What is Cognition?

- “All cognitive systems are symbol systems. They achieve their intelligence by symbolizing external and internal situation and events, and by manipulating those symbols”
- “reasoning, perception, language, memory, control of movement”
- “cognitive functions → reasoning, learning and planning”
- “cognitive science was focused on identifying the mechanisms and processes that intervene between the recording of sensory data and the production of decisions and actions… . Although later reformulations of the research agenda emphasized the need to better incorporate behavior as an action-perception loop… .”

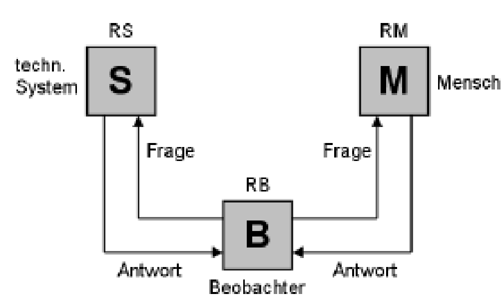
What is Artificial Intelligence?

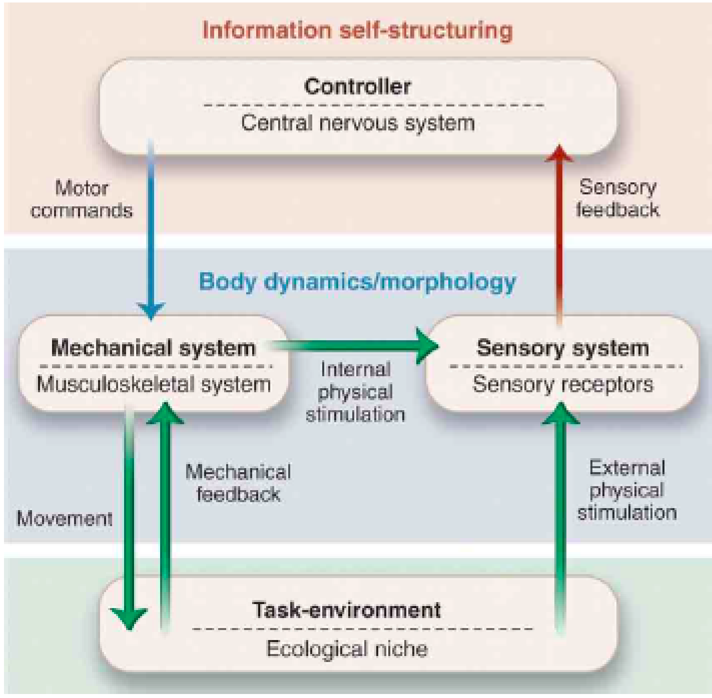
- The construction of intelligent systems
- Formalization and representation of knowledge and reasoning based on that knowledge
- Development and use of computational models to understand humans and artificial agents
- Build the bases for natural human-system interaction on each level (common sense to expert use)

Systems that think like humans

- Top-down approach : **Cognitive System**
  - Theories about internal activities in the human brain
  - What level(s) of abstraction are appropriate?
- Bottom-up approach : **Neurosciences**
  - Understanding of natural neural network still a challenge
  - Sensing of brain activity improving (skin, fMRI, implanted neural interfaces)

- **Turing-test** : Imitation game using a screen/keyboard interface to communicate with the other agent
  - Goal : identify whether the communication partner is human or a machine
  - Contains many key aspect of AI:
    - natural language processing
    - knowledge representation
    - (logical interface)
    - learning
  - “Total Turing Test” also includes
    - computer vision
    - robotics





Cognitive Systems
Lehrstuhl Informatik VI – Robotics and Embedded Systems

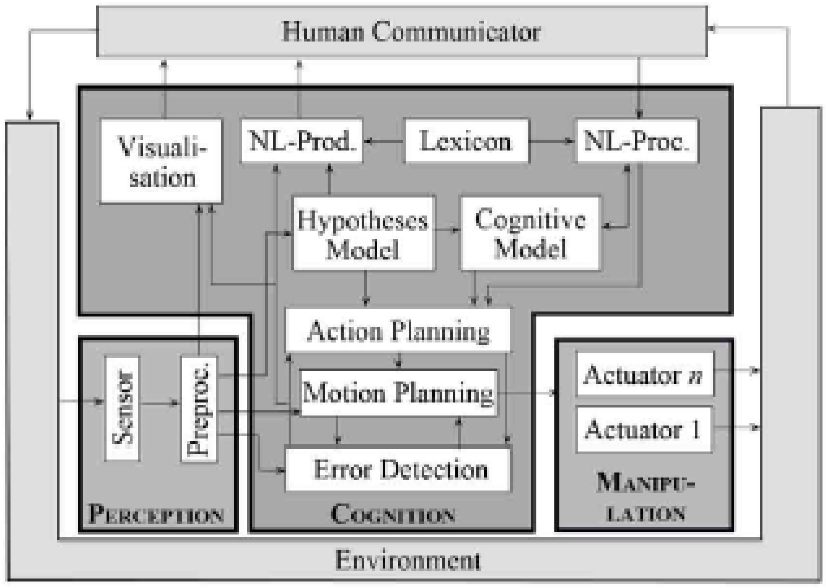
Some thought about humans and machines

Technical Systems	**Humans**
High Speed, accuracy, forces	Slower, less accurate, less powerful
Fast Feedback	Slower Feedback
Less adaptive	Highly adaptive to unforeseen situations
Typically specific power and sensitivity	Large range: power vs. sensitivity
Typically for specific purpose	“Universal” capabilities

Branch of Cognitive Systems: Neurorobotics

- **What is Neurorobotics?**
    - “A robotic system comprised of a controller, a body, actuators and sensors, whose controller architecture is derived from a model of the brain”
- **Why Neurorobotics?**
    - better understanding of the brain
    - building useful machines → Understanding through building

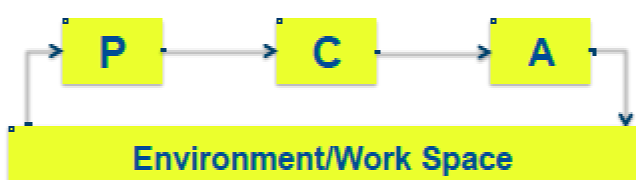
Basic Architecture of Robot System



- Communication and Adaptation between
  - Robot ←→ Human instructor/co-worker
  - Robot ←→ Environment

Future-Oriented Car Architecture will Look Similar

- For automatic driving adoption of architecture from robotics for high-level functions will be necessary
  - sensor processing
  - data fusion and data association services
  - redundancy (management)
  - precise timing services
  - unified communication services
  - trajectory planning
- Characteristics : multitude of sensors of different working principles
  - real-time reaction necessary → fast feedback
  - different levels of abstraction → direct control reaction up to high-level strategy



  - Perception (all kinds of sensors) → Cognition (all kinds of process) → Action (all kinds of actuators)
  - Static & dynamic modularity with respect to different aspect,
    - selection of information sources

Tasks:

- **Look at data first before you start implementing you robot system**
- **“What do humans do?”**
- **Make your robots behavior predictable**
- **robot should say what it does**
- **Think about when and what sequence your robot takes an action and how trajectories are determined**

Cognitive Architecture

What is a Model?

- Models are abstractions of real world systems or implementations of hypothesis to investigate particular questions about, or to demonstrate particular features of a system or hypothesis
  - descriptive models → What?
  - mechanistic models → How?
  - interpretive models → Why?

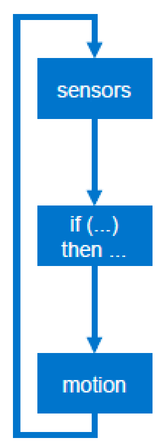
Models…

- Meaning of Models in our context → *Mathematical representations of system dynamics*
  - Models allow the dynamics to be simulated/analyzed without having to build the system
  - Models are never exact, but can be predictive
- Models can be used in ways the system cannot
  - Certain types of analysis cannot be done in a real system
  - Resource efficiency
- The model to use depends on the questions asked
  - A system has many models
  - Choice of time and space scale
  - Always formulate questions before building a model

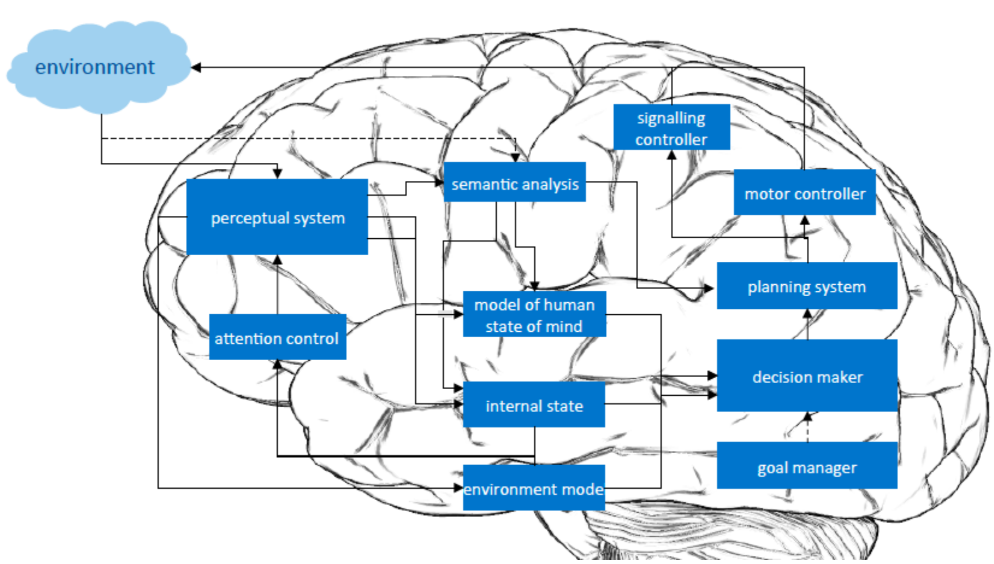
Cognitive Architecture as models of cognition

The Early Ages of Robotics

- string everything together in a single program
  - not good
  - hard to maintain
  - no flexibility
  - not scalable



Solution → build a system based on Cognitive Architecture



Definitions : A Cognitive Architecture

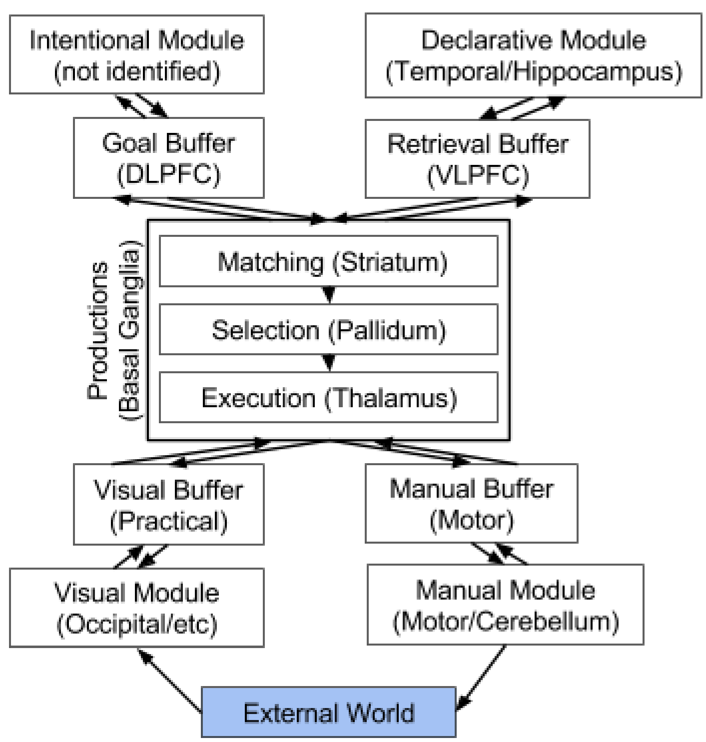
- “... is a broadly-scoped, domain-genetic computational cognitive model, capture the essential structure and process of the mind, to be used for a broad, multiple-level, multiple-domain analysis of behavior”
- “... is a specification of the structure of the brain at a level of abstraction that explain how it achieves the function of the mind”
- “... specifies the underlying infrastructure for an intelligent system. Briefly, an architecture includes those aspects of a cognitive agent that are constant over time and across different application domains”

Types of Cognitive Architectures

“Mind models”	“**Technical Architectures**”
• abstraction of the human mind • psychological and AI background • lots of research, esp. in psychology • research goal → Theory of Cognition	• emulation of human cognition • focus on robotics application • relatively new research topic • research goal → Smarter Robots

CA Examples: ACT-R

- cognitivist mental model
- successor of ACT
- integrates several major branches of current psychol. AI research.
- uses “buffers” to store and retrieve rule-based “productions”
- no parallel processing
- perception and motor control added recently



Cognitive Architecture Paradigms

- **Cognitivists**
  - treat cognition as a form of (possibly stochastic) computation
  - ie. the processing of (symbolic) information by means of algorithm and logic
- **Emergent systems**
  - mostly based on self-organization, do not rely on many assumption about the environments and develop cognitive capabilities as they evolve
- **Hybrid systems**
  - exploit the speed and robustness of emergent systems
  - by complementing them with the clear structure and programmability of cognitivist systems

Different Approaches to Cognition

	Cognitivist	Connectionist	Dynamical	Enactive
What is cognition?	Symbolic computation : rule-based, manipulation of symbols	The emergence of global states in a network of simple components	A history of activity that brings forth change and activity	Effective action : history of structural coupling that enacts a world
How does it work?	Through any device that can manipulate symbols	Through local rules and changes in the connectivity of the elements	Through the self-organizing processes of interconnected sensorimotor subnetworks	Through a network of interconnected elements capable of structural changes
What does a good cognitive system do?	Represent the stable truths of the real world	Develop emergent properties that yield stable solutions to tasks	Become an active and adaptive part of an ongoing and continually changing world	Become a part of an existing world of meaning or shape a new one

Paradigm Pros & Cons

Cognitist	Emergent	Hybrid
• easy to include prior knowledge • easy analysis of “thought processes” • complex tasks possible • tend to be less flexible • manual programming of facts/behaviors • slowed down a lot be accumulated symbolic knowledge	• closer to biology • few assumptions, evolve capabilities on their own • flexible, robust and fast • hard to extract learned knowledge and rules • prior knowledge hard to include • complex tasks difficult	• “best of both worlds” • probably most suitable for future robots • difficult to find common ground of paradigms • overall behavior not well understood

Example: Hierarchical Temporal Memory

→ **Emergent mental model**

- not really a complete CA (yet), but rather a “cortex algorithm” forming the basis of a new theory of cognition
- main idea:
    - cortex is a hierarchical association and prediction engine all signals reaching the brain (external senses, internal reflection) are just patterns hence there is no need for “artificial” functional submodules, everything is treated the same way for a CA, would need to add “old brain” function, esp. drives

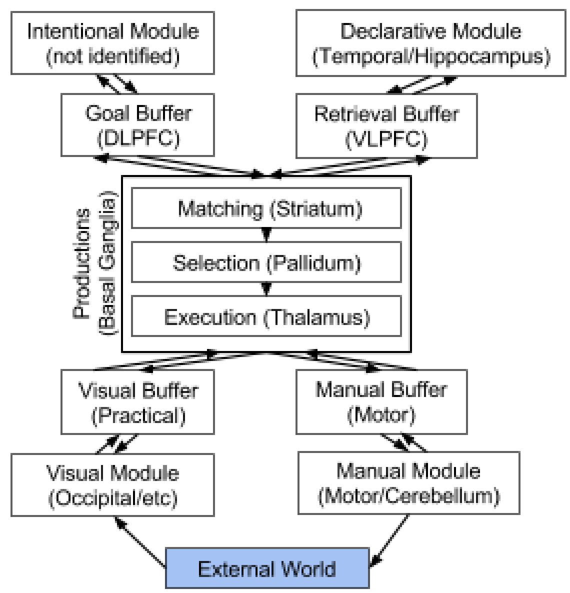
Cognitive Architecture ACT-R

Goals of this Unites

- To get an overview of ACT-R
    - Main concept & background
    - Implementation
- To gain basic experience in cognitive modeling with ACT-R
- To be able to model simple chunks and productions

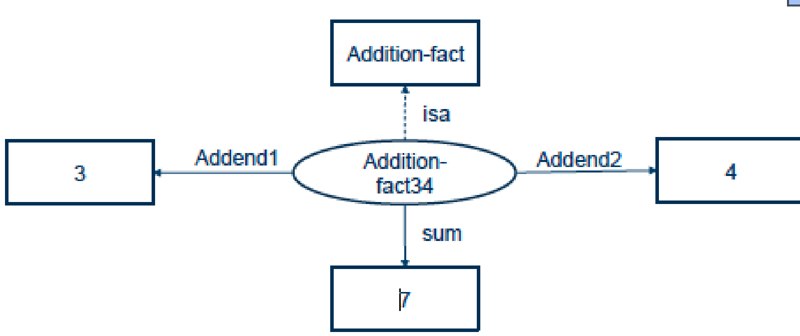
CA Examples: ACT-R

- cognitivist mental model
- successor of ACT
- integrates several major branches of current psychol. AI research.
- uses “buffers” to store and retrieve rule-based “productions”
- no parallel processing
- perception and motor control added recently



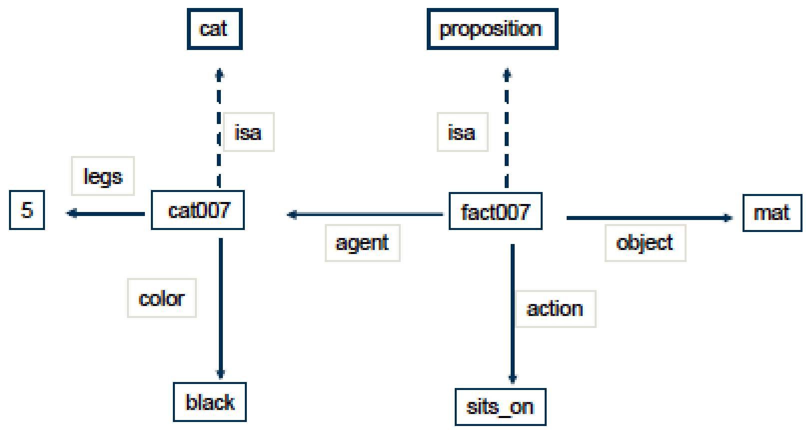
Declarative Memory - chunks

A chunk represents a fact
ex : 3+4=7



Modeling of more complex facts

ex : The black cat with 5 legs sits on the mat



Productions

- Rules for modifying chunks
  - Buffer tests ⇒ Buffer modifications
- Several productions can fulfill the test conditions
  - selection of most suitable production needed
  - use of **Subsymbolic** mechanisms
  - utility equation estimates the relative cost and benefit of each production
  - production with the highest utility is selected

What, if more than one chunk fulfils the buffer test of a productions?
Again, subsymbolic decision : Chunk activation

Past experiences indicates usefulness at the particular moment:

- **Base - level** : general past usefulness
- **Associative Activation :** relevance to the general context
- **Matching Penalty :** relevance to the specific match required
- **Noise :** stochastic is useful to avoid getting stuck in local minima

Implementation - Practical use of the ACT-R software

Knowledge representation in ACT-R

- **Types of knowledge**
  - declarative knowledge - chunks ⇒ conscious, facts can be described
  - procedural knowledge - productions ⇒ unconscious, like use of language
- **Chunks**
  - represent facts of different complexity
    - ex : “George Washington was the first president of the US” / “1+1=2”
  - define by a
    - **chunk-type** (similar to category, such as bird)
    - **slots** (attributes, such as color or size)
  - syntax
    - chunk-type name slot-name-1 slot-name-2 … slot-name-n
  - special slot ISA to specify the type of a chunk

```
Action023
  isa chase
  agent dog
  object cat
Fact3+4
  isa addition-fact
  addend1 three
  addend2 four
  sum seven
```

Defines a chunk (fact) named Action 023
Defines the chunk-type
Defines the slot values

  - creating chunks:
    - 1st step : definition of **chunk-type**

```
(chunk-type bird species color size)
(chunk-type column row1 row2 row3)
(chunk-type count-order first second)
```

    - 2nd step : declaration of **facts**

```
Action023
  isa chase
  agent dog
  object cat
Fact3+4
  isa addition-fact
  addend1 three
  addend2 four
  sum seven
```

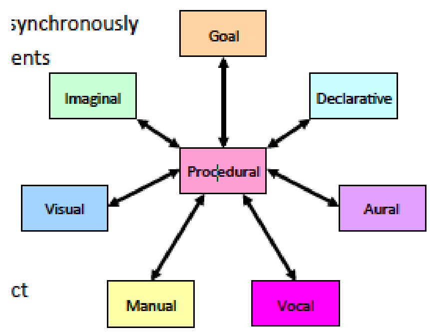
Defines a chunk (fact) named Action 023
Defines the chunk-type
Defines the slot values

- **Productions**
  - statements that control behavior
  - represented as IF… THEN… rules
  - EXAMPLE in english

IF the goal is to classify a person
and he is unmarried } Conjunction of features to be tested
THEN classify him as a bachelor } Operation to be performed, if selected

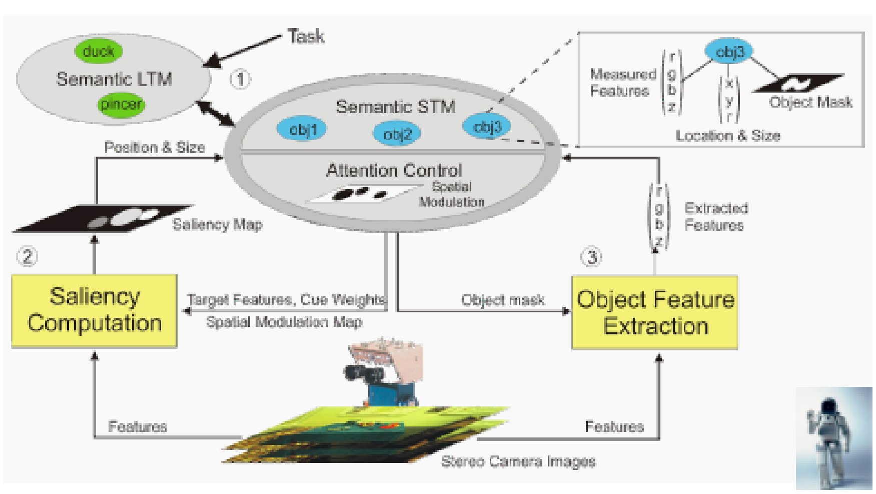
IF the goal is to add two digits d1 and d2 in a column
and d1 + d2 = d3
THEN set as a subgoal to write d3 in the column

- **Models and Buffers**
  - Modules
    - represent independent units, work asynchronously
    - responsible for scheduling its own events
    - main modules are: **Declarative** memory, **Visual** module, **Manual** module
  - Buffers
    - can hold one **chunk** representing a fact
    - interface of the modules
    - chunk is copied into buffer and modified locally



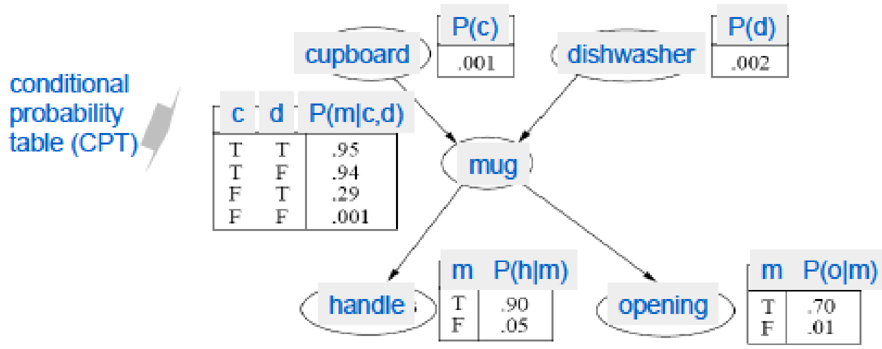
Motivations for a Cognitive Architecture

- **Philosophy:** provide a unified understanding of the mind
- **Psychology:** account for experimental data
- **Education:** provide cognitive models for intelligent tutoring systems and other learning environment
- **Human Computer Interaction:** evaluate artifacts and help in their design
- **Neuroscience:** provide a framework for interpreting data from brain imaging

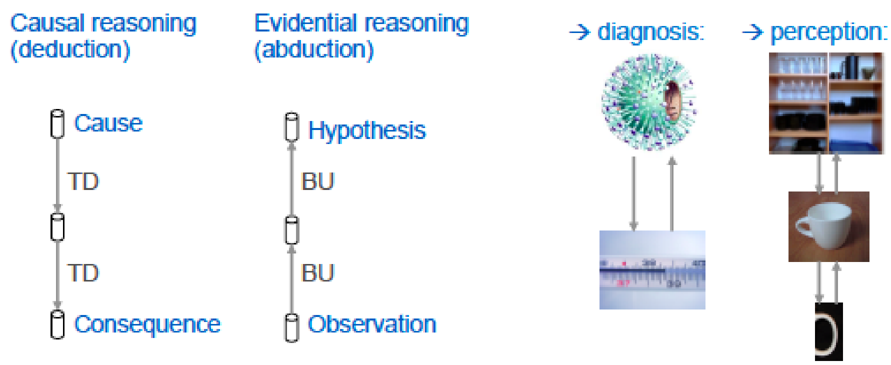


Structure of Bayesian Networks

- Likelihood functions → Encoding of sensory information
- Priors → Constraints on possible scenes



Interface in Bayesian Network



Task: compute posterior for a set of query variables given some event

query variables

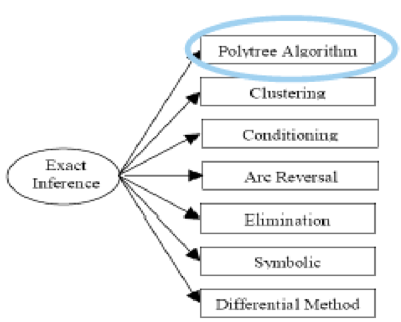
$$p(X|e) = \alpha \sum_{y} p(X, e, y)$$

hidden variables

evidence variables

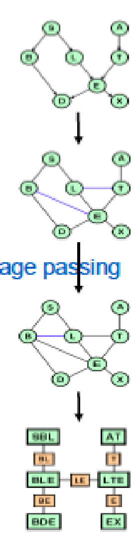
Exact Interface Procedures

Inference in Bayesian Networks is NP-hard
- Critical: width of largest clique
- Restrictions → e.g. polytrees
- Approx. inference (anytime algo's)



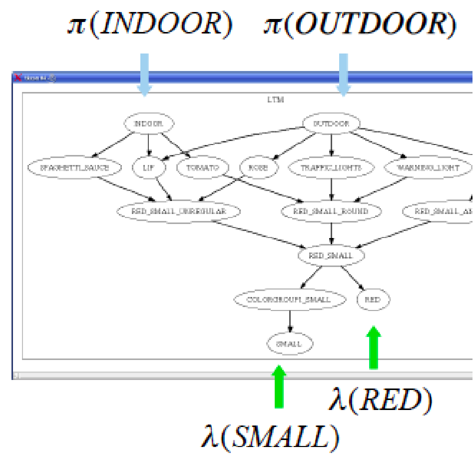
= Belief propagation, Message passing

= Clique-tree propagation



How Interface Works?

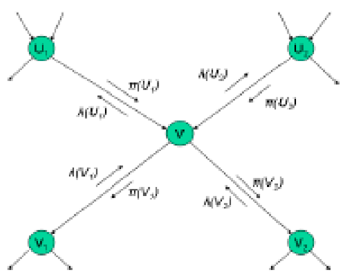
- Which kind of knowledge enters the Bayesian Network?
  - Statistic information
    - structural dependencies
    - conditional probabilities
  - Dynamic information
    - priors for root nodes
    - observation / measurements
- What do you get?
  - posterior of all variables of interest

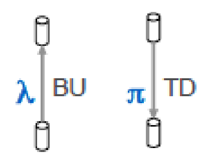


Bayesian Belief Propagation

$$BEL(x) = \alpha\lambda(x)\pi(x)$$

$$\lambda(x) = \prod_j \lambda_{Y_j}(x) \qquad \pi(x) = \sum_{u_1,\ldots,u_n} P(x \mid u_1,\ldots,u_n) \prod_i \pi_X(u_i)$$

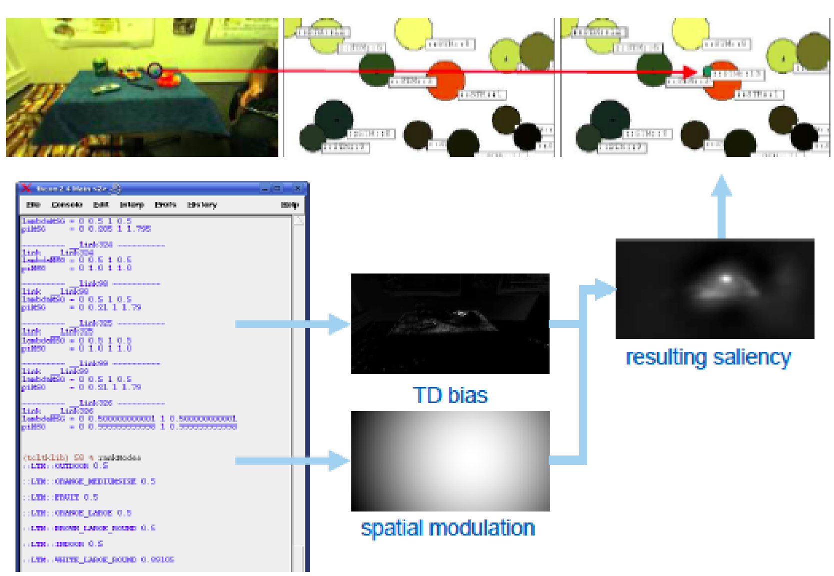




$$\pi_{Y_j}(x) = BEL(x) / \lambda_{Y_j}(x)$$

$$\lambda_X(u_i) = \beta \sum_x \lambda(x) \sum_{u_k : k \neq i} P(x \mid u_1,\ldots,u_n) \prod_{k \neq i} \pi_X(u_k)$$

TD Saliency Modulation



Assumption about Cognitive Architecture

- A Cognitive Architecture
  - specifics the infrastructure that holds constant over domains. as opposed to knowledge, which varies
  - focuses on functional structures and processes, not on the knowledge or implementation levels
  - commits to representations & organizations of knowledge and processes that operate on them (mostly)
  - comes with a programming language for encoding knowledge & constructing intelligent systems
  - should demonstrate generality and flexibility rather than success on a single application domain

Cognitive Architecture ACT-R - Part2

ACT-R Syntax

- Production : Syntax

```
(p Name "optional documentation string"
  buffer tests
==>
  buffer changes and requests
)
```

- Buffers :
  - interfaces between the modules of ACT-R
  - can hold one chunk at a time
  - are modified by production rules

- Production : Syntax
  - variables start with a " = ", e.g. `=num1` or `=goal`, scope is the individual production
  - names of buffers to be tested or modified are followed by a " > ", e.g. `=goal>`
- Tests :
  - the slot value of a buffer can be tested for a constant or a variable
    - `isa count` or `number =num1`
  - negating the logical value of a test can be done by a " - " preceding the slot name
    - `- number -num1`

- Buffer modifications:
  - `=buffername>`
    `slotname value`
  - The respective slot of that buffer is immediately modified
  - `+buffername>`
    …
    - request to the buffer's module, can be different, here used only to retrieve a chunk from the declarative memory which matches the given specification
    - in other contexts visual attention, manual control requests, and some other types of actions can be requested

---

(P counting-example	**English Description**
=goal>	If the goal chunk is
isa count	of the type count
state incrementing	the state slot has the value incrementing
number =num1	there is a number we will call =num1
=retrieval>	and the chunk in the retrieval buffer
isa count-order	is of type count-order
first =num1	the first slot has the value =num1
second =num2	and the second slot has a value =num2
==>	Then
...	...

(P counting-example	**English Description**
=goal>	If the goal chunk is
......	...
==>	Then
=goal>	change the goal
number =num2	to continue counting from =num2
+retrieval>	and request a retrieval
isa count-order	of a count-order chunk
first =num2	to find the number that follows =num2
)	

Example 1 :

- **Counting**

Setting up the model:

```
(chunk-type count-order first second)
(chunk-type count-from start end count)
(add-dm
      (b ISA count-order first 1 second 2)
      (c ISA count-order first 2 second 3)
      (d ISA count-order first 3 second 4)
      (e ISA count-order first 4 second 5)
      (f ISA count-order first 5 second 6)
      (first-goal ISA count-from start 2 end 4))
(goal-focus first-goal)
```

- **production “start”, “increment”, “stop”** are needed!

<table>
<tr><th>start</th><th>increment</th><th>stop</th></tr>
<tr>
<td><pre>
(p start
  =goal>
      ISA count-from
      start =num1
      count nil
==>
  =goal>
      count =num1
  +retrieval>
      ISA count-order
      first =num1
)
</pre></td>
<td><pre>
(P increment
  =goal>
      ISA count-from
      count =num1
      - end =num1
  =retrieval>
      ISA count-order
      first =num1
      second =num2
==>
      ...

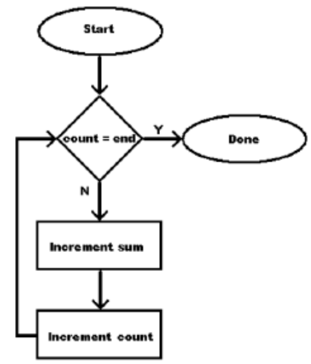
(P increment
      ...
==>
  =goal>
      count =num2
  +retrieval>
      ISA count-order
      first =num2
  !output! (=num1)
)
</pre></td>
<td><pre>
(P stop
  =goal>
      ISA count-from
      count =num
      end =num
==>
  -goal>
  !output! (=num)
)
</pre></td>
</tr>
</table>

Example 2 :

- **Addition**
  - idea :
    - Add two numbers, `num1` and `num2`
    - start from `num1`
    - increment `num2` times
  - declarations
    - declarative memory holds similar facts like in the previous example

      ```
      (chunk-type add arg1 arg2 sum count)
      (second-goal ISA add arg1 5 arg2 2)
      ```

      - note that sum and count are not given, they are nil which can be tested
    - Productions needed “
      - `Initialize-addition`
      - `Increment-sum`
      - `Increment-count`
      - `Terminate-addition`



- **production “initialize-addition”**

(P initialize-addition	**English Description**
=goal>	If the goal is
ISA add	to add the arguments
arg1 =num1	=num1 and
arg2 =num2	=num2
sum nil	but the sum has not been set
==>	Then
=goal>	change the goal
sum =num1	by setting the sum to =num1
count 0	and setting the count to 0
+retrieval>	and request a retrieval
isa count-order	of a chunk of type count-order
first =num1	for the number that follows =num1
)	

(P terminate-addition	**English Description**
=goal>	If the goal is
ISA add	to add
count =num	and the count has the same value
arg2 =num	as arg2
sum =answer	and there is a sum
==>	Then
=goal>	change the goal
count nil	to stop counting
)	

- **production "increment-sum"**

Code	English Description
`(P increment-sum`	**English Description**
`  =goal>`	If the goal is
`     ISA add`	to add
`     sum =sum`	and the sum is =sum
`     count =count`	and the count is =count
`     - arg2 =count`	and the count has not reached the end
`  =retrieval>`	and a chunk has been retrieved
`     ISA count-order`	of type count-order
`     first =sum`	where the first number is =sum
`     second =newsum`	and it is followed by =newsum
`==>   ...`	Then

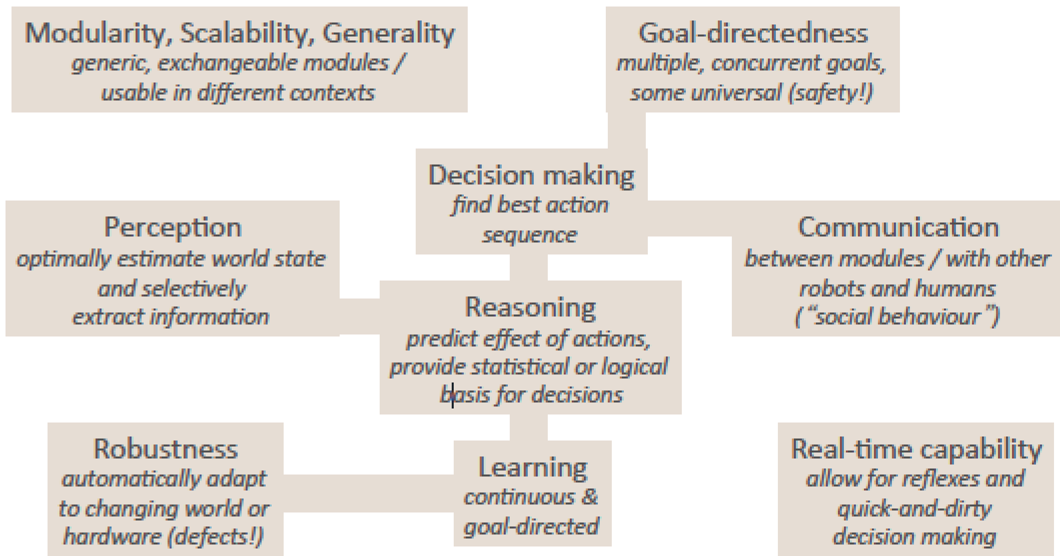
Code	English Description
`(P increment-sum`	**English Description**
`     ...`	
`==>`	Then
`  =goal>`	change the goal
`     sum =newsum`	so that the sum is =newsum
`  +retrieval>`	and request a retrieval
`     isa count-order`	of a chunk of type count-order
`     first =count`	for the number that follows =count
`)`	

- **production "increment-count"**

Code	English Description
`(P increment-count`	**English Description**
`  =goal>`	If the goal is
`     ISA add`	to add
`     sum =sum`	and the sum is =sum
`     count =count`	and the count is =count
`  =retrieval>`	and a chunk has been retrieved
`     ISA count-order`	of type count-order
`     first =count`	where the first number is =count
`     second =newcount`	and it is followed by =newcount
`==>`	Then
`     ...`	

Code	English Description
`(P increment-count`	**English Description**
`     ...`	
`==>`	Then
`  =goal>`	change the goal
`     count =newcount`	so that the count is =newcount
`  +retrieval>`	and request a retrieval
`     isa count-order`	of a chunk of type count-order
`     first =sum`	for the number that follows =sum
`)`	

Design Goals for Technical Cognitive Architecture

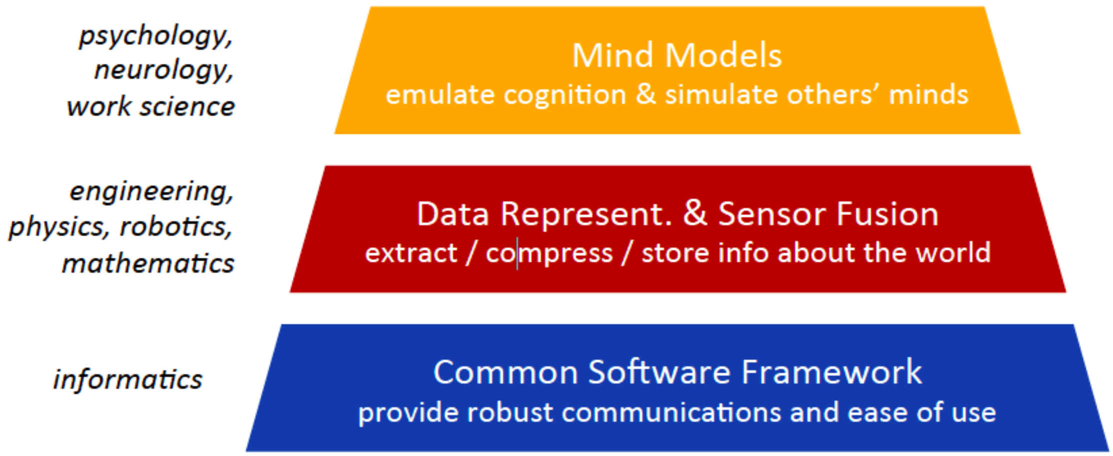


Evaluation of CAs

- difficult for mental models
  - depends on the underlying theory of mind and the specific research goal somewhat easier for technical CAs, several practice-relevant benchmark parameters can be defined:
- **Taskability**
  - ability of a system to adapt to novel problems without human intervention
  - Measured by exposing CA created for a given task to an unknown new task and measuring the performance in this transfer task
- **Incrementality**
  - ability to (manually) extended system from one set of tasks to another set of similar tasks
  - Measured as proportion of unchanged lines of code between the change from one set of tasks to anothers
- ***Generality***
  - the more environments in which the architecture supports intelligent behavior, and the broader the range of those environments
    → the greater its generality
- ***Versatility***
  - the less effort it takes to get an architecture to produce intelligent behaviour across a given set of tasks and environments
    → the greater its versatility

- **Efficiency & Scalability**
  - *Efficiency*
    - time and space required by the system at the level of architecture's recognize-act cycle or at level of complete tasks.
  - *Scalability*
    - the less an architecture's efficiency is affected by task difficulty, environmental uncertainty, etc,
      → the greater its scalability (Utility problem)
- **Reactivity & Persistence**
  - *Reactivity*
    - speed with which an architecture responds to unexpected situation or events
  - *Persistence*
    - degree to which an architecture continues to pursue its goals despite changes in the environment
- **Improvability**
  - it refers to agent's ability to improve its behaviour over time.
  - we can measure it in terms of agent's to perform tasks that it could not handle before the addition of knowledge
- **Autonomy & Extended Operation**
  - *Autonomy*
    - agent's ability to create its own tasks and goals
  - *Extended operation*
    - agents must be **robust** enough to
      - keep from failing when they encountered unexpected situations
      - keep from slowing down as they accumulate experience over long periods of time

Interdisciplinary Effort
Designing CAs requires contributions from different scientific fields on three conceptual levels:



Cognitive System - Robot Hands

Goals:

- why is grasping difficult?
- why is building hands difficult?
- definitions
  - grasping
  - form closure
  - force closure
- grip taxonomy → disambiguation between grips
- finray gripper → working principle
- modeling grasping → Assumptions, Disturbances
- grasping pipeline → task to grasped object

Building a Hand Challenges

mechanical	Electrical	Control
• weight vs. force • soft materials • actustion	• power vs. energy storage • sensors : touch, temperature • cabling	• high number of DoF • redundancy • sensing

Robot grasping -Introduction
Goal:

- To hold an object in the robot's end-effector firmly, preventing loss of contact.
- To maintain the grasp in the face of unknown disturbing forces or moments applied to the object.

Definition:

- **Grasping**
  - holding something firmly relative to the end-effect
- **Form closure**
  - The object completely enclosed by the hand, without any wiggle room
  - No movement is possible in any direction (assuming that all contacting bodies are rigid)
  - Seven contact points are necessary for an arbitrary object in 3D with 6DoF
- **Force closure**
  - The object is held applying some force
  - "Similar to form closure, but relaxed to allow friction forces to balance the object wrench"

- 2 or 3 contact points necessary for a 3D object with 6DoF, depending on the type of contact

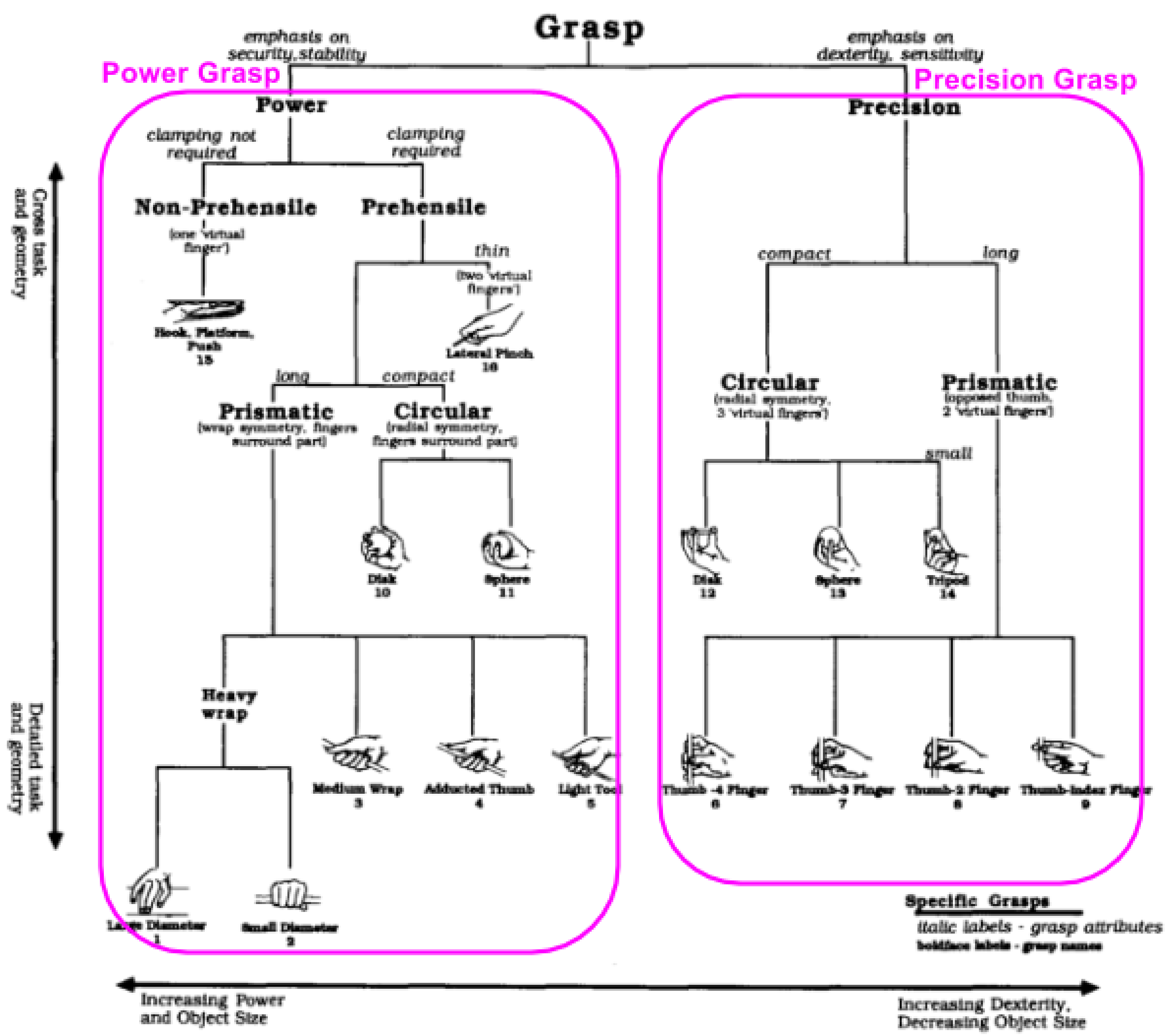
Robot end-effectors for manipulation

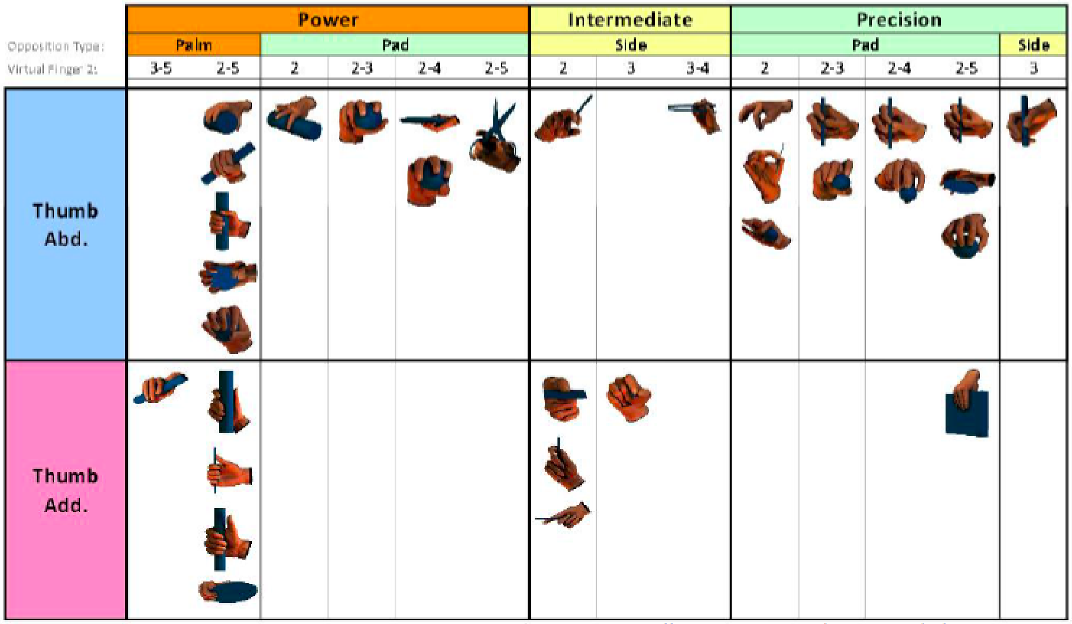
Simple and advanced grippers

		
1 DOF (open-close) mainly for industrial routine manipulations	Manipulator used in the lab exercise of this course	Advanced 3-finger grippers, fingers can be rotated Top: Schunk, Bottom: Barrett

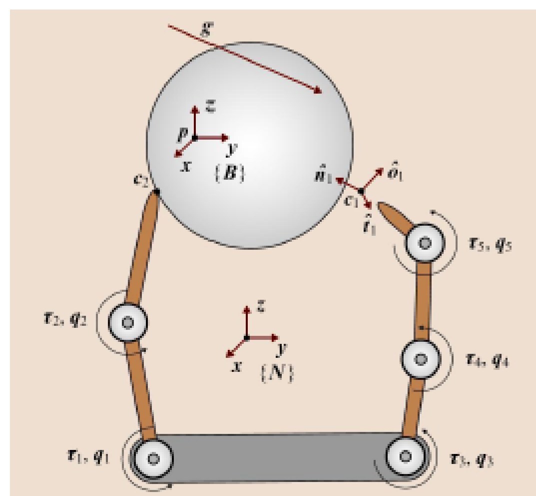
Models for grasping

Taxonomy for human grasping





Rigid modeling of grasping systems



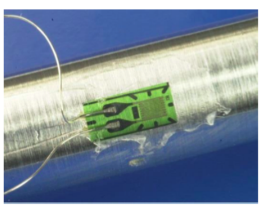
- Mathematical model capable of predicting the behavior of the hand and the object
- Typical disturbances : inertia forces, gravity
- Grasp maintenance : the **contact forces** applied by the hand **prevent** contact separation and unwanted contact sliding
- **Assumptions:**
  - hand links are rigid
  - the object is rigid
  - each contact point has a unique, well-defined tangent plane
- {N} = inertial frame (world)
- {B} = frame attached to the object
- $p \in \mathbb{R}$^3 defines the position of $\{N\}$ rel. to $\{B\}$
- $c\_j \in \mathbb{R}$^3 is the position of each contact point
- Each contact has a frame : $\{C\}\_i$ = {, n_i, t_i, o_i} with n_i being normal to the contact
- q_i are the joint positions
- τ_i are the joint efforts (torque or force)

Object Modeling

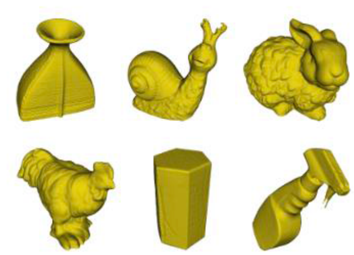
- For grasping, the **external geometry** is interesting
- Usually, the model will be a triangle mesh (as used in computer games), or CAD model
- Important assumption
    - **Rigid (**non-deformable**) object**
    - **Shape is well known**
- A model can be estimated from 3d point data (laser, stereo, time of flight)
- Find stable grasp through simulation

Systems for Grasping & Manipulation

Sensors for grasping

Contact / force sensors to Keep contact to the object	Sensors to identify objects and to find an optimal grasp
 Strain gauges to measure forces  BioTac sensor: - force - vibration - temperature	 Time-of-flight cameras  Stereo cameras  RGB-D sensors

Identifying grasp pose → 3D Object Recognition

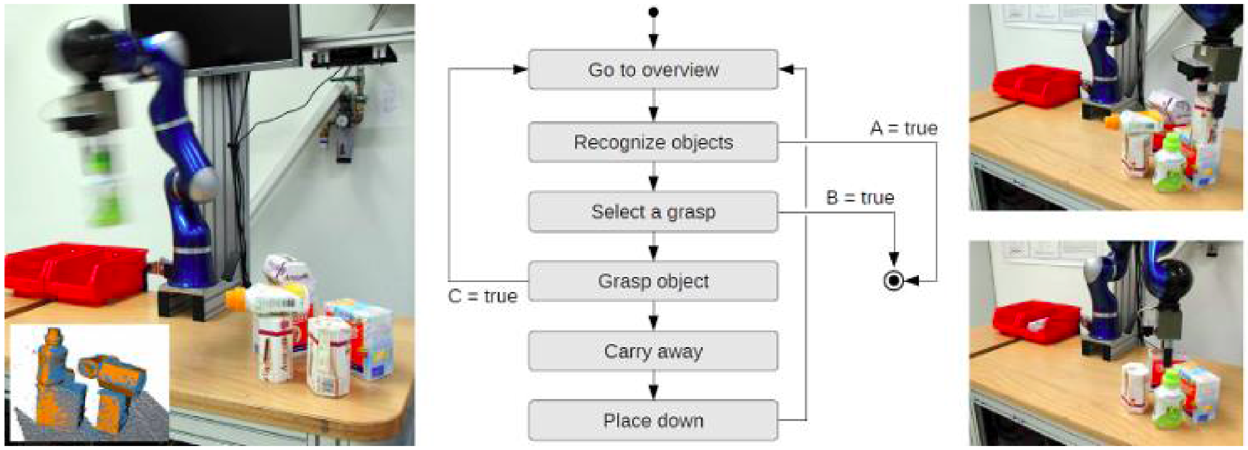


- Problem statement
    - Given : a set of 3D models
        - Questions : **Which** models are located **Where** in the 3D scene?
    - Goal : for each model → Compute a rigid transform that aligns it with the scene

Robot grasping : Unmodeled object

- Strategy :
    - Estimate the position and size of the object
    - Find a grasp where the hand and fingers build a cage around the object

Example - Sorting Object



- The logical clauses A, B, C defining the transition conditions,
    - A = no-object
    - B = no-grasp
    - C = collision v failed-grasp

Integrating into real systems

- Deciding how to grasp an object is not as trivial as it feels
- Different grasp planning methods have their own advantages & disadvantages
- For the robot to act intelligently in the environment, a lot has to come together :
    - mobile base can move the robot precisely
    - the arms have to place the end-effectors where they are needed
        - avoiding collision
    - the robot must be able to grasp things without destroying them
    - grasping is only the first part of the story
        - then lifting and placing still are needed

Beyond grasping

- Grasping is usually limited to holding the objects in the end-effector
- Most robots are limited to pick-and place tasks
- **In-hand-manipulation** requires better dexterity
    - Taking an object, and repositioning it in the hand
    - Rotate and move the object to ease hand-over
    - Or even grab a device and touch buttons
    - Using an object effectively as a tool

Soft robotics

Two views of intelligence according to **Rolf Pfeifer**
classical: cognition/intelligence as computation
embodied: cognition from movement and interaction (enactive cognition)

Goals for real-world robots:

- robust and adaptive behaviour
- sophisticated sensory-motor skills (running, walking, playing soccer,...)
- smooth (“soft”) interactions and cooperations
- Autonomy

Approaches:

- Embodiment (using soft robot bodies)
- Morphological computation

Main idea of soft robotics:
“Make all of the components in the robot soft and flexible in order to move in very limited spaces and change gaits (german: *Grundgangarten*) fairly easily”

Main advantage:

- safer
- using modern sensors and control systems, they can be controlled as precise as classical robots
- lower cost and complete new kinematics

Morphological computation:
“If the material properties are similar to those of humans, a lot of computation for emulating human dynamics can be avoided because the dynamic behaviour is similar”

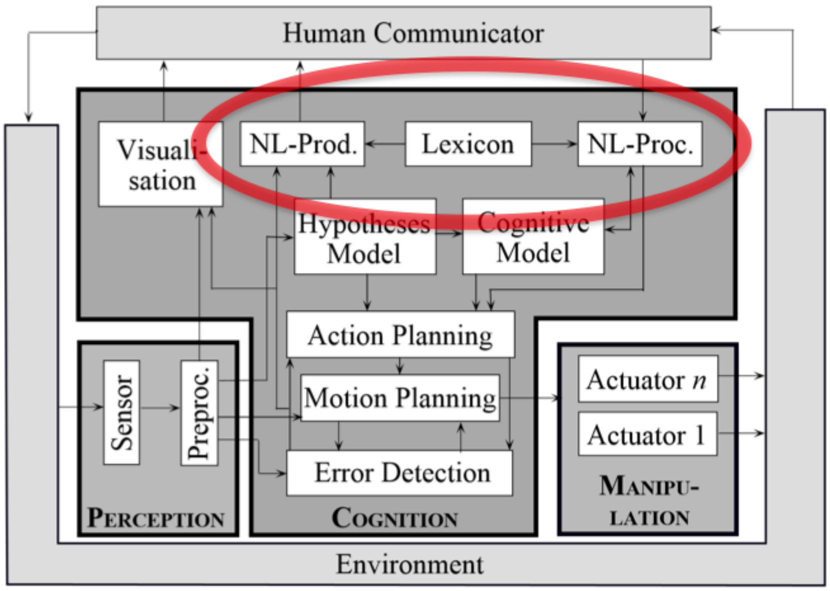
“**Materials** can **take over** some of the **processes normally attributed to control**”

Actuation principles:
(Note : An actuator is a type of motor that is responsible for moving or controlling a mechanism of system)

- Pneumatics (using air pressure)
- Shape memory alloys
- Electro active polymers
- Fishing rope
- Hydraulic
- Ultra sonic motors
- Motors

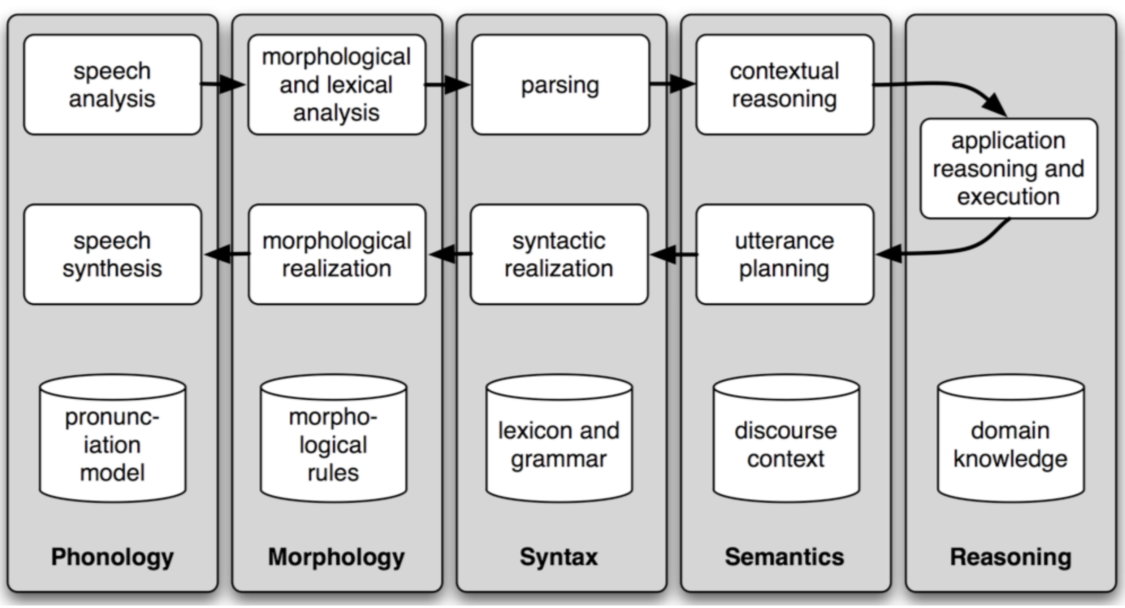
Variable Stiffness Actuators (Antagonist principle, Change of Lever Arms)
Human Computer Interaction

… as part of the cognitive architecture

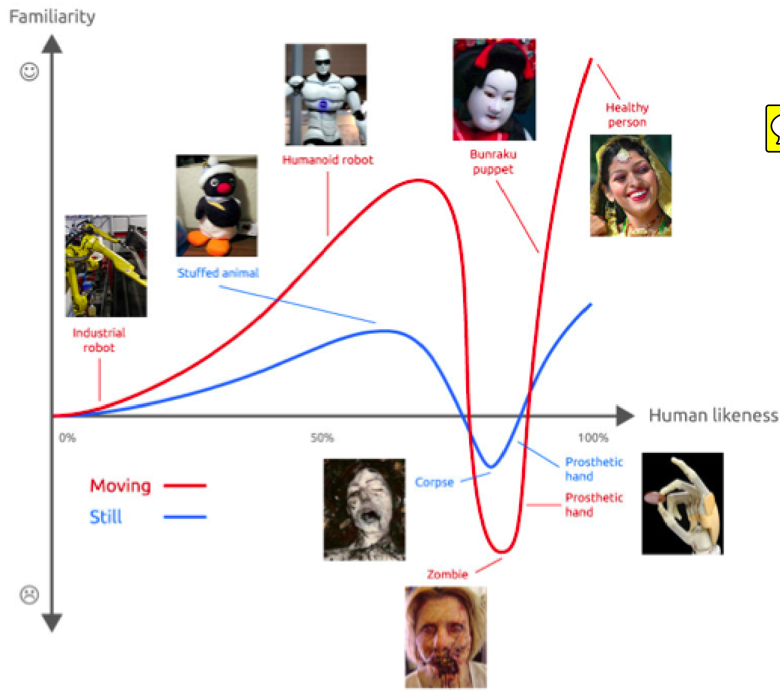


… as detailed overview

Bird, Klein & Loper (2009)



Human - Computer Interaction : Familiarity - Human likeness Appearance



How do we evaluate Dialogue Systems?

Dialogue Efficiency
- Overall time taken
- Mean response time

Dialogue Quality
- Number of times, the robot gave assembly instructions
- User gaze behaviour before and after system-generated references

Task Success
- Proportion of target object assembled correctly

Example for a dialogue systems robot: **JAMES** (Robot bartender)

Why do we need user modelling and signal processing?

- Really important for an robot: When does a customer wants **to start and end an interaction**?

How do you model a user for JAMES?

- Focus on analysis of ordering requests (manual analysis of interactions)
- Result: Humans have three cues to indicate interaction:
    - Body posture
    - Head pose
    - Face-to-Face transformation
- Evaluate the ordering requests by drawing a automata
- Train a Hidden Markov Model to estimate in which state JAMES should be

Some notes about industrial robots nowadays:

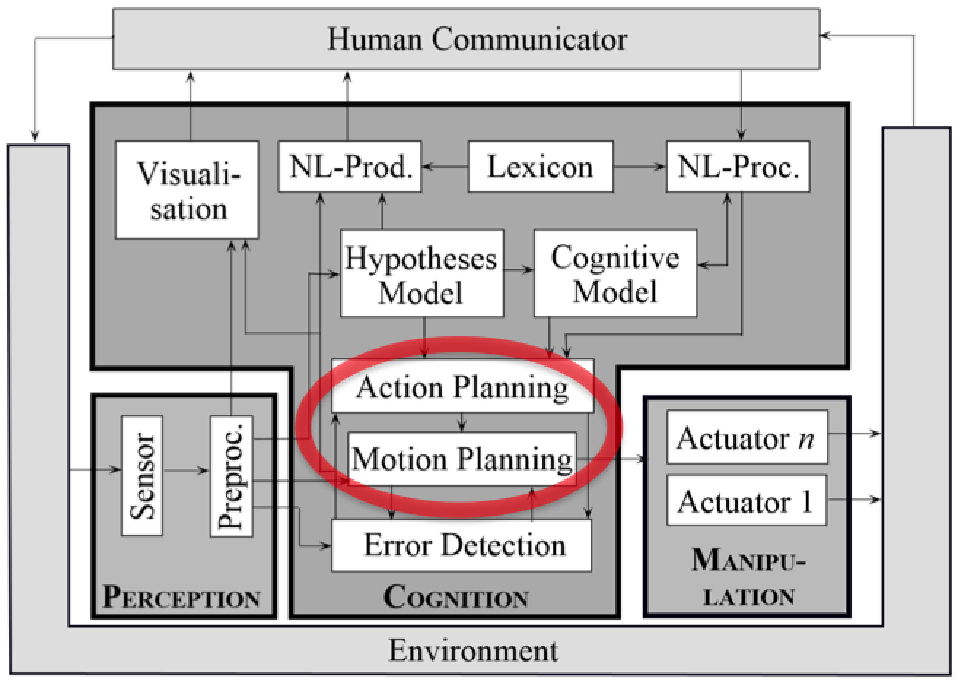
- Highly automated
- Off-line programmed robots work on their own
- Strict separation of workspaces between human and robots

Some notes about humans in industry:

- Easy handling of detailed/asthetic tasks
- Fast adaption to new jobs and changes in the environment

The Challenge of Human-Computer Interaction: **Combining the skills of robots & humans**!

Planning task in cognitive systems



*Description of a **planning task**:*
Given an **initial state**, **find a sequence of actions** which is supposed to lead me to a **goal**

Elements of a planning task:

- Goals
- State descriptions
- Possible actions

*Example: Drive from **A** to **B***

State descriptions	Actions	Required knowledge
Topological position description	Choose road at a crossing	Schematic road map
Position on map	Drive, change road	geometrical road map
Actual position on road	Accelerate, break, steer	Precise map, environment model

Two kinds of planning for cognitive systems

Task planning

- State description: attributes, properties
- Actions: state transitions with preconditions and state changes

(Low level) Motion planning

- State description: position, angles
- Actions: Control commands

In real life, both have to be combined!

STRIPS - Classical symbolic planning language

Representation of states in STRIPS

- Representation of a states as a conjunction (AND) of positive literals
- Closed world assumption (Everything which cannot be modelled is false/doesn't exist)
- Propositional literals: (Poor AND Unknown)
- First-Order-Literals: (At(Plane1,Melbourne) AND At(Plane2,Sydney))
- Representation of goal as a specified state, goal is achieved if current state contains all literals of goal

Representation of actions in STRIPS

- Actions have
  - Preconditions, which has to be fulfilled
  - Effect, which will change states. Everything which is not changed remains

Example in STRIPS:

$Init(On(A, Table) \wedge On(B,Table) \wedge On(C,A) \wedge Block(A) \wedge Block(B) \wedge Block(C) \wedge Clear(B) \wedge Clear(C))$

$Goal(On(A,B) \wedge On(B,C))$

$Action(Move(b,x,y)$
PRECOND: $On(b,x) \wedge Clear(b) \wedge Clear(y) \wedge Block(b) \wedge (b \neq x) \wedge (b \neq y) \wedge (x \neq y)$
EFFECT: $On(b,y) \wedge Clear(x) \wedge \neg On(b,x) \wedge \neg Clear(y))$

$Action(MoveToTable(b,x)$
PRECOND: $On(b,x) \wedge Clear(b) \wedge Block(b) \wedge (b \neq x)$
EFFECT: $On(b,Table) \wedge Clear(x) \wedge \neg On(b,x))$

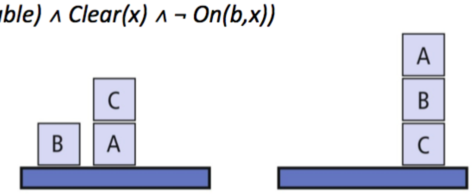


Image source: Russel/Norvig

Requirements for logical planning approaches:

- Observable
- Discrete
- Environment fully known
- Deterministic

Planning in real world adds additional constraints:

- Execution times are relevant
- Resources
- Schedule is important because it determines the overall plan

Hierarchical planning

Plan refinement (i.e. self driving car - route within cities planned offline, exact steering wheel parameters planned online later based on sensor information

- Simplest form : Description as tuple (original plan, refined plan)

Plan refinement example with resource limitations

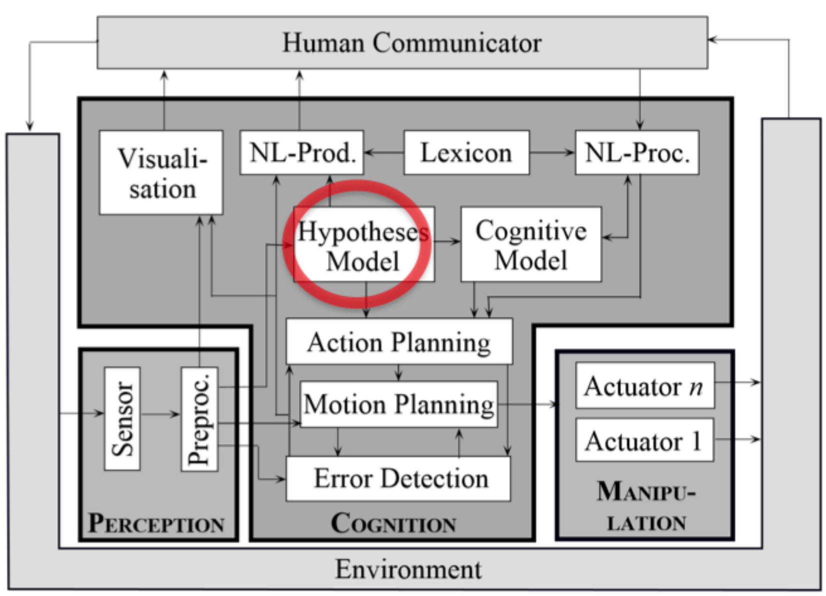
Formal description of the refinement by replace operator

$$replace(original,\quad substitute,\quad precondition,\quad aditional\_property)$$

In the example (off-line variant)

$$\text{replace}\begin{pmatrix} (move), \\ (\text{grasp}(manipulator), \text{pull}(manipulator)), \\ (\text{available}(manipulator)), \\ (\text{addcost}(rentalfee)) \end{pmatrix}$$

State estimation in cognitive systems

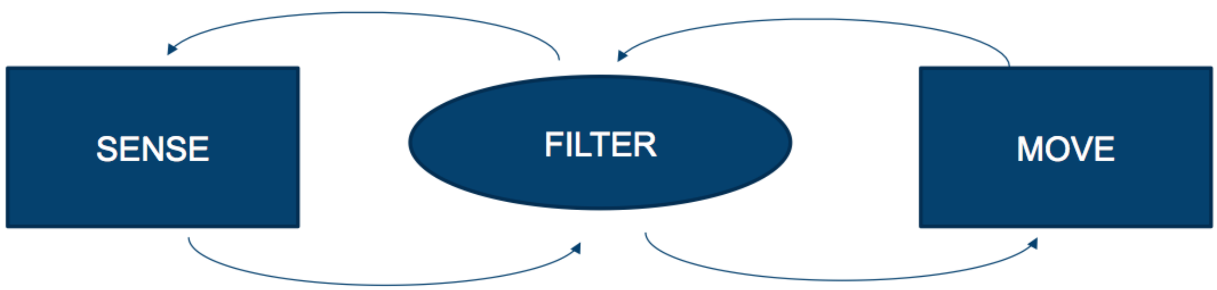


Motivation

- A robot is in an unknown environment
- Each location has a certain probability
- Everything is has equal probability before the robot knows anything
- Robot has sensors, however those sensor data might be noisy

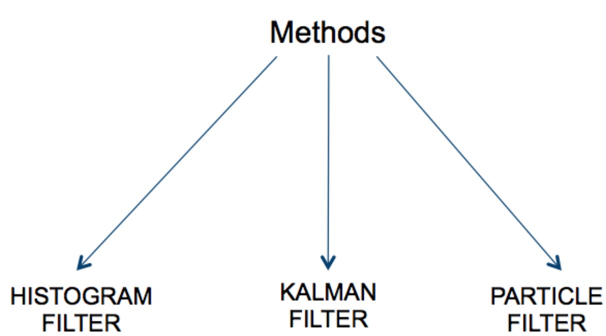
Vocabulary of State Estimation

- Perception (sensor data)
- Actions (in this case: movements)
- States (variables)
- Beliefs (probabilities)
- Priors (beliefs before sensor input)
- Posteriors (beliefs after sensor input)



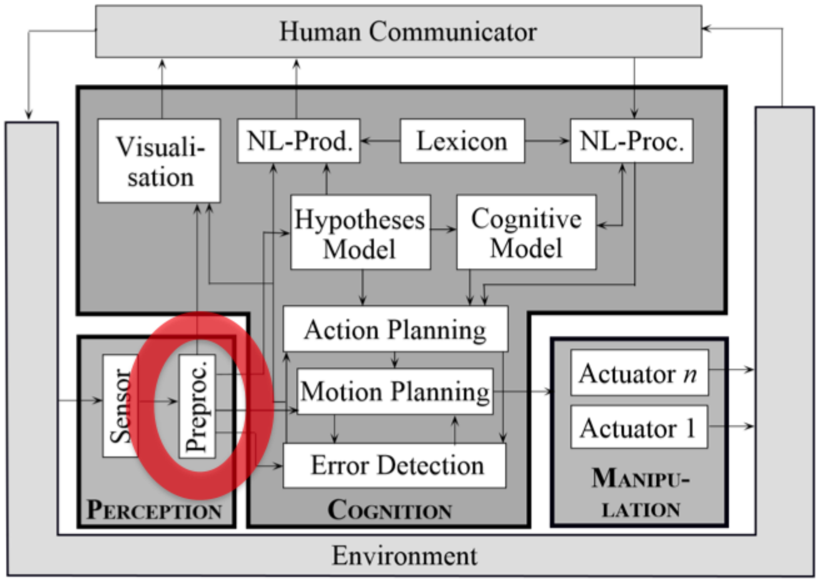
- Everytime a robot senses it gains information
- Everytime a robot moves it loses information
- Filters try to infer information about the robot state through different probabilistic methods

Methods for Localization / Tracking



Filter	State Space	Belief	Efficiency
Histogram	Discrete	Multimodal	Exponential
Kalman	Continuous	Unimodal	Quadratic
Particle	Continuous	Multimodal	?

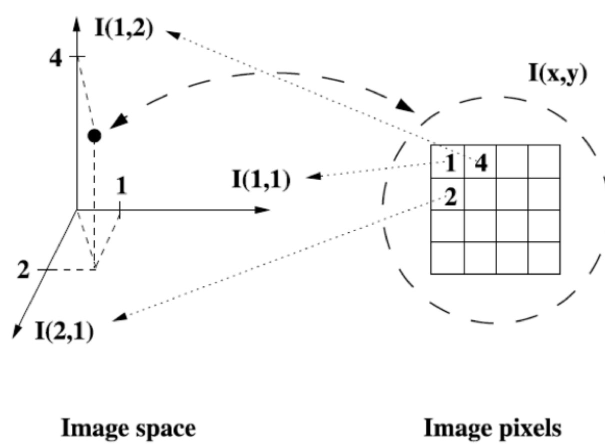
Perception in cognitive systems



How does the image space looks like for images?

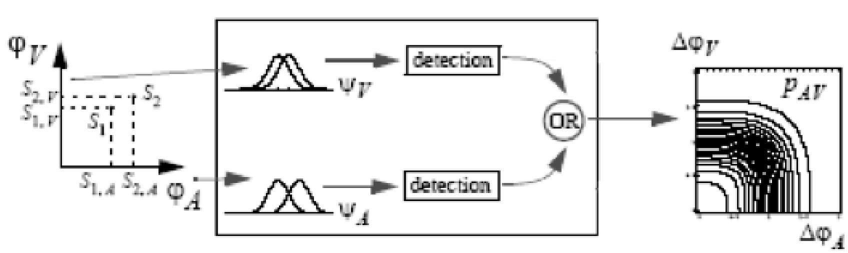
We can transform images into fourier space!
(Basic signal processing)

The most Information for images are hidden in the **low frequencies** because they are **responsible for gradual changes in the image**

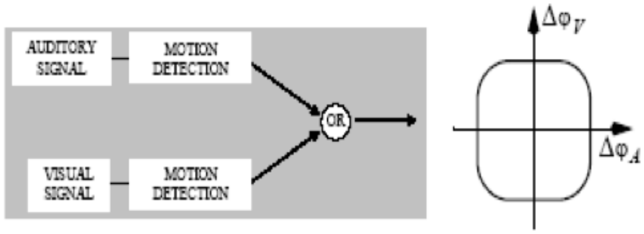


How do we handle many input signals?

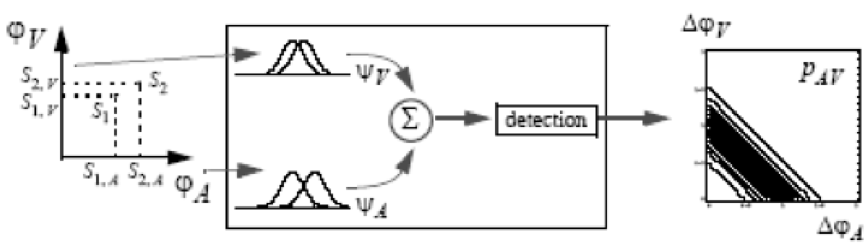
Decision Fusion Models



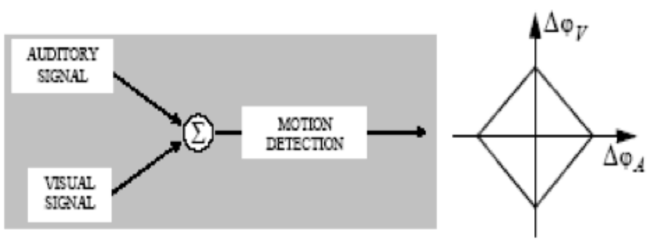
Predicted 2D-Thresholds:



Data Fusion Models

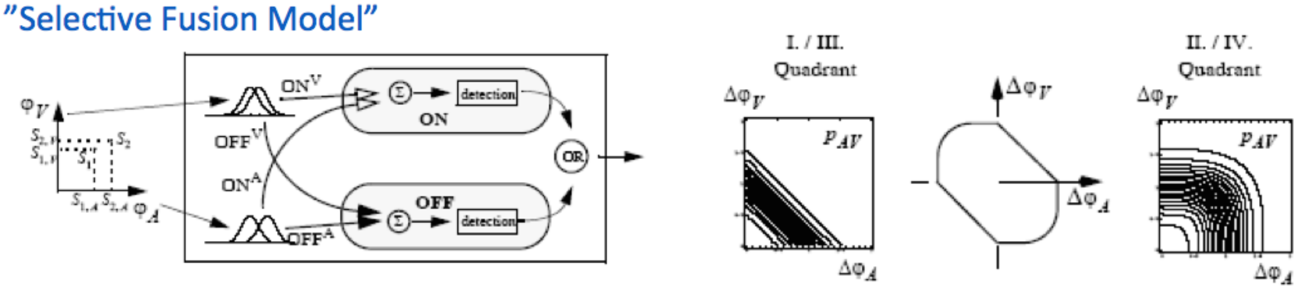


Predicted 2D-Thresholds:



Measured 2D thresholds

"Selective Fusion Model"



This model comes closest to model real observations!
It means, if things make sense -> if things get louder and simultaneously visually bigger, we get easier stimulated! (i.e. a car coming closer)