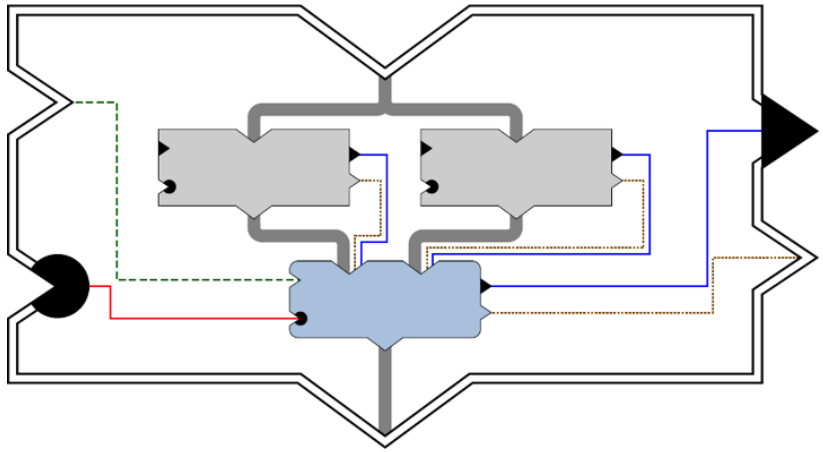
RRLAB
THE ROBOTICS RESEARCH LAB

Autonomous Mobile Robots

10. Control Architectures for AMR



Prof. Karsten Berns
Robotics Research Lab
Department of Computer Science
University of Kaiserslautern, Germany

Contents

- Introduction to control architectures
- Deliberative control architectures
- Reactive control architectures
- Behavior-based control architectures

Introduction to Control Architectures

Control Approaches in Robotics

- Challenge
  - Design a system for controlling complex robots having complex tasks
  - Find a suitable way of representing information
  - Analyze built up system with respect to errors
- Required: Methodology for building up control systems

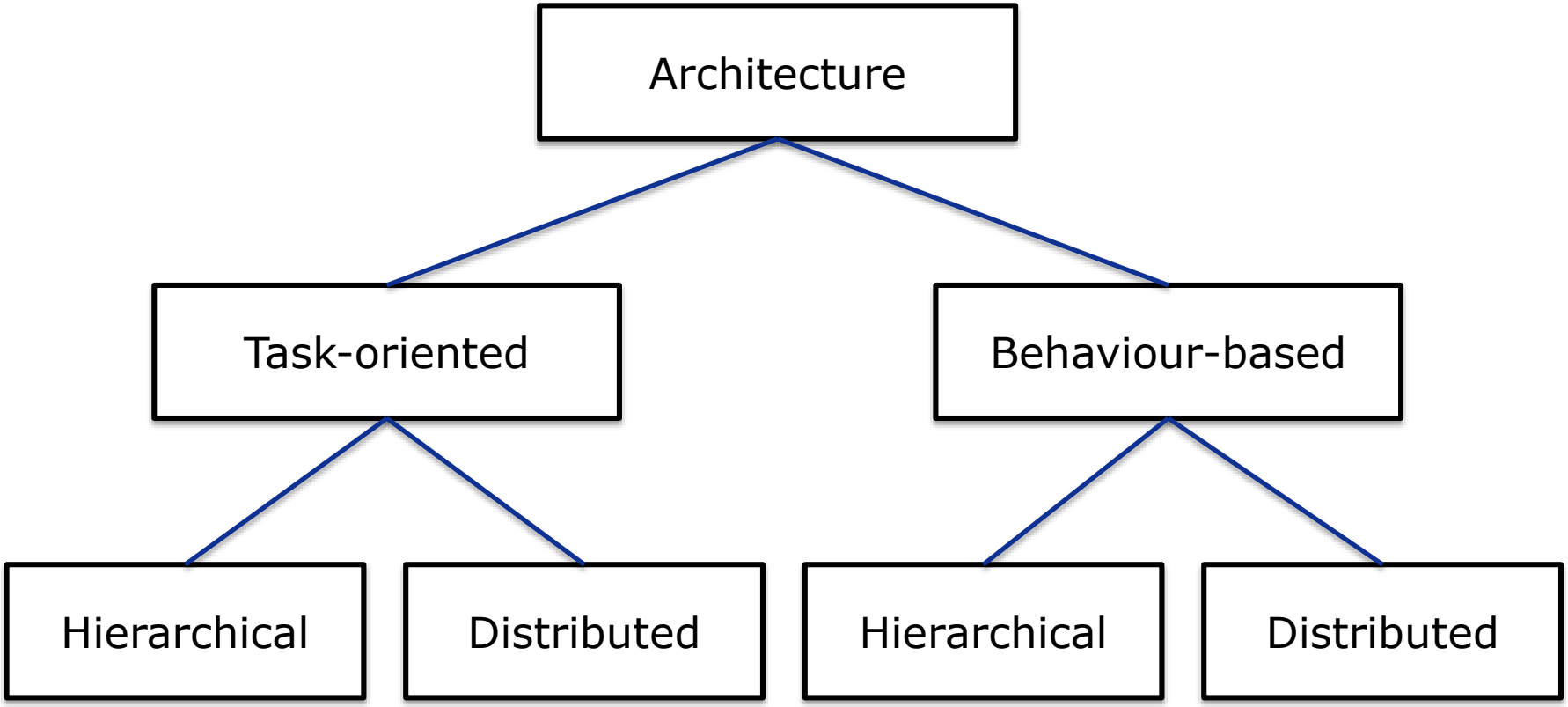
Definition of a Control Architecture

- A control architecture is a framework enabling a system to fulfill the following tasks
- Fusion of sensor data into logical sensors: Sensor preprocessing for localization, generation of obstacle maps, generation of maps for navigation and planning, object recognition, display and representation of knowledge
- Motor controller: Access to the hardware by convenient motion control interface, e.g. velocity v and angular velocity ω
- Pilot function: Path control via specification of motion commands, required for e.g. collision avoidance, driving through narrow passages, turning in dead-end situations

Definition of a Control Architecture

- Navigation: Calculation of accessible tracks to be traversed via the pilot function; the plan includes avoidance of obstacles and requires knowledge about the surrounding area
- Planning function: Generation of actions, setting targets for the navigation component. This includes strategic decisions and keeping the overview concerning given tasks
- User interaction: Access to the control software by a suitable Human-Machine-Interface (HMI)
- For all tasks mentioned above world knowledge has to be considered, which has to be provided adequately

Distinction of Control Architectures



Overview of control architectures
commonly used in the field of robotics

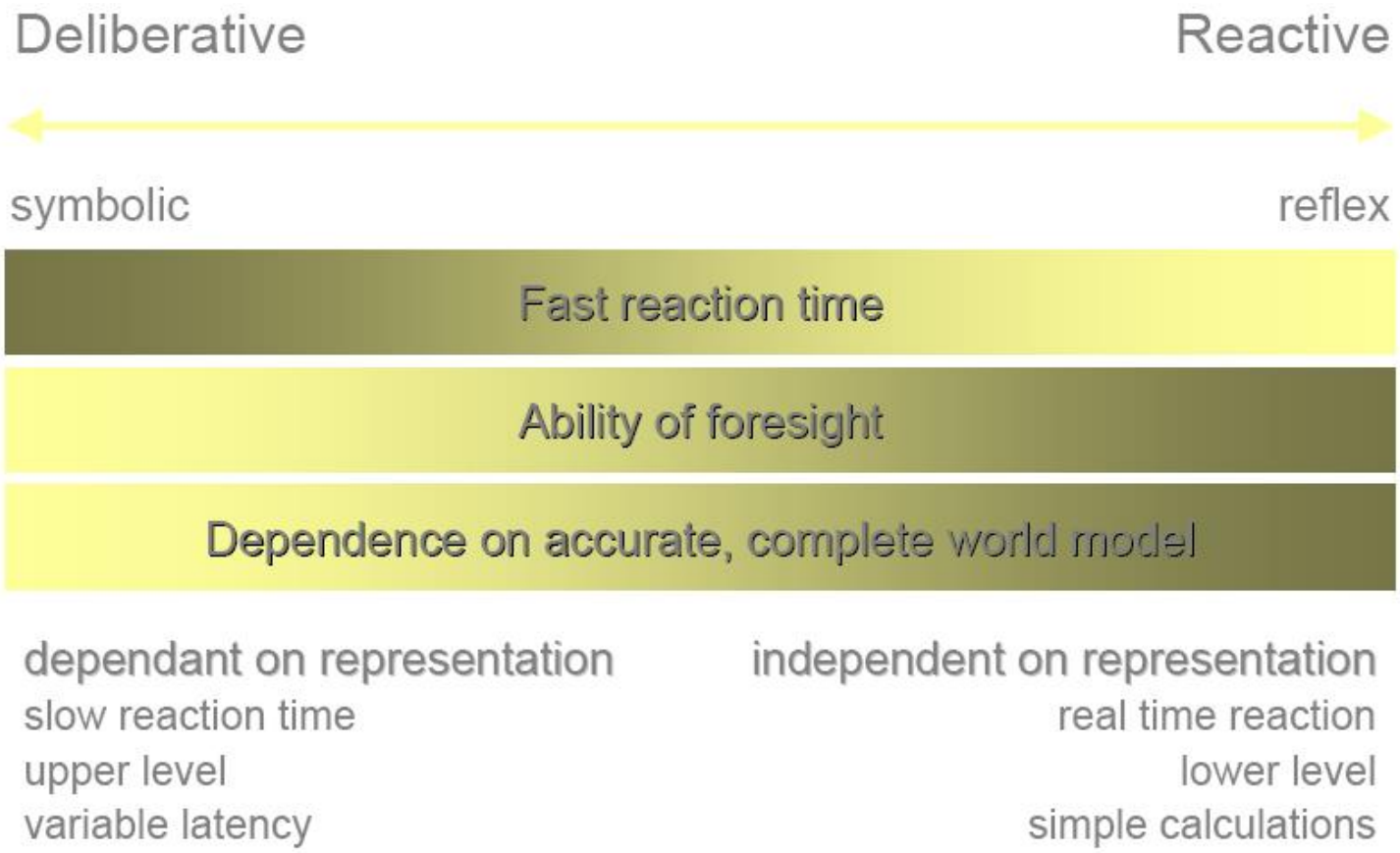
Hierarchical vs. Distributed Control Architectures

- Hierarchical architectures
  - Depend on the assumption that tasks can be divided into subtasks
  - Subtasks are arranged such that higher level components generate subgoals for lower level components
- Distributed architectures
  - Allow the assignment of subtasks to independent components
  - Suitable communication mechanism for data transfer required

Task-Oriented vs. Behavior-Based Control Arch.

- Task-oriented architectures
  - Depend on a central world model which is manipulated and evaluated by the different components (sensing, modeling, planning, execution)
  - Responsible components (having exclusive access) for sensor data processing and control value generation
- Behavior-based control architectures
  - Decomposition of a given task into independent behaviors
  - Each behaviour keeps its own compact representation of the environment which is required for task execution
  - Unlimited access to sensor data and control interface
  - Mechanisms for the coordination of conflicting data required

Distinction of Control Architectures



Deliberative vs. reactive control

Degree of Deliberation

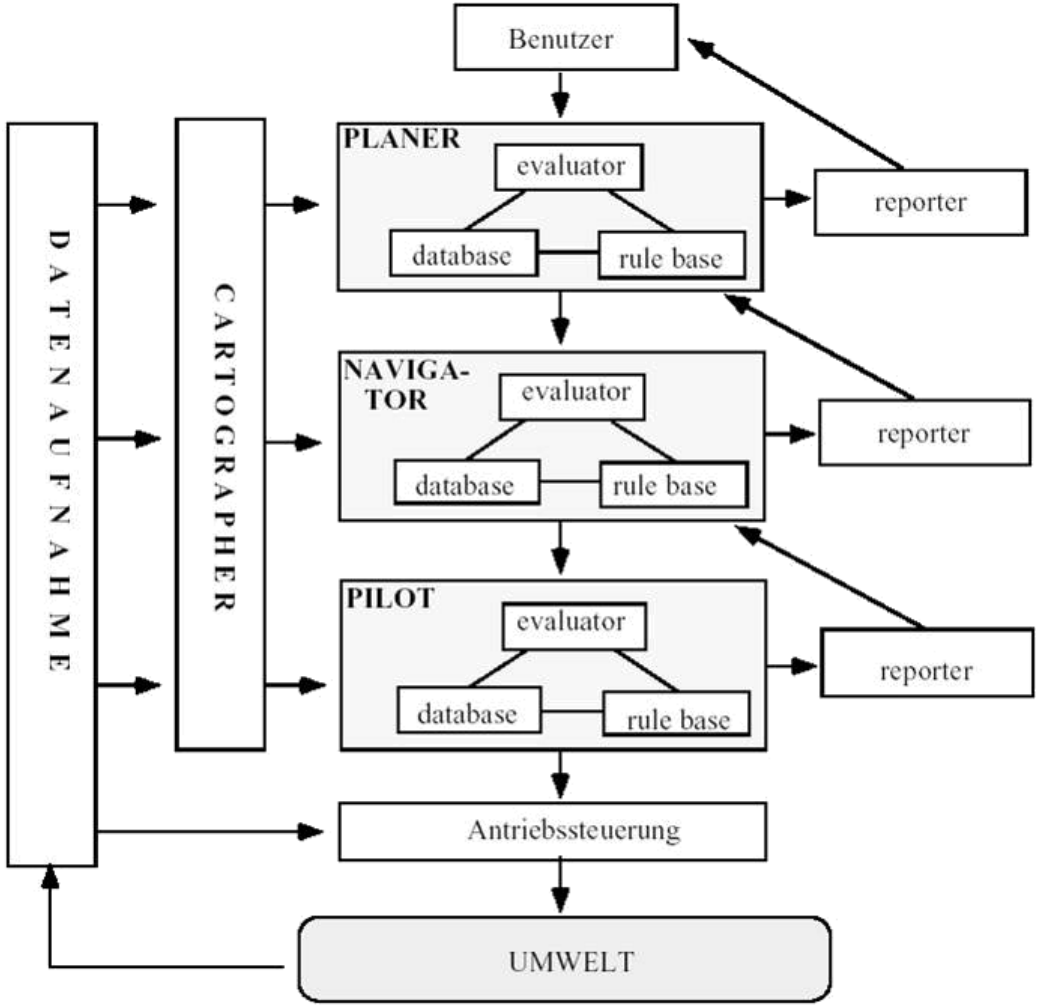
- Distinction with respect to the degree of deliberation [Mataric97]
- Reactive architectures
  - No storage of knowledge
  - Immediate reaction on occurring situations
  - Weaknesses with respect to planning tasks requiring a certain degree of memory
- Deliberative architectures
  - Elaborate model of the world
  - Suited to complex tasks
  - Drawbacks concerning outdated knowledge, false sensor readings, or reaction time

Degree of Deliberation

- Desired: combination of the advantages of both reactive and deliberative architectures
- Hybrid architectures
  - Lower reactive layer and a higher deliberative layer
  - Breaks the control system into two inhomogeneous parts with different characteristics
- Behavior-based architectures
  - Storage of a representation of the environment which is distributed among the single components
  - Can combine reactive and deliberative components into one architectural design
  - However: often confused with reactive architectures

Deliberative Control Architectures

Control Architecture according to Isik & Meystel



Control Architecture according to Isik & Meystel

- Introduction of reporters
- Pilot → Navigator
  - Deviation from default path (dead-ends, impassable obstacles, further deviation from given track)
  - Deviation from default driving speed (emergency brake)
  - Malfunctions
  - Request of partial targets/track segments after last target was reached

Control Architecture according to Isik & Meystel

- Navigator → Planner
  - Reaching partial targets and requesting for new partial targets
  - Obstructions (path is blocked)
  - Malfunctions that are not manageable by navigator
- Planner → User
  - Messages when final target is reached or system is ready
  - Malfunctions that are not manageable by planner

Control Architecture according to Isik & Meystel

- Knowledge base of each level has the following task
  - Compare: knowledge ⇔ environment impressions
  - Assignment: object ⇔ meaning (obstacle/free space)
  - Aggregation: knowledge ⇔ environment representation
- Cartographer supplies different levels with environment representation
  - Pilot: obstacle information/-map (robot centered)
  - Navigator: geometric map of environment (in world coordinates)
  - Planner: Graph of complete environment

Control Architecture according to Isik & Meystel

- Levels with different time frame
  - Pilot: $\approx 0.1\,\mathrm{s}$, real-time control of AMR
  - Navigator: $\approx 1\,\mathrm{s}$, generates drive commands to get to way points or provides path
  - Planner: long time horizon, complex algorithms for path planning
- Disadvantage of hierarchical task-oriented control architectures: AMR only works, if all components work properly

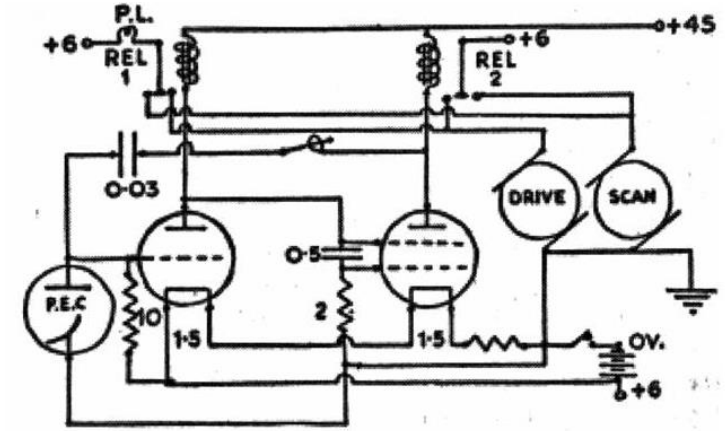
Reactive Control Architectures

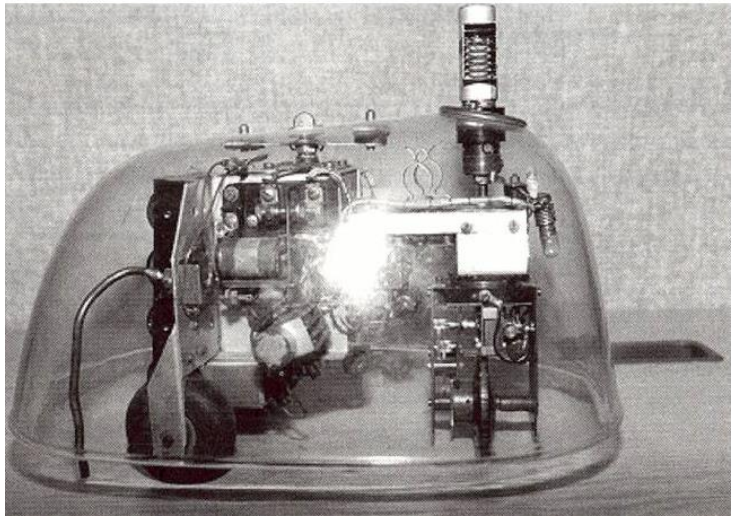
Cybernetics: Machina Speculatrix

- Cybernetics : Combination of control theory, computer science and biology, to explain behavior of animals and machines (Wiener and Ashby, 1948-52) → Situatedness: Strong mutual connection between organisms (machine) and it's environment
- 1953: W. Grey Walter makes use of this principle: Machine Speculatrix
- Principles
  - The simpler the better
  - Exploration: System is in constant movement
  - Attraction: Motivation in the direction of a surrounding object
  - Repulsion: System moves away from negative stimuli
  - Ability to distinguish: what is productive, what is not

Machina Speculatrix

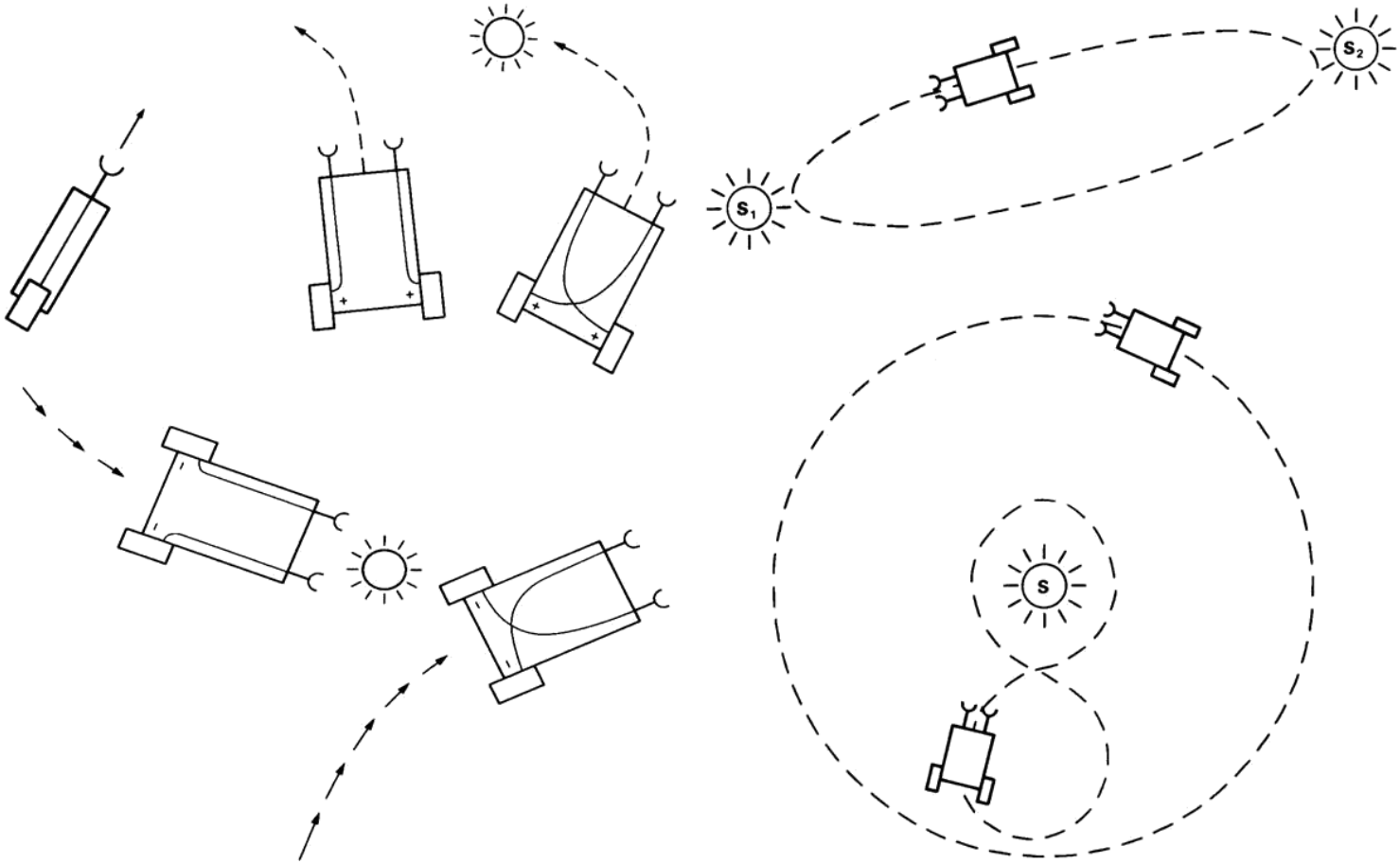
- Transferred behavior
  - Search for light
  - Approach weak light sources
  - Move away from strong light sources
  - Turn and push
  - Reload battery
- Behaviors have different priorities, e.g. obstacle avoidance prior to light tracking, principle is still in use today (arbitration coordination mechanism)





Machina Speculatrix

Braitenberg Vehicle



Thought experiment of Valentino Braitenberg (1984)

Behavior-based Control Architectures

Breaking with Traditions

- Classic AI approach (sense, model, plan, act) dominated robotics research of the next 3 decades
- Brooks (1987): ”Planning is just a way of avoiding figuring out what to do next”
- Shift in Paradigm: sensing and acting → behavior-based robotics
- At the same time: distributed artificial intelligence
- Multiagent-systems as foundation of all intelligence (Society of the Mind, Minsky 1986)
- Preferably simple agents show intelligence through coordinated behavior

The Behavior-Based Control Approach

- Extension of reactive architectures: Distributed storage of representation of the environment
- No centralized world model (unlike deliberative architectures)
- System relies on various forms of distributed representations and performs distributed computations on them

Why Behavior-Based Robotics?

- Complex behavior don’t necessarily arise from complex control systems
- The real world is the best model
- Simplicity of programming
- Robustness at noisy sensor readings
- Systems should be designable in an incremental way
- All calculations performed on-board
- Thomas Huxley: ”The great end of life is not knowledge, but action”

Characteristics of Behavior-Based Systems

- Behavior-based approaches are in general modular and support reuse of components
- Based on embedding in and interaction with environment
- Behaviors cannot simply be serialized or implemented centralized.
- A central challenge in the design process is the coordination of behaviors (arbitration/command fusion)
- Behaviors interact with each other through the world rather than using internal models.
- Testing of single behaviors can take place immediately at times when only parts of the system are implemented

Behaviors for Mobile Robots

- Exploration (movement in general direction)
- Targeted (movement in direction of attractors)
- Avoidance (avoid collision)
- Path following (wall, path, stripe, …)
- Posture behavior (balance, stability)
- Social behavior (parting, hives, …)
- Remote-autonomous behavior (user interaction, coordin.)
- Perceptual behavior (visual search, …)
- Walking behavior (for walking robots, …)
- Manipulator behavior, grip behavior

Examples for Behavior-Based Control Architectures

- R. Brooks: Subsumption Architecture [Brooks86] (although sometimes classified as reactive architecture)
- R. Arkin: AuRA (Autonomous Robot Architecture, behaviour-based component of a hybrid architecture) [Arkin87, Arkin98]
- M. Mataric [Mataric97]
- Miscellaneous fuzzy logic approaches [Saffiotti95, Konolige97, Selekwa05]
- DAMN (Distributed Architecture for Mobile Navigation) [Rosenblatt97a]
- Althaus/Christensen: Dynamical System Approach [Althaus02]

Subsumption Architecture: Introduction

- Developed by Rodney Brooks, MIT, 1986
- Alignment of behaviors along horizontal layers
- All behaviors can access all sensor data to generate actions for all actuators
- Interaction of behaviors through interdiction of inputs and overruling of outputs
- Primary feedback via the environment
- Special C derivative: Interactive C
- Application on mobile vehicles and walking machines

Subsumption Architecture: Examples

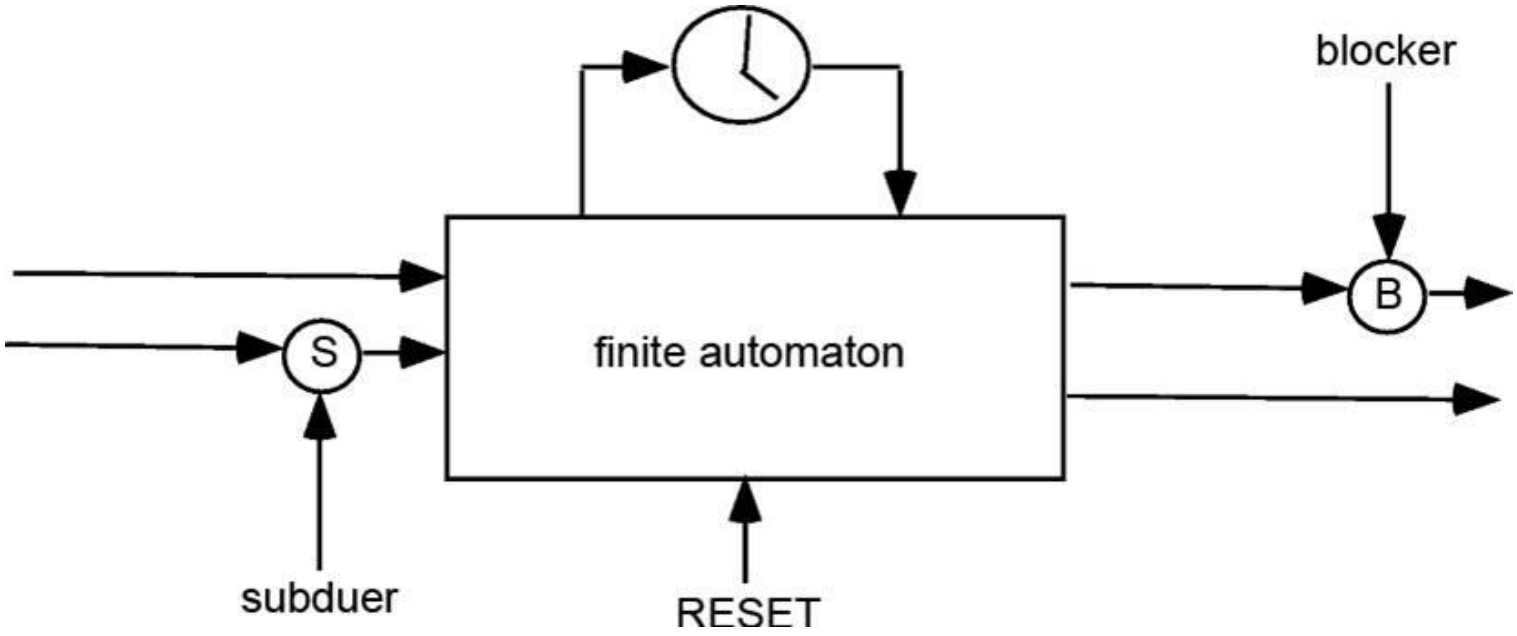
Genghis (1990)

Attila (1992)

Characteristics of the Subsumption Architecture

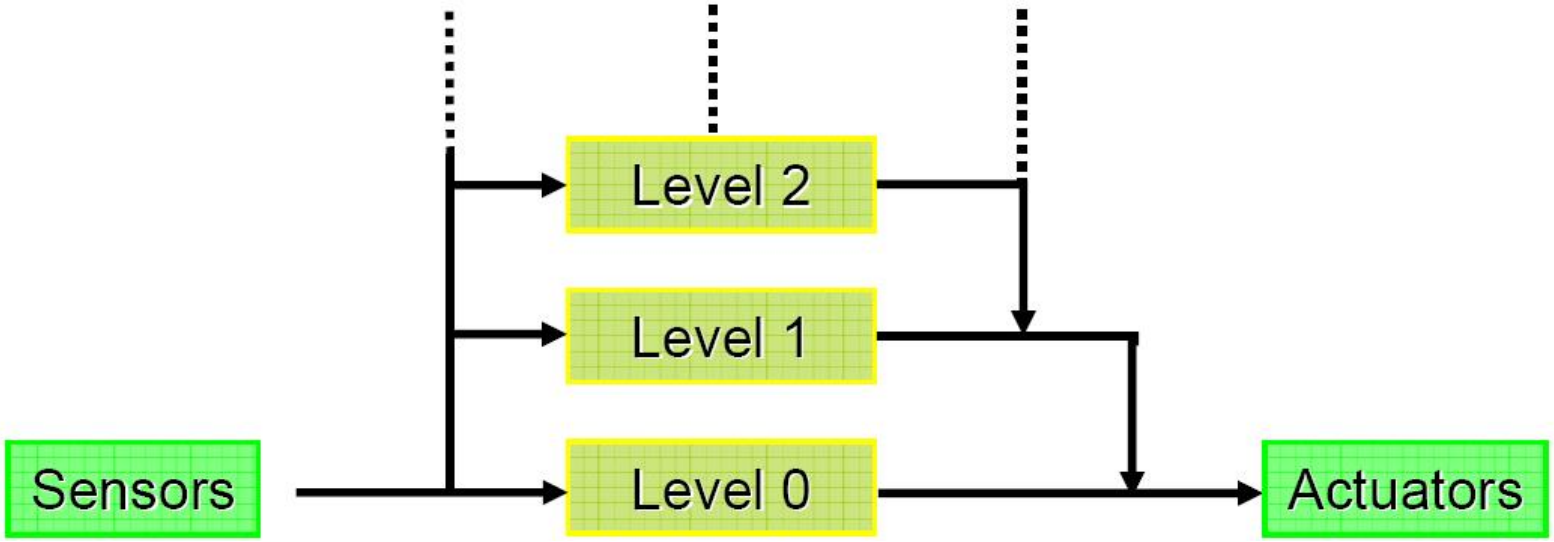
- Support for parallelization
- Extensibility (sensors, behaviors)
- Simple implementation of complex system behavior
- Robustness (malfunction of behavior modules and levels)
- Ability to simultaneously achieve multiple tasks
- The following key elements are defined
  - Situatedness: The robot is bound as a unit to its environment
  - Embodiment: The robot has a physical presence
  - Intelligence: Observed intelligence arises both from computation and interaction with the environment
  - Emergence: Actions result from the interaction of subcomponents and the environment.

Subsumption: Basic Element



Basic element of the Subsumption Architecture

Layout of the Behavior Levels



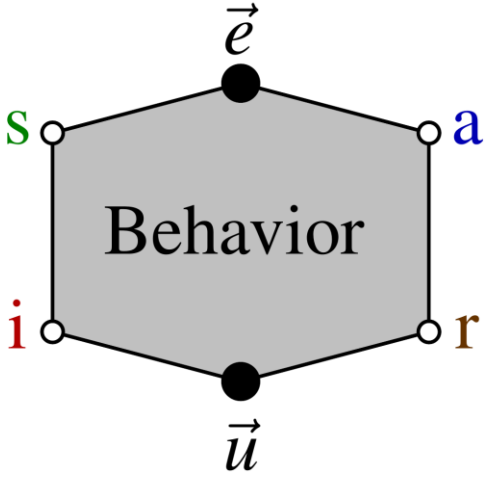
Layout of the behavior levels in the Subsumption Architecture

Subsumption Architecture: Discussion

- Benefits
  - Successful implementation of complex tasks: Walking, obstacle avoidance
  - Early testing: fully functional subsystems
  - Behaviors are based on imprecise, simple geometric information of low-level sensors
- Drawbacks
  - Scalability issue: Increasing number of behaviours
  - Restriction to reactive behaviors hinders planning
  - Further support of the development process required: guidelines, tools
  - System behavior hard to evaluate

The iB2C Behavior

- Fundamental unit in iB2C: Behavior module
$$B = (f_a, F)$$
- f_a: Activity function
- F: Transfer function determines output vector



- $\vec{e} = \overrightarrow{(d, \sigma)} \in \mathbb{R}^{2^m}$: Input vector
- $\vec{u} = \overrightarrow{(d, \sigma)} \in \mathbb{R}^{2^n}$: Output vector
- $d \in \mathbb{R}$: Data value
- $\sigma \in \mathbb{R}^+$: Tolerated error margin

- $s \in [0,1]$: Stimulation
- $i \in [0,1]$: Inhibition
- $a \in [0,1]$: Activity
- $r \in [0,1]$: Target Rating

Input/Output Vector

- Input vector $\vec{e} \in R^m$
  - Behaviors receive data needed for fulfilling their work via the input $\vec{e}$ which can be composed of sensory data or information from other modules
- Output vector $\vec{u} \in R^n$
  - Transmission of data generated by the behavior
  - This output describes the influence a behavior can have on the environment via the robot's actuators, or on other modules

- Data in Input/Output vector are decorated by deviation value describing tolerated deviation

Transfer Function

- The transfer function $F(\vec{e})$ determines the output vector $\vec{u}$, where

$$F: \mathbb{R}^m \times [0,1] \rightarrow \mathbb{R}^n,\ F(\vec{e}) = \vec{u}$$

- F provides the intelligence of a module, calculating actions depending on input values and internal representations
- No limitation on F: reactive respond to input values, state machine, sophisticated algorithms, etc.

Meta Signals: Stimulation and Inhibition

- Each behavior has an input stimulation $s \in [0,1]$ determining the intended relevance of B
- $s = 0$ indicates no stimulation, $s = 1$ a fully stimulated behavior. Values between 0 and 1 refer to a partially stimulated behavior
- The inhibition $i \in [0,1]$ has the inverse effect of the stimulation s and reduces the relevance of the inhibited module
- $i = 1$ refers to full inhibition, $i = 0$ to no inhibition
- Stimulation can be used to adjust the relevance of competing behavior or to enable higher-level module to recruit lower-level behaviors and their functionality by explicitly stimulating them

Meta Signals: Activation (Internal)

- Activation $\iota \in [0,1]$: indicates the effective relevance in the network
- Limits behavior's activity
- Is composed of the stimulation s and the inhibition i

$$\iota = s \cdot (1 - i)$$

Meta Signals: Activity

- Activity $a \in [0,1]$: represents the amount of influence of B
- $a = 1$: highest impact intended; $a = 0$: inactive
- Limited by activation

- Defined by the activity function f_a with

$$f_a : \mathbb{R}^m \times [0,1] \rightarrow [0,1] \times [0,1]^q$$
$$a = \min\left(\iota, f_a(\vec{e}, \iota)\right)$$

Meta Signals: Target Rating

- Target rating $r \in [0,1]$: indicates the behavior's internal desire to be active without activation and represents the contentment of B with the current system state
- $r = 1$: dissatisfied; $r = 0$: content

- Defined by activity function f_a

$$r = f_a(\vec{e}, \iota)$$

Behavior Module Properties: Principles

- Activity limitation principle: The activity a of a behavior B is limited by the activation ι of B: $a \leq \iota$
- Control/Arbitration separation: Meta signals are only used for arbitration and must not be used in the control algorithm
- Cycle avoidance: A behavior must not be in its own stimulation predecessor set

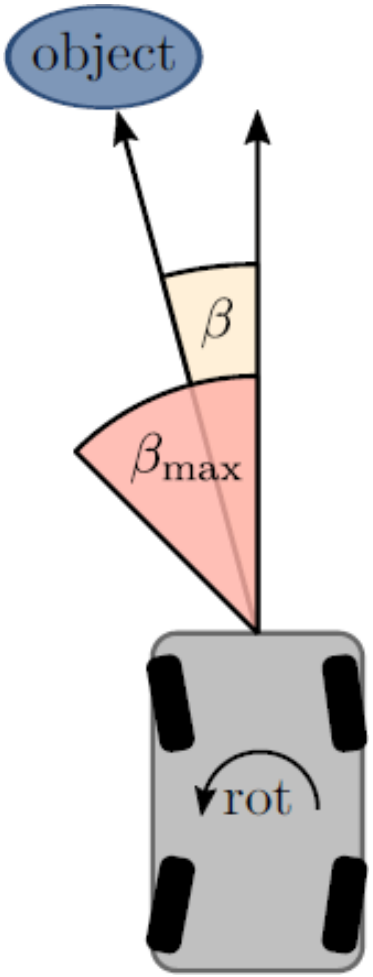
Example: Turn to Object

- Example behavior: Rotate vehicle to object
- Input $\vec{e}$: Angle β to an object detected by a camera system, visibility flag vis

$$\vec{e} = \begin{pmatrix} \beta \\ vis \end{pmatrix}$$

- Output $\vec{u}$: normalized rotation value $rot \in [-1,1]$: $\vec{u} = (rot)$
- Transfer function

$$rot = \begin{cases} 0 & \overline{vis} \\ -1 & vis \;\&\; (\beta < -\beta_{max}) \\ \dfrac{\beta}{\beta_{max}} & vis \;\&\; (-\beta_{max} \leq \beta \leq \beta_{max}) \\ 1 & vis \;\&\; (\beta > \beta_{max}) \end{cases}$$



Example: Turn to Object

- Activity Function: The behavior increases its activity if the angle to the object grows, i.e. according to β, given that it is visible

$$f_a(\vec{e}, \iota) = \begin{cases} 0 & \overline{vis} \\ \dfrac{|\beta|}{\beta_{max}} & vis \;\&\; (|\beta| \leq \beta_{max}) \\ 1 & Else \end{cases}$$

- The activation ι limits the activity a in order to meet the principle $a \leq \iota$

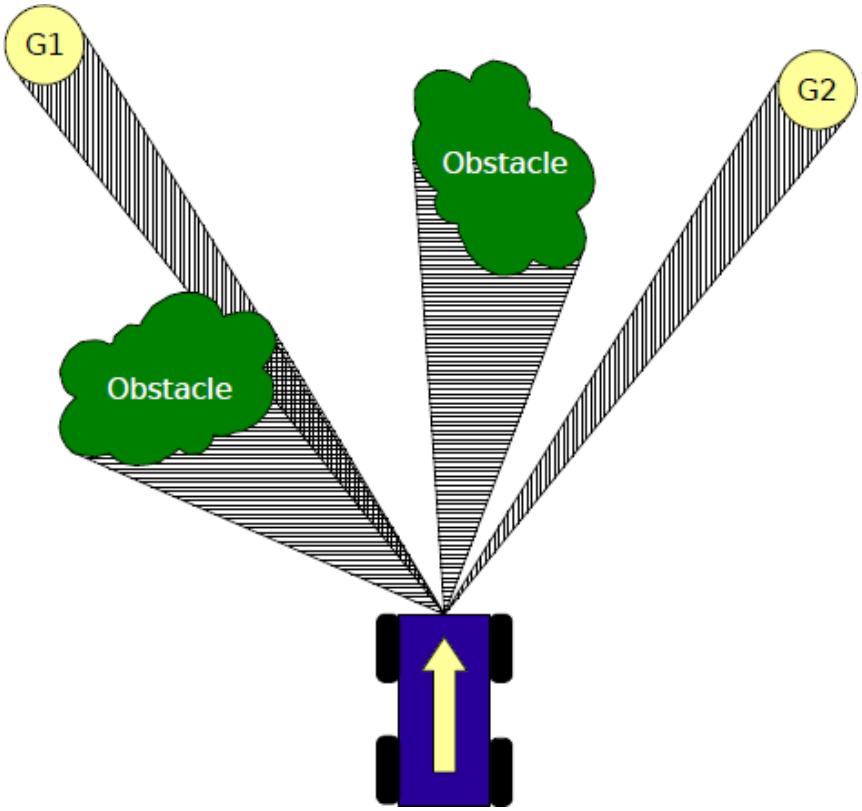
$$a = \min(\iota, f_a(\vec{e}, \iota))$$

- Target rating: The behavior is content as long as the object is centered in front. It becomes discontent if the angle β to the object grows or if it is no more visible. Therefore, the target rating r is set to

$$r = f_a(\vec{e}, \iota)$$

Coordination of Behaviors with Multiple Goals

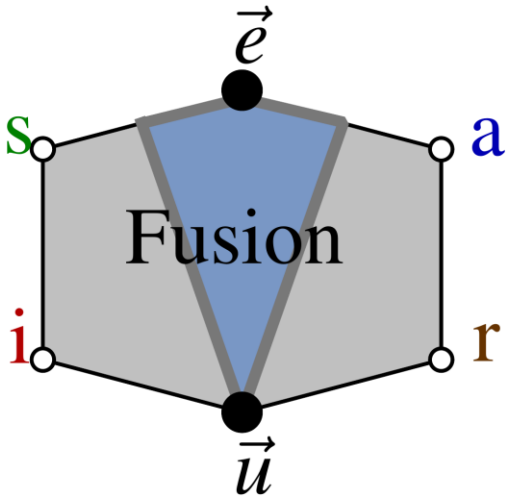
- Behaviors having multiple goals
- Example
  - *Prefer open space* behavior prefers directions without obstacle
  - *Turn to target* behavior prefers directions to several target points
- Coordination taking into account multiple goals necessary



Coordination problem of behaviors with multiple goals

Fusion Behavior Module

- Common interface
- Coordinate p competing behaviors B_c
- F is the fusion function processing input values to a merged output control vector $\vec{u}$



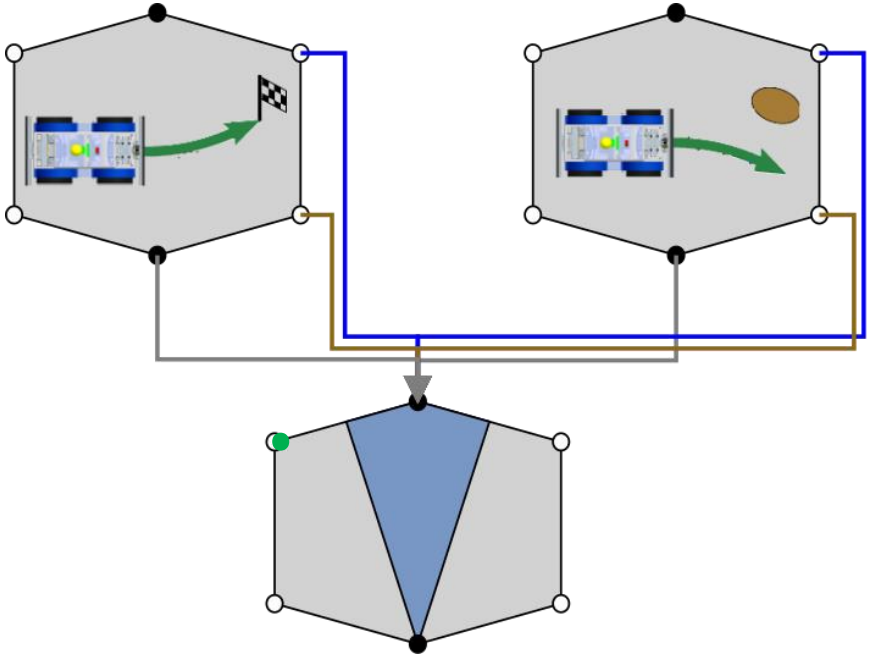
- Maximum fusion: $\vec{u} = \vec{u}_s$, $a = a_s$, $r = r_s$ where
$$s = \text{argmax}_c(a_c)$$

- Weighted average fusion: if $(1 = \max_c(r_c))$ then maximum fusion else

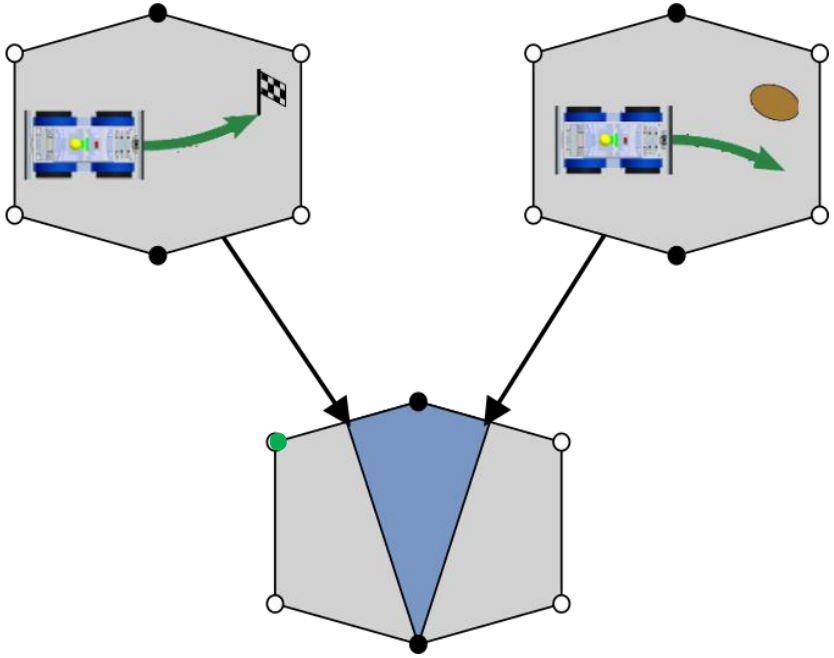
$$\vec{u} = \frac{\sum_{j=0}^{p-1} a_j \cdot \vec{e}_j}{\sum_{k=0}^{p-1} a_k} \quad a = \frac{\sum_{j=0}^{p-1} a_j^2}{\sum_{k=0}^{p-1} a_k} \quad r = \frac{\sum_{j=0}^{p-1} a_j^2 \cdot r_j}{\sum_{k=0}^{p-1} a_k}$$

Coordination Example

- *Turn to target competes* with *Prefer open space*
- For clarification the input vector of the fusion behavior is drawn separately



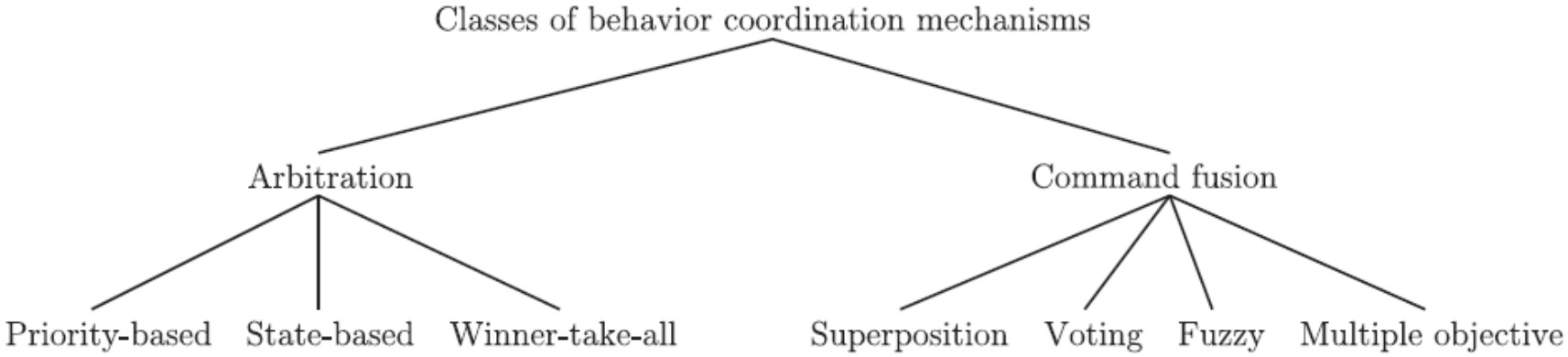
Complete Connection Setup



Simplified visualization

Coordination Mechanism Classification

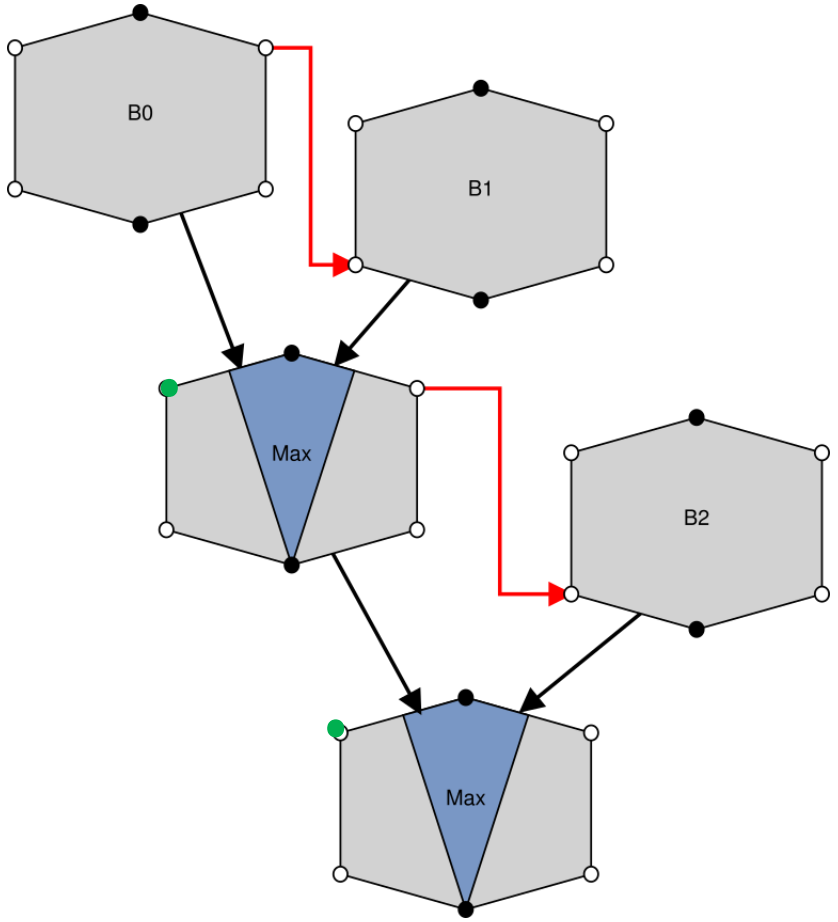
- Arbitration: One behavior or a set of them has control for a period of time
- Command fusion: Combination of control outputs
- Directly supported in iB2C: All arbitration mechanisms, superposition, voting (see below)



Classification of behavior coordination mechanims [Pirjanian99]

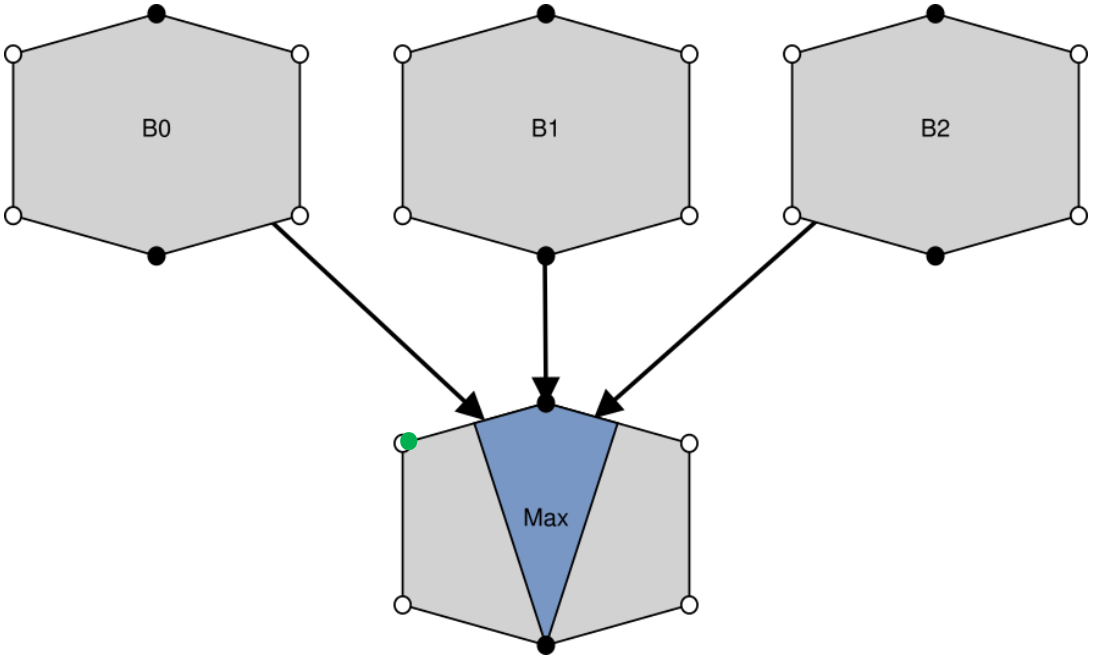
Priority-Based Arbitration

- The inhibition links define the priorities of the involved behaviors
- Competing outputs are arbitrated using a maximum fusion behavior



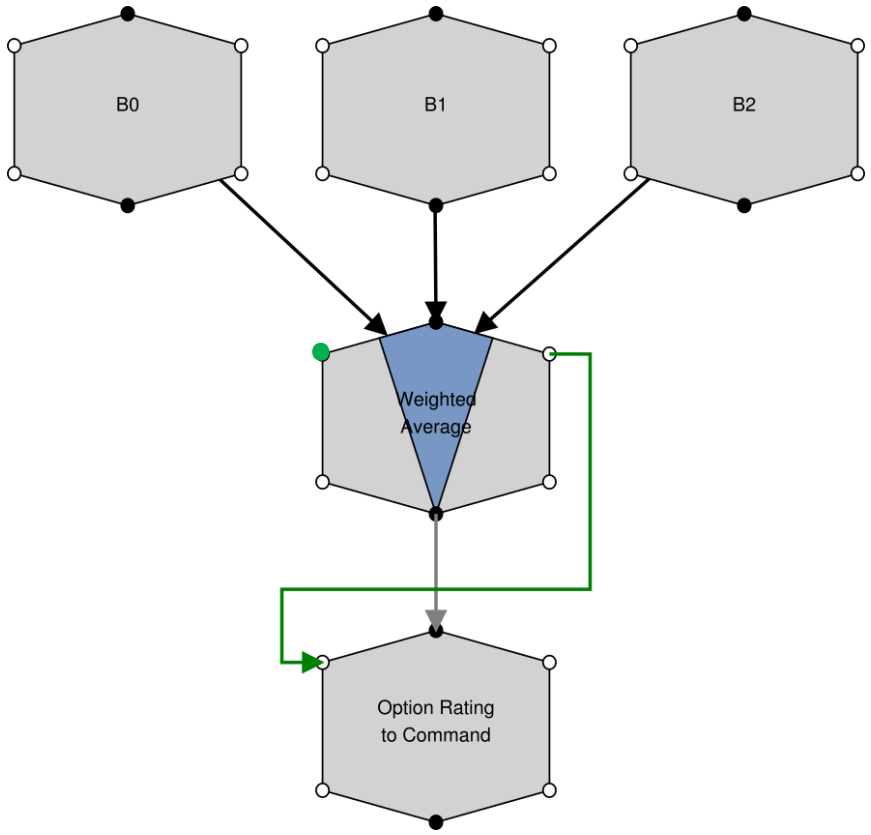
Winner-Takes-All Arbitration

- Directly supported by maximum fusion behavior
- Behavior with highest activity is chosen



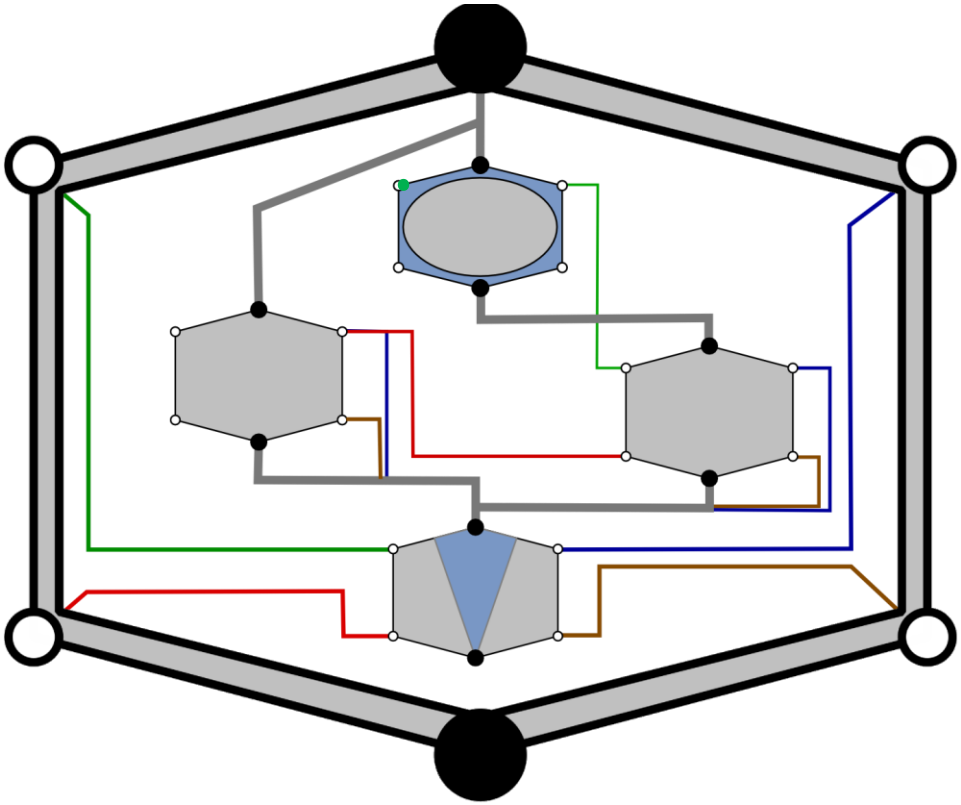
Voting

- Votes of each behavior for each of the n options
- *Option rating to command*: Map maximal option rating to motion command



Behavioral Groups

- Hierarchical abstraction: Simplification of complex structures
- Seamless integration

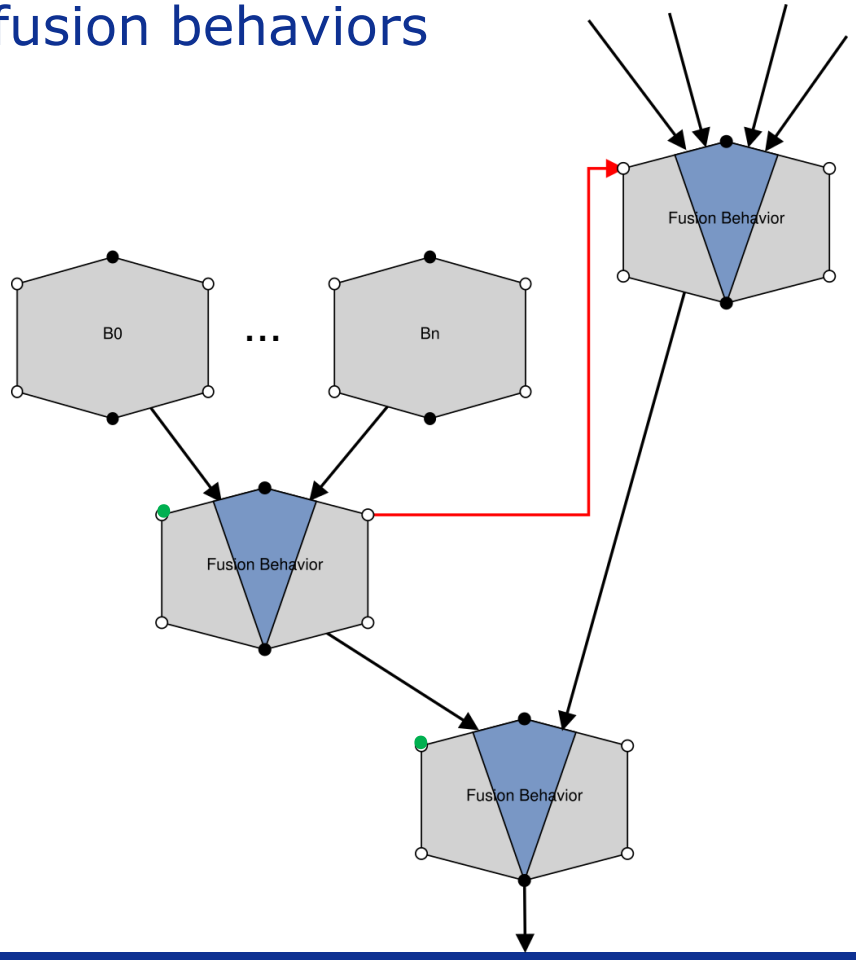


Structural Patterns: Best practice behavior interactions

- Behavior coordination: Arbitration (priority-based, state-based, winner-takes-all), command fusion (superposition, voting), see above
- Behavioral group (see above)
- State switching
- Inhibition layer
- Degrees of freedom access
- Combined stimulation

Inhibition Layer Pattern

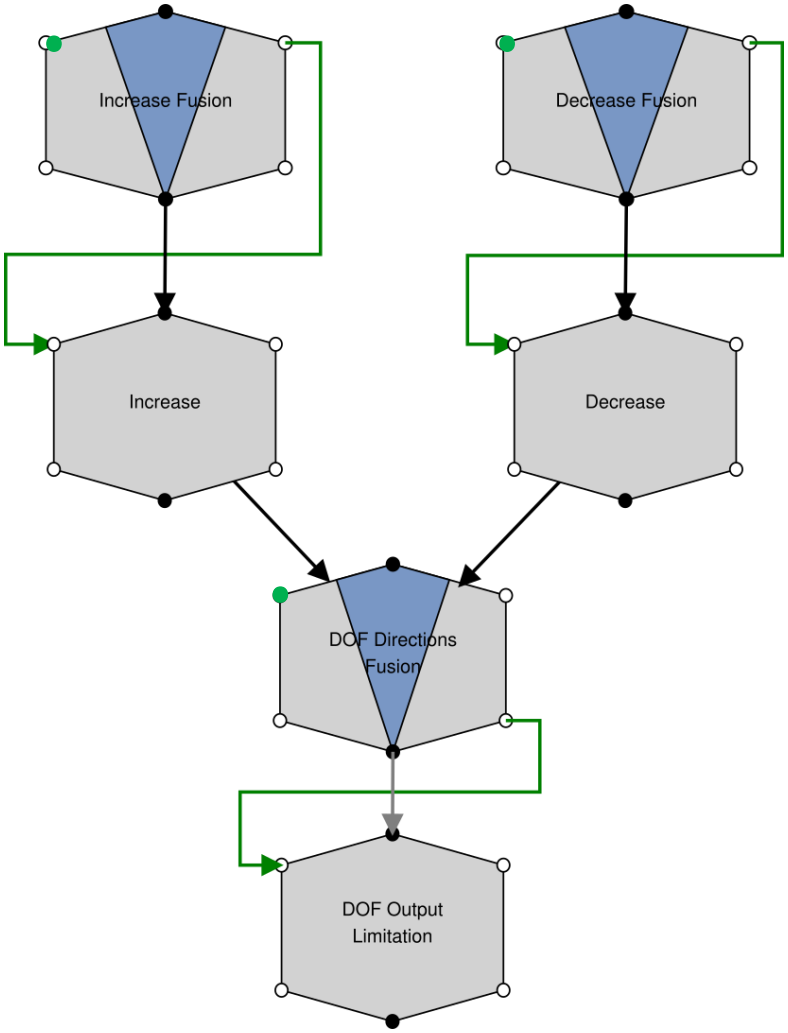
- (Partially) override commands of higher layers
- Structuring by fusion behaviors



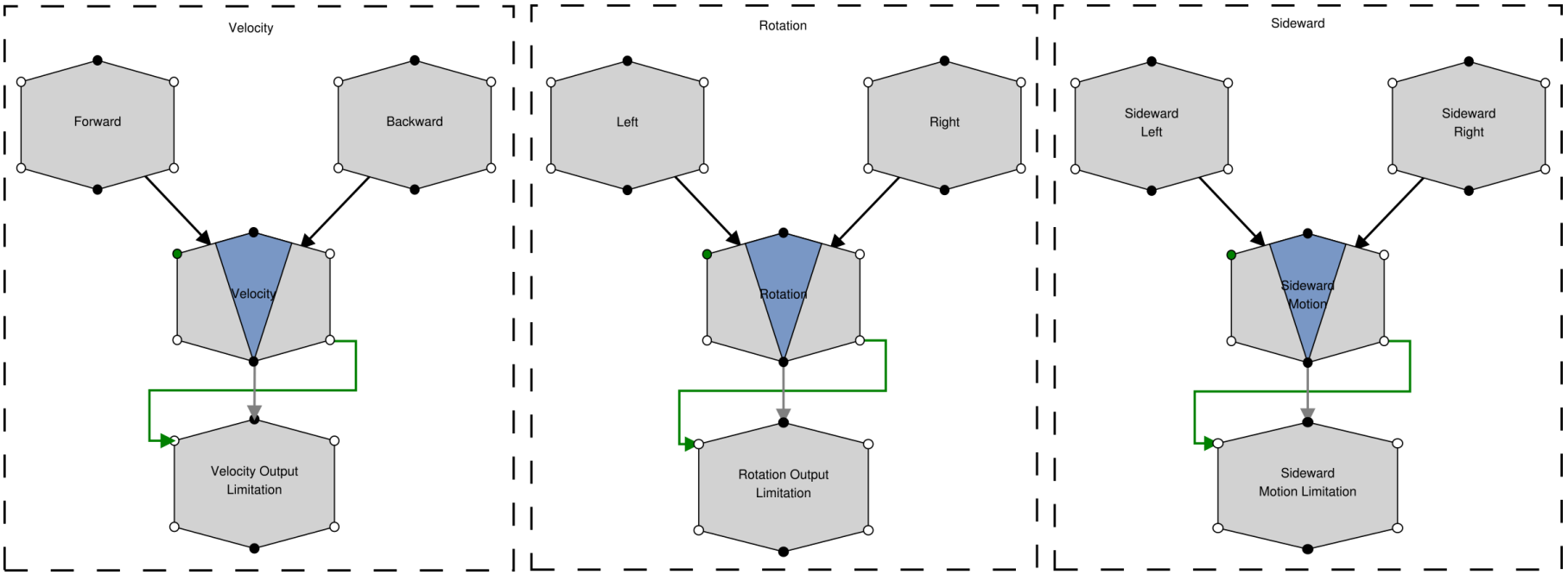
Access of Degrees of Freedom

- Access of available Degrees of Freedom (DOF), e. g., vehicle motions, camera pan and tilt (→ exercise)
- Approach: Divide DOF into positive and negative part, e. g., increase and decrease a steering angle, lift and lower an arm
- Selective support or weakening of options by coordinated behaviors
- Activity indicates strength of motion intention

Degrees of Freedom Access Pattern

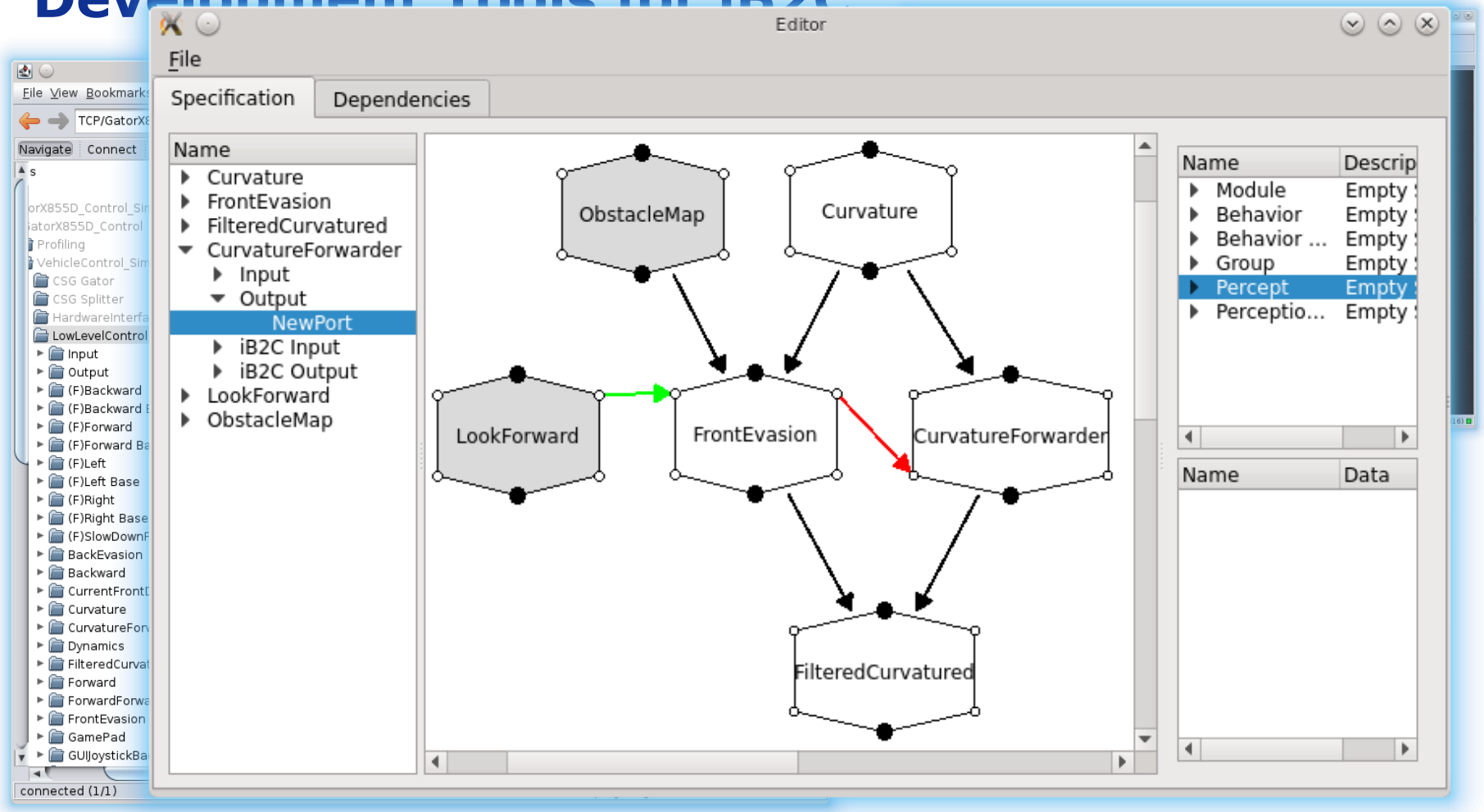


Degrees of Freedom Access Pattern: Example



The three fundamental control chains of RAVON:
Velocity, Rotation, and Sideward Motion [Schaefer11].

Development Tools for iB2C

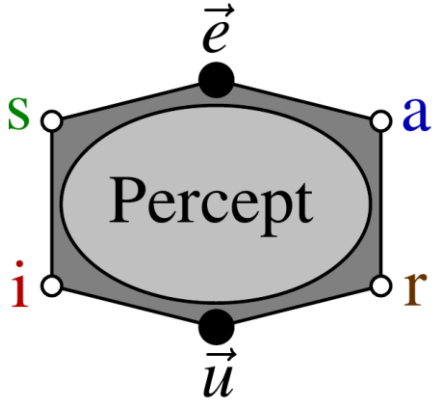


User interface of the tools Finstruct, Fingui, IB2Cdesigner

The iB2C Percept I

- Fundamental perception unit

$$P = \left(f_a, f_{p_u}, F\right)$$

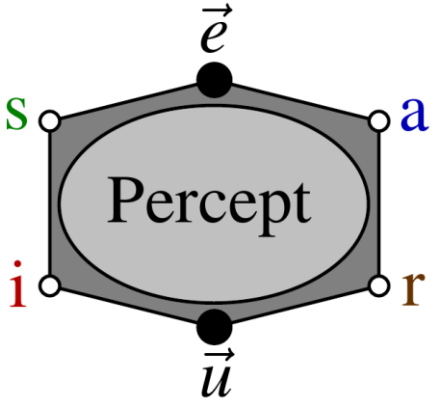


- f_a: Activity function
- f_{p_u}: Data quality transfer function
- F: Transfer function

- $s \in [0,1]$: Stimulation
- $i \in [0,1]$: Inhibition
- $a \in [0,1]$: Activity
- $r \in [0,1]$: Target rating

The iB2C Percept II

- $\vec{e} = \overrightarrow{(d, \sigma, \sum, \alpha)} \in \mathbb{R}^{2^m}$: Input vector
- $\vec{u} = \overrightarrow{(d, \sigma, \sum, \alpha)} \in \mathbb{R}^{2^n}$: Output vector



- $d \in \mathbb{R}$: Data
- $\sigma \in \mathbb{R}^+$: Standard deviation of d
  - Uncertainty of data

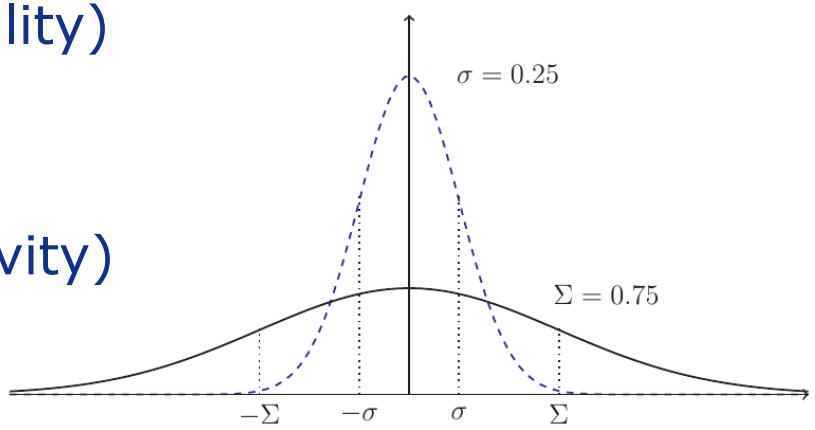
- $\sum \in \mathbb{R}^+$: Target deviation of d
  - Desired / required uncertainty

- $\alpha \in [0,1]$: Relative quality

- $s \in [0,1]$: Stimulation
- $i \in [0,1]$: Inhibition
- $a \in [0,1]$: Activity
- $r \in [0,1]$: Target rating

The iB2C Data Quality

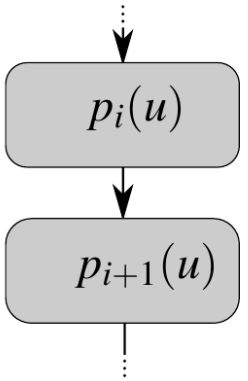
- Data uncertainty is assumed to be Gaussian distributed
  - $p(u) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}\left(\frac{u-\mu}{\sigma}\right)^2}$, with $\mu = 0$
- Absolute Data Quality (current quality)
  - $\sigma \in \mathbb{R}^+$
- Target Data Quality (required quality)
  - $\sum \in \mathbb{R}^+$
- Relative Data Quality (partial activity)
  - $\alpha \in [0, 1]$
  - $\alpha = \min\left(1, \frac{\Sigma}{\sigma}\right) \cdot \iota$



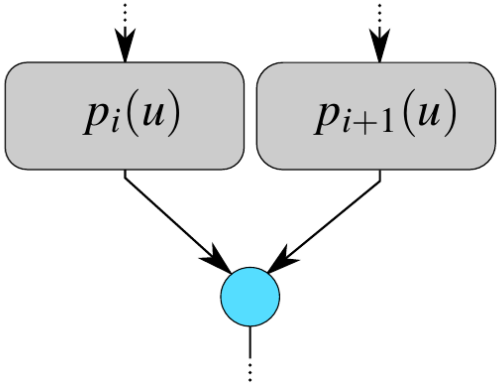
The iB2C Data Quality Combination

Data quality can be combined:

- sequential
  - Multiple percepts degrade the overall quality
  - $\sigma_t = \left(\sum_{i=0}^{N-1} {\sigma_i}^2\right)^{\frac{1}{2}}$
  - $\sigma_t = \left(\sum_{i=0}^{N-1} w \cdot {\sigma_i}^2\right)^{\frac{1}{2}}$ (weighted)

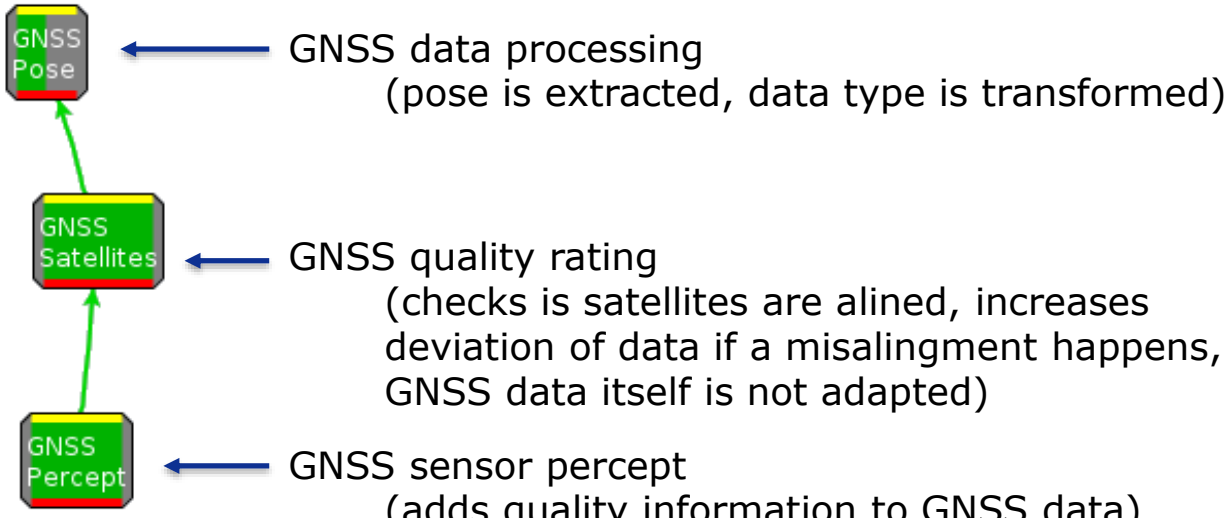


- redundant
  - Multiple percepts evaluate the same quality aspect and average the result
  - $\sigma_t = \frac{1}{\sqrt{N}}\left(\sum_{i=0}^{N-1} {\sigma_i}^2\right)^{\frac{1}{2}}$
  - $\sigma_t = \frac{1}{\sqrt{N}}\left(\sum_{i=0}^{N-1} w \cdot {\sigma_i}^2\right)^{\frac{1}{2}}$ (weighted)

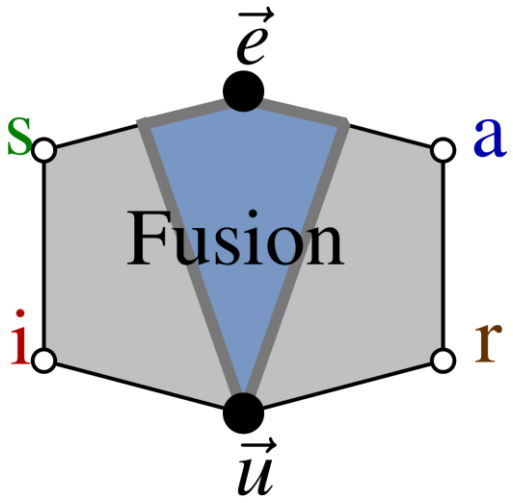


Example: Sequential Quality Evaluation

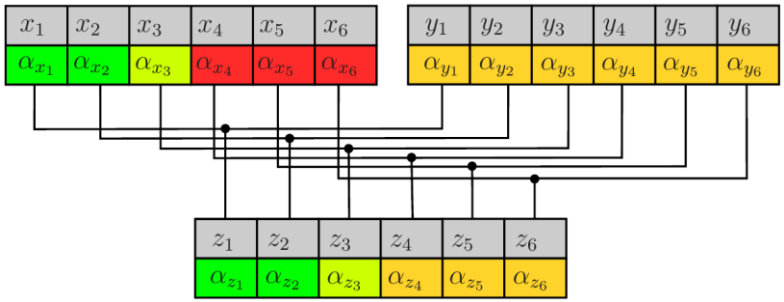
- Differentiation between data processing percepts and quality evaluation percepts
  - Data processing: Data types are transformed and quality information is combined into the new data structure
  - Quality evaluation: Data types are **not** adapted, instead the quality values are adapted.



The iB2C Percept Fusion



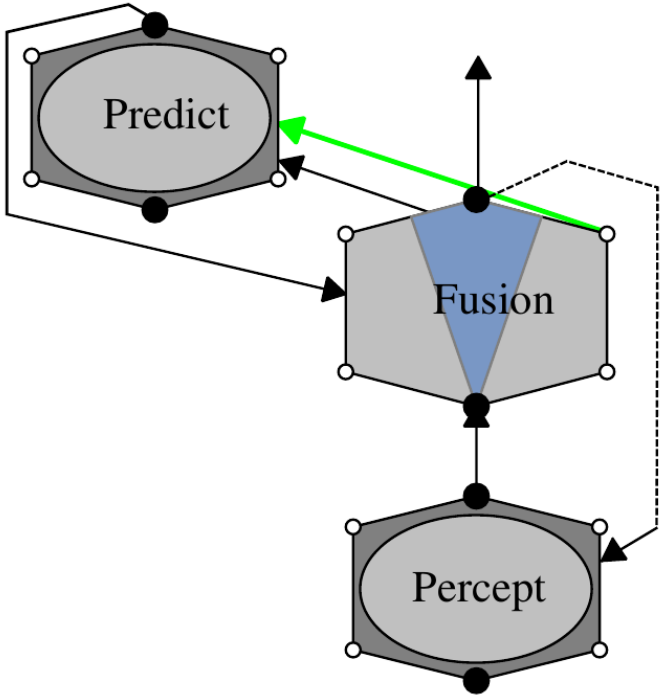
- Similar to standard iB2C fusion
- Fusion is component-based
  - Relative data quality α is applied
- Standard fusion applies per component
  - Maximum
    - $u_f = u_s$ where $s = \max(\alpha_c)$
    - $\sigma_f = \min(\sigma_c)$
    - $r_f = r_s$ where $s = \max(a_c)$
  - Weighted Average
    - $u_f = \frac{\sum_{c=0}^{p-1} \alpha_c \cdot u_c}{\sum_{c=0}^{p-1} \alpha_c}$, $\sigma_f = \frac{\sqrt{\sum_{c=0}^{p-1} \alpha_c \cdot \sigma_c}}{\sum_{c=0}^{p-1} \alpha_c}$
    - $r_f = \frac{\sum_{c=0}^{p-1} \alpha_c \cdot r_c}{\sum_{c=0}^{p-1} \alpha_c}$



Example: Robot Pose Fusion

The iB2C Predictive Fusion Patterrn

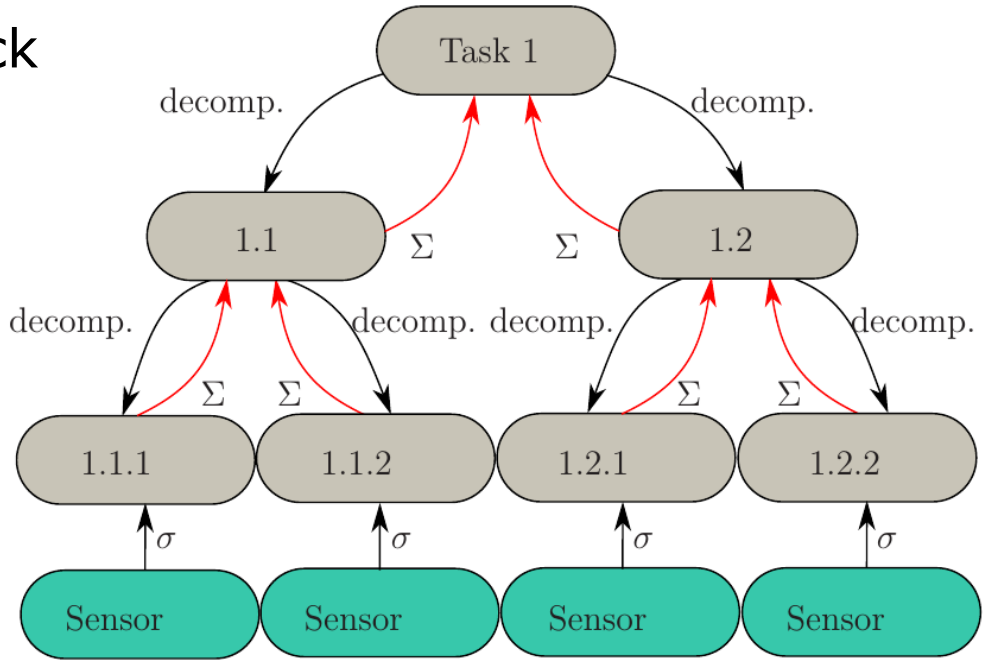
- Percept Fusion is extended by prediction percept and provides feedback to input percepts



- Filter similar to Kalman filter
- IIR characteristic
- High extendability
- Open filter structure
- Tracability
- Non-linear filter (any characteristic can be modeled and is considered by percept fusion)

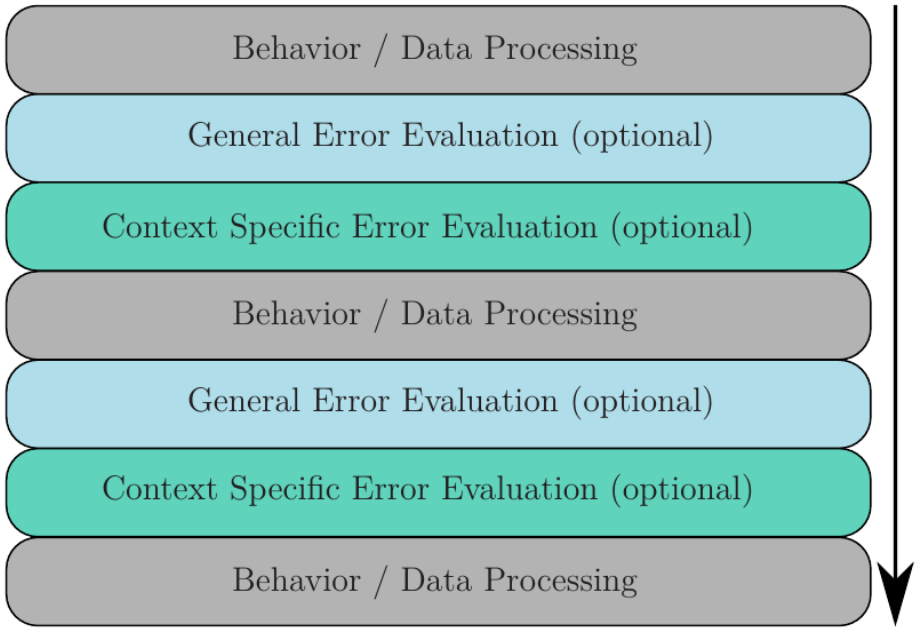
Task Decompostion and Sensor Selection

- Quality information can be used to derive system structure
  - A task has a specific target quality
  - Can the sensors fulfill it
  - Top-Down decomposition
  - Bottom-up condition check



iB2C Layered Percept Design

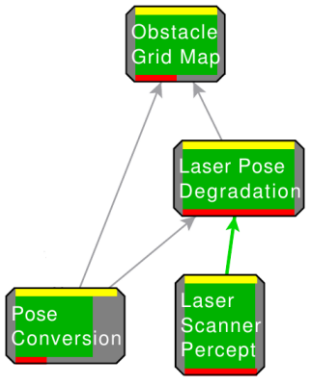
- Percept quality evauation and data processing is separated
  - Layered structure
  - Data processing
  - Error Evaluation
    - General errors
    - Applies to all behaviors, e.g. power loss
    - Context errors
    - Applies to a limited number of behaviors, e.g. GNSS disturbances do not influence local camera processing but localization



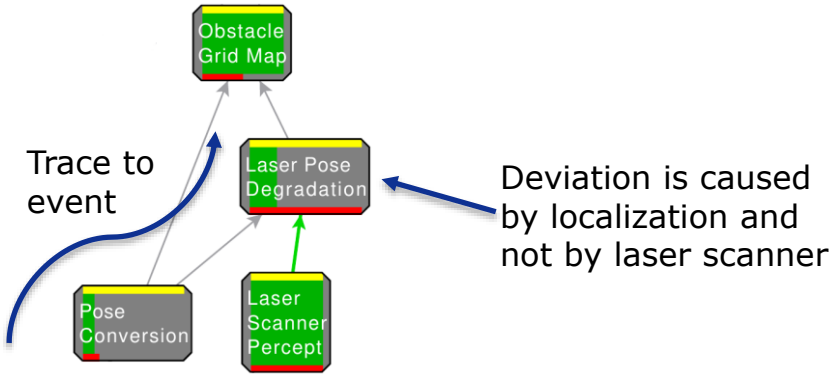
The iB2C Percept – Activity

- Traceablity of quality influencing events in the network
- Stimulate/inhibit based on the quality of sensor data (enable safety behaviors, adapt robot control)

- Example: Insert laser data into a global map



Low deviation of data



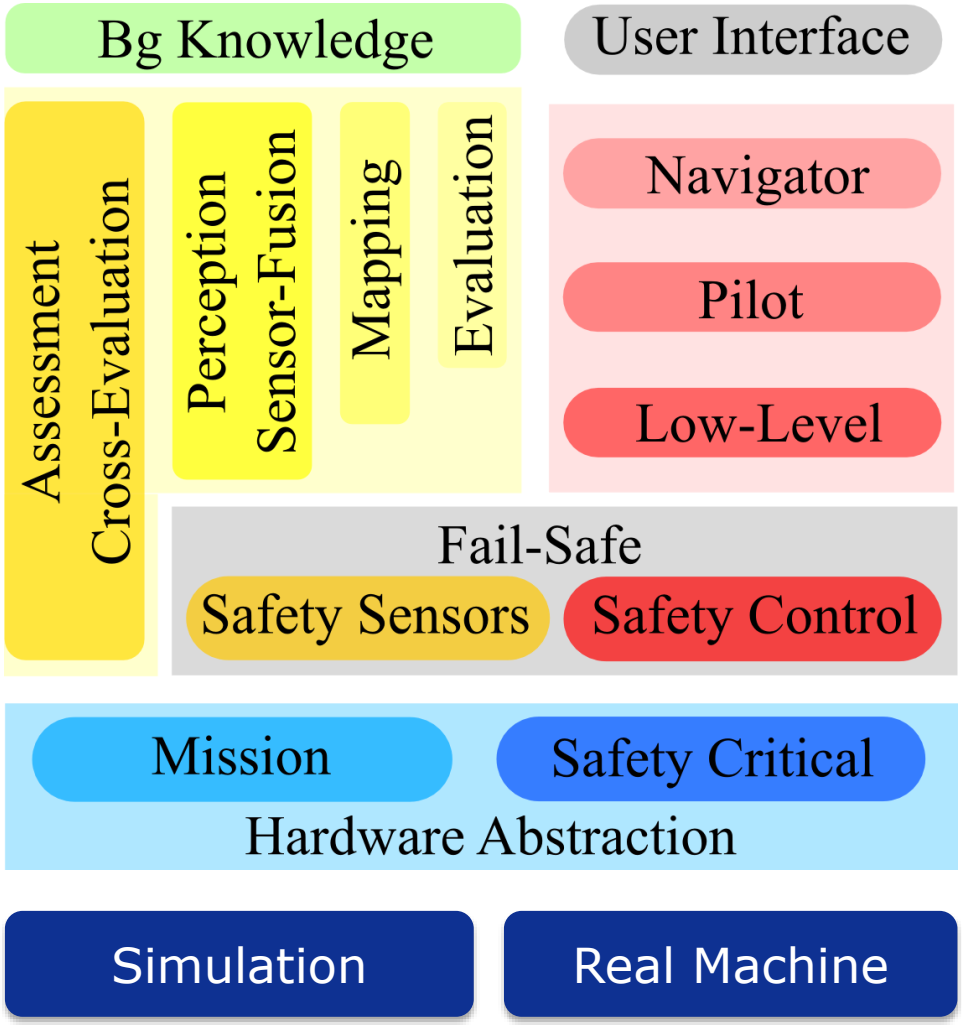
High deviation of data

Applications of iB2C

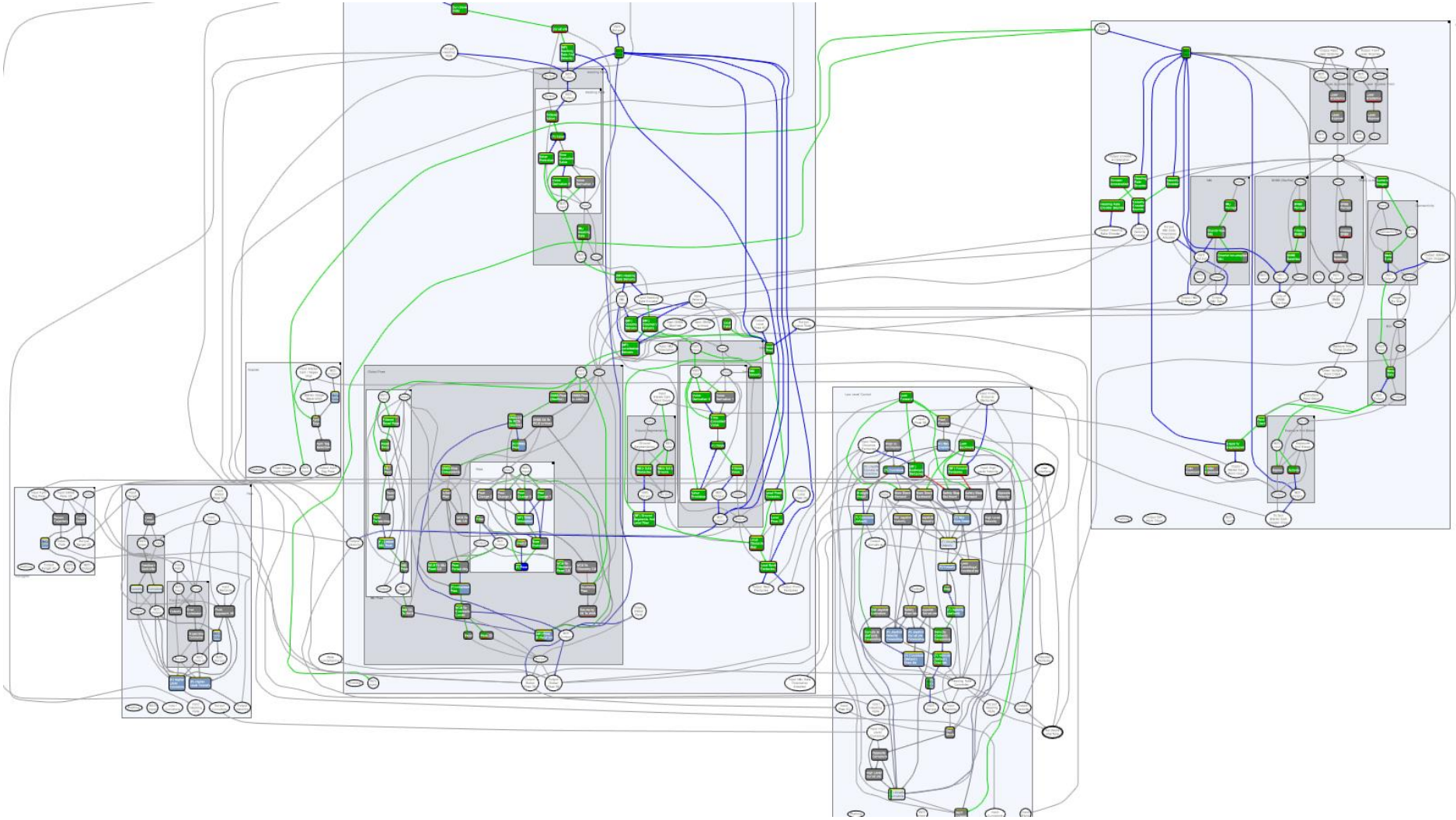
GatorX855D

- Ackermann all wheel drive vehicle
- Sensors:
  - Micro Strain 3DM-GX3-25 (IMU)
  - u-blox NEO-7P (GNSS)
  - Starfire 3000 (GNSS)
  - Velocity (via CAN bus)
  - Heading Rate (via CAN bus)
  - Bumblebee X3 (Stereocam front, 60° opening angle)
  - Bumblebee XB2 (Stereocam front, 120° opening angle)
  - Vislab 3dv (Stereocam rear)
  - 2x ifm ToF Camera (front)
  - 4x SICK Laser Scanner (front, rear, left, right)

Gator Control and Software Architecture



Gator Behavior Network



Hardware

Simulation

Hardware Abstraction

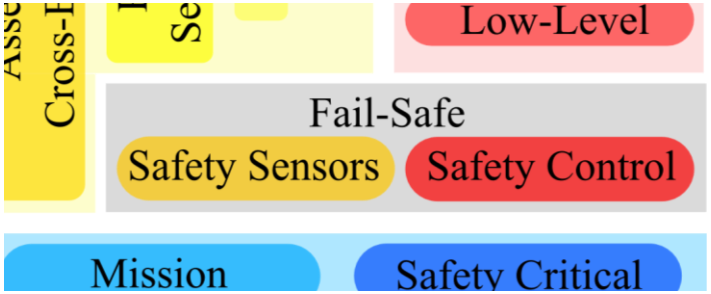
- Encapsulates hardware access + driver management
  - CAN
  - LAN
  - FireWire
  - ...

- Interface to Software Framework

- Separation into Safety-Critical + Mission Hardware
  - Errors in mission related HW (e.g. localization system) does not influence system safety
  - Safety-Critical HW (drive-train control + near field proximity sensors) encapsulated in small + robust part

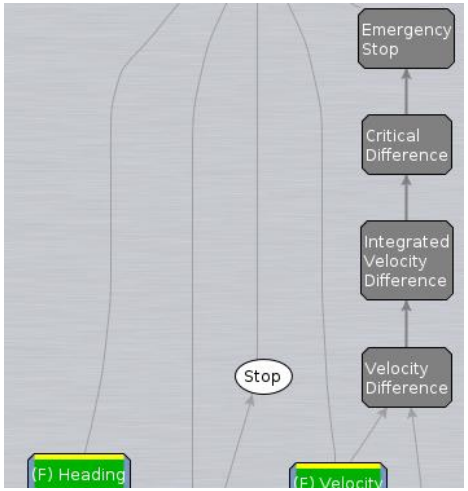
Fail-Safe

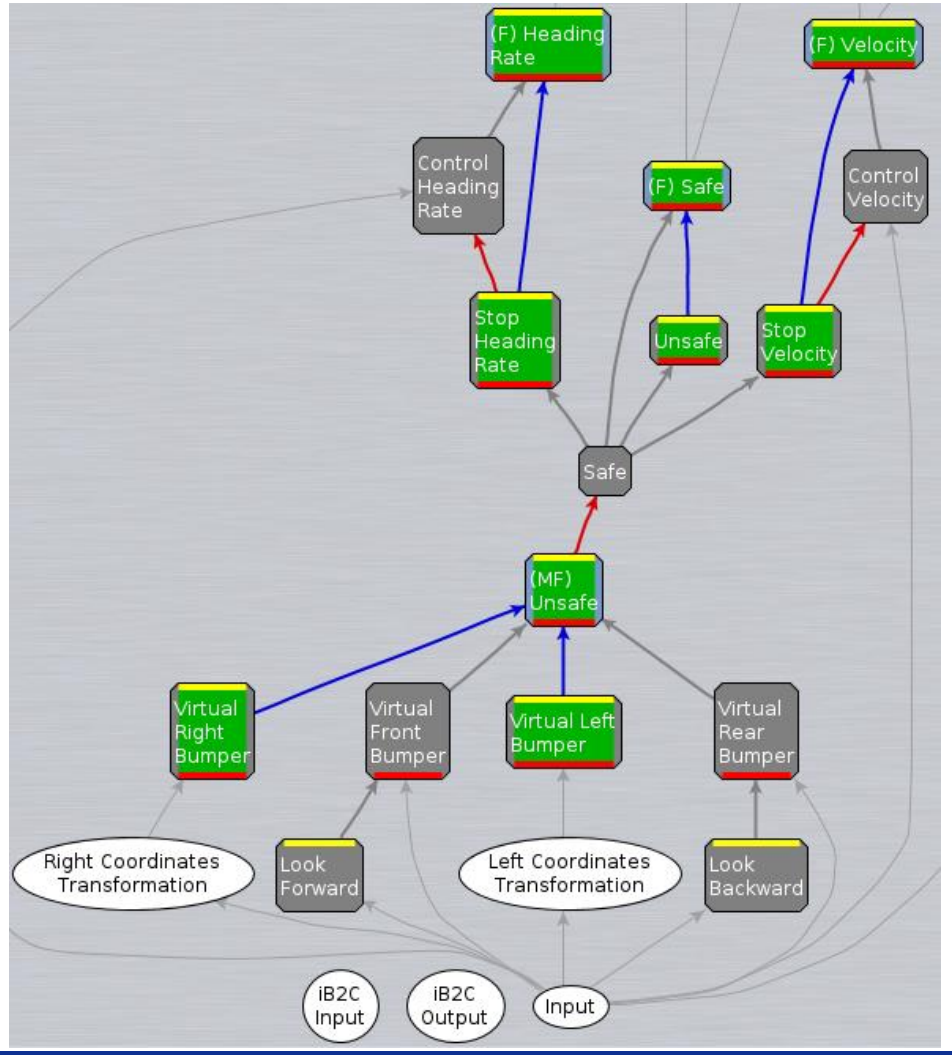
- Low-Level Safety Control
- Monitoring of Safety Zones
- Watchdog for High-Level Control
- Emergency Stop/Evasion
- Limited Hardware Access
- Small + Robust Implementation



Fail-Safe - Network

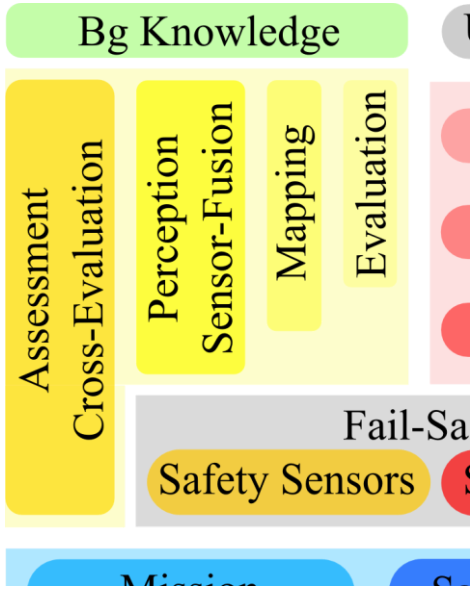
- Virtual bumpers around Gator
  - 4 Sick Laser scanners
- Comparison between desired and perceived velocity
  - Localization sensors





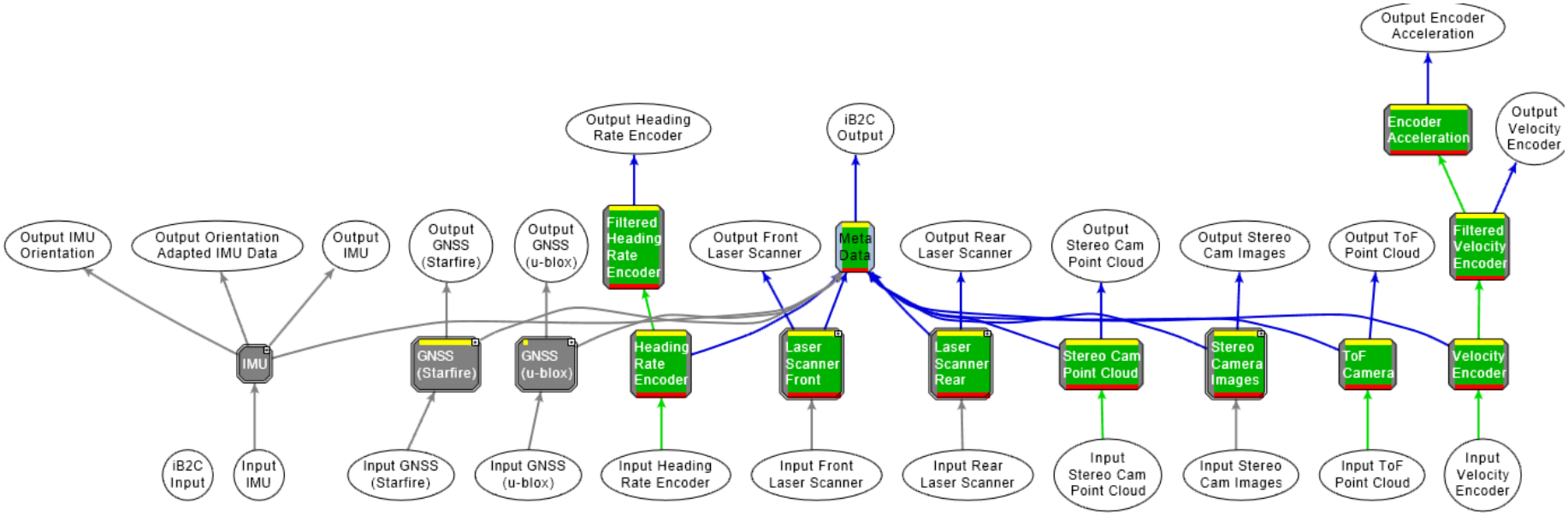
Perception

- Data Quality driven Sensor Processing
  - Assessment on Single Sensor
  - Cross-Evaluation exploiting multiple Sensors
- Sensor Fusion using Quality Data
  - Localization
  - Obstacle detection
- Mapping for Single + Multiple Sensor Fusion exploiting time domain
- Map Evaluation
  - Obstacle detection
  - Drivability
  - Incorporation of background knowledge

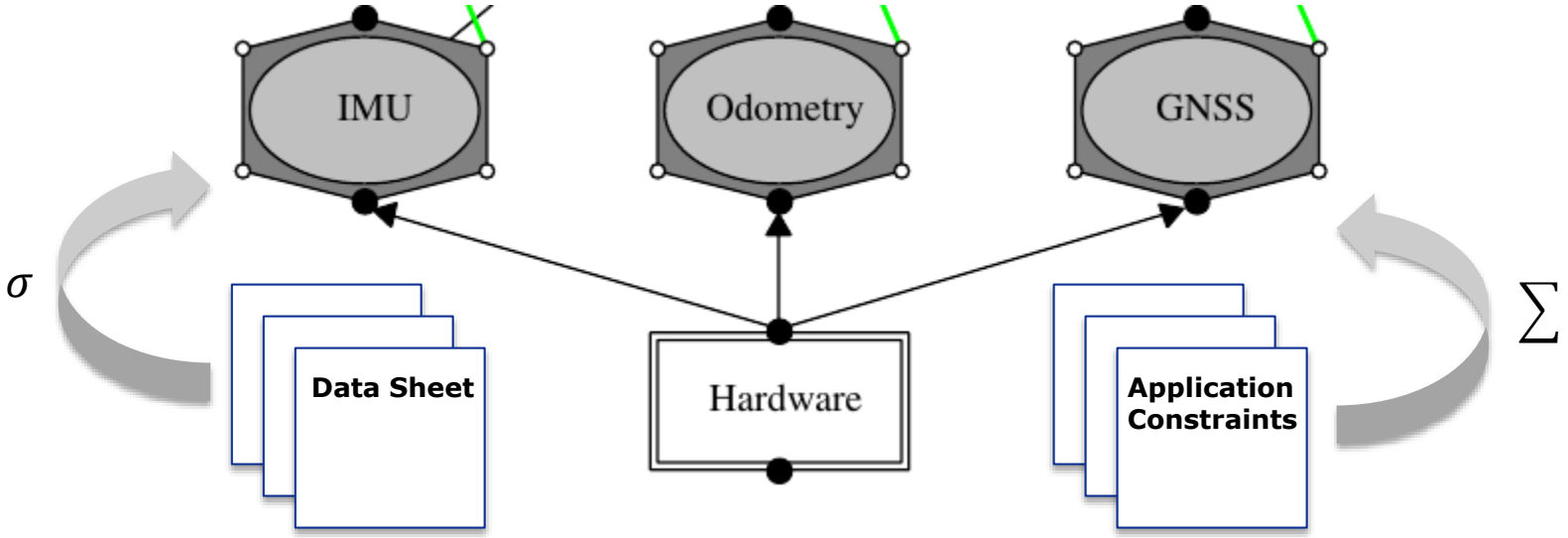


Perception – Sensors / Cross Evaluation

- Quality estimation of sensors
  - Single sensors
  - Cross evaluation with multiple sensors
  - Activity represents current quality
  - Here: IMU and GNSS failed



Perception – Localization I



- Sensors:
  - IMU: $\sigma_{acc_x} = 0.557\, m \cdot s^{-2},\ \sigma_{v_\Psi} = 0.637° \cdot s^{-1}, \ldots$
  - GNSS: $\sigma_x = 1\, m, \sigma_y = 1\, m, \ldots$
  - Velocity: $\sigma_v = 0.1\, m \cdot s^{-1}$
  - Heading Rate: $\sigma_\Psi = 1° \, s^{-1}$

Perception - Localization II

- Add Processing Percepts:
  - Sensor data to desired data type
    - IMU Data -> Pose
    - GNSS Data -> Pose
    - Velocity & Heading Rate -> Pose

- Add Error Evaluation Percepts
  - Odometry can suffer from slippage
  - GNSS quality depends on satellite configuration

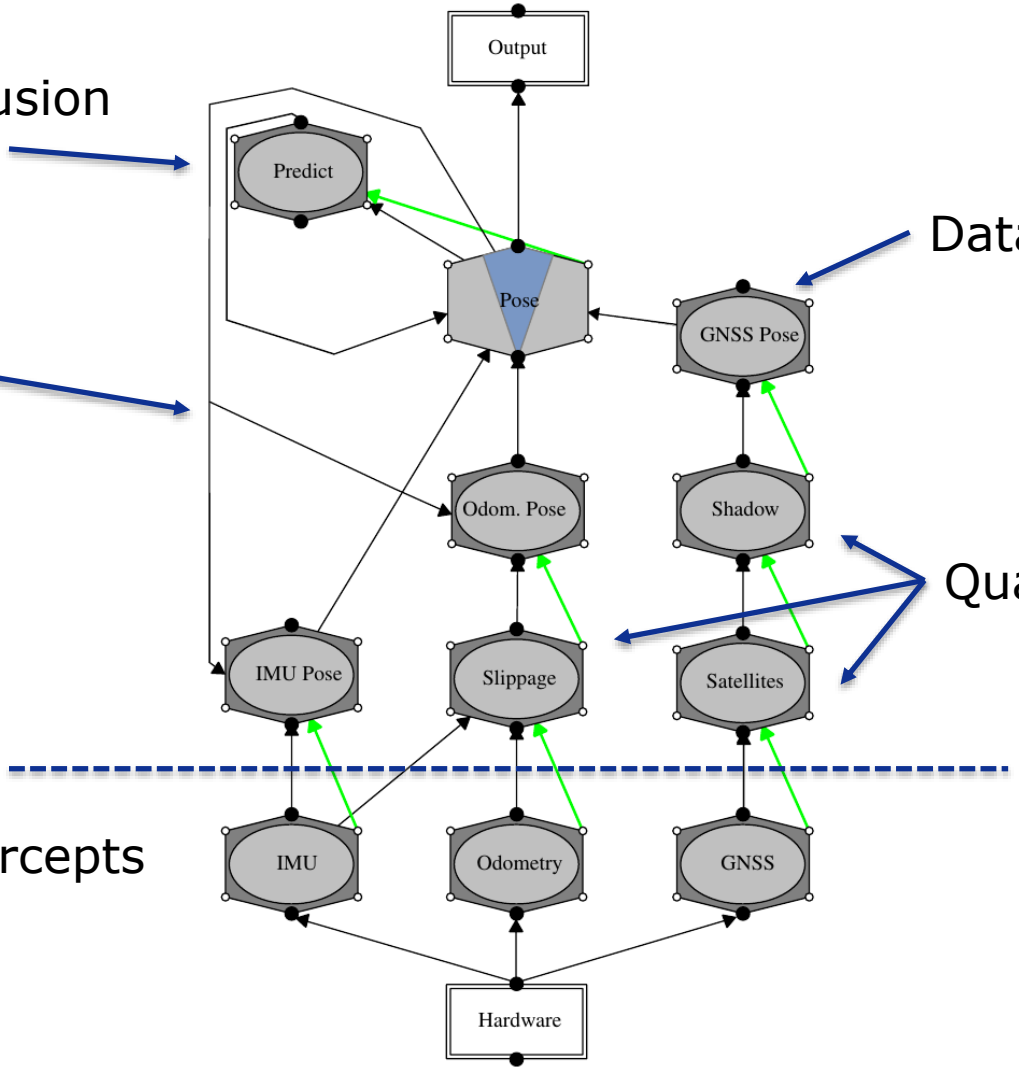
- Quality evaluation percepts do not change data but only quality values!

Perception - Localization III

Predictive Fusion Pattern

Feedback

Sensor percepts



Data processing

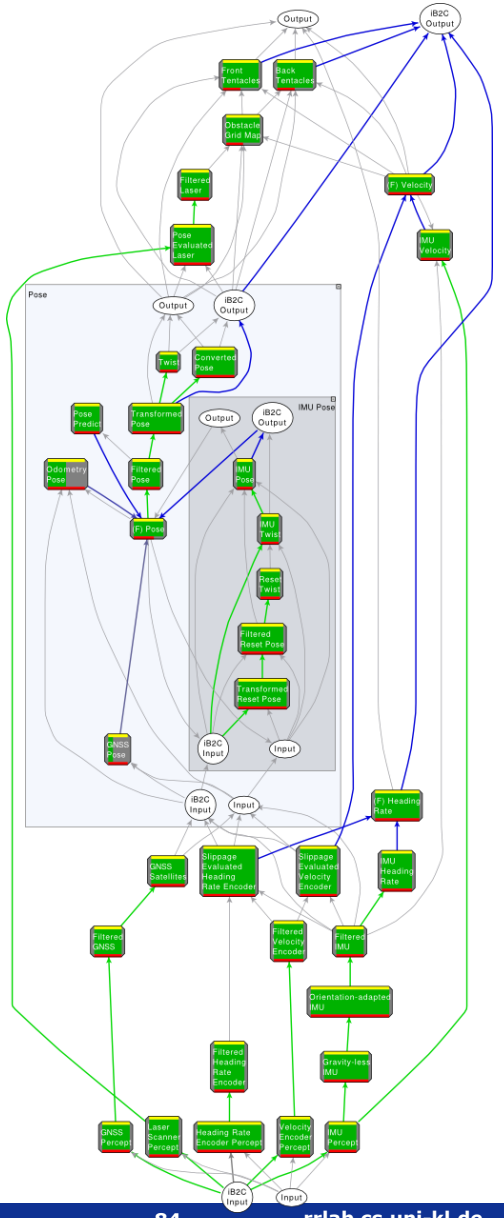
Quality evaluation

Perception - Localization IV

Red: localization result

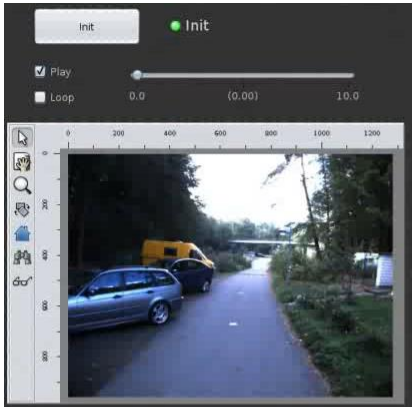
GNSS shadow in tunnel and bridge

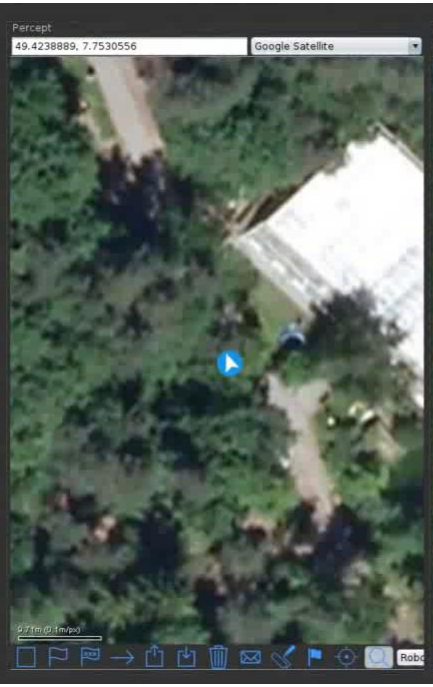
Starting position

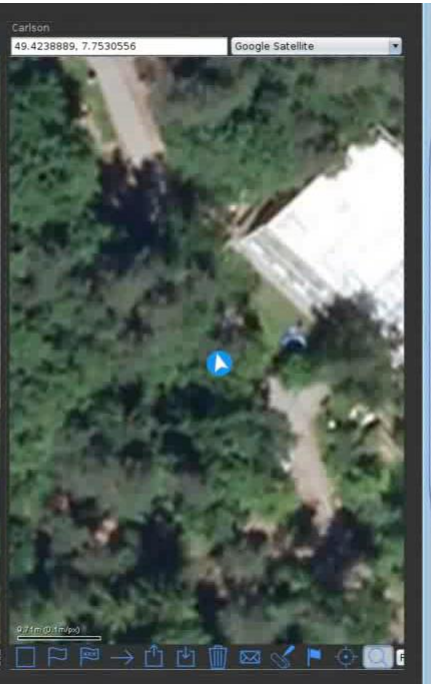


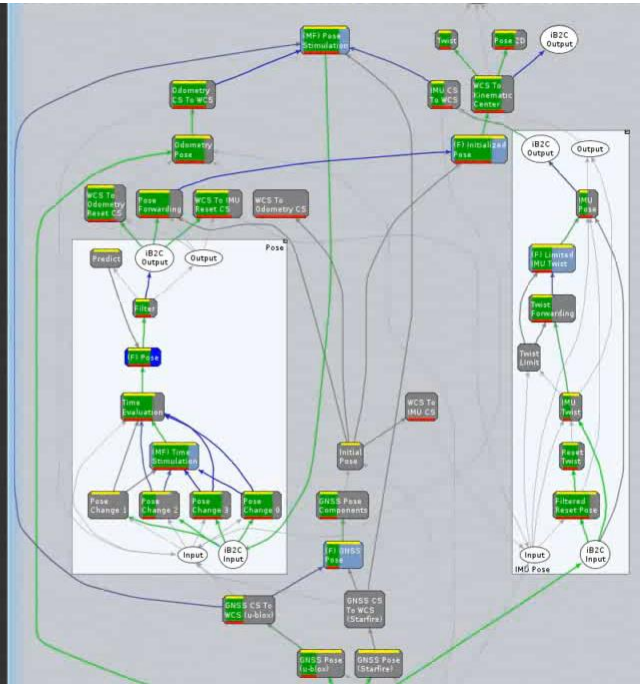
Perception - Localization V

Camera | Percept filter | Carlson filter | Network



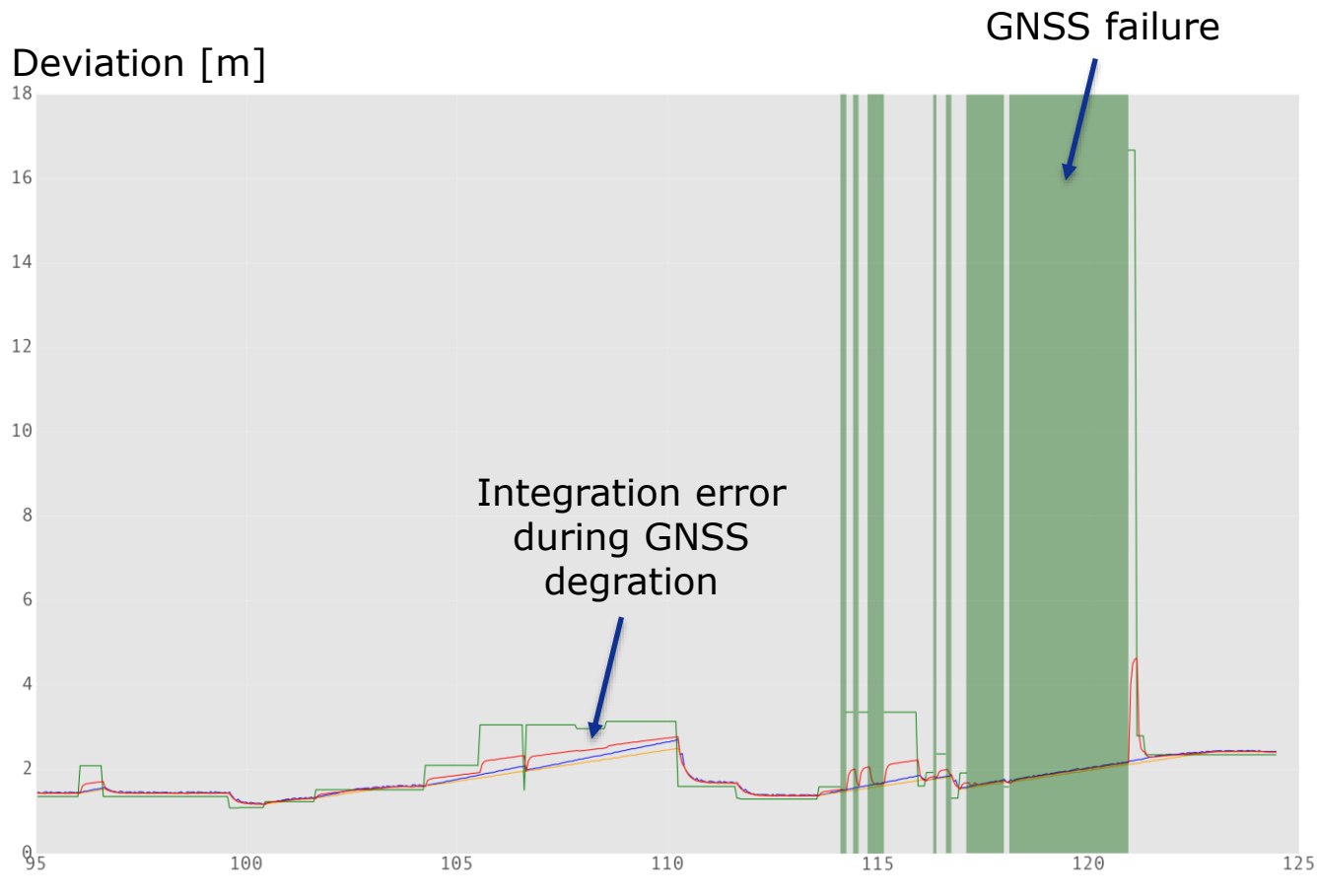






Perception - Localization VI

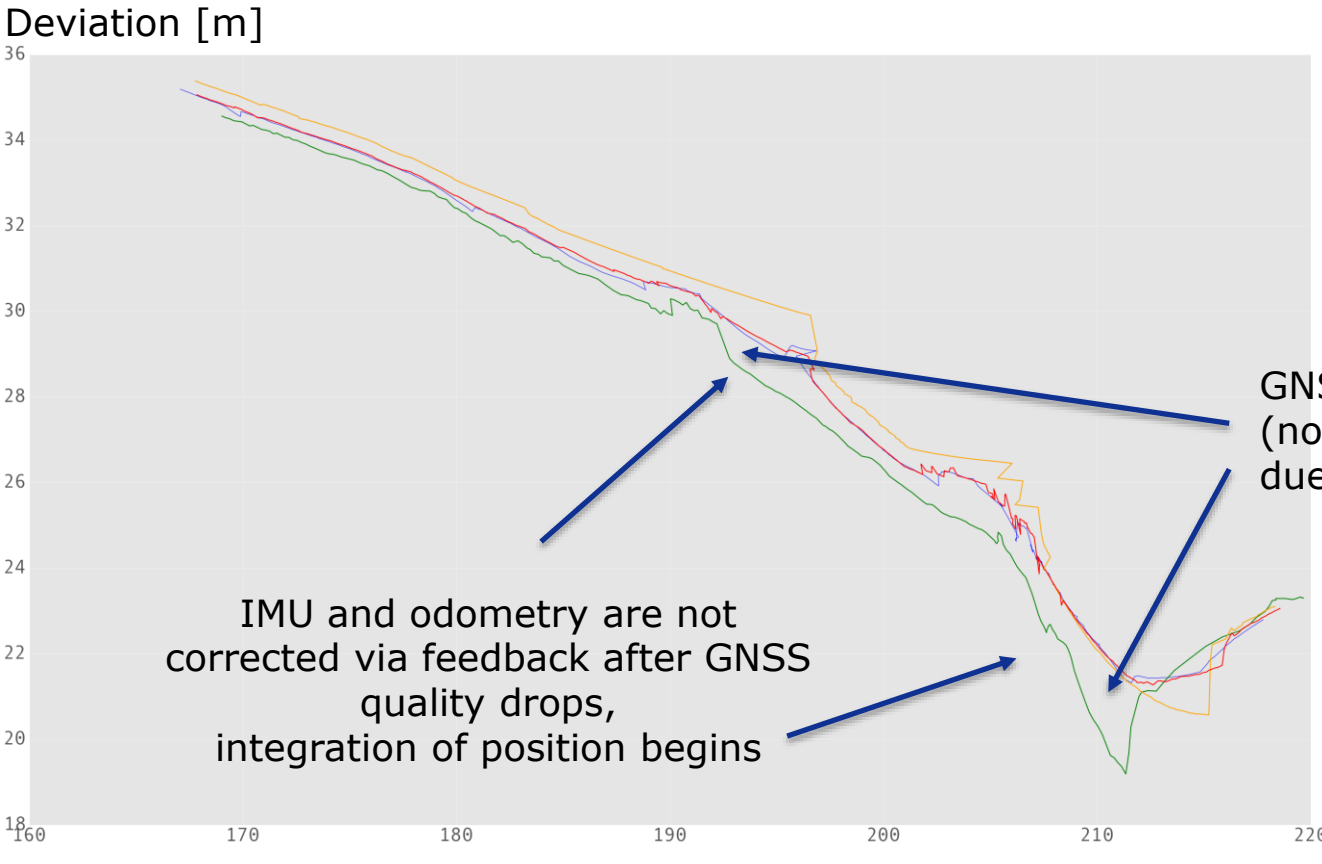
Quality visualization in tunnel



Red: Combined
Green: GNSS
Blue: IMU
Yellow: Odometry

Perception - Localization VII

Spatial localization result

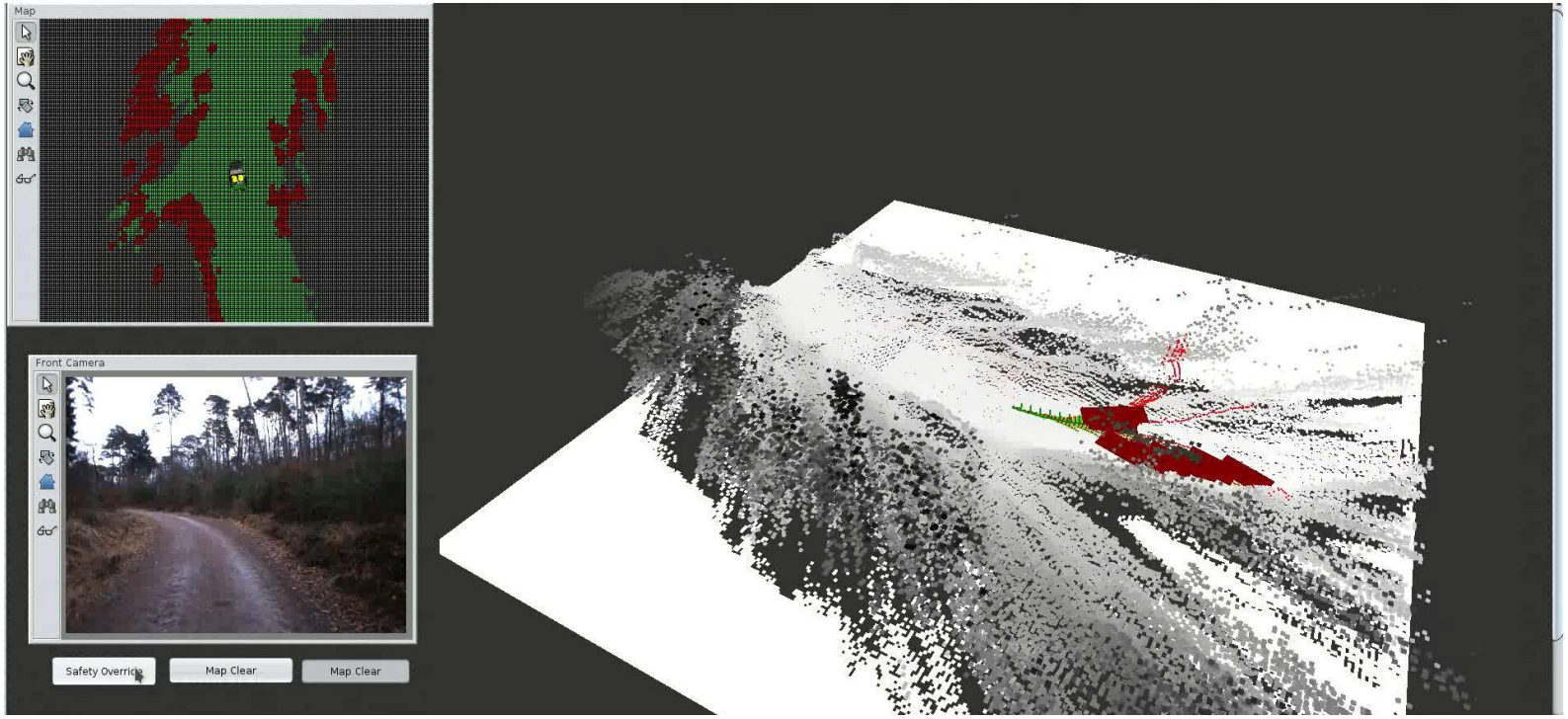


Perception – Mapping I

- Obstacle detection
- Path detection

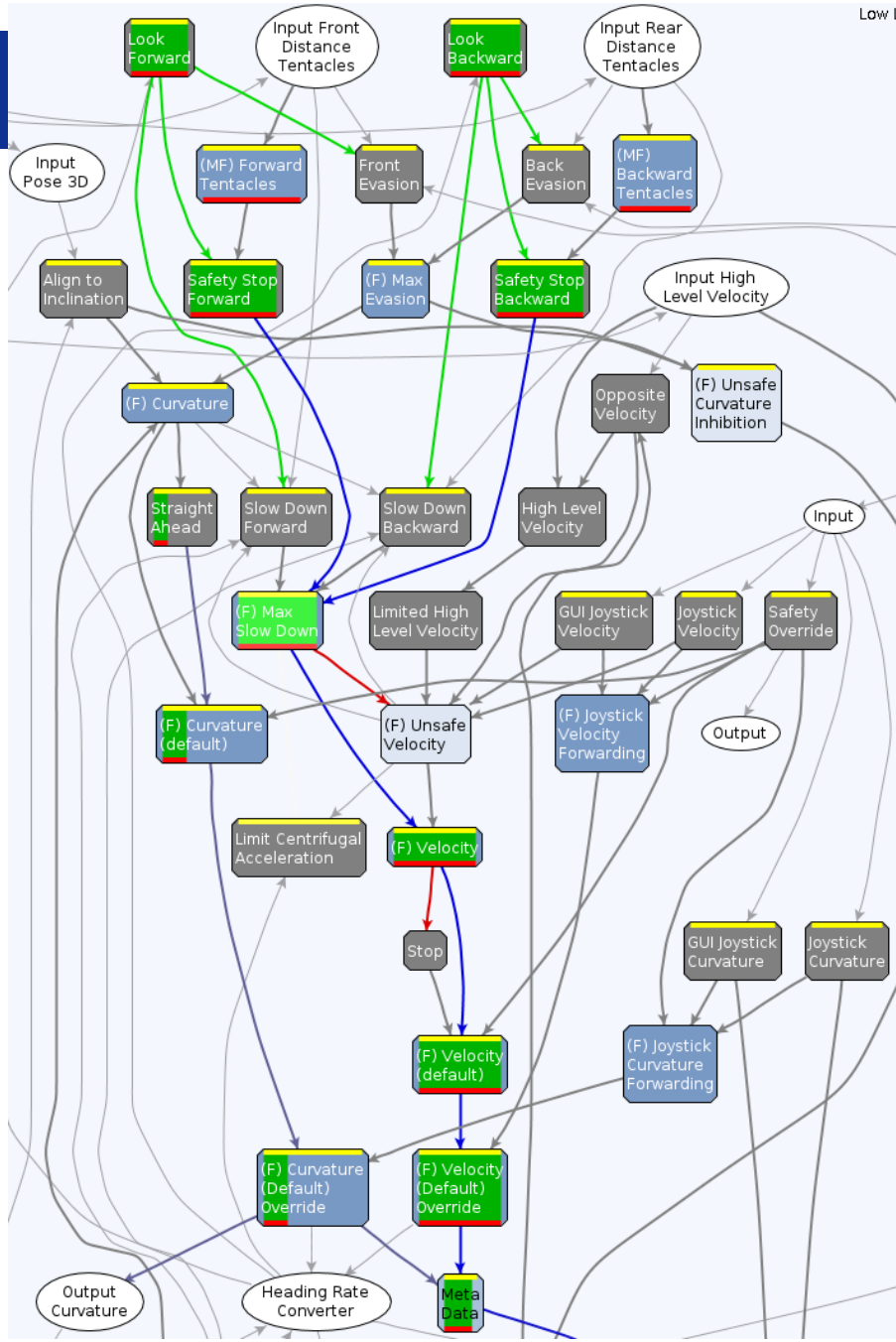
Perception – Mapping II

- 2D Obstacle Map
  - Unvisited
  - Occupied
  - Free
- 2.5D Map
  - Contact points with frame
  - Average z-Coordinate



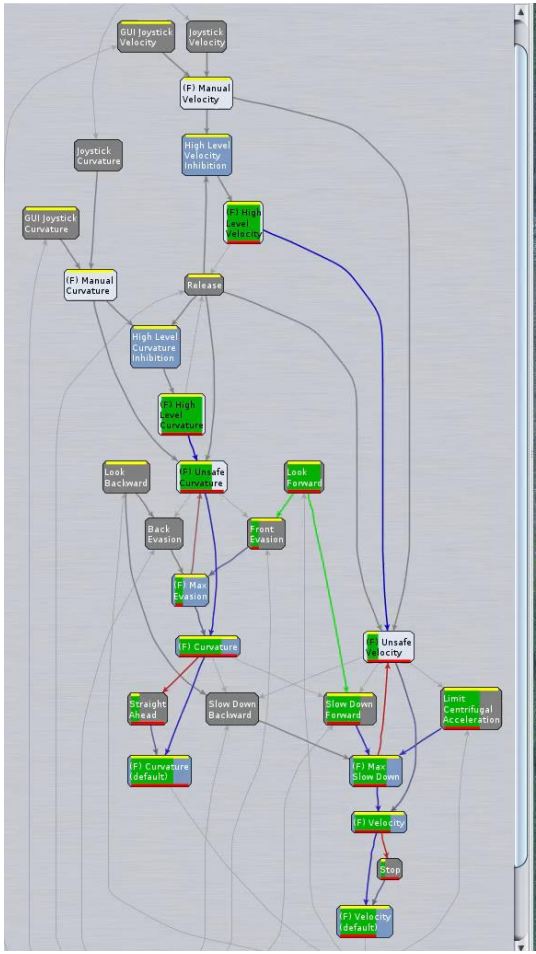
Low Level

- Reactive Low-Level Control provides commands to final safety component
  - Collision avoidance
  - Obstacle evasion on multiple sensors
  - Roll-Over prevention



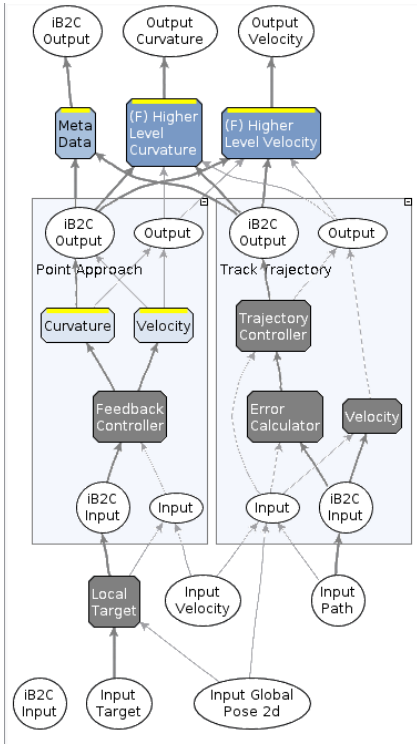
Low Level – Collision Avoidance and Got-Stuck

- Gator approaches an obstacle
- Collsion avoidances becomes active
- Gator stops
- Got-stuck gets active
- Gator drives backward until path is free and got-stuck releases vehicle



Pilot

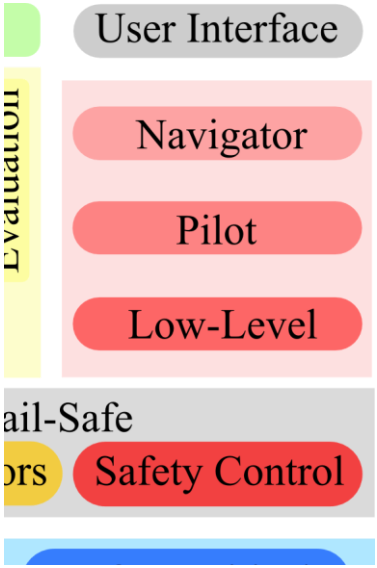
- Point approach
- Trajectory control



- Gator tracks front vehicle and generates trajectory
- Pilot follows trajectory (yellow)

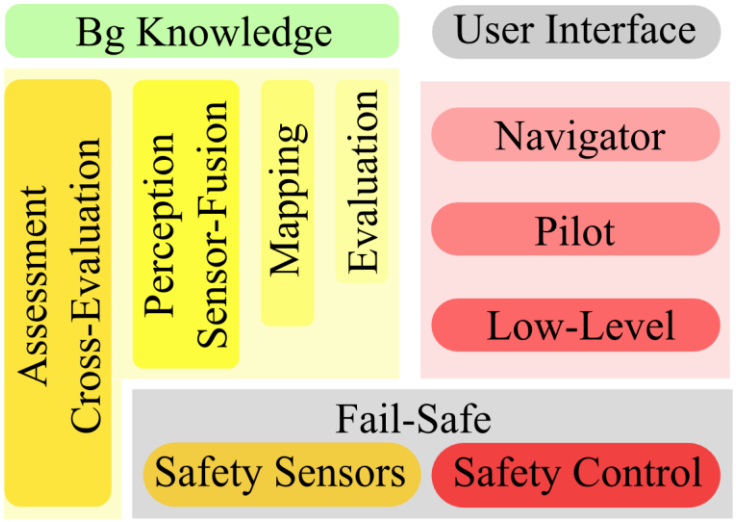
Navigator

- Path planning
- Intermediate and final targets
- Trajectories to follow



User Interface

- Abstract state visualization
  - Sensor failures
  - Maps + Evaluation
  - Safety Control Interventions
- Manual control
  - Override
  - Assisted driving
  - Mission goal selection



Lecture

Lecture

- What was good/bad?
- Obscurities?
- Structure?
- …
- Courses Survey of the Dept. of Computer Science (https://vlu.cs.uni-kl.de)

Thank You