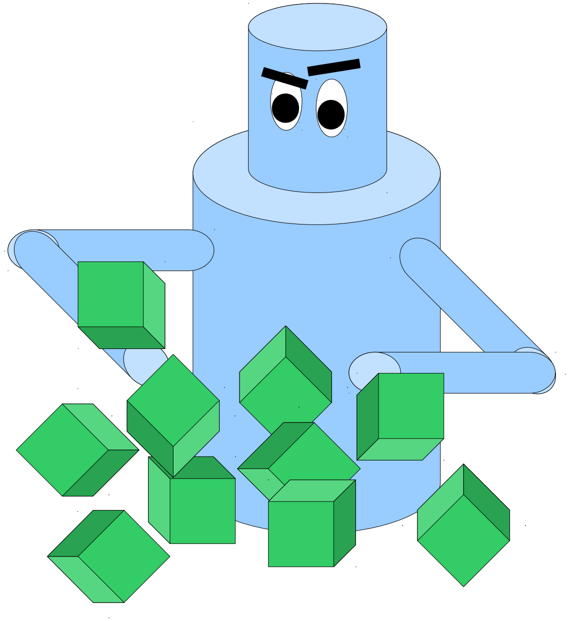
Bootstrapping Object and Grasping Knowledge with Object-Action Complexes

Norbert Krüger, Justus Piater,
Christopher Geib, Ron Petrick,
Mark Steedman, Florentin Wörgötter,
Ales Ude, Tamim Asfour, Dirk Kraft,
Damir Omrcen, Alejandro Agostini,
Rüdiger Dillmann

Objective

- An autonomous robot should reason in terms of (typically discrete) symbolic concepts. *(~language)*
- Origin of symbols and rules?
  - Grounded in (typically continuous) physical interaction.
  - Learnable/refinable.



Object-Action Complex

- Describes how an object is affected by an action.
- Can be executed to actually do it.
- Allows reasoning based on experience.
- Combines notions of
  - affordances (perception)
  - prediction (action, state transitions)
  - reasoning (~STRIPS)

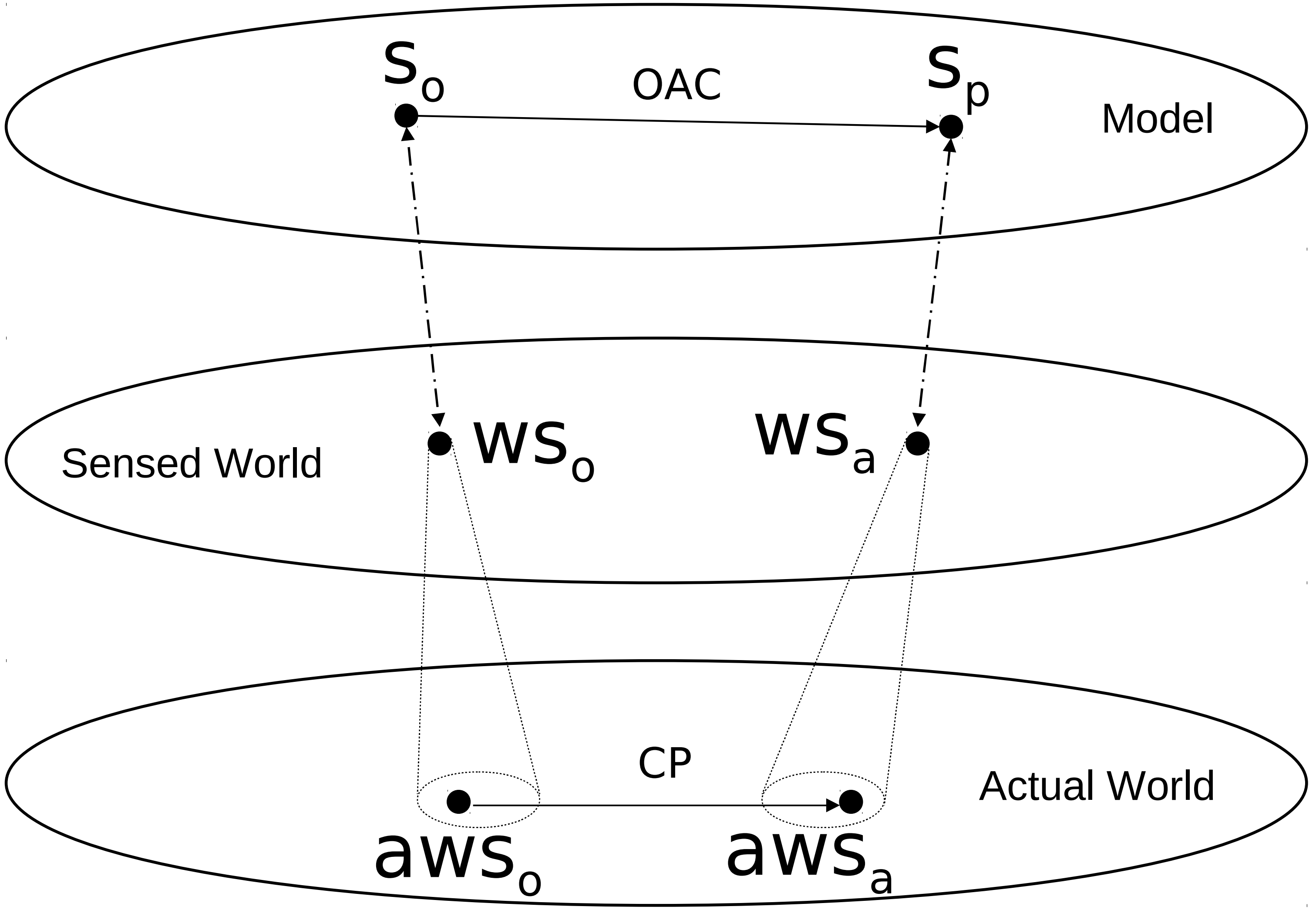
A Unified Framework

- Rule-based reasoning systems:
  - mostly discrete, deterministic, fixed rules
  - OACs: symbolic or sub-symbolic state spaces, quantitative and uncertain results, rules grounded in physical interaction
- Unified concept of predictive rules:
  - various continuous or discrete domains
  - various levels of abstraction
- Learning and self-evaluation:
  - representations, control programs, predictions

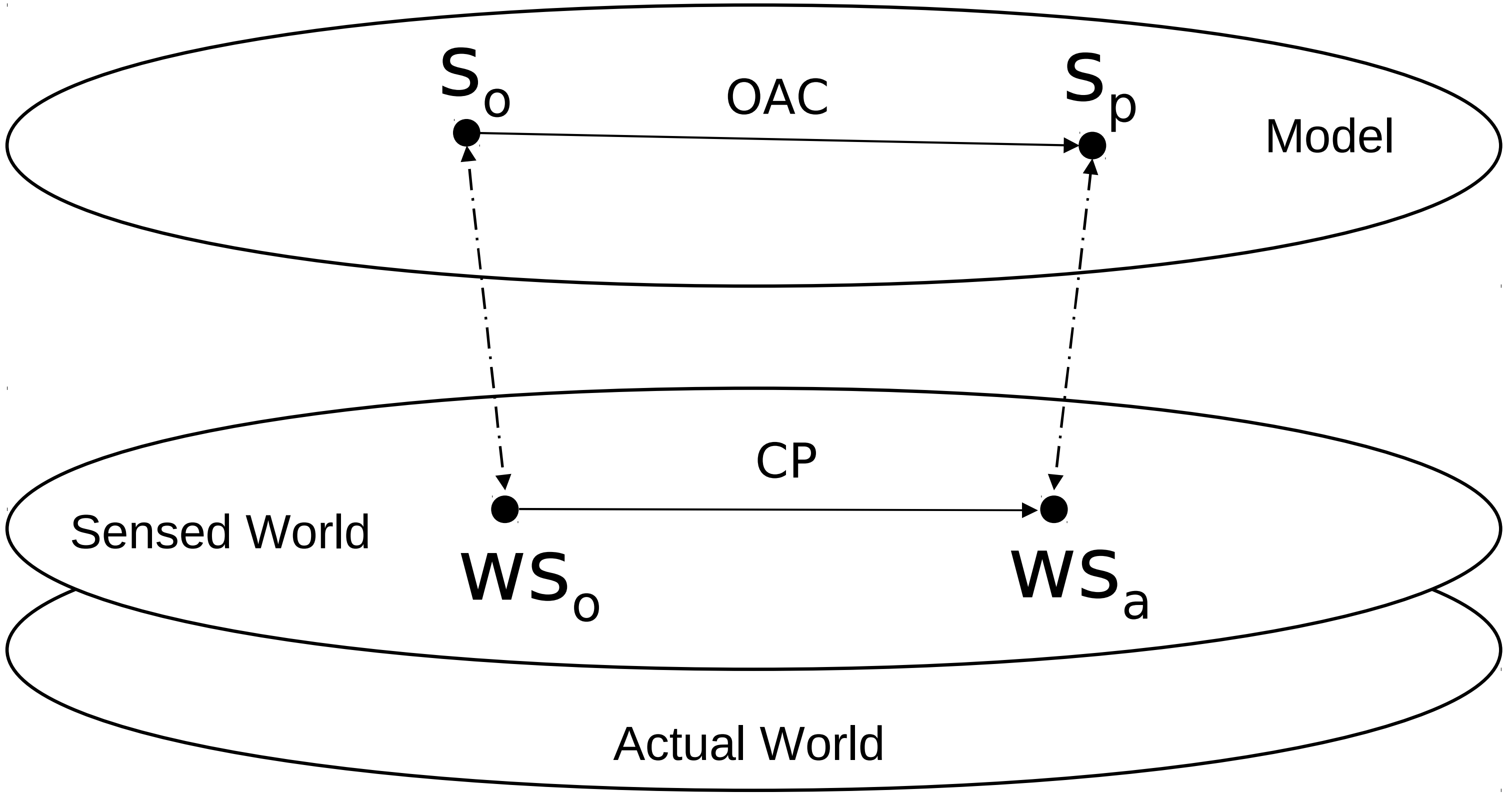
Outline

- Part I:
  - Definitions around OACs
- Part II:
  - Examples of individual OACs
  - Examples of OACs in interaction

How an OAC sees the world



How an OAC sees the world



Levels of Abstraction

- An OAC represents the state of affairs at a certain level of abstraction,
- and maps it onto a lower level of abstraction.
- State descriptions may differ between levels, but must be mappable between them (representational congruency).

Design Principles

- Attributes to express relevant aspects of states.
- Prediction of action effects.
- Span the abstraction hierarchy down to physical sensors and effectors. (Build real robots!)
- Evaluate actions by comparing expected and observed action effects.
- Learn and adapt in various ways.
- Measure reliability in terms of success statistics.

OAC Definition

- States $s \in S$
- Prediction function $T : S \to S$
- Statistical evaluation measure M
- OAC (id, T, M)
- Compact state descriptions for

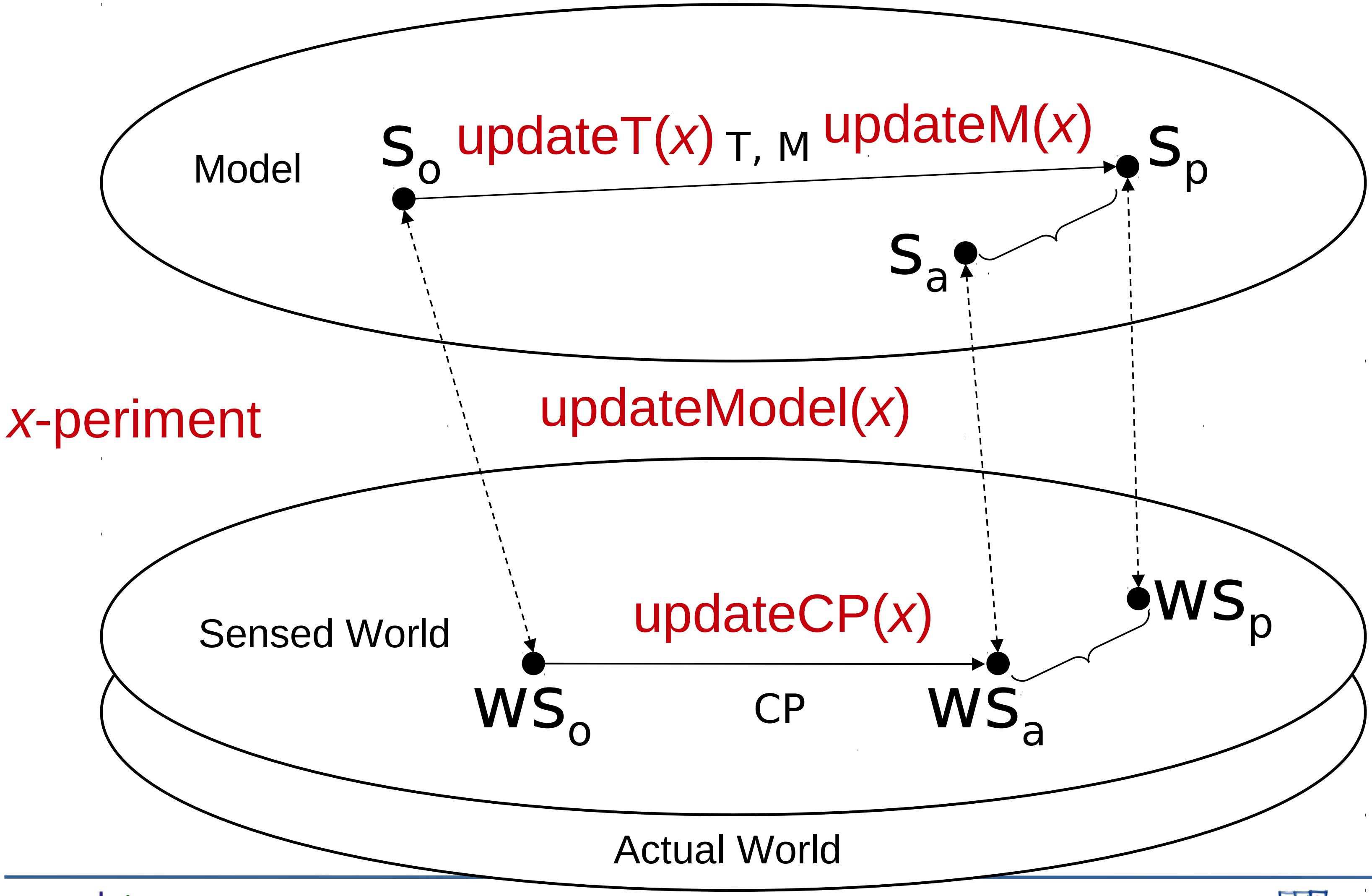
$$\mathrm{range}(T), \mathrm{domain}(T)$$

Execution

$$\text{execute}:\text{id}\rightarrow\left(s_o,\text{id},s_p,s_a\right)$$

- Maps an OAC id to an experiment.
- May call upon other OACs.
- Descriptions of $s_?$ may include whatever is useful (often more than the caller cares about).
- No arguments, but state attributes.

Learning

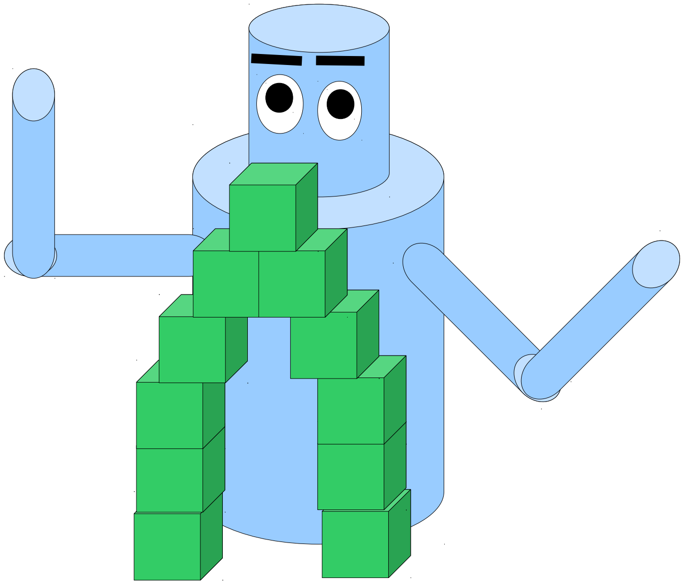

Model
S_o
updateT(x)
T, M
updateM(x)
S_p
S_a
x-periment
updateModel(x)
Sensed World
updateCP(x)
WS_p
WS_o
CP
WS_a
Actual World

OACs and Memory

- T, execute() and updateX() may share memory:
  - updateCP() may refine control program parameters used by execute().
  - Object models may be used by T and by execute().
  - ...

Building Blocks for Cognition

- OACs are formed and refined through
  - sensorimotor exploration of effects of actions on percepts
  - exploration of effects of OACs
- Hierarchical abstractions of experienced object-action effects.

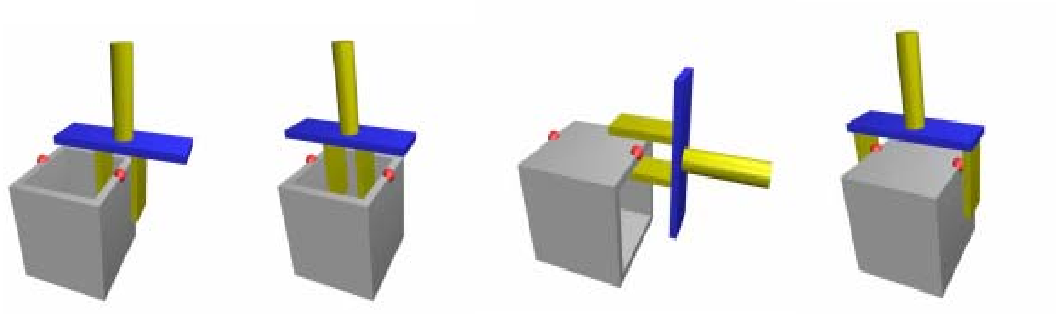


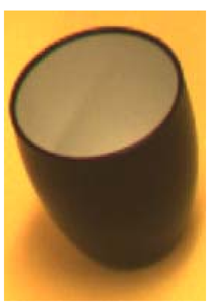
Part II: Examples of individual OACs and their Interactions

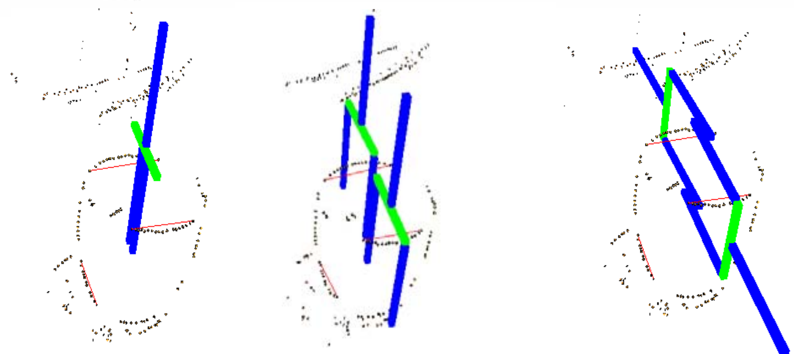
- Graping on three abstraction levels
  - Generic Grasping
  - Object Specific Grasping
  - Grasping on planning Level
- Grounding by interaction of OACs

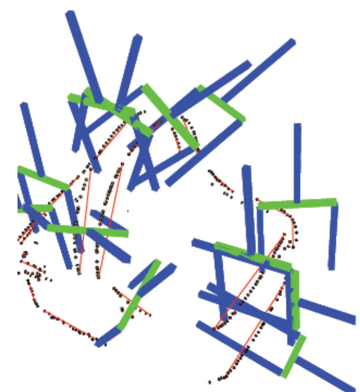
$\text{oac}^{\text{genGrasp}}$: Grasping unknown objects

- Co-planarity Relation between visual entities define potential grasping affordances









.

T, Domain and Range

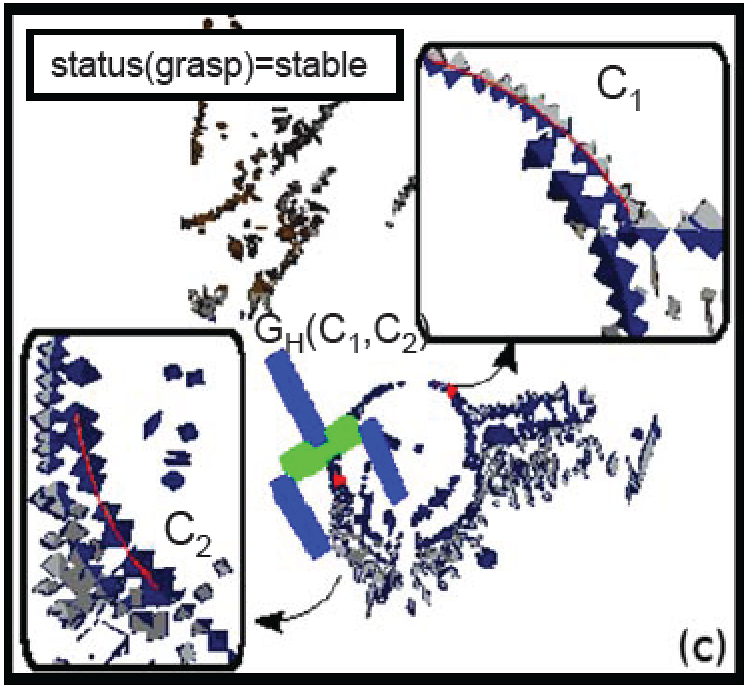
- Defintion: $oac^{genGrasp}$=(genGrasp; *T; M*)
- domain(*T*) = $\{\Omega \neq \emptyset, \mathrm{status}(\mathtt{gripper}) = \mathtt{empty}, \mathcal{C} \times \mathcal{C}\}$
  - Ω Set of coplanar contours
- range(*T*) = status(grasp) =
  {noplan, collision, void, unstable, stable}
  - *Autonomous success evaluation*
    - noplan: Motion planner did not find an executable path
    - collision: Force-torque sensor above threshold
    - void: Distance between fingers=0 after grasping attempt
    - unstable: distance b.f. grasping attempt > 0 and =0 after lifting
    - stable: distance b.f. > 0 after grasping attempt and after lifting
- T predicts constantly 'stable'

Execution and Experiment

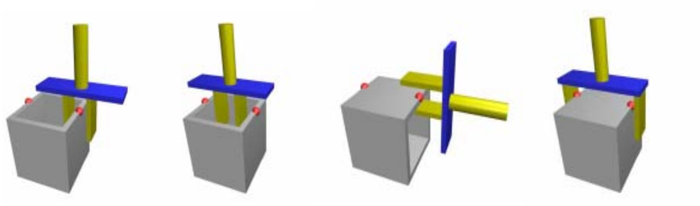
- Execution
  - selection of contour pair
  - compute potential end-effector positions
  - compute valid path
  - move gripper to pregrasp position
  - grasp
  - lift

Movie

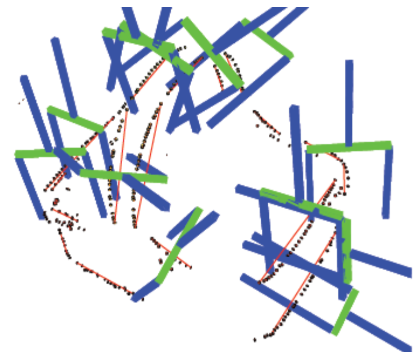
Experiment

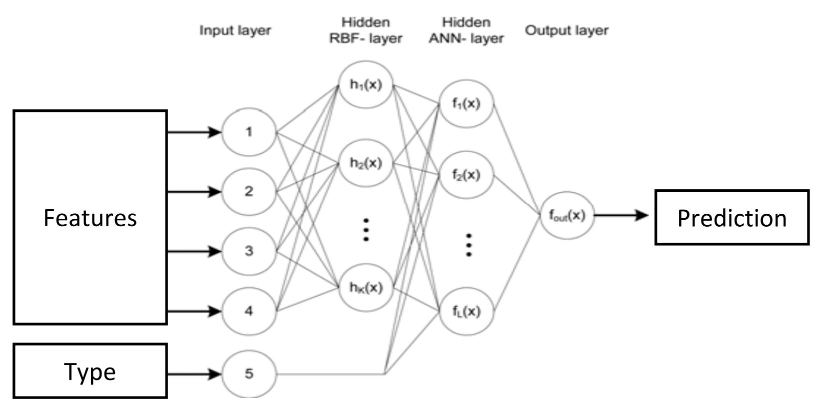


Learning: updateCP



- Since there are many co-planarity relations a large number of potential grasp options is computed
- The system can choose which option to execute by evaluating the relevance of the different relations
  - coplanarity
  - distance
  - co-colority
  - parallelism
  - collinearity



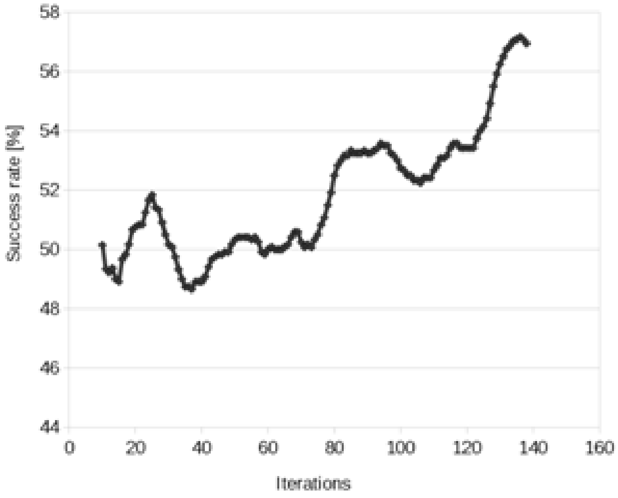


Learning: updateCP

- Learning Cycle:

while *true* **do**
 choose pair of contours C_1, C_2
 experiment=`execute`(`GenGrasp`);
 `updateCP`(experiment);
 `updateM`(experiment);
 drop object
end

Long Term Statistics M



Results

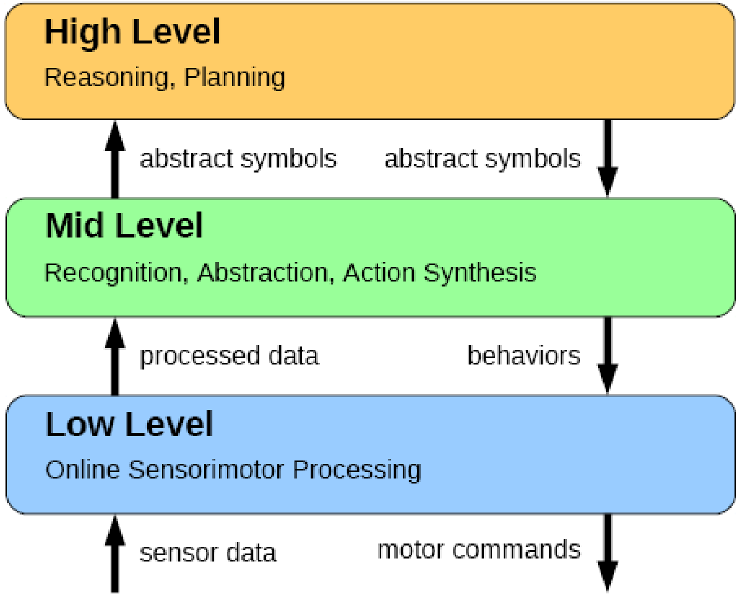
Cylindrical

Non-cylindrical

Learning Set	Test Set		
	Cylindrical	Non-cylindrical	Combined
Cylindrical	57.6%	58.9%	58.3%
Non-cylindrical	33.3%	51.3%	43.1%
Combined	57.5%	45.7%	51.1%
Without Learning	38.3%	45.7%	42.0%

Summary oac^{genGrasp}

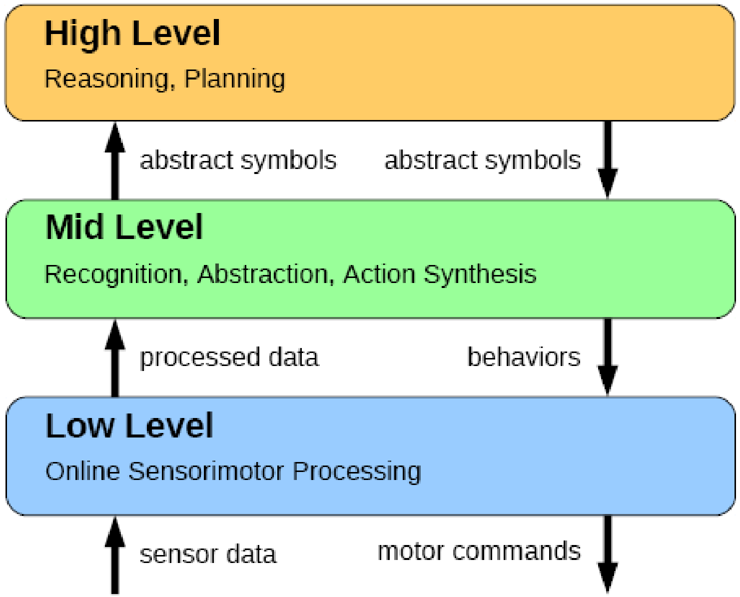
- oac^{genGrasp} associates visual features directly to potential grasps
- OAC on a lower level
  - direct link between sensor and motor information
  - no object memory required
- 'Cheap way' to achieve control over objects



$oac_o^{graspObj}$: Object specific grasping

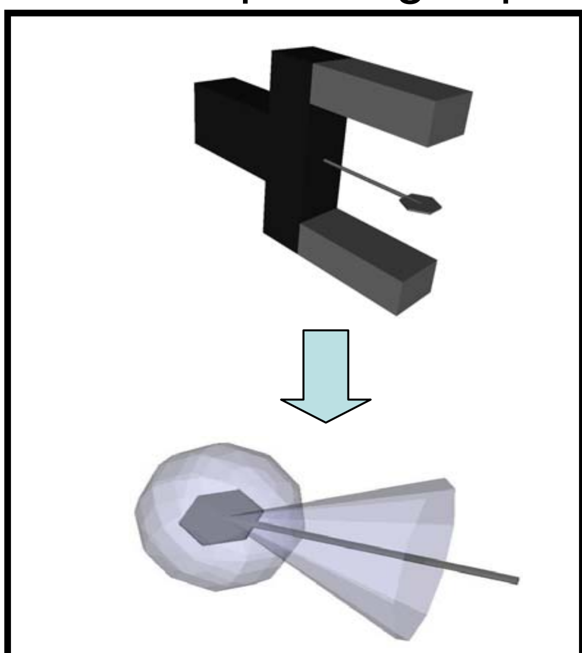
- Given a 3D object represenation in the memory

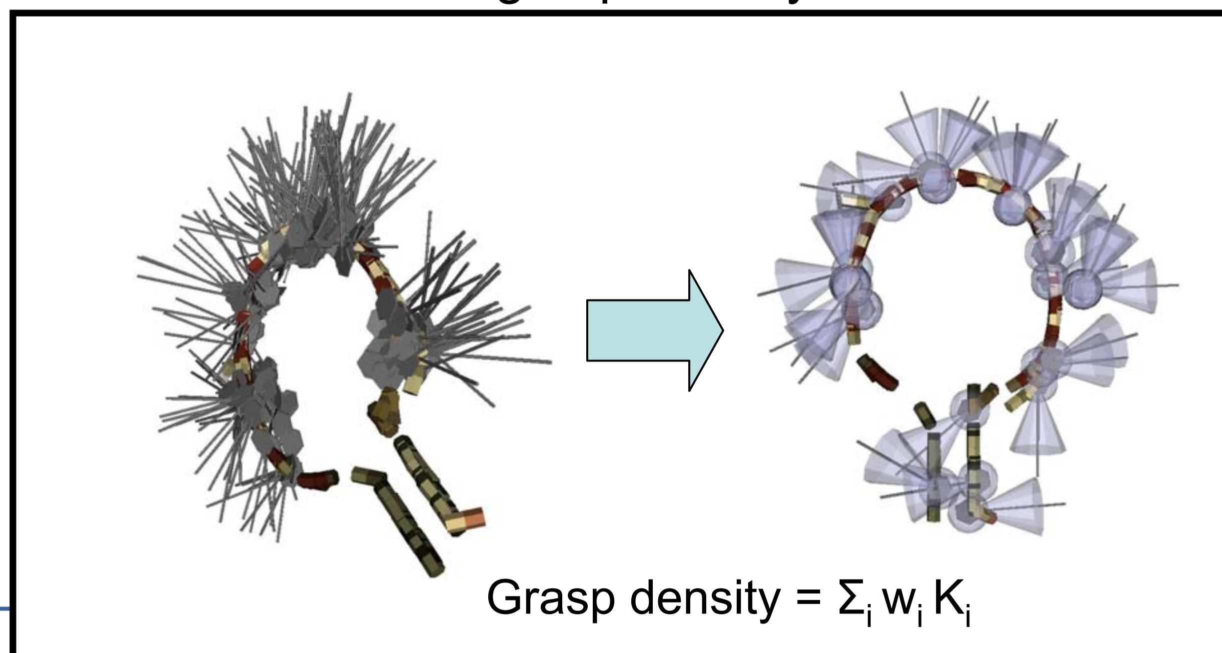
- $oac_o^{graspObj}$ codes the complete set of grasping affordances
- Mid-level oac
  – requiring abstracted object knowledge



$oac_o^{graspObj}$

- Coding Grasp Densities:
  - A grasp is just coded by the pose of the end-effector
  - A grasp attempt can be transformed to a 6D kernel which is the basic building of the grasp density
  - A full grasp density is build up by a number of kernels
- Advantage
  - Representing the manifold of affordances
  - Optimal grasp coded as maximum on grasp density



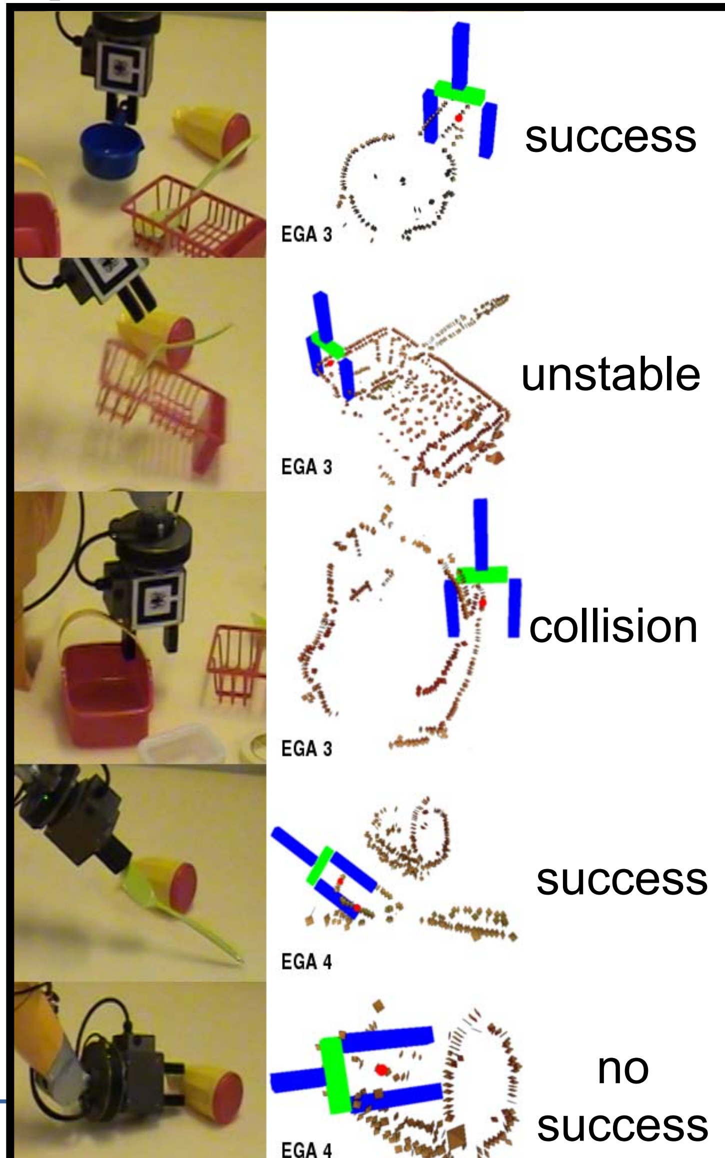


T, Domain and Range

- Defintion: $oac^{graspObject}$=(genGrasp; *T; M*)
- domain(*T*) =
  {status(gripper) = empty; targetObj = o; o in memory}
- range(*T*) = status(grasp) =
  {nopose, noplan, collision, void, unstable, stable}
- T predicts constantly ’stable’

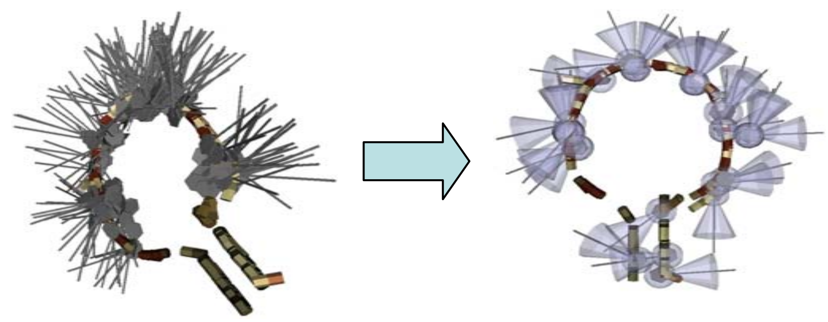
Execution and Experiment

- Two modes
  - Learning: explore all Grasping possibilities
  - Planning: Execute grasping option with highest success likelihood



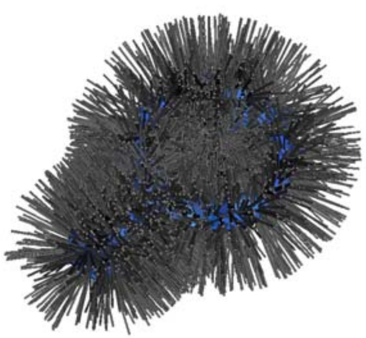
Learning: updateCP

- Learning of grasp-object associations
- Sampling of densities through kernels starting with initial idea triggered e.g., by oac^{genGrasp}

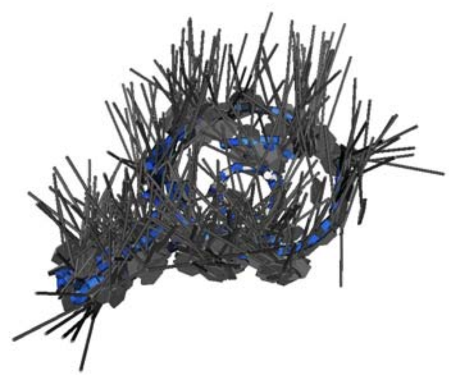


Grasp density = $\Sigma_i w_i K_i$

Before Learning

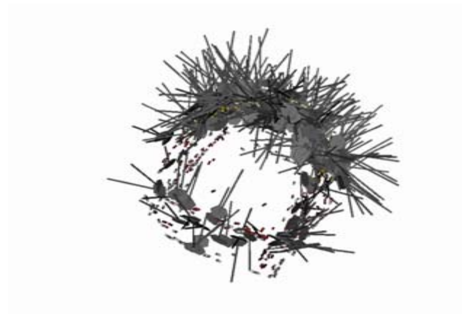


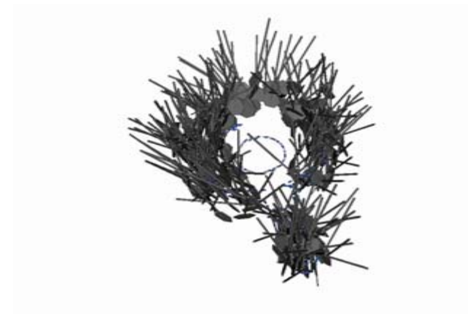
After Learning



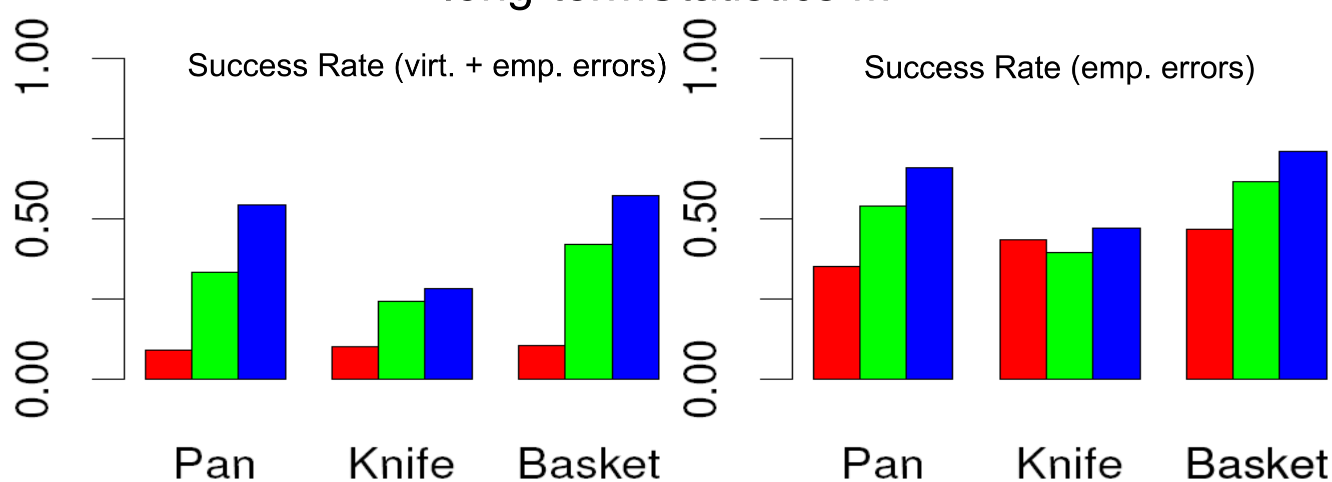
Movie

Results



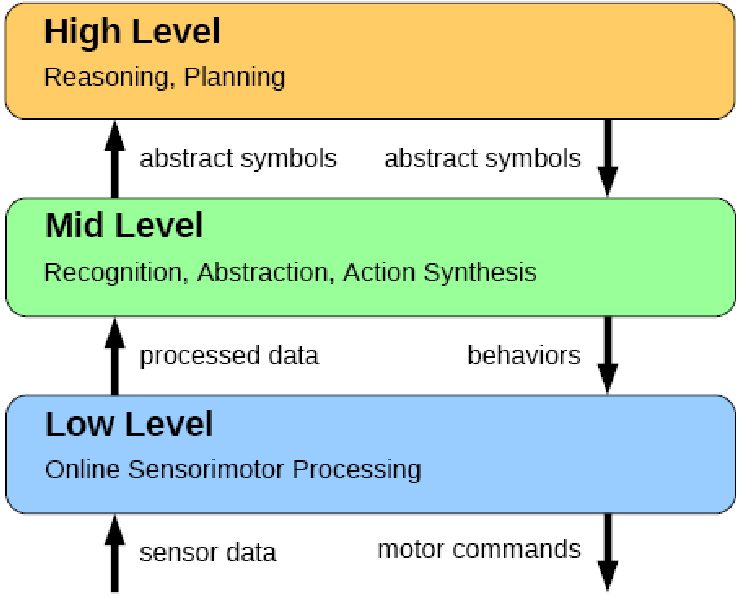


long-termStatistics M



Summary $oac_o^{graspObj}$

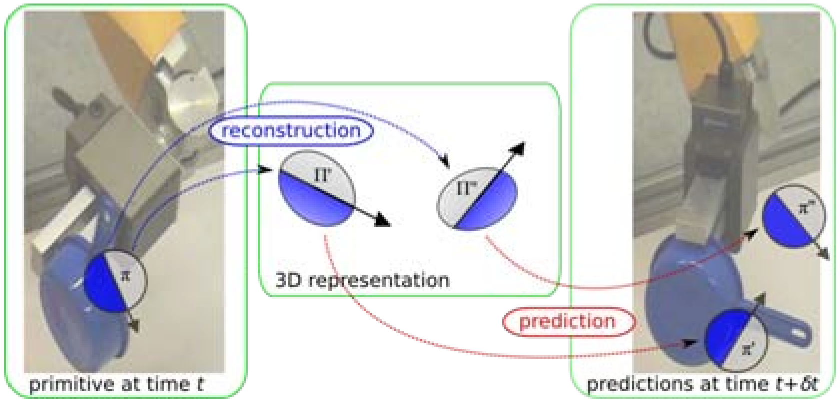
- $oac_o^{graspObj}$ associates grasps to objects
- OAC on a mid level
  - object memory required
- Can be linked to grasping for planning



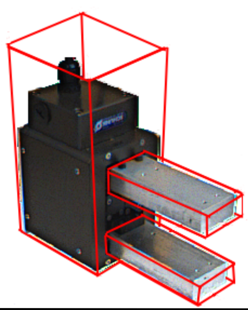
Interaction of OACs: Grounding of objects and grasping affordances

- Given
  - an agent being able to grasp and
  - an arbitrary (rigid, edge-dominated) object in a scene the agent does not know anything about beforehand
- Without any supervision, the agent is supposed to
  - find out that there is a (novel) object in the scene,
  - compute a representation of the object and memorize it,
  - use the memorzied representation to recognize a new appearance of the object in the scene and detect its pose,
  - learn how to grasp the object in a way that allows for an optimal grasp in a given situation.
- Basically it can be read as: Learning from ’scratch’
  - that there is an object,
  - how it looks like,
  - and how to grasp it.

Accumulation



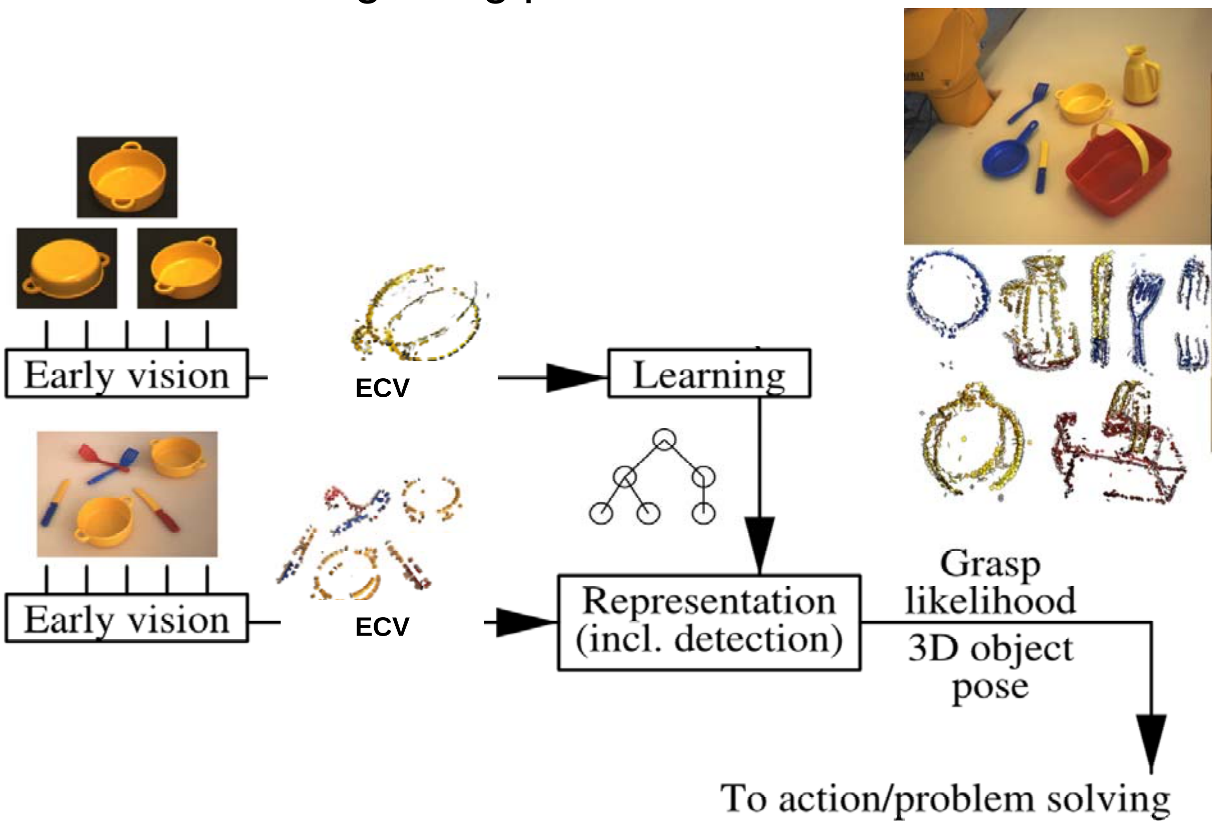
- Once physical control over an object is achieved based on the predictions based on robot motion can be made that establish an object

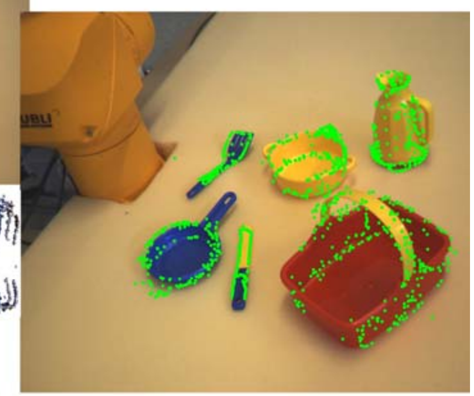


- Body knowledge can be used to substract gripper

Pose estimation

- Objects can be recognized and their pose being estimated
- Method
  - Learn probablistic relational models of ECV feature combinations
  - Matching using probablistic inference

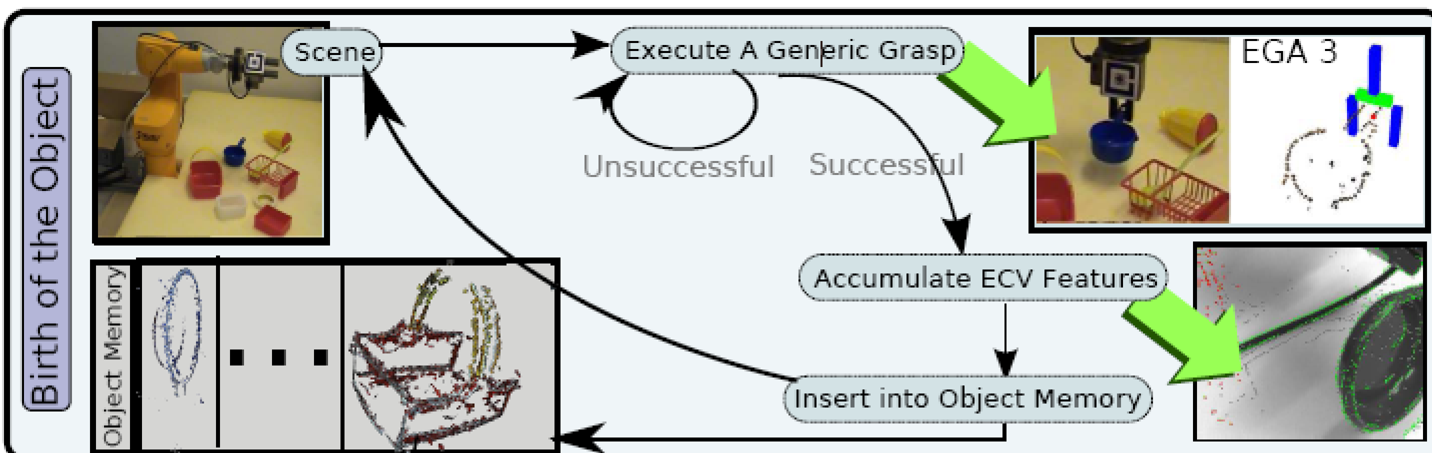




R. Detry, N. Pugeault, J. Piater. A Probabilistic Framework for 3D Visual Object Representation. IEEE PAMI.

The first learning cycle: Birth of the object

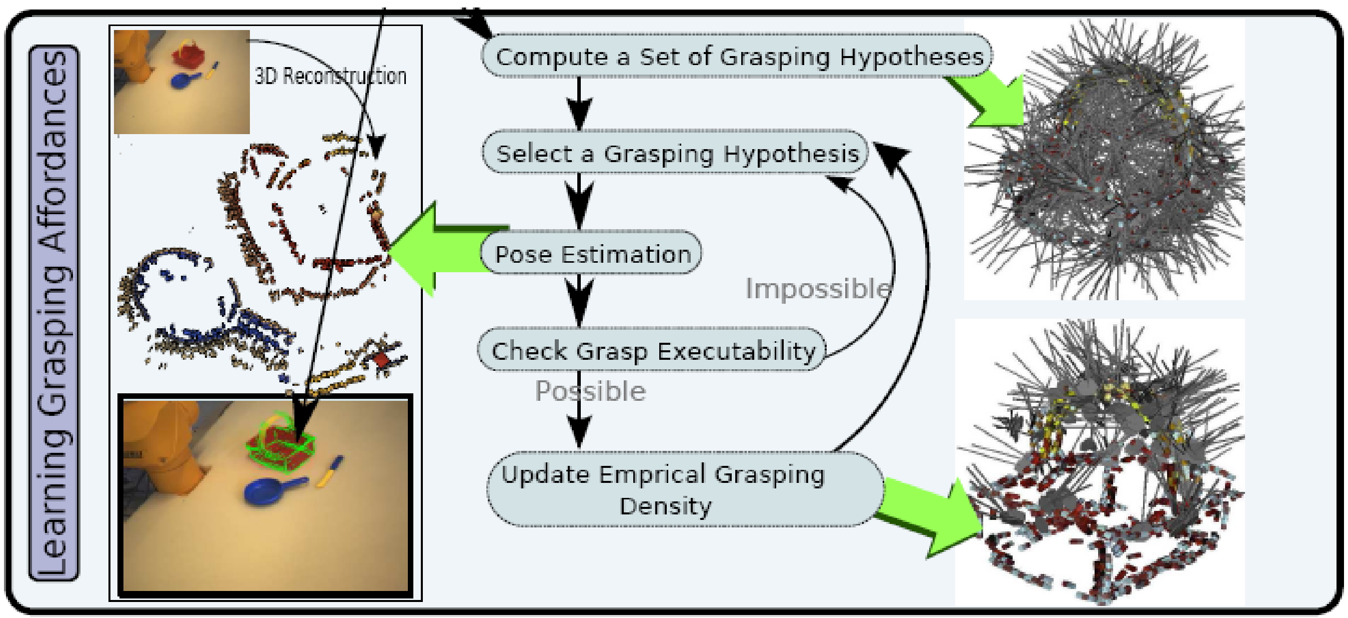
while status(grasp) $\neq$ **stable do**
  experiment = `execute(GenGrasp)`;
  `updateCP`(experiment);
  `updateM`(experiment);
  open gripper
Accumulate object representation o_i
if *accumulation successful* **then**
  transfer o_i into object memory $\mathcal{M}^{O}$



Kraft et al. 2009. Birth of the Object: Detection of Objectness and Extraction of Object Shape through Object Action Complexes. International Journal of Humanoid Robotics (IJHR), 2008, 5, 247-265.

Second learning cycle: $oac_o^{graspObj}$

while *instance of object o_i in scene* **do**
    state.targetObj $= o_i$
    experiment $=$ execute(graspObj$_{o_i}$);
    updateCP(experiment);
    updateM(experiment);
    open gripper
end



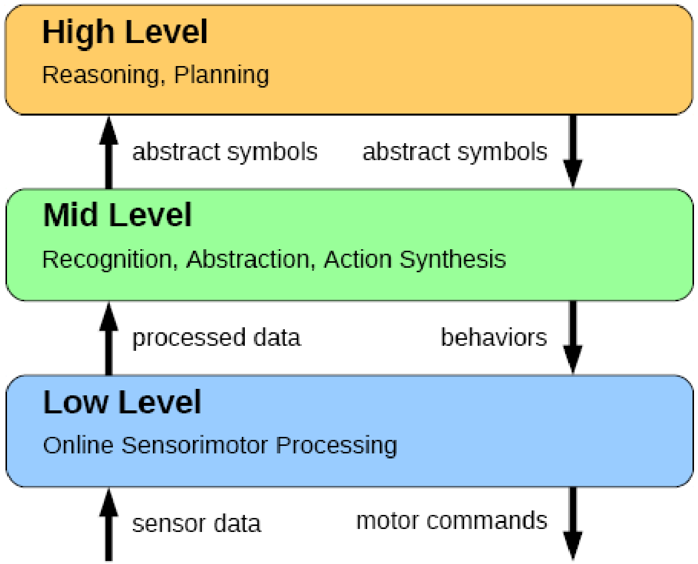
Grounding of objects and grasping affordances: Definition of the problem

- Given
  - an agent being able to grasp and
  - an arbitrary (rigid, edge-dominated) object in a scene the agent does not know anything about beforehand
- Without any supervision, the agent is supposed to
  - find out that there is a (novel) object in the scene,
  - compute a representation of the object and memorize it,
  - use the memorzied representation to recognize a new appearance of the object in the scene and detect its pose,
  - learn how to grasp the object in a way that allows for an optimal grasp in a given situation.
- Basically it can be read as: Learning from 'scratch'
  - that there is an object,
  - how it looks like,
  - and how to grasp it.

oac^{graspObjPlan}: Grasping for Planning

- State space of discrete attributes
  - E.g., no knowledge about where to grasp is coded

Properties	
focusOfAttn(X)	A predicate indicating that object X is the agents focus of attention.
clear(X)	A predicate indicating no object is stacked in X.
gripperEmpty	A predicate describing whether the robot's gripper is empty or not.
inGripper(X)	A predicate indicating that the robot is holding object X in its gripper.
inStack(X,Y)	A predicate indicating that object X is in a stack with object Y at its base.
isIn(X,Y)	A predicate indicating that object X is stacked in object Y.
onShelf(X)	A predicate indicating that object X is on the shelf.
onTable(X)	A predicate indicating that object X is on the table.
open(X)	A predicate indicating that object X is open.
radius(X) = Y	A function indicating that the radius of object X is Y.
pushable(X)	A predicate indicating that object X is pushable by the robot.
reachable(X)	A predicate indicating that object X is reachable for grasping by the gripper.
shelfSpace = X	A function indicating that there are X empty shelf spaces.



$oac^{graspObjPlan}$: Using $oac_o^{graspObj}$ for Planning

- Prediction T:

Name	Initial Conditions	Prediction
$oac^{graspObjPlan}$	focusOfAttn(X)	inGripper(X)
	reachable(X)	not(gripperEmpty)
	clear(X)	not(onTable(X))
	gripperEmpty	
	onTable(X)	

- Learning: update T
  - The model learns the change to each attribute (effects) by treating the learning problem as a set of binary classification problems, with one classier for each feature.

Plan using grasping: Downward Congruency

$$\text{experiment}_{topLev} = \texttt{execute}(\texttt{oac}^{\texttt{graspObjPlan}})$$
$$\quad \text{experiment}_{midLev} = \texttt{execute}(\texttt{oac}_o^{\texttt{graspObj}})$$
$$\quad \texttt{updateCP}(\text{experiment}_{midLev})$$
$$\quad \texttt{updateM}(\text{experiment}_{midLev})$$
$$\texttt{updateT}(\text{experiment}_{topLev})$$
$$\texttt{updateM}(\text{experiment}_{topLev})$$

- Execution of high level oac on the planning level
  - acting in a discrete state space
  - triggers execution of the mid-level oac taking care of the optimal pose for the grasp based on the experiments made so far
- Planning with grounded entities
- Learning is an ongoing process that is taking place at all times at all levels

OACs: Building Blocks for Cognition

- OACs are formed and refined through sensoriomotor experience expressed in *experiments*
- *Learning* is taking place *all times at all levels*
  - as a process parallel to the processes steared by, e.g., plans or automized behaviors.
- OACs can be *chained* to create complex behaviors or plans
- *Grounding* of symbolic entities used for planning can be achieved by the interplay of OACs
- Statements about *success likelihoods* of behaviours and plans can be made based on long-term statistics M