



FP6-004381-MACS

MACS

Multi-sensory Autonomous Cognitive Systems Interacting with Dynamic
Environments for Perceiving and Using Affordances

Instrument: Specifically Targeted Research Project (STReP)

Thematic Priority: 2.3.2.4 Cognitive Systems

**D4.2.1+4.3.1 Tentative Proposal for a Formal Theory of Affordances
Tentative Proposal for an Affordance Support Architecture
Prototype: Affordance-Based Motion Planner**

Due date of deliverable: March 31, 2005
Actual submission date: October 10, 2005

Start date of project: September 1, 2004

Duration: 36 months

Linköpings Universitet (LiU-IDA)

Revision: Version 1

Project co-funded by the European Commission within the Sixth Framework Programme (2002–2006)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

EU Project



Tentative Proposal for a Formal Theory of Affordances

Tentative Proposal for an Affordance Support Architecture

Prototype: Affordance-Based Motion Planner

WP6 Combined Deliverable Report: D4.2.1 and D4.3.1

Patrick Doherty, Torsten Merz, Piotr Rudol, Mariusz Wzorek

*Number: **MACS/4/2.1***

WP: 4.2,4.3

Status: draft, version 1 (for internal use only)

Created at: August 31, 2005

Revised at: October 10, 2005

FhG/AIS

Fraunhofer Institut für

Autonome Intelligente Systeme, Sankt Augustin, D

JR_DIB

Joanneum Research Graz, A

LiU-IDA

Linköpings Universitet, Linköping, S

METU-KOVAN

Middle East Technical University, Ankara, T

OFAI

Österreichische Studiengesellschaft für Kybernetik,
Vienna, A

This research was partly funded by the European Commission's 6th Framework Programme IST Project MACS under contract/grant number FP6-004381. The Commission's support is gratefully acknowledged.

© FhG/AIS 2005

Author addresses:

Patrick Doherty
Linköpings universitet
Institutionen för Datavetenskap
581 83 Linköping, Sweden



Fraunhofer Institut für
Autonome Intelligente Systeme
Schloss Birlinghoven
D-53754 Sankt Augustin
Germany

Tel.: +49 (0) 2241 14-2683
(Co-ordinator)

Contact:
Dr.-Ing. Erich Rome



Joanneum Research
Institute of Digital Image Processing
Computational Perception (CAPE)
Steyrergasse 9
A-8010 Graz
Austria

Tel.: +43 (0) 316 876-1769

Contact:
Dr. Lucas Paletta



Linköpings Universitet
Dept. of Computer and Info. Science
Linköping 581 83
Sweden

Tel.: +46 13 24 26 28

Contact:
Prof. Dr. Patrick Doherty



Middle East Technical University
Dept. of Computer Engineering
Inonu Bulvari
TR-06531 Ankara
Turkey

Tel.: +90 312 210 5539

Contact:
Prof. Dr. Erol Şahin



Österreichische Studiengesellschaft
für Kybernetik (ÖSGK)
Freyung 6
A-1010 Vienna
Austria

Tel.: +43 1 5336112 0

Contact:
Prof. Dr. Georg Dorffner

Contents

1	Introduction	1
2	State of the Art	4
3	Affordances	6
3.1	Gibson’s view on affordances	6
3.2	A definition of affordances	6
4	Ontology	7
4.1	Summary	10
4.2	Operational Remarks	11
5	Classification	11
5.1	Operational Remarks	13
6	Tolerance Spaces and Similarity	13
6.1	The basic idea	13
6.2	A Formal Framework	15
6.3	Defining Tolerance on Complex Representational Structures	18
7	Formalizing Actions	19
8	Formalizing situations	20
8.1	A formal framework	20
9	A Scenario	22

10 Some Initial affordances	24
10.1 Standability	26
10.2 Turnability	30
10.3 Traversibility	31
10.4 Reachability	34
10.5 Graspability	34
10.6 Liftability	36
11 Affordance Based Motion Planning	36
11.1 The motion planning problem	37
11.2 Our approach	39
11.3 Behaviors and Dataflow	39
11.4 Results	43
12 Architectural Support for Affordances	43
12.1 Introduction	43
12.2 Architectural Components for Affordance Support	44
12.3 Affordance Support from Different Perspectives	51
12.4 Affordance Support from a Robot Behavior Perspective	51
12.5 Affordance Support from a Module Perspective	53
13 Relation to Other Workpackages in MACS	54
14 Terminology	57
15 Conclusions	57

1 Introduction

The contents of this deliverable should be viewed as a *living document* and one that requires a number of iterations before it is ready for the general public. It should be read in this spirit as a draft requiring more iterations and changes. It should not be disseminated outside the project group.

Since we consider this a draft for revision, little effort has been spent formatting and assuring proper placement of figures. There are also sections which remain blank due to work remaining regarding conceptualization or simply because we consider writing about these sections premature. In addition, references to related work will eventually be added to this document

This document is intended as an integrated version of two specific deliverables from the WP6 group based on two tasks

- Task 4.2 – **Specify affordance representations** - Develop a number of alternative representations and data structures for affordances which can be used in robotics platforms at various levels of abstraction.
- Task 4.3 – **Specify and implement a software subunit for affordance support** - Provide a specification and implementation for a software subunit in a robotics platforms which supports the use of affordances as specified in task 4.2. This subunit should take account of feedback loops from sensors to representations of objects in the environment to action execution on these objects and back to sensor usage.

The deliverables are described as

1. D4.2.1 - A conference or journal article summarizing the results of task 4.2
2. D4.3.1 - A specification of a software module for affordance representation (document)

As stated in the progress reports, we feel it is still too early to submit the results associated with task 4.2 and have chosen instead to combine these with the software module specification. We do believe that a number of conference or journal articles are possible based on a few more iterations of the material provided in this joint deliverable.

This deliverable contains the following:

1. A short state of the art section specific to the activities associated with the WP4 group is provided.
2. We then begin with a definition of affordance and a logical specification of that definition which provides the conceptual framework for thinking about affordances.
3. We introduce the beginning of an ontology and some of the representational structures required to reify the ontology in a robotic system.
4. We consider the very important topic of classification and provide a means of measuring similarity between entity structures which are intended as primary representational structures for objects in the broad sense. In doing this, we introduce the concept of tolerance spaces.
5. We believe the use of *situations* will play a central role in the formal characterization of affordances and provide some brief comments about them. A formal characterization is in the works and a first iteration is provided.
6. We provide a description of a robotic environment and scenarios which we have used to do some empirical work necessary to shape our understanding of affordances.
7. We then target a number of affordances deemed useful for such scenarios, specify them pragmatically and then try and relate them to the logical specification provided at the beginning of the report. In doing this, we also instantiate the idea of affordance maps, an iconic representational structure which is fundamental to our approach.
8. We then present what we call "affordance-based motion planning", an attempt to integrate the intuitions associated with the idea of affordances with sample-based motion planning techniques. A prototype has been implemented and used in our experimentation.
9. We also present a number of behaviors used by our robot system which are specified as hierarchically concurrent state machines and used in the motion planning work.
10. We then proceed to the important section on architectural support for affordances where a number of modules are described.
11. We then proceed to integrate these modules in one coherent structure which should be considered the 1st draft proposal of an architecture for affordance support. We look at these ideas from two perspectives, one a behavior-based and the second a more neutral module based approach.
12. We then relate the ideas proposed in the architecture with some of the on-going work in the other packages. In particular, we show that the architecture for affordance support plays a dual role. In one mode it is intended to be used by the robotic system when achieving tasks which require the use and monitoring of affordances. In the second mode, it is intended to be used for on-line learning of affordances in the sense that it can generate the raw time-series data required by the learning module to

learn correlations between robot action, movement and task achievement, isolating invariants which can then be defined as affordances for use in the first mode.

2 State of the Art

Before proceeding to the new results in this deliverable, it is worthwhile summarizing some state of the art as regards affordances with a focus on that which is relevant to the work in this report. The best source for state of the art in psychology is perhaps the *Journal of Ecological Psychology*. The best source for state of the art in robotics is Arkin [Ark98] and Murphy [Mur00; Mur99]. Certainly, there have been many conference papers related to affordances, but both Arkin and Murphy offer the best summary of affordances and their use in pragmatic robotic systems.

The behavior-based robotics movement can be characterized by a number of features. Some of the most important are the recognition of the very tight coupling between sensing and acting, avoiding global construction of models in terms of symbolic representations of knowledge, and decomposition of task achieving behavior into highly granular behaviors which often execute concurrently and where perceptual processing is specific to the needs of the behaviors in question.

This latter feature is particularly important and recent advances in the study of perception have been driven by the symbiotic nature of action and perception. In particular, action-oriented perception is guided by the belief that perceptual processing should be tuned to the need of specific motor activities and active perception is guided by the belief that motor control can be used to enhance perceptual processing by positioning sensors in locations appropriate for the action at hand.

In this context, Gibson's ideas about perception and affordances can be viewed as providing inspiration for much of the behavior-based robotic's movement. An affordance is a relation between an agent and an environment which provides a capability or potential for action. The potential is direct in the sense that perception is direct. Perceptual entities are not semantic abstractions but opportunities for action mediated by a specific relation between the agent and its environment. This implies a very tight loop between actions (or behaviors) and opportunities the environment affords (perceptual entities). The behaviors and perceptual entities are symbiotic.

If this is the case, then one might ask the question what the MAC's project can achieve that has not already been achieved in the behavior-based robotics movement or in the related areas of action-oriented perception and active perception? We believe there are several answers to this.

1. The behavior-based robotic movement has obviously received inspiration from the work of Gibson and his ideas about affordances. On the other hand, the affordance concept is implicit in the framework and has not been the focus of the research. In fact, many papers appeal quite loosely to the affordance idea without making any claims that this is specifically what the work is about. In our case, the affordance concept is intended to be explicit in the framework and is the focus of the research. This direct focus on affordances influences the methodology used in the develop-

ment of affordance-based robotic architectures and with functionalities not normally associated with affordances such as motion planning as will be shown in this report.

2. Affordances are intended to be built into a robotics architecture as 1st-class citizens. There is explicit support for the concept in the architecture and the hypothesis is that the resulting performance of the robot will benefit in terms of robustness and generality. In fact, these are essentially the only criteria that can be used to empirically evaluate whether an affordance-based robotic system is better than a non-affordance-based robotic system.
3. The MACS project also focusses on how the tight couplings between behavior and perception can be learned by the robotic system itself. In other words, an infrastructure is set up for the robot to learn affordances through acting and observing the effects of acting on the environment. There has been some work in this area, but no principled approach has yet been proposed as to how an on-line affordance learning system should be designed.
4. The behavior-based robotics approach avoids any use of internal symbolic models, although hybrid architectures which combine behavior-based approaches with more traditional sense-plan-act approaches do. In the case of MACs, we are interested in extending the affordance idea to the semantic level where functional classification and use of objects [SB89] is a focus of research and hopefully part of the affordance-based architecture we are developing. This moves MACS away from a standard Gibsonian approach to affordances where symbolic representations are not condoned to one where representations are very much a part of the architectural framework.

The relation between perception and action is at the heart of the affordance concept. Since the MACS project is not only concerned with the non-representational approach to affordances but also an approach where representation is condoned, traditional theories of action and change become central to the study and logical characterizations of affordances become a valid subject of study. A description of state of the heart in regard to formal specification of affordances will be brief because there are few articles on the topic [Ste00]. A very good starting point is the work of Chemero [Che03] which very briefly attempts to set up a definition of what an affordance is in terms of a predicate representing a relation between an agent and its environment. This is the starting point for the formal specification we construct in this report. Another important source of inspiration is the situation semantics movement which specifically sets out to provide a semantic or meaning theory for language grounded in a realist philosophy which uses situations and constraints between situations as its basis. In some sense, certain types of such constraints can be viewed as affordances. Their intent is to transfer information from one situation to another or better yet characterize the relation between situations of different types as information bearing.

In summary, state of the art regarding representation of affordances and affordance support architectures is sparse. There is little to go on other than inspiration provided by Gibson's ideas, pragmatic use of his ideas in behavior-based robotics and some intuition and direction provided in Chemero's work.

3 Affordances

3.1 Gibson's view on affordances

Gibson provides a loose definition of an affordance:

The *affordances* of the environment are what it *offers* the animal, what it *provides* or *furnishes*, either for good or ill. The verb to *afford* is found in the dictionary, but the noun *affordance* is not. I have made it up. I mean by it something that refers both to the environment and the animal in a way that no existing term does. It implies the complementarity of the animal and the environment. ([Gib79], p. 127)

Gibson also states that affordance recognition is independent of purpose,

The observer may or may not perceive or attend to the affordance, according to his needs[purpose], but the affordance, being invariant, is always there to be perceived. ([Gib79], p. 139)

3.2 A definition of affordances

Based on Gibson's and other work related to affordances. the following definition for an affordance is proposed:

Definition 1 (Affordance) *An affordance is a relation between an agent and its environment which affords a capability. The agent/environment relation affords a capability if the agent*

1. *has the capacity to recognize that it is in such a relation between itself and its environment, and it*
2. *has the ability to act to bring about that capability.*

3.2.1 A logical specification

We require a logical specification for an affordance. In order to do this, we use the definition above and introduce a number of predicates related to its specification. Later in the report,

we will provide sufficient conditions for each of these and then relate the definition to actions and situations. The longer term goal is to develop a logic for affordances.

We can introduce the following predicate as an initial representation:

$$\text{Affords}(\text{agent}, \text{capability}, \text{environment}) \quad (1)$$

The first requirement of the definition states that the agent has the capacity to recognize that it is in such a relation. For the moment, let's introduce a new predicate for being capable of recognizing or perceiving such relations,¹

$$\text{CanPerceive}(\text{agent}, \text{affords}(\text{agent}, \text{capability}, \text{environment}), \text{env}) \quad (2)$$

The second requirement of the definition states that the agent also has the ability to act to bring about the capability. For the moment, let's introduce an additional predicate for the ability to execute an action in an environment,

$$\text{CanDo}(\text{agent}, \text{action}, \text{environment}) \quad (3)$$

The effect of a successful execution of an action also has to be able to achieve the capability,

$$\text{CanAchieve}(\text{agent}, \text{action}, \text{capability}, \text{environment}) \quad (4)$$

Putting this all together, we can now specify the sufficient conditions for an affordance to be present in an agent's environment,

$$\begin{aligned} \forall \text{agt}, \text{cap}, \text{env}. [\text{CanPerceive}(\text{agt}, \text{affords}(\text{agt}, \text{cap}, \text{env}), \text{env}) \\ \wedge \exists \text{act}. [\text{CanDo}(\text{agt}, \text{act}, \text{env}) \wedge \text{CanAchieve}(\text{agt}, \text{act}, \text{cap}, \text{env})] \rightarrow \\ \text{Affords}(\text{agent}, \text{capability}, \text{environment})] \end{aligned} \quad (5)$$

Note that the affordance is always there in the environment for the agent to perceive provided the agent has the sensory ability to perceive the affordance and the motoric abilities to use the capability associated with the affordance. In this sense, the affordance is invariant whether the agent perceives it or not or attends to it or not. The important point is that the agent has the ability to both perceive and attend to the affordance according to its needs.

4 Ontology

Gibson classifies the embedding environment of an agent into the following:

- surfaces

¹Note that the original predicate is now reified as a functional argument and that the predicate `CanPerceive` is intended to represent the fact that the agent *can* perceive the relation, but does not necessarily perceive it.

- spaces
- places or regions
- objects
 - attached to surfaces
 - unattached from surfaces

For representational purposes, we require a neutral concept which can be used to construct geometrically cohesive structures from sensory and other input:

Definition 2 (Entity) *An entity is any thing, object or individual with cohesive structure. One can distinguish between physical entities or mental entities. These will be called external and internal entities, respectively. An entity consists of aspects.*

Gibson's primary concern is with external entities, but we will be concerned with both internal and external entities.

Definition 3 (Aspect) *An aspect is a property or characteristic of an entity, or a relation an entity shares with other entities.*

Aspects are broad in character and will be used to describe the various characteristics of surfaces, spaces, places or regions, and objects. Additionally, aspects can be used to describe the relations between physical and spatial structures.

In the agent system, attribute/value pairs will be used to represent aspects:

Definition 4 (Attribute) *An attribute is a data type representing an aspect of an entity. Attributes have types.*

Definition 5 (Values) *A value is a quantity or quality of an attribute. The type of the value set a value belongs to is the value's type.*

An uncertainty measure can be associated with any attribute/value pair. The type of uncertainty measure used may be probabilistic, fuzzy, rough, etc.

In the agent system, entity frames will be used to represent cohesive structures:

Definition 6 (Entity Frame) *An entity frame is a data structure representing an entity. It consists of a name and a set of attribute/value pairs.*

Values to attributes can refer to other entity names. In this way, complex, recursive entity structures may be constructed. An attribute with an entity name as a value represents a relation between the entity with the attribute and the entity referred to by the attribute.

Attribute/Value pairs (or attribute/value/umeasure triples) and entity frames will be the main building blocks in our knowledge representation.

Entities in the environment have dynamics. Entity and aspect trajectories are introduced to represent such dynamics:

Definition 7 (Entity Trajectory) *An entity trajectory is a data structure representing a set of aspects of an entity through time. It consists of a name, a subset of attribute/value pairs associated with the entity definition, a temporal interval and a sequence or continuum of entity (sub)frames ordered relative to a suitable temporal structure.*

An entity trajectory represents the temporal/spatial progression of an entitie's aspects during a temporal interval. It can be viewed as a bundle of signals or time series.

Definition 8 (Aspect Trajectory) *An aspect trajectory is an entity trajectory with a single attribute.*

An aspect trajectory represents the temporal progression of an single aspect of an entity. It can be viewed as a signal or time series.

Events occur in the environment and they most often represent encapsulated change relative to some part of the environment. There is a similarity with entity trajectories, but an ontological distinction should be made between entity trajectories and events:

Definition 9 (Event) *An event is a collection of aspects and it includes a temporal boundary.*

Definition 10 (Event Frame) *An event frame is a data structure representing an event. It has a name, a set of attribute value pairs and a temporal boundary.*

Event frames are the building blocks for complex events. Event frames may point to parent event frames or subevent frames. Events must have a duration. Aspects within an events boundaries are not required to change value.

One will require a taxonomy of event types where events can refer to their event class membership. For instance, a special case of an event is an event where change within its temporal boundaries is necessary.

Definition 11 (Transition Event) *An transition event is an event where the values of one or more aspects at the start of its temporal boundary are different (greater than some value(s) ϵ_i) at the end of its temporal boundary*

Situations are fundamental to the idea of affordances since an affordance is intended to be a complementarity between an agent and a local temporal/spatial configuration of its surroundings:

Definition 12 (Situation) *A situation is a temporal spatial configuration of entities and events. Situations must have duration.*

Definition 13 (Situation Frame) *A situation frame is a data structure representing a situation. It includes a temporal boundary and often includes a spatial boundary. Temporal and spatial structures constrain the temporal and spatial entities included in the temporal and spatial boundaries, respectively.*

There are various ways to view a situation, all of which should be consistent with the definitions above. For example, frame structures in AI represent prototypical situations that can be instantiated. The use of situations in situation semantics and situation calculus are also strongly related. It should be possible to define temporal structures in terms of simple temporal networks or disjunctive temporal networks. Similarly, spatial structures could be defined using qualitative spatial reasoning techniques.

4.1 Summary

The ontology we require consists of situations. Situations contain entities, events, and relations between each. Situations also contain local temporal structures, local spatial structures and causal dependencies among entities, events, and their actions.

Note that actions are not yet part of the base ontology. This will eventually be added. There are a number of ways to specify actions and it is not yet clear which is best in the context of this project.

Note that affordances are not part of the ontology. This is because affordances are not *in* or *part of* the environment or objects, they are the result of a relation between an agent and its environment while the agent acts in its environment (sometimes using objects). The definition for affordances reflects this important point.

4.2 Operational Remarks

Robotics systems require ontologies just as humans do. How one chooses to slice up the world ontologically determines what reasoning and other mechanisms one has at one's disposal when solving problems or getting around in the world. In some sense, affordances provide one way to slice up the world ontologically that differs from many of the traditional approaches found in the philosophical literature. We believe the affordance perspective should supplement the traditional perspective, not replace it, and vice-versa. This is the stance that will be taken as our representational philosophy.

The basic idea is that the robotic system has functionality which permits it to construct entities at various levels of abstraction. These entities can represent chunks of surfaces of the environment, individuals, objects or aggregates of objects, etc. In addition, entity structures within the robotic system do not have to represent cognized structures, They can equally well be parts of structures associated with sensory processing, etc. In fact, a complex entity structure is intended to include a little of both in order to ensure grounding. A higher level symbolic specification of the robot's capabilities and its relation to such entities will necessarily include symbol references to the entities. Such entities will also be used internally at the behavior and control based levels of the robotic architecture.

5 Classification

One of the main goal achieving functionalities of a robotic system is its ability to acquire necessary resources in its surrounding environment which can be used in the execution of tasks required for the achievement of the robot's goals. An essential component in the architecture has to be the ability to classify the entity structures constructed by the robotic system relative to various form or functional taxonomies. The latter has the character of a high-level interpretation of affordances as properties of the environment (objects/entities) useful for achieving tasks.

We can assume one or more taxonomic hierarchies associated with the domains we will operate our robots in. Two types of taxonomies that will be fundamental to the representation of affordances are *form* taxonomies and *function* taxonomies. The former represent the more traditional notion of taxonomies where one looks for necessary and sufficient conditions for the definition of objects and concepts, the latter is less traditional but strongly related to affordances and the functional use of objects. Functional characteristics slice horizontally across more traditional taxonomies.

For the former we require the construction of entity structures interpreted in the traditional sense such as *chair*, *table*, *block*, etc. In the latter, we require entity structures interpreted relative to functional characteristics associated with capabilities.

Taxonomies should be described in terms of prototypes, either in terms of form, function

or combinations of both. Given an entity structure generated by the robotic system, one will require a similarity measure between the entity structure and the prototype classes.

Given an entity structure and a similarity measure, we should be able to ask the following questions relative to our taxonomies:

Does the entity structure have the functionality required for achieving a particular capability?

Does the entity structure have the characteristics which would allow classification as an object of a particular type?

Given the identification of an entity structure as an object of a certain type, does it have functionality associated with it that would be useful to achieve a certain capability?

Given the identification of an entity structure as one having certain functionalities, can the entity structure be mapped to an object type which would permit acquiring more information useful to the achievement of a capability?

The point to make here is that we often think about the world in terms of both form and function, sometimes in separation and sometimes in combination. Both views should be part of any good theory about affordances.

Of course, Gibson has much to say about these complementary approaches to classification. In his work, he makes a very strong case for classification relative to functional use.

Another important aspect of Gibson's theory of affordances is that "...to perceive an affordance is not to classify an object." ([Gib79], p. 134). Gibson goes on to state that,

The fact that a stone is a missile does not imply that it can not be other things as well. It can be a paperweight, a bookend, a hammer, or a pendulum bob. ...

If you know what can be done with a graspable object, what it can be used for, you can call it whatever you please. ...

The theory of affordances rescues us from the philosophical muddle of assuming fixed classes of objects, each defined by its common features and then given a name. ...

But this does not mean you cannot learn how to use things and perceive their uses. You do not have to classify and label things in order to perceive what they afford. ([Gib79], p. 134)

5.1 Operational Remarks

Entity structure classification mechanisms can be generated in a number of ways. Machine learning techniques in the project can be used to automate the process of learning relationships between an agent and environment relative to a task which require certain affordances. The robotic system then uses these classifiers in the problem solving phase. Automatically learning affordances, or functionality associated with entities for a particular task is obviously a very challenging and difficult task. This being the case, one can begin by "hard wiring" some definitions of affordances which make intuitive sense just to be able to experiment with the use of affordances in a robotic system. This is the strategy taken in this document.

6 Tolerance Spaces and Similarity

In section 5, we saw the need for a means of comparing arbitrarily complex entity structures to each other for similarity. The assumption is that in any robotic system, values associated with attributes in entity structures are approximate due to noise, uncertainty, etc. This implies that when comparing complex entity structures to other entity structures or to entity structure prototypes used for classification purposes we require a generalization of an equivalence relation which is approximate in nature and takes account of the approximate nature of the attribute/value pairs and their aggregation into entity frames. In fact, we believe that this would also be necessary when comparing events and situations. The following idea was presented in Doherty [DLS03] and we believe it is more less what would be required in the affordance context.

6.1 The basic idea

Let us begin with the notion of tolerance and tolerance measures. Webster's dictionary defines tolerance as "the amount of variation allowed from a standard, accuracy, etc."

For example, suppose a system receives data about an attribute a from two sources, where source one asserts that that $a = 1.04$ and source two asserts that $a = 0.98$. Depending on the context, the system might want to consider the values 1.04 and 0.98 as the same relative to some tolerance measure since their distance is only 0.06. In another application this difference may have serious repercussions on system safety, so it is important to make sure that tolerance measures are contextual and can be tuned either automatically or manually relative to the application and context at hand.

The strategy that will be used is to define tolerance and distance measures on the value sets associated with attributes or primitive data domains associated with particular applications. These tolerance and distance measures will be induced through the different levels of data and knowledge abstraction in complex representational structures. The represen-

tational structures will in some sense inherit the tolerance measures from the primitive data domains and value sets used in these structures at lower levels of abstraction and taken into account when comparing for similarity or reasoning. By defining parameterized measures of tolerance via distance measurements on values sets and primitive domains, one can cluster sets of values into tolerance neighborhoods and view the clusters as individual elements. Similarly, individuals whose identities are dependent on sets of attribute/value pairs can also be clustered into tolerance neighborhoods and viewed as indiscernible entities to a particular degree of tolerance when used in other data structures.

The basic primitive in the ideas presented is that of a tolerance function. Let's begin with a value set V and two elements $x, y \in V$. A tolerance function τ provides us with a distance measure between x and y normalized to the real interval $[0, 1]$ where the higher the value, the closer in tolerance the two elements are. Given a parameter $p \in [0, 1]$, a tolerance relation τ^p is then introduced among individuals with a threshold p which tunes the tolerance to be within a certain degree. If $\tau(x, y) \geq p$ then the pair $\langle x, y \rangle$ is in the relation τ^p . Both the tolerance function and the parameter p must be provided by a knowledge engineer or must be machine learned. One can continually refine these values.

Once this is done for individual value sets or primitive data domains, it can be generalized to tuples of values and tolerance can be measured between two tuples $\langle x_1, \dots, x_k \rangle$ and $\langle y_1, \dots, y_k \rangle$ using pairwise comparison of associated tolerance relations.

Given a value set V with associated tolerance measures, we can then take subsets $V_1, V_2 \subseteq V$ and induce tolerance measures and neighborhood functions on the subsets. Likewise, given a set T of k -tuples with associated tolerance measures, we can then take subsets $T_1, T_2 \subseteq T$ and induce tolerance measures and neighborhood functions on the subsets. Subsets of V can be viewed as properties or concepts and subsets of T can be viewed as k -argument relations.

These ideas can be generalized further to sets of sets and sets of sets of tuples, where the tolerance and similarity measures between these structures is induced from the primitive tolerance measures in the base value sets. Once the tolerance and similarity measures are in place, an important structuring generalization can be made where the idea of a *tolerance space* is introduced.

Given a universe U of objects in a tolerance space with the associated tolerance measures, we can provide a generalization of the notions of upper and lower approximations on sets used in rough set theory to subsets of U . The lower and upper approximations will again be induced from the particular tolerance measures provided by the tolerance space in question. Rather than using equivalence classes of individuals constructed from subsets of attributes as in rough set theory, one would work instead with neighborhoods generated from neighborhood functions of individuals.

There is an interesting connection between the idea of tolerance spaces proposed in this chapter and the work of Gärdenfors with conceptual spaces (see, e.g., [Gär00]). Conceptual spaces are built up using multi-dimensional spaces of quality dimensions (attributes)

and providing geometric constraints between these dimensions in order to model distance measures and similarity. However, we use the notion of semi-distances rather than of distances. Tolerance spaces contribute to a generalization of conceptual spaces in the sense that concepts can be generalized to approximate concepts based on tolerance measures and the geometric constraints used are less rigid than with conceptual spaces. In order to place tolerance spaces in the proper context with conceptual spaces, we define a simple version of conceptual spaces and show how tolerance spaces may be integrated in this framework.

6.2 A Formal Framework

In the following, we provide the formal framework for the idea.

6.2.1 Conceptual Spaces

A *semi-metric space* is a pair $\langle A, \delta \rangle$, where A is a set and δ is a function

$$\delta : A \times A \longrightarrow \mathcal{R}$$

which, for all $x, y \in A$, satisfies:

$$\delta(x, y) \geq 0, \delta(x, x) = 0 \text{ and } \delta(x, y) = \delta(y, x).$$

Any function δ satisfying the above properties is called a *semi-metric* for A and $\delta(x, y)$ is called the *semi-distance* between x and y .

Definition 14 *Let U be a finite nonempty set of objects. By a quality dimension over U we understand any semi-metric space $\langle U, \delta \rangle$. By a conceptual space over U we mean any pair $\langle U, Q \rangle$, where Q is a finite set of quality dimensions over U . ■*

Quality dimensions usually correspond to attributes of objects together with a semi-distance defined on the attributes value domains. For example, if one measures colors of objects, quality dimensions can correspond to hue, chromaticity and brightness. The concept “fruit” may have dimensions corresponding to weight, taste, color, etc.

Usually, with any quality dimension one associates a relational structure representing a domain of values corresponding to the quality dimension, together with functions and relations allowing one to calculate (semi-)distances.

For instance, with the quality dimension “weight” one can associate a relational structure defining arithmetic on the real numbers.

6.2.2 Tolerance and Inclusion Functions

We begin by defining a tolerance function on individuals. From this a parameterized tolerance relation follows naturally.

Definition 15 *By a tolerance function on a set U we mean any function $\tau : U \times U \rightarrow [0, 1]$ such that for all $x, y \in U$,*

$$\tau(x, x) = 1 \quad \text{and} \quad \tau(x, y) = \tau(y, x). \quad \blacksquare$$

Given a conceptual space $\langle U, Q \rangle$ and a quality dimension $\langle U, \delta \rangle \in Q$, a *tolerance function* τ , based on the quality dimension can be defined as follows:

$$\tau(u, u') \stackrel{\text{def}}{=} 1 - \frac{\delta(u, u')}{\max\{\delta(x, y) : x, y \in U\}}. \quad (6)$$

Of course, the same approach could be used for an attribute a and its value set V_a in a complex knowledge structure, provided δ is given, without appeal to conceptual spaces.

Definition 16 *For $p \in [0, 1]$ by a tolerance relation to a degree at least p based on τ , we mean the relation τ^p given by*

$$\tau^p \stackrel{\text{def}}{=} \{\langle x, y \rangle \mid \tau(x, y) \geq p\}.$$

The relation τ^p is also called the parameterized tolerance relation. ■

In the following, $\tau^p(x, y)$ is used to denote the characteristic function for the relation τ^p .

Intuitively, $\tau(x, y)$ provides a degree of similarity between x and y , whereas $\tau^p(x, y)$ states that the degree of similarity between x and y is at least p . In what follows we limit ourselves to tolerance relations where it is assumed that the parameter p has been provided and is tuned to fit particular applications.

Often one considers objects to be similar if a given distance between them is not greater than a given threshold, say d . Given a quality dimension $\langle U, \delta \rangle$ and a threshold $d \geq 0$, one can define the parameter p from Definition 16 to be

$$p \stackrel{\text{def}}{=} 1 - \frac{d}{\max\{\delta(x, y) : x, y \in U\}}. \quad (7)$$

A parameterized tolerance relation is used to construct tolerance neighborhoods for individuals.

Definition 17 *By a neighborhood function wrt τ^p we mean a function given by*

$$n^{\tau^p}(u) \stackrel{\text{def}}{=} \{u' \in U \mid \tau^p(u, u') \text{ holds}\}.$$

By a neighborhood of u wrt τ^p we mean the value $n^{\tau^p}(u)$. ■

6.2.3 Tolerance Spaces

The concept of tolerance spaces plays a fundamental rôle in this approach.

Definition 18 A tolerance space is defined as the tuple $TS = \langle U, \tau, p \rangle$, which consists of

- a nonempty set U , called the domain of TS ;
- a tolerance function τ
- a tolerance parameter $p \in [0, 1]$.

The parameterized tolerance relation τ^p is defined as in Definition 16. ■

Given a universe U of individuals, a set of attributes A and a set $X \subseteq U$, one often considers the lower and upper approximation of X as defined in terms of a partitioning of the universe U in indiscernibility classes relative to a subset of the attributes A . Given a tolerance space $TS = \langle U, \tau, p \rangle$, rather than considering an individual's indiscernibility class as a basis for defining the lower and upper approximation of $X \subseteq U$, we can instead use the neighborhood of an individual induced by the tolerance function/parameter pair(s) provided by the tolerance space. In addition, we can tune our definition of upper approximation via a parameter q which determines how much of a neighborhood must be part of X in order for it to be included in the upper approximation.²

Below, for any set X , by $|X|$ we mean the cardinality of X .

Definition 19 Let $U_1, U_2 \subseteq U$. By the standard inclusion function we mean the function given by

$$\mu(U_1, U_2) \stackrel{\text{def}}{=} \begin{cases} \frac{|U_1 \cap U_2|}{|U_1|} & \text{if } U_1 \neq \emptyset \\ 1 & \text{otherwise.} \end{cases}$$

Let $TS = \langle U, \tau, p \rangle$ be a tolerance space and $X \subseteq U$. The lower and upper approximations of X wrt TS to a degree $q \in [0, 1]$, $X_{TS^+}^q$ and $X_{TS^\oplus}^q$, are defined by

$$X_{TS^+}^q = \{u \in U : \mu(n^{\tau^p}(u), X) = 1\}, X_{TS^\oplus}^q = \{u \in U : \mu(n^{\tau^p}(u), X) > q\}.$$

The approximations $X_{TS^+}^0, X_{TS^\oplus}^0$ are called the lower and upper approximations of X wrt TS and are often denoted by $\underline{X}_{TS^+}, \overline{X}_{TS^\oplus}$, respectively. ■

²A different approach, based on a notion of approximation spaces, object neighborhoods and rough inclusion, has been introduced in [SS96].

6.3 Defining Tolerance on Complex Representational Structures

In this section we show how to induce a tolerance relation on complex structures on the basis of a tolerance relation defined on domain elements.

Consider a tolerance space $TS = \langle U, \tau, p \rangle$. First, we would like to extend the tolerance and neighborhood functions induced by TS to deal with subsets of U . We shall need a notion of generalized inclusion function ν^{τ^p} which will be used as a basis for measuring similarity between complex information structures.

One of the important motivations behind the definition provided is that we require a generalized inclusion function to coincide with the standard inclusion function in the case of a trivial tolerance space (identifying equal elements and distinguishing elements that are not equal).³

Definition 20 Let U be a set and $U_1, U_2 \subseteq U$. By the generalized inclusion function induced by τ^p we mean the function given by

$$\nu^{\tau^p}(U_1, U_2) \stackrel{\text{def}}{=} \begin{cases} \frac{|\{u_1 \in U_1 : \exists u_2 \in U_2 [u_1 \in n^{\tau^p}(u_2)]\}|}{|U_1|} & \text{if } U_1 \neq \emptyset \\ 1 & \text{otherwise.} \end{cases}$$

For $q \in [0, 1]$, we say that U_1 is included in U_2 to a degree at least q wrt ν^{τ^p} iff $\nu^{\tau^p}(U_1, U_2) \geq q$.

In the case of tuples⁴ $U_1 = \langle u_1, \dots, u_n \rangle$ and $U_2 = \langle u'_1, \dots, u'_n \rangle$, by the generalized inclusion function over tuples, induced by τ^p we mean the function given by

$$\nu_o^{\tau^p}(U_1, U_2) \stackrel{\text{def}}{=} \begin{cases} \frac{|\{u_i : 1 \leq i \leq n \text{ and } u_i \in n^{\tau^p}(u'_i)\}|}{|U_1|} & \text{if } n \neq 0 \\ 1 & \text{otherwise.} \end{cases} \quad \blacksquare$$

In the sequel we write $\nu_{TS}^{\tau^p}$ and $n_{TS}^{\tau^p}$, respectively, to denote ν^{τ^p} and n^{τ^p} , where τ^p is a tolerance relation induced from a tolerance space TS .⁵

Definition 21 Let $TS = \langle U, \tau, p \rangle$ be a tolerance space. By a power tolerance space induced by TS we mean $T^{TS} = \langle U^{TS}, \tau^{TS}, s \rangle$, where

- $U^{TS} \stackrel{\text{def}}{=} \text{Pow}(U)$, is the set of all subsets of U

³We also require such ‘‘continuity’’ in other definitions. Namely, the trivial tolerance space should always lead to standard notions that are accepted when tolerance is not considered.

⁴I.e., ordered sets of the same cardinality.

⁵We often drop the superscripts and subscripts when the tolerance spaces and relations are known from context.

- for $U_1, U_2 \in U^{TS}$, $\tau^{TS}(U_1, U_2) \stackrel{\text{def}}{=} \min \{ \nu^{\tau^p}(U_1, U_2), \nu^{\tau^p}(U_2, U_1) \}$
- $s \in [0, 1]$ is a tolerance parameter. ■

We define tolerance and neighborhood functions on tuples of elements in a similar manner.

Definition 22 Let $TS = \langle U, \tau, p \rangle$ be a tolerance space. By a k -tuple tolerance space induced by TS we mean $T^{TS^k} = \langle U^{TS^k}, \tau^{TS^k}, s \rangle$, where

- $U^{TS^k} \stackrel{\text{def}}{=} \underbrace{U \times \dots \times U}_{k\text{-times}}$, is the set of all k -tuples of U
- for $U_1, U_2 \in U^{TS^k}$, $\tau^{TS^k}(U_1, U_2) \stackrel{\text{def}}{=} \nu_o^{\tau^p}(U_1, U_2) = \nu_o^{\tau^p}(U_2, U_1)$,⁶
- $s \in [0, 1]$ is a tolerance parameter⁷. ■

Let us summarize the methodology we propose:

- we start with a quantitative representation of the similarity of considered concepts given by semi-distance or tolerance functions (see Definitions 14 and 15)
- the definition of tolerance spaces (Definition 18) and neighborhoods (Definition 17) allows us to transform the quantitative representation of the similarity into a qualitative representation of the concepts. Such a transformation can also be applied to complex representational structures using Definitions 21 and 22. Tolerance parameters allow us to tune the similarities to fit particular application domains
- the approximations provided in Definition 19 allow us to isolate objects that surely satisfy a given property and that might satisfy the property. In consequence, we also obtain a characterization of objects that surely do not satisfy the property
- finally one can apply various deduction mechanisms to reason about the considered concepts (see, e.g., [DLSS03]).

7 Formalizing Actions

The strategy we plan on using here is to base this work on TAL (Temporal Action Logics) [DGKK98]. What is lacking in this formalization is the use of situations as part of the ontology. Although we consider situations briefly in the next section, what would be

⁶The equality between $\nu_o^{\tau^p}(U_1, U_2)$ and $\nu_o^{\tau^p}(U_2, U_1)$ follows from the symmetry of τ .

⁷The tolerance parameter s specified in definitions 21 and 22 is not used in this paper.

required is an integration of situations with TAL. This work is in progress, quite challenging and quite difficult to do. If it can be achieved, the formalism fits in quite nicely with the ontological commitments we believe have to be made and with the pragmatic robotics architecture itself.

8 Formalizing situations

Let's take a closer look at the relation between an agent and its environment. In the previous section, we intended to use "environment" in the sense of the multitude of situations which make up an agent's ecological niche or in even broader terms, its reality. A situation is a set of properties that individuals, objects or entities have and the relations they stand in at various spatiotemporal locations. This has both the flavor of the Situation Calculus and of Situation Semantics. Events provide the temporal and causal structures which organize individuals or entities in situations. A situation is intended to have very rich structure with temporal duration.

Let's assume the existence of situations and some notion of situational structures yet to be defined. We require a predicate to assert aspects of situations. Let's assume the use of a Holds predicate that we will overload with different argument types. The following are some assertions of aspects of situations:

$$\text{Holds}(r(x, y), t, l, s) : r(x, y) \text{ holds at time } t \text{ at location } l \text{ in situation } s \quad (8)$$

$$\text{Holds}(\text{adjacent}(l, l'), t, s) : \text{location } l \text{ is adjacent to } l' \text{ at time } t \text{ in situation } s \quad (9)$$

$$\text{Holds}(\text{before}(t, t'), s) : \text{time } t \text{ is before } t' \text{ in situation } s \quad (10)$$

$$\text{Holds}(p(y), t, s) : p(y) \text{ holds at time } t \text{ in situation } s \quad (11)$$

$$\text{Holds}(p(y), l, s) : p(y) \text{ holds at location } l \text{ in situation } s \quad (12)$$

$$\text{Holds}(\text{exists}(x), t, l, s) : x \text{ is at location } l \text{ at time } t \text{ in situation } s \quad (13)$$

$$\text{Holds}(\text{occurs}(e), t, t', l, s) : \text{event } e \text{ occurs at location } l \text{ from time } t \text{ to } t' \text{ in situation } s \quad (14)$$

(15)

Observe that the overloading, syntax and meaning for this predicate has to be defined more strictly, but for the time being, this description should suffice.

8.1 A formal framework

In progress. The basic problem is finding a consistent formal specification of operations on situations such as union and intersection.

Let

- $\langle \text{DOM}_t, \leq_t \rangle$ be a structure of time points linearly ordered by \leq_t

- DOM_l be a domain of spatial locations
- DOM_s be a domain of situations.

We assume that predicate $holds(s, t, l, \bar{a})$ is given, where $s \in \text{DOM}_s$, $t \in \text{DOM}_t$ and $l \in \text{DOM}_l$. We do not care here what are its other arguments \bar{a} .

For $s \in \text{DOM}_s$, we define

- $\text{Inf}^+(s) \stackrel{\text{def}}{=} \{ \langle t, l, \bar{a} \rangle \mid holds(s, t, l, \bar{a}) = \text{TRUE} \}$
- $\text{Inf}^-(s) \stackrel{\text{def}}{=} \{ \langle t, l, \bar{a} \rangle \mid holds(s, t, l, \bar{a}) = \text{FALSE} \}$
- $\text{Tim}^+(s) \stackrel{\text{def}}{=} \{ t \mid \exists l, \bar{a}. holds(s, t, l, \bar{a}) = \text{TRUE} \}$
- $\text{Tim}^-(s) \stackrel{\text{def}}{=} \{ t \mid \exists l, \bar{a}. holds(s, t, l, \bar{a}) = \text{FALSE} \}$
- $\text{Loc}^+(s) \stackrel{\text{def}}{=} \{ l \mid \exists t, \bar{a}. holds(s, t, l, \bar{a}) = \text{TRUE} \}$
- $\text{Loc}^-(s) \stackrel{\text{def}}{=} \{ l \mid \exists t, \bar{a}. holds(s, t, l, \bar{a}) = \text{FALSE} \}$.

Definition 23

- By a situation knowledge ordering we understand the structure

$$\mathcal{K} \stackrel{\text{def}}{=} \langle \text{DOM}_s, \leq_K \rangle,$$

where $s \leq_K s' \stackrel{\text{def}}{=} \text{Inf}^+(s) \subseteq \text{Inf}^+(s') \wedge \text{Inf}^-(s) \subseteq \text{Inf}^-(s')$.

- By a situation temporal ordering we understand the structure

$$\mathcal{T} \stackrel{\text{def}}{=} \langle \text{DOM}_s, \leq_T \rangle,$$

where $s \leq_T s' \stackrel{\text{def}}{=} \text{Tim}^+(s) \subseteq \text{Tim}^+(s') \wedge \text{Tim}^-(s) \subseteq \text{Tim}^-(s')$.

- By a situation spatial ordering we understand the structure

$$\mathcal{L} \stackrel{\text{def}}{=} \langle \text{DOM}_s, \leq_L \rangle,$$

where $s \leq_L s' \stackrel{\text{def}}{=} \text{Loc}^+(s) \subseteq \text{Loc}^+(s') \wedge \text{Loc}^-(s) \subseteq \text{Loc}^-(s')$. ■

Proposition 8.1 Knowledge ordering is stronger than the temporal and spatial orderings, i.e.,

$$\begin{aligned} \forall s, s'. [s \leq_K s' \rightarrow s \leq_T s'] \\ \forall s, s'. [s \leq_K s' \rightarrow s \leq_L s']. \end{aligned}$$

◁

Then one can add intensional rules completing information about situations, e.g.

$$\exists t, t'. [t, t' \in \text{Tim}(s)] \rightarrow \forall t [t \leq_t t'' \leq_t t' \rightarrow t'' \in \text{Tim}(s)].$$

9 A Scenario

The LiU-IDA project node has set up a relatively challenging robotics environment for initial experimentation related to the representational aspects of affordances in robotics platforms. Although the project strategy is to use a common robotic platform for experimentation, there is a great deal of preliminary experimentation that can be done on separate platforms to speed up development of the common prototype system. In fact, this is a necessity of sorts since one must have a robotics platform to acquire hands-on experience with affordance related issues.

9.0.1 The LiU-IDA Robotic Platform

The LiU-IDA node has an ActivMedia Pioneer P3-AT outdoor robot with a 2 degree of freedom gripper. The platform also includes a Sick LMS-200 line-scanning laser range finder. It is mounted on a tilt mechanism we developed on our own specifically for this robot in order to make the laser useful for the project. For the experiments described here the horizontal scanning range is 100 degrees with 1 degree of resolution and the vertical range 80 degrees starting at 0 (forward) and going down to -80deg with 0.225 degree resolution. A Sony EVI-D31 pan-tilt analog color camera is also included in the platform. Both the camera and the laser sensor can see the gripper.

The hardware architecture is based on a PC104 embedded computer stack consisting of a 700MHz PIII CPU board with 256MB RAM and a 40GB harddrive, an analog frame grabber BT678, an 8xRS232 extension board (digital outputs/inputs used) and a 4xRS232 extension board modified to work with the scanner's 500Kbaud RS422 serial link. A Honeywell HMR3000 compass module is used for measuring attitude angles of the robot. Additionally, a wireless Ethernet 802.11b adapter is used. A frontal view of the LiU-IDA robotic platform is shown in figure 1.

9.0.2 The LiU-IDA Experimental Environment

The experimental environment is shown in figure 2. It currently includes two elevated platforms, a bridge constructed from plywood and a ramp. Each of these structures is portable and can be arranged in various ways. In addition, a blue bucket is present in the environment.

For experimentation with base structural affordances such as standability, turnability and traversability, and for entity-coupled affordances such as reachability, graspability and liftability, we experiment with goal-directed scenarios in the environment which require finding unattached geometric entities such as blue buckets and traversing to their location. In order to do this, the robot must continually construct a 3D model of its environment and generate affordance maps from the 3D map and other affordance maps, where each affor-



Figure 1: LiU-IDA Robot Platform

dance map represents different types of structural and entity-coupled affordances. These representations are then used in the generation of plans to traverse to such entities. To complicate matters, attached and unattached geometrical entities and structural surfaces may prevent the robot from traversing to the goal entities. In this case, the robot may try and remove or place various entities at other locations than those they currently occupy. In this manner, the robotic system can acquire or reacquire the appropriate affordances necessary for behaviors to achieve goals.

Part of the 3D model generated by our robot at the beginning of the mission shown in figure 2 is depicted in figure 3. Black regions represent the lowest elevations while white regions represent the highest elevations. One can discern the bucket, ramp and the bridge quite clearly.

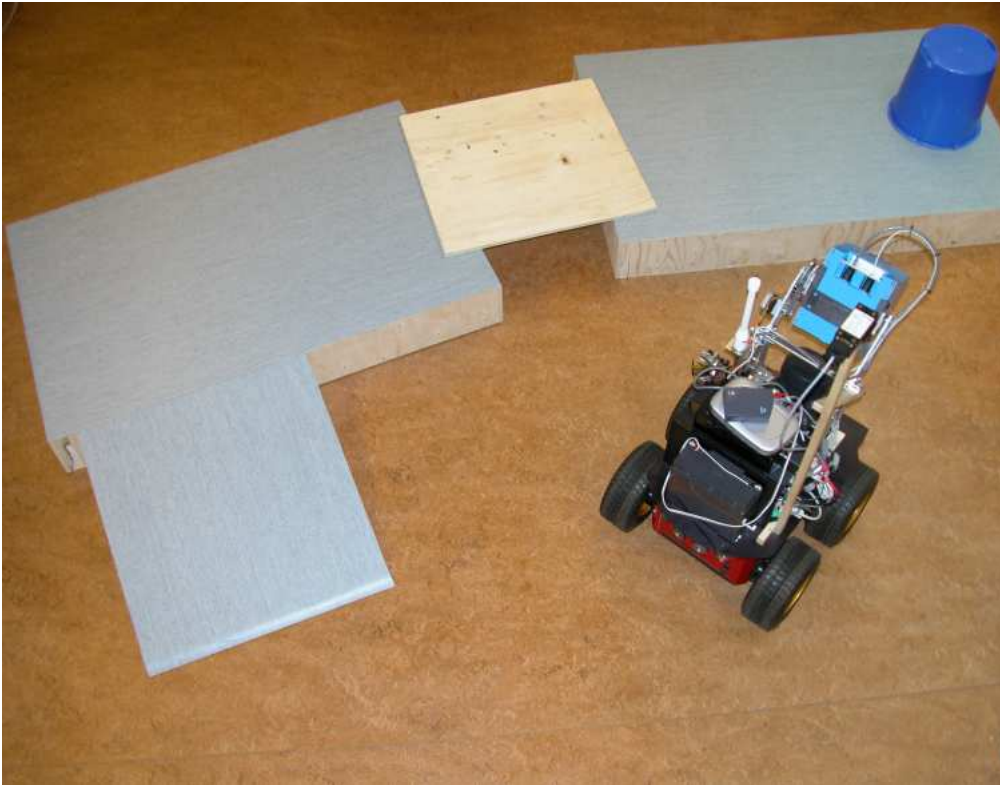


Figure 2: LiU-IDA Experimental Environment

10 Some Initial affordances

In this section, we will describe a number of affordances associated with the traversal of our robot in its ecological niche. There will be many simplifications in the definitions of the affordances used, but this is a first iteration and is only intended to close the conceptual loop. The primary goal then is to use these affordances as the ontological basis for an affordance based motion planner described in section 11 .

The affordances we will focus on are the following:

- Standability
- traversability
- turnability
- reachability
- Graspability
- Liftability

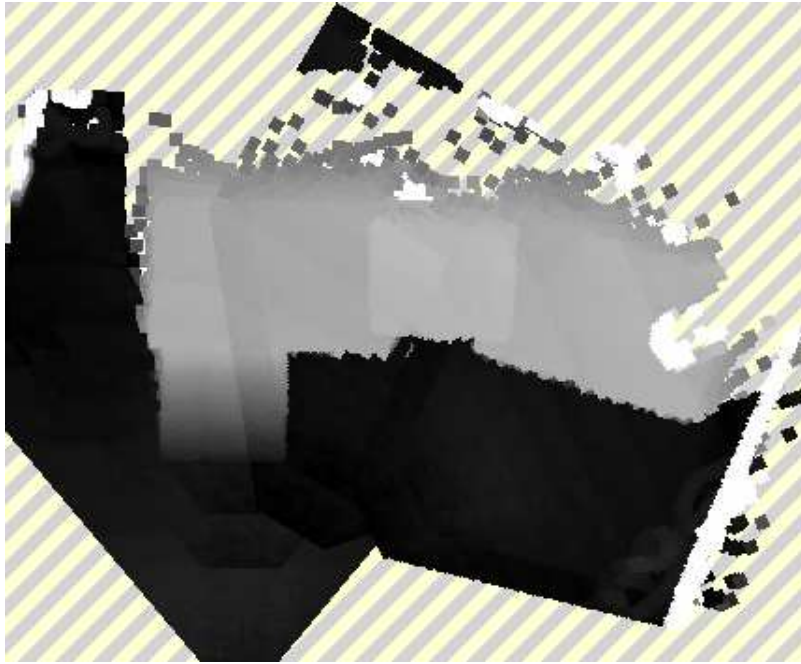


Figure 3: Elevation Map of the Environment

The first three are structural affordances while the latter three are entity-coupled affordances. Most of the detailed work has been done with the structural affordances and more detailed work with the entity-coupled affordances is in progress.

Keep in mind that the logical characterization requires much work before we can say that we have a logic for affordances with an inference mechanism that can be used on the robotic system or simply for reasons of specification. To our knowledge, integrating the concept of affordances with a logic for action and change has not yet been achieved and we consider what we are doing here a first step in that direction.

In each of the following sections, we will provide definitions of what each affordance means relative to our Active Media robot with its particular configuration. We will then try to relate the definition to the logical specification previously presented in section 3 and for some of the affordances we will provide iconic representations used in the motion planner. We call these iconic representations *affordance maps*. These will also have an important part to play in the our proposal for a support architecture for affordances.

10.1 Standability

An environment affords standability (being standable) for an agent at a location l if the following holds:

1. There is a situation s which the agent agt is in and a location l in s where the agent is at
2. There is a location l' and two regions r and r' which are part of s
3. $r \subseteq r'$ and $l \in r$ and l is at the center of r and r'
4. The size of r is the same as the size of the lowerbody of agt and the size of r' is the same as the size of the upperbody of agt
5. For any location $l' \in r$, the slope of the tangent plane at l' relative to the elevation map perceived by agt is less than a parameter k
6. For any location $l' \in r'$, there is no positive obstacle in the elevation map perceived by agt higher than a parameter k'

For a region perceivable by the agent, a primary affordance map for standability can be generated which encodes these constraints implicitly. The agent can use this iconic map to *directly* perceive the affordance of standability in its local region. If it moves around and has storage capabilities, it can store affordance maps for larger regions outside its purview. The danger here of course is that the secondary affordance maps for standability may not be accurate when used after their construction. From a computational perspective, storage of secondary affordance maps may be necessary. Conceptually, the agent should be generating primary affordance maps at relatively high sampling rates.

Suppose an agent has the goal of standing at a location. A prerequisite to standing at the location is that the situation the agent is presently in offers the capability to stand at the location or the agent has the ability to create a situation which offers that capability. For example, if there are objects at the location, the agent would have to move to the vicinity and remove the objects. If the agent simply was not at the location, it would have to traverse to the location. Let's begin with the simple situation where the agent can directly perceive that the location is standable.

Getting back to our logical characterization, the predicates `CanDo`, `CanAchieve` and `CanPerceive` require additional explication.

`CanDo` is used primarily to assert the presence of the standard preconditions associated with an action. Since this is intended to be asserted before execution of the action, one should not refer to any timepoints in the future.

$$\text{CanDo}(agt, \text{stand}(agt, l), env) \tag{16}$$

We can assume that there are a number of preconditions associated with the action `stand` related to balance, etc., that are encapsulated using `preconds(move(agt, l))`. The following appear to be sufficient conditions for the agent to be able to execute the action `stand(agt, l)`:

1. There is a situation s in env where
 - (a) There is a timepoint t and a location l in env where the preconditions to `stand(agt, l)` hold

In logic,

$$\exists s, t, l. s \sqsubseteq env \wedge t, l \in s \wedge \text{Holds}(\text{preconds}(\text{move}(\text{agt}, l)), t, l) \quad (17)$$

Essentially, the only condition required is that when and where one chooses to execute an action, its preconditions hold. One could also think of introducing resource issues into the specification, but for the time being, we will separate scheduling and resource allocation issues from action execution.

In order to be able to achieve the state of standing using the action `stand`, one would require the presence of the affordance of standability. `CanAchieve` is used to represent this constraint.

$$\text{CanAchieve}(\text{agt}, \text{stand}(\text{agt}, l), \text{standing}(\text{agt}, l), env) \quad (18)$$

The following appear to be sufficient conditions for the agent to be able to achieve the state of standing at a location:

1. There is a situation s in env where
 - (a) There is a timepoint t and a location l in env where the preconditions to `stand(agt, l)` hold
 - (b) The effects of the action `stand(agt, l)` include `at(l)`
 - (c) The affordance `standable(agt, l)` holds at location l in s

In logic,

$$\begin{aligned} \exists s, t, l. s \sqsubseteq env \wedge t, l \in s \wedge \text{Holds}(\text{preconds}(\text{stand}(\text{agt}, l)), t, l, s) \wedge \\ \text{at}(\text{agt}, l) \in \text{effects}(\text{stand}(\text{agt}, l)) \wedge \text{Holds}(\text{standable}(\text{agt}, l), t, l, s) \end{aligned} \quad (19)$$

Sufficient conditions for `Holds(standable(agt, l), t, l, s)` are listed as 1-6 in section 10.1.

`Holds(standable(agt), l, t, s)` is essentially a triggering affordance *loosely* associated with the action `stand(agt, l)` where the agent takes advantage of the fact that the affordance is present in the environment in order to achieve a goal (that of standing at a location)

associated with the affordance and action together. If the action `stand` has duration, then `standable(agt, l)` would be required as a durative affordance throughout the execution of the action.

Note that affordances are loosely coupled with actions since other actions may achieve the same result or the same action may achieve results that have nothing to do with this particular affordance. The idea would be that the features associated with pre- and postconditions to actions are separated from the features associated with affordances. For example, `at(l)` is a feature associated with an action specification while `standing(agt, l)` is a feature associated with the presence of an affordance in the context of an action. One would require an additional construct to relate these different feature classes if so desired. We will use *uniformity constraints* for this purpose.

A uniformity constraint relates the effects of an action with the affordances present. For instance, in this example:

$$\forall l, t, s. \text{Holds}(\text{at}(l), t, l, s) \wedge \text{Holds}(\text{standable}(agt, l), t, l, s) \rightarrow \text{Holds}(\text{standing}(agt, l), t, l, s) \quad (20)$$

The intention here is to keep the affordance specifications separate from the standard action theory. This is one of the reasons both `CanDo` and `CanAchieve` are introduced. `CanDo` refers to features specific to an action specification while `CanAchieve` refers to both type of features. In fact, it would generally be the case that

$$\text{CanAchieve}(agt, \text{stand}(agt, l), \text{standing}(agt, l), env) \rightarrow \text{CanDo}(agt, \text{stand}(agt, l), env), \quad (21)$$

But for now, it is useful to keep these assertions distinct.

Uniformity constraints are also useful as a means of specifying affordance monitoring constraints where the idea is that as an action is executed, not only are its action-based features monitored so their behavior follows the intent of the action specification, but any affordances associated with a particular action-affordance-affordance-feature triple are also monitored for their presence or absence. For example,

$$\forall l, t, t', t'', s. t \leq t'' \leq t' \wedge \text{Occurs}(\text{stand}(agt, l), t, t', s) \wedge \text{Holds}(\text{standable}(agt, l), t'', l, s) \rightarrow \quad (22) \\ \text{Holds}(\text{standing}(agt, l), t'', l, s)$$

Regarding the ability for the agent to perceive the affordance associated with standing, we use the predicate `CanPerceive`,

$$\text{CanPerceive}(agt, \text{affords}(agt, \text{standing}(agt, l), env), env) \quad (23)$$

Stating the sufficient conditions for this predicate are somewhat problematic. In some sense, the idea is that the agent is attuned to its environment in such a manner where its size, shape and sensory capabilities suffice for it to be in a position where it can directly perceive `l` and its surrounding region `r` and verify more or less directly that `Holds(standable(agt), t, l, s)`. The 6 conditions listed in section 10.1 in some sense can be parameterized differently to different robotic systems. One parameterization will work well

on a particular robotic system while on another it will not because that robots physical and sensory capabilities do not match the environment to the extent that the proper relation between the robot and its environment can not occur. In other words, there are no situations meeting the constraints necessary to verify 6 conditions above.

Putting the above arguments together, we see how one can verify at a specificational level that an agent has a particular affordance with its environment relative to a capability. In this case,

$$\text{Affords}(\text{agt}, \text{standing}(\text{agt}, l), \text{env}) \quad (24)$$

Even though asserting such affordances in a logical language and reasoning about them lies far from the intuitions associated with direct perception, the use of such specifications may prove to be beneficial at the knowledge level of a robotic system and definitely useful as a specification for constructing robotic systems which use affordances.

Some observations:

- Note that there is an underlying action theory that may be separated from the use of affordances. Affordances in some sense are an add-on feature to an existing theory of actions. This is a good feature of the theory because in some sense, the agent might not be at all conscious of the underlying affordance constraints in the environment when executing actions. They are separable in some sense from what is being reasoned about or cognized. The affordance infrastructure is operational at the reactive and control levels of the robotic system.
- $\text{Holds}(\text{standable}(\text{agt}, l), t, l, s)$ is intended to be verified as true via the underlying perceptual system associated with the robotic system. The 6 conditions listed in section 10.1, are easily checked using the standability affordance maps continually generated by the robotic system. This constraint may be directly coupled to a behavior in the traditional behavior based paradigm or be used as a constraint at higher conceptual levels in the action theory of the robot. In a sense, this syntactic assertion represents a reification of the direct perceptual mechanisms used by the robotic system to perceive its surrounding environment.

10.1.1 Standability Affordance Map

The standability affordance map shown in figure 4 is generated as the robot scans its environment with its laser scanner. The degree of standability is color coded relative to orientation of the robot since standability is dependent on the orientation the robot has relative to a reference direction.⁸ The color coding is based on the use of a 24 bit sequence (one byte each for red, blue, green) where the color represents the number of 18 orientations

⁸For those readers without a color version of this document, the ensuing description may make little sense.

that a robot can be in at a location and satisfy the standability constraints. For example, dark colors represent locations where the robot can only stand in few orientations, while the color yellow represents locations where the robot can stand in all 18 orientations. In fact, one can view this affordance map as an aggregation of 18 different affordance maps with 10 degree change in orientation (symmetry is assumed for the other 18 positions in a 360 degree sweep). So each affordance map would represent standable locations at one angle of orientation. Note that on the ramp, the robot can only be in one orientation whereas in the middle of the platforms it has standability at any orientation.

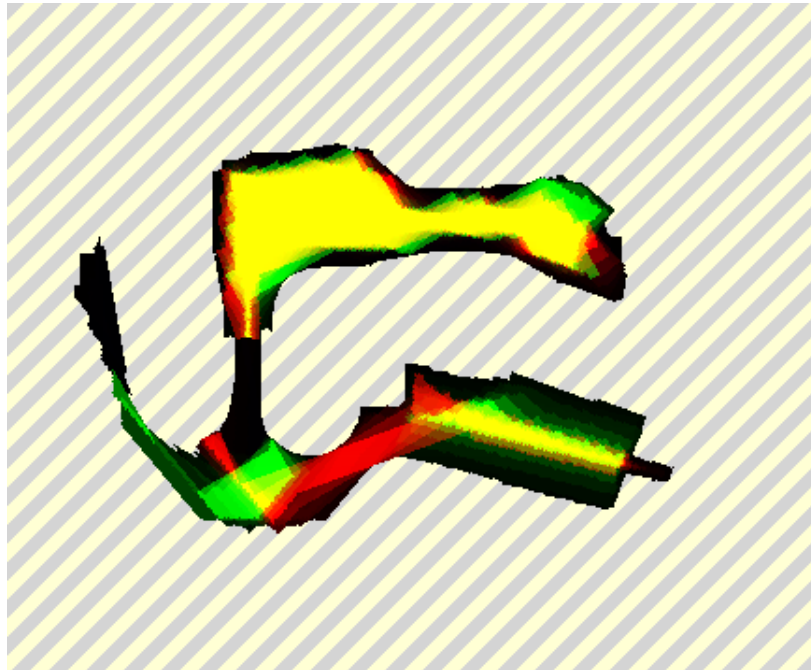


Figure 4: Affordance Map for Standability

10.2 Turnability

10.2.1 Turnability Affordance Map

In the system, locations where the robot can do full turns are represented as turnability affordance maps. Such maps can be generated from the standability affordance maps described previously. Figure 5 shows the turnability affordance map which is generated from Figure 4. Locations in white represent full turnability in any direction to any degree, while black regions represent lack of full turnability. We have used turnability affordance maps in order to simplify generation of legal motion plans.

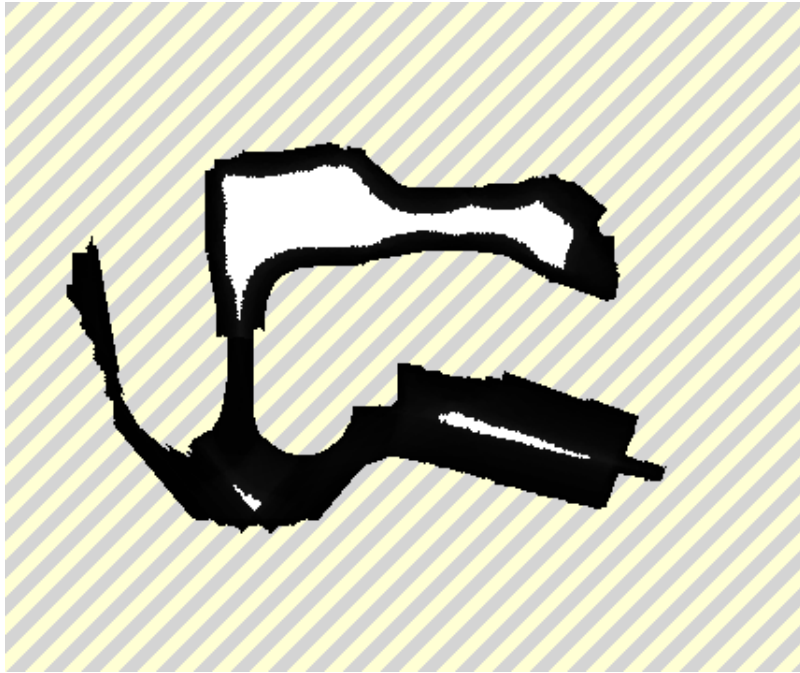


Figure 5: Affordance Map for Turnability

10.3 traversability

An environment affords traversability (being traversible) for an agent at a location l relative to a location l' in a region r if the following holds:

1. There is a situation s which the agent agt is in at a timepoint t and a region r in s which contains two locations l, l' , where the agent is at (l) and where it is going (l').
2. There is a subregion r' of r viewable by the agent which contains l and l' and for each l'' in r'
 - (a) the affordance of standability is present
 - (b) the affordance of turnability is present is present at points where direction changes.

Note that the affordance of traversability is a higher-order affordance in the sense that it is dependent on standability. For the region r' perceivable by the agent, a primary affordance map for traversability which encodes the constraints above implicitly can be generated using the primary affordance map for standability. This affordance map may then be used by an affordance-based motion planner to generate a specific path between locations l and l' . During the traversal of this path, the region has to be continually monitored to ensure that the traversability affordances are present as the agent moves in

the region. In other words, the following constraint must hold, for any two locations l, l' in the path generated by the motion planner, when the agent is in motion,

$$\text{Holds}(agt, \text{traversable}(agt, l, l', r), l, t, s) \quad (25)$$

One can of course relax this constraint and only monitor a subpath directly in front of the agent.

Let's specify the following affordance assertion for traversability relative to an action traverse,

$$\text{Affords}(agt, \text{traversable}(agt, l, l', r), env) \quad (26)$$

We assume the agent has an action $\text{traverseto}(agt, l, l', r)$ in its repertoire and begin with the following,

$$\text{CanDo}(agt, \text{traverseto}(agt, l, l', r), env) \quad (27)$$

The following appear to be sufficient conditions for the agent to be able to execute the action $\text{traverseto}(agt, l, l', r)$:

1. There is a situation s in env where
 - (a) There is a timepoint t , a region r , and location l, l' in s where the preconditions to $\text{traverseto}(agt, l, l', r)$ hold
 - (b) The agent agt is at location l during timepoint t .

In logic,

$$\begin{aligned} & \exists s, t, l, l', r. s \subseteq env \wedge t, l, l', r \in s \wedge l, l' \in r \wedge \\ & \text{Holds}(\text{at}(agt), t, l, s) \wedge \text{Holds}(\text{preconds}(\text{traverseto}(agt, l, l', r)), t, l, s) \end{aligned} \quad (28)$$

The agent has the ability to traverse between two locations if there is an action the robot can use to get from one location to another and the required affordances are present,

$$\text{CanAchieve}(agt, \text{traverseto}(agt, l, l', r), \text{traversing}(agt, l, l', r), env) \quad (29)$$

The following appear to be sufficient conditions for achieving traversability between two locations using the action traverse:

1. There is a situation s in env where
 - (a) There is a timepoint t , a region r , and locations l, l' in s where the preconditions to $\text{traverseto}(agt, l, l', r)$ hold
 - (b) The effects of the action $\text{traverseto}(agt, l, l', r)$ include being at $\text{at}(agt, l')$

(c) The affordance $\text{traversable}(agt, l, l', r)$ holds.

In logic,

$$\exists s, t, l, l', r. s \sqsubseteq \text{env} \wedge t, l, l', r \in s \wedge l, l' \in r \wedge \text{Holds}(\text{preconds}(\text{traverseto}(agt, l, l', r)), t, l, s) \wedge \text{at}(agt, l') \in \text{effects}(\text{traverseto}(agt, l, l', r)) \wedge \text{Holds}(\text{traversable}(agt, l, l', r), l, t, s) \quad (30)$$

We also introduce a uniformity constraint relating the action traverseto to the ability traversing via an affordance traversable :

$$\forall t, t', t'', l, l', l_1, l_2, r, s. t \leq t'' \leq t' \wedge l_1, l_2 \in \text{path}(l, l') \wedge \text{Occurs}(\text{traverseto}(agt, l, l', r), t, t', s) \wedge \text{Holds}(\text{traversable}(agt, l_1, l_2, r), l, t'', s) \rightarrow \text{Holds}(\text{traversing}(agt, l_1, l_2, r), l, t'', s) \quad (31)$$

In fact, it is this uniformity constraint that is required to be monitored during the execution of the traverseto action in order for the action to be deemed successful. If the traverseto action is executing and the ability traversing associated with the affordance traversable becomes false at a timepoint within the duration of the action, this implies that an affordance which is required is absent. The affordance monitor would then send a signal to the agent to re-acquire the affordance.

10.3.1 Traversability Affordance Map

Note that the affordance of traversability is a nested affordance in the sense that it is dependent on the affordances of standability and turnability. In this respect, there is really no iconic map to describe this affordance other than that a path plan generated by the motion planner can only generate paths consisting of configurations which are both standable and turnable if a change of direction is necessary. Of course, there are many variations that would work. For example, the fact that full turnability is required would rule out certain path plans that the current robotic system can traverse. Figure 7 shows a legal path plan generated from the motion planner and superimposed on the original elevation map. It is completely contained within the legal affordance space which constrains the motion planner.

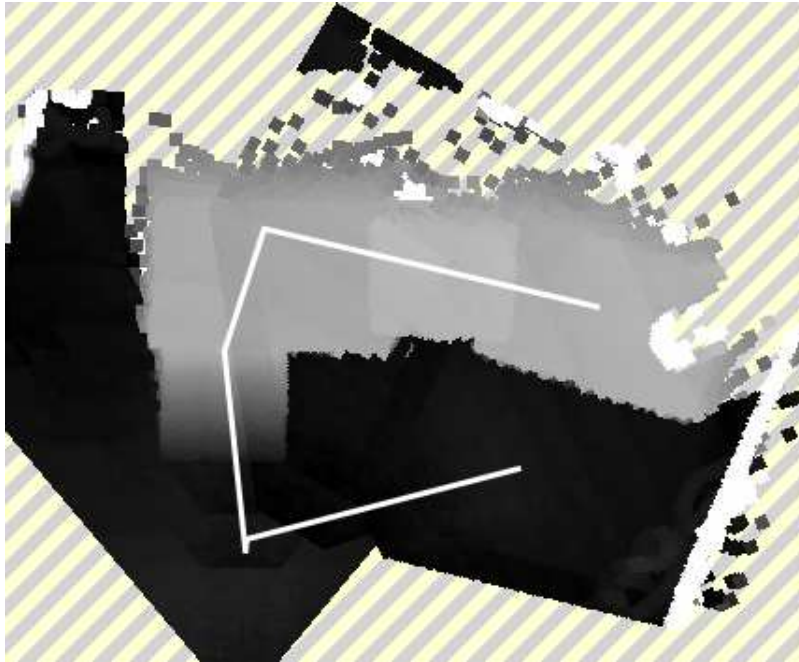


Figure 6: Affordance Map for Traversability

10.4 Reachability

In progress.

10.5 Graspability

Figure 7 shows an interesting situation the robot can get into when trying to traverse to its goal, a blue bucket. When it calls the motion planner, the motion planner is unable to return a legitimate plan due to obstacles. In this case, since the robot can identify entities with graspability and potential liftability affordances, it can internally simulate what the world would be like if a particular obstacle was not at its current location. If it calls the motion planner in this hypothetical world and it is successful then it has the motivation to go to a particular entity or entities and remove them or re-locate them. This is in fact what our robotic system would do in these circumstances. One can certainly claim that internal models are being used here for simulation, but this is of course the smart thing to do. In this case, we would like to combine intelligence with use of affordances not de-evolve to cockroach behavior. Of course, if that is the goal then one could modify the system to randomly choose a graspable and liftable object, move it randomly and start the planning process from wherever it is.

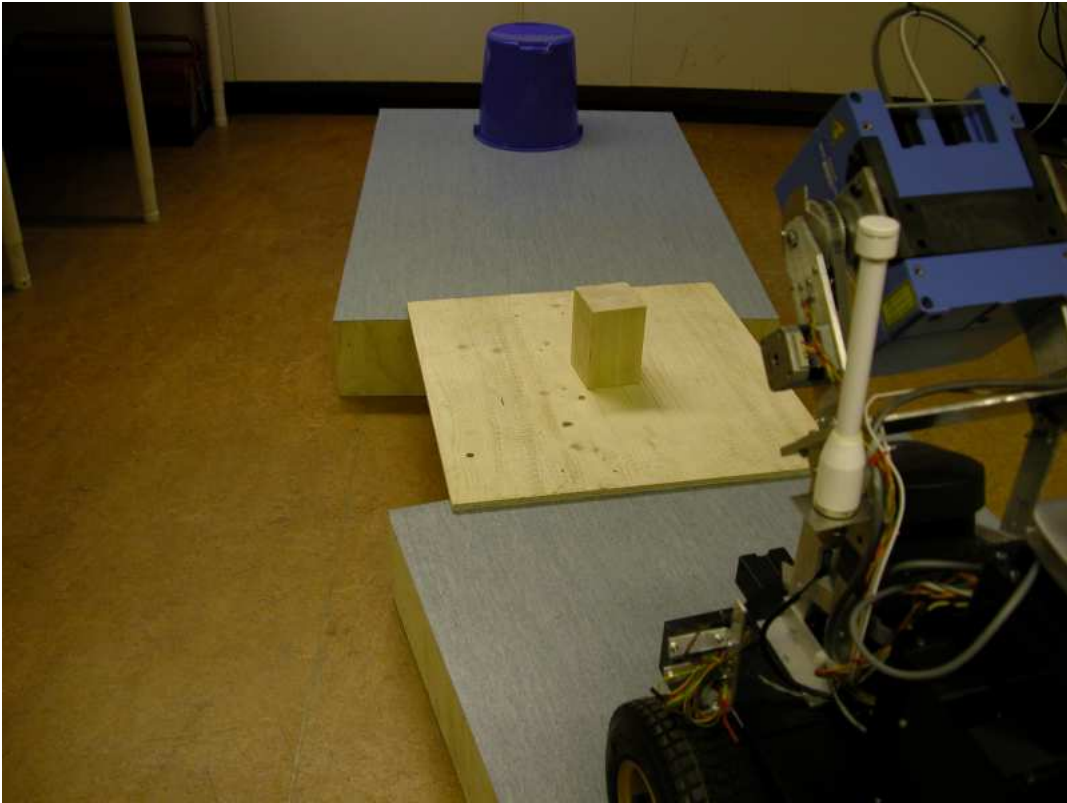


Figure 7: A graspability scenario

10.5.1 Graspability Affordance Map

An affordance map for graspability generated relative to the mission in figure 7, is shown in figure 8. In this case, one simply relates entity structures with the characteristics of graspability to an iconic map of the region under exploration.



Figure 8: Affordance Map for Graspability

10.6 Liftability

In progress.

10.6.1 Liftability Affordance Map

In progress.

11 Affordance Based Motion Planning

Any robotic system has to have functionality for moving from one location to another without colliding with obstacles in its surroundings. There are a great many proposals for motion planners, the majority of which fail to provide tractable performance in the real world. In this section, we propose using a number of sample-based approaches to motion planning which to a great extent can provide tractable motion planning functionality for a robotic system with many degrees of freedom. The novelty of the approach considered here will be to base the motion planner on the use of affordances rather than simple carving trajectories out of the robot's free configuration space.

The procedural framework used for this experimentation does not quite match the support architecture for affordances specified in section 12, but we explain this by the fact that the experimentation has provided great insight into how one could better support affordances in a cleaner manner in the architecture. Consequently, the framework proposed in section 12 is an outgrowth of insight gained in this experimentation.

A prototype implementation of the affordance-based motion planner described here has been deployed on our ActivMedia robotics platform.

11.1 The motion planning problem

In this section, we provide a brief overview of the sample-based path planning techniques used in the experiments. The problem of finding optimal paths between two configurations in a high-dimensional configuration space for robotics systems with many degrees of freedom is intractable in general. Sample-based approaches such as probabilistic roadmaps (PRM) or rapidly exploring random trees (RRT) often make the path planning problem solvable in practice by sacrificing completeness and optimality.

11.1.1 Probabilistic Roadmaps

The standard probabilistic roadmap (PRM) algorithm [KSLO96] works in two phases, one off-line and the other on-line. In the off-line phase a roadmap is generated using a 3D world model. This can be supplied to the system for a particular environment or it may be generated using SLAM techniques. Robot configurations are then randomly generated and checked for collisions with the model. A local path planner is then used to connect collision-free configurations taking into account the kinematic and dynamic constraints of the robotics platform. The local path planner can be more or less complex. For the purposes of our experimentation, we will generate straight line paths between configurations without any smoothing. Paths between two configurations are also checked for collisions.

In the on-line or querying phase, initial and goal configurations are provided and an attempt is made to connect each configuration to the previously generated roadmap using the local path planner. A graph search algorithm such as A* is then used to find a path from the initial to the goal configuration in the augmented roadmap.

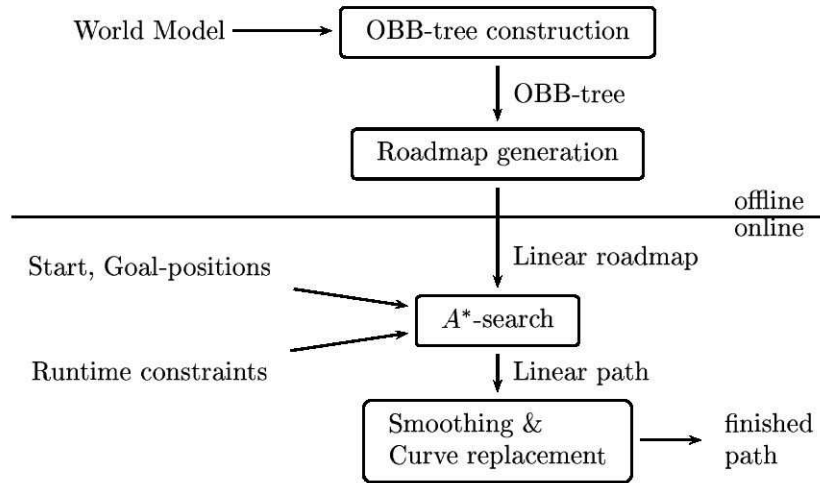


Figure 9: PRM path plan generation

Fig. 9 provides a schema of a PRM path planner that can be used in our robotic system. The planner uses an OBBTree-algorithm for collision checking and an A * algorithm for graph search. Here one can optimize for shortest path, etc.

One of the problems with the use of a PRM path planner is that the roadmap is generated once under the assumption that nothing changes in the world during the on-line phase. Additionally, the only constraints on the robot's configuration space taken into account are physical obstacles although there is some work on integrating nonholonomic constraints with such an approach.

11.1.2 Rapidly Exploring Random Trees

The use of rapidly exploring random trees (RRT) can be used to construct roadmaps online in realtime rather than off-line. This can be based on a map-generation, roadmap generation loop as the robot scans its environment.

The RRT method often produces relatively efficient motion planning capability. The algorithm [KL00] generates two trees rooted in the start and end configurations a robot wants to traverse by exploring the configuration space randomly in both directions. While the trees are being generated, an attempt is made at specific intervals to connect them to create one roadmap. After the roadmap is created, the remaining steps in the algorithm are the same as with PRMs.

11.2 Our approach

The general idea for our approach is that rather than thinking of the robot's free configuration space just in terms of physical constraints, we introduce the idea of affordance maps which the motion planner uses as additional constraints when generating paths. The intuition is that of generation of an affordance constrained polytope as output from the motion planner rather than just a free space constrained polytope.

When executing tasks, the robotic system may require certain affordances to be present in the configuration space it traverses. By generating and providing affordance maps of different types to the motion planner, the roadmaps generated will implicitly meet the affordance requirements. The motion planner will not generate any paths which violate the affordance constraints present.

In the first experiments we have pursued, we have generated affordance maps for stand-ability and turnability. These in turn are used to generate the traversability affordance map which is used by a RRT based motion planner to generate paths to locations in our robotic environment.

11.3 Behaviors and Dataflow

The LiU-IDA robotic platform is essentially structured around three conceptual levels. The first is the control level which includes basic drivers for sensing and motor activity; the second is the reactive layer which includes behaviors ranging from those with tight stimulus-act loops to more complex behaviors structures as hierarchically concurrent state machines (HCSMs). One special feature of this layer is that it includes a real-time interpreter for HCSMs. HCSMs can be specified as text files and be uploaded and executed directly without any compilation phase. The third conceptual layer includes all high level reactive and deliberative functionality.

This section provides diagrams of some of the HCSMs used in our path planning experimentation. A number of shortcuts have been taken to do the experimentation. Due to this, the actual implementation of the state machines does not separate the use and flow of affordances in the architecture to the extent described later in the report. This will of course be changed in the future. But this is a good start.

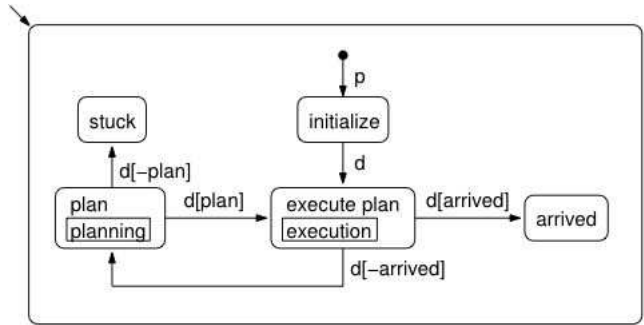


Figure 10: Root HCSM

Figure 10 depicts the root HCSM.⁹ The robot is initialized and moves into the execute plan stage (generation of an initial plan to move to a location in front of the robot is part of the initialization phase). If after that stage it has arrived at its target bucket then it goes into the arrive state, otherwise it goes into the planning state

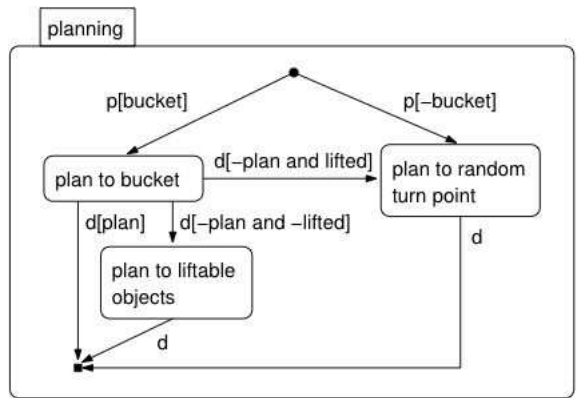


Figure 11: Planning HCSM

Figure 11 depicts the plan HCSM. If there is no bucket in sight, the system generates a plan to a random turnpoint. If there is a bucket in sight then it generates a motion plan to the bucket.

⁹In these diagrams, the p's represent pulse events which permit transition into states and the d[condition] events represent state termination events (d) and the conditions necessary to enter new states ([condition]). Note that states can be hierarchical and contain other HCSMs and can also be concurrent.

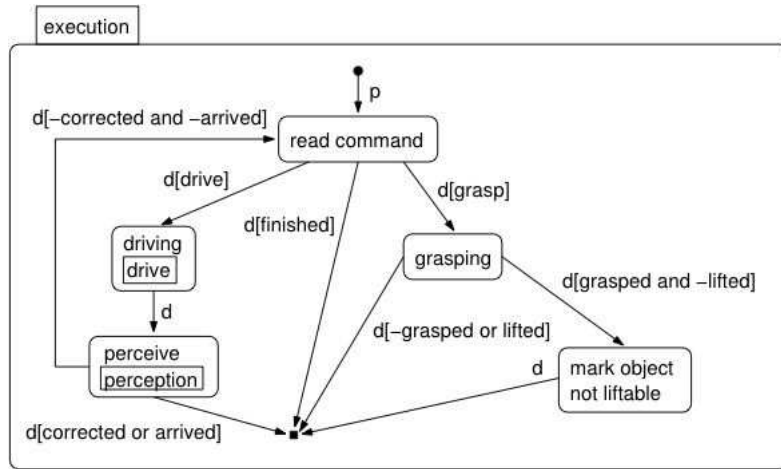


Figure 12: Execution HCSM

Figure 12 depicts the plan execution HCSM. Commands are read sequentially. If there is a drive command, the robot enters the drive state, drives a short distance and then enters the perceive state. If there is a grasp command the robot will enter the grasping state and try and lift the entity. Note that an affordance map for graspable objects is being used here. If it can not lift the object, it will mark it as not liftable.

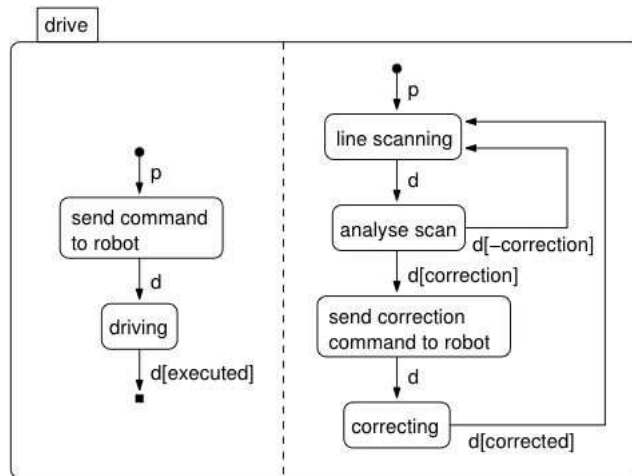


Figure 13: Drive HCSM

Figure ?? depicts the drive HCSM. This HCSM contains two concurrent HCSM's, one for driving in increments and one for line scanning for obstacles to react to and correct orientation so as to avoid the obstacle.

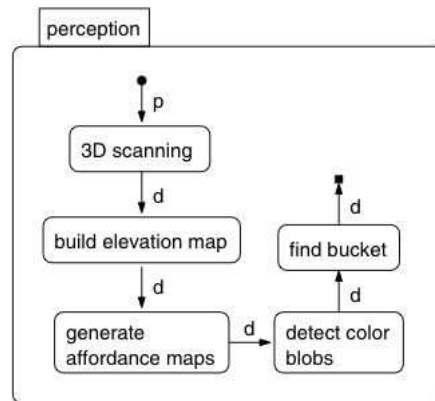


Figure 14: Perception HCSM

Figure 14 is perhaps the most interesting HCSM representing the perception HCSM. The idea is that the robot drives and perceives in increments at a high sample rate (although high rates are not yet achieved). The robot scans an area relative to its laser scanning capabilities, constructs an elevation map, merges this with previous scans and generates new affordance maps. Additionally, it uses its camera to detect colored blobs and localize buckets. The affordance maps have been described elsewhere, but it is important to note how these are used integrally in the motion planning stages and for the manipulation of entities of interest.

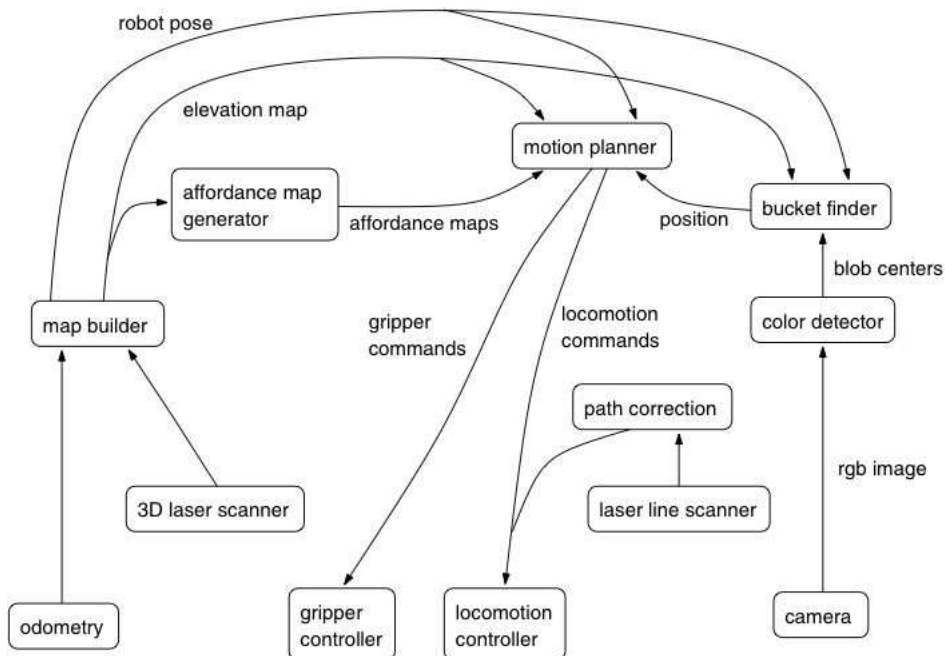


Figure 15: Dataflow in the LiU-IDA Robot Platform

Figure 15 is intended to represent data flow in the current iteration of the system for the

path planning experiments. We will not describe this in any greater detail for the moment.

11.4 Results

In order to acquire a hands-on feel as to just what types of representational support for affordances would be required in robotic systems embedded in domains such as ours (and similar to the MACs domain description), we chose to

1. build a mock environment for experimentation
2. engineer a tilt mechanism for our robot and SICK laser scanner
3. design HCSMs for this experimentation and implement a HCSM mechanism on the LiU-IDA robot
4. conceptualize, specify and implement an affordance-based motion planner
5. experiment with the above using different scenarios in the domain.

This of course involved a tremendous amount of man hours but we believe we have achieved what we set out to do. Rather than develop a theory of affordances from above and a framework for a support architecture ungrounded in practice, both the theory and the proposal for a support architecture have grown out of actual experience with a robotic system. We believe this detour was worthwhile and that the deliverable delays that ensued were warranted.

In section 12, we will propose a first iteration of a support architecture for affordances based on our experiments with affordance-based motion planning and limiting grasping activity.

12 Architectural Support for Affordances

12.1 Introduction

In the best of worlds, one would like to have a robotic architecture which is goal-driven and where the specification of high-level goals is made in some abstract language. Such goal specifications would be input to planners of various sorts and the resulting plans would be compiled continually into the base robotic behaviors which are part of the system. The base behaviors would execute in an affordance-based context. Of course, feedback both upwards and downwards in the architecture would also be required to avoid the criticisms associated with the traditional Think-Plan-Act cycle. The current state-of-the-art in hybrid autonomous systems has not yet achieved true integration of high-level

automated task planners with reactive and control levels in robotics architectures, although there is some progress in this direction. One of the main problems is the great amount of reconfigurability and monitoring that has to come into play as plans are generated, compiled into behaviors and executed.

Such integration is not one of the stated goals of the project, but we will find that similar issues would still have to be taken into account if one was to focus on goal-directed behaviors and combine these with affordances. This is in fact one of the goals of the project and a central part of the architectural support for affordances that must be built into the robotic system.

It is the intent of this section on architectural support for affordances to consider these issues.

12.2 Architectural Components for Affordance Support

12.2.1 Behaviors and Goals

Use implies dynamics and *purpose* implies one or more *goals* or *tasks* to achieve.

An agent operates in the environment, most often with a purpose. Let's assume that a set of *behaviors* or *activities* are associated with an agent's operation in its embedding environment. At the lowest level, an *elementary behavior* may be defined as a mapping of sensory inputs to a pattern of motor actions which is then used to achieve a *task* or purpose. *Composite behaviors* are built up recursively from other composite and elementary behaviors and do not necessarily imply direct mappings between sensory inputs and motor actions, but indirect mappings with a reactive or non-reactive performance. Consequently, behaviors of a deliberative nature are also composite behaviors but not elementary behaviors. Based on this broader definition of *behavior*, activities and actions can be *implemented* in terms of behaviors and goals or tasks can be achieved using concurrent execution of partially ordered sets of behaviors.¹⁰

For the time being, actions should be viewed more traditionally as high-level operator descriptions with pre-, durative, and post-conditions. Goals should be viewed as constraints on the values of sets of attributes which may have temporal duration (temporally extended goals). In this document, we have considered some proposals for suitable action theories

¹⁰Note that for the purposes of this discussion, the type and way behaviors are implemented is not central to the discussion. For example, most of what will be considered here fits in well with the use of Arbib's schemas, Arkin's approach to behaviors, behaviors specified as hierarchically concurrent state machines or some other form of automata, etc. What is important to some extent is that some subset of the behavioral repertoire of the robotic system is data-driven or event-based.

that could be used in this context as a basis for a formal specification of actions which could be combined with the material on affordances. A long term goal is to integrate this formal specification with the behavior-based part of the architecture. For the time being though, we will operate under the assumption that goals are engineered into certain types of high-level behaviors.

Start with your favorite behavior-based paradigm. We will assume that for any specific behavior,

- certain affordances may need to be present in order to invoke a behavior. Let's call these *trigger affordances*;
- certain affordances may need to be present in order to continue successful execution of a behavior. Let's call these *durative affordances*;
- certain affordances may need to be absent in order to continue successful execution of a behavior. Let's call these *inhibiting affordances*.
- certain affordances may be specified as goals which a behavior is required to achieve. Let's call these *goal affordances* and the behavioral process invoked as *affordance producing behavior*.

The union of trigger, durative, inhibiting and goal affordances for a specific behavior will be called *behavior-coupled affordances*.

Certain behaviors involve interaction between the agent executing the behavior and entities (attached or unattached) in the embedding environment. Grasping something is an instance of such a behavior. In order to successfully execute such a behavior, specific affordances associated with such entities in the particular task achieving context may have to be present or absent for successful execution of the task. Let's call such affordances *entity-coupled affordances*.¹¹

In addition, the specific configuration of affordances may be both temporally and causally dependent on perception and/or other events associated with the execution of the particular behavior or others occurring in parallel. This implies some form of higher-order relations between affordances in different spatio-temporal contexts. In order for behaviors to affirm the presence of such relations and dependencies, it is assumed that classification mechanisms can be called or event streams can be acquired with appropriate information about when and where certain affordances *occur*. In fact, such affordance events are an important part of the affordance-based support architecture.

In a manner similar to behaviors, the following types of affordances may be associated with any entity during the execution of behaviors which require particular kinds of entities,¹²

¹¹Note that such affordances are not intended to be intrinsic properties of the entities, but simply affordances that are present when the agent has a particular relation with such entities during the performance of a task.

¹²It is an interesting question, both conceptually and pragmatically, as to how such affordances should

- certain affordances may need to be present in order to invoke the behavior on an entity. Let's call these *trigger affordances for entities*;
- certain affordances may need to be present in order to continue successful execution of the behavior with the entity. Let's call these *durative affordances for entities*;
- certain affordances may need to be absent in order to continue execution of the behavior with the entity. Let's call these *inhibiting affordances for entities*.

The union of trigger, durative, and inhibiting affordances for a specific object will be called *entity-coupled affordances* or *entity-use affordances* .

12.2.2 Entity Generation Module

The *entity generation module* shown in figure 16 is intended as both a repository and generation mechanism for entity structures. Recall that entity structures consist of entity frames which consist of attribute/value pairs. These structures are intended to represent both cognitive and precognitive structures that are generated in the course of inhabiting and interacting with the environment. Entity structures can range from descriptions of surface patches to descriptions of complex objects such as other robots or geometric objects. *Dynamic entities* are intended to represent entities where attribute changes and transitions are the rule rather than the exception while *static entities* are intended to represent entities where there is little or no change. It is from such structures that affordance events can be created either by classifying the entity structures relative to affordance prototypes or creating entity/aspect trajectories and analyzing the dataflow generated.

be *attached* to such entities. There will probably be a requirement to distinguish between transient and static affordances for entities of particular types. For instance, certain types of entities have particular types of functional characteristics in particular contexts while other functional characteristics are invariant across context. Such invariant functional affordances could be part of the definition for certain object classes while others would have to be attached dynamically.

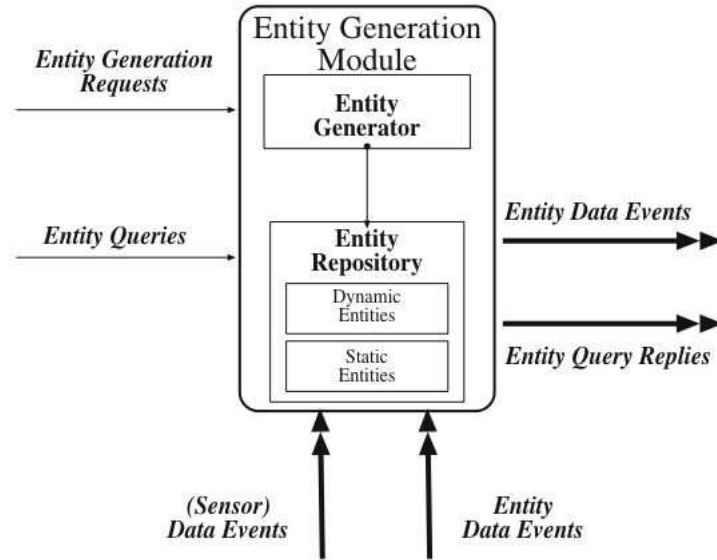


Figure 16: Entity Generation Module

12.2.3 Affordance Map Module

Affordance maps have already been considered in the context of the affordance-based motion planner described in section 11. In general, affordance maps are intended as an iconic form of representation which implicitly encodes spatial relations between entities in general or other characteristics related to surface entities. Primary affordance maps are intended to be used much like perceptual schemas whereas secondary affordance maps imply extended processing in their generation and can often be generated using other affordance maps as input. Some types of secondary affordance maps may even have pointers per pixel directly to different kinds of entity structures. For example, a graspability affordance map has pointers from the center pixel in a segmented region representing an entity to an entity structure in the entity repository. The affordance map repository and the entity repository should be considered primary representational elements in the affordance support architecture.

The *affordance map module* is shown in figure 17. It has a structure similar to the entity generation module and should permit certain types of querying and generation of affordance and entity data events.

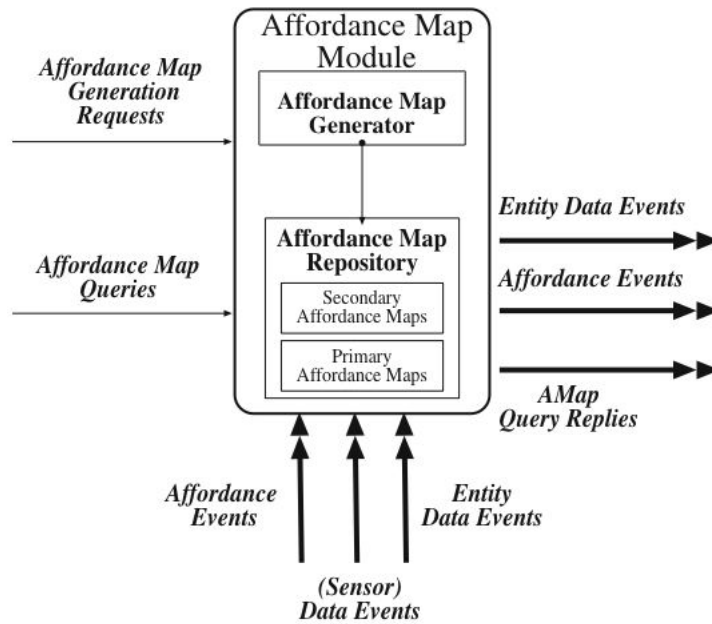


Figure 17: Affordance Map Module

12.2.4 Classifier Module

The *classifier module* shown in figure 18 is intended to be used for many different types of classification for entity structures, and situations. One of the ideas is that functional and form prototypes for entities could be stored here and as entities are generated by the system, similarity comparisons could be made between the generated entity and the prototype to determine classification relative to form or function. Situation classification is much more ambitious since a situation may contain temporal and event structures. In any case, there are a great many techniques that could be put to use in developing such a module. We have already alluded to the use of tolerance spaces in this respect and we expect that other MACs nodes could come up with classifiers for these different structures via machine learning.

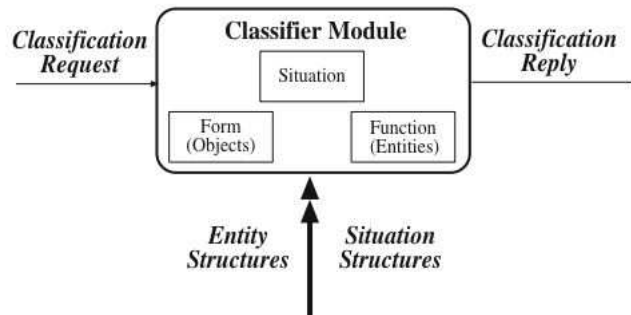


Figure 18: Classifier Module

12.2.5 Affordance Monitor Module

The *affordance monitor module* shown in figure 19 plays a central role in the affordance support architecture. Recall that as behaviors are executed, it is often the case that various affordances have to be present in the robots environment. It is in this module that affordance monitors can be set up to track the existence or non-existence of affordances. One can set up an affordance monitor by specifying a policy the monitor should use to track affordances. Such policies could be described in terms of attributes, entities, or even affordance events of interest.

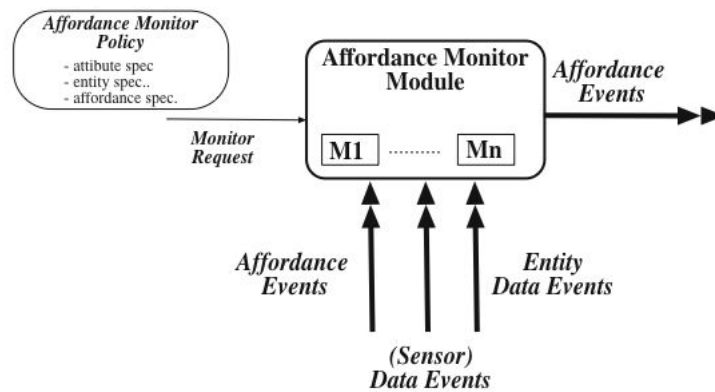


Figure 19: Affordance Monitor Module

12.2.6 Affordance Generation Module

The affordance generation module shown in figure 20 is one of the most important modules in the affordance support architecture. The basic idea is that in order for other components in the robotic system to maintain their services, they must be aware of the presence or absence of affordances which are relevant to their service requirements. Consequently,

these services need a means for stating their interest in various types of affordances (either explicitly or implicitly) and the ability to monitor the presence or absence of the appropriate affordances. This latter service is primarily the responsibility of an affordance monitor, but nothing rules out the possibility of a service directly receiving affordance events from an appropriate event stream in the robotic architecture.

We expect that from experience gained from robotic operations in different scenarios, a number of different affordances, both structural and entity-coupled will be discovered. In addition, such affordance will have different representational complexity. What we have in mind here is the spectrum ranging from affordances associated with the *direct perception* philosophy of Gibson to the use of affordances in a more indirect sense such as that of Norman. The intention with the architecture is to allow all types of affordances to be present and supported in the architecture.

In this sense, affordance generation module might often do nothing more than query an affordance map and generate an affordance event which is used by a behavior. This is well in the spirit of direct perception and behavior-based robotics with perceptual schemas. On the other hand, the affordance generation module may use many different services in the architecture to generate more sophisticated affordance events. It can for instance call affordance producing behaviors to try and acquire affordances which are absent in a particular situation, or it may call the classifier module and the affordance monitor module to generate additional entity data and affordance events used to define nested affordances or affordance structures. There is still much to be explored in this respect.

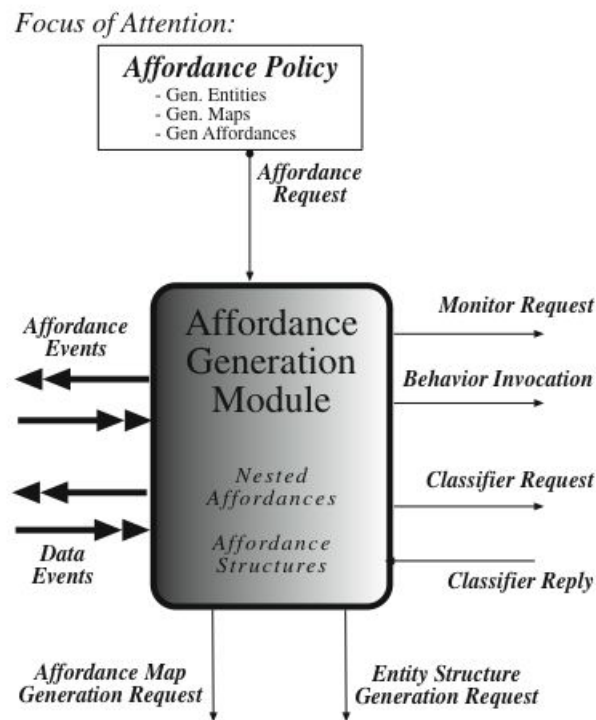


Figure 20: Affordance Generator Module

12.3 Affordance Support from Different Perspectives

In section 12.2, we provided a summary of the main modules or components we envision in the support architecture for affordances and have grounded these to some extent via formal considerations in terms of a logic for affordances (or at least a kernel theory) and through experimentation with a particular functionality, affordance-based path planning. In the following sections, we will consider integration of the components from two perspectives. The first is through the perspective of behaviors so as not to lose sight of the original motivation of directly perceived affordances which require as little processing of sensory stimuli as possible. The second perspective is essentially a bird's eye view of an architecture for affordance support where we abstract from the behavior perspective and provide functionalities which support the use of affordances for any component in an architecture, be it high-level deliberative services or behaviors.

12.4 Affordance Support from a Robot Behavior Perspective

The behavior-based robotics paradigm exemplified by Arbib's schema's or Arkin's behaviors has many of the intuitions associated with Gibson's use of the term affordances. In this case, there is a tight stimulus-act structure to behaviors with little or no intermediate representational structures involved. Although the spirit of affordances is implicit in such approaches, affordances are not 1st class citizens in such architectures, though they could be with some re-conceptualization and additional use of terminology. This being the case, it is important that such paradigms integrate well with the architecture we propose.

The main distinction is to make affordances 1st-class citizens in the architecture. They are reified and represented rather than remaining implicit in the use of perceptual schemas for instance. The change from a purely behavior-based perspective is simply to

1. associate trigger, durative, and goal affordances with behaviors
2. introduce the idea of affordance events, representing the presence or absence of affordances in a local situation
3. make sure there is a robotic service which uses affordance event flow to identify the presence or absence of affordances and feed this information back to specific behaviors, the result of which influences their associated processes
4. the addition of affordance maps, though not necessary provides an explicit iconic representational structure which lands somewhere between the islands of direct and indirect perception
5. A classification module of sorts would also seem to be necessary if one's intention is to allow entities segmented relative to both form and function.

One might still argue that most of these features are present implicitly in any sophisticated behavior-based robotic system and we ourselves would still have some trouble arguing otherwise at this point in the process of theorizing. One of our responses would be along the lines mentioned earlier in this report that affordances really offer a fresh ontological twist to the way one (and robots) think about the basic makeup of the world. Our hope is that constructing such systems from an affordance-based perspective will demonstrate less brittleness and more robustness and generality in the robot's interaction with the world. Such a hope is only a hypothesis that remains to be validated through empirical experimentation.

It is also important to observe that our comments pertain to behavior-based systems and an affordance perspective closest to direct perception. On the other hand, if one relaxes this constraint and thinks about affordances in the wider sense of the word where representations are allowed then this perspective has much potential in breaking new ground.

In any case, figure 21 provides an architectural framework for affordances from a robot behavior perspective. Here we observe all the components discussed above.

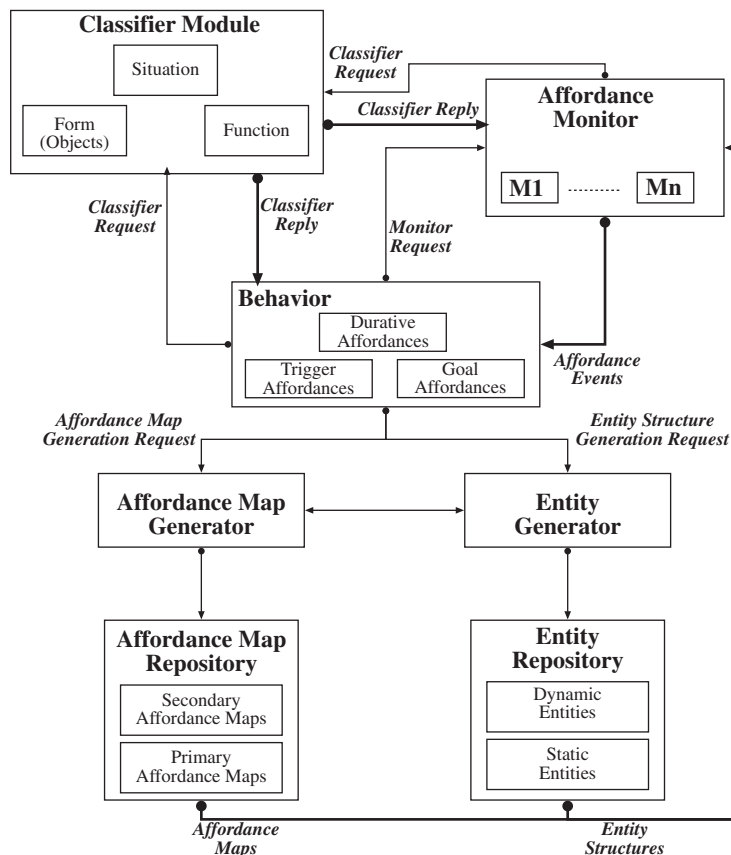


Figure 21: Affordances from a Robotic Behavior Perspective

12.5 Affordance Support from a Module Perspective

One of the main desiderata of the MACs project is to study the potential use of affordances in robotic systems from the perspective of both direct and indirect perception. Figure 22 provides an architectural framework for supporting affordance usage in robotic systems at both the behavior-based and knowledge-based levels of an architecture. Each of the modules has been explained in more detail in the document and it is straightforward to relate these modules to other activities in the project in the different workpackages. The basis for proposing these modules is tightly coupled to the experimentation done and the work with logical specification of the affordance idea.

Although the logical specification, high-level planning and its relation to this architecture is not yet present, we believe such components can be integrated quite nicely in the future and this topic is being pursued.

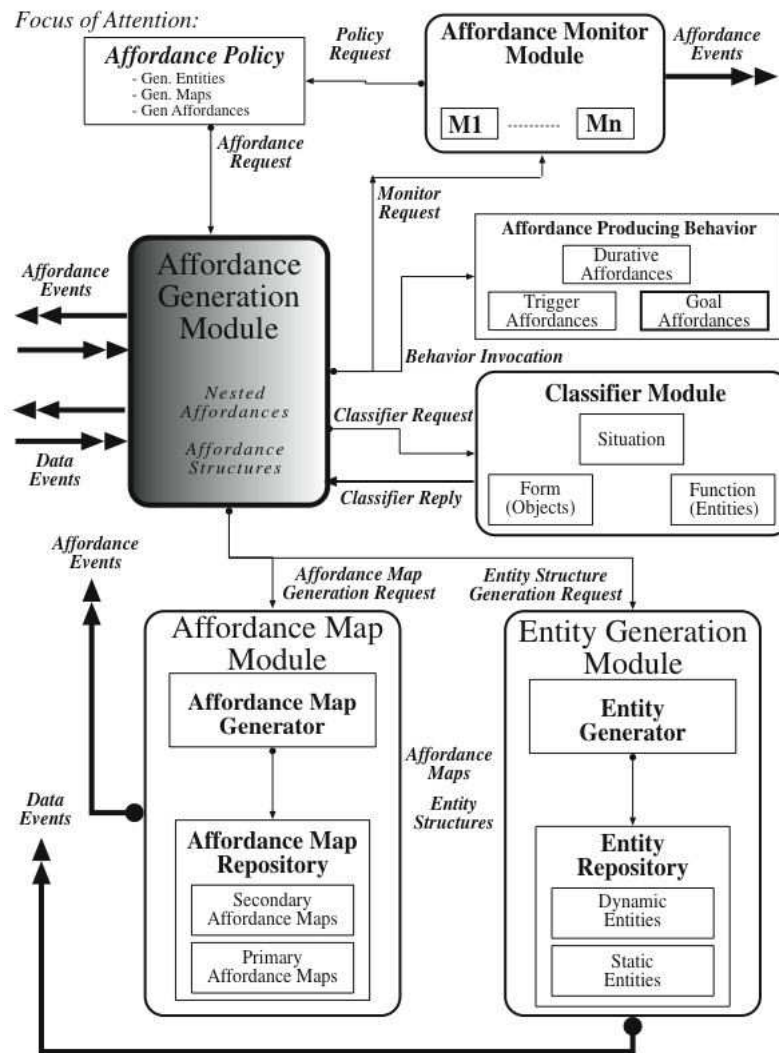


Figure 22: Affordances from a Module Perspective

13 Relation to Other Workpackages in MACS

The affordance support architecture described in sections 12.4 and 12.5 are specifically related to the usage mode in a robotic system. In other words, as the robot operates in a goal-directed manner to achieve mission tasks, it will use various affordances at both reactive and deliberative levels to help in achieving its goals. The affordance support architecture is intended to support this type of on-line use of affordances.

On the other hand, the affordances used must either be learned or hard-wired into the robotic system prior to their on-line usage. One suggestion for an architectural component

to support an on-line learning mode has been considered in WP2. It's instantiation with learning tasks has been described in WP5. In this section, we show how the affordance support architecture presented in this document can be integrated in a relatively seamless manner with the KURT robot and the on-line learning architecture presented in WP2 to support experimentation proposed in WP5.

Recall that an entity is defined as "any thing, object or individual with cohesive structure" and that "an entity consists of aspects". In the future experimentation with KURT, we will assume the use of a base set of sensors, a base set of manipulators and a base set of motor skills. Additionally, a feature extraction capability will be assumed that defines an abstract set of higher level features defined in terms of the sensors used. Features in this sense are simply primitive and complex entities. For example, a camera image is a complex entity consisting of an array of pixel entities with a number of aspects associated with each pixel entity. Feature extraction associated with WP3 can be integrated quite nicely with the entity structure idea. Additionally, combinations of state variables representing the movement and position of KURT and its parts in addition to that of its manipulators may also be defined in terms of primitive and complex entities.

In this context, the input to any learning module would be a selection of entity and aspect trajectories associated with KURT's sensors, manipulators and motor skills. Additionally, aspect and entity trajectories associated with the before and after states of specific reactive and composite behaviors used in experimentation would provide additional input to the learning module. This collection of complex time series data is the basic input from which the learning module's goal is to derive affordances based on behavior execution and the effects of that execution. This capability is fundamental to on-line learning experimentation to derive affordances or invariants associated with the time-series data.

Recall that the entity generation module in the affordance support architecture is intended to support the generation of complex time-series structures and that entity structure generation requests supply the focus to the entity generation module required for specific on-line learning tasks.

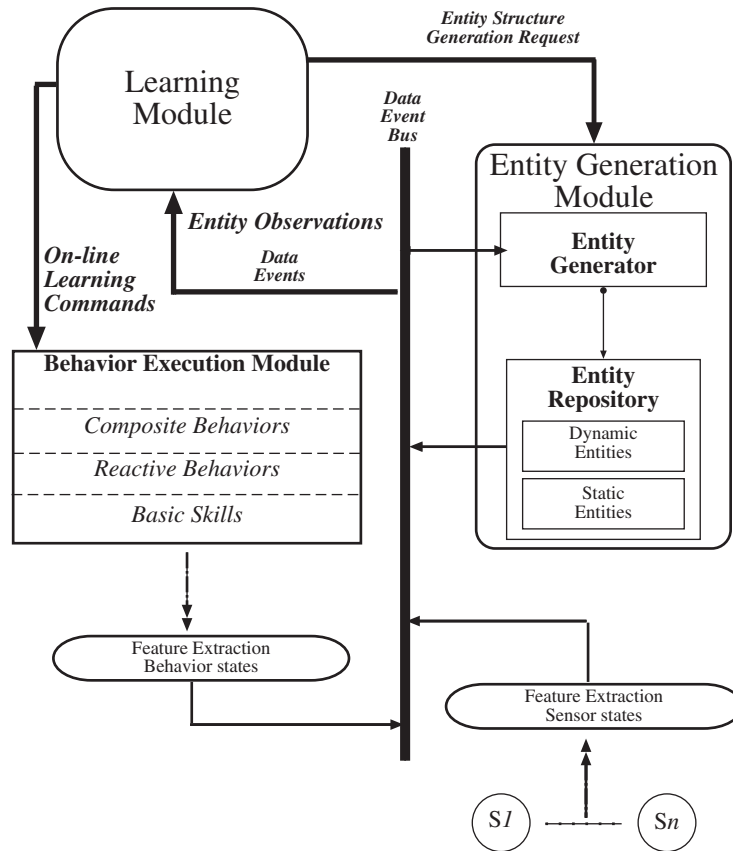


Figure 23: Integrating an On-Line Learning Module

Figure 23 shows how an on-line learning module would interface to the affordance support architecture described previously. For on-line learning experimentation, the learning module would be responsible for commanding which behaviors should be executed and which entity structures should be generated. The entity and aspect trajectories associated with the entity structures would provide input to the learning module.

Note that feature extraction can be a relatively trivial process where data values are pulled from sensor drivers, or it can be a complex set of signal and computational processes before a selection of abstract features are generated. In the diagram, the feature extraction processes are made distinct from entity generation, but in some sense, one can view the entity generation module as a support mechanism for feature extraction at higher levels of abstraction.

In any case, there is a nice level of integration between the support architecture for affordances proposed in this document with much of the work proposed in the other workpackages.

14 Terminology

To do. It is very important to have a concise vocabulary which is agreed to by all parts. A number of definitions and terms are used in this document. We expect to generate a glossary at a later date.

Entity – An entity is ...

15 Conclusions

The contributions in this deliverable are three-fold:

1. A first iteration at generating a formal theory for affordances which is true to the spirit of work in ecological psychology and which attempts to integrate ideas from situation semantics and theories of action and change has been presented. The point to the formal theory is to provide a clear specification of what affordances are and how they are intended to relate to the environment and agents acting in the environment. It is important to note that the mechanization of these ideas in a robotic system does not necessarily have to use the formal logic associated with the specification although we do believe that when higher-level affordances are required in such architectures, the logic and eventual inference mechanism associated with it can eventually be put to good use.
2. A prototype implementation of a sample-based motion planner for a robotics system that appeals to the use of affordances has been described. It is important to note here that it is the power of affordances as an abstraction and modeling mechanism that is of interest to us here. An ideologically strict behavior-based roboticist might claim that what is being done here is nothing new. Similar structures that do not mention affordances could do a similar job. Our answer to that is that that would be missing the point. In fact, one of our work package groups had just such a problem in understanding this point. We believe that by making explicit the types of structural affordances at play in robot navigation and defining motion planning in these terms will in fact result in a new level of robustness for such functionality and will make the task of specifying and designing this and related functionalities much easier to do. Some attempts have also been made to relate the ideas used in the affordance-based motion planner to the formal specification we proposed in the previous point.
3. An architecture for affordance support has been proposed and we have shown how it can be leveraged for on-line use as an affordance support mechanism for a robot and also as to how it can be leveraged for use with an on-line learning module. Both aspects are equally important in the MACS project. Certainly, the specification of an architecture and its actual implementation are two distinctly different tasks.

The devil is in the details and we believe the implementation task will be quite challenging.

Much of year 2 in the project will be spent developing the ideas put forth in the formal theory of affordances and beginning the process of implementation of the architecture for affordance support.

16 Acknowledgements

The logical specification for affordances and ontology is due to Doherty with some help by Szalas with situation operators. The affordance support module specification is primarily due to Doherty with much input and discussion from Merz, Wzorek, Rudol, Wingman. The affordance-based motion planner is due to Wzorek, Rudol, Merz, and Doherty. The development of the robotic platform is primarily due to Merz and Rudol. The conceptualization of many ideas here is due to the WP4 collective who have spent countless hours trying to make sense of the elusive idea of affordances.

References

- [Ark98] Ron Arkin. *Behavior-Based Robotics*. MIT Press, 1998.
- [Che03] A. Chemero. An outline of a theory of affordances. *Journal of Ecological Psychology*, 2003. To appear.
- [DGKK98] P. Doherty, J. Gustafsson, L. Karlsson, and J. Kvarnström. (TAL) temporal action logics: Language specification and tutorial. *Electronic Transactions on Artificial Intelligence*, 2(3-4):273–306, 1998. <http://www.ep.liu.se/ej/etai/1998/009/>.
- [DLS03] P. Doherty, W. Łukaszewicz, and Szalas. Tolerance spaces and approximative representational structures. In *Proceedings of the 26th German Conference on Artificial Intelligence*, Hamburg, Germany, 2003.
- [DLSS03] P. Doherty, W. Łukaszewicz, A. Skowron, and A. Szalas. Combining rough and crisp knowledge in deductive databases. In *Rough-Neuro Computing: Techniques for Computing with Words*. 2003.
- [Gär00] P. Gärdenfors. *Conceptual spaces: the geometry of thought*. MIT Press, Cambridge, Mass., 2000.
- [Gib79] J. J. Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin, Boston, 1979.

- [KL00] J. J. Kuffner and S. M. LaValle. RRT-connect: An Efficient Approach to Single-Query Path Planning. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 995–1001, 2000.
- [KSLO96] L.E. Kavraki, P. Svetstka, J.-C. Latombe, and M. Overmars. Probabilistic Roadmaps for Path Planning in High Dimensional Configuration Spaces. *Proc. of the IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.
- [Mur99] Robin R. Murphy. Case studies of applying gibson’s ecological approach to mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 29(1):105–111, 1999.
- [Mur00] Robin R. Murphy. *Introduction to AI Robotics*. MIT Press, 2000.
- [SB89] Louise Stark and Keven Bowyer. Achieving generalized object recognition through reasoning about association of function to structure. *International Journal of Intelligent Systems*, 10, 1989.
- [SS96] A. Skowron and J. Stepaniuk. Tolerance approximation spaces. *Fundamenta Informaticae*, 27:245–253, 1996.
- [Ste00] Mark Steedman. Formalizing affordance. In *Proceedings of the 24th Annual Meeting of the Cognitive Science Society*, 2000.