Object Action Complexes as an Interface for Planning and Robot Control

Christopher Geib, Kira Mourão, Ron Petrick,
Nico Pugeault, and Mark Steedman
School of Informatics
University of Edinburgh
Edinburgh EH8 9LW, Scotland
Email: cgeib@inf.ed.ac.uk

Norbert Krueger
The Maersk Mc-Kinney Moller Institute
University of Southern Denmark
DK-5230 Odense M, Denmark

Florentin Wörgötter
Institute for Informatics
University of Göttingen
37083 Göttingen, Germany

***Abstract*— Much prior work in integrating high-level artificial intelligence planning technology with low-level robotic control has foundered on the significant representational differences between these two areas of research. We discuss a proposed solution to this representational discontinuity in the form of object-action complexes (OACs). The pairing of actions and objects in a single interface representation captures the needs of both reasoning levels, and will enable machine learning of high-level action representations from low-level control representations.**

I. Introduction and Background

The different representations that are effective for continuous control of robotic systems and the discrete symbolic AI presents a significant challenge for integrating AI planning research and robotics. These areas of research should be able to inform one another. However, in practice, many collaborations have foundered on the representational differences. In this paper, we propose the use of object-action complexes[1] to address the representational difference between these reasoning components.

The representations used in the robotics community can be generally characterized as vectors of continuous values. These vectors may be used to represent absolute points in three dimensional space, relative points in space, joint angles, force vectors, and even world-level properties that require real-valued models [2]. Such representations allow system builders to succinctly specify robot behavior since most if not all, of the computations for robotic control are effectively captured as continuous transforms of continuous vectors over time. AI representations, on the other hand, have focused on discrete symbolic representations of objects and actions, usually using propositional or first-order logics. Such representations typically focus on modeling the high-level conceptual state changes that result from action execution, rather than the low-level continuous details of action execution.

Neither of the representational systems alone cover the requirements for controlling deliberate action, however, both levels seem to be required to produce human level behavioral control. Our objective is to propose an interface representation that will both allow the effective exchange of information between these two levels and the learning of high level action representations on the basis of the information provided by the robotic control system.

Any such representation must provide clear semantics, and be easily manipulable at both levels. Further it must leverage the respective strengths of the two representation levels. In particular, the robotic control system's access to the actual physical state of the world through its sensors and effectors is essential to learning the actions the planning system must reason about. Each low-level action executed by the robot offers the opportunity to observe a small instantiated fragment of the state transition function that the AI action representations must capture. Therefore, we propose that the robotic control system provide fully instantiated fragments of the planning domains state transition function, that is captured during low-level execution, to the high-level AI system to enable the learning of abstract action representations. We will call such a fragment an *instantiated state transition fragment (ISTF)*, and define it to be a situated pairing of an object and an action that captures a small, but fully instantiated, fragment of the planning domain's state transition function. The process of learning domain invariants from repeated, reproducible instances of very similar ISTFs will result in generalizations over such instances that we will call *object-action complexes (OACs)*. To see how this is done, the rest of this paper will first discuss a detailed view of a robot control system, then we will discuss an AI planning level description of the same domain. We will then more formally define ISTFs and OACS, show how ISTFs can be produced by the robot control system, and how OACs relate to the AI planning level description. We will then discuss the learning of OACs on the basis of ISTFs.

To do all this, we require a particular domain for the robot to interact with. Imagine a relatively standard but simple robot control scenario illustrated in Figure 1. It consists of an arm with a gripper, a table with two light colored cubes and one dark colored cube. The robot has the task of placing the cubes into a box, also located on the table. We will also assume the robot is provided with a camera to view the objects in the domain. However, at the initial stage, the system does not have any knowledge of those objects. The only initial world knowledge available to the system is provided by the vision module, and the hard-coded action reflexes that this visual input can elicit.

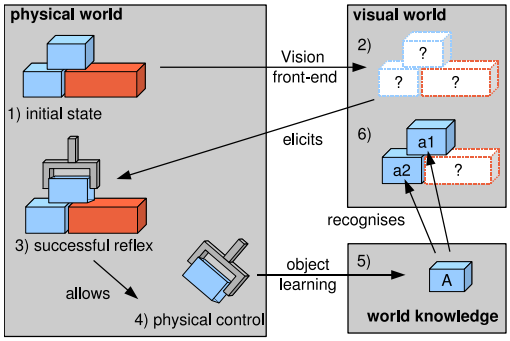

Fig. 1. Illustration of how object classes are discovered from basic uninformed reflex actions.

II. Vision-based Reflex Driven Discovery of Objects and Affordances

We assume a vision front-end based on an *Early Cognitive Vision* framework (see [3]) that provides a scene representation composed of local 3D edge descriptors that outline the visible contours of the scene [4]. Because the system lacks knowledge of the objects that make up the scene, this visual world representation is *unsegmented*: descriptors that belongs to one of the objects in the scene are not explicitly distinct from the ones belonging to another object, or to the background (this is marked by question marks in Figure 1-2). This segmentation problem has been largely addressed in the literature [5], [6], [7]. However, while these segmentation methods are purely vision-based and do not require of the agent to interact with the scene they are unsatisfying for our purpose because they assume certain qualities from the objects in order to segment them: e.g., constant color or texture, moving objects, etc.

Instead we will approach the problem from another angle: we will assume that the agent is endowed with a basic reflex action [8] (Figure 1-3) that is elicited directly by specific visual feature combinations in the unsegmented world representation. The outcome of these reflexes will allow the agent to gather further knowledge about the scene. This information will be used to segment the visual world into objects and identify their affordances.

We will only consider a single kind of reflex here: the agent tries to grasp any planar surface in the scene.[1] The likely locations of such planar surfaces are inferred from the presence of a coplanar pair of edges in the unsegmented visual world. This type of reflex action is described in [8]. Every time the agent executes such a reflex, haptic information allows the system to evaluate the outcome: either the grasp was successful and the gripper is holding something, or it failed and the gripper closed on thin air. A failed attempt drives the agent to reconsider its original assumption (the presence of a graspable plane at this location in the scene), whereas a successful attempt confirms the feasibility of this reflex. Moreover, once a successful grasp has been performed, the agent has gained physical control over some part of the scene (i.e. the object grasped, Figure 1-4). If we assume that we know the full kinematics of the robot's arm (which is true for an industrial robot), it is then possible to segment the grasped object from the rest of the visual world as it is the only part that moves synchronously with the arm of the robot. At this point a new "object" relevant for the higher level planning model is "born".

Having physical control of an object allows the agent to segment it and to visually inspect it under a variety of viewpoints and construct an internal representation of the full 3D shape of the object (see [9]). This shape can then be stored as the description of newly discovered class **A** (Figure 1-5) that affords **grasp-reflex-A** encoding the initial reflex that "discovered" the object.

The object held in the gripper is the first instance **a1** of the class **A**. The agent can use its new knowledge of class **A** to reconsider its interpretation of the scene: using a simple object recognition process (based on the full 3D representation of the class), all other instances (e.g., in our example **a2**) of the class in the scene are identified and segmented from the unknown visual world.

Thus through a reflex-based exploration of the unknown visual world object classes can be discovered by the system until it achieves an informed, fully segmented representation of the world, where all objects are instances of symbolic classes and carry basic affordances.

To distinguish the specific successful instances of the robot's reflexes, we will refer to the specific instance of the reflex that was successful for the object as a particular *motor program*. Note that such motor programs are defined relative to a portion of an object, in our example, the surface that was grasped. We will extend this by assuming *all* motor programs can be defined relative to some object.

The early cognitive vision system [4], the grasping reflex [8] as well as the accumulation mechanism [9] that together provides a segmentation of the local feature descriptors into independent objects currently exist in one integrated system that we will use as a foundation for this architecture.

III. Representing AI Planning Actions

As we have noted, we can also model this robot domain scenario using a formal AI representation. In this case, we will formalize the robot domain using the Linear Dynamic Event Calculus (LDEC) [10], [11], a logical language that combines aspects of the situation calculus with linear and dynamic logics, to model dynamically-changing worlds[12], [13], [14].

Our LDEC representation will define the following actions.

Definition 1: High-Level Domain Actions

- *grasp*(x) – *move the gripper to pick up object* x,
- *ungrasp*(x) – *release the object* x *in the gripper,*
- *moveEmptyGripperTo*(ℓ) – *move an empty gripper to the specified location* ℓ,
- *moveFullGripperTo*(ℓ) – *move a full gripper to the specified location* ℓ.

[1] Note that other kind of reflex actions could be devised to enable other basic actions than grasping.

These actions represent higher level counterparts of some of the motor programs available to the robot controller, but already these actions incorporate elements of the state of the world that are not part of robotic control representations of actions. For instance, *ungrasp* models an action that is quite similar to a motor program that performs this operation. Actions like *moveEmptyGripperTo* and *moveFullGripperTo*, on the other hand, are much more abstract and encode information about the state of the world (i.e. the gripper is empty or full). Note that in this case the actions partition the low-level "move gripper" motor-programs into two separate actions that, as we will see, can more readily be learned from the available ISTFs. This representation will also allow us to bypass the learning of the conditional effects[15] of such actions.

Our LDEC representation will also include a number of high-level properties.

Definition 2: High-Level Domain Properties

- $graspable(x)$ – *a predicate that indicates whether an object x is graspable or not,*
- $gripperLoc = \ell$ – *a function that indicates the current location of the gripper is* ℓ*,*
- $objInGripper = x$ – *a function that indicates the object in the gripper is x; x is nil if the gripper is empty,*
- $objLoc(x) = \ell$ – *a function that indicates the location of object x is* ℓ*.*

Finally, we also specify a set of "exogenous" domain properties.

Definition 3: Exogenous Domain Properties

- $over(x) = \ell$ – *a function that returns a location* ℓ *over the object x,*
- $locOnTable(\ell_1) = \ell_2$ – *a function that returns a location* ℓ_2 *relative to the table (e.g., on the table or in a box) for another location* ℓ_1 *above the table.*

Like the properties in Definition 2, the exogenous properties model high-level features of the domain. However, unlike domain properties that are directly tracked by the high-level AI model; exogenous properties are information provided to the high-level AI system by some external (possibly lower level) source. (We will say more about exogenous properties in Section VI.)

Using these actions and properties we can write LDEC axioms that capture the dynamics of the robot scenario described in Table I). Action precondition axioms describe the properties that must hold of the world to apply a given action (i.e., affordances), while the effect axioms characterize what changes as a result of the action. These axioms also encode the STRIPS assumption: fluents that aren't directly affected by an action are assumed to remain unchanged by that action [16].

We note our LDEC axiomatization is readily able to accommodate the *indexical*, or relative information. For example, an instantiated function like $over(box1)$ represents a form of indexical knowledge, rather than a piece of definite information like the coordinates of the box in three dimensional space. Moreover, our LDEC axiomatization can model spatial relationships expressed with respect to objects. For instance, $moveFullGripperTo(over(box1))$ can represent an action instance that moves the object in the gripper to a location "over *box1*"

TABLE I
LDEC Axiomatization of High-Level Domain Actions

LDEC Action Precondition Axioms

$objInGripper = nil \wedge graspable(x) \Rightarrow affords(grasp(x))$
$objInGripper = x \wedge x \neq nil \Rightarrow affords(ungrasp(x))$
$objInGripper = nil \Rightarrow affords(moveEmptyGripperTo(\ell))$
$objInGripper = x \wedge x \neq nil \Rightarrow affords(moveFullGripperTo(\ell))$

LDEC Effect Axioms

$\{affords(grasp(x))\} \multimap$
$[grasp(x)]\, objInGripper = x \wedge gripperLoc = objLoc(x)$
$\{affords(ungrasp(x))\} \multimap$
$[ungrasp(x)]\, objInGripper = nil \wedge objLoc(x) = locOnTable(objLoc(x))$
$\{affords(moveEmptyGripperTo(\ell))\} \multimap$
$[moveEmptyGripperTo(\ell)]\, gripperLoc = \ell$
$\{affords(moveFullGripperTo(\ell))\} \multimap$
$[moveFullGripperTo(\ell)]\, gripperLoc = \ell \wedge objLoc(objInGripper) = \ell$

Intuitively, the information encoded in a collection of LDEC axioms captures a generalization of the information in a larger set of ISTFs. The action precondition axioms capture information from the initial state of an ISTF and the action executed, while the effect axioms capture the generalities for the initial state to final state mappings from an ISTFs. As such we believe they can be learned from the ISTFs.

It is easy to show that this representation supports high-level planning. For instance, with these axioms it is trivial for an AI planner to construct the following simple plan:

$$[grasp(obj1); moveFullGripperTo(over(box1)); ungrasp(obj1)],$$

to put an object $obj1$ into $box1$, from a state in which the robot's gripper is empty. However, building even this sort of simple plan from first principles is well beyond the capability of the robot controller alone.

So far we have shown that a low level robot controller is capable of producing ISTFs for a domain, we have shown a way an AI level planner could formalize the same domain, and we have shown the necessity of using the AI planner with the robot controller to produce high level behavior. In the remainder of the paper we will outline a process whereby we can learn the AI level representation from the ISTFs produced by the robot controller.

IV. Bridging Robot Control and Planning with ISTFs and OACs

With these two views of the problem in hand, we now, consider how we can bridge the two representational levels. We see that we can obtain a wealth of object-centric information each time the robotic system successfully grasps an object: the object grasped, the type of grasping reflex used, the relative position of the gripper, the fact that the object has been

effectively grasped and is now in the gripper instead of being on the table, etc. This association of before and after states of a particular "grasp" motor program with a specific domain object meets our definition of an ISTF. It completely describes a fragment of the planning domain's transition function.

We more formally define an ISTF as a tuple $\langle s_i, mp_j, Obj_{mp_i}, s_{i+1} \rangle$ comprised of the initial sensed state of the world s_i, a motor program instance mp_j, the whole object containing the component the motor program was defined relative to Obj_{mp_i}, and the state that results from executing the motor program s_{i+1}. Keep in mind that the state representations for this ISTF contain all of the information the robot has about the two states of the world. Some of which may be relevant some of which may be completely irrelevant to the outcome of the action.

It will be the task of the learning module to abstract away this irrelevant information from the ISTFs to produce OACs that contain only the relevant instantiated information needed to effectively predict the applicability of the action and the likely effects of the action. This is only possible if the system is provided with multiple encounters with reproducible ISTFs. Thus as the system repeatedly interacts with the world it is presented with multiple very similar ISTFs which it generalizes into OACs, thereby learning a representation that is not unlike the one we specified in the previous section.

On this basis, we define an OAC as a generalized ISTF tuple: $\langle S_i, MP_j, Obj_k, S_{i+1} \rangle$ comprised of two abstracted states (S_i and S_{i+1}) a set of motor programs MP_j, and an object class Obj_k. The initial state of the world, S_i, is abstracted to contain only those properties that are necessary for any of the set of motor-programs in MP_j when acting on an object of class Obj_k to result in an state that is satisfied but the abstracted state definition S_{i+1}. Thus such an OAC contains all of the information found in our initial LDEC definitions for this domain.

Given the parallels to LDEC representations how are OACs different? The answer to this is, a very subtle point. OACs constrain the kinds of LDEC rules that can be learned. First OACs distribute information in a subtly different manner than LDEC rules. An OAC contains information normally found in two different parts of the LDEC representation. By bringing together information found in precondition rules with the effect rules and the object in question they allow learning to take place that previously couldn't have been accomplished. Second the heavy use of the object and the object centeredness of OACS produce LDEC representations that easily lend themselves to a simple forward looking planning algorithm that is heavily directed by the affordances of the available objects. Third and finally the use of OACs constrains the LDEC representations to a simple form of axioms that are easier to learn. For example, without more complex machinery, OACs induced from ISTFs are not able to create action representations with conditional effects. Learning such conditional effects of actions is a significant problem for other approaches.

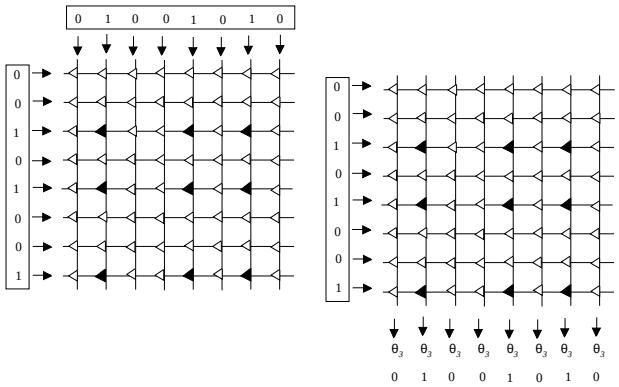

Fig. 2. Hetero-associative net: Storage and Retrieval

V. Learning Action Representations

The ability of a low level robotic control system to identify world-level objects only takes us part of the way to kind of representation we have just described. We must learn from the ISTFs coherent, high-level actions. Our current proposal for learning such action representations involves the use of Willshaw nets or Associative Nets(AN).

ANs were first was described in [17], [18] following early work by [19] and [20] extended by [21] and [22]. They illustrate three basic properties which are characteristic of mechanisms involved in phenomena of human memory and attention: 1) non-localized storage ("Distributivity"), 2) recovery of complete stored patterns from partial or noisy input ("Graceful Degradation"), and 3) effective functioning even in the face of damage ("Holographic Memory").

ANs associate pairs of input and output vectors using a grid of horizontal input lines and vertical output lines with binary switches (triangles) at the intersections (Figure 2). To store an association between the input vector and the output vector, switches are turned on (black triangles) at the intersection of lines which correspond to a 1 in both input and output patterns.

To retrieve the associate of the input, a signal is sent down each input line corresponding to a 1 in the input. When this signal encounters an "on" switch, it increments the signal on the corresponding output line by one. The output lines are then thresholded at a level corresponding to the number of "on" bits in the input. If we store an input pattern with itself as output (an *auto-associative net*), ANs can be used to complete partial patterns, as needed to recall perceptually non-evident properties of objects, such as the fact that the red cube on the table affords grasping. This is exactly the information that is encoded in action precondition axioms. Further it is worthwhile to notice that all of the information needed for this AN is available in each new instance of an ISTF. In this case, the input and output patterns for the AN are the same: the initial state, action, and object for a cluster of reproducable ISTFs observed in the course of interacting with the world. We thereby use repeated presentations of very similar ISTFs (clustered by action and object) to train auto-associative ANs to effectively store and retrieve associations between the LDEC action precondition axioms and the property of *affording* such

LDEC operators.

Now consider the LDEC style effect axioms. Rather than using an auto-associative net we can use a hetero-associative network for this task. In this case, we again use the initial state, action, and object as the input pattern from each ISTF, however as the output pattern we use the resulting state from the ISTF. This will allow us to learn and retrieve the state-change transitions associated with LDEC operators, with states represented as sparse vectors of relevant facts or propositions.

Thus, we hypothesize that such associations can be learned in ANs using repeated presenations of reproducable ISTFs using the Perceptron Learning Algorithm (PLA). We replace the binary AN switches with continuous valued switches and use multiple ISTFs that have the same action, object, and resulting state and the PLA to adjust the weights on the relevant switches. We believe that such an approach can learn consistent state changes or actions, and learn the association between preconditions and associated affordances.

More specifically, in the envisioned scenario, as the robot controller explores the world, successful grasps will produce ISTFs. On the basis of multiple reproducable experiences of particular ISTFs we can learn the instantiated versions of the precondition axioms and the effect axioms for the robots actions. The resulting state in each ISTF will vary only in terms of the object-type grasped and the grippers pose. Further, the invariants can be learned as a basis for classifying the world into object classes and action types. As we have discussed, identifiers for actions-types can then be associated with the input conditions for the action via an auto-associative net. Such affordances are added by adding new input and output lines to the net for the new affordance, and using the existing learning algorithm.

This network can be presented with a possibly incomplete set of properties representing the current state of the world, and used to retrieve a complete model of the world state, including non-perceptually available associates including the affordances and object classes.(Figure 3) For ease of exposition, in this and the following figures we will continue to show weights of 0 and 1. The full pattern including affordances can then be input to the other hetero-associative net, and used to retrieve the effects of carrying out particular actions. (Figure 4).

If the output states and affordances are the same following two different grasp actions for a particular input state, then clearly the effects (as far as the learner and planner are concerned) of the two grasps are the same for that input. If the effects are the same for all inputs then the grasps are equivalent and can be collapsed together. We discuss this next.

A. Learning Multiple Grasp Actions

Recall from our discussion of the high-level action *grasp* that at the lower level there may in fact be many low-level grasps available to the robot at any point. While many of these grasping actions may have effects that are indistinguishable from one another, there will also be grasping actions that result in very different effects. Given this, and our desire to avoid the difficulties of learning actions with conditional

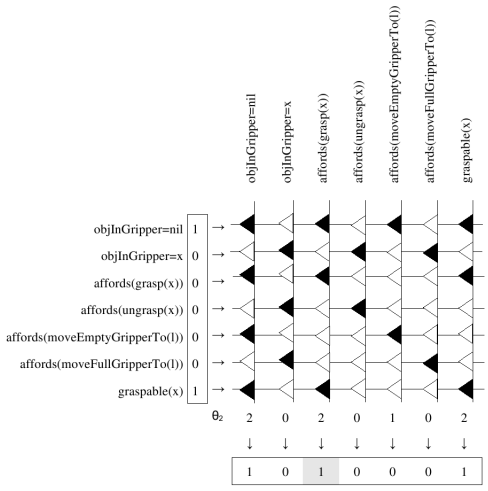

Fig. 3. Retrieval of *affords*(*grasp*(*x*)) from *objInGripper* = *nil* ∧ *graspable*(*x*) in the loaded auto-associative net

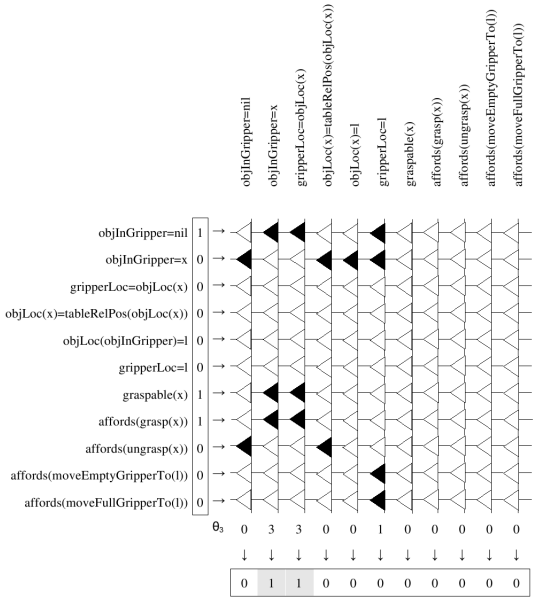

Fig. 4. Retrieval of effect *grasp*(*x*) from the hetero-associative net

effects, it becomes clear that we will need multiple grasp actions at the higher level of abstraction. To distinguish these actions and their effects during planning and learning we will introduce multiple predicates indicating "graspability" by particular motor programs.

Our learning process now operates as follows: when an object is "born" at the lower level of representation (See Section II), the message for the addition of the object (e.g., *obj*23) should include an identifier for the specific action that was executed (e.g., *grasp*28, *grasp*95, etc.) as well as asserting the existence of a new predicate indicating the object has that action as an affordance (e.g., *affords*(*grasp*28(*obj*23))).[2] This predicate is added to the AN and can be used for learning.

[2]Although we only consider grasping actions, we assume other actions, such as pushing, also result in the "birth" of an object-affordance complex.

We make the strong assumption that the invariants of the domain map onto the input units of the associative network, which we assume in animals have evolved to this end and for the robot must be built in, are such as to ensure that when distinct low-level motor programs are indistinguishable at the higher level of abstraction, they will automatically be classified as instances of a single action.

VI. Using Learned Abstract Action Representations

We have described a process that results in learning abstracted action representations that should be close to the LDEC representations we have sketched for this domain. However, by abstracting the actions in this way there remains a number of open concerns we must address.

a) Using Learned Action Knowledge with New Objects: All new objects are initially associated with "new" actions. Our problem is to associate a previously unseen motor-program object pair with an existing high-level action or to mark it as a new action that must be learned at the high level.

b) Using Learned Action Knowledge for Execution: It will be necessary to convert our learned abstract actions to specific motor programs for execution. Keeping the list of the motor program-object pairs abstracted by each high-level action should address this issue. Since all abstracted pairs for a given action should be equivalent, we suggest selecting any one that matches the object bound in the high level plan.

c) Learning Exogenous Domain Properties: Although we have described a process for learning certain domain properties, the question remains as to how we will learn the exogenous properties given in Definition 3. For the present we simply assume the presence of *over* as an exogenous domain property that is computed by a lower level function.

VII. Conclusion

This paper has argued that object-action complexes (OACs) grounded from instantiated actions in robot control-space, can be used as an interface between the very different representation languages of robot control and AI planning. We have shown that OACS can be embodied in an Associative Net, and that they can be learned by a very simple machine-learning algorithm. Almost all of these claims are unproven but we offer them as defining a research program that we shall be pursuing in the coming years in order to combine existing robot platforms and existing planners based on LDEC and other situation/event calculi.

References

[1] B. Hommel, J. Müsseler, G. Aschersleben, and W. Prinz, "The theory of event coding (tec): A framework for perception and action planning." *Behavioral and Brain Sciences*, vol. 24, pp. 849–878, 2001.

[2] R. Murray, Z. Li, and S. Sastry, *A mathematical introduction to Robotic Manipulation*. CRC Press, 1994.

[3] N. Krüger, M. V. Hulle, and F. Wörgötter, "Ecovision: Challenges in early-cognitive vision," *International Journal of Computer Vision*, 2006.

[4] N. Pugeault, F. Wörgötter, , and N. Krüger, "Multi-modal scene reconstruction using perceptual grouping constraints," in *Proceedings of the 5th IEEE Computer Society Workshop on Perceptual Organization in Computer Vision, New York City June 22, 2006 (in conjunction with IEEE CVPR 2006)*, 2006.

[5] J. Shi and J. Malik, "Motion segmentation and tracking using normalized cuts," in *ICCV*, 1998, pp. 1154–1160. [Online]. Available: citeseer.nj.nec.com/shi98motion.html

[6] F. Moscheni, S. Bhattacharjee, and M. Kunt, "Spatiotemporal segmentation and based on region merging," *IEEE transactions on Pattern Analysis and Machine Intelligence*, 1998.

[7] Y. Deng and B. Manjunath, "Unsupervised segmentation of color-texture regions in images and videos," *IEEE transactions on Pattern Analysis and Machine Intelligence*, 2001.

[8] D. Aarno, J. Sommerfield, D. Kragic, N. Pugeault, S. Kalkan, F. Wörgötter, D. Kraft, and N. Krüger, "Integration of elementary grasping actions and second order 3d feature relations for early reactive grasping," *submitted to 2006 IEEE-RAS International Conference on Humanoid Robots*, submitted.

[9] N. Krüger, M. Ackermann, and G. Sommer, "Accumulation of object representations utilizing interaction of robot action and perception," *Knowledge Based Systems*, vol. 15, pp. 111–118, 2002.

[10] M. Steedman, "Temporality," in *Handbook of Logic and Language*, J. van Benthem and A. ter Meulen, Eds. Amsterdam: North Holland/Elsevier, 1997, pp. 895–938.

[11] ——, "Plans, affordances, and combinatory grammar," *Linguistics and Philosophy*, vol. 25, pp. 723–753, 2002.

[12] J. McCarthy and P. J. Hayes, "Some philosophical problems from the standpoint of artificial intelligence," *Machine Intelligence*, vol. 4, pp. 463–502, 1969.

[13] D. Harel, "Dynamic logic," in *Handbook of Philosophical Logic, volume II*, D. Gabbay and F. Guenthner, Eds. Dordrecht: Reidel, 1984, pp. 497–604.

[14] J.-Y. Girard, "Linear logic," *Theoretical Computer Science*, vol. 50, pp. 1–102, 1987.

[15] E. P. D. Pednault, "ADL: Exploring the middle ground between STRIPS and the situation calculus," in *Proceedings of the First International Conference on Principles of Knowledge Reprensentation and Reasoning (KR-89*, R. J. Brachman, H. J. Levesque, and R. Reiter, Eds. San Mateo, CA: Morgan Kaufmann Publishers, 1989, pp. 324–332.

[16] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, pp. 189–208, 1971.

[17] D. Willshaw, P. Buneman, and C. Longuet-Higgins, "Non-holographic associative memory," *Nature*, vol. 222, pp. 960–962, 1969.

[18] D. Willshaw, "Holography, association and induction," in *Parallel Models of Associative Memory*, G. Hinton and J. Anderson, Eds. Hillsdale, NJ: Erlbaum, 1981, pp. 83–104.

[19] K. Steinbuch, "Die lernmatrix," *Kybernetik*, vol. 1, pp. 36–45, 1961.

[20] J. Anderson, "A memory storage model utilizing spatial correlation functions," *Kybernetik*, vol. 5, pp. 113–119, 1968.

[21] F. T. Sommer and G. Palm, "Bidirectional retrieval from associative memory," in *Advances in Neural Information Processing Systems*, M. I. Jordan, M. J. Kearns, and S. A. Solla, Eds., vol. 10. The MIT Press, 1998.

[22] T. Plate, "Holographic reduced representations: Convolution algebra for compositional distributed representations," in *Proceedings of the 12th International Joint Conference on Artificial Intelligence, San Mateo CA*. San Francisco, CA: Morgan Kaufmann, 1991, pp. 30–35.