Visual Navigation for Flying Robots

Welcome

Dr. Jürgen Sturm

Organization

- Tue 10:15-11:45
  - Lectures, discussions
  - Lecturer: Jürgen Sturm
- Thu 14:15-15:45
  - Lab course, homework & programming exercises
  - Teaching assistant: Nikolas Engelhard
- Course website
  - Dates, additional material
  - Exercises, deadlines
  - http://cvpr.in.tum.de/teaching/ss2012/visnav2012

Who are we?

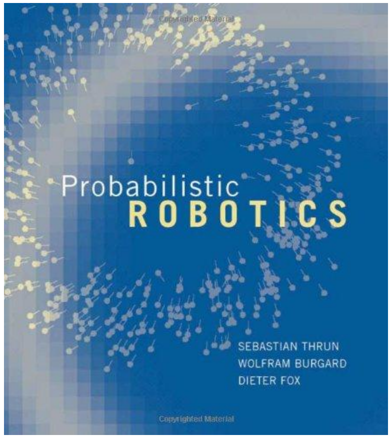
- Computer Vision group:
  1 Professor, 2 Postdocs, 7 PhD students
- Research topics:
  Optical flow and motion estimation, 3D reconstruction, image segmentation, convex optimization
- My research goal:
  Apply solutions from computer vision to real-world problems in robotics.

Goal of this Course

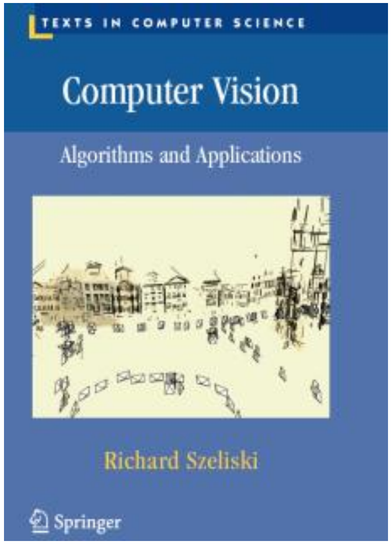
- Provide an overview on problems/approaches for autonomous quadrocopters
- Strong focus on vision as the main sensor
- Areas covered: Mobile Robotics and Computer Vision
- Hands-on experience in lab course

Course Material

- Probabilistic Robotics. Sebastian Thrun, Wolfram Burgard and Dieter Fox. MIT Press, 2005.



- Computer Vision: Algorithms and Applications. Richard Szeliski. Springer, 2010. http://szeliski.org/Book/



Lecture Plan

1. Introduction
2. Robots, sensor and motion models
3. State estimation and control
4. Guest talks
5. Feature detection and matching
6. Motion estimation
7. Simultaneous localization and mapping
8. Stereo correspondence
9. 3D reconstruction
10. Navigation and path planning
11. Exploration
12. Evaluation and Benchmarking

Basics on mobile robotics (1–4)

Camera-based localization and mapping (5–9)

Advanced topics (10–12)

Lab Course

- Thu 14:15 – 15:45, given by Nikolas Engelhard
  - Exercises: room 02.09.23
    (6x, obliged, homework discussion)
  - Robot lab: room 02.09.34/36
    (in weeks without exercises, in case you need help, recommended!)

Exercises Plan

- Exercise sheets contain both theoretical and programming problems
- 3 exercise sheets + 1 mini-project
- Deadline: before lecture (Tue 10:15)
- Hand in by email (visnav2012@cvpr.in.tum.de)

Group Assignment and Schedule

- 3 Ardrones (red/green/blue) + Joystick + 2x Batteries + Charger + PC
- 20 students in the course, 2-3 students per group → 7-8 groups
- Either use lab computers or bring own laptop (recommended)
- Will put up lists for groups and robot schedule in robot lab (room 02.09.36)

VISNAV2012: Team Assignment

Team Name				
Student Name				
Student Name				
Student Name				

Team Name				
Student Name				
Student Name				
Student Name				

VISNAV2012: Robot Schedule

- Each team gets one time slot with programming support
- The robots/PCs are also available during the rest of the week (but without programming support)

	Red	Green	Blue
Thu 2pm – 3pm			
Thu 3pm – 4pm			
Thu 4pm – 5pm			

Safety Warning

- Quadrocopters are dangerous objects
- Read the manual carefully before you start
- Always use the protective hull
- If somebody gets injured, report to us so that we can improve safety guidelines
- If something gets damaged, report it to us so that we can fix it
- **NEVER TOUCH THE PROPELLORS**
- **DO NOT TRY TO CATCH THE QUADROCOPTER WHEN IT FAILS – LET IT FALL/CRASH!**

Agenda for Today

- History of mobile robotics
- Brief intro on quadrocopters
- Paradigms in robotics
- Architectures and middleware

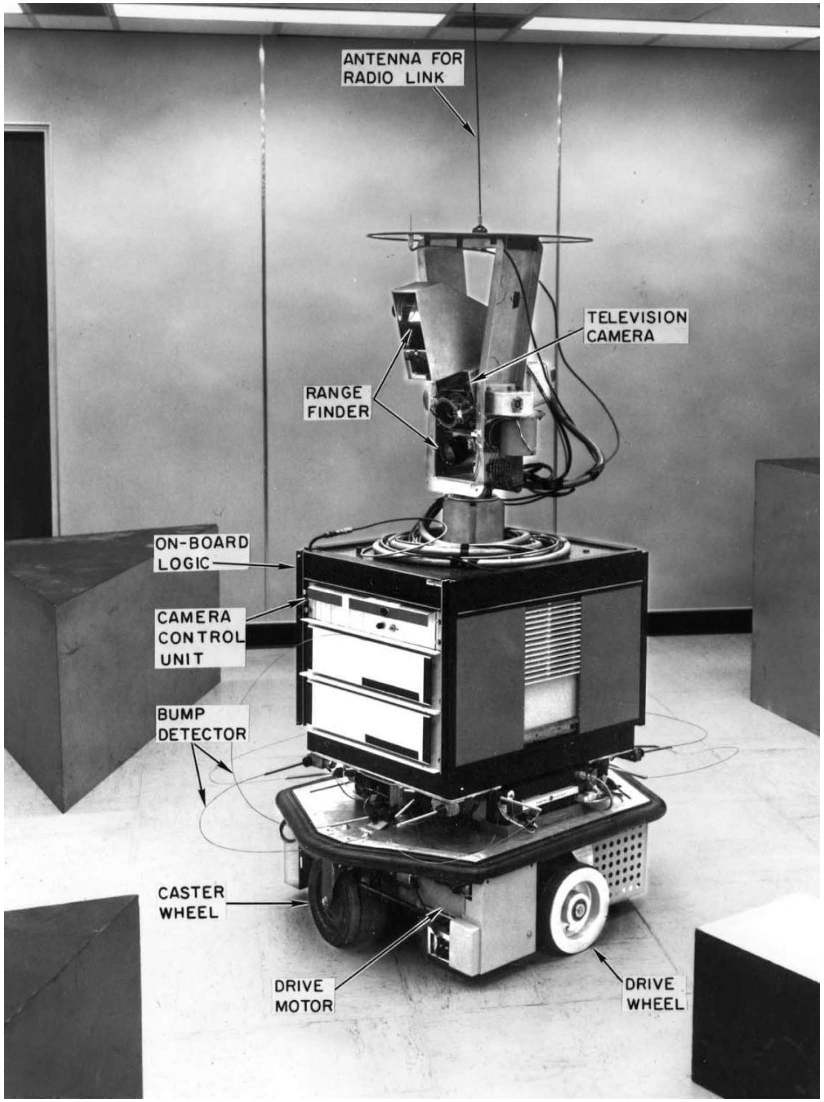
General background

- Autonomous, automaton
  - self-willed (Greek, auto+matos)
- Robot
  - Karel Capek in 1923 play R.U.R. (Rossum's Universal Robots)
  - labor (Czech or Polish, robota)
  - workman (Czech or Polish, robotnik)

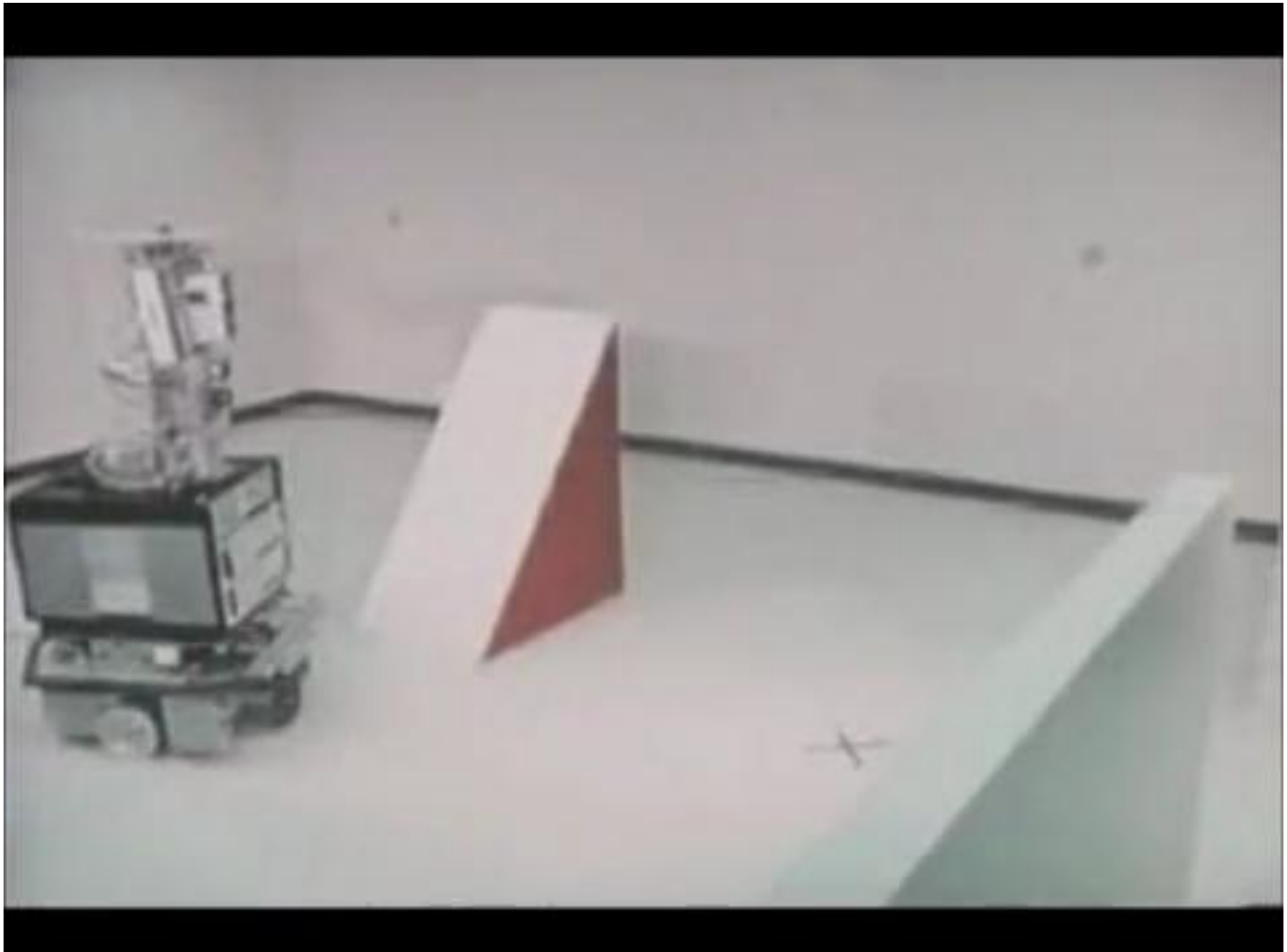
History

In 1966, Marvin Minsky at MIT asked his undergraduate student Gerald Jay Sussman to "spend the summer linking a camera to a computer and getting the computer to describe what it saw". We now know that the problem is slightly more difficult than that. (Szeliski 2009, Computer Vision)

Shakey the Robot (1966-1972)



Shakey the Robot (1966-1972)



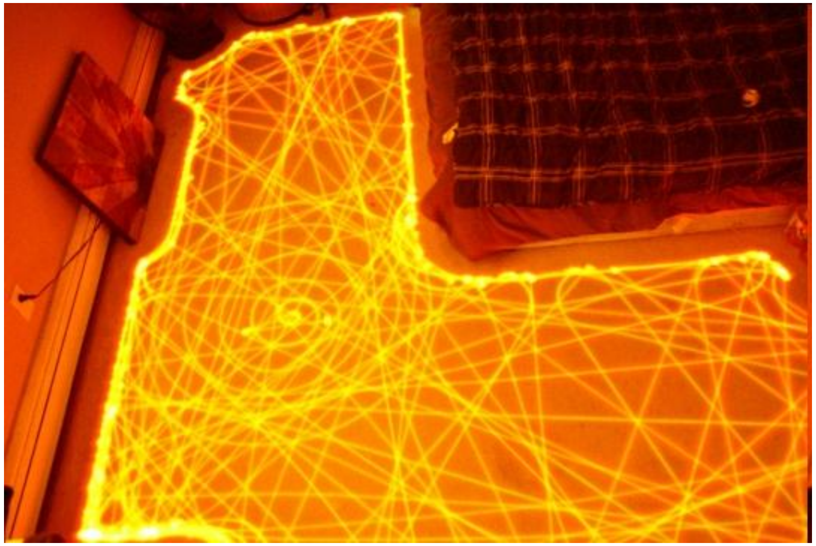
Stanford Cart (1961-80)

Rhino and Minerva (1998-99)

- Museum tour guide robots
- University of Bonn and CMU
- Deutsches Museum, Smithsonian Museum

Roomba (2002)

- Sensor: one contact sensor
- Control: random movements
- Over 5 million units sold



Neato XV-11 (2010)

- Sensors:
  - 1D range sensor for mapping and localization
  - Improved coverage

Darpa Grand Challenge (2005)

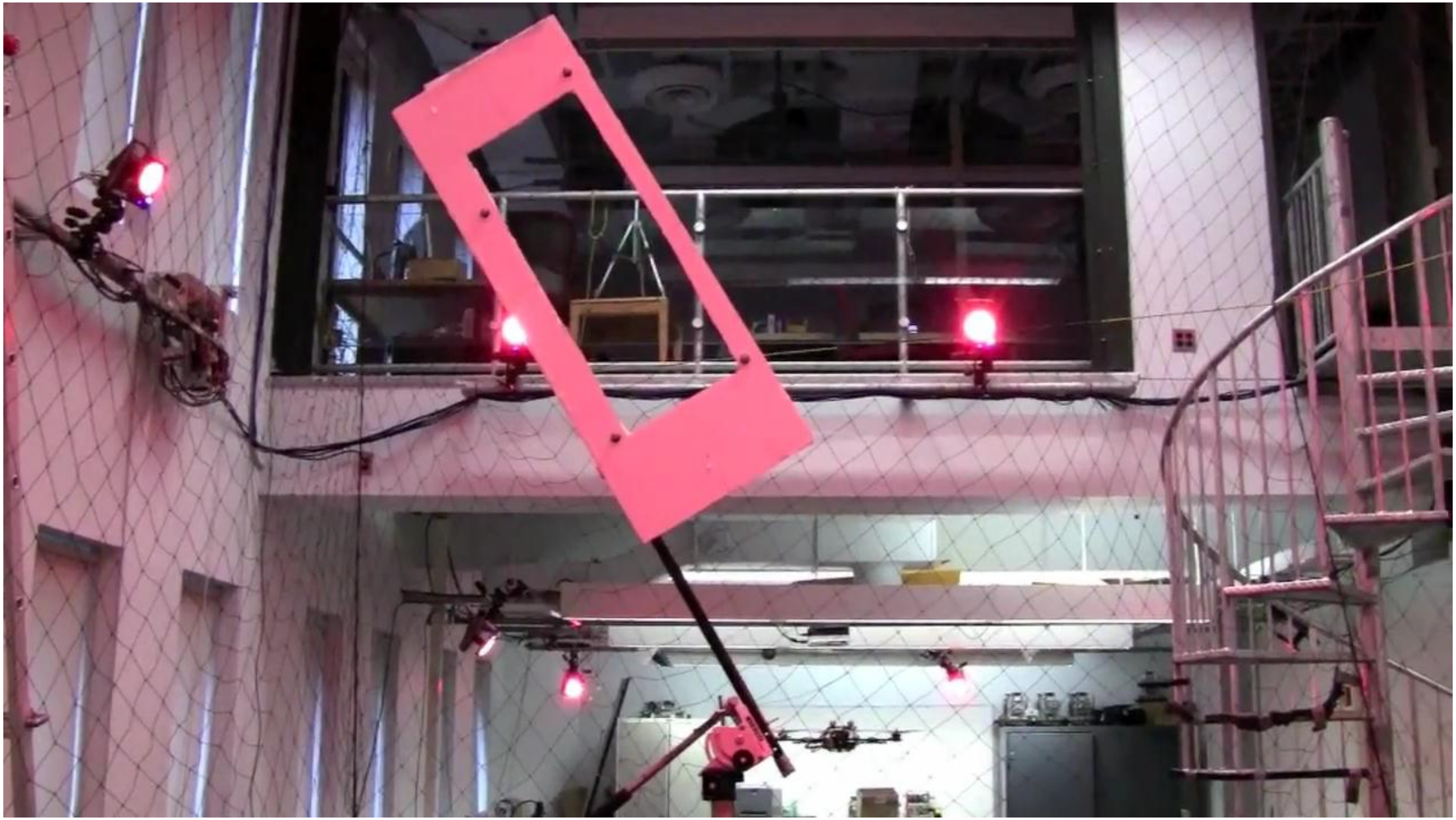
Kiva Robotics (2007)

- Pick, pack and ship automation

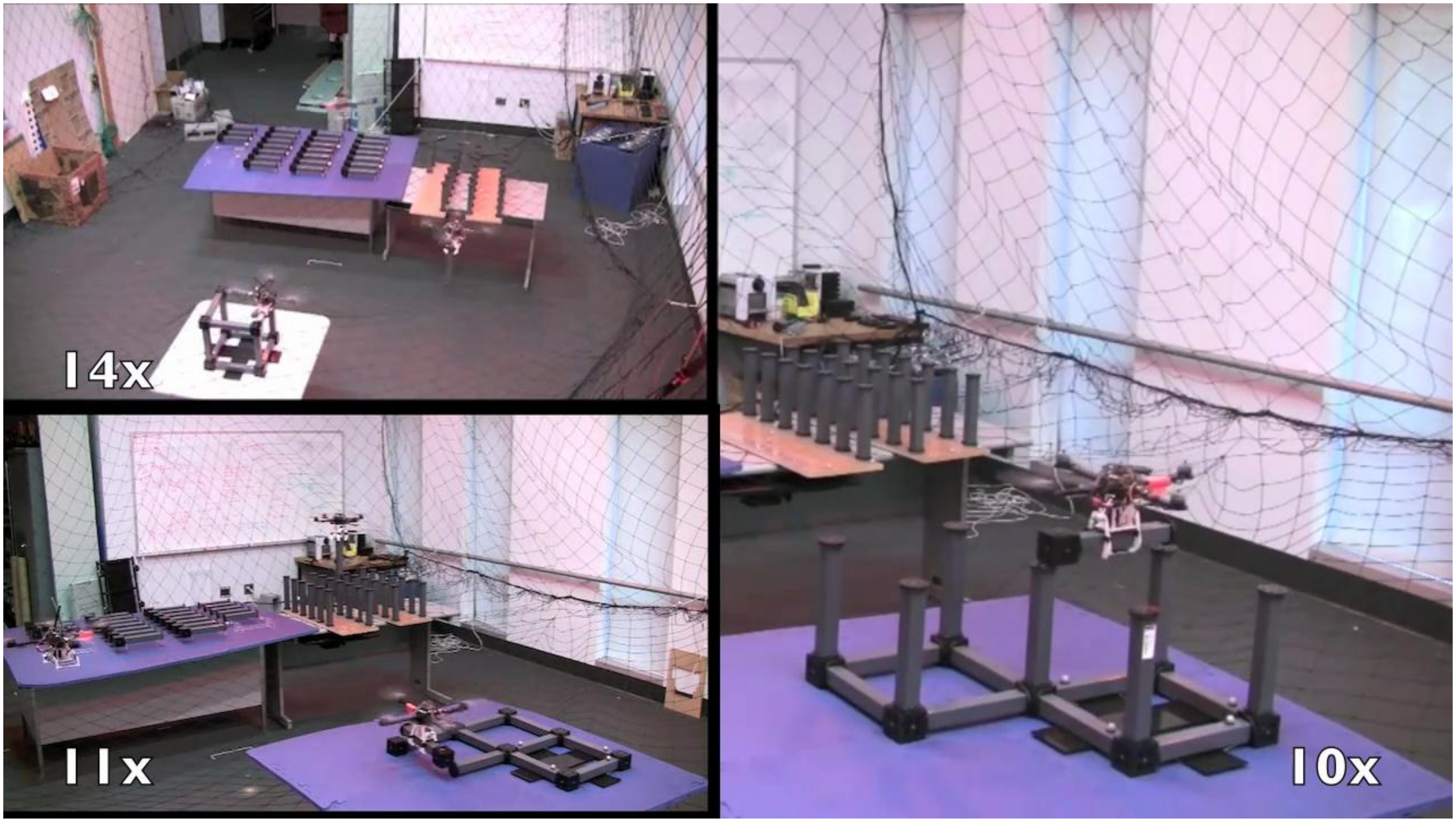
Fork Lift Robots (2010)

Quadrocopters (2001-)

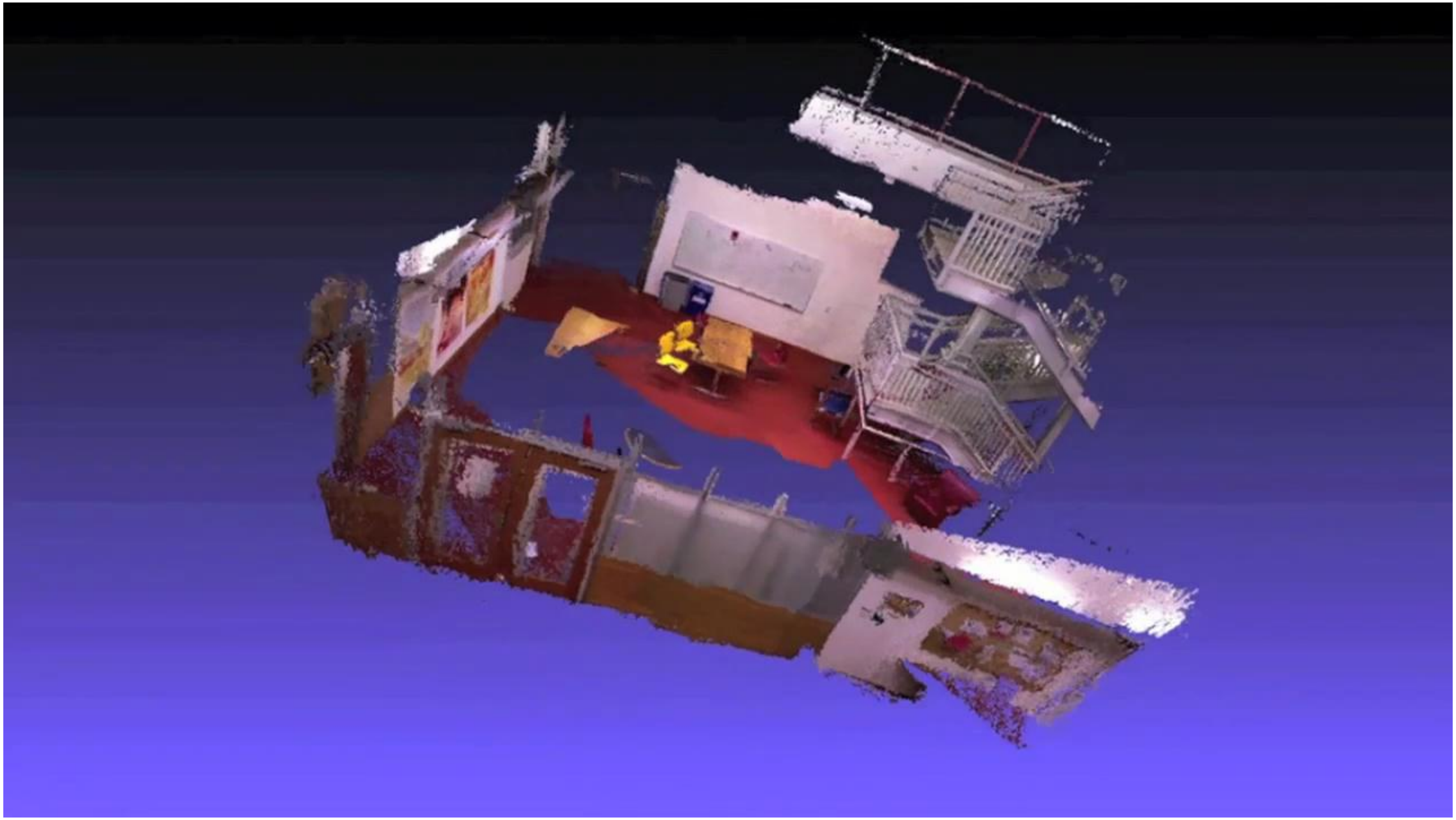
Aggressive Maneuvers (2010)



Autonomous Construction (2011)



Mapping with a Quadrocopter (2011)



Our Own Recent Work (2011-)

- RGB-D SLAM (Nikolas Engelhard)
- Visual odometry (Frank Steinbrücker)
- Camera-based navigation (Jakob Engel)

Current Trends in Robotics

Robots are entering novel domains

- Industrial automation
- Domestic service robots
- Medical, surgery
- Entertainment, toys
- Autonomous cars
- **Aerial monitoring/inspection/construction**

Flying Robots

- Recently increased interest in flying robots
  - Shift focus to different problems (control is much more difficult for flying robots, path planning is simpler, …)
- Especially quadrocopters because
  - Can keep position
  - Reliable and compact
  - Low maintenance costs
- Trend towards miniaturization

Application Domains of Flying Robots

- Stunts for action movies, photography, sportscasts
- Search and rescue missions
- Aerial photogrammetry
- Documentation
- Aerial inspection of bridges, buildings, …
- Construction tasks
- Military
- Today, quadrocopters are often still controlled by human pilots

Quadrocopter Platforms

- Commercial platforms
  - Ascending Technologies
  - Height Tech
  - Parrot Ardrone ⟵ **Used in the lab course**
  - …
- Community/open-source projects
  - Mikrokopter
  - Paparazzi
  - …

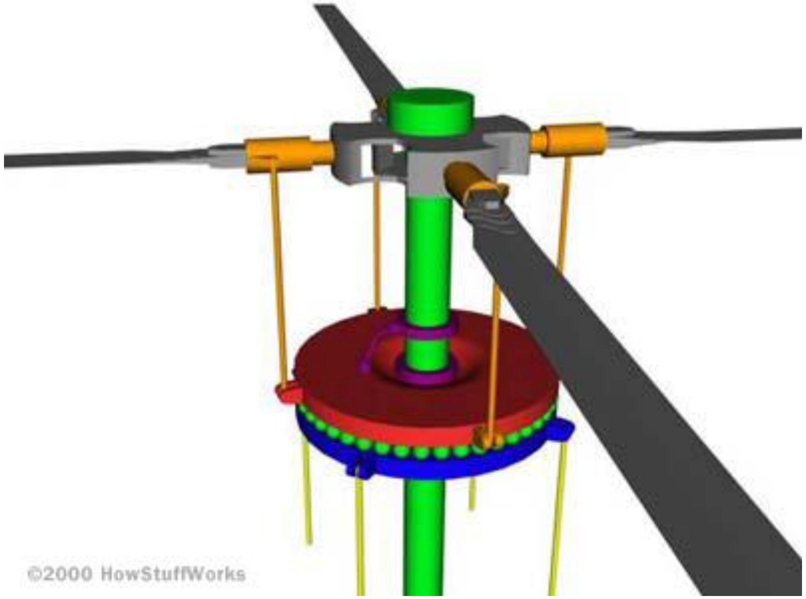
For more, see http://multicopter.org/wiki/Multicopter_Table

Flying Principles

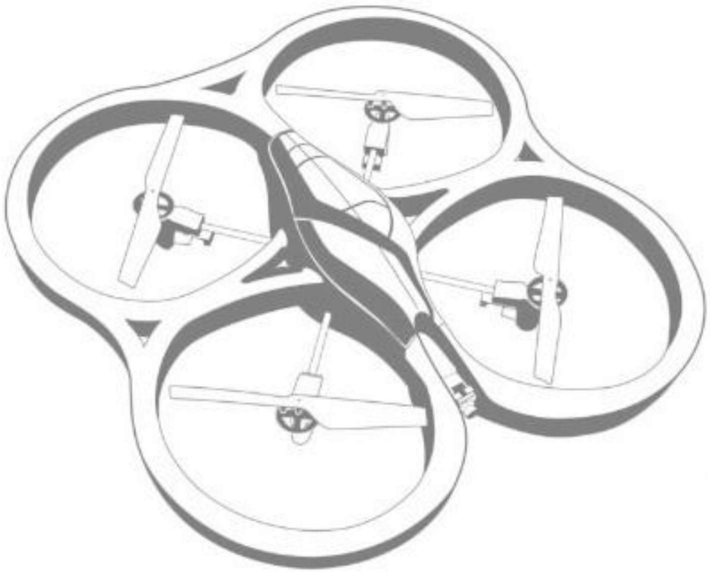
- Fixed-wing airplanes
  - generate lift through forward airspeed and the shape of the wings
  - controlled by flaps
- Helicopters/rotorcrafts
  - main rotor for lift, tail rotor to compensate for torque
  - controlled by adjusting rotor pitch
- Quadrocopter/quadrotor
  - four rotors generate lift
  - controlled by changing the speeds of rotation

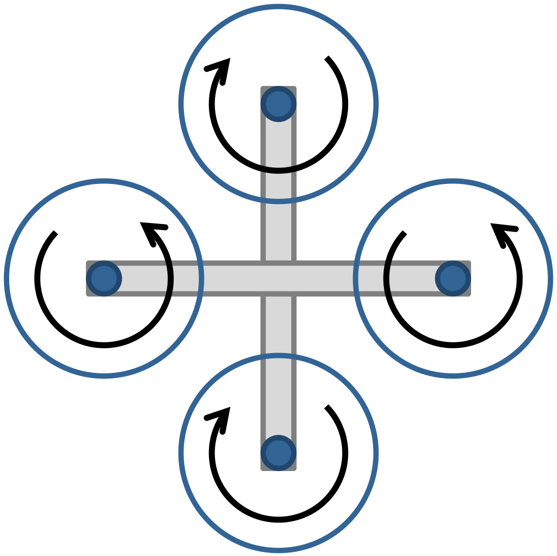
Helicopter

- Swash plate adjusts pitch of propeller cyclically, controls pitch and roll
- Yaw is controlled by tail rotor



Quadrocopter

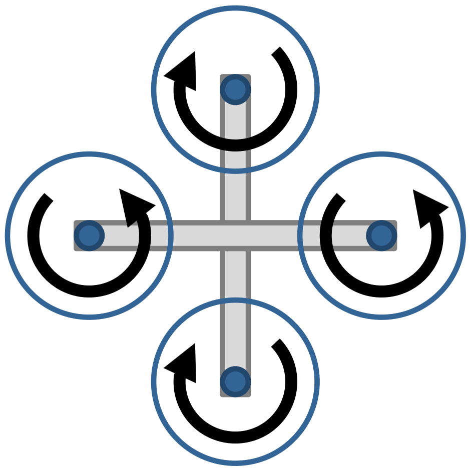




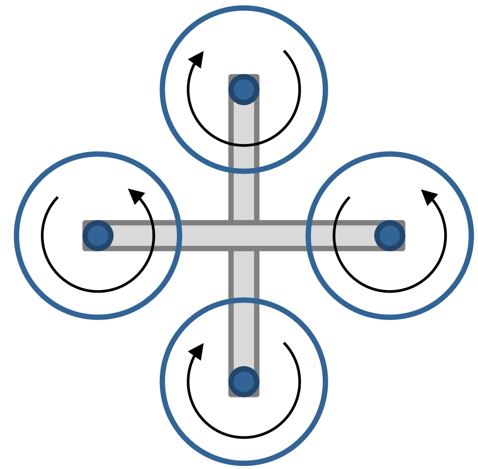
Keep position:

- Torques of all four rotors sum to zero
- Thrust compensates for earth gravity

Quadrocopter: Basic Motions

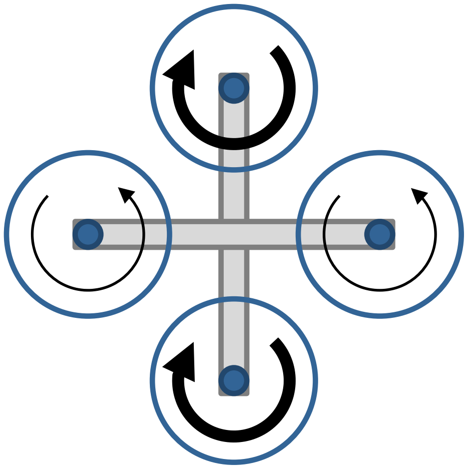


Ascend

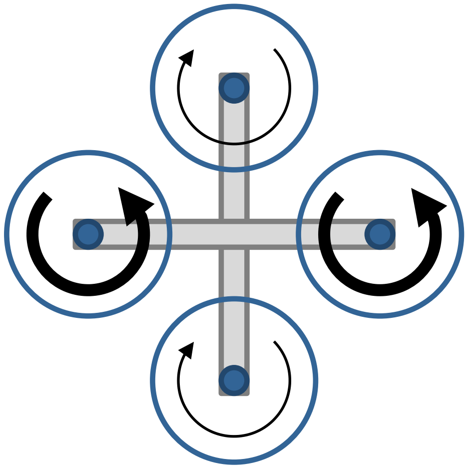


Descend

Quadrocopter: Basic Motions

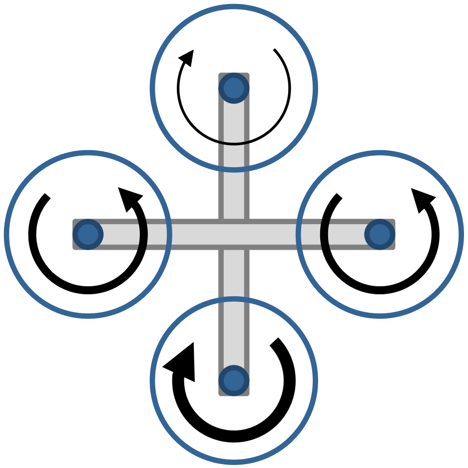


Turn Left

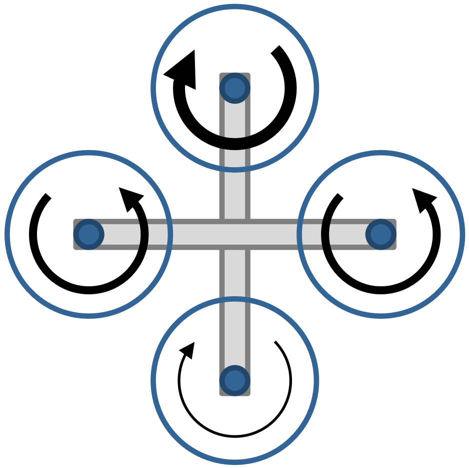


Turn Right

Quadrocopter: Basic Motions

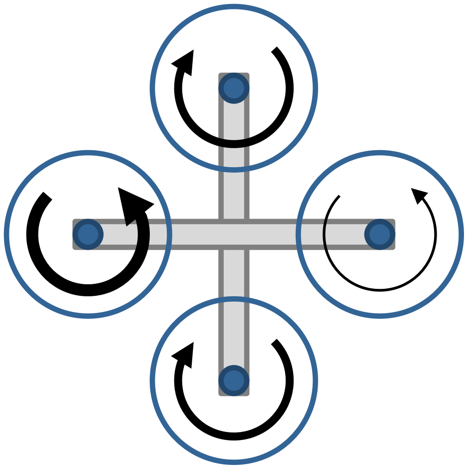


Accelerate Forward

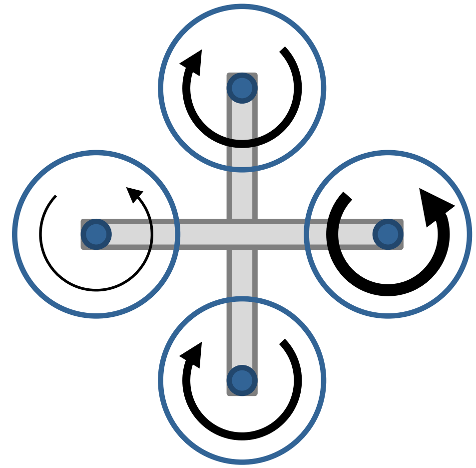


Accelerate Backward

Quadrocopter: Basic Motions



Accelerate
to the Right



Accelerate
to the Left

Autonomous Flight

- Low level control (not covered in this course)
  - Maintain attitude, stabilize
  - Compensate for disturbances
- High level control
  - Compensate for drift
  - Avoid obstacles
  - Localization and Mapping
  - Navigate to point
  - Return to take-off position
  - Person following

Challenges

- Limited payload
  - Limited computational power
  - Limited sensors
- Limited battery life
- Fast dynamics, needs electronic stabilization
- Quadrocopter is always in motion
- Safety considerations

Robot Ethics

- Where does the responsibility for a robot lie?
- How are robots motivated?
- Where are humans in the control loop?
- How might society change with robotics?
- Should robots be programmed to follow a code of ethics, if this is even possible?

Robot Ethics

Three Laws of Robotics (Asimov, 1942):

- A robot may not injure a human being or, through inaction, allow a human being to come to harm.
- A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.
- A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws.

Robot Design

Imagine that we want to build a robot that has to perform navigation tasks…

How would you tackle this?

- What hardware would you choose?
- What software architecture would you choose?

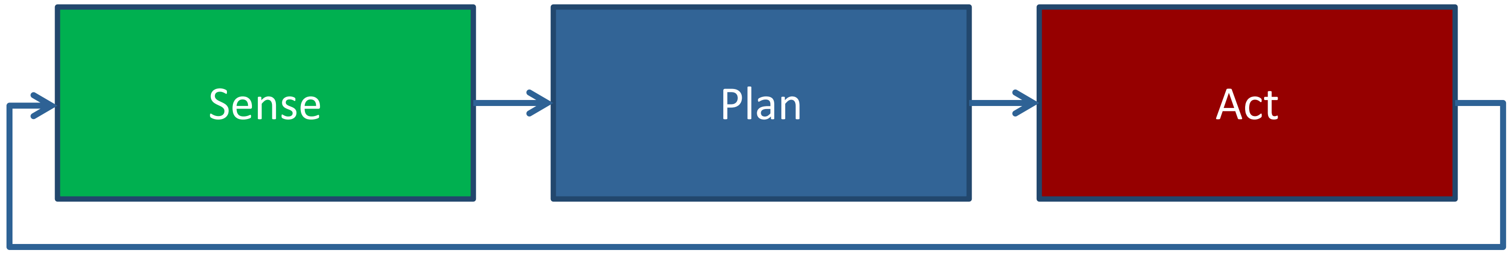
Robot Hardware/Components

- Sensors
- Actuators
- Control Unit/Software

Evolution of Paradigms in Robotics

- Classical robotics (mid-70s)
  - Exact models
  - No sensing necessary
- Reactive paradigms (mid-80s)
  - No models
  - Relies heavily on good sensing
- Hybrid approaches (since 90s)
  - Model-based at higher levels
  - Reactive at lower levels

Classical / hierarchical paradigm

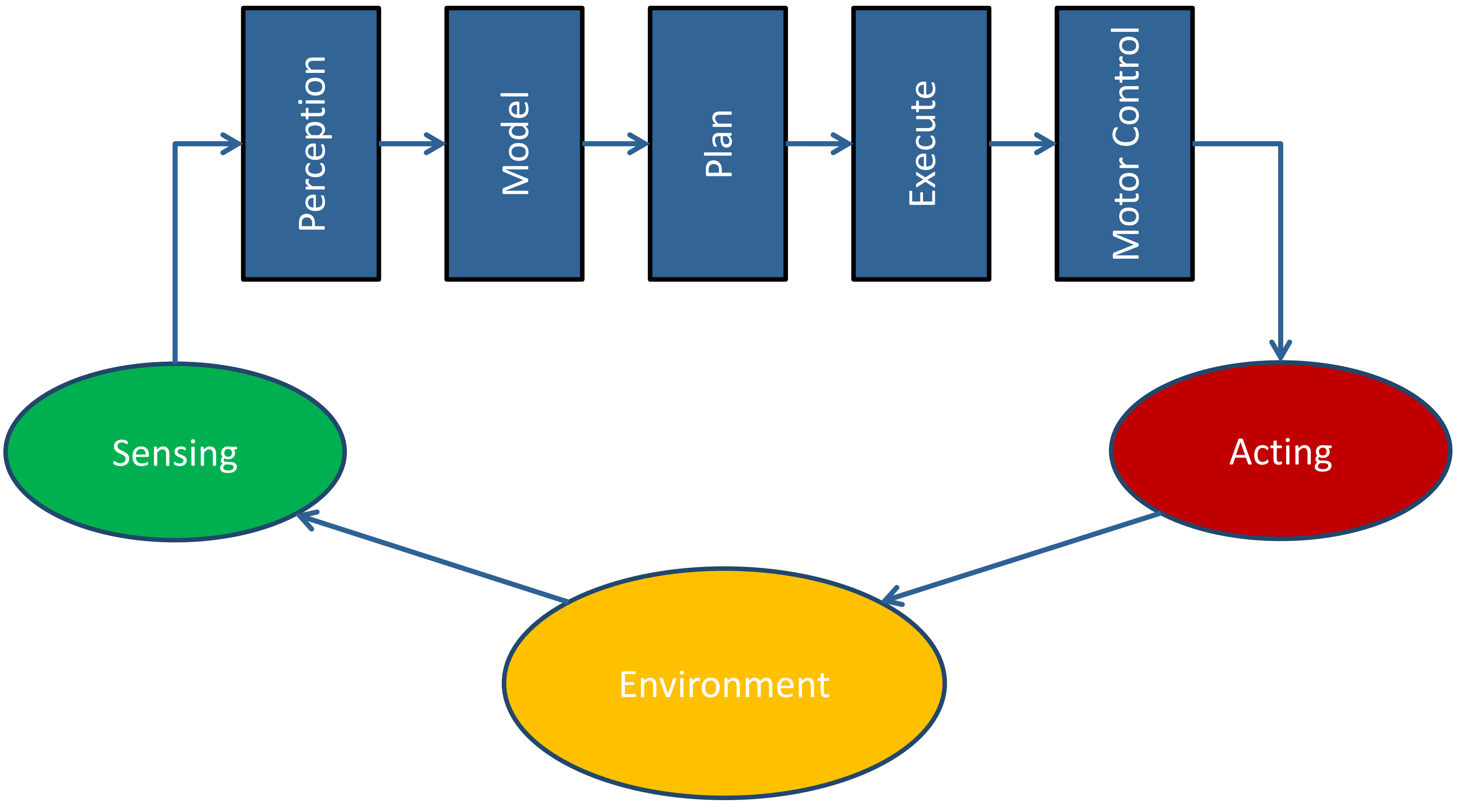


- Inspired by methods from Artificial Intelligence (70’s)
- Focus on automated reasoning and knowledge representation
- STRIPS (Stanford Research Institute Problem Solver): Perfect world model, closed world assumption
- Shakey: Find boxes and move them to designated positions

Classical paradigm: Stanford Cart

- Take nine images of the environment, identify interesting points, estimate depth
- Integrate information into global world model
- Correlate images with previous image set to estimate robot motion
- On basis of desired motion, estimated motion, and current estimate of environment, determine direction in which to move
- Execute motion

Classical paradigm as horizontal/functional decomposition

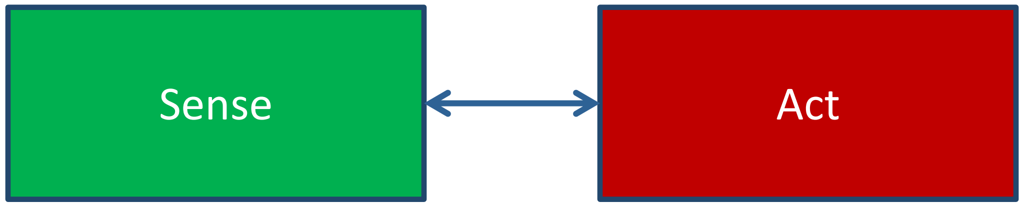


Characteristics of hierarchical paradigm

Good old-fashioned Artificial Intelligence (GOFAI):

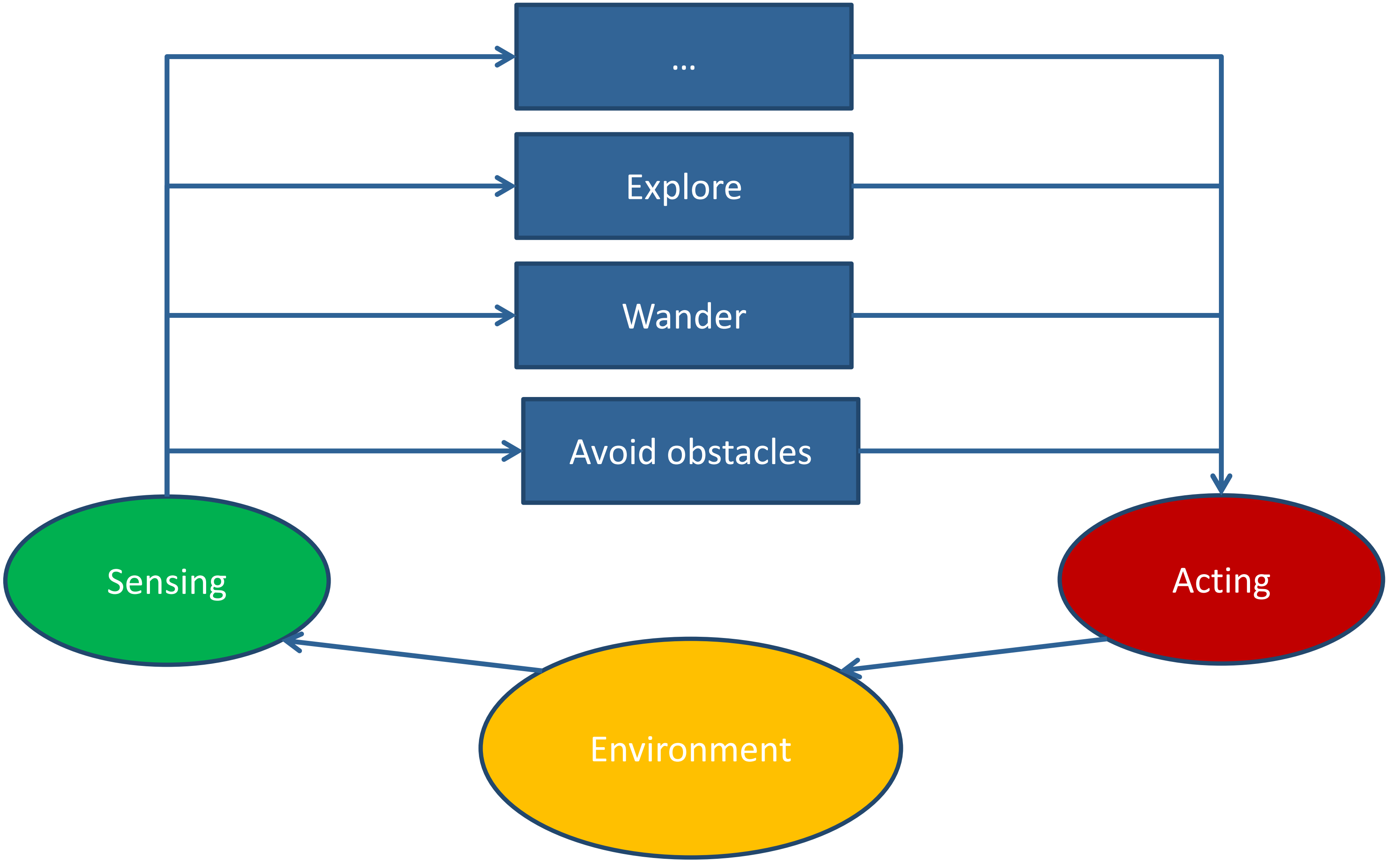
- Symbolic approaches
- Robot perceives the world, plans the next action, acts
- All data is inserted into a single, global world model
- Sequential data processing

Reactive Paradigm



- Sense-act type of organization
- Multiple instances of stimulus-response loops (called behaviors)
- Each behavior uses local sensing to generate the next action
- Combine several behaviors to solve complex tasks
- Run behaviors in parallel, behavior can override (subsume) output of other behaviors

Reactive Paradigm as Vertical Decomposition



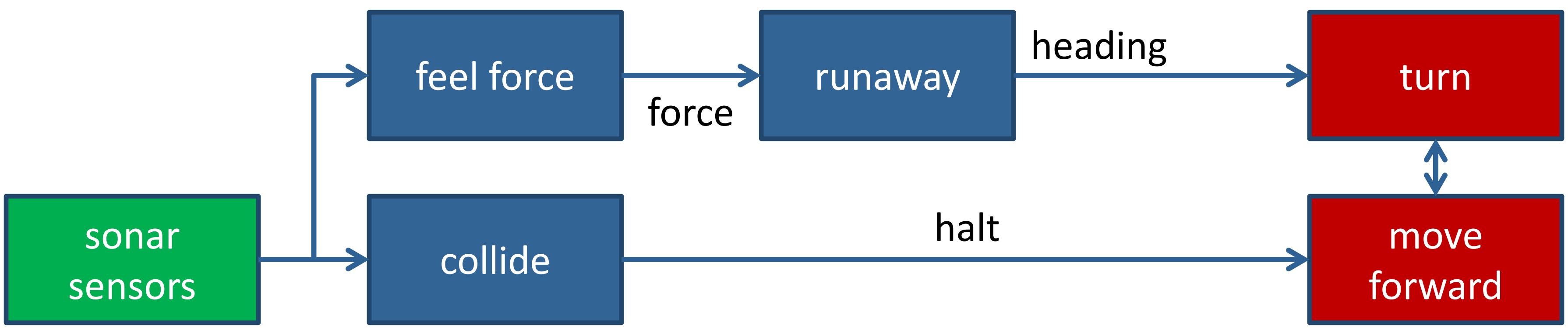
Characteristics of Reactive Paradigm

- Situated agent, robot is integral part of the world
- No memory, controlled by what is happening in the world
- Tight coupling between perception and action via behaviors
- Only local, behavior-specific sensing is permitted (ego-centric representation)

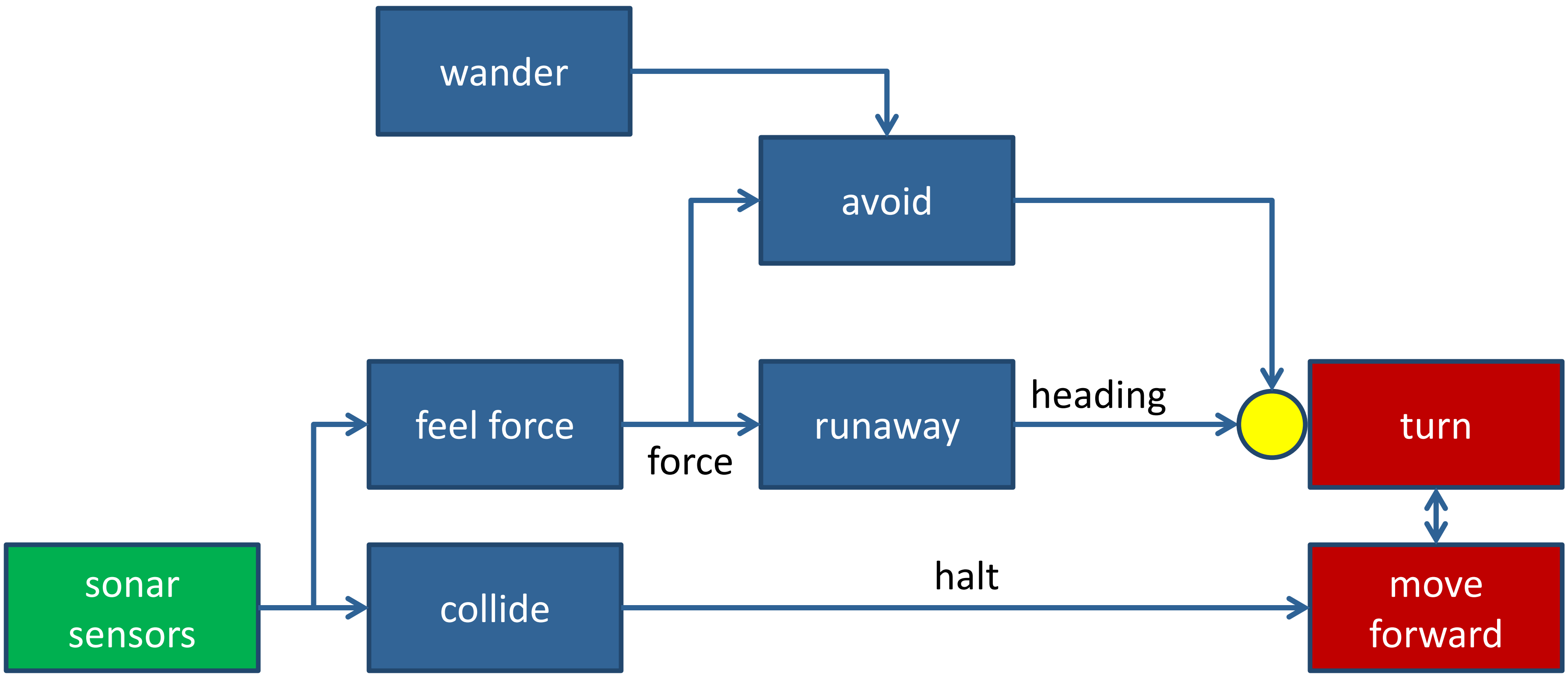
Subsumption Architecture

- Introduced by Rodney Brooks in 1986
- Behaviors are networks of sensing and acting modules (augmented finite state machines)
- Modules are grouped into layers of competence
- Layers can subsume lower layers

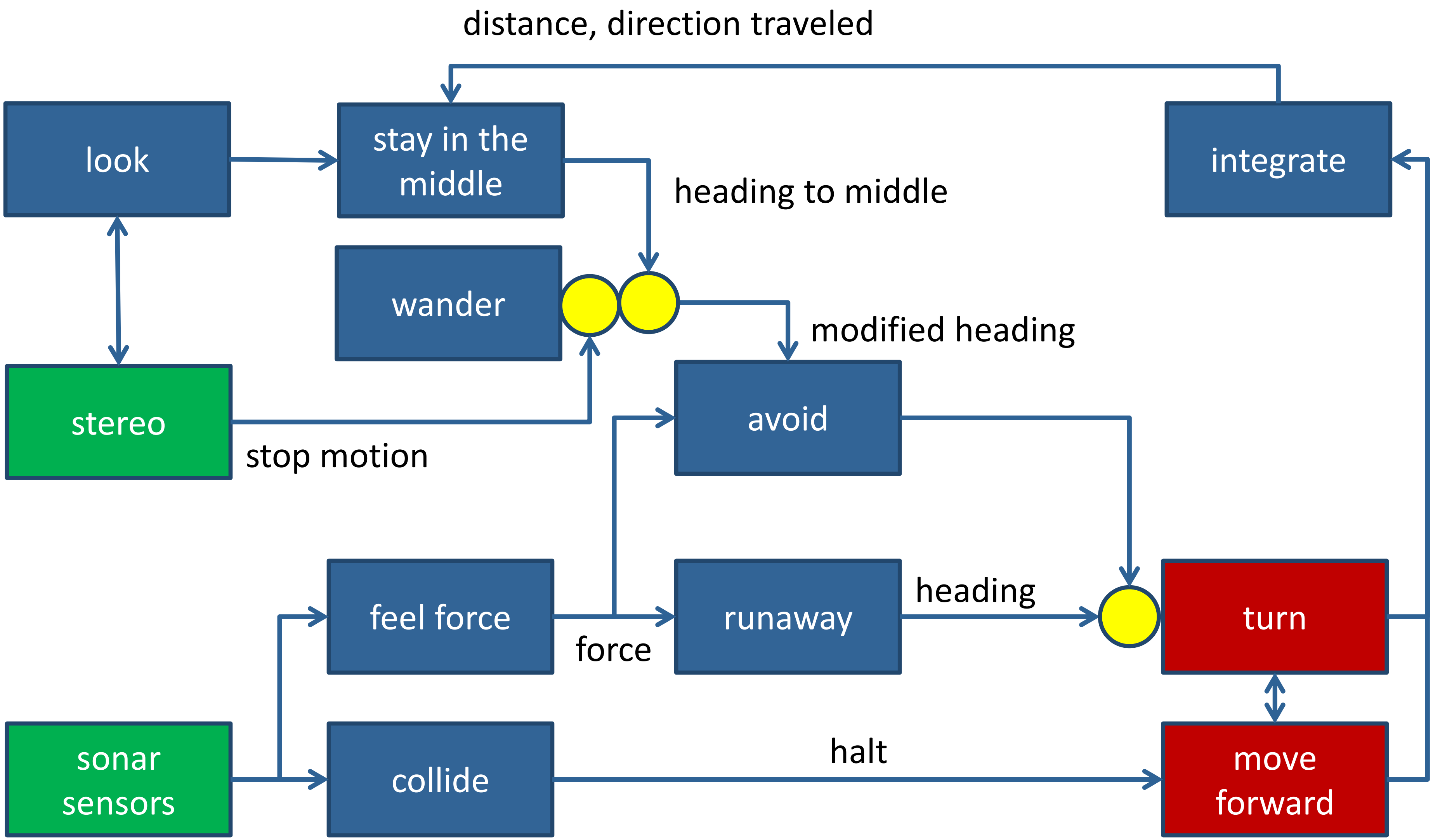
Level 1: Avoid



Level 2: Wander



Level 3: Follow Corridor



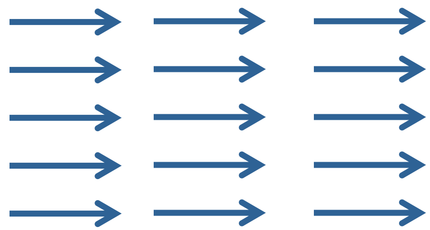
Roomba Robot

- Exercise: Model the behavior of a Roomba robot.

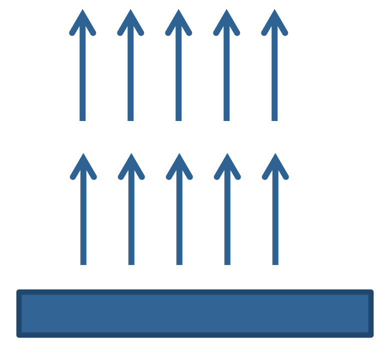
Navigation with Potential Fields

- Treat robot as a particle under the influence of a potential field
- Robot travels along the derivative of the potential
- Field depends on obstacles, desired travel directions and targets
- Resulting field (vector) is given by the summation of primitive fields
- Strength of field may change with distance to obstacle/target

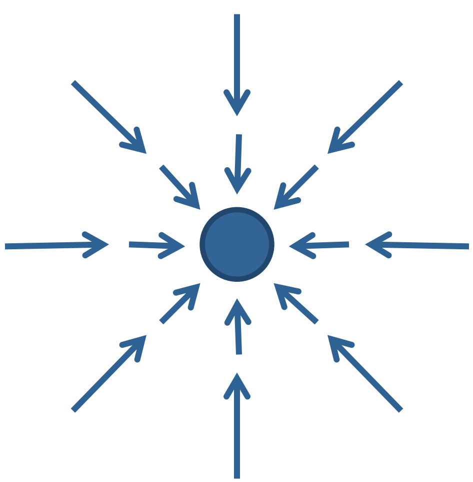
Primitive Potential Fields



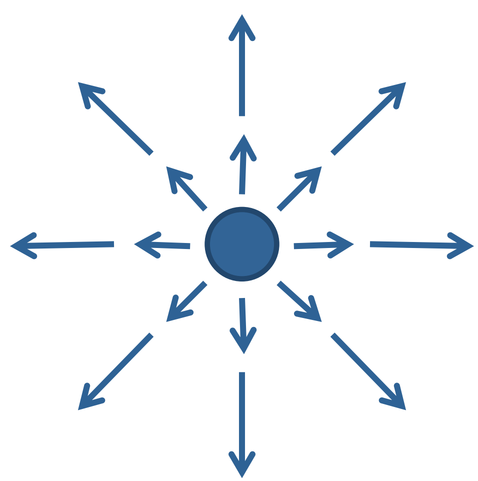
Uniform



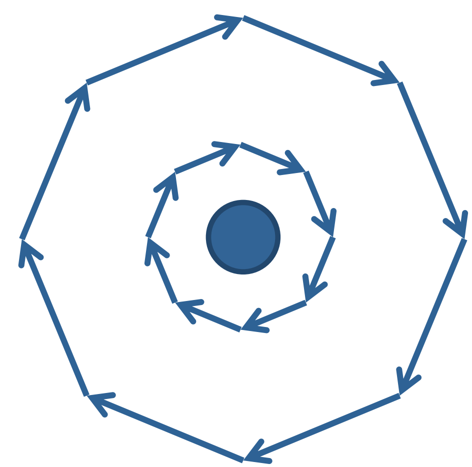
Perpendicular



Attractive

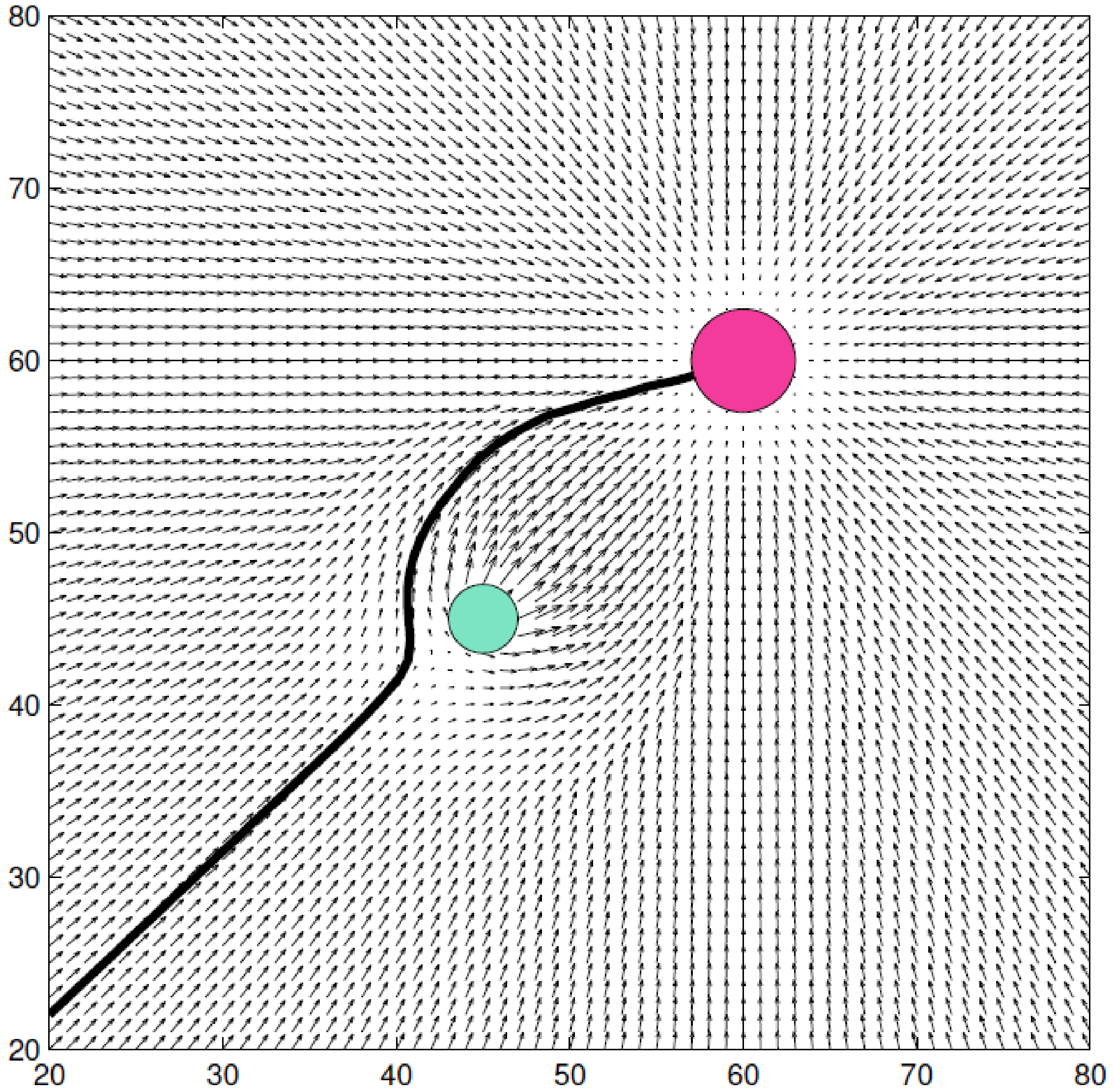


Repulsive



Tangential

Example: reach goal and avoid obstacles

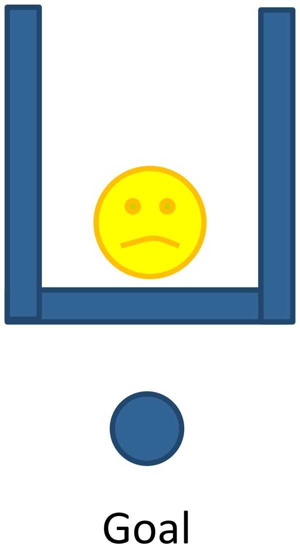


Corridor Following Robot

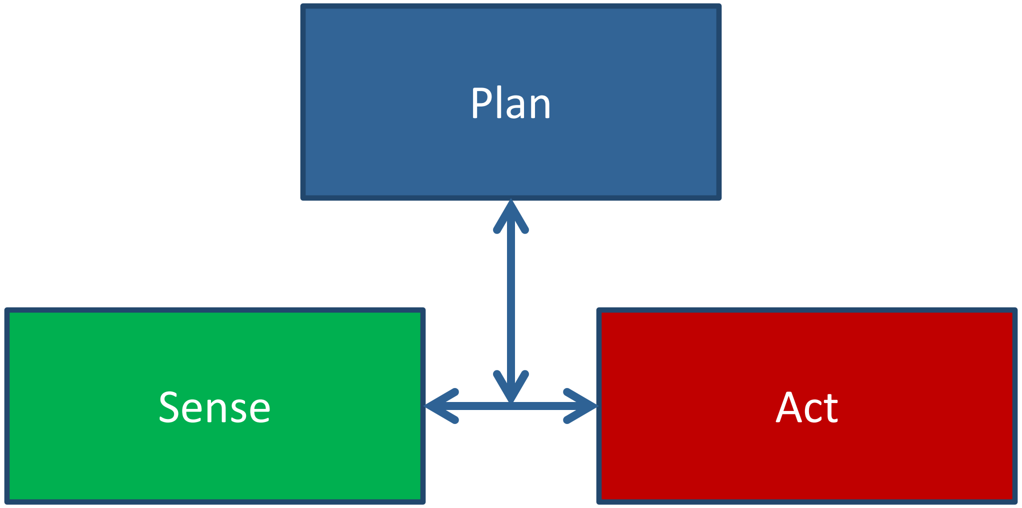
- Level 1 (collision avoidance)
  add repulsive fields for the detected obstacles
- Level 2 (wander)
  add a uniform field into a (random) direction
- Level 3 (corridor following)
  replaces the wander field by three fields (two perpendicular, one parallel to the walls)

Characteristics of Potential Fields

- Simple method which is often used
- Easy to visualize
- Easy to combine different fields (with parameter tuning)
- But: Suffer from local minima
  - Random motion to escape local minimum
  - Backtracking
  - Increase potential of visited regions
  - High-level planner



Hybrid deliberative/reactive Paradigm



- Combines advantages of previous paradigms
  - World model used in high-level planning
  - Closed-loop, reactive low-level control

Modern Robot Architectures

- Robots became rather complex systems
- Often, a large set of individual capabilities is needed
- Flexible composition of different capabilities for different tasks

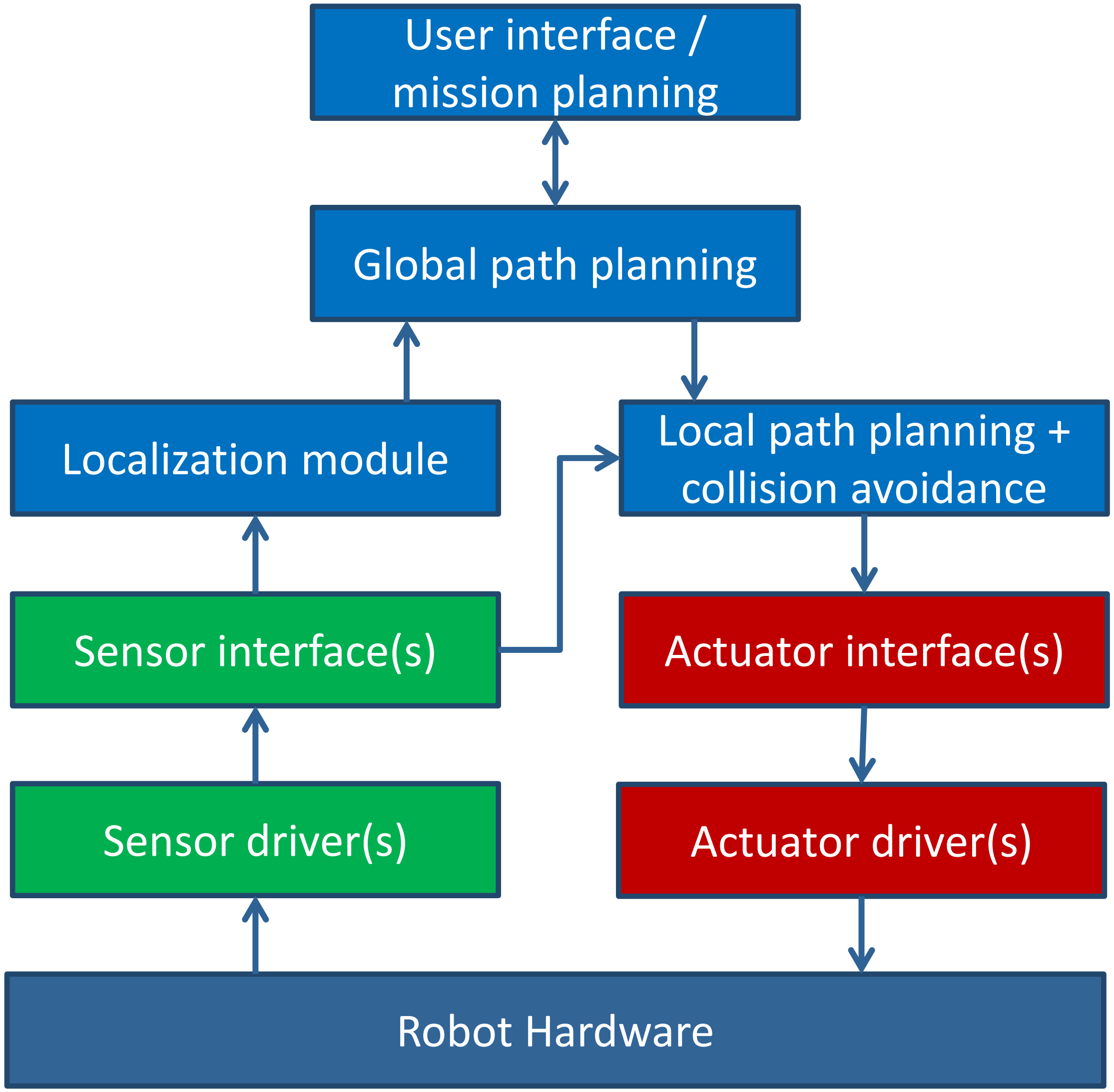
Best Practices for Robot Architectures

- Modular
- Robust
- De-centralized
- Facilitate software re-use
- Hardware and software abstraction
- Provide introspection
- Data logging and playback
- Easy to learn and to extend

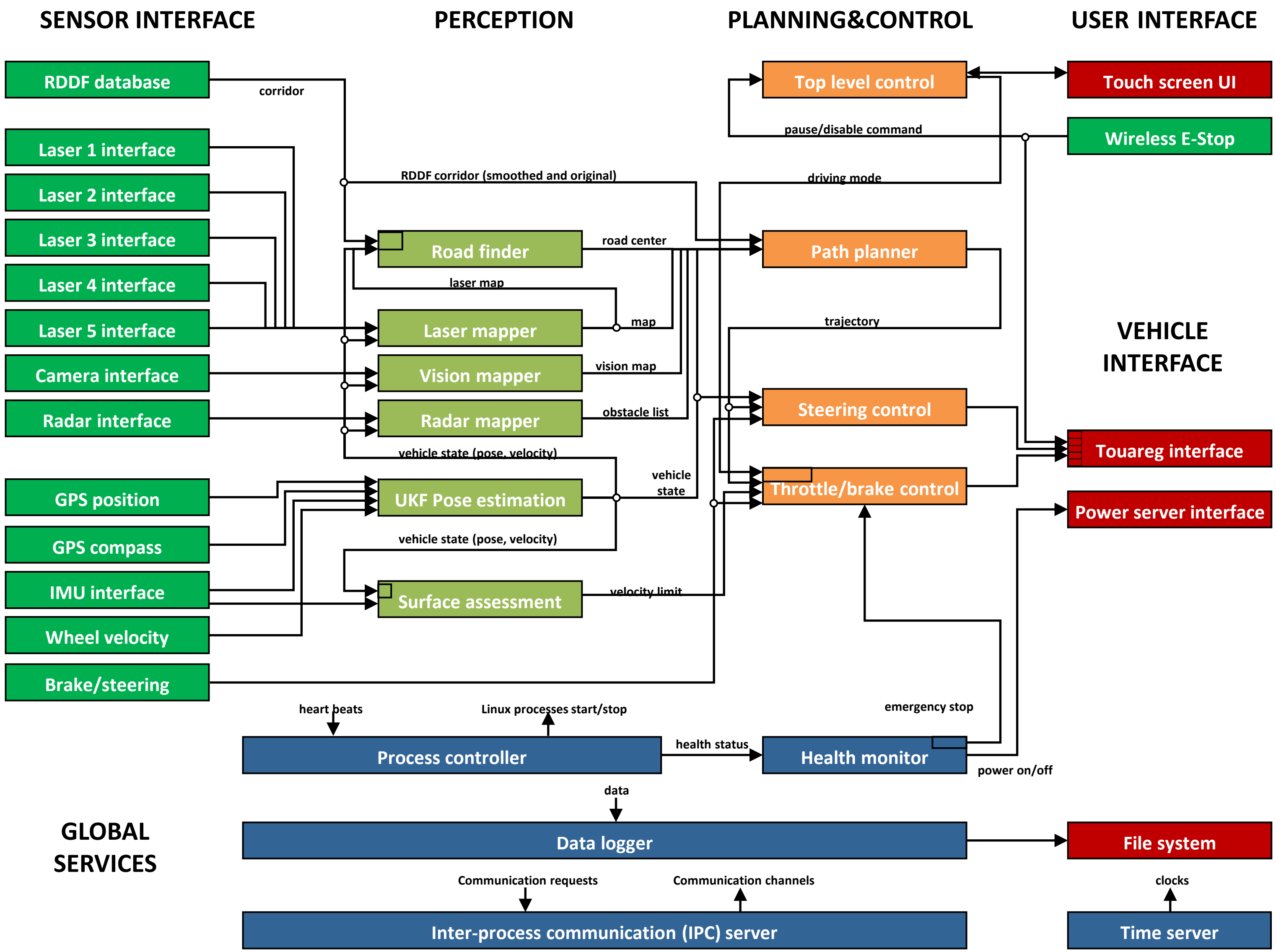
Robotic Middleware

- Provides infrastructure
- Communication between modules
- Data logging facilities
- Tools for visualization
- Several systems available
  - Open-source: ROS (Robot Operating System), Player/Stage, CARMEN, YARP, OROCOS
  - Closed-source: Microsoft Robotics Studio

Example Architecture for Navigation

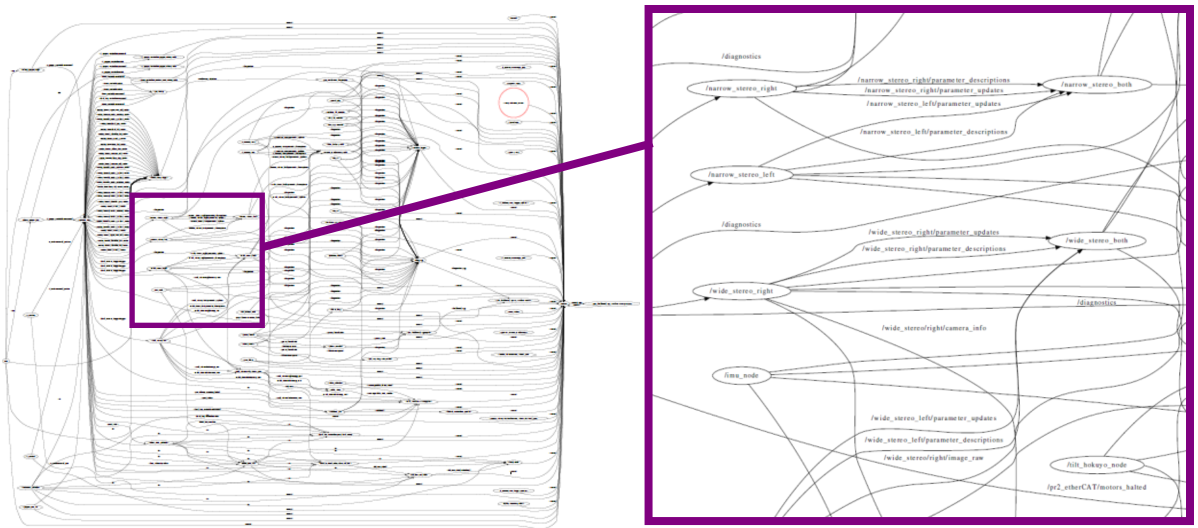


Stanley's Software Architecture



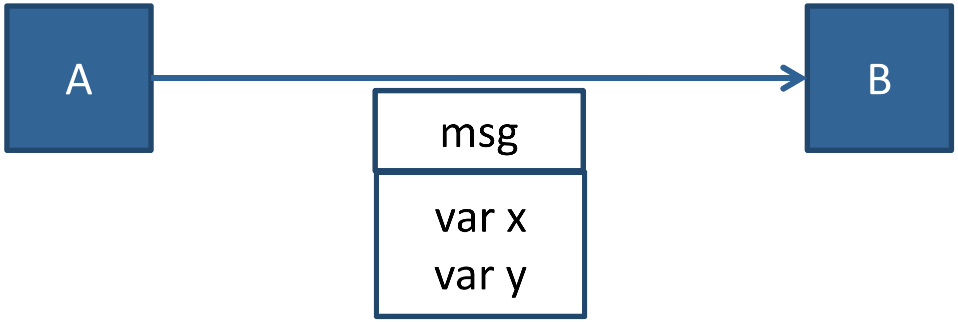
PR2 Software Architecture

- Two 7-DOF arms, grippers, torso, 2-DOF head
- 7 cameras, 2 laser scanners
- Two 8-core CPUs, 3 network switches
- 73 nodes, 328 message topics, 174 services

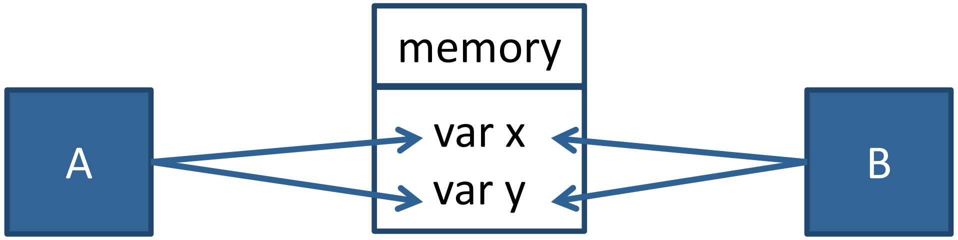


Communication Paradigms

- Message-based communication



- Direct (shared) memory access



Forms of Communication

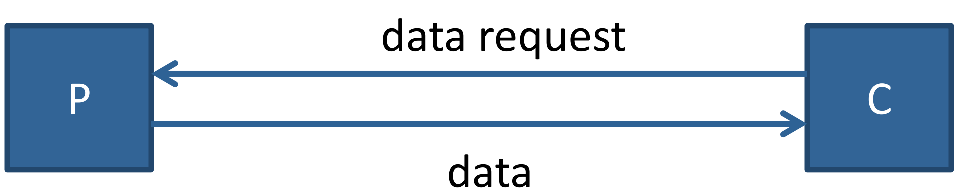
- Push
- Pull
- Publisher/subscriber
- Publish to blackboard
- Remote procedure calls / service calls
- Preemptive tasks / actions

Push

- Broadcast
- One-way communication
- Send as the information is generated by the producer P

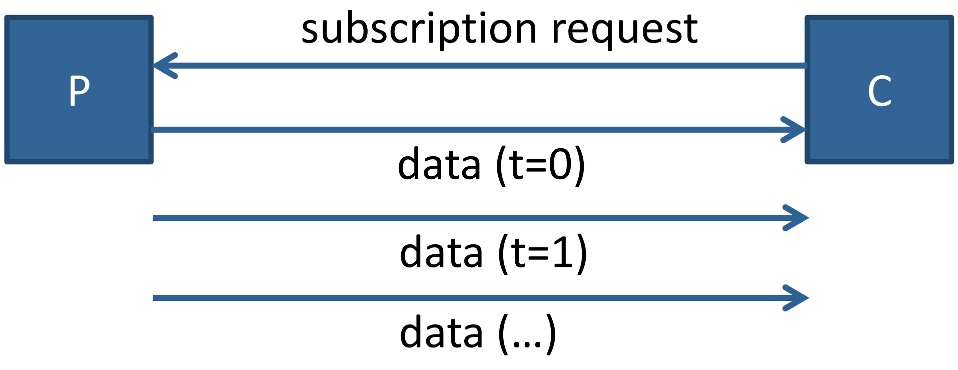
Pull

- Data is delivered upon request by the consumer C (e.g., a map of the building)
- Useful if the consumer C controls the process and the data is not required (or available) at high frequency



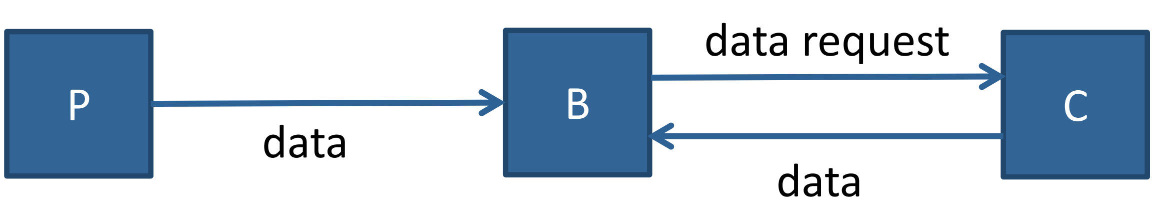
Publisher/Subscriber

- The consumer C requests a subscription for the data by the producer P (e.g., a camera or GPS)
- The producer P sends the subscribed data as it is generated to C
- Data generated according to a trigger (e.g., sensor data, computations, other messages, …)



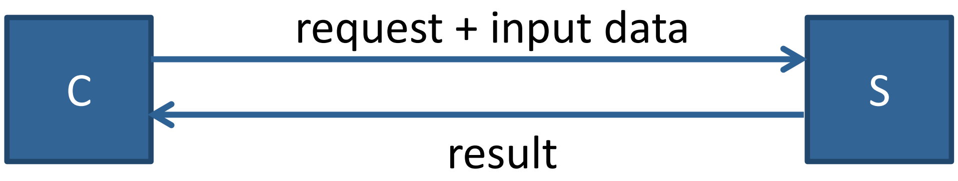
Publish to Blackboard

- The producer P sends data to the blackboard (e.g., parameter server)
- A consumer C pull data from the blackboard B
- Only the last instance of data is stored in the blackboard B



Service Calls

- The client C sends a request to the server S
- The server returns the result
- The client waits for the result (synchronous communication)
- Also called: Remote Procedure Call



Actions (Preemptive Tasks)

- The client requests the execution of an enduring action (e.g., navigate to a goal location)
- The server executes this action and sends continuously status updates
- Task execution may be canceled from both sides (e.g., timeout, new navigation goal,…)

Robot Operating System (ROS)

- We will use ROS in the lab course
- http://www.ros.org/
- Installation instructions, tutorials, docs

Concepts in ROS

- Nodes: programs that communicate with each other
- Messages: data structure (e.g., “Image”)
- Topics: typed message channels to which nodes can publish/subscribe (e.g., “/camera1/image_color”)
- Parameters: stored in a blackboard

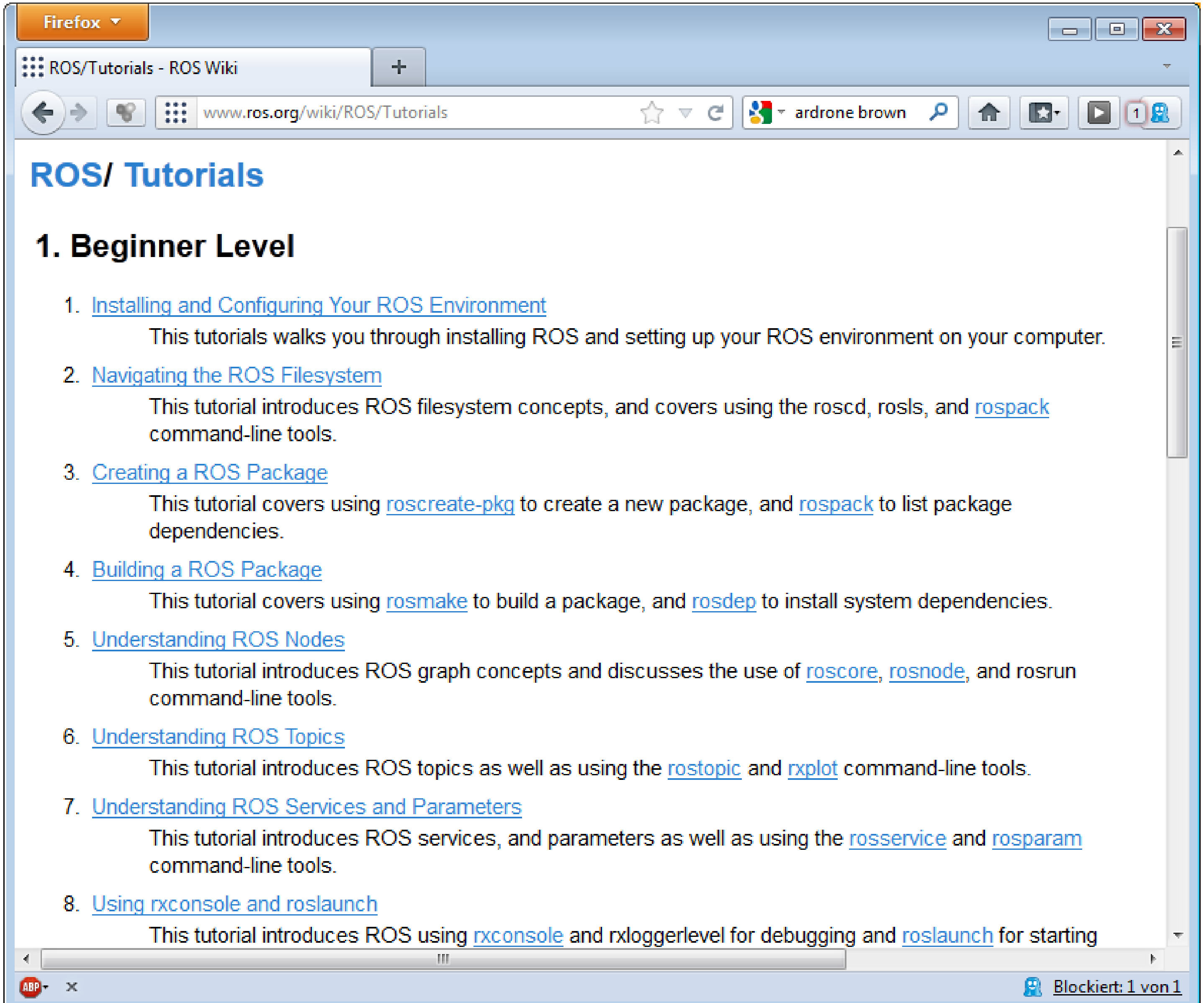
Software Management

- Package: atomic unit of building, contains one or more nodes and/or message definitions
- Stack: atomic unit of releasing, contains several packages with a common theme
- Repository: contains several stacks, typically one repository per institution

Useful Tools

- roscreate-pkg
- rosmake
- roscore
- rosnode list/info
- rostopic list/echo
- rosbag record/play
- rosrun

Tutorials in ROS



ROS/ Tutorials

1. Beginner Level

1. Installing and Configuring Your ROS Environment
   This tutorials walks you through installing ROS and setting up your ROS environment on your computer.
2. Navigating the ROS Filesystem
   This tutorial introduces ROS filesystem concepts, and covers using the roscd, rosls, and rospack command-line tools.
3. Creating a ROS Package
   This tutorial covers using roscreate-pkg to create a new package, and rospack to list package dependencies.
4. Building a ROS Package
   This tutorial covers using rosmake to build a package, and rosdep to install system dependencies.
5. Understanding ROS Nodes
   This tutorial introduces ROS graph concepts and discusses the use of roscore, rosnode, and rosrun command-line tools.
6. Understanding ROS Topics
   This tutorial introduces ROS topics as well as using the rostopic and rxplot command-line tools.
7. Understanding ROS Services and Parameters
   This tutorial introduces ROS services, and parameters as well as using the rosservice and rosparam command-line tools.
8. Using rxconsole and roslaunch
   This tutorial introduces ROS using rxconsole and rxloggerlevel for debugging and roslaunch for starting

Exercise Sheet 1

- On the course website
- Solutions are due in 2 weeks (May 1st)

- Theory part:
  Define the motion model of a quadrocopter
  (will be covered next week)
- Practical part:
  Playback a bag file with data from
  quadrocopter & plot trajectory

Summary

- History of mobile robotics
- Brief intro on quadrocopters
- Paradigms in robotics
- Architectures and middleware

Questions?

- See you next week!