EU FP7 CogX
ICT-215181
May 1 2008 (52months)

DR 9.3:
Proceedings of Summer School 2010

Danijel Skočaj, Luka Čehovin and Marc Hanheide

University of Ljubljana, University of Birmingham
⟨danijel.skocaj@fri.uni-lj.si⟩

Due date of deliverable:	July 31, 2010
Actual submission date:	July 28, 2010
Lead partner:	UL
Revision:	final
Dissemination level:	PU

This document describes the CogX Summer School organised at University of Ljubljana in Ljubljana, Slovenia April 24-30, 2010. This was the second out of three Summer Schools planned for. The main parts of the school were technical tutorials covering the use of the common representations, planning techniques and related software and hardware in CogX, invited tutorials about the state of the art approaches on related topics, and a project to be solved in groups to get hands-on experience and act as a team building activity.

Executive Summary

The second out of three Summer Schools planned for in the CogX project was organised by UL April 24-30, 2010. As promised in the workplan for the project, rather than having many invited speakers and a full schedule which is the standard setup for summer schools, the emphasis was on hands-on project work. The rationale behind this being that participants should get to use the common hardware platform and sensors as well as work with the common software platform CAST [1] and other tools, representations and mechanisms that are commonly used in the integrated systems that are being developed.

There were three invited speakers. David Hogg from the School of Computing, University of Leeds presented a tutorial about the role of representation and learning in activity analysis. Norbert Kruger from the University of Southern Denmark gave a tutorial on early cognitive vision. Ron Petrick from the School of Informatics at the University of Edinburgh presented a tutorial about representations for classical and knowledge-level planning. The topics of the presented tutorials were highly related to the main research issues we have been addressing in the project and which were in the focus of this summer school (visual perception, learning, representations, planning).

At the end of the week, each participant had helped create one of five integrated systems where components, both new and existing, had been combined to allow the robot to move around autonomously and visually detect objects. The robots had to plan how to move around the environment and what questions to ask persons that were present. The summer school provided an important opportunity for the participants to meet and interact both in work and social situations. Integration is at the heart of an IP project and knowing the hardware and software system as well as each other are important ingredients for making this process smooth and efficient. To conclude, the summer school was very successful.

Role of the Summer Schools in CogX

The CogX project aims not only to contribute new theories but also to implement and create instantiations in robots to test these theories. In CogX the Summer Schools provide an important vehicle towards this.

The objectives of the CogX Summer Schools include:

- train the researchers in the techniques and tools to be used in the project, and in the methods employed in the state of the art in the community
- establish a common ground of theoretical knowledge

- efficiently communicate knowledge to the researchers, both from external parties in the form of invited speakers and from researchers within the consortium
- increase impact of the dissemination by including external parties (invited speakers and participants) in the summer school who get a close look at the project
- build strong connections between the researchers within the consortium by getting together for an extended time, interacting in working and as well as social contexts

Contribution to the CogX scenarios and prototypes

Building a robotic system, which integrates different functionalities into one coherent system, is one of the main goals of the CogX project. Although the system is expected to exhibit very different capabilities (from navigation, to learning, communication, and manipulation), all different subsystems that implement these capabilities are supposed to use the same hardware basis, the same architecture schema and toolkit (CAST), and the same principles of representation sharing (based on the Binder and Planning SA). It is therefore very important that all the researchers are familiar with these principles and that they know how to integrate the components they have been developing in CAST. A particular focus of this spring school was on familiarisation of the researchers with how to create and manage the robot beliefs about the environment and how to plan actions to fulfill the given goals. These questions are essential when developing system prototypes for CogX scenarios. In addition, the chosen tasks were inspired by both scenarios, Dora and George. The spring school in Ljubljana enabled us to address all the questions above and to help the researchers to understand these mechanisms better. It has therefore made a very important contribution to scenario based integration.

1 Tasks, objectives, results

While the first CogX summer school (CSS) in Stockholm in 2009 aimed to introduce the general hard- and software in the project and emphasised spatial representations and actions this year's school focuses on autonomous deliberative behaviour and knowledge representations. The school was designed as a combination of guest lectures and practical sessions. The lecturers invited for the school were selected to complement the practical experience of implementing AI planning and manipulating knowledge representations as a result of epistemic actions. It was decided to build upon experiences participants gathered in the first school and not start completely from scratch. Therefore, the mobile robot platform was employed with its ability to move autonomously and to maintain spatial representation. The tasks in the practical sessions were inspired by both scenarios – Dora and George – and in fact were built upon the software and hardware basis developed in these two scenarios in the first year of CogX.

1.1 Preparations

Hardware: In order to provide a really hands-on robotic experience we shipped four robot platforms equipped with pan-tilt unit and stereo camera setup to Ljubljana. The setup was exactly as is being used in the two integration scenarios Dora and George to guarantee maximal compatibility with the continuous integration in CogX. Besides the real platform, we also developed a mostly complete simulation environment that allowed for testing the developed system independently of any hardware. Here, we reused the setup and system documented in DR7.2 to a great extent.
Software: Following the software release schema in CogX the summer school is linked to one of two major annual milestones per year. Consequently, integration efforts peak the weeks before the school and lead to a stable collection of software that is then used by all the participants. The software employed at the CSS 2010 was based on the integration system Dora and George. In addition to the abilities documented in DR7.2 some components have been added:

- a people detector component that allows us to combine laser measurements and visual input to detect people in the vicinity of the robot, enabling it to initiate interaction
- a revised model of beliefs in our system as the foundation for the advanced knowledge representation for the school
- a GUI-based simple dialogue system to enable the robot to ask questions and receive answers, avoiding difficulties with speech recognition

- a new Markov-logic based binding and tracking mechanism on the basis of the beliefs to maintain a consistent knowledge representation in the robot's working memory.

These extensions all go in line with the integration efforts for the next milestones of integrated demonstrators for George and Dora.

To ease implementation and maximise the learning opportunity, a virtual machine image with Ubuntu and all required software components and the simulation environment pre-installed was set up and distributed among participants prior to the school. Also prior to the start of the school, participants were asked to install the (virtual) system and have a few test runs to get used to the general procedure.

1.2 Project Work

All participants where divided into project groups, including PIs. The two main factors when forming the groups were i) diversity with regard to the institutions people work for and ii) to distribute the knowledge and skills as evenly as possible among the groups. The tutorials on belief models and on planning provided the foundation for the tasks that had to be solved by each group individually. Scores were assigned for each of the three tasks which were:

1. "Whom do all these records belong to?" This first task was designed to make people aware of the challenges they have to face. In this task there was no autonomous behaviour of the robots, but the participants had to plan action sequences by themselves and remote-control the robot, to find (vinyl) records in the environment, recognise them visually, and find persons to ask who the owner of that particular record might be. The task also sketched the final task, where the robot has to learn the associations and the location of records all autonomously. See page 21 for further details.

2. "Play it again" was designed as a first task the robot plans and executes autonomously. Its task was to explore the test arena and find records and remember their locations. It is a revival of the final task in the first school, but now involving the whole CogX processing framework including belief models, goal management, and planning. The participants had to implement the processing chain that adds recognized objects into the knowledge representation, develop a suitable planning domain, and decide how they represent acquired knowledge in terms of beliefs. These are key research questions of CogX that were studied in this simplistic scenario. See page 23 for further details.

3. "InAct Pablo": We called the robot developed during the spring school Pablo (Planning And Belief models to Look for Objects) to reflect the

central aims of the school. This final task was designed to be a revival of the first one but now performed completely autonomously by the robot. The participants had to program the robot such that is was able to localise humans and records, acquire knowledge by asking questions or by looking around, and finally, demonstrate all the knowledge it acquired. This interactive task is detailed on page 26.

During the five days of practical work including implementation and testing all five teams successfully accomplished task 1, four teams managed task 2 (one team decided to skip this task and focus on the final task), and all had some success in the final task. The best team managed to find all of the records, and also ask the right questions to humans in the environment to learn which record belonged to whom in this final task.

1.3 Lessons Learned

Following the lessons learned in the first CSS we again kept the number of speakers quite small and focused on the practical sessions. The heterogenous group composition of about 5 participants per group really fostered communication and helped creativity. Basically, the decision to not start from scratch but rather to implement new abilities on the basis of the system developed in year 1 worked quite well. But an additional rather spontaneous lecture was required to help a number of new project members to catch up with the concepts. The different level of prior knowledge among participants is a challenge that needs to be explicitly coped with in the coming summer schools in the project. Having three tasks to compete in, on the one hand provided some scaffolding for incremental implementation, but on the other hand also increased time pressure and caused some overhead. In the future, one would probably reduce the effort to two incremental tasks. It is promising that despite the short amount of time, all the teams did very well in developing a system that was ready for a competition. These kinds of code marathons have proven to be very promising approaches to boost development in the project. The final summer school system provides the code basis to develop the year 2 milestones of Dora and George.

1.4 Relation to the state-of-the-art

The tasks for the summer school were all designed along the lines of research in CogX. In this year we placed special emphasis on interactive learning and autonomous planning; in particular to plan for knowledge gathering actions. We employed the state-of-the art technology in continual planning and provided the participants a first hand experience in the foundation, limits, and logics of AI planning. Hence the emphasis of this summer school was on work accomplished in work packages 4 and 6.

References

[1] N. Hawes and J. Wyatt, "Engineering intelligent information-processing systems with CAST," *Advanced Engineering Infomatics*, vol. 24, no. 1, pp. 27–39, 2010.

2 Proceedings

The proceedings is a modified version of the proceedings handed out to the participants of the CogX Spring School. Most of the local information has been removed and some of the information that was only provided online on the CogX intranet has been included.

First, some general information about the Spring school is given. Then the course material is presented. This material was provided for the participants on the CogX Wiki and served as main instructions for work. We present it here in its original form, including two technical tutorials that were given. The second part of the proceedings is composed of three tutorials presented by the invited speakers.

2.1 General information

2.1.1 Schedule

Sat, 24th April

09:00-09:30	Welcome and opening (Jeremy, Danijel, Marc)
09:30-10:45	Tutorial, David Hogg
10:45-11:00	*Coffee*
11:00-12:30	Tutorial, David Hogg
12:30-13:45	*Lunch (Hombre)*
13:45-14:30	first task induction & technological introduction (Marc)
14:30-15:30	Robot setup and preparation

15:30-15:45	*Coffee + snacks*
15:45-17:30	**1st (fun) competition**, 15 minutes performance per team + 5 minutes setup time
17:30-19:00	Binding tutorial (Pierre) + Q & A
20:00-end	*Dinner (Šestica)*

Sun, 25th April

09:30-10:45	Tutorial, Norbert Krüger
10:45-11:00	*Coffee*
11:00-12:30	Tutorial, Norbert Krüger
12:30-13:45	*Lunch (Hombre)*
13:45-15:00	Tutorial, Ron Petrick
15:00-15:15	*Coffee*
15:15-16:45	Tutorial, Ron Petrick
16:45-17:00	*Coffee + snacks*
17:00-18:30	Planning tutorial (Moritz)
18:30-end	*Walk to castle & dinner (Vodnikov hram)*

Mon, 26th April

09:30-10:30	Task induction (Moritz, Marc) + Q & A
10:30-12:00	1st team session
12:00-12:30	another Q & A in plenary
12:30-13:45	*Lunch (Hombre)*
13:45-16:00	Hack & Test
16:00-16:30	*Coffee + snacks*
16:30-19:00	Hack & Test
19:00-20:30	*Dinner (self-organised)*
20:30-22:00	(optional) Hack & Test

Tue, 27th April

09:30-10:00	Morning Q & A
10:00-12:30	Hack & Test
12:30-13:45	*Lunch (Hombre)*
13:45-16:00	**2nd competition**
16:00-19:00	*Social event*
19:00-20:30	*Dinner (Sempre)*

Wed, 28th April

09:30-12:30	Hack & Test
12:30-13:30	*Lunch (IJS Sodexo)*
13:30-16:00	Hack & Test

16:00-16:30	*Coffee + snacks*
16:30-19:00	Hack & Test
19:00-20:30	*Dinner (self-organised)*
20:30-00:00	(optional) Hack & Test

Thu, 29th April

09:30-12:30	Hack & Test
12:30-13:30	*Lunch (IJS Sodexo)*
13:30-16:00	Hack & Test
16:00-16:30	*Coffee + snacks*
16:30-19:00	Hack & Test
19:00-20:30	*Dinner (at department, Pizza)*
20:30-02:00	(optional) Hack & Test

Fri, 30th April

09:00-09:30	Setup time
09:30-13:00	**final competition** (30 minutes performance + 10 minutes setup)
13:00-14:00	*Lunch (IJS Sodexo)*
14:00-14:30	Award ceremony & closing
after 14:30	Leaving

2.1.2 Participants

ALU-FR

- Moritz Goebelbecker
- Thomas Keller

BHAM

- Rustam Stolkin
- Charles Gretton
- Marek Kopicki
- Marc Hanheide
- Jeremy Wyatt
- Veronica Arriola Rios
- Richard Dearden

TUW

- Michael Zillich
- Kai Zhou
- Thomas Mörwald
- Andreas Richtsfeld

KTH

- Kristoffer Sjöö
- Yasemin Bekiroglu
- Patric Jensfelt
- Andrzej Pronobis

DFKI

- Geert-Jan M. Kruijff
- Pierre Lison
- Shanker Keshavdas
- Benoit Larochelle
- Harmish Khambhaita

UL

- Danijel Skočaj
- Aleš Leonardis
- Matej Kristan
- Alen Vrečko
- Marko Mahnič
- Barry Ridge
- Luka Čehovin

Invited speakers

- David Hogg
- Norbert Krüger
- Ron Petrick

2.1.3 Venue

The Spring school took place in the building of Faculty of Computer and Information Science, University of Ljubljana, where the Visual Cognitive Systems Laboratory is located. Most of the building was occupied. Each team had its own room to work in, a part of the building was reserved for the test area, while the plenary sessions were held in the lecture room.

2.2 Course materials

2.2.1 System setup and preparation

This should be carried out by everyone participating in the school. Most important: The system has to be installed on every partner's robot (Bamboo Laptop)

We (Nick & Marc) will provide online support on Wednesday 21st April 2010 to sort out your problems. Please try before that date, so you have something to ask!

This year we will be working both, on a real robot and in a simulation environment. That means everybody can have the system running on their own machines!

Preparing your Ubuntu System

We will only provide support for Ubuntu-based Linuxes. We had success with 9.04, 9.10, and 10.4b releases. We recommend to install Ubuntu 9.10 Desktop edition as all this documentation refers to that release.

You can also use a VirtualBox to run the system in a virtual machine if you don't want to install it natively on your machine. A prepared virtual appliance can be downloaded from here. Unpack the archive and follow option 1 in this tutorial.

Using the virtual machine is recommended for everyone who does not want to install Ubuntu on her/his machine. Please assign as much memory and processing power to the virtual machine so you can actually run our system! However, every partner has to at least install their Bamboo with the real system to run it on the robot. This has to be done before the school starts.

Installing all required ubuntu packages

1. run sudo synaptic in your terminal
2. select from the menu [File->Read Markings...]
3. select the attached file (This file is for ubuntu 9.10 [karmic], it might not work with others. You can open it in your favourite editor and see which packages you should have installed in your system)
4. apply all changes by clicking [apply]
5. you might also want to install eclipse and subclipse to ease development for the school.

Install all cogx specific dependencies (CAST etc.) We provide a so-called GAR-installer to ease installation. Simply follow these steps and you should be able to install all relevant packages:

1. create your local folder which you want to use to build everything in (doesn't really matter, everything will be installed in /usr/local anyhow). do cd <name of your directory.
2. check out the installer: svn co https://codex.cs.bham.ac.uk/svn/nah/cogx/code/tools/gar-installer/trunk gar-installer! You will need your cogx svn username and password. This will create a directory "gar-installer", change into it: cd gar-installer
3. GAR has a management of dependencies. So we can now simply change into the directory "cogx/full-blown" and tell GAR to install this. It will automatically also fetch, configure, build, and install all require packages. So, let's do it (here a 'sudo' is required to gain administrative permissions, enter your password when being asked for it: cd cogx/full-blown; sudo make install
4. GAR now should be doing everything for you. It might find some ubuntu packages still missing if you haven't installed everything in step 1 (see above). It then asks you to confirm the installation and you should simply hit [return].

This procedure of course is prone to fail. Just let us know when you experience any problems. Missing libraries are usually to be blamed on the one you modeled the dependencies in the GAR configuration files and most ofen on "boost" libraries that create a version mess across Ubuntu releases. First try to install missing ones again using "synaptic" or give us (the school team) a shout. Another hint is to look at last year's tutorial to install CAST.

Almost there: learn the basics If you reached here, you are almost done. You have all 3rd party stuff installed. Now you should, if you haven't done already last year or sometimes else, familiarise yourself with CAST. The tutorial from last year is still valid and it is assumed that you know the basics about CAST as discussed in this tutorial.

2.2.2 Introduction

General

- this year's organisers (content-wise): Pierre Lison, Moritz Göbelbecker, Marc Hanheide (and thanks to all the supporters: Michael, GJ, Kristoffer, Danijel, Marko,...)

- remember last spring school, where we focused on navigation and spatial (and object detection)
  - this time we'll build on that
  - the mobile robot will still search, but not manipulate
  - it will plan and execute actions to learn more about its world
  - its world is composed of objects and humans, places and rooms
- we focus on
  - belief models as a generic representation of the robot's world that is maintained and consolidated through binding processes...
  - ...and on planning to extend this representation by carrying out actions.

What you are supposed to learn

- how to create a planning domain for a given problem / task
- how create beliefs about the world that are used to derive the planning state
- how the general processing of (the new) binding works and how it can be used
- how to employ the overall processing chain motivation → planning → execution operates and intertwines with the belief-oriented representation of the world
- where the problems in real world application lies and why we would need more probabilistic beliefs
- working in a developer team (using svn, talking to people, understanding other's code,...)
- where we still introduced bugs :-)

Technologically

- CAST is used once again, the processing principles remain the same, though some extensions could be learned. You should have familiarised yourself with these Tutorials:
  - CAST tutorial (this one is important)
  - nav.sa tutorial (you should learn this, but not necessarily for this spring school)

  - vision.sa tutorial (you'll need the object learning bit from this one)

- it'll be more JAVA than C++ this time! You can use C++, but we strongly recommend using JAVA this time. If you don't like it, do pair programming, try to find your way with C++, do the brain work (define the planning domain, think) or go implementing CAST bindings for your favourite language and port CAST to Windows...

  - actually, there is not too much programming involved, most of it is already there.

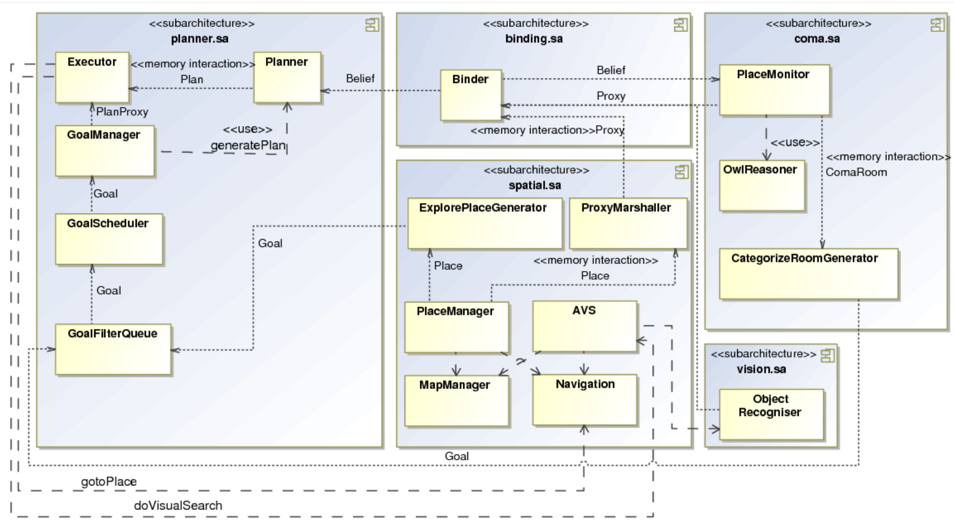
- we have a simulation environment (and even a virtual appliance to be installed in VirtualBox) => everybody can test and program!

Subarchitectures

- for those who haven't heard about subarchitectures: There are a function-oriented of our systems:

  - nav.sa
  - spatial.sa
  - coma.sa
  - vision.sa
  - comsys.sa
  - planning.sa
  - binding.sa
  - ... (there might be even more)

- our focus is on binding and planning



Pablo

- our robot PaBLO
- "Planning and belief models to look for objects"
- PaBLO...
  - can drive around
  - can build maps
  - can see objects & persons (ok, the name should be PabloP?!)
  - can be engaged in simple GUI dialogues
  - is smart (that's your part)

Prove Pablo's smartness in tasks

- we'll have not one, but three competitions!

  1. Be Pablo's binder and planning component yourself: Me-Pablo: You will operate the robot using the same level of abstraction that is available to the component your are using later on to implement autonomous Pablo.
  2. First year's task revised: Play-it-again-Pablo: Pablo will plan and act autonomously to find objects and report there whereabouts.
  3. Harvesting information from humans:IntAct-Pablo: The holy grail to be found in this school; find and interact with humans and relate information to each other in a meaningful way. Find out, who ones which record!

Schedule

- we might adjust if we need to
- we have several Q & A session (e.g. every morning) where you should ask your questions regarding the task and other lacks of clarity
- we are around most of the time
- well, here is detailed the schedule

Checking out the spring school system for your team

- we created an SVN branch for each team.
- check it out in any directory you like to work in your machine. You can use eclipse with subclipse as well to check out here (actually that might be smart if you plan to use eclipse)

  - check out from this location: https://codex.cs.bham.ac.uk/svn/nah/cogx/code/schools/css-2010/team-”<team-name>”. team names:
    - blue (Jeremy,...)
    - violet (Richard,...)
    - green (Danijel,...)
    - red (Patric,...)
    - orange (GJ,...)
  - an example would be: svn co https://codex.cs.bham.ac.uk/svn/nah/cogx/code/schools/css-2010/team-orange my-spring-school-system,don't touch other people's SVN! Don't even look at it
  - it'll look like this: schools/css-2010/master
    - the idea is, that you mostly implement in spring-school-implementation, though changes are allows elsewhere (not everywhere though, not in the svn:externals!).
  - This example will create a directory my-spring-school-system, you can choose any other, too

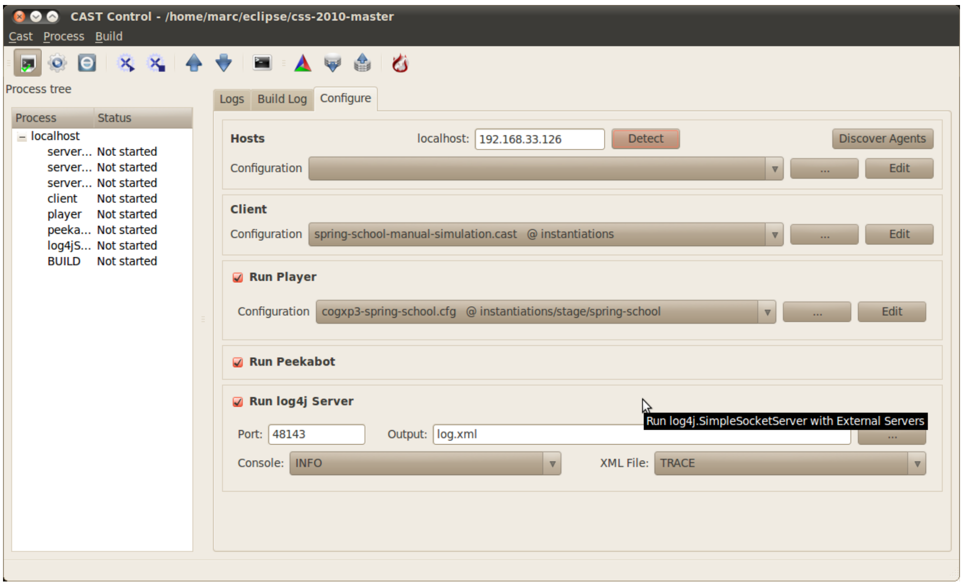
Compile and test it

- change into the directory you just checked out the code
- test the simple simulation environment by running player instantiations/stage/spring-school/cogxp3-spring-school.cfg (you are free to changeinstantiations/stage/spring-school/cogxp3-spring-school.cfg

- compile the spring school system by the following (assuming you are still in your spring school top-level directory):
  1. mkdir BUILD if it does not yet exist
  2. cd BUILD
  3. ccmake .. default could/should be fine here, but you might want to toggle some switches. press [c] twice and then [g] to generate the make files.
  4. make install should build all C++/Python code everything and also run ant to build the java code
- make sure your system has python-qt installed by running sudo apt-get install python-qt4 for castcontrol

Running the first task in simulation

- remove old robot pose: rm -f robotpose.ccf
- copy the map from stored maps (you have to do this prior to every run): cp stored-maps/1sttask-simulation/tmpmap.* .
- run castctrl: tools/castctrl/castcontrol.py
- make sure your settings look like this (your settings will be remembered):

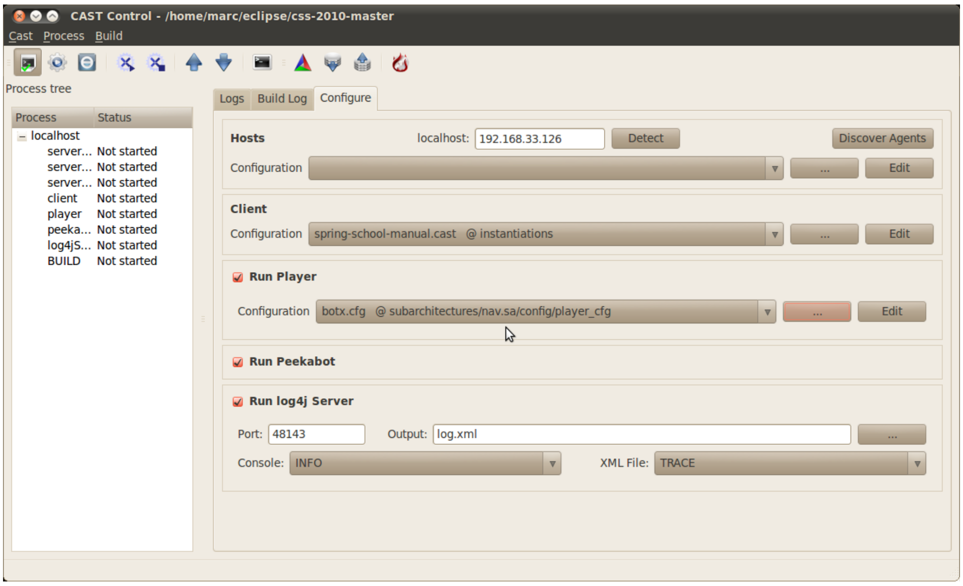


- click [Detect] in the configure tab

- hit [F3] to start peekabot, player, log-server
- hit [F4] to start cast-servers
- hit [F5] to run your configuration (in this case, the simulation of the first task)
- you should see the GraphicalActionInterface where the robot is at your command.
- for the first task we'll use "goto" and "look-for-persons" only
- test it!
- stop the system by hitting [Shift-F5,Shift-F4,Shift-F3]
- Notes
  - if you experience problems with starting peekabot using castcontrol, untick it in castcontrol and run it manually
  - for peekabot you need a link in peekabot's config dir. If you don'thave it yet you might want to create it:
    cd ~
    mkdir -p .peekabot
    cd .peekabot
    rm -rf data
    ln -s <path to your installation of the spring-school-system>/instantiations/peekabot-models/data .

Run it in real

- all you have to do is to change settings and setup your robot:



2.2.3 Taks

First task: Whom do all these records belong to?

There was a party back there in the 80s and everybody brought their own vinyl albums. The next morning they are all spread everywhere and it's about time to clean-up. The robot helps by finding out who is the owner of each of the records?

Task Task: "Find all records in the environment and learn who owns which record"

- the team remote-controls the robot (the team "replaces" planning and beliefmodels)
  - perception: blurred video image & Peekabot visualisation is all you have
  - use the GraphicalExecutionManager.java to execute actions with the robot. The following actions do exist:
    - gotoPlace
    - turnAround
    - askQuestion (by asking them loud, not using the robot's action)

Rules

- each person owns zero or one record (not more)

- each record has an owner
- persons are static at distinct places in the environment (they are too tired from partying to move)
- records are static at distinct places in the environment (they are too object-ish to move)
- humans are at least 2.5m apart from each other
- there are a minimum of two records and a minimum of three persons
- there is a maximum of four records and a maximum of six persons
- the number of records and persons and the pre-loaded map is the same for all teams.
- the starting position is fixed for all teams.
- to position of objects and persons is changed for each run.
- there is hard time constraint of 15 minutes for the task, using this stop watch.
- the robot is remote controlled from a room from which the team members can't neither see the persons nor the objects nor the robot at any time.
- The following questions are allowed and will be answered by humans only if the robot is within communication distance (within 2m, it does not have to face the person). Humans have limited knowledge, for some questions there is a likelihood of knowing the answer. If the person does not know she will either not answer or state she doesn't know. What a person knows or doesn't know is defined prior to each task by throwing dices.

Question	**Answer**	**Likelihood of knowledge**
"Is the record of the artist XXX your record?"	Yes|No	100%
"Can you tell me the artist of the record person YYY brought here, please"	"It is XXX!"	50%
"Is the record of the artist XXX in this room?"	Yes|No	100%
"In which room is the record with artist XXX?"	"It is in room ZZZ!"	50%

"In which room is person YYY?"	"She is room ZZZ!"	50%
"In which room is person YYY's album?"	"It is in ZZZ!"	25%
"What is the artist of your record?"	"It is XXX!"	100% (if this person doesn't have a record she will not answer)
"What is your name?"	"My name is YYY"	100%
"Which room is this?"	"It is the ZZZ!"	100%

- each team should use their own robot
- the goal is to have a representation (on paper) of the following:
    - where is each record (the records are discriminated by their artist and the place by the place-id from peekabot)?
    - where is each person (persons discriminated by their name and the place by the place-id from peekabot)?
    - who is the owner of each record
- rooms: we have three rooms (C, D, corridor)

Scoring

Achievement	Score
each identified and localized person (name<->place association)	100
each identified and localized record (title<->place association)	100
each fully correct assignment ("YYY, your record XXX is at place PPP")	200
asking a question a person cannot answer (logically)	-50
each wrong association (e.g. wrong place, wrong owner, no guessing)	-50

- if task is completely achieved the fastest team wins
- scores are doubled if task is accomplished with self-installed system

Task 2: Play it again!

- This is a first task running autonomously
- It's about finding objects
- It's the last spring school revisited, with the robot's behaviour being generated by binding and planning

Task

- The robot has to find object in the environment autonomously and find out their names
- In the end, the teams have to prove (e.g. by showing a visualisation) that the robot knows
- The goal is to determine the place id each of the objects is in!
- the objects to be found are again records

Rules

- there will be three records randomly distributed across the two rooms: C, corridor.
- the three records will randomly be selected from the set of four records: James, Jesus Jones, Heartbreakers, Chaka Khan.
  - They have to referred to by these identifiers.
- the map is entirely pre-recorded and will be provided:
- The object detectors for the four possible objects have to be trained in advance by the teams themselves. See last year's tutorial.
- the robot has to prove somehow where it found which object. Teams are free to choose how they realize it, e.g. using a component like

  JAVA MG WMViewer castutils.viewer.ViewerCastComponent --subscribe "beliefmodels.autogen.beliefs.StableBelief"

  in your CAST configuration (in instantiations/spring-school.*) is absolutely sufficient!
- all objects will be positioned in a way that the robot can see them.
- the object is assumed to be correctly localized if the shortest walking distance between object and place is less than 2m (generosity applies)
- a hard time constraint of 15 min applies

Scoring

achievement	**score**
each correctly reported place id of an object (incl object name)	100

each falsely reported place of an object	-20
doing it only in simulation	all score divided by 3
robot notices when it has achieved it's goal	100
style points for "cool" result presentation (only if any meaningful result)	between 0-100

Implementation hints

- There is API documentation here: http://www.cs.bham.ac.uk/~hanheidm/spring-school-javadoc/index.html
- decompose the task, so everybody can work on something
- first think how you would model the problem. Define a goal string, and show it to use to get advise!
- you might want to start with the simple goal and test that both in real and simulation
- try to solve the task in simulation, but be aware that in real world, perception is far from perfect

Getting objects on the binder

- in the system it is already implements for places, relations between places, the position of the robot, and for persons
- the object detector whenever triggered puts VisualObject structures in the WM (this data type is defined in subarchitectures/vision.sa/branches/stable-0/src/slice/VisionData.ice). You don't have to change it.
- you need to get this object on the binder as a PerceptBelief? (see schools/css-2010/master/tools/beliefs/src/slice/beliefmodels.ice)

    - you might want to use the percept mediator (look at the example for persons in schools/css-2010/master/subarchitectures/binder/src /java/binder/perceptmediator/components/PersonMediator.java). This is an easy(?) way of implementing it, though it requires you to understand what is going on...
    - you can alternatively create a Java CAST component to do this based on the beliefmodel API (see schools/css-2010/master/tools/beliefs/src/java for the source or check http://www.cs.bham.ac.uk/~hanheidm/spring-school-javadoc/index.html for API documentation. People are advised to look at thetest.beliefmodels.* package in the API

http://www.cs.bham.ac.uk/~hanheidm/spring-school-javadoc/test/beliefmodels/builders/package-summary.html for examples of what to do (and what not) when creating beliefs.

- you can write a C++ component that only uses the CAST-API, but this is not recommended. You have to understand the belief model API before you do this.

- try to understand how it works for Persons and adopt for visualobjects
- look at the output of the WMViewer (look for StableBeliefs?)

Working on planning

- look at the output of the WMViewer (look for StableBeliefs?)
- switch on debugging for the planner
- test your planning outside the system as well

Extending execution

- your central entry point: DoraExecutionMediator?, that has to be modified for the third task.

Task 3: InAct Pablo

- This is the final autonomous task
- It is similar to the first task, but you are now supposed to use belief models and planning to generate the robot's behaviour
- please also refer to the first task's description: meetings/css10/material/task1

Task Task: "Find all records in the environment and learn who owns which record"

- perception: person and object detectors
  - detect that there is person near by
  - detect pre-learned objects (they are identified)
- The following actions do exist:
  - gotoPlace

- detect-person (try to detect a person without actively searching for it)
- detect-object (try to detect any of the pre-trained objects without actively searching for it)
- look-for-object (active search including full rotation)
- look-for-person (active search including full rotation)
- ask-for-<something> (several actions have to be implemented to ask for specific features, a basic GUI is provided)
- confirm-<something> (several actions have to be implemented to confirm hypotheses, a basic GUI is provided)

- the knowledge the robot has to acquire is:
  - in which place are the record? indicating the place-id for each object (referred to by its name)
  - which record belongs to whom? Represented by the record name and the person name
  - in which place is each person? indicating the place-id for each person (referred to by her/his name)
  - in which room is each record? Represented by the record name and the room name
  - in which room is each person? Represented by the person name and the room name

Rules If some rules are unclear, please ask and monitor this page for updates!

General

- no additional constant domain knowledge is allowed (e.g. no augmented maps, no pre-stored information abut places) besides the provided map, the probided rooms, and the trained object detectors. And of course, the domain knowledge in MAPL and MLN.
- each person owns zero or one record (not more)
- each record has an owner
- persons are static at distinct places in the environment (they are too tired from partying to move)
- records are static at distinct places in the environment (they are too object-ish to move)

- humans are at least 2.5m apart from each other (at different places)
- there will be three records randomly distributed across the three rooms: C, D, corridor.
    - Rooms always have to be referred to by these names.
- the three records will randomly be selected from the set of four records: James, Jesus Jones, Heartbreakers, Chaka Khan.
    - They have to referred to by these names.
- the map is entirely pre-recorded and will be provided, as well will be the assignment of places to rooms:
    - every regular place is part of a room, gateway places are not part of any room
    - each room has more than 1 place
- The object detectors for the four possible objects have to be trained in advance by the teams themselves. See last year's tutorial.
- the robot has to prove somehow what it learned. You'll get higher scores the more "natural" you present your results. The robot could use speech output, saying e.g.: "The record Heartbreakers in room C at place 33 belongs to Pierre who is in room corridor at place 2"
- all persons will be positioned in a way that the robot can see them.
- all objects will be positioned in a way that the robot can see them.
- the setup will be the same for all teams, though not previously be announced
- the objects and persons are assumed to be correctly localized if the shortest walking distance between object and place is less than 2m (generosity applies)
- a hard time constraint of 30 min applies
- On Friday we will be strict with the time The order of performing is determined by the ranking in Bowling. The Bowling champion are to choose first when they want to perform.

Rooms

- rooms are represented as ComaRoom in the WM.
  - Coma in CogX is conceptual mapping that does reasoning on concepts about rooms.
  - a room is represented as a set of places that shared the same concept.
  - in this year's school, we use a "Fake-Coma".
- they are already being propagated as PereptBelief by appropriate mediators.
- you have to have a component in your CAST configuration that reads pre-stored rooms from a file. The coma subarchitecture part should be extended to:

```
# coma ########################
INCLUDE includes/coma.sa/coma-base.cast
JAVA MG player castutils.components.WorkingMemoryPlayer  --state
--file stored-maps/3rdtask-real/coma.log
```

- this will load the stored ComaRoom elements and put them on the working memory where other (existing) components mediate these to PerceptBelief.
- in fact, starting up takes about 12 seconds with the room representation in place (due to synchronisation issues).
- in order to get the pre-stored map, of course, you have to update from svn
- There is no provided simulation map for the 3rd task. You can create it yourself. Simply follow these steps:
  - clear the map before startup: rm tmpmap.*
  - add the following to your CAST file (e.g. spring-school-simulation.cast):
    ```
    # coma ########################
    INCLUDE includes/coma.sa/coma-base.cast
    JAVA GD FakeComa fakecoma.components.GraphicalComa --debug
    JAVA MG recorder castutils.components.WorkingMemoryRecorder
    -- file coma-simulation.dump
    ```
  - drag the robot carefully in stage to create a map

- you create rooms by selecting all places in the GraphicalComa window and give it a name: C, corridor, D
- all memory content of the COMA working memory is now stored in a file and can be reloaded later on
- to reload you simple add the reader component to your CAST file (see above for the real system):

```
# coma #########################
INCLUDE includes/coma.sa/coma-base.cast
JAVA MG player castutils.components.WorkingMemoryPlayer
--state --file coma-simulation.dump
```

Asking humans

- The following questions are allowed and will be answered by humans
  - people always answer correct if they have the knowledge
  - people can answer only if the robot is within communication distance (within 2m, it does not have to face the person)
  - Humans have limited knowledge, the likelihood is given in the table. The likelihood is per-value. For instance, if a person doesn't know the room of record A, it might still know the room of record B. Please see attached spread sheet.
  - If the person does not know she will state she doesn't know.
  - If the robot asks a person that is non-existing, the asking action will wail and no information will be reported
  - What a person knows or doesn't know is defined prior to each task by throwing dices.

Question	**Answer**	**Likelihood of answering**
"Is record XXX your record?"	Yes|No	100%
"Is there a record in room ZZZ?"	Yes|No	can be derived
"Is there a person in room ZZZ?"	Yes|No	can be derived
"Is there a person in room ZZZ?"	Yes|No	can be derived
"Which record is owned by person YYY "	"XXX"	50%, if YYY is not you, otherwise 100%
"Who owns record XXX?"	"XXX"	50%, if XXX is not yours, otherwise 100%
"In which room is the record XXX?"	"ZZZ"	50%

"In which room is person YYY?"	"ZZZ"	50%
"In which room is person YYY's album?"	"ZZZ"	25%, if you are not YYY, other wise 50%
"Which one is your record?"	"XXX"	100% (if this person doesn't have a record she will not answer)
"What is your name?"	"YYY"	100%

- you don't have to use/implement all questions
- you can only ask for information in a way that allows the person being asked to unambiguously dereference the question. For example, you must not ask for the owner of a record if you don't know the label. Internal information (such as place ids, belief ids) are not eligible to talk to people. You need to use thenames.
- you may call a restart at any time, max 2 times. the time runs and you have to restart from the maps origin
- you may decide to perform in simulation at any time.
- The end time if a HARD deadline
- people in the test must face towards the robot at all times, the must not make a grimace or anything
- one team member operates the robot (supervised by a referee), this person and all spectators must always keep out of sight for the robot

Scoring

each correctly reported place id of a record (incl record name)	200
each correctly reported room name of a record (incl record name)	100
each correctly reported place id of a person	100
each correctly reported name of a person	100
each correctly reported room name of a person (incl person name)	100
each correct ownership association (incl person name and record name)	200
each mistake in reporting the place of an object (also, duplicates)	-25
each mistake in reporting the place of a person (also, duplicates)	-25

each mistake in reporting the room of an object	-50
each mistake in reporting the room of a person	-50
asking a question a person cannot answer (logically, e.g. incorrectly dereferenced)	-50
doing it only in simulation	all score divided by 4
robot reports achievement of task and finishes	100
every minute finished earlier than 30 minutes	10
style points for "cool" result presentation (only if any meaningful result)	between 0-200

Implementation hints

- start with a simple, brute force attempt and refine
- not all question are necessary useful, some actions need to be implemented, some are more complex than others
- Here are your entry points to start hacking:
  - model your domains for planning here: schools/css-2010/master/spring-school-implementation/domains
  - design your Markov logic networks here: schools/css-2010/master/spring-school-implementation/src/markovlogic

Final Score The ranks of the first task is not included in the final score

Rank	rank score T2	rank score T3
1	10	20
2	6	12
3	3	6
4	1	2
5	0	0

The final score is computed as the sum of the rank score of the individual tasks. The overall score

2.3 Technical tutorials

2.3.1 Tutorial on binding and belief models

See the slides below.

2.3.2 Tutorial on planning

See the slides below.

2.3.3 A short recap

General about CAST and working memories

- CAST is an integration toolkit, tailored to the needs of cognitive systems engineering
- it's kind of a blackboard (basically some key-value maps) architecture with basic operations: ADD, OVERWRITE, and DELETE
- event-driven processing, components register themselves to receive event on ADD, OVERWRITE, and DELETE of content
  - the WMViewer is a simple plugin-based viewer for such kind of memory content
- CAST files define the system structure: schools/css-2010/master/instantiations/spring-school-simulation.cast

The perception side

- OBJECTS: we use the FERNS detector and we simulate object detection using coloured blobs in stage.
- PEOPLE: we use a multimodal people detection that has been developed by a MSc student in Birmingham (again: simulated in stage using coloured blobs, in the standard configuration these are yellow.
- The visual components are configured in schools/css-2010/master/instantiations/includes/vision.sa
- perception is the beginning of our processing chain:perception -> binder -> planner -> execution
  - for sake of simplicity we here also consider the places the robot is at and the robots position as perceptual entities that are updated if required.

- there is a set of perceptmediators schools/css-2010/master/instantiations/includes/binder.sa/perceptmediators.cast
  - these are a generic implementation for the use case of a 1-to-1 mapping of working memory entries to PerceptBelief with specialized transfer functions, e.g. for Persons.
  - look at schools/css-2010/master/subarchitectures/binder/src/java/binder/perceptmediator

The binder

- See tutorial
- in the current system we use belief models as a generic representation of the world (or, more, would the system beliefs the world is like)
- Belief generally have distribition(s) of different types. In this schools, we are only using discrete feature distribution that are independent of eachother.
  - thus, you can think of a belief as a structure that has simply a set of named features of different types:
    - PointerValue: to refer to other beliefs
    - StringValue, IntegerValue, Boolean (see API)
- all perceptual information has to be put to CAST's working memories as PerceptBeliefs.
  - their addition or modification (ADD, OVERWRITE) trigger the Binder to propagate these beliefs in a multi-stage procedure:
    - PerceptBelief
    - PerceptUnionBelief
    - MultimodalBelief
    - TemporalUnion
    - StableBelief
- The binder framework does all the propagation for you and the components that work on it are defined in CAST files again: schools/css-2010/master/instantiations/includes/binder.sa
- While most of the binder components are rather simple Forwarder, the tracker component uses already Markov logic networks to implement the actual tracking. The current implementation is in schools/css-2010/master/spring-school-implementation/src/markovlogic/tracking/tracking-objects.mln, you can change this formulars to make tracking more robust. How?

- in the end, the binder creates StableBeliefs which are the basis for the Planner. You can easily see these StableBeliefs in the WMViewer component.

The planner

- the planner has a generic component that translates StableBeliefs into planner states. You don't have to write this, it's done by the planner framework.
- the planner state (the problem) generated by the planner is created in subarchitectures/planner.sa/src/python/problem*.mapl.
- in this transformation process, the planner uses the following rules:
  - stable beliefs are transformed into objects
  - PointerValues are dereferenced to objects in the planner domain
  - features of Type BooleanValue are transformed into predicates of the object or functions of type Boolean (depending on the domain definition)
  - StringValue and IntegerValue are represented as functions in the planner domain (see systems/spring-school-2010/spring-school -implementation/domains)
  - in fact, Moritz slides (page 25) tell that internally a predicate is only a function of type boolean
- it is very important that the domain file correctly defines the functions, predicates, and objects that are generated... otherwise, the planner will crash (with some useful explanation in the logs!)
- it is highly recommended to generate "problems" by letting the system run and do planner debugging "offline" as explained in the planning tutorial.
- the planner receives its goal from motivation.
  - this general framework does goal generation, filtering and management.
- in our school, we only use one generator, the manual filter, and the goal managers : schools/css-2010/master/subarchitectures/motivation.sa/config/cast-includes/motivation.cast
- for some historic reason, execution is part of motivation

Execution mediators

- create action instances for specific action triggered by the planner
- a mapping between the name of the action in the planner domain and actual implementing datatype has to be established here
- look here for an almost complete example:
schools/css-2010/master/spring-school-implementation/src/java/execution /components/SpringSchoolExecutionMediator.java

Belief modelling & binding: A short introduction

Pierre Lison & Geert-Jan M. Kruijff

Language Technology Lab
DFKI GmbH, Saarbrücken
http://talkingrobots.dfki.de

Deutsches Forschungszentrum für Künstliche Intelligenz
German Research Center for Artificial Intelligence

Outline

Outline

Introduction

- The spring school system for this year includes the *binder* subarchitecture
- The binder is a central hub gathering and processing information about the world, coming from various sources
- This information is represented as **beliefs**
- We've been working over the last months on a new framework for representing and constructing multi-modal beliefs of the environment
- What you are going to use during this spring school is a very early prototype of this framework

What is the binder?

- The binder constructs explicit, spatio-temporally grounded *representations* of the environment
  - based on perceptual inputs retrieved from the various modalities
  - and on information communicated by other agents
- These representations are called **beliefs**, and are expressed in a single, unified formalism
- The binder is more than a mere information repository. The goal is to use the binder as an integrated process for information *fusion*, *refinement*, and *abstraction:*
  - Provides a rich, multi-modal model of the external context
  - Essential for high-level cognitive abilities (planning, interaction, etc.)

A new framework

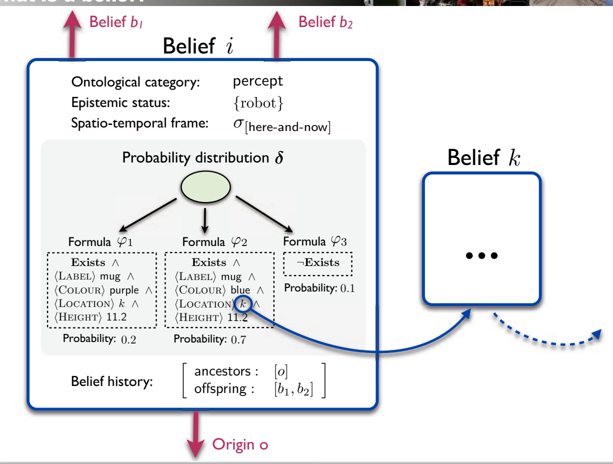
- We recently developed a new theory for "belief model formation" (aka binding) to formalise this process
  - It is based on *Markov Logic*, a first-order probabilistic language (combination of graphical models and first-order logic)
  - Why? Need to capture both the rich *relational structure* of the environment and the *uncertainty* of our observations
  - Internal engine for probabilistic inference is based on an existing software package for Markov Logic: *Alchemy*
- See our WP1 extended report for the theoretical foundations of our work
- This theory is partly implemented in the system you will be using for the spring school task

Outline

Representation of beliefs

- The environment we need to model is simultaneously *complex*, *multi-agent*, *dynamic* and *uncertain*
  - We need a representation which can handle this!
- In the “new binder”, every unit of information is a **belief**
  - Beliefs are constrained both spatio-temporally and epistemically
  - Their content is expressed as a probability distribution over alternative values, which can be expressed as features or propositional formulae
  - Beliefs can be linked with each other to capture the relational structure
- The initial beliefs are low-level, and are gradually fused, refined and abstracted by the binder system
  - The final outcome is a collection of final, stable beliefs

What is a belief?



Constructing and updating beliefs

- Beliefs are inserted by the local subarchitectures into the binding working memory
- Belief representing sensory inputs observed from perceptual modalities are called *percepts*
- The insertion of percepts into the binder can be done directly (via CAST), or via percept mediators
  - Percept mediators provide an intermediary layer of generic processes to create percept beliefs
  - Useful for synchronization
- Of course, percepts can also be updated (overwrite operation) or deleted at any time

Belief content

- The belief content is expressed as probability distribution over alternative values
- It is typically instantiated via an global existence probability for the entity combined with a set of specific features
  - Each feature is also expressed as a probability distribution
  - The features are usually assumed to be conditionally independent of each other
- Example: a belief about a red ball
  - P(Exists) = 0.85
  - P(type=ball | Exists) =0.9 , P(type=mug | Exists) = 0.05
  - P(colour=red | Exists) = 0.8, P(colour=pink | Exists) = 0.1
- Builders are available in `tools/beliefs` for easily constructing beliefs

Example code in Java

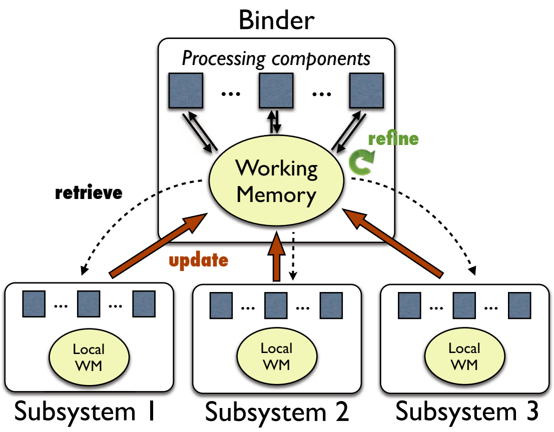
```
CondIndependentDistribs features = BeliefContentBuilder.createNewCondIndependentDistribs();

List<FeatureValueProbPair> labelPairs = new LinkedList<FeatureValueProbPair>();
labelPairs.add(new FeatureValueProbPair(FeatureValueBuilder.createNewStringValue("Mug"), 0.9f));
labelPairs.add(new FeatureValueProbPair(FeatureValueBuilder.createNewStringValue("Ball"), 0.05f));

BasicProbDistribution labelDistrib =
    BeliefContentBuilder.createNewFeatureDistribution("label", labelPairs);
BeliefContentBuilder.putNewCondIndependentDistrib(features, labelDistrib);

ProbDistribution beliefcontent =
    BeliefContentBuilder.createNewDistributionWithExistDep(0.85f, features);

CASTBeliefHistory hist =
    PerceptBuilder.createNewPerceptHistory(new WorkingMemoryAddress ("local_subarch_id","vision"));

PerceptBelief belief =
    PerceptBuilder.createNewPerceptBelief(id, "object", "here", this.getCASTTime(), beliefcontent, hist);

insertBeliefInWM(belief)
```

More practical infos

- The code for constructing beliefs is to be found in `tools/beliefs`
- The representation is specified in a slice file: `tools/beliefs/src/slice/beliefmodels.ice`
- *JUnit* tests are also available (in `tools/beliefs/src/java/test`) and provide you with plenty of example code for constructing beliefs
- Once a belief is constructed, it can be inserted/updated/deleted in the working memory using the usual CAST operations, or the `BeliefWriter` interface

Outline

The binder subarchitecture

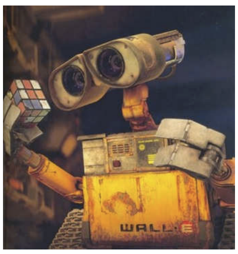


Binding operations

- The process of constructing high-level representations from perceptual inputs is formalised in four steps:
  - **Perceptual grouping**: beliefs from distinct modalities which may pertain to the same entity are grouped together
  - **Multi-modal estimation**: the measurements from the uni-modal beliefs are used to derive multi-modal estimates
  - **Tracking**: beliefs are unified over time (by relating current observations of an entity to past observations)
  - **Temporal smoothing**: the estimates in the beliefs are refined using both past and present measurements
- For the spring school task, only the *tracking* step will be directly useful

Good luck!

- Good luck in your work!
- And have fun playing with the system :-)
- If you have any problems or questions regarding the binder implementation, please don't hesitate to ask me



Planning Tutorial
CogX Spring School 2010

Moritz Göbelbecker

26. April 2010

You are going to learn about:

- Planning basics
- PDDL and MAPL
- The CogX planner implementation
- Interactions between Planner and Binder

The Planning Problem

Definition

Given an initial state and a goal formula, find a sequence of actions that leads to a state which satisfies the goal.

Classical Planning

- Deterministic
- Fully observable
- Propositional statements
- Closed world assumption
  - That which isn't known to be true is false
  - The set of objects is fixed
- No numeric rescources
- No time

Object Fluents

- Multi-valued state variables
- Introduced in PDDL 3.1
- In addition to propositional statements
- Fluents that have no known value are explicitely unknown

Continual Planning

- Interleave planning and execution
- Monitor the plan execution
- When the plan is no longer valid: replan

PDDL

- A standardised language to describe planning problems
- Used by the International Planinng Competition
- Separate domain and problem descriptions.

Domain description

```
(define (domain cogx)
    (:requirements ...)
    (:types ...)
    (:constants ...)
    (:predicates ...)
    (:functions ...)

    (:action ...)
    (:action ...)
    (:action ...)
)
```


Requirements

```
(define (domain cogx)

    (:requirements :mapl :adl :object-fluents)

    (:types ...)
```


Type definition

```
    (:requirements ...)

    (:types
        place - object
        place_status place_name - object
        robot - agent
        robot person - movable
    )

    (:constants ...)
```


Domain constants

```
    (:types ...)

    (:constants
        placeholder trueplace - place_status
    )

    (:predicates ...)
```

Predicates

```
(:constants ...)

(:predicates
    (connected ?n1 - place ?n2 - place)
    (occupied ?p - place)
)

(:functions ...)
```


Introduction
Planning
Planner.SA
Tasks
PDDL
MAPL

Functions

```
(:predicates ...)

(:functions
    (is-in ?r - movable) - place
    (placename ?n - place) - place_name
    (placestatus ?n - place) - place_status
)
```

Moritz Göbelbecker Planning Tutorial

Introduction
Planning
Planner.SA
Tasks
PDDL
MAPL

Action description

```
(:action move
    :parameters (?a - robot ?to ?from - place)
    :precondition (and
        (= (is-in ?a) ?from)
        (connected ?from ?to)
        (not (occupied ?to)))
    :effect (and
        (assign (is-in ?a) ?to))
)
```

Moritz Göbelbecker Planning Tutorial

Introduction
Planning
Planner.SA
Tasks
PDDL
MAPL

Conditions

- Atoms `(p ?args), (= (f ?arg) ?value)`
- Negations `(not (Condition))`
- Conjunctions `(and A B C ...)`
- Disjunctions `(or D E F ...)`
- Implications `(imply A B)`
- Existential Quantifier `(exists (?x - type) (p ?x))`
- Universal Quantifier `(forall (?x - type) (p ?x))`

Moritz Göbelbecker Planning Tutorial

Introduction
Planning
Planner.SA
Tasks
PDDL
MAPL

Effects

- Atoms `(p ?args)`
- Negations `(not (Atom))`
- Assignments `(assign (f ?arg) ?value)`
- Conjunctions `(and A B C ...)`
- Universal effects `(forall (?x - type) (Effect))`
- Conditional effects `(when (Condition) (Effect))`

Moritz Göbelbecker Planning Tutorial

Introduction
Planning
Planner.SA
Tasks
PDDL
MAPL

Action costs

```
(:requirements :action-costs ...)

(:action move
    :parameters (?a - robot ?to ?from - place)
    :precondition (and
        (= (is-in ?a) ?from)
        (connected ?from ?to)
        (not (occupied ?to)))
    :effect (and
        (assign (is-in ?a) ?to)
        (increase (total-cost) 42))
)
```

Moritz Göbelbecker Planning Tutorial

Problem description

```
(define (problem cogx-problem)
    (:domain cogx)

    (:objects ...)
    (:init ...)
    (:goal ...)
    (:metric ...)
)
```


Objects

```
(:domain cogx)

(:objects dora - robot
          p1 p2 p3 p4 - place)

(:init ...)
```


Initial state

```
(:objects ...)

(:init
    (= (is-in dora) p1)
    (connected p1 p2)
    (connected p2 p3)
    (connected p3 p4)
)

(:goal ...)
```


Goal formula

```
(:init ...)

(:goal (and
    (= (is-in dora) p4))
)

(:metric ...)
```


Optimization function

```
(:goal ...)

(:metric minimize (total-costs))
```


MAPL

- MultiAgent Planning Language
- Extension of PDDL
- Support for continual planning
- Support for knowledge representation

MAPL Types

- `boolean`
  - Predefined constants `true`, `false`
  - Internally, all predicates are functions of type `boolean`
- `agent`
  - Represents entities that can execute actions.

Actions in MAPL

```
(:action move
    :agent(?a - robot)
    :parameters (?to - place)
    :variables (?from - place)
    :precondition (and
        (= (is-in ?a) ?from)
        (connected ?from ?to)
        (not (occupied ?to)))
    :effect (and
        (assign (is-in ?a) ?to))
)
```


Modal predicates

- Allow the expression of beliefs about facts.
- `(kval ?a (variable))`: ?a knows the value of *variable*.
- `(in-domain (variable) ?value)`: *variable* can possibly take the value ?value.

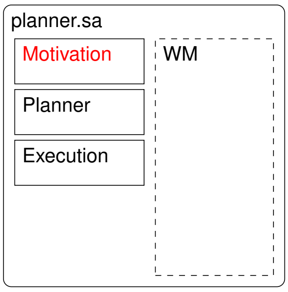
Assertions

- Assertions allow us to decide when to replan
- Replan-Condition: subset of the precondition that triggers replanning when satisfied.
- Assertions will never be executed.

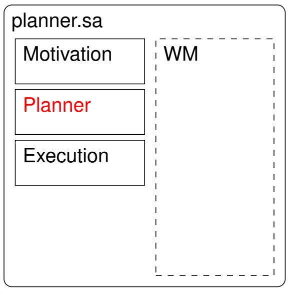
Assertions

```
(:action assertion_pick_up
    :agent(?a - robot)
    :parameters (?o - thing)
    :variables (?p - place)
    :precondition (and
        (= (is-in ?a) ?p)
        (in-domain (is-in ?o) ?p))
    :replan (kval ?a (is-in ?o))
    :effect (and
        (assign (is-in ?o) ?a))
)
```

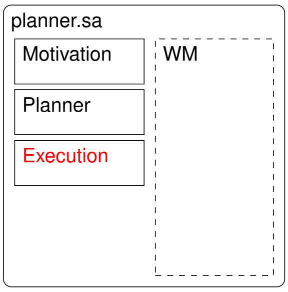
Components



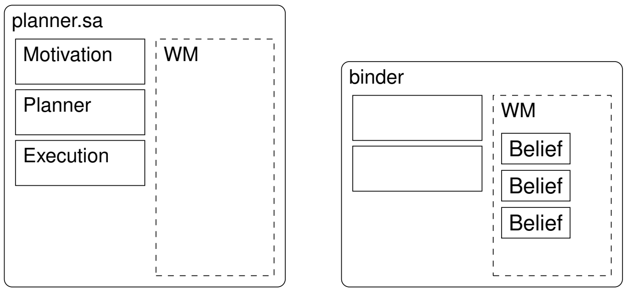
Components



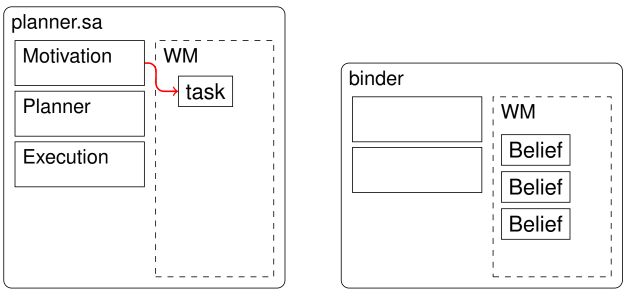
Components



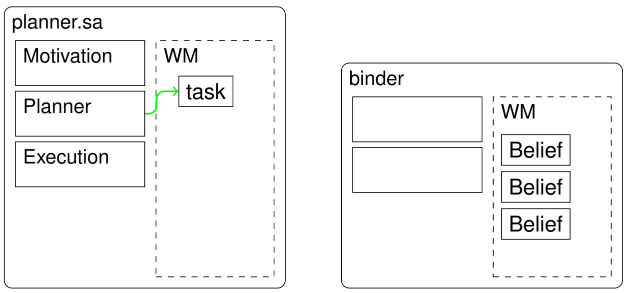
Motivation



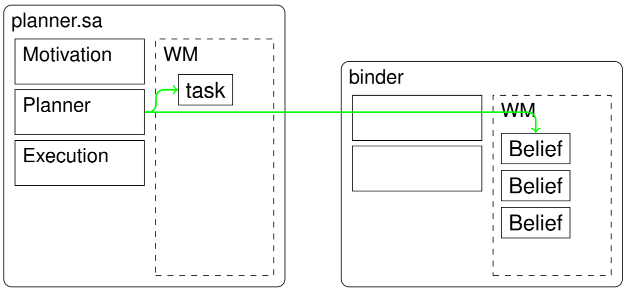
Motivation



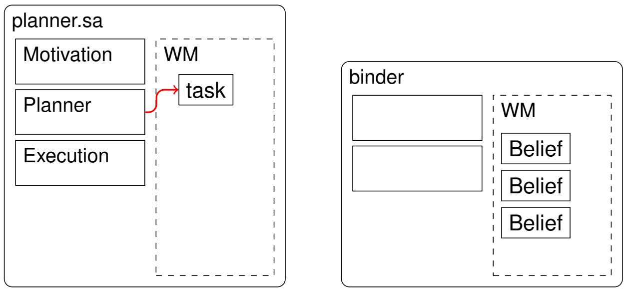
Planner



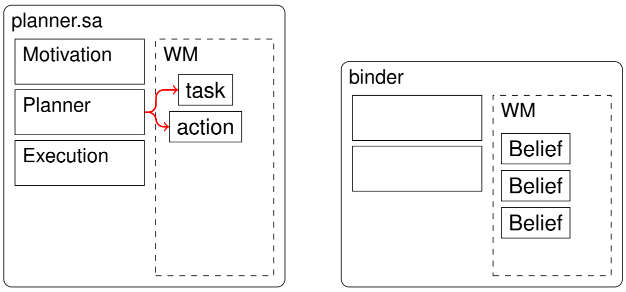
Planner



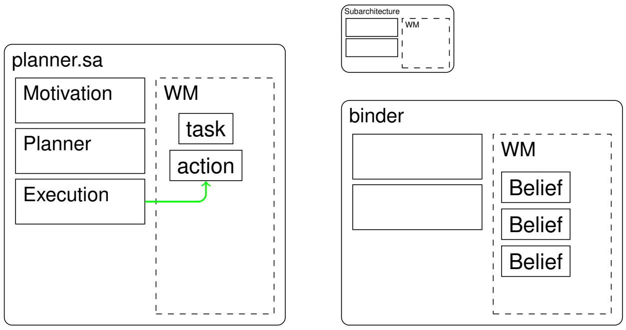
Planner



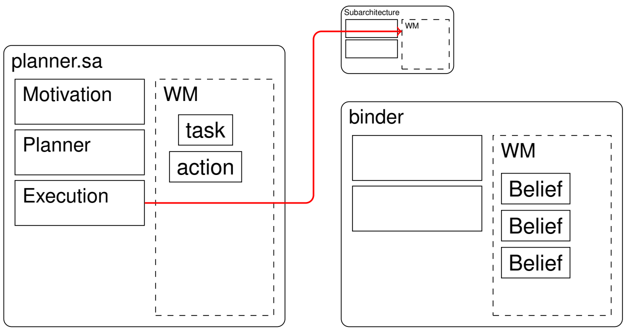
Planner



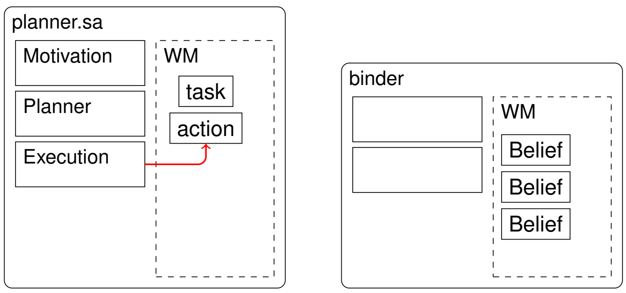
Execution



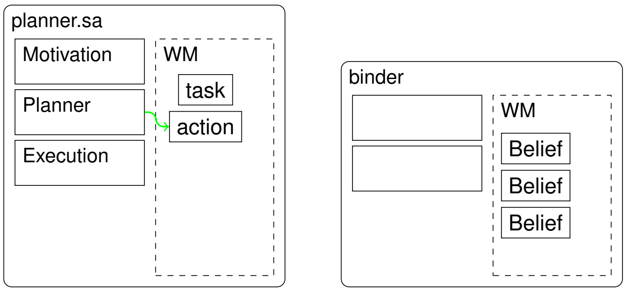
Execution



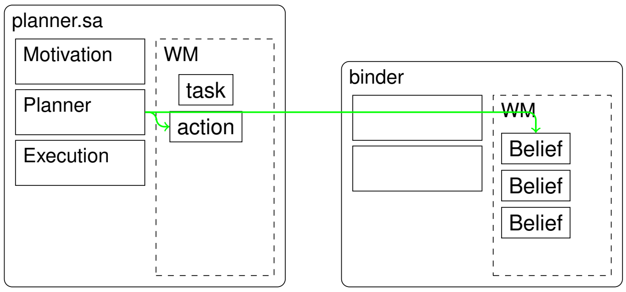
Monitoring



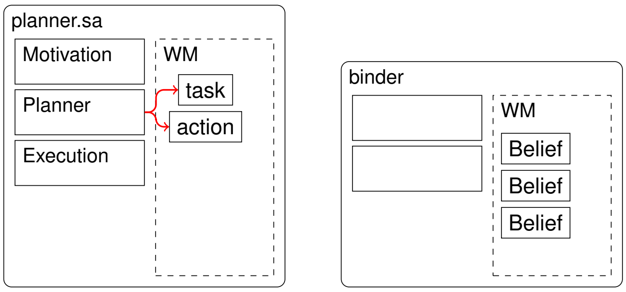
Monitoring



Monitoring



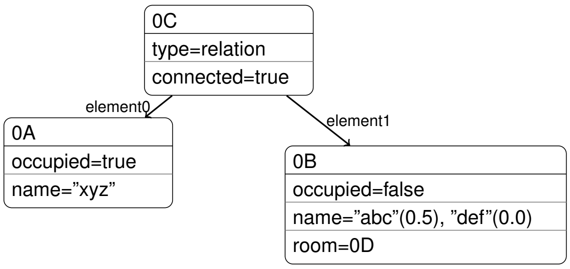
Monitoring



State Generation

World state on the Binder:

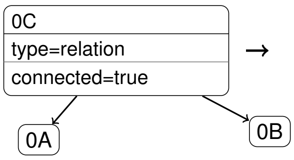
- Beliefs representing objects with features
- Beliefs representing relations between objects



State Generation

0A
occupied=true
name="xyz"
placestatus="trueplace"

→ (occupied place_0A)
(= (name place_0A) place_name_xyz)
(= (placestatus place_0A) trueplace)



→ (connected place_0A place_0B)

State Generation

0A
name="xyz" (1.0)

→ (= (name place_0A) place_name_xyz)

0B
name="xyz" (0.5)
name="abc" (0.5)

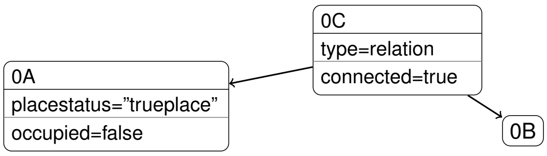
→ (in-domain (name place_0B) place_name_xyz)
(in-domain (name place_0B) place_name_abc)

0C
name="xyz" (0.0)
name="abc" (0.0)

→ (not (in-domain (name place_0C) place_name_xyz))
(not (in-domain (name place_0C) place_name_abc))

Type Inference

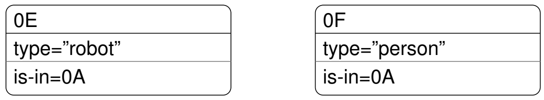
- No hardcoded type information in Beliefs
- Types are determined by the features they have or occur in.



(connected ?n1 - place ?n2 - place) - boolean
(placestatus ?n - place) - place_status
(occupied ?n - place) - boolean

Type Inference

- *type*-member can be used as additional type hint



robot person - movable

(is-in ?m - moveable) - place

Feeding back information to the binder

- Sometimes you might want to keep track of success/failure of previous actions.
- Change the execution modules to put updates on the binder.
- Would be nice to have this in the planning domain.

Update Effects

```
(:action move
  :agent(?a - robot)
  :parameters (?to - place)
  :variables (?from - place)
  :precondition (and
      (= (is-in ?a) ?from)
      (connected ?from ?to)
      (not (occupied ?to)))
  :effect (and
      (assign (is-in ?a) ?to)
      (update (occupied ?from) false)
      (update-failed (occupied ?to) true))
)
```


Make the robot move

- Action: `(move ?a - robot ?to - place)`
- Goal: “The robot is at place 2”
- What to do if you can't predict belief ids?

Visit everything

- Keep track of what you visited.

Asking questions

- Action: `(ask-for-placename ?a - robot ?p - belief)`
- Action: `(verify-placename ?a - robot ?p - belief ?val - value)`
- Will ask about the “name” feature of a belief.
- Find out the names of all places.

Planning Tutorial - Addendum

CogX Spring School 2010

Moritz Göbelbecker

28. April 2010

Planner options

- Fast Downward can use lots of combinations of heuristics
- Unfortunately, most don't work with axioms (used internally)
- of seems to be a good value
- Set in `subarchitectures/planner.sa/src/python/standalone/config.ini`

Axioms

- Axioms allow the definition of predicates that depend on other predicates.
- Derived predicates may be set in the initial state but not in action effects.

Example

```
(:derived (connected ?loc ?loc2 - place)
              (connected ?loc2 ?loc))
```



```
(forall (?x - type) (or (a ?x)
                        (b ?x)))
```

- Will cause an exponential blowup in grounded axiom size.
- Axioms to the rescue:

```
(:derived (c ?x - type) (a ?x))
(:derived (c ?x - type) (b ?x))

(forall (?x - type) (c ?x))
```

2.4 Invited tutorials

2.4.1 David Hogg: Activity analysis: representation and learning

See the slides below.

2.4.2 Norbert Kruger: Early Cognitive Vision: Vision for Cognition

See the slides below.

2.4.3 Ron Petrick: Representations for classical and knowledge-level planning

See the slides below.

Activity analysis: representation and learning

David Hogg
University of Leeds

CogX meeting, April 2010

Introduction

Representing what is possible
Learning about what is possible
Dealing with visual uncertainty

Representing what is possible
Learning about what is possible
Dealing with visual uncertainty

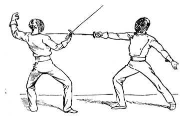
Representing activities

Instantaneous configuration:

$$\mathbf{x} = (a_1, a_2 \cdots a_n)$$

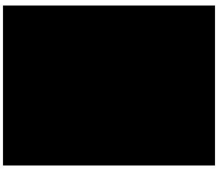
Joint angles
Pixel values

Joint configuration
$\mathbf{x} = (\mathbf{x}^L, \mathbf{x}^R)$

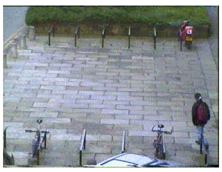


Representing activities

Time-series of configurations $(\mathbf{x}_1, \mathbf{x}_2 \cdots \mathbf{x}_t)$

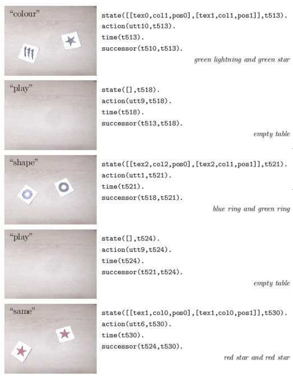


(x,y) position in image



*(x,y,**s**) position & shape in image*

Representing activities



Time-series of sets of logical atoms

state([[tex2,col2,pos0],[tex2,col1,pos1]],t521).
action(utt1,t521).
time(t521).
successor(t518,t521).

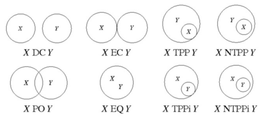
Needham et al., AIJ 2005

Sequence of propositions (adjacencies of person to location)

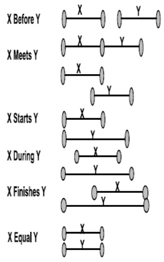
(1 3 8 2 10 6)

Hamid et al., AIJ 2009

Qualitative spatial and temporal relations

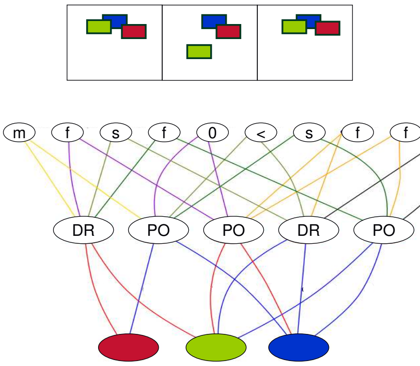


Region Connection Calculus – RCC8
(Randell et al., ICKRR 1992),



Allen's temporal relations
(*Allen, CACM 1983*)

Representing spatial-temporal configuration as a graph



Temporal relations

Spatial relations

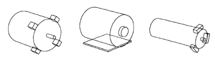
Objects

Sridhar et al., AAAI 2010

Representing object categories

from Heinrich Bulthoff

As a subset of all configurations X, defined by algebraic constraints (Brooks, AIJ 1981)



17.0 < MOTOR-LENGTH x MOTOR-RADIUS < 21.0

As a probability function p(X) over configurations

Representing event categories

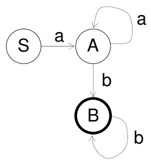
As a subset of all configuration sequences

String grammar, Chomsky, 1956

Regular Grammar

S => aA
A => aA
A => bB
B => b

Equivalent to a Finite State Machine



Context-free grammar (CFG)

Allows for 'subroutines' (e.g. raising arm)

S => aAb
A => aB
B => ab

Probabilistic models for event categories

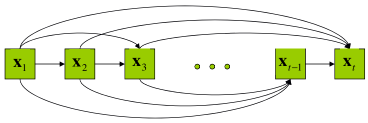
As a probability function over configuration sequences

$$p(\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_t)$$

This probability function defines a *stochastic process*

Factorisation of joint distribution

$$p(\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_t) = p(\mathbf{x}_1)\, p(\mathbf{x}_2 \mid \mathbf{x}_1)\, p(\mathbf{x}_3 \mid \mathbf{x}_1, \mathbf{x}_2) \ldots p(\mathbf{x}_t \mid \mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_{t-1})$$



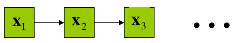
Represent as a dynamic Bayesian network

Simplification/approximation

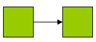
Reduce to a tractable form by simplifying terms where possible, for example

$$p(\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_t) = p(\mathbf{x}_1)\, p(\mathbf{x}_2 \mid \mathbf{x}_1)\, p(\mathbf{x}_3 \mid \mathbf{x}_2) \cdots p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$$

Known as a 1st order Markov process



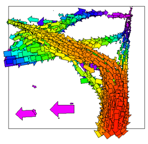
By convention, show one 'cycle' only and omit the variables



A Markov model for pedestrian motion

K quantised configurations of $(x, y, \dot{x}, \dot{y})$

1st order Markov chain

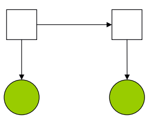


Anomaly detection*

Prediction*

* From a similar model

Hidden Markov models (HMMs)



$$p(s_1 \ldots s_n, \mathbf{x}_1 \ldots \mathbf{x}_n) = p(s_1)\, p(\mathbf{x}_1 \mid s_1) \prod_{t=2}^{n} p(s_t \mid s_{t-1})\, p(\mathbf{x}_t \mid s_t)$$

$$p(\mathbf{x}_1 \ldots \mathbf{x}_n) = \sum_{\substack{(s_1 \ldots s_n) \in \text{ all} \\ \text{state sequences}}} \left(p(s_1)\, p(\mathbf{x}_1 \mid s_1) \prod_{t=2}^{n} p(s_t \mid s_{t-1})\, p(\mathbf{x}_t \mid s_t) \right)$$

Stochastic grammar

Stochastic Context-Free grammar

TRACK	→		CAR-TRACK	[0.5]
		|	PERSON-TRACK	[0.5]
CAR-TRACK	→		CAR-THROUGH	[0.25]
		|	CAR-PICKUP	[0.25]
		|	CAR-OUT	[0.25]
		|	CAR-DROP	[0.25]
CAR-PICKUP	→		ENTER-CAR-B CAR-STOP PERSON-LOST B-CAR-EXIT	[1.0]
ENTER-CAR-B	→		CAR-ENTER	[0.5]
		|	CAR-ENTER CAR-HIDDEN	[0.5]
CAR-HIDDEN	→		CAR-LOST CAR-FOUND	[0.5]
		|	CAR-LOST CAR-FOUND CAR-HIDDEN	[0.5]
B-CAR-EXIT	→		CAR-EXIT	[0.5]
		|	CAR-HIDDEN CAR-EXIT	[0.5]
CAR-EXIT	→		car-exit	[0.7]
		|	SKIP car-exit	[0.3]
CAR-LOST	→		car-lost	[0.7]
		|	SKIP car-lost	[0.3]
CAR-STOP	→		car-stop	[0.7]
		|	SKIP car-stop	[0.3]
PERSON-LOST	→		person-lost	[0.7]
		|	SKIP person-lost	[0.3]

Probabilities for alternative rewrites

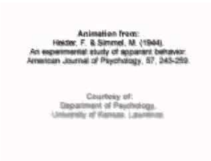
Ivanov and Bobick, PAMI 2000

Stochastic regular grammar equivalent to a Markov chain

Behaviour as plan execution

Human activities result from agents executing goal-directed plans

People can't help perceiving motion in this way



Heider & Simmel, 1944

Long history of work in AI on planning:

- Hierarchical planning
- Handling uncertainty
  - non-deterministic plan decomposition into sub-plans
  - non-determinism in outcomes from different actions
  - uncertainty in the observation of these outcomes
- Plan recognition by probabilistic inference over the stochastic process modelling execution of an actor's plans

General model for probabilistic hierarchical planning

- Abstract Markov policies (AMP), Sutton, Precup and Singh, 1999
- Abstract Hidden Markov Models (AHMM), Bui, Venkatesh and West JAIR 2002

Abstract Hidden Markov Models
Bui, Venkatesh and West, JAIR 2002

Possible states of the world S

Possible actions A

In state s, action a results in state s' with probability $\sigma_a(s, s')$

'Local' policy $\pi = (S_\pi, D_\pi, \beta_\pi, \sigma_\pi)$

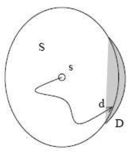
applicable states

destination states

probability of stopping for each destination state

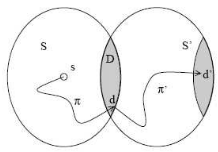
probability of performing action a in state s

$\sigma_\pi : S_\pi \times A \rightarrow [0,1]$

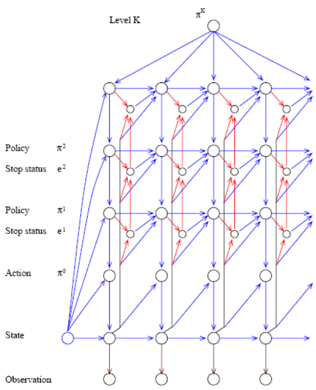


Abstract policy $\pi^* = (S_{\pi^*}, D_{\pi^*}, \beta_{\pi^*}, \sigma_{\pi^*})$ over a set of abstract policies Π

Like a local policy, except actions replaced by policies from Π

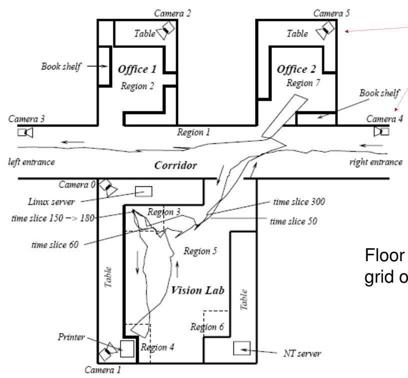


$\sigma_{\pi^*} : S_{\pi^*} \times \Pi \rightarrow [0,1]$



From Bui et al., JAIR 2002

An example



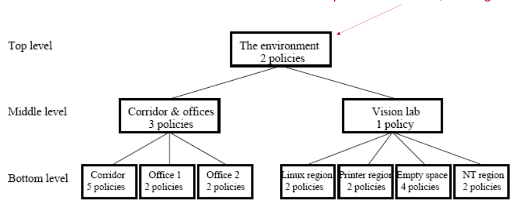
6 cameras

Floor divided into a grid of cells (state)

From Bui et al., JAIR 2002

Region (state space) and policy hierarchy

2 policies: ‘exit left’, ‘exit right’

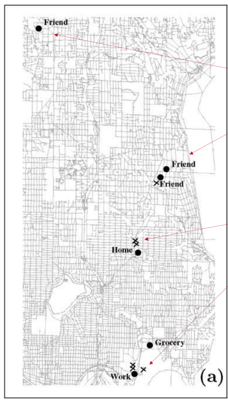


2 policies: ‘using’, ‘passing through’

From Bui et al., JAIR 2002

Plan execution for transportation

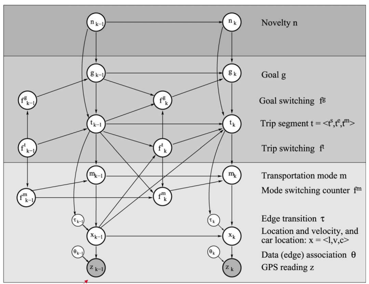
Liao, Patterson, Fox, and Kautz, AIJ 2007



goals

trip switching locations

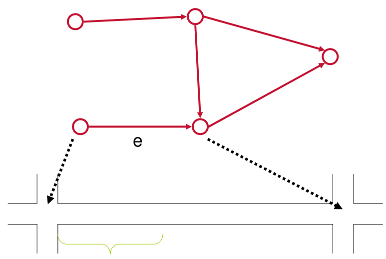
From Liao et al., AIJ, 2007



From Liao et al., AIJ, 2007

Observations from GPS

Represent streets and junctions with a directed graph G=(V,E)



$x_k = (l_k, v_k, c_k)$

$l_k =$ person location

$v_k =$ person velocity

$c_k =$ car location

Person location given by road segment (edge e) and distance from junction d

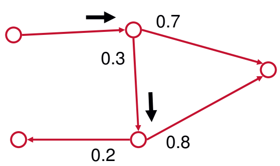
GPS observation

Sensor model: $p(z_k \mid l_k)$ given by a Gaussian density function

Motion model: $p(l_k \mid l_{k-1}, v_k, \tau_k)$ given by a Kalman filter

Transportation mode: BUS, FOOT, CAR, BUILDING
- determines Gaussian velocity distribution of person
- c_k only changes in CAR mode

Trip segment: (start location, end location, transportation mode)
- determines transition probabilities at junctions (nodes)



Goals, e.g. friend's home, workplace, grocery store
- determines transition probabilities at end of trip segments

Goal sequence determined by transition probabilities:

		New goal		
		g1	g2	g3
Current goal	g1	0	0.2	0.8
	g2	0.6	0	0.4
	g3	0.1	0.9	0

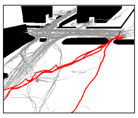
Novelty is TRUE or FALSE: when true ignore goal and trip segment levels

Policy (goal) recognition using probabilistic inference over the Bayesian network (*Rao-Blackwellized filter*)

Experiment
- 60 days of GPS from one person
- 30 days used for learning parameters of the model (e.g. transition probabilities for goal segments and trip segments)
- 30 days for testing

Plan execution using generic strategies

Problems with plan execution over a fixed network:
- unfamiliar scenes
- changes in scene layout
- rare events

Need a general planning strategy.

Anomaly detection via plan recognition

Ranking of path planning strategies (Golledge, 1995)
1. Shortest distance
2. Least time
3. Fewest turns
4. Most scenic/aesthetic
5. First noticed
6. Longest leg first
7. Many curves
8. Many turns
9. Different from previous (novelty)
10. Shortest leg first

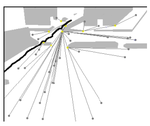
A person's path is anomalous if not explainable as the execution of a goal-directed plan: following the shortest route to a known exit

Hannah Dee, VS 2006

Method

Learning phase
- Locate obstacles by hand (e.g. hedges, buildings)
- Track cars/people and locate 'exits' (e.g. doorways, stationary cars)


Monitoring phase
- Track cars/people and for each:
  - generate shortest paths from entry-point towards all known exits
  - score explicability of actual path by comparison to the closest of these

Results

Compare with the performance of people doing a similar task
- "If you were a security guard, would you regard the behaviour of the agent highlighted in this video as interesting? Please indicate on the following questionnaire, with 1 being uninteresting and 5 being interesting"

Car park dataset, with 269 people/car movements

High rank correlation between automatic explicability scores and the mean human interest scores

Representing what is possible
Learning about what is possible
Dealing with visual uncertainty

Learning about activities

What can be learnt by passive visual observation?

Do we need other sensory modalities, active exploration, tutoring?

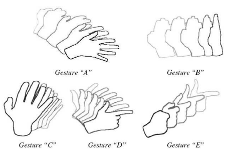
Learning about activities

Two broad approaches:

(1) Objects detected and tracked; activities derived from the resulting configuration space (trajectory, moments etc.)

(2) Activities derived directly from pixel-based 'configurations' (e.g. histograms of salient motion-features, flow)

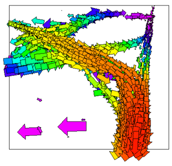
With labelled examples



from *Al-Rajab et al., AMDO 2008*

and without…

Learning activity classes without labels



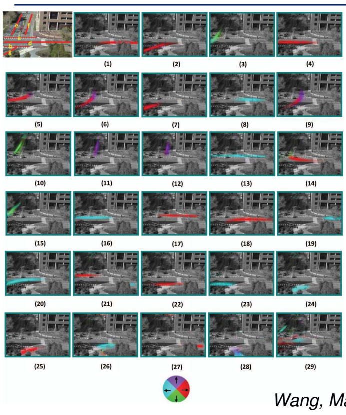
Detect moving objects

Visual words: Quantised position and velocity of these objects

Atomic activities (person trajectories): co-occurring visual words

Johnson & Hogg, IVC 1996

Learning (layered) activity classes without labels and without objects



Visual words: Quantised position and flow direction of 'changed pixels'

Atomic activities: Co-occurring visual words (in short clips)

Interactions: Co-occurring atomic activities (in short clips)

Wang, Ma and Grimson, TPAMI 31(3) 2009

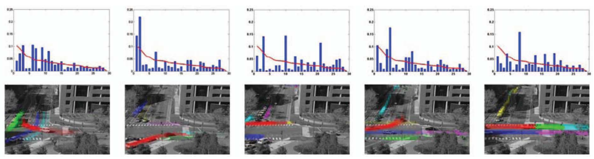
Interactions

Mixture over 29 atomic activities for each interaction.

Instances of each discovered interaction:

colours distinguish between atomic activities

red curve is the average mixture over whole corpus



Learning activity classes without labels and without objects

By modelling the motion in the vicinity of each pixel

Boiman & Irani, ICCV 2005

Discovering event classes from *activity graphs*

Aircraft turnaround domain
(EU project Co-Friend).

temporal relations
spatial relations
objects

3-layer activity graph

Set of event classes C
Set of event instances G

Sridhar et al., AAAI 2010

Assume a generative model

Generative model for activity domains and instances:

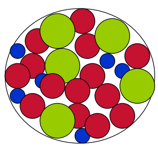
$$p(C,G)$$

set of event classes — set of event instances

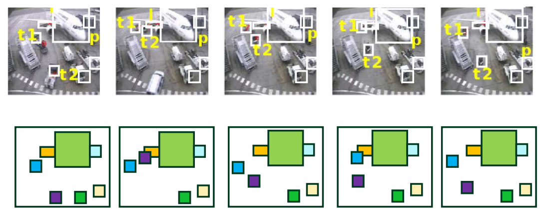
$$\arg\max_{C,G}(p(C,G)) = \arg\max_{C,G}(p(G \mid C, AG)\,p(C))$$

$P(C)$ favours a small number of classes, each generating similar and complex events

$P(G \mid C, AG)$ discourages sharing of objects between events



Typical event mined from multiple turnarounds



More events

Inducing a functional object taxonomy

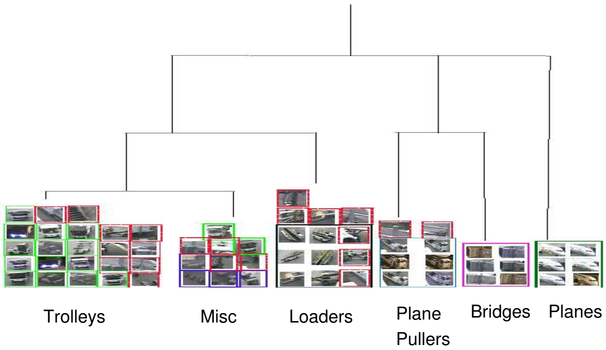
Form Boolean matrix of the role played by objects in each event class
(+ partially generalised classes)

Event classes

Objects	E_1			E_2		$\cdots$	E_m			
o_1	0	1	0	0	0		0	0	1	0
o_2	1	0	0	0	1		0	0	0	0
$\vdots$										
o_n	1	0	0	0	0		0	0	0	0

Compress the rows (pattern for each object) using PCA
Obtain object taxonomy by hierarchical-clustering of the compressed rows

Emergent object categories from aircraft turnaround domain

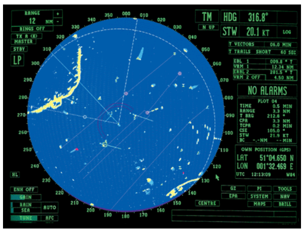


Representing what is possible
Learning about what is possible
Dealing with visual uncertainty

Radar tracking

Dealing with

- missed detections
- spurious detections



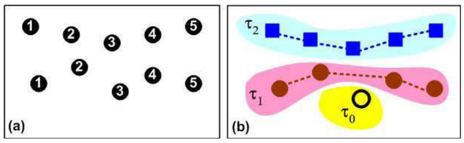
Long history from radar literature and elsewhere:

Ingemar Cox, A Review of Statistical Data Association Techniques for Motion Correspondence, International Journal of Computer Vision, vol. 10, pp. 53-66, 1993.

Standard approach

Find the optimal global explanation:

Given a set of noisy observations Y over a period of time.
An explanation is a partition of these observations $\omega = \{\tau_0, \tau_1, \ldots \tau_K\}$ where each part defines a track and τ_0 contains all spurious observations (false alarms)



Seek $\underset{\omega\in\Omega}{\operatorname{argmax}}(p(\omega | Y))$

Formulation from Oh, Russell and Sastry, CDC-04

Defining $p(\omega | Y)$

Assumptions:

(1) each track behaves as a stochastic linear system:

$$x_{t_{i+1}} = Ax_{t_i} + \eta$$
$$y_{t_i} = Cx_{t_i} + \upsilon$$

(note that matrix A and noise term scaled according to the width of interval

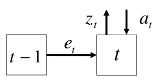
(2) new objects and false alarms occur as Poisson processes

(3) objects disappear and are undetected with fixed probability at each time-step

---

For a given ω at time-step t, assume:

e_t objects persist from t-1
a_t new objects appear
z_t objects disappear
d_t objects detected
f_t false alarms
$u_t = e_t - z_t + a_t - d_t$ objects undetected

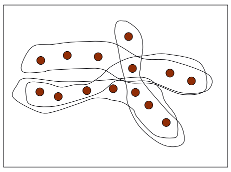


$$P(\omega | Y) = \frac{1}{Z}\prod_{t=1}^{T} \underbrace{p_Z^{z_t}(1-p_z)^{e_t-z_t}}_{\text{track terminations}} \underbrace{p_d^{d_t}(1-p_d)^{u_t}}_{\text{missing observations}} \underbrace{\frac{\lambda_b^{a_t}}{a_t!}\frac{\lambda_f^{f_t}}{f_t!}}_{\text{new objects and false alarms}} \underbrace{\prod_{\tau\in\omega\setminus\{\tau_0\}}\prod_{i=1}^{|\tau|-1} \mathrm{N}(\tau(t_{i+1}) \mid C\bar{x}_{t_{i+1}}(\tau), B_{t_{i+1}}(\tau))}_{\text{stochastic linear system}}$$

Integer Programming

Morefield, IEEE-TAC 1977

- Create a large set of feasible tracks F (a covering), many of which will be inconsistent with one another.



- Seek the optimal partition from a subset of these tracks + false alarms

$$\underset{\substack{\omega \subset F \\ \omega \in \Omega}}{\operatorname{argmax}}(p(\omega \mid Y))$$

Example

from *Leibe, Schindler, and Van Gool, ICCV 2007*

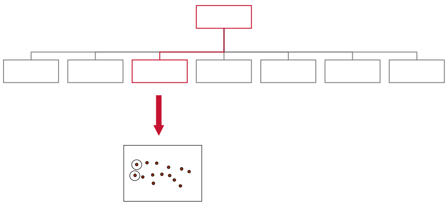
Tracking from the output of a pedestrian detector

from http://www.vision.ee.ethz.ch/~bleibe/index.html

Multiple-Hypothesis Tree (MHT)

Reid, IEEE-TAC 1979

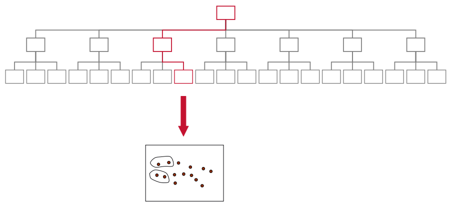
- Iteratively extend partial tracks at each time-step
- Pursue multiple hypotheses where there is ambiguity
- Prune unlikely hypotheses to keep search tractable



Multiple-Hypothesis Tree (MHT)

Reid, IEEE-TAC 1979

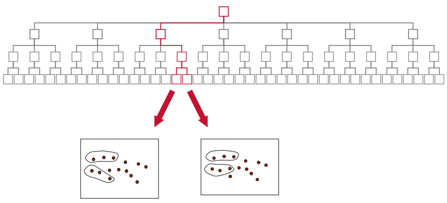
- Iteratively extend partial tracks at each time-step
- Pursue multiple hypotheses where there is ambiguity
- Prune unlikely hypotheses to keep search tractable



Multiple-Hypothesis Tree (MHT)

Reid, IEEE-TAC 1979

- Iteratively extend partial tracks at each time-step
- Pursue multiple hypotheses where there is ambiguity
- Prune unlikely hypotheses to keep search tractable



Markov Chain Monte Carlo Data Association

Oh, Russell, and Sastry, CDC-04, 2004

- Draw samples from posterior $p(\omega \mid Y)$ and select the maximum. Use Markov Chain Monte Carlo (MCMC) to do this efficiently.

initialise ω
repeat many times
  Sample ω' from proposal distribution $q(\omega, \omega')$
  Replace ω by ω' with (acceptance) probability:

$$A(\omega, \omega') = \min\left(1, \frac{p(\omega' \mid Y)q(\omega', \omega)}{p(\omega \mid Y)q(\omega, \omega')}\right)$$

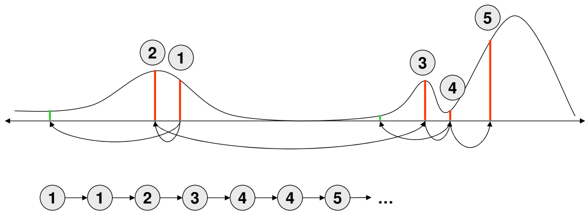
end

Introduction to MCMC

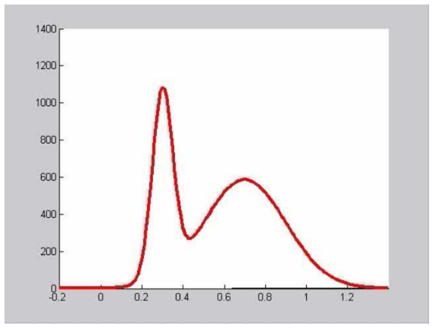
MCMC – Markov Chain Monte Carlo

When to use?

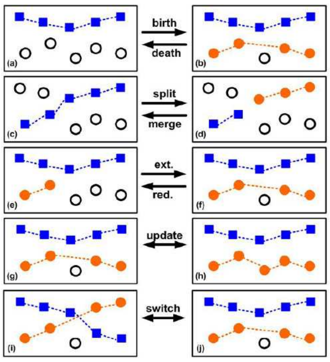
- You can't sample from the distribution itself
- Can evaluate it at any point



Introduction to MCMC



MCMC moves



From Oh, Russell and Sastry, CDC-04, 2004

Same approach at a higher level

Task: link people dropping-off and picking-up bikes

Damen & Hogg, BMVC 2009

Method

- Track people (+/- bikes) entering and leaving rack area
- Detect new clusters of dropped & picked bikes each time rack area becomes empty
- Find optimal combination of drop-pick and pass-through events

$$\arg\max_{\omega}(p(\omega \mid Y))$$

Formal definition of the model

Attribute multiset grammar

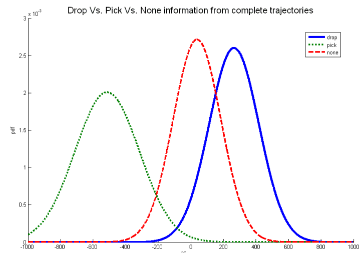
Syntactic Rule (r)				Attribute Rules (M)			Attribute Constraints (C)		
p_1	S	$\rightarrow$	V^*, x^*, y^*	y.action	=	"noise"	y.count	$<$	1
				x.action	=	"pass-by"	x.count	$\neq$	1
p_2	V	$\rightarrow$	Z_1, Z_2	V.action	=	"drop-pick"	Z_1.au	$<$	Z_2.au
				Z_1.action	=	"drop"	Z_1.count	$\neq$	1
				Z_2.action	=	"pick"	Z_2.count	$\neq$	1
				V.match	=	ψ_V (Z_1.pos, Z_2.pos)			
				Z_1.count	=	Z_2.count = 1			
p_3	V	$\rightarrow$	Z, u	V.action	=	"drop-only"	Z.count	$\neq$	1
				Z.action	=	"drop"			
				Z.count	=	1			
p_4	V	$\rightarrow$	u, Z	V.action	=	"pick-only"	Z.count	$\neq$	1
				Z.action	=	"pick"			
				Z.count	=	1			
p_5	Z	$\rightarrow$	x, y	x.action	=	Z.action	x.au	=	y.au
				y.action	=	Z.action	x.count	$\neq$	1
				Z.au	=	x.au			
				Z.pos	=	y.pos			
				Z.match	=	ψ_Z (x.traj, y.pos)			
				x.count	=	1			
				y.count	=	y.count+1			

Defining $p(\omega | Y)$

Based on:

- Change in the area of person-blobs between entering and leaving rack
- Proximity of people to bike clusters
- Similarity of bike clusters between drop and pick
- Prior probabilities for the different events

Likelihood of a person dropping, picking or passing through



Area difference of person-blob entering and leaving the rack

Likelihood of a drop/pick linkage

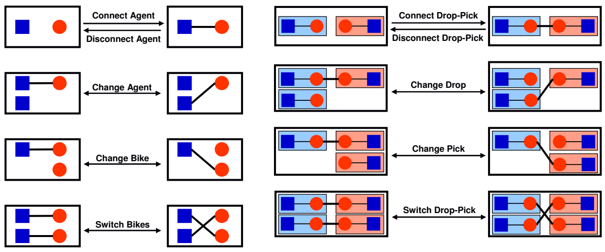
Drop

Pick

Pixel-wise comparison of bike-clusters

Optimise using an annealed MCMC

Possible moves:



Enter-exit problem

	Syntactic Rule (r)	Attribute Rules (M)	Attribute Constraints (C)
p_1	S $\rightarrow$ X*, E*, t*, b*	b.action = "noise"	b.count $\neq$ 1
		t.action = "pass-by"	t.count $\neq$ 1
p_2	X $\rightarrow$ C_1, C_2	C_1.action = "exit"	C_1.action $\neq$ "enter"
		C_2.action = "enter"	C_2.action $\neq$ "exit"
		X.action = "exit-enter"	C_1.time $<$ C_2.time
		X.match = ψ_M (C_1, C_2)	C_1.xCount $\neq$ 1
		X.bagDiff = |C_1.NoBags - C_2.NoBags|	C_2.xCount $\neq$ 1
		C_1.xCount = C_2.xCount = 1	
p_3	X $\rightarrow$ C, u	C.action = "exit"	C.action $\neq$ "enter"
		X.action = "exit-u"	C.xCount $\neq$ 1
		C.xCount = 1	
p_4	X $\rightarrow$ u, C	C.action = "enter"	C.action $\neq$ "exit"
		X.action = "u-enter"	C.xCount $\neq$ 1
		C.xCount = 1	
p_5	E $\rightarrow$ C_1, C_2	C_1.action = "enter"	C_1.action $\neq$ "exit"
		C_2.action = "exit"	C_2.action $\neq$ "enter"
		E.action = "enter-exit"	C_1.time $<$ C_2.time
		E.match = ψ_M (C_1, C_2)	C_1.eCount $\neq$ 1
		E.bagDiff = |C_1.NoBags - C_2.NoBags|	C_2.eCount $\neq$ 1
		C_1.eCount = C_2.eCount = 1	
p_6	E $\rightarrow$ C, u	C.action = "enter"	C.action $\neq$ "exit"
		E.action = "enter-u"	C.eCount $\neq$ 1
		C.eCount = 1	
p_7	E $\rightarrow$ u, C	C.action = "exit"	C.action $\neq$ "enter"
		E.action = "u-exit"	C.eCount $\neq$ 1
		C.eCount = 1	
p_8	C $\rightarrow$ t, B	t.action = C.action	t.trajID = B.trajID
		B.action = C.action	t.count $\neq$ 1
		C.NoBags = B.NoBags	B.count $\neq$ 1
		C.time = t.time	
		t.count = B.count = 1	
p_9	C $\rightarrow$ t	t.action = C.action	t.count $\neq$ 1
		C.NoBags = 0	
		C.time = t.time	
		t.count = 1	
p_{10}	B $\rightarrow$ b*	b.action = "carried"	b_i.trajID = b_j.trajID
		b.count = 1	b.count $\neq$ 1
		B.NoBags = |b*|	
		B.trajID = b.trajID	

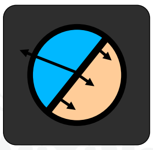
Early Cognitive Vision: Vision for Cognition

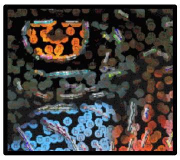
Emre Baseski,
Norbert Krüger,
Dirk Kraft
University of Southern Denmark

Florentin Wörgötter
University of Göttingen

Sinan Kalkan
METU

Nicolas Pugeault
University of Surrey



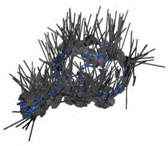


Overview

- Motivation from human vision
- An Early Cognitive Vision System
- Grounding of objects and grasping affordances
- Relation to Marr and 'mainstream computer vision'
- Exercise

Main Application

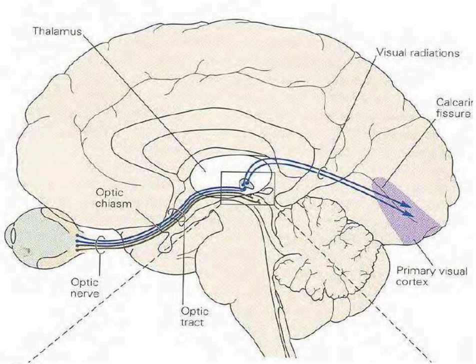
- **Grounding Objects and grasping affordances in (co-operation with Renaud Detry and Justus Piater)**
  - Grasping of unknown objects
  - Pose estimation
  - Birth of the object: Detection of objectness and object shape learning
  - Learning of object specific grasping affordances

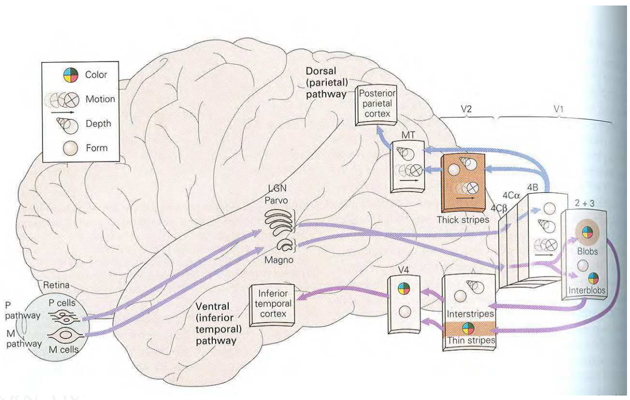


Overview

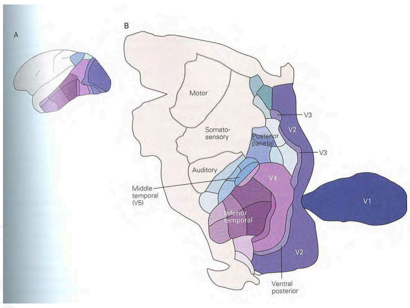
- **Motivation from human vision**
- An Early Cognitive Vision System
- Grounding of objects and grasping affordances
- Relation to 'mainstream computer vision'
- Exercise

The visual pathways



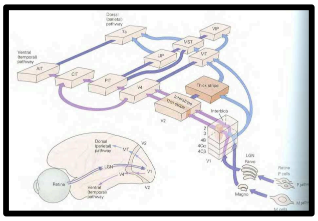


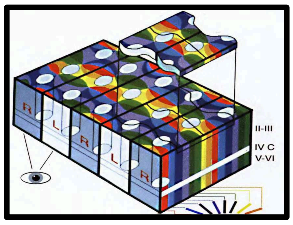
The visual pathways



- **Large areas (V1-V4) responsible for feature processing**

CoViL Cognitive Vision Lab University of Southern Denmark

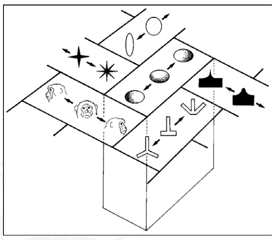


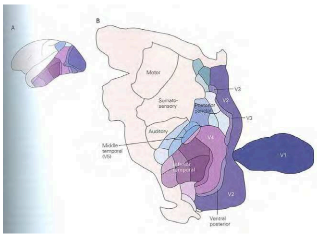


- **Visual Modalities**
  - **Orientation (Hubel, Wiesel 1962)**
  - **Phase (Jones, Palmer 1987)**
  - **Disparity (Barlow et. al 1967)**
  - **Colour (Hubel, Wiesel 1967)**
  - **Motion (Hubel, Wiesel 1967)**
  - **Junctions (Shevelev 1995)**

CoViL Cognitive Vision Lab University of Southern Denmark

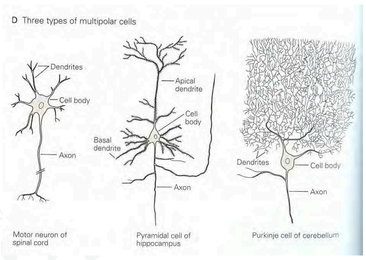
Very Complex Cells in IT (Tanaka 1997)

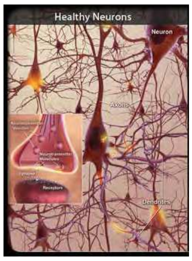




CoViL Cognitive Vision Lab University of Southern Denmark

Connectivity





- **High Degree of Connectivity**
  - 10^{12} Neurons
  - 10^{15} Connections

CoViL Cognitive Vision Lab University of Southern Denmark

Some Reflections about Human Visual Scene Representations

- **The human visual representation is multi-purpose**
  - many tasks (e.g., grasping, navigation, recognition, categorization, ...) need to be solved
  - → need of a generic representation
- **Large cortical areas are devoted to feature processing with a large intra and intercortical connectivity**
- **Richness**
  - covering multiple aspects of visual information and eventually also other sensorial data (haptic information)
  - Geometric and appearance based information
- **Hierarchy**
  - Change of representation from level to level
  - Reusabilty of parts
- **Visual representations are needed for actions**
- **These thoughts are not new**
  - Biederman, Geman, Perona, Buelthof, and more
  - but how much become they realized by currently used representations in Computer Vision?

CoViL Cognitive Vision Lab University of Southern Denmark

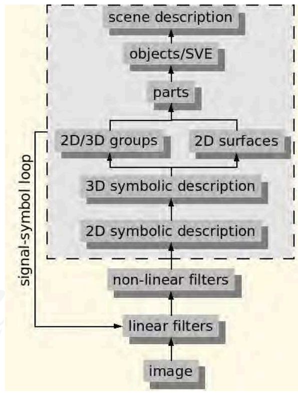
Overview

- Motivation from human vision
- **An Early Cognitive Vision System**
- Grounding of objects and grasping affordances
- Relation to Marr and 'mainstream computer vision'
- Exercise

Early Cognitive Vision System

- Local Primitves
  - Local symbolic image descriptors
  - Properties: Predictivity and Bandwidth Limitation
- Relations between Primitives
  - Temporal: (Rigid) Motion
  - Spatial: Extended structures
  - Disambiguation
- Feedback
  - Signal Symbol Loops

Overview of ECV system



Early Cognitive Vision System

- **Local Primitves**
  - Local symbolic image descriptors
  - Properties: Predictivity and Bandwidth Limitation
- Relations between Primitives
  - Temporal: (Rigid) Motion
  - Spatial: Extended structures
  - Disambiguation
- Feedback
  - Signal Symbol Loops

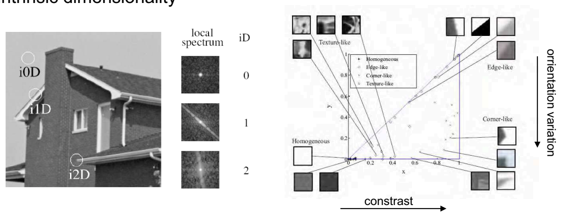
Local Image Structures

		Homog. patches	Edges	Junctions	Texture
2D	geometry	none	2D orientation	2D intersection multiple orientations	none
	appearance	mean colour shading gradient	2 or 3 colours phase	probably too unstable	not understood
3D	geometry	surface patch	3D point and 3D orientation	3D intersection multiple orientations (exception: T-junct.)	surface patch
	appearance	as in 2D	as in 2D	probably too unstable	not understood

Distinguishing Image Structures using the concept of Intrinsic Dimensionality

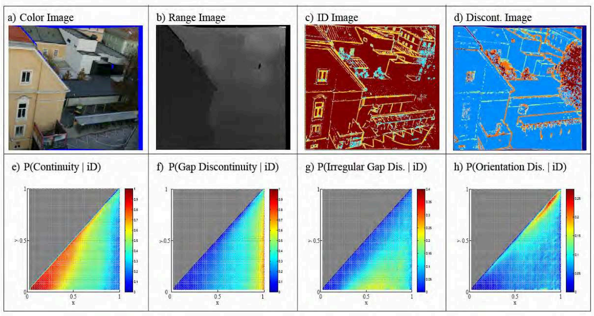
Image structures by their intrinsic dimensionality

The iD Triangle



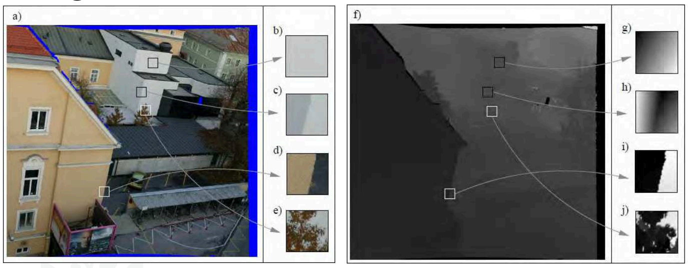
M. Felsberg, S. Kalkan and N. Krüger. Continuous Dimensionality Characterization of Image Structures. Image and Vision Computing 27:628-636, 2009.
D. Goswami, S. Kalkan, N. Kruger. Bayesian Classification of Image Structures. Scandinavian Conference on Image Analysis (SCIA) 2009.

Local Image Structures and their relation to Depth



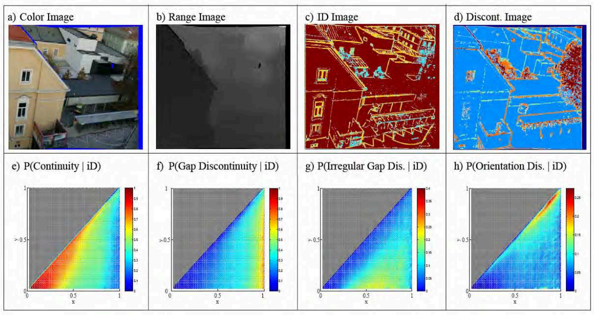
S. Kalkan, F. Wörgötter and N. Krüger. First-order and Second-order Statistical Analysis of 3D and 2D Structure. Network: Computation in Neural Systems, 18(2), pp. 129-160, 2007.
S. Kalkan, F. Wörgötter and N. Krüger. Statistical Analysis of Local 3D Structure in 2D Images. CVPR 2006.

Image and 3D Structures



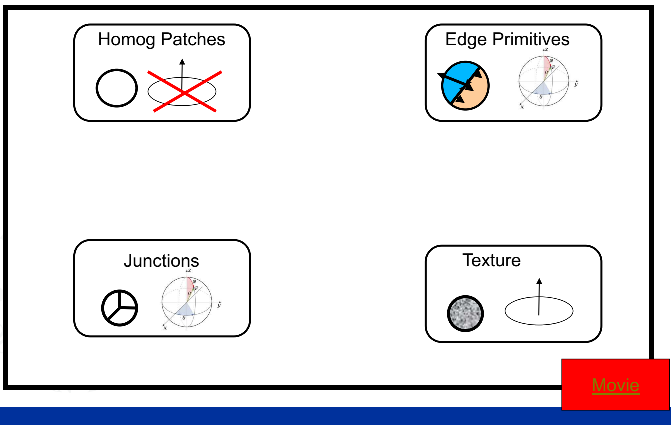
S. Kalkan, F. Wörgötter and N. Krüger. First-order and Second-order Statistical Analysis of 3D and 2D Structure. Network: Computation in Neural Systems, 18(2), pp. 129-160, 2007.
S. Kalkan, F. Wörgötter and N. Krüger. Statistical Analysis of Local 3D Structure in 2D Images. CVPR 2006.

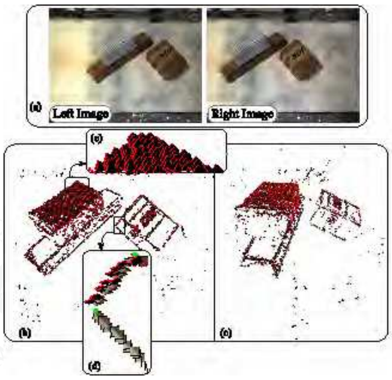
Local Image Structures and their relation to Depth



S. Kalkan, F. Wörgötter and N. Krüger. First-order and Second-order Statistical Analysis of 3D and 2D Structure. Network: Computation in Neural Systems, 18(2), pp. 129-160, 2007.
S. Kalkan, F. Wörgötter and N. Krüger. Statistical Analysis of Local 3D Structure in 2D Images. CVPR 2006.

Early cognitive vision: A local perspective





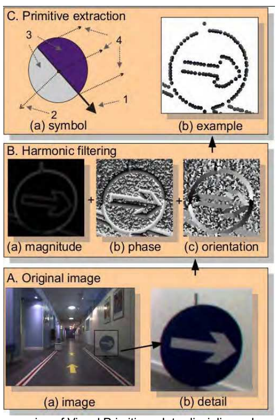
Visual Primitives (2D)

- **Visual primitives**
  - Local contour descriptor
  - Multi-modal

Geometry
- Position x
- Orientation θ

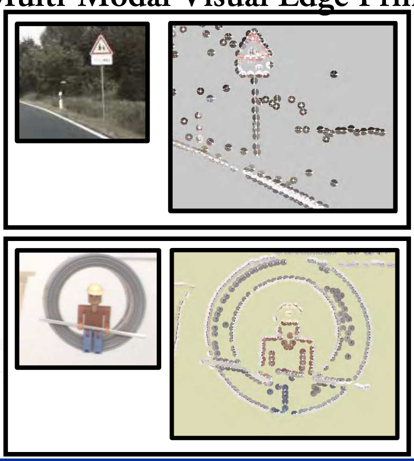
Appearance
- Phase ϕ
- Colour c
- Optic flow f

$$\pi = (x, \theta, \varphi, c, f)^T$$

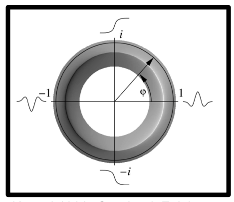


N. Krüger, M. Lappe and F. Wörgötter. Biologically Motivated Multi-modal Processing of Visual Primitives. Interdisciplinary Journal of Artificial Intelligence the Simulation of Behaviour, AISB Journal, 1(5): 4 17-427, 2004.
N. Krüger and F. Wörgötter. Multi-modal Primitives as functional Models of Hyper-columns and their use for contextual Integration. Proceedings of the 1st International Symposium on Brain, Vision and Artificial Intelligence.

Multi-Modal Visual Edge Primitives



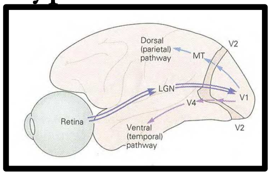
Phase

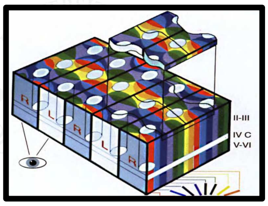


Kovesi 1999, Granlund, Felsberg

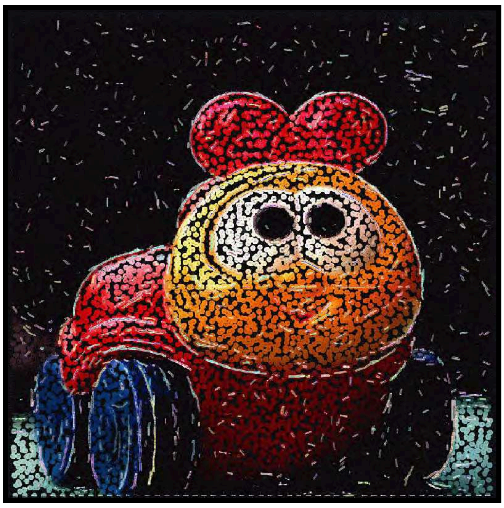
The Phase
- describes the local image structure by one parameter
- symbolic desription is different for line and edge structure (2 versus 3 colours)

Primitives as functional Abstraction of Hyper-columns in V1



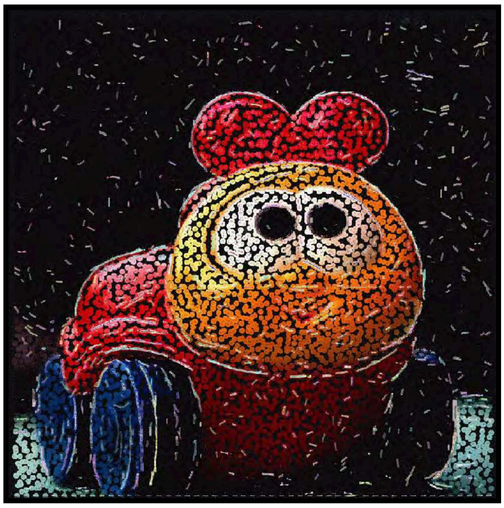


- Visual Modalities
  - Orientation (Hubel, Wiesel 1962)
  - Phase (Jones, Palmer 1987)
  - Disparity (Barlow et. al 1967)
  - Colour (Hubel, Wiesel 1967)
  - Motion (Hubel, Wiesel 1967)
  - Junctions (Shevelev 1995)
- They give a **condensed** information about the local edge patch for the next stage of processing
  - Bandwidth
  - Semantic Content
  - Split of geometry and appearance
  - High Predictivity



N. Krüger and F. Wörgötter. Symbolic Pointillism: Computer Art motivated by Human Brain Structures. Leonardo, MIT Press 38(4) p:337-340,2005.

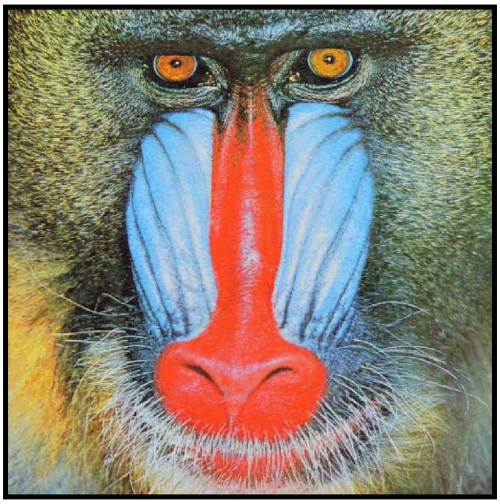
Kruger and Worgotter 2005, Leonardo



Kruger and Worgotter 2005, Leonardo

Kruger and Worgotter 2005, Leonardo

Kruger and Worgotter 2005, Leonardo



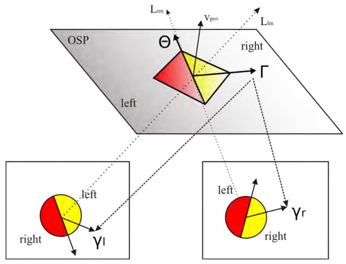
Kruger and Worgotter 2005, Leonardo

Kruger and Worgotter 2005, Leonardo

3D-primitive reconstruction

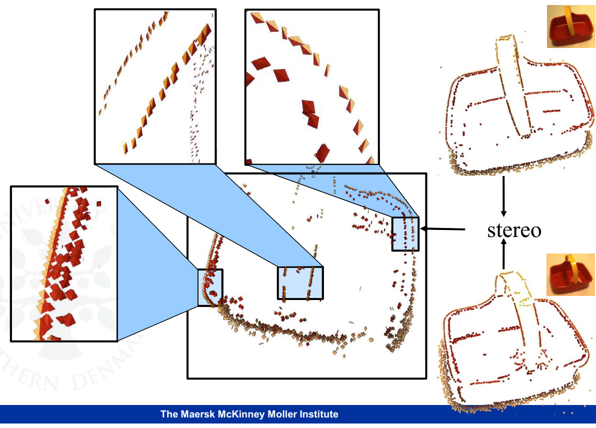
- **3D-primitive:** Π
  - position X
  - orientation Θ
  - reference plane Γ
  - phase Ω
  - colour C
  - size Λ

$$\Pi = (\underbrace{X, \Theta, \Gamma}_{\text{Geometry}}, \underbrace{\Omega, C}_{\text{Appearance}}, \Lambda)^T$$

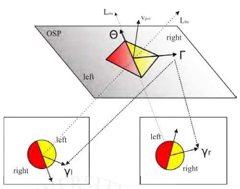


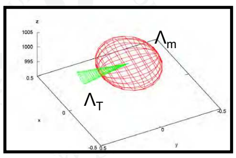
N. Pugeault (2008). Early Cognitive Vision: Feedback Mechanisms for the Disambiguation of Early Visual Representation, PhD

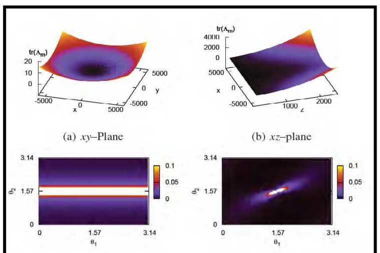
Visual Primitives (3D)



Reconstruction Uncertainty (very different from point features)







(a) $\mathrm{tr}(\Lambda_M)$ (b) $\mathrm{tr}(\Lambda_T)$

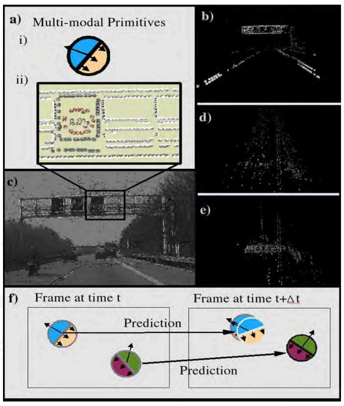
Differences to SIFT

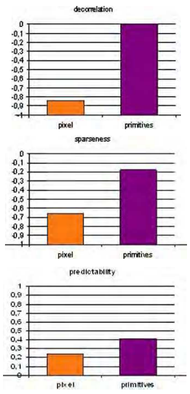
- **SIFT**
  - Very good for matching
  - Implicit representation of image patch content in a histogram
- **Primitives**
  - Condensation
  - Completeness
  - Explicit representation of content of local signal
  - (Embedded in local context)

Two Problematic Issues

- **Question 1: Why do we want to make such a transformation. We could stay at the pixel level (e.g., using pdes)**
  - explicitness allows the definition of higher level relations
  - condensation reduces number of those relations
  - separation of appearance and geometry allows for separated access
  - predictivity for disambiguation
- **Question 2: Is there and justification for designing it exactly that way**
  - I do not know
  - One way to address this is feature learning with additional constraints
    - P. König and N. Krüger. Perspectives: Symbols as self-emergent entities in an optimization process of feature extraction and predictions. Biological Cybernetics 94(4):325-334. 2006.
  - Another way: Justification by successful applications
  - Some design choices can be justified by image statistics
- **Bias/variance dilemma (Geman, Bienenstock)**
  - We might need to define prior knowledge justified by arguments or resemblance to th ebiological system

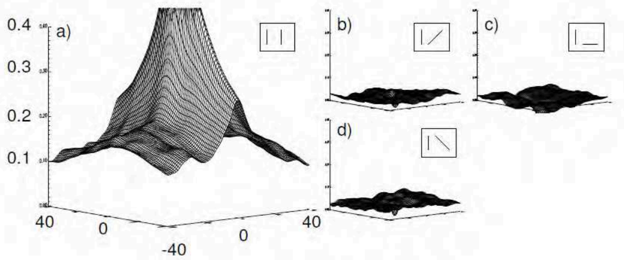
A case for condensation





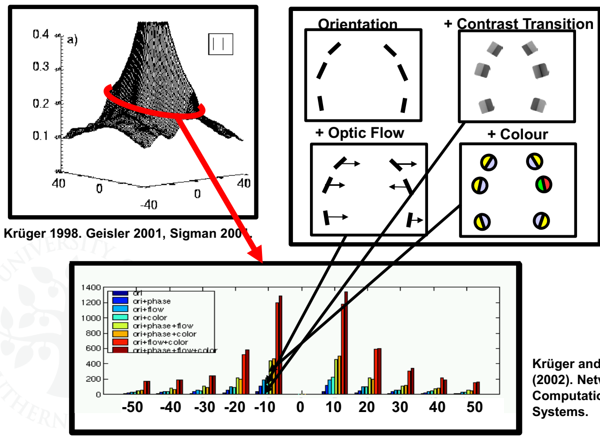
Peter König, Norbert Krüger (2007). Symbols as self-emergent entities in an optimization process of feature extraction and predictions. Biol Cybern (2006) 94: 325–334.

Natural Scene Statistics



N. Krüger. Collinearity and Parallelism are Statistically Significant Second Order Relations of Complex Cell Responses. Neural Processing Letters 8:117-129, 1998.
M. Sigman, G.A. Cecchi, C.D. Gilbert, and M.O. Magnasco. On a common circle: Natural scenes aand gestalt rules. PNAS, 98(4):1935{1949, 2001.
W.S. Geisler, J.S. Perry, B.J. Super, and D.P. Gallogly. Edge co-occurrence in natural images predicts contour grouping performance. Vision Research, 41:711{724, 2001.

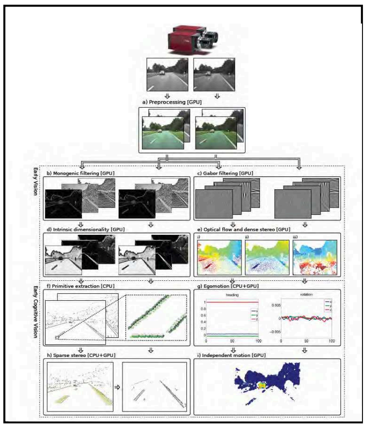
Gestalt Laws and natural Scene Statistics



Krüger and Wörgötter (2002). Network: Computation in Neural Systems.

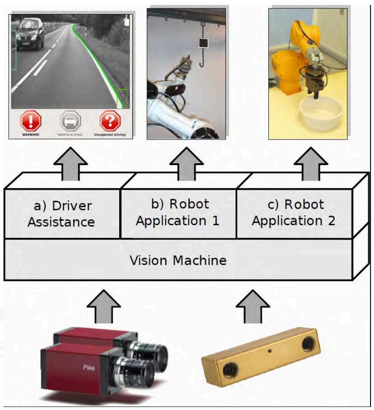
Vision Machine

- **Pre-processing**
  - Bayer pattern and rectification
- **Dense representation**
  - Phase based Stereo and Optic Flow
- **Symbolic descriptors**
  - 2D and 3D edge descriptors
- **Ego-motion**
  - 5D vector
- **IMOs**
  - using dense representation and egomotion

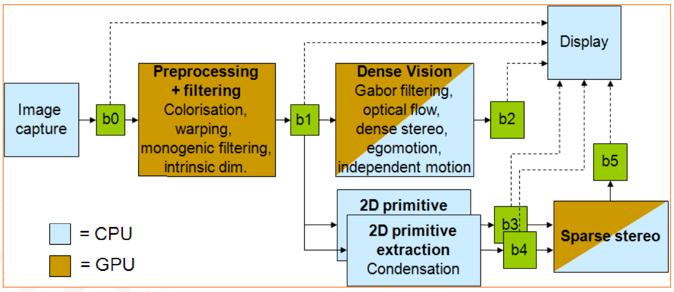


Jensen et al. (submitted), Lars B.W. Jensen ('PhD')

Vision Machine as Visual Frontend for Systems



Real-time processing of Primitives in a hybrid architecture (CPU and GPU)



Real-time processing of Primitives

- Performance Measurement in comparison to single CPU
  - a speed up of approximately a factor 90, and
  - a reduction of latency of a factor 26
- Standard Hardware (PC + 2 GPUs)
  - Price approx. 2500 Euros (with PC, without cameras)

Processing stage	Input data	Processing time Hybrid	Processing time Single CPU
Image capture	2048x1536 image pair	1.0 ± 0.2 ms	1.0 ± 0.2 ms
Preprocessing + filtering	2048x1536 image pair	18.7 ± 0.5 ms	985.0 ± 2.9 ms
Dense vision	512x384 image pair	39.1 ± 1.1 ms	2760.0 ± 1.0 ms
Primitive extraction**	7x 512x384 image pairs	$2 * 31.0 \pm 2.5$ ms	$2 * 48.6 \pm 3.5$ ms
Sparse stereo**	2 vectors of 2D primitives	41.9 ± 7.1 ms	32.5 ± 6.9 ms
Average total latency		148.3 ms	3875.7 ms
Average frame rate		23.2 fps	0.26 fps

Movie

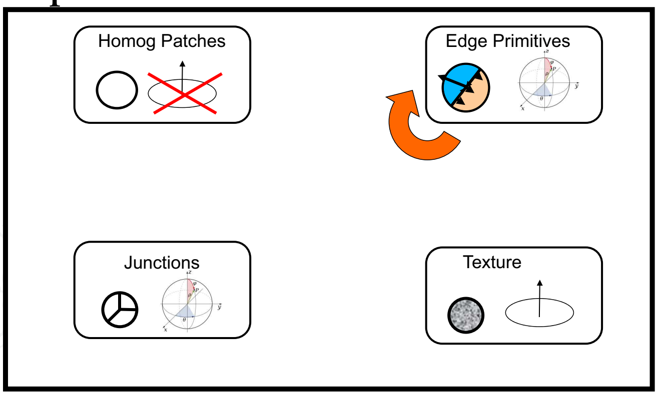
Early Cognitive Vision System

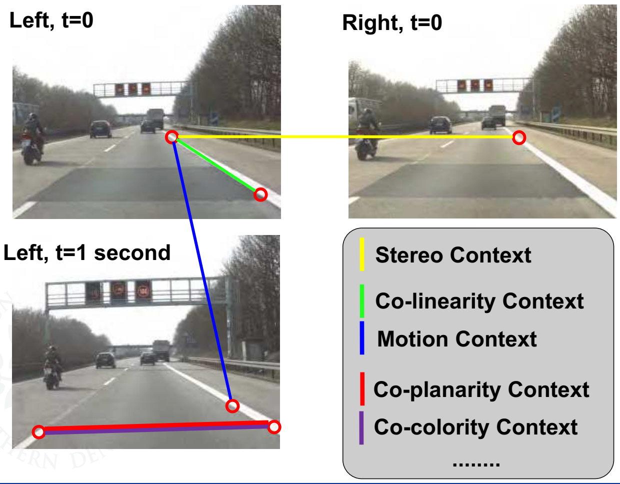
- Local Primitves
  - Local symbolic image descriptors
  - Properties: Predictivity and Bandwidth Limitation
- **Relations between Primitives**
  - Spatial: Extended structures
  - Temporal: (Rigid) Motion
- Feedback
  - Signal Symbol Loops

Early Cognitive Vision System

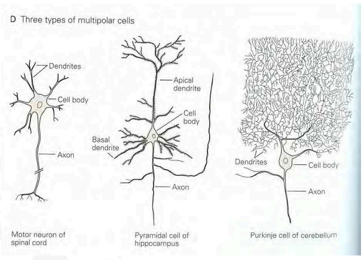
- Local Primitves
  - Local symbolic image descriptors
  - Properties: Predictivity and Bandwidth Limitation
- **Relations between Primitives**
  - **Spatial: Extended structures**
  - Temporal: (Rigid) Motion
- Feedback
  - Signal Symbol Loops

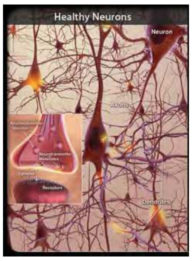
Early cognitive vision: A semi-global perspective



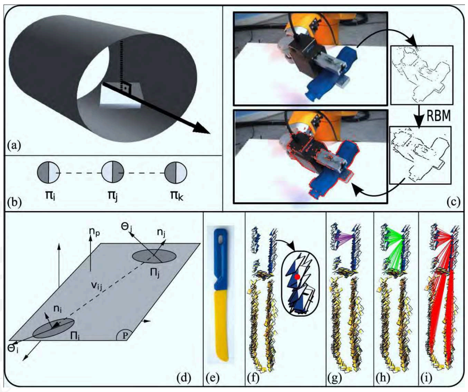


Connectivity



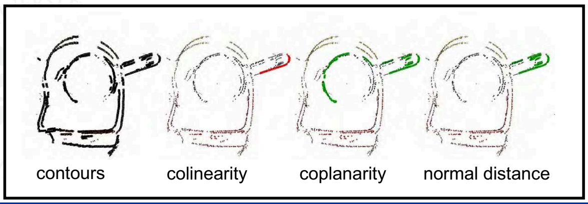


Some Relations



Problems of Reasoning with Relations

- **Relations make relevant structural information explicit**
  - Action assciation
  - Object recognition
- **Space of relations increases exponentially**
- **Hence relations need to be defined between contours**



Embedding of Primitives in contours

- **NURBS**
  - Parametrization of inbetween positions
  - Smoothing out noise in reconstruction process

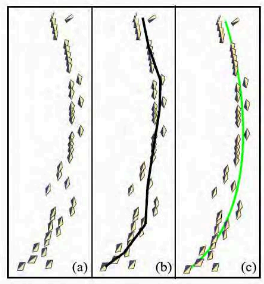
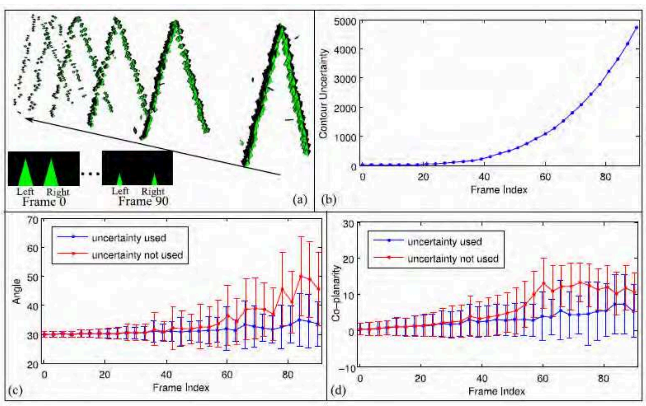


Figure 3: NURBS parametrization of a sample contour. (a) 3D primitives of a contour. (b) The line segment representation of the contour is shown with black line segments. (c) The approximated NURBS is shown as a green curve.

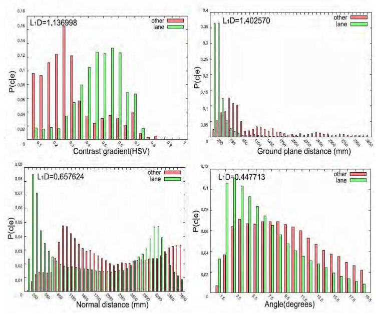
Emre Baseski (PhD, submitted)

Problems when dealing with 3D Relations



Application of Relations I: Lane Detection

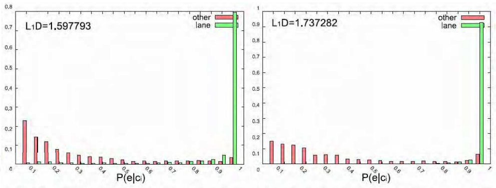
- **Learning of lane structure**
  - Images with labeled lanes
  - We can associate 3D descriptors as well as groups therefof to lanes and non-lane structures
  - Prior probability distributions and conditional probability densities for individual cues
  - The probability of entities belong to the lane or non-lane can be computed by Bayes formula



Bosemann et al. (2009)

Lane Detection

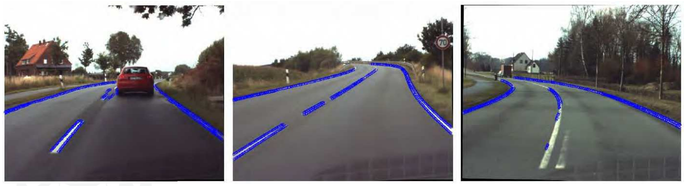
- **Posterior probability densities**



(b) The posterior probability distribution based on groups of maximum 6 primitives without uncertainties (c) The posterior probability distribution based on groups of maximum 6 primitives after applying an uncertainty threshold

The posterior probability densities for individual and grouped entities, together with the influence of grouping on the L1 − norm value.

Lane Detection



Results of the Bayesian framework applied on different frames.
The blue color marks the 2D contours that contain the 3D lane contours.

Lane Detection

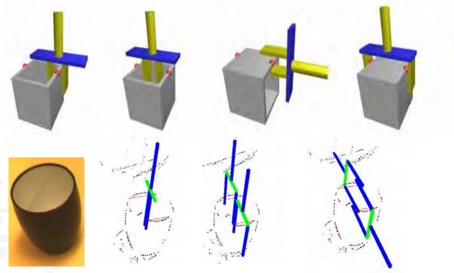
- **Evaluation**

Class	Amount of selected entities			
	Primitives	Contours of 6 primitives	Contours of 6 primitives with 2D extension	Contours of 6 primitives with uncertainties
True positive	12347	2666	2874	2247
True negative	21304	803	834	259
False positive	8687	796	588	316
False negative	572	91	60	34

Using the 2D extension we obtain a classification success rate to 85.1% and a positive success rate to 83%. Once the contour relations are used after applying an uncertainty threshold, we obtain a classification success rate of 87.7% and a positive success rate of 87.7% as well.

Application II: Grasp-Feature Associations

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



Movie

M. Popović et a. (in press). A Strategy for Grasping Unknown Objects based on Co-Planarity and Colour Information. Robotics and Autonomous Systems.

Other Applications

- **Object Recognition**
- **Pose Estimation**
- **Tracking**

Early Cognitive Vision System

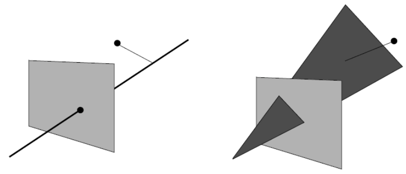
- Local Primitves
  - Local symbolic image descriptors
  - Properties: Predictivity and Bandwidth Limitation
- **Relations between Primitives**
  - Spatial: Extended structures
  - **Temporal: (Rigid) Motion**
- Feedback
  - Signal Symbol Loops

Motion Estimation Motivation

- **Motion information is fundamental for humans and animals**
  - Egomotion
  - Motion of Objects
- **Its estimation is based on Constraints defined by Correspondences**
  - Different kinds of correspondences (associated to different kind sof image structures) are possible

Motion Estimation Motivation

- **Its estimation is based on Constraints defined by Correspondences**

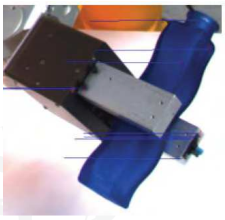


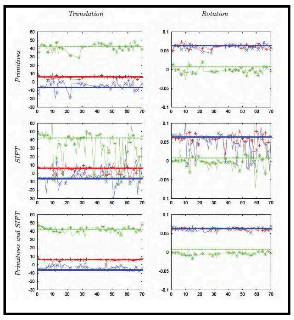
3D Point/3D Line Constraint 3D Point/3D Plane Constraint

- Bodo Rosenhahn, Oliver Granert, Gerald Sommer (2002). Monocular Pose Estimation of Kinematic Chains.
- Bodo Rosenhahn, Gerald Sommer (2002). Adaptive Pose Estimation for Different Corresponding Entities.

Motion Estimation with different kinds of Entities

- Controlled Robot motion
- Ground truth known
- http://www.mip.sdu.dk/covig/sequences.html



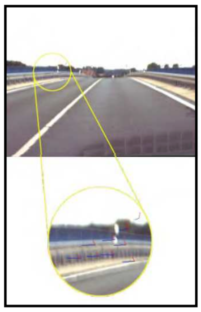


Feature Set	Tx (mm)	Ty (mm)	Tz (mm)	Rx (rad)	Ry (rad)	Rz (rad)
Primitives	1.0	4.1	3.2	0.003	0.005	0.005
SIFT	3.2	6.9	14.1	0.009	0.018	0.008
Primitives + SIFT	0.8	3.0	2.5	0.002	0.003	0.004

Florian Pilz, Nicolas Pugeault, and Norbert Kruger. Comparison of Point and Line Features and Their Combination for Rigid Body Motion Estimation. Statistical and Geometrical Approaches to Visual Motion Analysis, Springer LNCS 5604, p: 280-304, 2009.

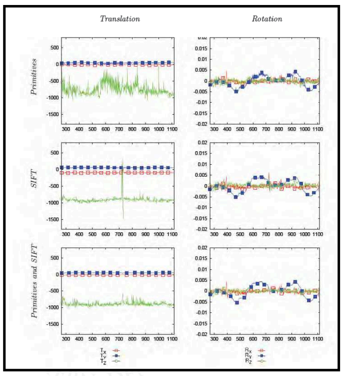
Motion Estimation based on Sparse Correspondences

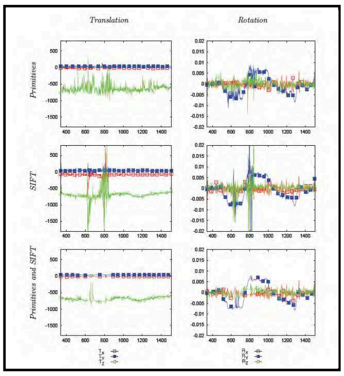
- **Combining Point and Edge Features**
  - stabilzes motion estimation
  - edges: more frequent but weaker constraints
  - point features: less frequent but stronger constraint
- **There are are situations in outdoor scenes where there ar enot enough point features**



Florian Pilz, Nicolas Pugeault, and Norbert Kruger (2009). Comparison of Point and Line Features and Their Combination for Rigid Body Motion Estimation. In: Statistical and Geometrical Approaches to Visual Motion Analysis.

Motion Computation

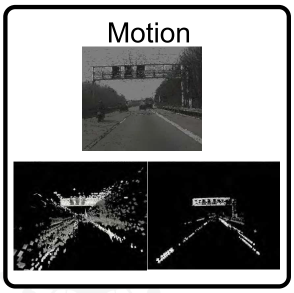


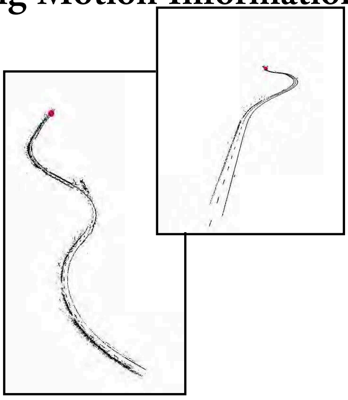


Disambiguation using motion information

- **Stereo faces two problems**
  - Wrong correspondences leading to outliers
  - Uncertainties
  - Occclusion
- **By means of motion all three problems can be solved**
  - Structure from Motion
  - Bundle Adjustment
- **One particularity of our approach**
  - Not points or lines are the entities on which we operate but symbolic primitives
  - We want to develop an structure from motion algorithm that can be applied to all kind of entities

Disambiguation using Motion Information





Movie

N. Pugeault, K. Pauwels, M. Van Hulle, F. Pilz & N. Krüger. A Three-Levels Architecture for Model-Free Detection and Tracking of Independently Moving Objects. VISAPP 2010.

On top Bayesian Filtering amd integration of new Features

State vector s_t=(t,αr,A) with A = (ω; c_l; c_r), with associated uncertainty matrix Σ_t

Prediction s_{t+1}=(RBM(t,r),A), appearance unchanged

Three processes:

1) Scaled Unscented Kalman (SUR): high flexibility for using different modalities since predicted state covariances become estimated from the data

2) Bayesian accumulation of confidences

$$p\left[\mu_t(Z_i,Z_j)\right] = \frac{\dots}{(2\pi)^{n/2}\sqrt{|\Sigma_t|}}$$ Match Probability

$$\mu^*_{i,t} = \{\mu_{i,1},\dots,\mu_{i,t}\}$$ Match History

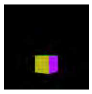
$$p\left[Z_i|\mu^*_{i,t}\right] = \frac{p\left[\mu^*_{i,t}|Z_i\right]p\left[Z_i\right]}{p\left[\mu^*_{i,t}|Z_i\right]p\left[Z_i\right]+p\left[\mu^*_{i,t}|\bar{Z}_i\right]p\left[\bar{Z}_i\right]}$$ Accumulated Matching probability

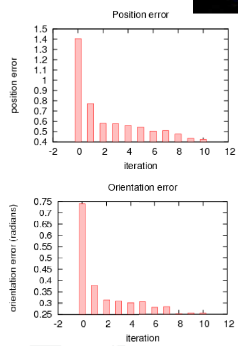
3) Including new unmatched entities

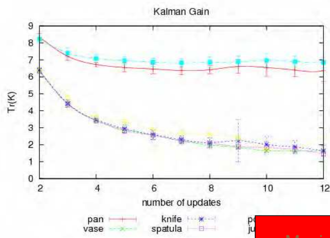
Accumulation

- **State vector s_t=(G,A) with**
  - With geometry vector G=(X,Θ)with associated uncertainty matrix Σ_t
  - Appearance bector A = (ω; c_l; c_r),
- **Prediction s_{t+1}=(RBM(t,r),A)**
- **Three processes address the three problems**
  - Bayesian Filtering elliminates outliers
  - Unscented Kalman Filter (UKF) improves 3D reconstruction
    - Since the UKF computes Covariance Matrices on the fly, every kind of feature can be easily integrated
  - Mechanisms for introducing novel pbject aspects

Convergence







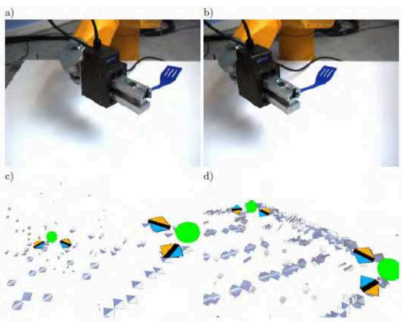
a) artificial example

b) real objects

Movie

Pugeault et al. (BMVC 2008), Pugeault PhD 2008

Extension to other descriptors



K. Simonsen, M. Nielsen, F. Pilz, N. Krüger, N. Pugeault, Spatial-Temporal Junction Extraction and Semantic Interpretation. 5.th International Symposium on Visual Computing. Lecture Notes for Computer Science (LNCS), Springer Verlag 2009.

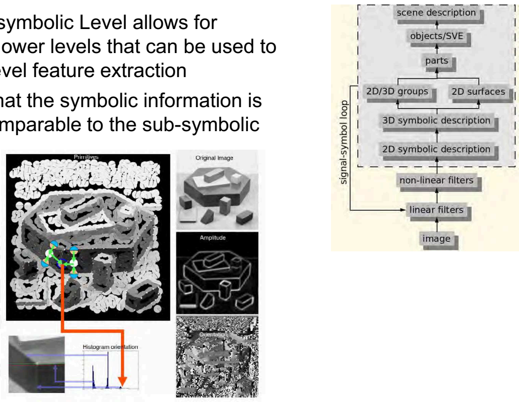
Early Cognitive Vision System

- Local Primitves
  - Local symbolic image descriptors
  - Properties: Predictivity and Bandwidth Limitation
- Relations between Primitives
  - Spatial: Extended structures
  - Temporal: (Rigid) Motion
- **Feedback**
  - **Signal Symbol Loops**

CoViL Cognitive Vision Lab University of Southern Denmark

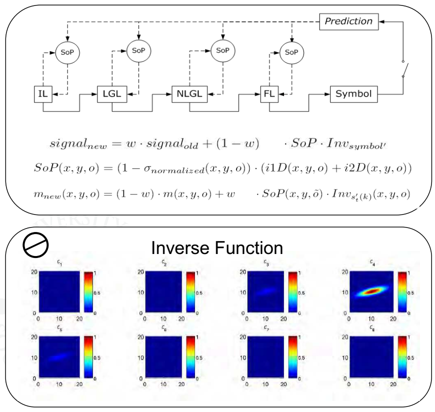
Signal Symbol Loop: Concept

- Reasoning on symbolic level allows for predictions on lower levels that can be used to enhance low-level feature extraction
- This requires that the symbolic information is made again comparable to the sub-symbolic information



CoViL Cognitive Vision Lab University of Southern Denmark

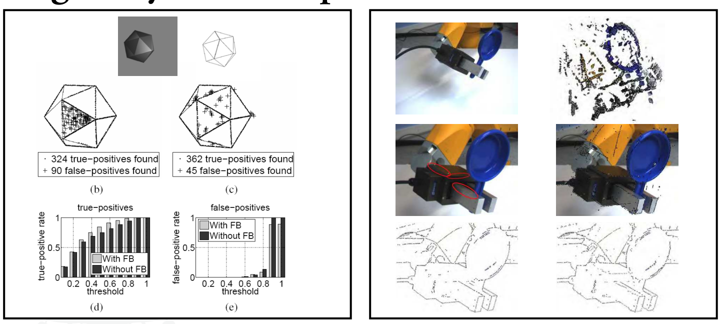
Signal Symbol Loop: Mathematics



- **Contexts on symbolic level**
  - Rigid body motion
  - Part and Object Knowledge
- **Becoming fed back by inverse function**

CoViL Cognitive Vision Lab University of Southern Denmark

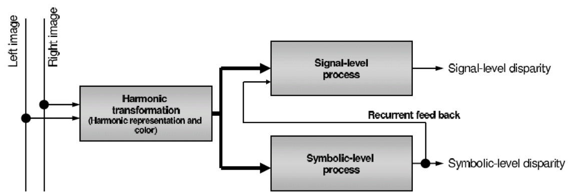
Signal Symbol Loop: Results



S. Kalkan, S. Yan, V. Krüger, F. Wörgötter and N. Krüger. A Signal-Symbol Loop Mechanism For Enhanced Edge Extraction. Int. Conf. on Computer Vision Theory and Applications (VISAPP'08), January, 2008.

CoViL Cognitive Vision Lab University of Southern Denmark

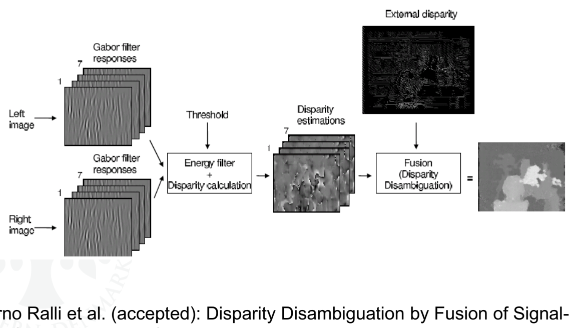
Improving Stereo by guiding dense stereo through sparse stereo



Jarno Ralli et al. (accepted): Disparity Disambiguation by Fusion of Signal- and Symbolic-Level Information, Machine Vision and Aplications.

CoViL Cognitive Vision Lab University of Southern Denmark

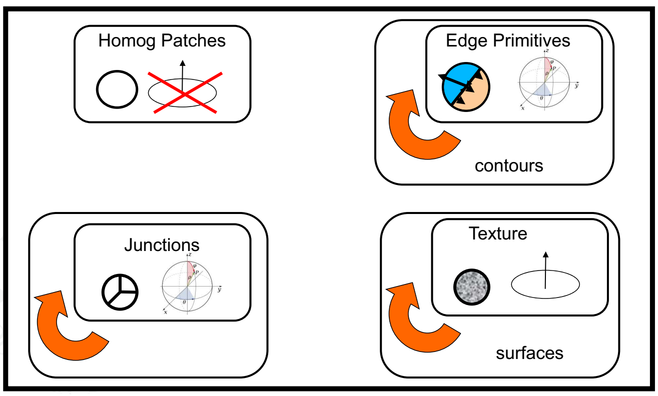
Improving Stereo by guiding dense stereo through sparse stereo



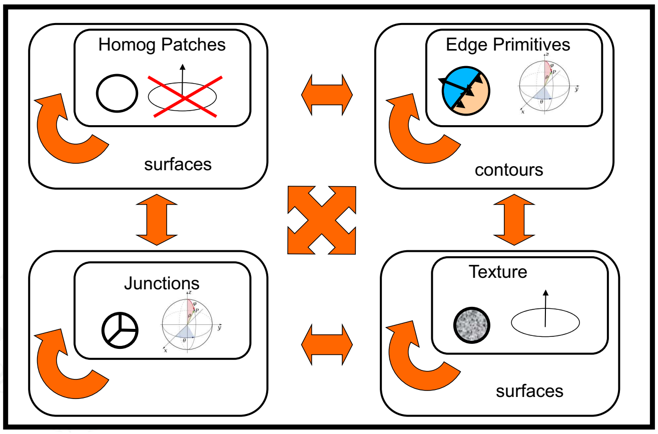
Jarno Ralli et al. (accepted): Disparity Disambiguation by Fusion of Signal- and Symbolic-Level Information, Machine Vision and Aplications.

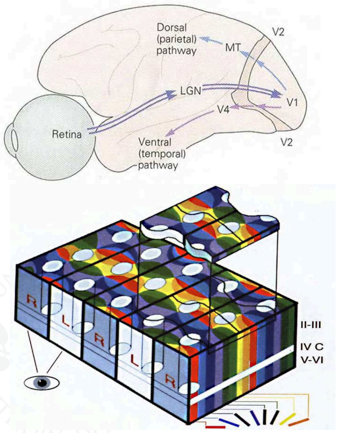
CoViL Cognitive Vision Lab University of Southern Denmark

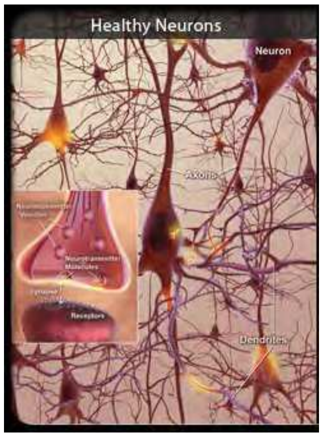
Early cognitive vision: Status



Complete Early Cognitive Vision System







Overview

- Motivation from human vision
- An Early Cognitive Vision System
- **Grounding of objects and grasping affordances**
- Relation to Marr and 'mainstream computer vision'
- Exercise

Grounding of Objects and Grasp Affordances (in co-operation with Justus Piater and Renaud Detry)

- Problem Statement
- Sub-modules
  - Grasping Unknown Objects
  - Detection of 'Objectness' and Learning Object Shape
  - Pose estimation
  - Learning grasp affordance densities
- A system perspective

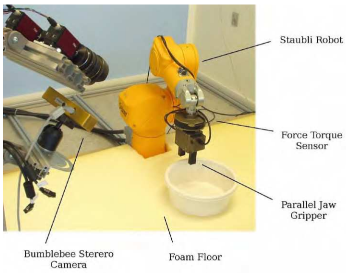
Grounding of objects and grasping affordances: Definition of the problem

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

Baron von Muenchhausen also called 'Der Luegenbaron'

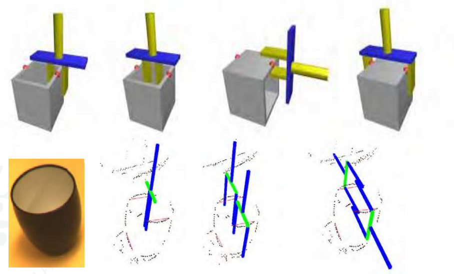
Embodiment at SDU

- **Keeping things simple**
  - High precision industrial robot
  - Controlled camera robot setting
  - Two finger gripper
- **Exploration**
  - Actions in unpredictable situtations
    - Foam
    - Force-torque sensor
- **Basic knowledge on body movements and environment**
  - Pro-prioceptive information is (more or less directly) usable
  - Path planning
  - Collision avoidance



Module I: Grasping unknown objects

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

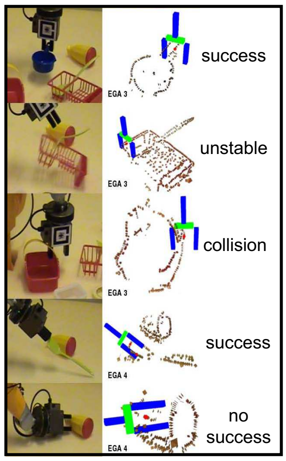


Grasping unknown objects

- Surprising result: A success rate between 30-40% is already achievable by such a simple mechanisms.
  - One reason is high level mechansism for hypotheses rejections through motion planning
- There is an *autonomous success evaluation* based on haptic information
  - Collision
    - Force-torque sensor above threshold
  - no success
    - distance between fingers =0 after grasping attempt
  - unstable
    - distance b.f. > 0 after grasping attempt and =0 after lifting
  - stable
    - distance b.f. > 0 after grasping attempt and after lifting

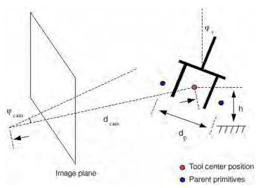
Movie

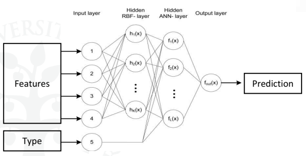
Establishment of Episodic memory

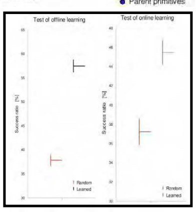


Episodic memory can be used for learning important features and relations given by ECV system

- Since there are many co-plpanarity relations a large number of potential grasp options become computed
- The system chan choose which option to execute







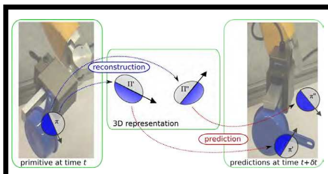
L. Bodenhagen et al., Learning to Grasp Unknown Objects Based on 3D Edge Information. 2009 IEEE International Symposium on Computational Intellegence in Robotics and Automation (CIRA2009)

Summary Module 1

- **It is possible wihout any supervision**
  - to achieve physical control over objects without explicit object knowledge by a rather simple behaviour
  - to establish an episodic memory with labeled data in terms of grasping attempts and their success

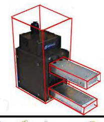
Module 2: Accumulation

- **What is an object?**
  - Corresponding to the three criteria for 'objectness' by Gibson
    - Temporal Stability
    - Manipulatability
    - Appropriate size
- **An object is**
  - something in the robots hand
  - that moves according to the robots pro-prioceptive information



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

Pugeault, N., Wörgötter, F., and Krüger, N. Accumulated Visual Representation for Cognitive Vision (BMVC 2008).



- Body knowledge can be used to substract gripper

SLAM,
Structure from Motion

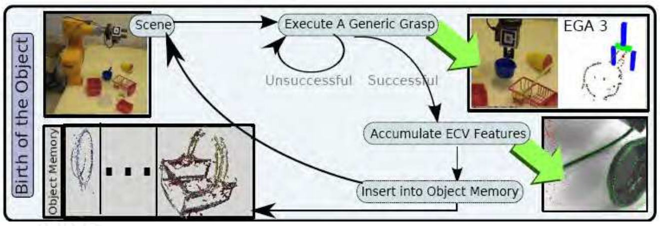
Movie

D. Kraft, N. Pugeault, E. Başeski, M. Popović, D. Kragić, S. Kalkan, F. Wörgötter and N. Krüger. Birth of the Object: Detection of Objectness and Extraction of Object Shape through Object Action Complexes. International Journal of Humanoid Robotics (IJHR), 2008, 5, 247-265.

Summary module 2

- **Once the agent has physical control over an object, it**
  - can detect 'objectness'
  - compute a geometric/appearance representation of the object

The first learning cycle: Birth of the (first) object

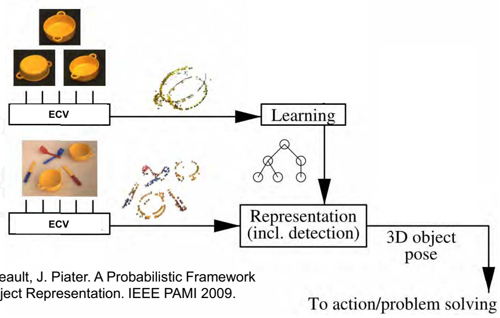


D. Kraft, N. Pugeault, E. Başeski, M. Popović, D. Kragić, S. Kalkan, F. Wörgötter and N. Krüger. Birth of the Object: Detection of Objectness and Extraction of Object Shape through Object Action Complexes. International Journal of Humanoid Robotics (IJHR), 2008, 5, 247-265.

Some objects being 'born'

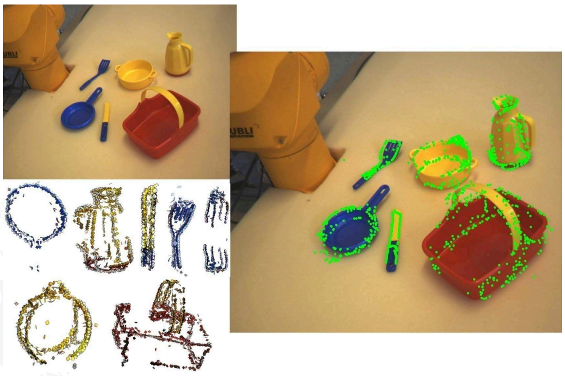
Module 3: Pose estimation

- **Based in object knowledge objects can be recognized and their pose being estimated**
- **Method**
  - Learn probablistic relational models of ECV feature combinations
  - Matching using probablistic inference



R. Detry, N. Pugeault, J. Piater. A Probabilistic Framework for 3D Visual Object Representation. IEEE PAMI 2009.

Results



CoViL Cognitive Vision Lab University of Southern Denmark

Summary module 3

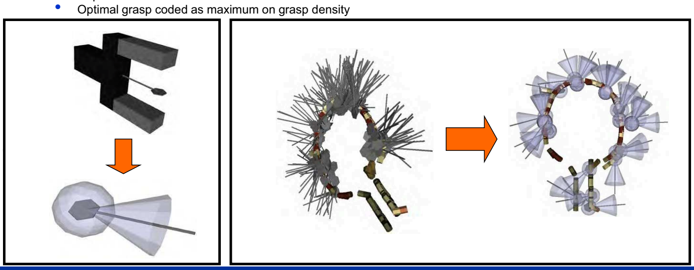
- **Once the agent has an object in its memory**
  - it can find the object in the scene and
  - compute its pose

CoViL Cognitive Vision Lab University of Southern Denmark

Module 4: Learning grasp affordance densities

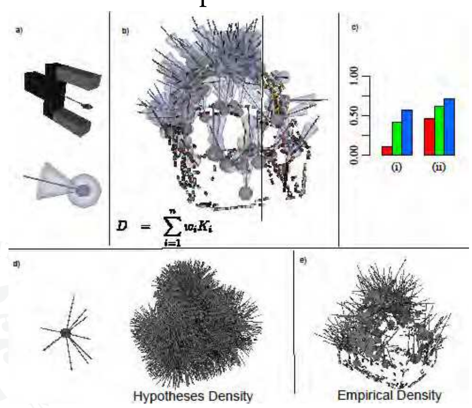
(with Justus Piater, Renaud Detry, Oliver Kroemer and Jan Peters)

- **Module 1: Learning grasping without explicit object knowledge**
- **With the additional object knowledge after the 'Birth of the Object' the system can now explore how to grasp this specific object**
- **Coding Grasp Densities:**
  - A grasp is just coded by the pose of the end-effector
  - A grasp attempt can be transformed to a 6D kernel which is the basic building of the grasp density
  - A full grasp density is build up by a number of kernels
- **Advantage**
  - Representinh the manifold of affordances
  - Optimal grasp coded as maximum on grasp density



CoViL Cognitive Vision Lab University of Southern Denmark

Grasp Denisties



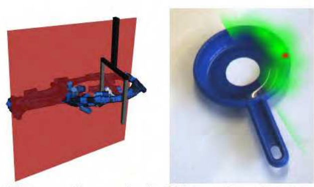
Renaud Detry, Dirk Kraft, Anders Glent Buch, Norbert Krüger and Justus Piater. Refining Grasp Affordance Models by Experience. IEEE International Conference on Robotics and Automation, 2010.

CoViL Cognitive Vision Lab University of Southern Denmark

Grasp Densities

- **Grasp Densities express all grasping affordances associated to a specific object**
- **The optimal grasp in a specific context can then be computed online**

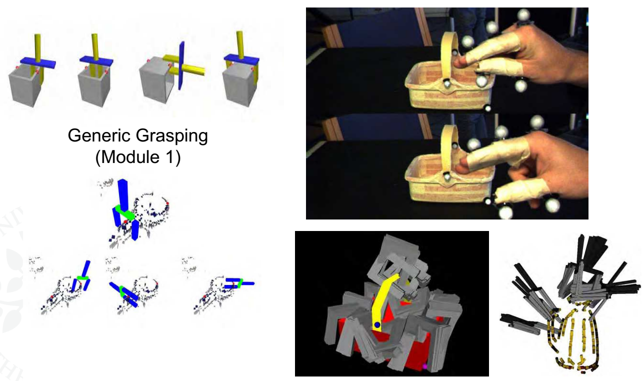
(a) Downward grasp (b) 45° rotation around z (c) 90° rotation around z (d) 135° rotation around z



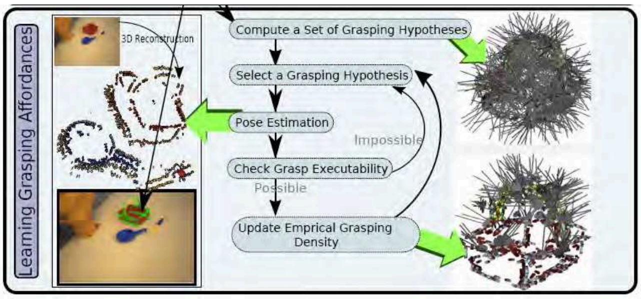
(a) Best grasp within region of reach (b) Constrained grasp density (green) and optimal grasp position (red)

CoViL Cognitive Vision Lab University of Southern Denmark

Two ways of learning: Exploration or Learning by Demonstration



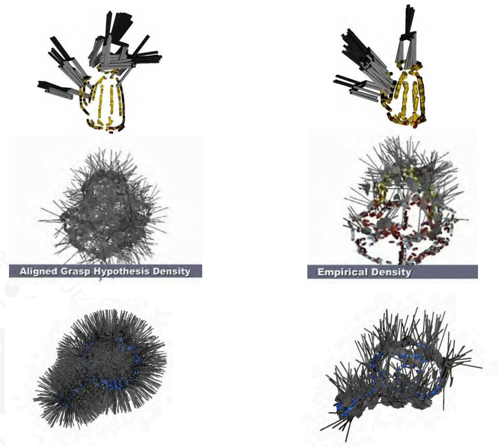
Second learning cycle (Grasp Exploration)



Movie

Hypothesis Density

Empirical Density



Summary module 4

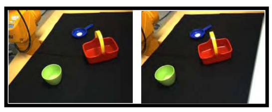
- **Once the agent has an object in its memory**
  - it can find the object in the scene, compute its pose
  - and associate grasp hypothesis to the object in the scene
  - test the hypothesis (->empirical affordance density)

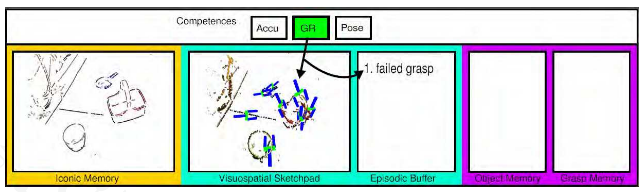
Grounding of objects and grasping affordances: Definition of the problem

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

Bootstrapping from a System Point of View

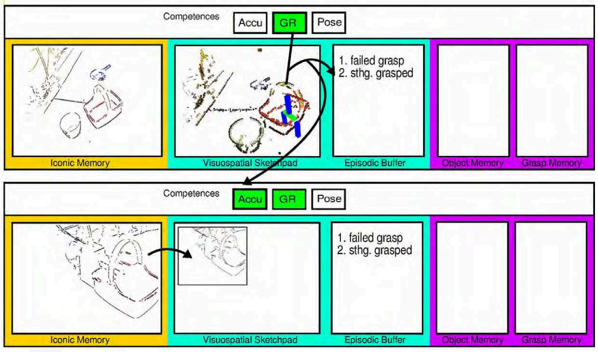
World-view of the Innate System



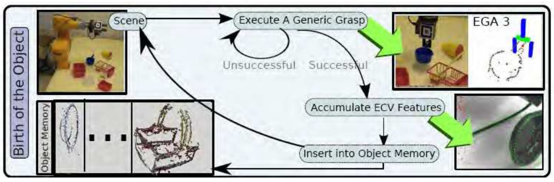


- **Early Cognitive Vision (ECV) System**
  - Semantically rich and highly structured visual representations
- **GR: First 'reflex-like' behaviour**

World view of the Cognitive System

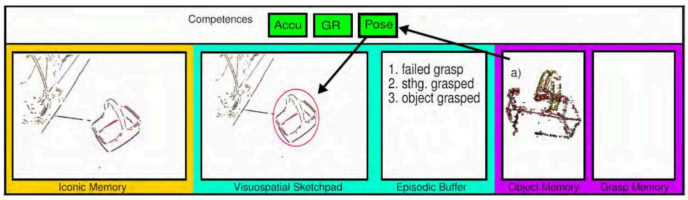


Birth of the Object

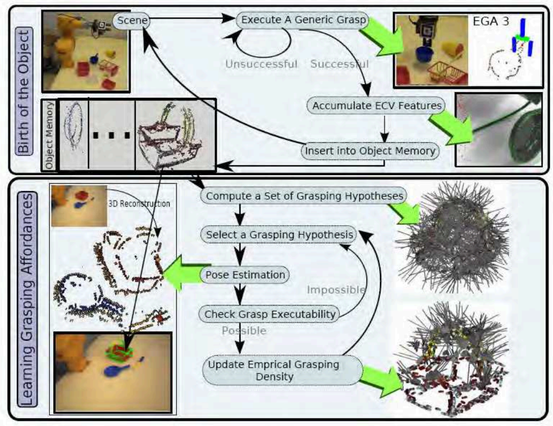


- **Object**
  - Visual features changing according to pro-prioceptive information
- **Three criteria by Gibson**
  - Temporal stability
  - Relative size
  - Manipulatability

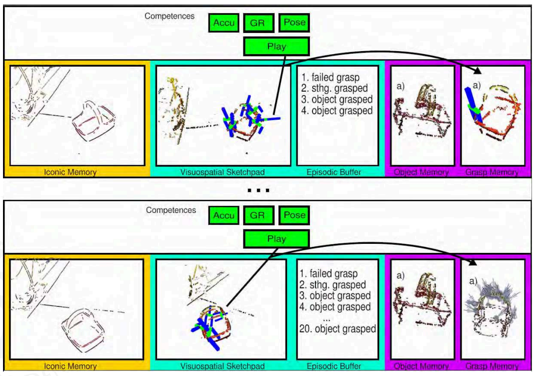
World view of the Cognitive System



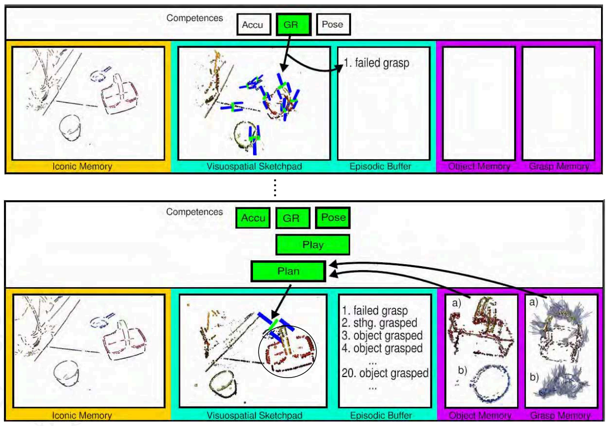
Pugeault et al. (2008), BMVC; Detry et al (2009), PAMI



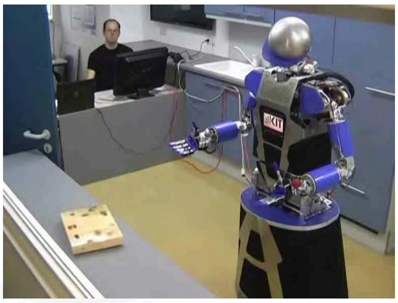
World view of the Cognitive System



World view of the Cognitive System



Performing a plan using pushing and grasping OAC



Movie

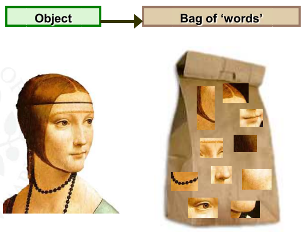
D. Omrcen, A. Ude, and A. Kos (2008), Learning primitive actions through object exploration, International Conference on Humanoid Robots.
R. Petrick et al. (2009), Combining Cognitive Vision, Knowledge-Level Planning with Sensing, and Execution Monitoring for Effective Robot Control. ICAPS.

Overview

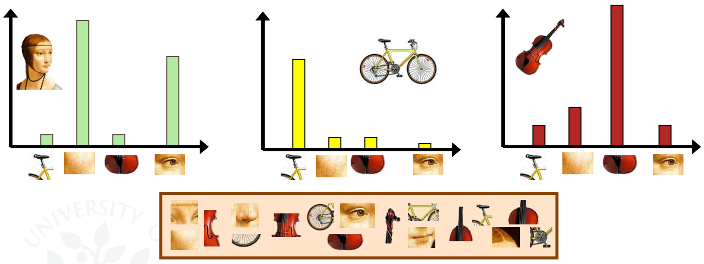
- Motivation from human vision
- An Early Cognitive Vision System
- Grounding of objects and grasping affordances
- **Relation to Marr and 'mainstream computer vision'**
- Exercise

'Main Stream Computer Vision'

- **Two steps**
  1. Some feature extraction (e.g., SIFT)
  2. Learning a classifier on a large set of data
- **Example: Bag of Words**



Classifier



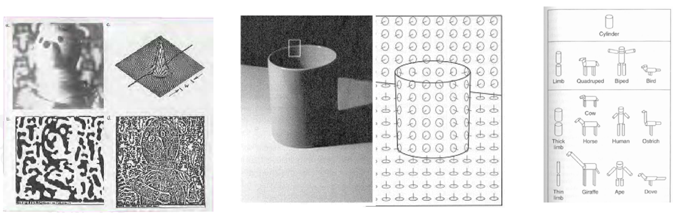
- **Two Issues**
  - Words are not understood (implicit representation)
  - And hence the relations between these words neither

Six properties of the ECV System

- **ECV is an in-between processing stage that bridges early vision and high-level cognitive vision**
- **ECV computes a generic and task independent scene representation which facilitates higher level processes**
- **ECV computes explicit and interpretable contextually embedded visual information**
  - ECV represents local image and scene structures according to their 2D as well as their underlying (statistically justified) 3D properties, leading to four different kinds of local descriptors with appropriate contextual embeddings
- **ECV makes use of the redundancy of visual information to disambiguate ambiguous local processing of visual information**
- **ECV makes use of a carefully selected choice of hard-wired knowledge as well as appropriate degrees of freedom in the design of local descriptors and their relations**
- **ECV provides a natural interface for `Vision for Action'**

Marr 1982

- **3 Stages**
  - Primal Sketch: Multi-scale Edge Detection
  - 2.5D Sketch: Viewer centered Scene Representation
  - 3D Sketch: Object Centered Representation



Relation to Marr

- **Hierarchical processing**
  - As in Marr's approach the ECV make use of an explicit hierarchical processing scheme.
- **Different structures in different streams**
  - As in Marr's approach we use different streams for different sub-aspects of visual information.
    - We can make use of the research in the last three decades on understanding the processing of the different visual modalities.
  - A difference to the approach of Marr however is the use of condensed local descriptors as an intermediate step between early processing and the processing of global entities.
- **Disambiguation**
  - A major difference to Marr's approach is to explicitly account for the ambiguity of visual information at early stages and to realize disambiguation processes to arrive at more reliable information at higher levels.
- **Relation to Actions**
  - The ECV system does not solve 'the vision problem' but has to be understood as part of a cognitive bootstrapping agent which is able to perform actions in the world. These actions themselves can facilitate the actual extraction process (closed-loop systems).
- **Real-time processing**
  - The complexity of the involved processes is huge. Forty years ago Marr's paradigm was simply not realizable. However, increased computational power (also supported by special hardware such as GPUs [54]) allow for the realization of processes with a complexity as required in the ECV system.

Conclusion

- **Maybe**
  - a modification of Marr's approach is possible today!
  - Computer Vision should try again to build generic representations
- **The ECV system tries to do that**
  - However, it is not as OpenCV a selection of algorithm but a concrete visual representation
- **We like to make it available to the community**
  - Just send me an email: norbert@mmmi.sdu.dk

Thanks to

Overview

- Motivation from human vision
- An Early Cognitive Vision System
- Grounding of objects and grasping affordances
- Relation to ’mainstream computer vision’
- **Exercise**

Grounding of objects and grasping affordances: Definition of the problem

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

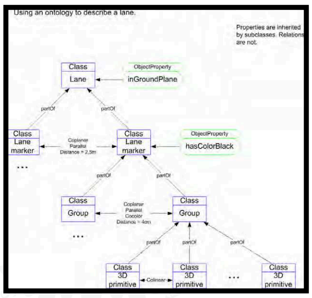
Learning from Scratch?

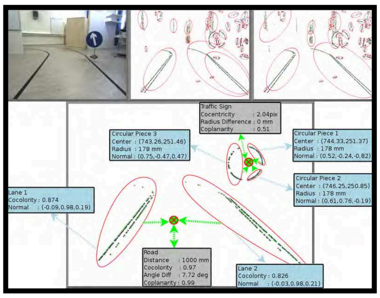
- **The system is able to learn objects and grasping affordances without any supervision**
- **Exercise**
  - Does the systen learn from scratch?
  - If no, what is the prior knowledge used?

- **This does not work in physics**
- **However, we can define boostrapping mechanisms which establish world knowledge through exploration based on**
  - combination of behaviours
  - a suitable body
  - and a sufficient degree of prior knowledge in the processing system

Some comments on autonomy?

Semantic Object and Scene Representation





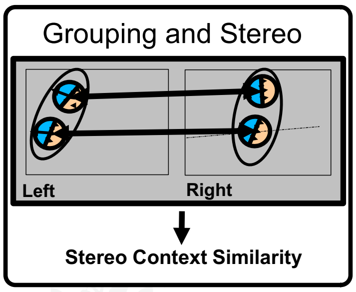
Lars B.W. Jensen and Emre Baseski

Signal-Symbol Loops

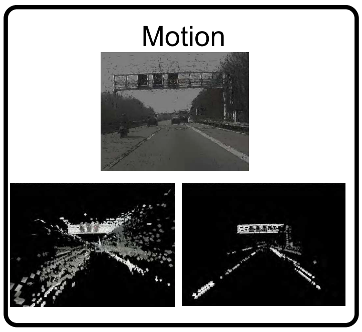
Summary

- **Early Cognitive Vision System**
  - Definition: ...
  - Motivated by human visual system
  - Rather complete in the edge domain
  - Other image structures and their relations become added ...
  - Make it available to the community
- **Structural properties of ECV system became exploited in two applications**
  - Grounding of objects and grasping affordances
  - Scene representations in Driver assistance domain

Disambiguation



Pugeault et al. (2006)



Pugeault et al. (2006)

Important Property of Representations:
Predictivity at the primitives is higher than at the original pixel level
Koenig and Kruger (2006)

Cognitive Vision Lab
University of Southern Denmark
Odense

Cognitive Vision Lab University of Southern Denmark

Eight properties of the ECV System

1. ECV is an in-between processing stage that bridges early vision and high-level cognitive vision
2. ECV computes a generic and task independent scene representation which facilitates higher level processes
3. ECV computes symbolic (explicit and interpretable) contextually embedded visual information
4. ECV makes use of the redundancy of visual information to disambiguate ambiguous local processing of visual information
5. ECV represents local image and scene structures according to their 2D as well as their underlying (statistically justified) 3D properties, leading to four different kinds of local descriptors with appropriate contextual embeddings
6. The ECV system realizes three kinds of disambiguation processes making use of hierarchical congruency between levels
7. ECV reflects the bias/variance dilemma by making use of a carefully selected choice of hard-wired knowledge as well as appropriate degrees of freedom in the design of local descriptors and their relations
8. ECV provides a natural interface for `Vision for Action‘

Representations for classical and knowledge-level planning

Ron Petrick

School of Informatics
University of Edinburgh
Edinburgh, Scotland, United Kingdom
rpetrick@inf.ed.ac.uk

CogX Spring School, Ljubljana, Slovenia

25 April 2010

Outline

1. What is planning?
2. Classical STRIPS-style planning
3. Belief space/knowledge-level planning
4. PKS: Planning with Knowledge and Sensing
5. Planning and natural language
6. Related problems

⇒ This talk will focus on representation.

What I won't talk about

- Probability
- Preferences
- Hierarchical planning
- Spatial reasoning
- Cost optimization
- Temporal planning
- Control knowledge
- SAT planning
- Heuristic search
- ...

What is planning?

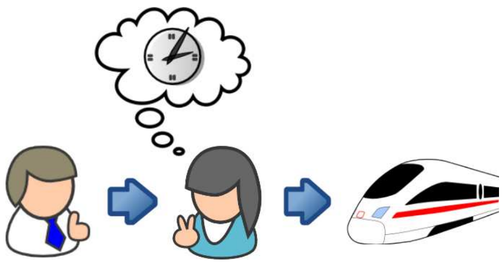
What is planning?

"I want to take the train from Ljubljana to Munich."

Go to the station, buy a ticket, check the departure board for track information, go to the track, board the train, . . .

What is planning?

"I want to take the train from Ljubljana to Munich."



Go to the station, buy a ticket, ask someone for track information, go to the track, board the train, . . .

Two plans

Plan 1	Plan 2
Go to the station	Go to the station
Buy a ticket	Buy a ticket
Check the departure board	Ask someone for information
Go to the track	Go to the track
Board the train	Board the train
...	...

- Physical “task” actions (e.g., “go to the station”).
- Observational/information gathering actions (e.g., “check the departure board”).
- Dialogue (speech) actions (e.g., “ask someone for information”).

Automated planning

- Automated planning techniques are good at building goal-directed plans of action under many challenging conditions, given a suitable description of a domain.
- A planning problem consists of:
  1. A representation of the properties and objects in the world and/or the agent’s knowledge, usually described in a logical language,
  2. A set of state transforming actions,
  3. A description of the initial world/knowledge state,
  4. A set of goal conditions to be achieved.
- A plan is a sequence of actions that when applied to the initial state transforms the state in such a way that the resulting state satisfies the goal conditions.

STRIPS (Fikes & Nilsson 1971)

- A world state is represented by a closed world database $\mathcal{D}$ and negation as failure. This gives rise to a simple and efficient way of representing facts about the world:
  - ϕ is true if $\phi \in \mathcal{D}$,
  - $\neg\phi$ is true if $\phi \notin \mathcal{D}$, where ϕ is a ground atom.
- Actions are the sole means of change in the world.
- An action’s preconditions specify the conditions under which an action can be applied, evaluated against $\mathcal{D}$ (qualification problem).
- An action’s effects specify the changes the action makes to the world, applied by updating $\mathcal{D}$.
  - Add list: properties A makes true are added to $\mathcal{D}$,
  - Delete list: properties A makes false are removed from $\mathcal{D}$,
  - All other properties are unchanged (frame problem) (McCarthy & Hayes 1969).

STRIPS actions

Action	Preconditions	Add list	Delete list
$pickup(x)$	$handEmpty$ $onTable(x)$	$holding(x)$	$handEmpty$ $onTable(x)$
$dropInBox(x, y)$	$holding(x)$ $box(y)$	$inBox(x, y)$ $handEmpty$	$holding(x)$ $empty(y)$

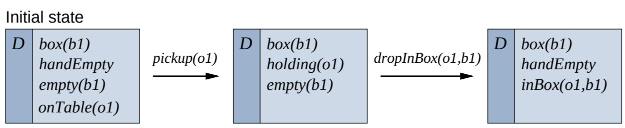
- Action operators: $pickup$, $dropInBox$
- Properties: $handEmpty$, $onTable$, ...
- Objects: $b1$, $o1$, ...

STRIPS actions...(2)

Action	Preconditions	Add list	Delete list
$pickup(x)$	$handEmpty$ $onTable(x)$	$holding(x)$	$handEmpty$ $onTable(x)$
$dropInBox(x, y)$	$holding(x)$ $box(y)$	$inBox(x, y)$ $handEmpty$	$holding(x)$ $empty(y)$

- Actions are state transforming.
- Applying the effects of an instantiated action A to a database $\mathcal{D}$ updates the database to produce a new database (denoting a new state) resulting from the execution of A.
- We can generate plans by chaining together fully instantiated actions.

Planning with STRIPS actions



- Example: achieve a state where $inBox(o1, b1)$ holds.
- The resulting plan is the action sequence:

$$[pickup(o1),\ dropInBox(o1, b1)].$$

Recent STRIPS planning

- STRIPS forms the core of PDDL (McDermott *et al.* 1998), the language of many modern planners and the language of the International Planning Competition (`http://ipc.icaps-conference.org/`).
- Many popular planners use a form of STRIPS, e.g., FF (Hoffmann & Nebel 2001), LAMA (Richter & Westphal 2008), SGPlan (Hsu *et al.* 2006), . . .
- "Classical" planning with complete knowledge and deterministic action effects is the most popular track in the International Planning Competition. E.g., see the results of the IPC 2008 deterministic track at `http://ipc.informatik.uni-freiburg.de/`.

Planning Domain Definition Language (PDDL)

- Version 1.2 (IPC-1998): STRIPS + ADL (conditional effects, general preconditions),
- Version 2.1 (IPC-2002): temporal planning, numeric state variables, durative actions,
- Version 2.2 (IPC-2004): derived predicates, times literals,
- Version 3.0 (IPC-2006): trajectory constraints (temporal logic), preferences (soft constraints),
- Version 3.1 (IPC-2008): functional state variables.

International Planning Competition (IPC)

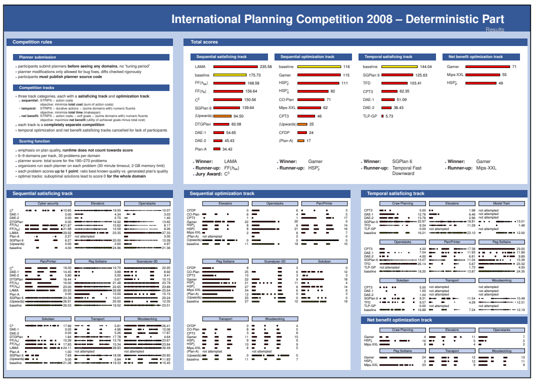


Image: (Helmert, Do, & Refanidis 2008) from `http://ipc.informatik.uni-freiburg.de/`

Action compilation

- Compile complex actions into ordinary PDDL actions (McIlraith & Fadel 2002, Claßen *et al.* 2007, Baier & McIlraith 2006, Baier, Fritz, & McIlraith 2007).

```
action processDataset(?d)
  precondition:
    dataset(?d) and
    not(processedDataset(?d))
  effect:
    i = 1 ;
    while (i <= size(?d))
      count = count + i ;
      i = i + 1
    endWhile ;
    processedDataset(?d)
endAction
```

Compilation example: (Petrick 2009)

```
(:action processDataset
  :parameters (?d)
  :precondition
    (and (not (context-loop))
         (dataset ?d)
         (not (processedDataset ?d)))
  :effect
    (and (assign (i) 1)
         (context-loop)
         (context-loop-params ?d)))

(:action processDataset-inLoop
  :parameters (?d)
  :precondition
    (and (context-loop)
         (context-loop-params ?d)
         (<= (i) (size ?d)))
  :effect
    (and (increase (count) (i))
         (increase (i) 1)))

(:action processDataset-endLoop
  :parameters (?d)
  :precondition
    (and (context-loop)
         (context-loop-params ?d)
         (not (<= (i) (size ?d))))
  :effect
    (and (processedDataset ?d)
         (not (context-loop))
         (not (context-loop-params ?d))))
```


Example: PACO-PLUS project (EU FP6)

"Perception, Action, and Cognition through learning of Object-Action Complexes"

`http://www.paco-plus.org/`

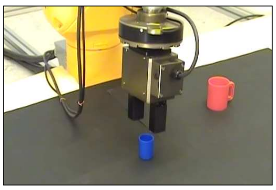


Image: Kraft & Krüger, University of Southern Denmark

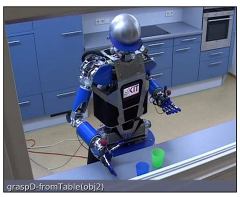


Image: Asfour *et al.*, Universität Karlsruhe

- Multiple robot platforms.
- Task and dialogue planning.
- Portions of the planning-level representation are induced from the robot's interaction with the real world.

PACO-PLUS: task planning in a kitchen domain

"Classical" planning actions

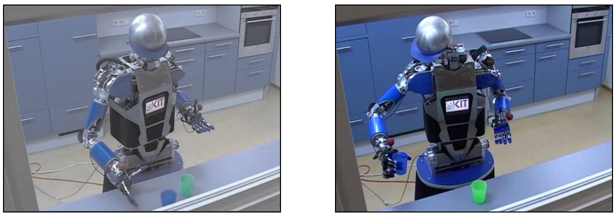
$grasp(o, l, h)$	Grasp object o from l using gripper h.
$graspFromEdge(o, l, h)$	Grasp object o from the edge of l using gripper h.
$move(l_1, l_2)$	Move the robot from location l_1 to location l_2.
$nudgeToEdge(o, l, h)$	Nudge flat object o to the edge of l using gripper h.
$open(l, h)$	Open l with gripper h.
$openPartial(l, h)$	Partially open l with gripper h.
$openComplete(l, h)$	Finish opening l with gripper h.
$close(l, h)$	Close l with gripper h.
$passObject(o, h_1, h_2)$	Pass object o from gripper h_1 to h_2.
$placeUpright(o, l, h)$	Put object o upright at l using gripper h.
$putDown(o, l, h)$	Put object o down at l using gripper h.
$putIn(o, l, h)$	Put object o into l using gripper h.
$removeFrom(o, l, h)$	Remove object o from l using gripper h.

Example plan: ensure the *applejuice* is in the *fridge*:

placeUpright(*applejuice*, *sideboard*, *lefthand*),
grasp(*applejuice*, *sideboard*, *righthand*),
move(*sideboard*, *fridge*),
openPartial(*fridge*, *lefthand*),
passObject(*applejuice*, *righthand*, *lefthand*),
openComplete(*fridge*, *righthand*),
putIn(*applejuice*, *fridge*, *lefthand*),
close(*fridge*, *lefthand*).

(Petrick *et al.* 2009)

PACO-PLUS: task planning in a kitchen domain...



Images: Asfour *et al.*, Universität Karlsruhe

- Classical STRIPS-style planning is often sufficient for many task-based domains in PACO-PLUS.
- The ability to model a particular task depends on the level of abstraction.

Belief space/knowledge-level planning

Recall: properties of classical planning

- A completely specified initial world state (closed world assumption).
- Actions are deterministic and map world states to world states.

⇒ The completeness of the world state is preserved.

⇒ Agent do not need to sense the environment.

⇒ Not always (usually?) realistic.

Action and belief/knowledge

- In many real-world domains, an agent may not have complete knowledge of its environment (e.g., open world).
- Examples
  - Robot with sensors exploring an unknown building.
  - Software agent in an operating system domain.
  - Agents interacting with other agents using natural language.
- Physical actions not only have effects that change the state of the world, but also the mental state of the agent performing the action.
- Actions may have nondeterministic effects (e.g., sensing actions).
- What if an agent performs actions that change it's mental state but not the state of the world? What if the agent believes something that isn't true in the world?

⇒ Representing and reasoning about belief/knowledge.

Types of open world planning

- Conformant planning
  - Incomplete world states,
  - No sensing actions.
- Contingent planning
  - Incomplete world states,
  - Sensing actions.
- Planning under uncertainty
  - Usually probabilistic.
- Also see the terms: "planning with incomplete information and sensing", "belief space planning", "planning with knowledge", ...

Representing an agent's knowledge

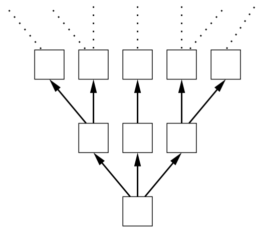
- How can we represent an agent's incomplete knowledge about the state of the world for planning?
- Issues to consider:
  - What types of knowledge should be represented? Restrictions? Agents may have knowledge of facts, functions, universals, etc.
  - How do we represent the effects of sensors? At plan time, a planner must reason about information that will become known at some point in the future.
  - Does this representation enable practical plan generation? General reasoning with expressive representations can lead to intractability.
- Many approaches in the knowledge representation (KR) community.

Possible worlds

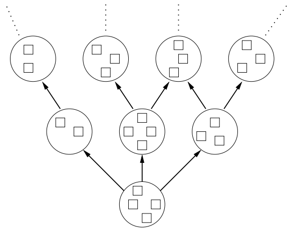
- The incomplete knowledge of an agent can be modelled by a set of worlds, each a possible version of how real world configured.
- Each world can be thought of as a first-order model.
- Worlds are related by an accessibility relation.
- Intuitively:
  - Knowing ϕ to be true $\approx$ ϕ is true in every possible world.
  - Not knowing whether or not ϕ is true $\approx$ there exist worlds in which ϕ is true and worlds in which ϕ is false.
- The representation can be quite general, however, there is a computational drawback from working directly with possible worlds:

n atomic formulae $\Rightarrow$ potentially 2^n possible worlds.

Belief space planning



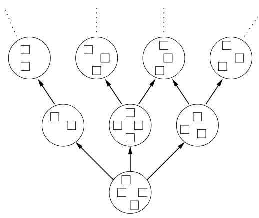
Classic state space planning



Belief state planning

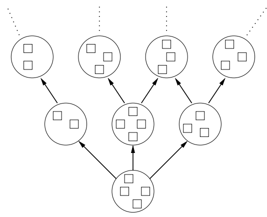
- Many planning approaches model belief states—sets of ordinary states representing the planner's beliefs—and construct plans by reasoning about these states (Smith & Weld 1998, Weld *et al.* 1998, Cimatti & Roveri 2000, Bertoli *et al.* 2001, Brafman & Hoffmann 2004, Hoffmann & Brafman 2005, Bryce 2007, . . .).

Belief space planning...(2)



- Verifying action preconditions: must check truth in each ordinary state.
- Applying action effects: update each ordinary state with effects.
- What about belief/knowledge? Must consider changes across all ordinary states in a belief state.

Belief space planning...(3)



- State space explosion is a potential problem when the planner's knowledge is particularly uncertain.
- Open question: what is the best way to represent and reason with these sets? Heuristic search, model checking, compression techniques, . . .

Modal logics of belief/knowledge

- Belief/knowledge (and action) can be modelled using a logical representation, and understood in terms of possible worlds (Hintikka 1962, Kripke 1972, Moore 1985, Fagin *et al.* 1995).
- Syntactically: an operator K is added to an ordinary first-order language, extending it by the rule: if ϕ is a formula then so is $K(\phi)$. Intuitively, $K(\phi)$ means "ϕ is known".
- Semantically: language can be interpreted over a collection of possible worlds, each a first-order model.
- A non-modal formula ϕ is true at a particular world w ($w \models \phi$) iff it is true according to standard rules for interpreting first-order formulae.
- A formula of the form $K(\phi)$ is interpreted to be true at w iff ϕ is true at every world accessible from w.
- Problem: general reasoning is intractable.

Restricted logics

- Alternative approaches have addressed the reasoning problem by placing restrictions on the syntactic form of the underlying logic, and avoiding the use of possible worlds.
  - Interval-valued epistemic fluents (Funge 1998): a special fluent $\mathcal{I}_F$ measures the (un)certainty of a corresponding fluent F by maintaining an interval of possible values for F. E.g., $\mathcal{I}_F = \langle c, d \rangle$ means that F must take a value between c and d.
  - Belief fluents (Demolombe & Parra 2000): associate a pair of fluents KF ("F is known to be true") and $K\neg F$ ("F is known to be false") with each ordinary fluent F and directly model how KF, $K\neg F$, and F change due to action.
  - 0-approximation (Son & Baral 2001): represent knowledge by sets of predicates known to be true and known to be false, to approximate the knowledge modelled by a set of possible worlds.
- Other approaches in the literature, e.g., (Soutchanski 2001, Petrick & Levesque 2002, Liu & Levesque 2005, Vassos & Levesque 2007, . . .)

Compilation approaches

- A recent trend in the planning community has been to find ways of transforming "difficult" classes of planning problems into "simpler" problems that are more easily solved (e.g., using classical planners).
- Compile belief space problems into a form that can be used with ordinary PDDL planners like FF.
  - Conformant domains (Palacios & Geffner 2009),
  - Contingent domains (Albore *et al.* 2009).
  - Closely related to ideas in approaches like (Son & Baral 2001, Petrick & Levesque 2002, ...)
- No guarantee technique will work on all domains; transformed problem may be an approximation of the original problem.
- Good performance on standard benchmarks. See, e.g., (Palacios & Geffner 2009).

PKS: Planning with Knowledge and Sensing

Planning at the knowledge level

- Main idea: build plans based on what the planner knows.
  - What types of knowledge do we need?
  - How do we represent this knowledge?
  - How do we reason and plan with this knowledge?
- Model actions by the changes they make to the planner's knowledge state, rather than the world state.
- Theory: use a restricted modal logic of knowledge.
- Practice: use an extended STRIPS representation.
- Focus:
  - Correct but incomplete knowledge,
  - Actions that can sense and manipulate the environment,
  - Emphasis on contingent planning.

⇒ PKS – "Planning with Knowledge and Sensing" (Petrick & Bacchus 2004, 2004).

What knowledge does PKS represent?

- Relational facts about the world

$$handEmpty, inDir(gcc, /usr/bin), \neg rainy, \ldots$$

- Functional information

$$combo(safe) \neq 23{-}42{-}12, parentDir(parentDir(dirA)) = dirC, \ldots$$

- Disjunctive information

"I know that the light switch is on or off."

- Plan time knowledge that will be resolved at execution time

"After checking the thermometer I will come to know the temperature."

- Local closed world information (Etzioni *et al.* 1994)

"I know what objects are in the box."

- ...

How does PKS represent knowledge?

- Use a collection of databases: K_f, K_w, K_v, K_x, LCW.
- Each database is restricted to a particular type of knowledge.
- Knowledge is assumed to be correct, but is incomplete.
- The contents of each database have a fixed translation to formulae in a modal logic of knowledge.
- Given a set of databases (**DB**) ⇒ formal translation defines the planner's knowledge state (**KB**).
- Rather than modelling sets of possible worlds, the modal formulae directly represent the planner's knowledge state.
- Planning: actions update **DB** ⇒ update **KB**.

K_f database

- Knowledge of positive and negative facts

$$rainy, \neg inDir(gcc, music), combo(safe) = c_1, temperature \neq 32.$$

- Similar to a standard STRIPS database.
- Not closed world! Negative facts must be explicitly represented.

Translation

For $\ell \in K_f$, **KB** includes the formula

$$K(\ell).$$

K_w database

- Knowledge of binary sensing effects

$$\phi \in K_w : \text{the planner "knows whether" } \phi.$$

- Example: sense whether $boxA$ is open or not,

$$open(boxA) \in K_w.$$

  - At plan time the planner knows that it will come to know whether the box is open or not.
  - At execution time it will have definite knowledge of the actual outcome (i.e., the disjunction will be resolved).

Translation

For $\phi(\vec{x}) \in K_w$, **KB** includes the formula

$$(\forall \vec{x}).K(\phi(\vec{x})) \vee K(\neg\phi(\vec{x})).$$

K_v database

- Knowledge of function values, multi-valued sensing effects

$$f \in K_v : \text{the planner "knows the value" of } f.$$

- Example: read a combination for a safe from a piece of paper,

$$combo(safe) \in K_v.$$

  - At plan time the planner knows that it will come to know the value of the combination.
  - At execution time the planner will have definite knowledge of the actual combination (i.e., a number).

Translation

For $f(\vec{x}) \in K_v$, **KB** includes the formula

$$(\forall \vec{x})(\exists v).K(f(\vec{x}) = v).$$

K_x database

- Exclusive-or knowledge

$$(\ell_1|\ell_2|\ldots|\ell_n) \in K_x : \text{ exactly one of the } \ell_i \text{ must be true.}$$

- Example: either c_1 or c_2 is the combination of the safe,

$$combo(safe) = c_1 \mid combo(safe) = c_2 \in K_x.$$

- Additional information may "resolve" or "simplify" K_x knowledge. E.g., in the above example, coming to know c_2 is not the combination of the safe means we can conclude c_1 is the combination of the safe.

Translation

For $(\ell_1|\ell_2|\ldots|\ell_n) \in K_x$, **KB** includes the formula

$$K(\bigvee_{i=1}^{n} \ell_i \wedge (\neg\ell_1 \wedge \ldots \wedge \neg\ell_{i-1} \wedge \neg\ell_{i+1} \wedge \ldots \wedge \neg\ell_n)).$$

LCW database

- Local closed world information (Etzioni *et al.* 1994).
- LCW formulae assert that K_f contains a complete list of all objects satisfying the formula. The planner can conclude that an object does not satisfy the formula if it isn't explicitly listed in K_f.
- Example: the planner knows all the objects in $boxA$,

$$inBox(x, boxA) \in LCW.$$

If $inBox(cupB, boxA) \notin K_f$ then conclude $K(\neg inBox(cupB, boxA))$.

Translation

For every formula $\phi(\vec{x}) \in LCW$, $\phi(\vec{x}) = \alpha_1(\vec{x}) \wedge \ldots \wedge \alpha_k(\vec{x})$, $C = \{\vec{c} : \alpha_i(\vec{x}/\vec{c}) \in K_f, 1 \le i \le k\}$ (C is the set of tuples of constants explicitly listed in K_f as satisfying ϕ), **KB** includes the formula:

$$(\forall \vec{x}). \bigwedge_{\vec{c} \in C} \neg(x_1 = c_1 \wedge \ldots \wedge x_n = c_n) \supset K(\neg\phi(\vec{x}/\vec{c})).$$

PKS knowledge states

- Given a set of databases (**DB**), the formal translation defines the planner's knowledge state (**KB**).
- Restrictions on databases contents means that there are restrictions on the kinds of knowledge that can be modelled.
- Cannot model certain types of knowledge, e.g., general disjunctions

$$K(P(a) \vee Q(b, c)).$$

- Tradeoff expressiveness for tractable reasoning.
  - Cannot model certain planning problems.
  - Avoid reasoning directly about individual possible worlds.

How does PKS reason about knowledge?

- A primitive query language is used to ask simple questions about the planner's knowledge state
  - $K(\alpha)$, is α known to be true?
  - $K_v(t)$, is the value of t known?
  - $K_w(\alpha)$, is α known to be true or known to be false?
  - The negation of the above queries.
- A sound, but incomplete, inference procedure checks the database contents to determine the truth of a query.
- Reasoning is restricted but more than single database lookup.
- Used to evaluate preconditions, conditional rules, and goals.

PKS planning problems

- Planning problem: $\langle I, G, \mathcal{A}, \mathcal{U} \rangle$
- Initial knowledge state I: initial contents of databases.
- Goal conditions G: set of primitive queries
  - ⇒ Must be satisfied in every knowledge state that could arise from executing a plan.
- $\mathcal{A}$: non-empty set of action specifications.
- $\mathcal{U}$: set of domain specific knowledge update rules.

PKS actions ($\mathcal{A}$)

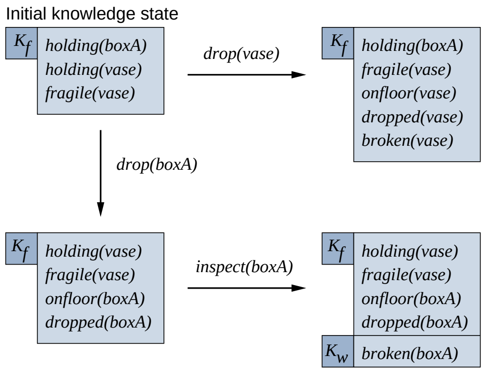
- Actions are described by their parameters, preconditions, and effects, based on an extension of STRIPS.
- Parameters: a set of variables bound to produce an action instance.
- Preconditions: conjunctive set of primitive queries, each of which must evaluate to true in order for the action to be applied.
- Effects: conditional effect rules of the form $C \Rightarrow E$,
  - Effect preconditions C: set of primitive queries,
  - Database updates E: list of *add* and *del* operations to any of the databases.

⇒ Easy to compute new knowledge states by forward chaining.
  - Evaluate preconditions against a set of databases **DB** (**KB**),
  - Effects update **DB** ⇒ update **KB**.

Example: *drop* and *inspect* actions

Action	Preconditions	Effects
$drop(x)$	$K(holding(x))$	$del(K_f, holding(x))$ $add(K_f, onFloor(x))$ $add(K_f, dropped(x))$ $del(K_f, \neg broken(x))$ $K(fragile(x)) \Rightarrow$ $\quad add(K_f, broken(x))$
$inspect(y)$		$add(K_w, broken(y))$

Example: *drop* and *inspect* actions...(2)



Domain specific update rules ($\mathcal{U}$)

- PKS also provides a facility to specify rules that could be triggered after any action is applied.
- Such rules can be used to describe properties like knowledge-level state invariants.
- Allow sets of conditional effect rules of the form $C \Rightarrow E$: effect E is triggered in any knowledge state where C holds.
- Example: come to know an object is fragile if it has been dropped and is broken,

$$K(dropped(x)) \wedge K(broken(x)) \Rightarrow add(K_f, fragile(x)).$$

- Such rules could also be included as part of each action specification.

How does PKS generate plans?

1. If the goals G are satisfied in the final **DB** of each plan branch then return the current plan.
2. Otherwise, choose a plan branch where the goals aren't satisfied and try to extend the branch. Choose an action A whose preconditions are satisfied in the final **DB** of that plan branch, apply A's effects to **DB**, and continue planning.
3. Alternatively, add a conditional branch point to a plan branch.
   - Add a 2-way branch by picking a ground instance $\alpha \in K_w$. Along one branch, add α to K_f; along the other branch, add $\neg\alpha$ to K_f.
   - Add an n-way branch by picking a ground instance $f \in K_v$, provided f is restricted by an entry $(f = c_1 | f = c_2 | \ldots | f = c_n) \in K_x$. Along each branch i, add $f = c_i$ to K_f.
   - Continue planning along each branch.
4. Fail if the plan cannot be expanded and the goals are not satisfied along all plan branches.

Plan correctness

- Plan correctness relies on two criteria (Levesque 1996)
  - Plan time: an agent must know it will have enough information at execution time for a plan to achieve the goals.
  - Execution time: an agent must have sufficient knowledge at every step of the plan to execute it.
- PKS satisfies Levesque's criteria
  - Goals are satisfied along every branch of a conditional plan.
  - Plan branches based on sensed K_w formulae resolve to definite knowledge at execution time.
  - The planner will have sufficient knowledge at the right time to determine which branch to execute.

PKS planning domains

Opening a safe

Action	Effects
$dial(x)$	$add(K_w, open)$ $del(K_f, \neg open)$ $add(K_f, justDialled = x)$ $K(combo = x) \Rightarrow add(K_f, open)$

- Build a plan to open a safe given a fixed set of possible combinations.
- Initial knowledge of combinations

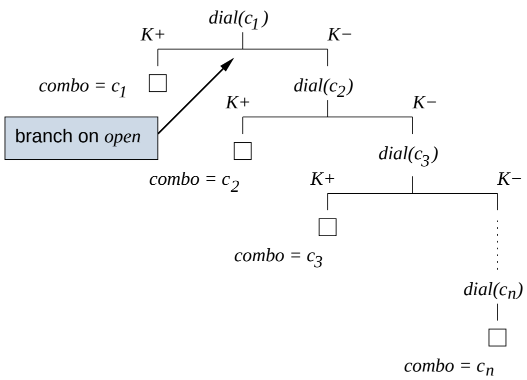
$$(combo = c_1 | combo = c_2 | \ldots | combo = c_n) \in K_x.$$

- Update rules

$$K(justDialled = x) \wedge K(\neg open) \Rightarrow add(K_f, combo \neq x),$$
$$K(justDialled = x) \wedge K(open) \Rightarrow add(K_f, combo = x).$$

- Goal: $K(open)$.

Opening a safe...(2)



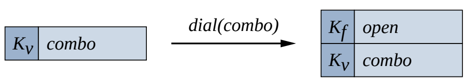
Opening a safe...(3)

Action	Effects
$dial(x)$	$add(K_w, open)$ $K(combo = x) \Rightarrow add(K_f, open)$

- Open a safe when the planner "knows the value" of the combination.
- Initial knowledge of combinations

$$combo \in K_v.$$

- Goal: $K(open)$.



The simple plan $dial(combo)$, where $combo$ is a function, suffices here.

Operating system domain

Action	Preconditions	Effects
$cd(d)$	$K(dir(d))$ $K(inDir(d, pwd))$	$add(K_f, pwd = d)$
$cd{-}up(d)$	$K(dir(d))$ $K(inDir(pwd, d))$	$add(K_f, pwd = d)$
$ls(f, d)$	$K(pwd = d)$ $K(file(f))$ $\neg K_w(inDir(f, d))$	$add(K_w, inDir(f, d))$

- Count the number of copies of a file in a directory tree where the directory tree is known but not necessarily the directory contents.
- There is at most one copy of a file in each directory.

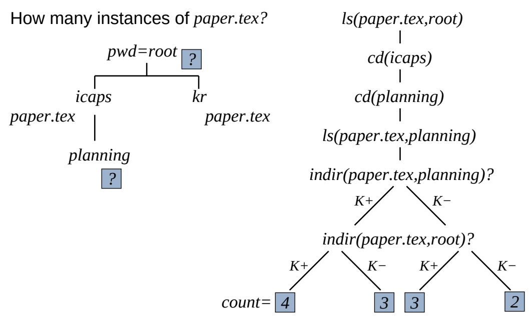
Operating system domain...(2)

Domain specific update rules

$\neg K(processed(f,d)) \wedge K(inDir(f,d)) \wedge K_v(size(f,d)) \Rightarrow$
$\quad t = [(sizeMax > size(f,d))?\ sizeMax : size(f,d)],$
$\quad add(K_f, sizeMax = t),$
$\quad add(K_f, count = count + 1),$
$\quad add(K_f, processed(f,d))$
$\neg K(processed(f,d)) \wedge K(inDir(f,d)) \wedge \neg K_v(size(f,d)) \Rightarrow$
$\quad add(K_f, sizeUnk = sizeUnk + 1),$
$\quad add(K_f, processed(f,d))$
$\neg K(processed(f,d)) \wedge K(\neg inDir(f,d)) \Rightarrow$
$\quad add(K_f, processed(f,d))$

- Domain encoding uses a function *count* and numerical expressions.
- If a directory has not been *processed*, and the file is known to be in the directory, increment the count.

Operating system domain...(3)



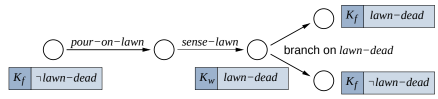
Poisonous liquid

Action	Effects
pour-on-lawn	$\neg K(\neg$*poisonous*$) \Rightarrow$ $del(K_f, \neg$*lawn-dead*$)$ $K($*poisonous*$) \Rightarrow$ $add(K_f,$ *lawn-dead*$)$
sense-lawn	$add(K_w,$ *lawn-dead*$)$

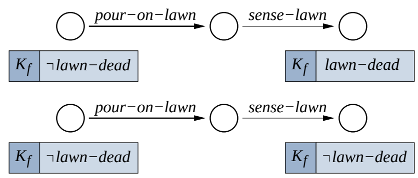
- What happens when an unknown bottle of liquid is poured on a healthy lawn? If the lawn is dead after pouring the liquid then the liquid must be poisonous and must have initially been poisonous.
- Postdictive reasoning (Sandewall 1994): perform additional reasoning about generating plan sequences. New conclusions don't follow solely from action effects but also from action non-effects.

Poisonous liquid...(2)

1. Generate a conditional plan

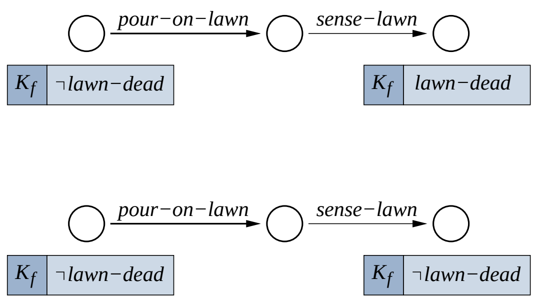


2. Form linearisations (possible execution branches)



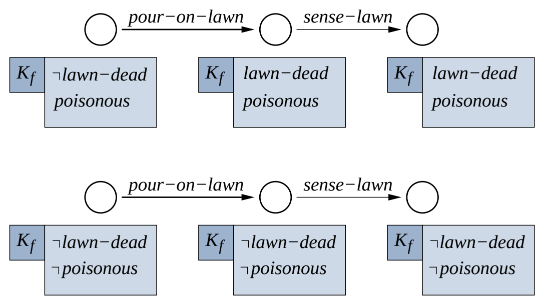
3. Augment states by postdiction rules

Poisonous liquid...(3)



$\Rightarrow$ Linearisations of conditional plan.

Poisonous liquid...(4)



$\Rightarrow$ Plan states augmented by postdiction.

PKS in PACO-PLUS

PACO-PLUS: object manipulation

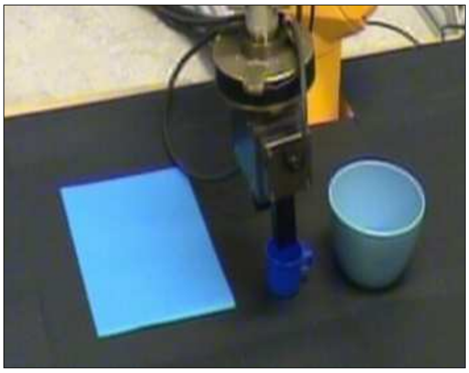


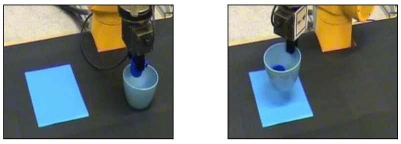
Image: Kraft & Krüger, University of Southern Denmark

PACO-PLUS: object manipulation...(2)

Action	Preconditions	Effects
$graspA{-}table(x)$	$K(clear(x))$ $K(gripperEmpty)$ $K(onTable(x))$ $K(reachableA(x))$ $K(radius(x) \geq minA)$ $K(radius(x) \leq maxA)$	$add(K_f, inGripper(x))$ $add(K_f, \neg gripperEmpty)$ $add(K_f, \neg onTable(x))$
$findout{-}open(x)$	$\neg K_w(open(x))$ $K(onTable(x))$	$add(K_w, open(x))$
...	...	...

(Petrick *et al.* 2009)

PACO-PLUS: remove all objects

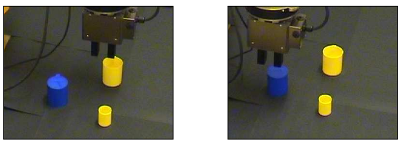


Images: Kraft & Krüger, University of Southern Denmark

- A simple plan to remove all objects:

$$graspD{-}table(obj1),\\ putInto{-}object(obj1, obj2),\\ graspB{-}table(obj2),\\ putAway(obj2).$$

PACO-PLUS: remove all “open” objects



Images: Kraft & Krüger, University of Southern Denmark

- A conditional plan to remove an open object:

$$\begin{array}{ll} \multicolumn{2}{l}{findout{-}open(obj1),} \\ \multicolumn{2}{l}{branch(open(obj1))} \\ K^+ : & \\ & graspA{-}table(obj1), \\ & putAway(obj1) \\ K^- : & \\ & nil. \end{array}$$

PACO-PLUS: locating objects in the kitchen

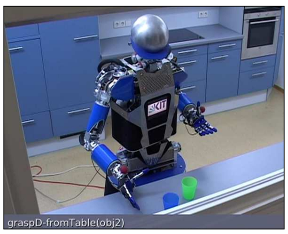


Image: Asfour *et al.*, Universität Karlsruhe

- “Search and retrieve” plans
- Example: find the cereal and bring it to the sideboard.
  - Check the sideboard,
  - Check the cupboard,
  - Check the stove,
  - ...

Planning and natural language

Natural language and planning

- Natural Language Generation (NLG) is a major subfield of natural language processing, concerned with computing natural language sentences or texts that convey a piece of information to a user.
- Dialogue systems are computer systems designed to carry out natural language conversations with human users. An important component of most dialogue systems is the dialogue manager which is responsible for making appropriate conversational moves.
- Can be viewed as problems involving actions, beliefs, and goals:

  A speaker tries to change the mental state of the hearer by applying actions that correspond to the utterance of words or sentences.

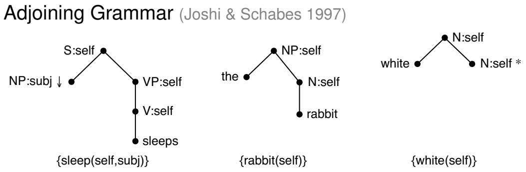
⇒ Obvious parallels to planning.

Natural language and planning...(2)

- The link between natural language and planning has a long tradition, e.g., (Perrault & Allen 1980, Appelt 1985, Clark 1996, Stone 2000), including early BDI-based approaches, e.g., (Litman & Allen 1987, Cohen & Levesque 1990, Grosz & Sidner 1990), . . .
- Early approaches suffered due to inefficient planning techniques.
- Recent work has tended to separate "task" planning from other types of natural language planning and has focused on specialized approaches, e.g., finite state machines, information state, rule-based approaches to speech act theories, dialogue games, . . .
- There has been a renewed interest in applying modern planning techniques to natural language problems, e.g., (Koller & Stone 2007, Benotti 2008, Brenner & Kruijff-Korbayová 2008, Koller & Petrick 2008).

Sentence generation as planning

- Sentence generation is the problem of computing, from a grammar and semantic representation, a sentence that expresses this meaning. E.g., "The white rabbit sleeps."
- Tree Adjoining Grammar (Joshi & Schabes 1997)



(Koller & Stone 2007, Koller & Petrick 2008)

Elementary trees contribute semantic content.

- Trees are instantiated by substituting individuals from a knowledge base for semantic roles. E.g., a and b are rabbits, a is white, b is brown, and e is an event in which a sleeps.

Sentence generation as planning...(2)

- Can be translated into a planning problem (Koller & Stone 2007).
- Classical planning actions model operations that add elementary trees to a derivation (Koller & Petrick 2008), e.g.,

```
(:action add-sleeps
    :parameters (?u - node
                 ?xself - individual
                 ?xsubj - individual)
    :precondition
       (and (subst S ?u)
            (referent ?u ?xself)
            (sleep ?xself ?xsubj))
    :effect
       (and (not (subst S ?u))
            (expressed sleep ?xself ?xsubj)
            (subst NP (subj ?u))
            (referent (subj ?u) ?xsubj)
            (forall (?y - individual)
               (when (not (= ?y ?xself))
                  (distractor (subj ?u) ?y)))))
```

- A plan must satisfy the syntactic and semantic linguistic requirements

$$[add{-}sleeps(root,a), add{-}rabbit(subj(root),a), add{-}white(subj(root),a)].$$

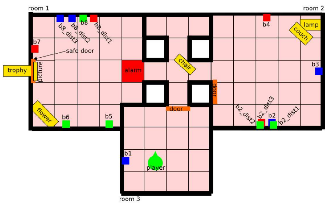
Planning in instruction giving

- GIVE: "Generating Instructions in Virtual Environments" (Koller *et al.* 2007).
- Build a system capable of producing natural language instructions to guide a human user in performing some task in a virtual environment.

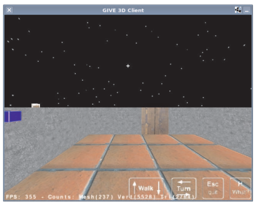
  `http://www.give-challenge.org/research/`

- A theory-neutral task that tests all components of an NLG system.
- A GIVE problem is very similar to a Grid planning problem (IPC 1998).
  - Discretised tiles of equal size,
  - Users can turn 90º left or right, or move forward one tile,
  - Additional requirement to press buttons in the right order, reason about large numbers of world objects, and navigate complicated room shapes.

Planning in instruction giving...(2)



Example GIVE map



Virtual 3D GIVE client

(Koller & Petrick 2008)

Planning in instruction giving...(3)

- Classical planning actions describe task-level operations, e.g., "move", "turn", "press button", etc. (Koller & Petrick 2008).

```
(:action move
   :parameters (?from - position
                ?to - position
                ?ori - orientation)
   :precondition
      (and (player-pos ?from)
           (adjacent ?from ?to ?ori)
           (player-orient ?ori)
           (not-blocked ?from ?to)
           (not-alarmed ?to))
   :effect
      (and (not (player-pos ?from))
           (player-pos ?to)))
```

- Plans are often non-trivial (e.g., 108 steps using the example map)

$[turn{-}left(north, west), move(pos\_5\_2, pos\_4\_2, west), manipulate(b1, pos\_4\_2, off, on), \ldots].$

- Domain offers challenges beyond classical planning, e.g., plan summarisation and elaboration, strict run-time requirements, user's actions failing to match generated instructions, etc.

Situated dialogue and tacit actions (Benotti 2007, 2008)

- Situated dialogue-based interactions can be advanced using dialogue actions, physical actions, and sensing actions.
- In particular, actions can be explicit or tacit.
- Tacit physical actions can be inferred using classical planning, while tacit sensing actions can be inferred using a knowledge-level planning approach, e.g., PKS.
- Application to text adventure games.

Action	Preconditions	Effects
$trykey(x, y)$	$K(accessible(x))$ $K(locked(x))$ $K(key(y))$ $K(inventory\_obj(y))$	$add(K_w, fits\_in(x, y))$
...	...	...

$take(silver\_key, table),$
$take(golden\_key, table),$
$trykey(silver\_key, door),$
$branch(fits\_in(silver\_key, door))$
$K^+:$
$\quad unlock(door, silver\_key),$
$\quad open(door)$
$K^-:$
$\quad unlock(door, golden\_key),$
$\quad open(door).$

Example PKS actions and a conditional plan in a text adventure domain (Benotti 2007, 2008)

PKS and dialogue planning

PKS dialogue planning (Steedman & Petrick 2007)

Plan 1	Plan 2
Go to the station	Go to the station
Buy a ticket	Buy a ticket
Check the departure board	Ask someone for information
Go to the track	Go to the track
Board the train	Board the train
...	...

Observational action step　　Dialogue step (speech act)

⇒ Both actions serve as information gathering steps in the plan.

⇒ Can we reason about dialogue acts in the same way as ordinary actions? Can we use the same machinery for planning?

PKS dialogue planning with speech acts

- Motivation: some actions like *ask* and *tell* can be modelled as sensing actions. We can represent certain dialogue problems as instances of planning with incomplete information and sensing.
- Can we apply knowledge-level planning techniques to this problem?
  - Dialogues involve multiple participants,
  - Actions correspond to speech acts, e.g., *ask*, *tell*, ...
  - Plans specify mixed-initiative discourse among participants.
- Formal representations exist: many logical languages for reasoning about actions and change, e.g., (Moore 1985; Scherl & Levesque 1994, 2003; Stone 1998; Steedman 1997, 2002; ...).

⇒ What kinds of changes do we need to make to PKS?
How tractable is the reasoning?
In what kinds of domains can we apply these techniques?

Participants and common ground

- We use labels (modalities) for referencing dialogue participants and common ground.

$[\mathrm{S}]$	Speaker supposition
$[\mathrm{H}]$	Hearer supposition
$[\mathrm{X}], [\mathrm{Y}], \ldots$	Other participant/agent suppositions
$[\mathrm{C_{XY}}]$	Common ground between X and Y

Examples

$[\mathrm{S}]\, p$	"The speaker supposes p."
$[\mathrm{S}]\, [\mathrm{H}]\, p$	"The speaker supposes the hearer supposes p."
$[\mathrm{H}]\, [\mathrm{C_{SH}}]\, [\mathrm{S}]\, p$	"The hearer supposes it's common ground between the speaker and hearer that the speaker supposes p."

Knowledge assertions

- Use restricted PKS knowledge assertions as the basis (content) of knowledge expressions, i.e.,

Kp	"Know p",
$K_v t$	"Know the value of t",
$K_w p$	"Know whether p".

- Use K_v and K_w to represent indefinite information, such as the information returned by sensing actions.
- Combine labels with knowledge assertions.

Examples

$[\mathrm{S}]\, \neg K combo(safe) = c_1$
$[\mathrm{S}]\, [\mathrm{H}]\, K_v track$
$[\mathrm{S}]\, [\mathrm{C_{SH}}]\, K_w open(boxA)$

Reasoning with labelled knowledge

- Rules for reasoning about speaker-hearer suppositions and common ground modalities (Steedman & Petrick 2007):

A1.	$[\mathrm{X}]\, \phi \Rightarrow \phi$	Supposition Veridicality
A2.	$[\mathrm{X}]\, \neg\phi \Rightarrow \neg [\mathrm{X}]\, \phi$	Supposition Consistency
A3.	$\neg [\mathrm{X}]\, \phi \Rightarrow [\mathrm{X}]\, \neg [\mathrm{X}]\, \phi$	Negative Introspection
A4.	$[\mathrm{C}]\, \phi \Leftrightarrow ([\mathrm{S}]\, [\mathrm{C}]\, \phi \wedge [\mathrm{H}]\, [\mathrm{C}]\, \phi)$	Common Ground
A5.	$[\mathrm{X}]\, [\mathrm{C}]\, \phi \Rightarrow [\mathrm{X}]\, \phi$	Common Ground Veridicality

- We require rules similar to these to augment PKS's standard inference procedure.

$\Rightarrow$ No specific conversational rules or intent recognition rules are used.

Knowledge requirements of *ask* and *tell*

R1. "If X doesn't know p and X knows Y does, X can ask Y about it."

$\Rightarrow$ Knowledge-level preconditions for *ask*.

R2. "If X asks Y about p, it makes it common ground X doesn't know it."

$\Rightarrow$ Knowledge-level effects for *ask*.

R3. "If X knows p, and X knows p is not common ground, X can tell Y p."

$\Rightarrow$ Knowledge-level preconditions for *tell*.

R4. "If X tells Y p, Y stops not knowing it and starts to know it."

$\Rightarrow$ Knowledge-level effects for *tell*.

Knowledge-level dialogue actions

Action	Preconditions	Effects
$ask(\mathrm{X},\mathrm{Y},p)$	$\neg [\mathrm{X}]\, p$ $[\mathrm{X}]\, [\mathrm{Y}]\, p$	$add([\mathrm{C_{XY}}]\, \neg [\mathrm{X}]\, p)$
$tell(\mathrm{X},\mathrm{Y},p)$	$[\mathrm{X}]\, p$ $[\mathrm{X}]\, \neg [\mathrm{C_{XY}}]\, p$	$add([\mathrm{Y}]\, p)$

- We can encode the knowledge requirements for speech acts like *ask* and *tell* in terms of their preconditions and effects.
- We can build plans by chaining together actions using our extra rules for reasoning about modalities.

Example: taking a train

F1. "If I know what time it is then I'll know what track my train is on."

$$[\mathrm{S}]\, K_v time \Rightarrow add([\mathrm{S}]\, K_v track)$$

F2. "I don't know what track my train leaves from."

$$[\mathrm{S}]\, \neg K_v track$$

F3. "I suppose you know what time it is."

$$[\mathrm{S}]\, [\mathrm{H}]\, K_v time$$

F4. "I suppose it's not common ground I don't know what time it is."

$$[\mathrm{S}]\, \neg [\mathrm{C_{SH}}]\, \neg [\mathrm{S}]\, K_v time$$

Dialogue plan generation

Goal: $[\mathrm{S}]\, K_v track$ (I need to know which track to go to)

(D1)	$\Rightarrow\ [\mathrm{H}]\, K_v time$	(F3),(A1)
(D2)	$\Rightarrow\ \neg\, [\mathrm{S}]\, K_v time$	(F2),(A2),(F1)
(D3)	$\Rightarrow$ Apply action: $\boldsymbol{ask}(\mathbf{S}, \mathbf{H}, \boldsymbol{K_v time})$	
(D4)	$\Rightarrow\ [\mathrm{C_{SH}}]\, \neg\, [\mathrm{S}]\, K_v time$	(D3),(R2)
(D5)	$\Rightarrow$ Apply action: $\boldsymbol{tell}(\mathbf{H}, \mathbf{S}, \boldsymbol{K_v time})$	
(D6)	$\Rightarrow\ [\mathrm{S}]\, K_v time$	(D5),(D2),(R4)
(D7)	$\Rightarrow\ [\mathrm{S}]\, K_v track$	(D6),(F1)

Plan: $[ask(\mathrm{S}, \mathrm{H}, K_v time),\ tell(\mathrm{H}, \mathrm{S}, K_v time)]$.

Dialogue plan generation...(2)

Goal: $[\mathrm{S}]\, K_v track$ (I need to know which track to go to)

(D1)	$\Rightarrow\ [\mathrm{S}]\, \neg\, [\mathrm{S}]\, K_v time$	(F2),(A2),(F1),(A3)
(D2)	$\Rightarrow\ [\mathrm{S}]\, \neg\, [\mathrm{C_{SH}}]\, \neg\, [\mathrm{S}]\, K_v time$	(F4)
(D3)	$\Rightarrow$ Apply action: $\boldsymbol{tell}(\mathbf{S}, \mathbf{H}, \neg\, [\mathbf{S}]\, \boldsymbol{K_v time})$	
(D4)	$\Rightarrow\ [\mathrm{C_{SH}}]\, \neg\, [\mathrm{S}]\, K_v time$	(R2)
	$\Rightarrow\ \ldots$	
	$\Rightarrow$ Apply action: $\boldsymbol{tell}(\mathbf{H}, \mathbf{S}, \boldsymbol{K_v time})$	
	$\Rightarrow\ \ldots$	

Plan: $[tell(\mathrm{S}, \mathrm{H}, \neg\, [\mathrm{S}]\, K_v time),\ tell(\mathrm{H}, \mathrm{S}, K_v time)]$.

Dialogue plan generation...(3)

Plan 1	Plan 2
$ask(\mathrm{S}, \mathrm{H}, K_v time)$	$tell(\mathrm{S}, \mathrm{H}, \neg\, [\mathrm{S}]\, K_v time)$
$tell(\mathrm{H}, \mathrm{S}, K_v time)$	$tell(\mathrm{H}, \mathrm{S}, K_v time)$

- Plan generation takes place in the space of multi-agent plans
  - No reasoning about other agents' goals or intentions,
  - Cannot guarantee other agents' actions.
- Approach is driven by the knowledge state, i.e., what the planning agent knows about the world and the other agents' beliefs.
- Both direct and indirect speech acts can be generated from the same mechanisms for reasoning about knowledge and common ground.

PACO-PLUS: dialogue planning

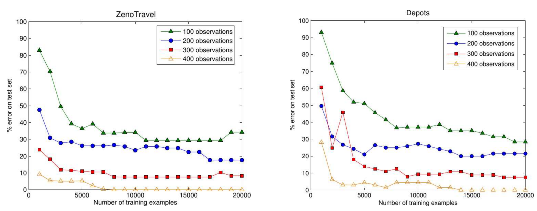
Robot1:	Let's make breakfast.	[goal-propose(breakfast)]
Robot2:	I don't know how to make breakfast.	[assert(¬know(breakfast))]
Robot1:	To make breakfast we must bring the cereal and the milk to the sideboard.	[explain(breakfast :- loc(cereal,sideboard) ∧ loc(milk,sideboard))]
Robot2:	Is the cereal at the sideboard?	[ask(loc(cereal,sideboard))]
Robot1:	No.	[tell(no)]
Robot2:	Where is the cereal?	[ask(loc(cereal, X))]
Robot1:	The cereal is in the cupboard.	[tell(loc(cereal,cupboard))]
Robot2:	Is the milk at the sideboard?	[ask(loc(milk,sideboard))]
Robot1:	No.	[tell(no)]
Robot2:	Where is the milk?	[ask(loc(milk, X))]
Robot1:	The milk is in the fridge.	[tell(loc(milk,fridge))]
Robot2:	Okay. I suggest I go to the cupboard, pickup the cereal, bring it to the sideboard, then go the fridge, pickup the milk, and bring it to the sideboard.	[assert-plan(move(sideboard,cupboard),...]

$\Rightarrow$ Use the same underlying plan generation mechanisms for both task planning and dialogue planning.

Related problems

Learning action representations

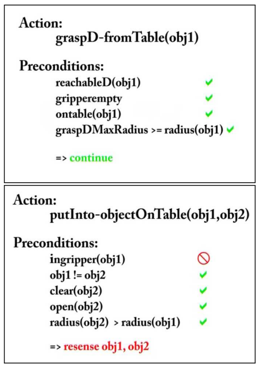
- Where do planning representations come from?
- Use machine learning techniques to learn actions from state snapshots. E.g., (Wang 1995; Amir & Chang 2008; Modayil & Kuipers 2008; Pasula, Zettlemoyer, and Kaelbling 2007; Mourão *et al.* 2009).



Example: results of learning STRIPS action effects in partially observable ZenoTravel and Depots domains using voted perceptrons (Mourão *et al.* 2009).

Plan execution monitoring

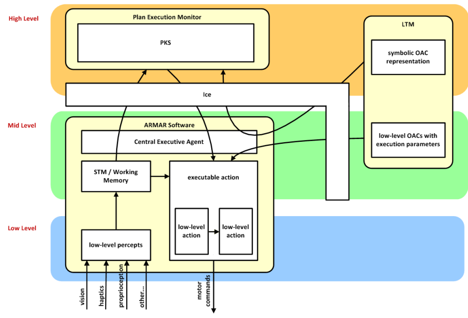
- How are plans executed after they're generated? How do we know whether plan execution was successful?
- Compare predicted/expected states against actual/observed states.
- Assesses whether to continue executing an existing plan, replan, (partially) resense the world, etc.
- Many approaches in the literature, often system dependent.



Images: Kraft & Krüger, University of Southern Denmark. See also (Petrick *et al.* 2009).

System architecture

- How does everything fit together?



Example: PACO-PLUS ARMAR system architecture (Petrick *et al.* 2010)

What else is there?

- Probability
- Preferences
- Hierarchical planning
- Spatial reasoning
- Cost optimization
- Temporal planning
- Control knowledge
- SAT planning
- Heuristic search
- ...

ICAPS 2009 at a glance

A word cloud of the top 100 words appearing in titles of ICAPS 2009 papers, generated at `http://www.wordle.net/`

References

- A. Albore, H. Palacios, and H. Geffner. 2009. A Translation-based Approach to Contingent Planning. *Proceedings of IJCAI-09*, 1623–1628.
- E. Amir and A. Chang. 2008. Learning partially observable deterministic action models. *Journal of Artificial Intelligence Research* 33:349–402.
- D. Appelt. 1985. *Planning English Sentences*. Cambridge, UK: Cambridge University Press.
- J. A. Baier, C. Fritz, and S. A. McIlraith. 2007. Exploiting procedural domain control knowledge in state-of-the-art planners. *Proceedings of ICAPS-07*, 26–33.
- J. A. Baier and S. A. McIlraith. 2006. On planning with programs that sense. *Proceedings of KR-06*, 492–502.
- L. Benotti. 2007. Incomplete knowledge and tacit action: enlightened update in a dialogue game. *Proceedings of the 11th Workshop on the Semantics and Pragmatics of Dialogue (DECALOG 2007)*, 17–24.
- L. Benotti. 2008. Accommodation through tacit sensing. *Proceedings of the 12th Workshop on the Semantics and Pragmatics of Dialogue (LONDIAL 2008)*, 75–82.
- P. Bertoli, A. Cimatti, M. Roveri, and P. Traverso. 2001. Planning in nondeterministic domains under partial observability via symbolic model checking. *Proceedings of IJCAI 2001*, 473–478.
- R. Brafman and J. Hoffmann. 2004. Conformant planning via heuristic forward search: a new approach. *Proceedings of ICAPS 2004*, 355–364.
- M. Brenner and I. Kruijff-Korbayová. 2008. A continual multiagent planning approach to situated dialogue. *Proceedings of the 12th Workshop on the Semantics and Pragmatics of Dialogue (LONDIAL 2008)*, 67–74.
- D. Bryce. 2007. Scalable Planning Under Uncertainty. PhD Dissertation, Department of Computer Science and Engineering, Arizona State University.

References...(2)

- A. Cimatti and M. Roveri 2000. Conformant planning via symbolic model checking. *Journal of Artificial Intelligence Research*, 13:305–338.
- H. Clark. 1996. *Using Language*. Cambridge, UK: Cambridge University Press.
- J. Claßen, P. Eyerich, G. Lakemeyer, and B. Nebel. 2007. Towards an integration of Golog and planning. *Proceedings of IJCAI-07*, 1846–1851.
- P. Cohen and H. Levesque. 1990. Rational interaction as the basis for communication. *Intentions in Communication*, 221–255. MIT Press.
- R. Demolombe and M. P. Pozos Parra. 2000. A simple and tractable extension of situation calculus to epistemic logic. *Proceedings of ISMIS-2000*, 515–524.
- O. Etzioni, K. Golden, and D. Weld. 1994. Tractable Closed World Reasoning with Updates. *Proceedings of KR-94*, 178–189.
- R. Fagin, J. Halpern, Y. Moses, and M. Vardi. 1995. *Reasoning About Knowledge*. MIT Press.
- R. E. Fikes and N. J. Nilsson. 1971. STRIPS: A new approach to the application of theorem proving to to problem solving. *Artificial Intelligence*, 2:189–208.
- J. Funge. 1998. Interval-valued epistemic fluents. *AAAI Fall Symposium on Cognitive Robotics*, 23–25.
- B. Grosz and C. Sidner. 1990. Plan for discourse. *Intentions in Communication*, 417–444. MIT Press.
- M. Helmert, M. Do, and I. Refanidis. 2008. Poster of the results of IPC 2008 (deterministic part). Available from: `http://ipc.informatik.uni-freiburg.de/Presentations`.

References...(3)

- J. Hintikka. 1962. *Knowledge and Belief*. Ithaca, NY: Cornell University Press.
- J. Hoffmann and R. Brafman. 2005. Contingent planning via heuristic forward search with implicit belief states. *Proceedings of ICAPS 2005*, 71–80.
- J. Hoffmann and B. Nebel. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302.
- C. W. Hsu, B. W. Wah, R. Huang, and Y. X. Chen. 2006. New features in SGPlan for handling soft constraints and goal preferences in PDDL 3.0. *5th International Planning Competition at ICAPS 2006*.
- IPC 1998. Webpage for the 1998 International Planning Competition. `http://ftp.cs.yale.edu/pub/mcdermott/aipscomp-results.html`.
- A. Joshi and Y. Schabes. 1997. Tree-Adjoining Grammars. *Handbook of Formal Languages*. Berlin: Springer-Verlag.
- A. Koller and R. Petrick. 2008. Experiences with Planning for Natural Language Generation. *Scheduling and Planning Applications woRKshop (SPARK 2008)*.
- A. Koller and R. Petrick. To appear. Experiences with Planning for Natural Language Generation. *Computational Intelligence — Special Issue on Scheduling and Planning Applications*.
- A. Koller and M. Stone. 2007. Sentence generation as planning. *Proceedings of the 45th ACL*, 336–343.
- S. Kripke. 1971. Semantical considerations on modal logic. *Reference and Modality*, Oxford University Press, 63–72.
- H. J. Levesque. 1996. What is planning in the presence of sensing? *Proceedings of AAAI-96*, 1139–1146.

References...(4)

- D. Litman and J. Allen. 1987. A plan recognition model for subdialogues in conversation. *Cognitive Science*, 11:163-200.
- Y. Liu and H. Levesque. 2005. Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions. *Proceedings of IJCAI-05*, 522–527.
- J. McCarthy and P. Hayes. 1969. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502.
- D. McDermott and the AIPS-98 Planning Competition Committee. 1998. PDDL – The Planning Domain Definition Language. Technical Report CVC TR-98-003 / DCS TR-1165, Yale Center for Computational Vision and Control.
- S. McIlraith and R. Fadel. 2002. Planning with complex actions. *Proceedings of NMR-02*, 356–364.
- J. Modayil and B. Kuipers. 2008. The initial development of object knowledge by a learning robot. *Robotics and Autonomous Systems*, 56(11):879–890.
- R. C. Moore. 1985. A formal theory of knowledge and action. *Formal Theories of the Commonsense World*, 319–358. Norwood, NJ: Ablex Publishing.
- K. Mourão, R. Petrick, and M. Steedman. 2009. Learning action effects in partially observable domains. *Proceedings of the ICAPS 2009 Workshop on Planning and Learning*, 15–22.
- H. Palacios and H. Geffner. 2009. Compiling Uncertainty Away in Conformant Planning Problems with Bounded Width, *Journal of Artificial Intelligence Research*, 35:623–675.
- H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling. 2007. Learning symbolic models of stochastic world domains. *Journal of Artificial Intelligence Research*, 29:309–352.
- C. R. Perrault and J. Allen. 1980. A plan-based analysis of indirect speech acts. *American Journal of Computational Linguistics*, 6(3-4):167–182.

References...(5)

- R. Petrick. 2009. P^2: A baseline approach to planning with control structures and programs. *Proc. of ICAPS 2009 Workshop on Generalized Planning: Macros, Loops, Domain Control*, 59–64.
- R. Petrick, N. Adermann, T. Asfour, and M. Steedman. 2010. Connecting Knowledge-Level Planning and Task Execution on a Humanoid Robot using Object-Action Complexes. Poster in the *Proceedings of CogSys 2010*.
- R. Petrick and F. Bacchus. 2002. A knowledge-based approach to planning with incomplete information and sensing. *Proceedings of AIPS-2002*, 212–221.
- R. Petrick and F. Bacchus. 2004. Extending the knowledge-based approach to planning with incomplete information and sensing. *Proceedings of ICAPS-2004*, 2–11.
- R. Petrick, C. Geib, and M. Steedman. 2009. Integrating Low-Level Robot/Vision with High-Level Planning and Sensing in PACO-PLUS. Technical Report, PACO-PLUS project Deliverable 4.3.5, available at `http://www.paco-plus.org/`.
- R. Petrick, D. Kraft, N. Krüger, and M. Steedman. 2009. Combining cognitive vision, knowledge-level planning with sensing, and execution monitoring for effective robot control. *Proceedings of the ICAPS 2009 Workshop on Planning and Plan Execution for Real-World Systems*, 58–65.
- R. Petrick and H. Levesque. 2002. Knowledge equivalence in combined action theories. *Proceedings of KR-2002*, 303–314.
- S. Richter and M. Westphal. 2008. The LAMA Planner: Using landmark counting in heuristic search. *6th International Planning Competition at ICAPS 2008*.
- E. Sandewall. 1994. *Features and Fluents*, volume 1. Oxford University Press.
- R. Scherl and H. Levesque. 1994. The frame problem and knowledge-producing actions. Technical report, University of Toronto.
- R. Scherl and H. Levesque. 2003. Knowledge, action, and the frame problem. *Artificial Intelligence*, 144(1-2):1–39.

References...(6)

- D. Smith and D. Weld. 1998. Conformant Graphplan. *Proceedings of AAAI-98*, 889–896.
- T. C. Son and C. Baral. 2001. Formalizing sensing actions - a transition function based approach. *Artificial Intelligence* 125(1–2):19–91.
- M. Soutchanski. 2001. A correspondence between two different solutions to the projection task with sensing. *Commonsense 2001*, `http://cs.nyu.edu/faculty/davise/commonsense01/`.
- M. Steedman. 1997. Temporality. *Handbook of Logic and Language*, 895–938. Amsterdam: North Holland/Elsevier.
- M. Steedman. 2002. Plans, affordances, and combinatory grammar. *Linguistics and Philosophy*, 25:723–753.
- M. Steedman and R. Petrick. 2007. Planning dialog actions. *Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue (SIGdial 2007)*, 265–272.
- M. Stone. 1998. Abductive planning with sensing. *Proceedings of AAAI-98*, 631–636.
- M. Stone. 2000. Towards a computational account of knowledge, action and inference in instructions. *Journal of Language and Computation*, 1:231–246.
- S. Vassos and H. Levesque. 2007. Progression of situation calculus action theories with incomplete information. *Proceedings of IJCAI-07*, 2029–2034.
- X. Wang. 1995. Learning by observation and practice: An incremental approach for planning operator acquisition. *Proceedings of ICML-95*, 549–557.
- D. Weld, C. Anderson, and D. Smith. 1998. Extending Graphplan to Handle Uncertainty & Sensing Actions. *Proceedings of AAAI-98*, 897–904.