGraspPlanner*
    Can be configures with
    GraspQualities and ApproachGenerators
  - *MATGraspPlanner*
    uses the medial axis representation
    of objects (experimental)



The *GraspPlanner* demo

Master Motor Map - MMM

- Git repositories:
  - https://i61wiki.itec.uka.de/git/mmmcore.git
  - https://i61wiki.itec.uka.de/git/mmmtools.git
- Documentation, Setup and Installation
  - https://mmm.humanoids.kit.edu/
- Questions, Support:
  - Mailing List:
    h2t_mmm@lists.kit.edu

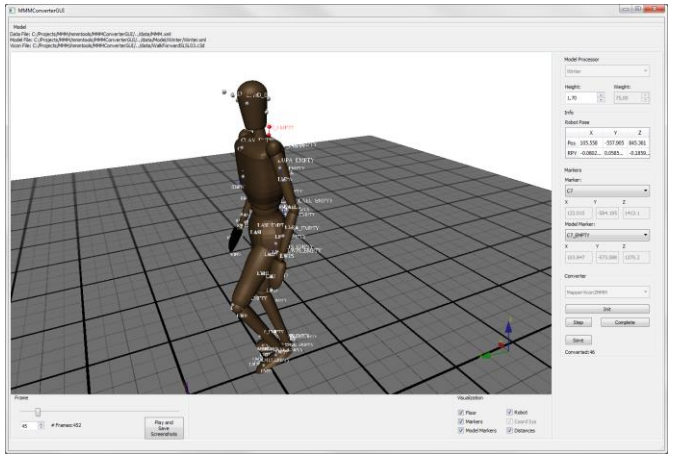


MMM – C++ API

- MMMCore
  - Specification and I/O
    - Models
    - Motion data (C3D)
    - MMM data (XML)
  - Converter Framework
  - Lightweight library, low number of dependencies
- MMMTools
  - Tools for
    - Visualization
    - Extended IK
    - Converter reference implementations
  - Viewers
  - Command line tools

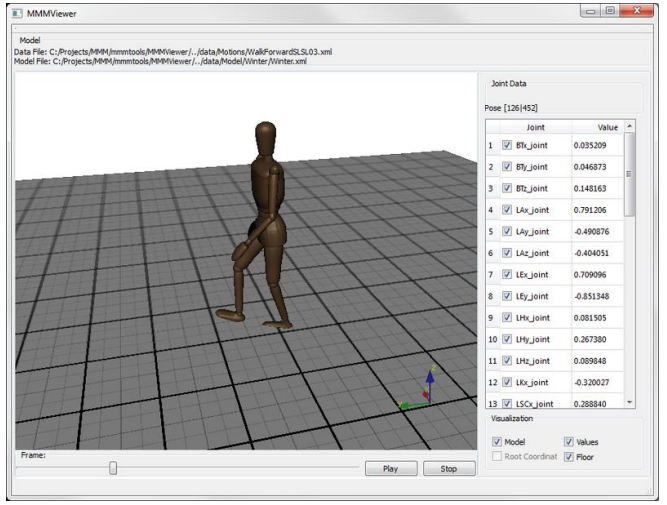
MMM – Tools

- MMMConverterGUI
  - GUI for marker based converters (e.g. VICON -> MMM)



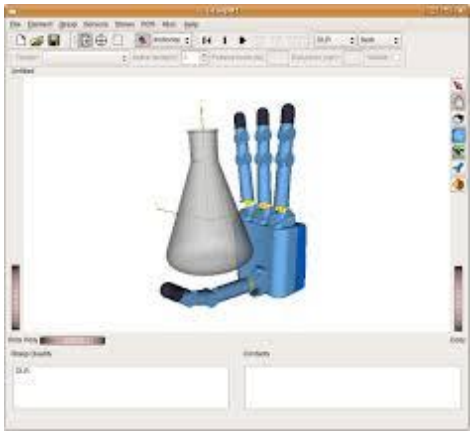
- MMMViewer
  - GUI for displaying mmm motions
  - Command line option:

    --motion <mmmFile.xml>



GraspIt!

- Grasp Planning Tool and Grasping Simulator
  - developed at Columbia University
  - C++ package with several hand models
- Main Page
  - http://graspit-simulator.github.io/
- Software
  - https://github.com/graspit-simulator



H²T Robot Editor

- Am H²T entwickeltes Plugin zur **Robotermodellierung** für die quelloffene 3D-Modellierungssoftware ***Blender***

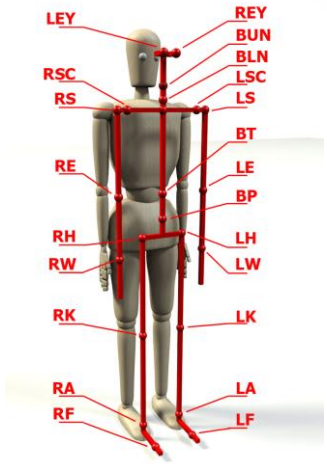
- **Erweiterung von Blender** mit domänenspezifischer Funktionalität:
  - Definition kinematischer Parameter, z.B. Gelenktypen-/winkelgrenzen, Segmente, kinematische Ketten usw.
  - Zuweisung von Mesh-Modellen zu Segmenten der Kinematik
  - Definition dynamischer Parameter, z.B. Segment-Massenschwerpunkte und –trägheitstensoren
  - Export nach Simox-Roboterformat und COLLADA 1.5
  - Import von Simox-Roboterformat
  - Laden von Bewegungen im XML-basierten MMM-Format

- Selbst unter **Open-Source-Lizenz** (GPLv3) veröffentlicht:
  `https://gitlab.com/h2t/roboteditor`

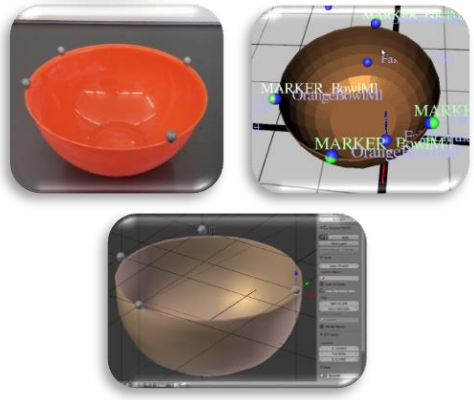
H²T Robot Editor

- Modellierung von **Robotern**:
  Einsatz zur Modellierung zahlreicher Roboter, z.B. ARMAR-3/4, YouBot, usw.

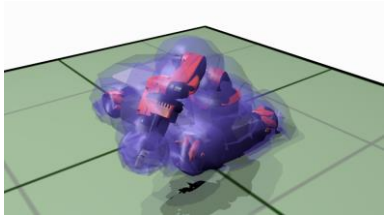
- Modellierung des **Menschen**:
  MMM-Modell als Referenzmodell des menschlichen Körpers mit 104 DoF



- Modellierung von **Objekten**:
  Unterstützung für virtuelle Marker ermöglicht Rekonstruktion der Objektpose mit Motion Capture



- Erstellung von **Kollisionsmodellen** und **fotorealistisches Rendering** von Robotermodellen und -bewegungsabläufen

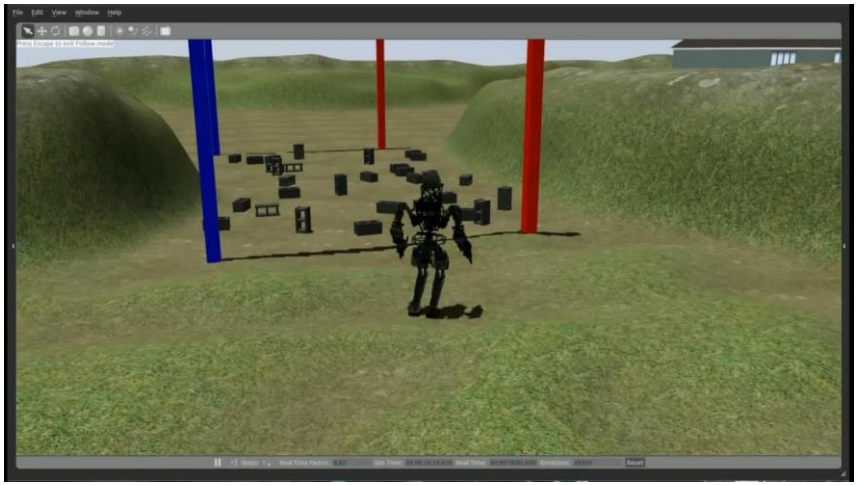


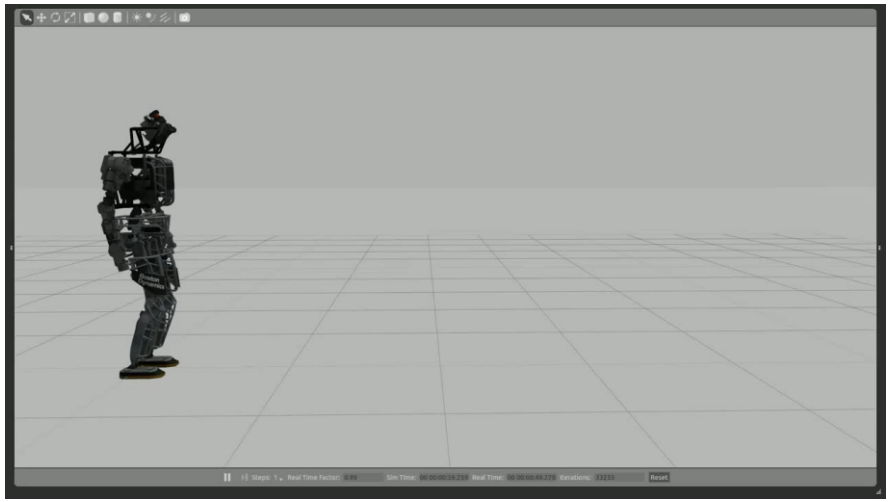
Gazebo Robotics Simulator

- **Robotics Simulator**
  - Open source project
  - Web: http://gazebosim.org/
  - Heavily used by the ROS community
- **Features**
  - **Dynamics Simulation**
    Access multiple physics engines including ODE, Bullet, Simbody, and DART.
  - **Cloud Simulation**
    run Gazebo on Amazon, Softlayer
  - **Plug-in based architecture**
    Plugins provide direct access to Gazebo's API.
  - **Advanced Visualization Features**
    Ogre Game Engine

Gazebo Robotics Simulator

- Examples





V-Rep Robotics Simulator

Robotics Simulator

- Web: http://www.coppeliarobotics.com
- Commercial simulator, but free educational license

- **Features**
  - **Dynamics Simulation**
    Access multiple physics engines including
    ODE, Bullet, Vortex, Newton
  - **Powerful API**
    - C++, Lua, Java, Matlab, …
    - Remote Access
  - **Plug-in based architecture**
  - **Scripting**
    Scripts can be attached to any objects in a scene

V-Rep Robotics Simulator

- Examples