



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

DR 9.2:

Title

P. Jensfelt, Nick Hawes & Michael Zillich

KTH

`<patric@kth.se>`

Due date of deliverable: July 31, 2009

Actual submission date: July 27, 2009

Lead partner: KTH

Revision: final

Dissemination level: PU

This document describes the CogX Summer School organised at KTH in Stockholm, Sweden March 1-7, 2009. This was the first out of three Summer Schools planned for. The main parts of the school were, invited talks to provide a glimpse of activities in related projects, tutorials covering the use of the common software and hardware in CogX and a project to be solved in groups to get hands-on experience and act as a team building activity.

1	Tasks, objectives, results	4
1.1	Preparations	4
1.1.1	Project Work	5
1.1.2	Lessons Learned	6
1.1.3	Relation to the state-of-the-art	6
2	Proceedings	7

Executive Summary

The first out of three Summer Schools planned for in the CogX project was organised by KTH March 1-7¹, 2009. 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].

There were two invited speakers. Giorgio Metta from the Cognitive Humanoids Laboratory at the Italian Institute of Technology presented the work on the iCub and the software framework YARP² (Yet Another. Robot Platform) which has gained a lot of interest in the robotics community. Marc Hanheide from the Applied Informatics group at the University of Bielefeld presented the group's work on cognitive systems such as the work on active memory architectures that is of particular interest as it is similar in spirit to the CogX approach but with interesting differences.

At the end of the week, each participant had helped create one of six integrated systems where components, both new and existing had been combined to allow the robot to move around autonomously and visually detect objects. 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.
- efficiently communicate knowledge to the researcher, both from external parties in the form of invited speaker and researchers within the consortium

¹Sunday to Saturday

²developed for the iCub but not limited in use to it

- increase impact of the dissemination by including external parties (invited speakers and participants) in the summer school who get a close 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

The first Summer School introduced the common hardware and software platforms. Because CogX builds on a software framework (CAST) developed in CoSy it was important that new researchers were familiarised with this integration tool. All integration in CogX is done using CAST. In addition CAST underwent a significant revision early in CogX, primarily to the underlying communication framework. This meant it was important that researchers familiar with CAST were able to understand the impact of these changes. Finally the consortium purchased six identical robot platforms to support integration as well as training. To enable early integration it was important that CAST and key components and sub- architectures (navigation, mapping, object recognition) were tested with the robots. In addition the task picked was the active visual search part of the Dora scenario. The spring school in Stockholm enabled us to fulfil all these requirements. It has therefore made a very important contribution to scenario based integration.

1 Tasks, objectives, results

1.1 Preparations

As already mentioned the summer school aimed at introducing the hardware and software that is going to be used in the project. To this end a large portion of the school was allocated to project work where each participant worked actively with both hardware (platform, sensors, computers) and software (architecture framework (CAST), components, algorithms, etc) in a group of 4-5 people.

The six identical robot platforms bought by the project for the integration effort and training were retrofitted after the summer school instead of as initially planned before. There were two important reasons for this. The design of the platform has an important impact on the rest of the project and it is therefore essential that it is well done. The summer school provided valuable input for this process. A simpler robot setup for the school with a single camera and a laser scanner provided a better balance in terms of hardware complexity as it was the first contact with robotics for some of the researchers. There was thus, no superstructure, no arm, no pan-tilt-unit, and only one instead of three cameras. In hindsight it is clear that this was the right decision. The challenge was just right with the intermediate setup and some of the experience from the summer school could go into the final design of the robot platform.

In terms of software, the common architectural framework CAST [1] was largely re-implemented before the summer school using a new underlying communication framework (ICE instead of CORBA). Each participant was asked to install CAST before arriving at the summer school and go through a tutorial illustrating CAST. This worked well and saved time at the summer school.

CAST divides the system into so called subarchitectures (SAs). Two such subarchitectures were prepared for the participants to use, the vision SA and the navigation SA. The vision SA provided means to acquire images (from file or camera) and perform object detection using the FERNS based planar object detector by Vincent Lepetit et al [2]. The navigation SA provided means to move the robot without running into obstacles and perform localisation and mapping [3]. An introduction with all participants of each SA was given in a lecture like fashion, followed by a hands-on tutorial that illustrated the finer details of the components in each SA.

Given these basic building blocks a task was designed for the project where each project group would need to add some functionality and equally important get valuable experience in integrating it all. The task was inspired by the active visual search task for the Dora integration scenario. It consisted of moving through the environment looking for objects and move these objects to a designated drop-off area.

In addition to the project work and introduction to software and hardware used in the project, two invited speakers were invited. Giorgio Metta from the Cognitive Humanoids Laboratory at the Italian Institute of Technology presented the work on the iCub and the software framework YARP. Marc Hanheide from the Applied Informatics group at the University of Bielefeld presented gave an overview of the group's work on cognitive systems and went in to detail about their work on active memory architectures which is of particular interest as it is similar in spirit to the CogX approach but with interesting differences.

In addition to the scientific program there was a program for social integration to even further make people interact and get to know each other. All but the last lunch was organised for all to have together and most dinners were also common. A tournament was organised in the indoor sport "floor ball" which is one of the most popular sports in Sweden but not played in many other countries.

The full program for the summer school, a more detailed description of the project task, the SA tutorials and the slides presented by the invited speakers can be found in the proceedings at the end of this document.

1.1.1 Project Work

All participants were divided into project groups. This included PIs and one of the invited speakers. The two main factors when forming the groups were i) to make people that would work together later start to know each other and ii) to distribute the knowledge and skills as evenly as possible among the groups. The tutorials introducing the two SAs provided the foundation for the task that had to be solved by each group. Wednesday afternoon to Saturday lunch was set aside for each group to work on the project without any tutorials or invited talks. All groups but one managed to build a system that could move the robot through the environment and search for objects. Most groups detected at least one object. One group successfully completed the task by transporting one object to the drop-off area.

Looking more closely at what the groups implemented, integrated and tested reveals that all groups were able to build maps and localise and move around autonomously in the environment. All groups could use the object recogniser and train it on novel objects and thereby expand the set of known objects. Most groups created a sensor model for the FERNS object detector to be able to incorporate the information provided it in a probabilistic way. Most groups had a strategy for exploring the environment and most groups even had a strategy for grabbing the object to be able to move it.

1.1.2 Lessons Learned

Given the aims of the summer school, the decision to only have two invited speakers was good. In fact, one might have been enough to even further emphasise the work on the project task which in the end probably would have helped the overall CogX integration project more. Ensuring a common view both in terms of software and hardware is very valuable. It creates a better understanding and makes people more open to accept that problems in other domains than their own can be equally challenging and effectively illustrates that integration is hard, very hard.

Each participant also learned that even with limited time, a small group of people working together can accomplish very much. The importance of communication in this process was also highlighted. This experience will in the integration effort in the project.

1.1.3 Relation to the state-of-the-art

During the first year the main emphasis was on getting all members in the project comfortable to work with the software framework used in the project and get some hands-on experience using the common robot platform. As such, the work was not aiming at state-of-the-art in the work but to build the group and set the stage for moving beyond state-of-the-art in the future.

References

- [1] N. Hawes, A. Sloman, J. Wyatt, M. Zillich, H. Jacobsson, G. J. Kruijff, M. Brenner, G. Berginc, and D. Skočaj, “Towards an integrated robot with multiple cognitive functions,” in *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI 2008)*, R. C. Holte and A. Howe, Eds. Vancouver, Canada: AAAI Press, July 2007, pp. 1548 – 1553.
- [2] M. Ozuysal, P. Fua, and V. Lepetit, “Fast Keypoint Recognition in Ten Lines of Code,” in *CVPR’07*, 2007.
- [3] J. Folkesson, P. Jensfelt, and H. Christensen, “The m-space feature representation for slam,” *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 1024–1035, Oct. 2007.

2 Proceedings

The proceedings is a modified version of the proceedings handed out to the participants of the CogX 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.



Proceedings for
CogX “Spring” School
Stockholm, March 2009

P. Jensfelt, N. Hawes & M. Zillich

Table of contents

1. Program
2. Task overview
3. nav.sa tutorial
4. vision.sa tutorial
5. Invited talk Tuesday 3/3: Giorgio Metta
6. Invited talk Wednesday 4/3: Marc Hanheide

Program

Sunday 1/3

Time	Activity
day	Arrival
19:30-	Dinner

Monday 2/3

Time	Activity
09:00-09:15	Introduction (Jeremy)
09:15-09:40	Local arrangements (Patric + Jeanna)
09:40-10:30	Practical content overview (Nick)
10:40-11:00	Coffee, CAS floor 5
11:00-12:30	Installation and setup (Patric, Nick, Michael)
12:30-14:00	Lunch Syster & Bror
14:00-14:30	nav.sa (Patric)
14:30-15:40	nav.sa tutorial
15:40-16:00	Coffee
16:00-	nav.sa tutorial
19:00-	Dinner

Tuesday 3/3

09:00-10:40	Invited talk Giorgio Metta
10:40-11:00	Coffee
09:40-10:30	Invited talk Giorgio Metta, cont'd
12:30-14:00	Lunch Syster & Bror
14:00-14:30	vision.sa (Michael Z.)
14:30-15:40	vision.sa tutorial
15:40-16:00	Coffee
16:00-	visions.sa tutorial
19:00-	Dinner

Wednesday 4/3

09:00-10:40	Invited talk Marc Hanheide
10:40-11:00	Coffee
09:40-10:30	Invited talk Marc Hanheide, cont'd
12:30-14:00	Lunch Syster & Bror
14:00-15:40	Project work
15:40-16:00	Coffee
-	Optional visit at Nordiska Museet or Vasa museum
18:00-	Dinner

Thursday 5/3

09:00-09:30 Inspection of the test area
09:30-10:40 Project work
10:40-11:00 Coffee
09:40-10:30 Project work
12:30-14:00 Lunch Syster & Bror
14:00-15:40 Project work
15:40-16:00 Coffee
16:00- Project work

Friday 6/3

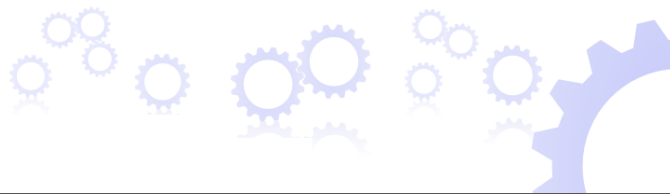
09:00-10:30 Inspection of the test area
10:30-11:00 Travel to “Frescatihallen”
11:00-13:00 Innebandy tournament
13:00-14:00 Change and lunch
14:00-15:40 Project work
15:40-16:00 Coffee
16:00- Project work

Saturday 7/3

09:00-14:00 Final tuning and lunch on your own
14:00-17:00 Project presentations
17:00-18:00 Closing the spring school (Jeremy)

Task Overview

CogX “Spring” School **Task Overview**



Carrie, The Robot Gopher

- The robot starts from a fixed point in a known environment.



Carrie, The Robot Gopher

- It must find 3 target objects amidst distractors....



Carrie, The Robot Gopher

- ... and return them to a fixed collection point.
 - How they're moved is up to you!



Objects and Environment

- The task area will have multiple rooms linked by a corridor.
- Each object will:
 - be planar, with at least 2 distinct sides,
 - be at least 35cm tall,
 - have a (fairly) distinct set of colours,
 - be placed on the floor in a room.
- The environment will also contain distractor objects.
- Start and end points will be clearly, visually, marked during the preparation period.

Carrie-on Items

- You will be provided with two CAST subarchitectures (navigation and vision) to get you started, plus a training environment and objects.
- You can modify **anything** you like about the software and hardware.
- However, all sensing and actuation must be done on the mobile platform.
- And the objects must not be modified.

Schedule

- **Monday & Tuesday:** instruction on the robot and the navigation and vision subarchitectures.
- **Wednesday & Thursday:** inspection of the test area, development.
- **Friday:** development and Innebandy for seeding.
- **Saturday am:** test area reconfigured for mapping, and test objects available.
- **Saturday pm:** show time!

Groups

Red Team

Ivana
Marko
Charles
Alper
Moritz

Yellow Team

Hendrik
Vero
Patrick
Alen
Marc

Green Team

Kristoffer
Luka
Kai
Richard
Thomas K

Blue Team

Rustam
Yasemin
Thomas M
Pierre

Purple Team

Adrian
Manuel
Michael
Gj

Black Team

Andrzej
Andreas
Sergio
Danijel

Human In-The-Loop

- Map the area by hand (i.e. pen and pencil)
- One person drives the robot using the joystick, but is only allowed to look at the map.
- The remaining team members act as sensors:
 - **rangers**: can only shout distances.
 - **object detectors**: can only shout class labels, distance and relative orientation.
 - **bumpers**: can only shout "BUMP!"
- Try to navigate through a cluttered environment.

CogX Spring School '09: Getting Started Guide

After coffee find your group and retire to your team HQ. These are all on CAS floor 5 and are as follows (including which robot you will be bonding with):

1. Red Team: Room 521, BHAM robot
2. Yellow Team: Room 518, ALU robot
3. Green Team: Room 525, KTH robot
4. Blue Team: Room 513, TUW robot
5. Purple Team: Room 519, DFKI robot
6. Black Team: Room 520, UOL robot

The two remaining rooms open to us are 514 and 523. These are room in which you can test your robots (lookout though, 514 may also contain some humans!). Please do not use any other rooms as these are working offices for our hosts and another department.

Once you're in your room one of the organising team will come along and give you a quick overview of the robot and answer any burning questions you may already have. In the meantime, we suggest you go through the following steps.

1. Get yourself on the wireless network. There are two SSIDs for the school: cogx24 (2.4 GHz band) and cogx5 (5 GHz band). They both have the same WPA password: cogxxgoc.
2. Check out a copy of your group's subversion repository. Each group has a repository containing all the supplied code which only they can read from and write to. To access it do the following (where ? should be replaced by the list number which mentions your group above):

```
svn co https://codex.cs.bham.ac.uk/svn/nah/cogx/css09/group?/trunk
```

3. Using this newly checked out code, compile and install CAST.
4. you have not completed the CAST tutorial yet, shame on you! Do so now.
5. If you get this far you can try installing the prerequisites for the navigation subarchitecture.
6. Once over that hurdle, do the same for the vision subarchitecture.
7. Is that the time? Go for lunch!

Project: Competition Scores

Score Sheet

- Each object found: +1 point.
- Each object returned: +4 points.
- Object touched by human, or robot assisted in obtaining or releasing object: -3 points (-2 in rooms and corridors, -1 in drop-off area)
 - Robot and human must be stationary.
 - Human can only touch an object previously touched by the robot.
 - Human cannot move feet when holding object.
 - Human cannot move feet when holding object.
- Marks for overall style: +5 points.
 - System crashes or restarts –
 - Human steering or assistance –
 - Robot in control of a human ++
- In the event of a tie on points, the quicker team will win. The clock will start
- In the event of a tie with no separating time, a final task will be used to separate the teams.

System, Objects and Environment

- Each object will:
 - be planar, with at least 2 distinct sides,
 - be at least 35cm tall,
 - have a (fairly) distinct set of colours,
 - be placed on the floor in a room.
- The environment will also contain distractor objects.
- Start and end points will be clearly, visually, marked during the preparation period.
- The task area will have multiple rooms linked by a corridor.
- You can modify anything you like about the software and hardware.
- However, all sensing and actuation must be done on the mobile platform.
- And the objects must not be modified.

The Competition Day Itself

In the morning each team will be given at least 30 minutes in the test environment and at least 30 minutes with the test objects. The environment will be as it will be for the test, minus the target and distractor objects (which will be repositioned for each team).

In the afternoon you will be given a strict 30 minutes to perform the task. The clock will start when the previous team has vacated the test area.

The Navigation Subarchitecture - nav.sa

The navigation subarchitecture or nav.sa for short is responsible for maintaining the lower levels of the spatial representation, keeping the robot localized and for moving it around in the environment. The lower levels of the spatial representation refers to a metric, line based map and a so called navigation graph that encodes free space and its connectivity.

Installation

CURE To build and run the nav.sa you need to have CURE installed. CURE is a software package developed at KTH aiming at providing a "turn key solution" for navigation, localization and mapping. Most of the components in the version of nav.sa that you will be working with are in fact wrappers around components in CURE. CURE also contains some nifty little helper classes for performing transformations between coordinate frames and working with matrices. To install CURE you need to do the following. Below, \$COGX_ROOT is the place where you got the code for the summer school, that is the directory where the css folder resides.

```
sudo apt-get install automake autoconf libtool
cd $COGX_ROOT
svn co https://codex.cs.bham.ac.uk/svn/nah/cosy/development/kth/cosycure
cd cosycure
./do-it-all.sh
```

This will install cosycure in the directory cosycure itself. When you build the nav.sa you need to specify where CURE is installed and in the example above that would be in \$COGX_ROOT/cosycure.

Peekabot Peekabot is the visualization tool used in nav.sa. It is not strictly necessary to have peekabot installed but working with nav.sa is much easier when you see what is going on "inside" it. The homepage of peekabot can be found at <http://www.peekabot.org>. You need boost 1.35 to compile peekabot plus some additional libraries. Under ubuntu 8.10 you would need to do

```
sudo apt-get install libboost1.35-dev libfltk1.1-dev libpng12-dev
libxerces-c2-dev bzip2 libfreetype6-dev libglut3-dev libtool
automake autoconf
cd <SOMEWHERE NICE>
bzip2 branch lp:peekabot
cd peekabot
autoreconf -i
./configure
```

```
make
sudo make install
```

If you're on an Intel Mac you can install the binary version of Peekabot from sourceforge. This is not the latest and greatest version, but it should be enough to get you started. Peekabot allows you to load model from file. These models could for example be of robot base, arms and objects that the robot can recognize. Some of these models have been made specifically for CoSy/CogX. The models are found in subarchitectures/nav.sa/config/peekabot-models. You need help peekabot to find these models. There are two ways to accomplish that. One is to copy the files from the subdirectories models and scenes into the corresponding directories under `/.peekabot`. The other way to do it is to configure peekabot to look in for model and scene files directly in the nav.sa subdirectory. You do this by editing the file `/.peekabot/config.xml`. You need to add paths to the scenes and models directories under the "model_dir" section and a path to the scenes directory under the "scene_dirs" section. When you have compiled and installed peekabot and made the CogX model files available you should be able to test the installation by starting peekabot with "peekabot&" and selecting "Load Scene" under the File menu. If you enter "CogXp3.xml" you should see a model of the CogX demo platform. If you get a black screen instead of a 3D scene, you may have to deactivate the 3D visual effects of your desktop (on Ubuntu: System -> Preferences -> Appearance -> Visual effects). If you want to update peekabot from the repository you do

```
cd peekabot
bzip pull
make
sudo make install
```

Player To interface all(?) the hardware except cameras (so robots, lasers, etc) we use player at the bottom. The main advantage of this is that it allows us to run the system in simulation without any changes to any of the code in the system. You should use player version 2.1.1. If you get player 2.1.1 from sourceforge you will need to edit some files to get it to run. Therefore we recommend that you download the version linked to below instead We use player 2.1.1 since the 2.1.2 has a broken driver for the P3DX robots it seems. Before compiling player please install the ltdl development files (stage needs player to be compiled with it)

```
sudo apt-get install libltdl7-dev libgtk2.0-dev
```

Also make sure that you have the file `/etc/X11/rgb.txt`. Get a copy of it under subarchitectures/nav.sa/config/rgb.txt and copy it into `/etc/X11` Building and installing player follows the standard automake routine

```
./configure
make
sudo make install
```

Stage Download stage 2.1.1. Building and installing stage follows the standard automake routine

```
./configure
make
sudo make install
sudo ldconfig
```

When you run configure make sure that stage finds your installation of player. Otherwise it will not build the stageplugin for player. To test that the installation worked do

```
cd stage-2.1.1/worlds
player simple.cfg
```

this should bring up a window with a simulated world. If not follow the error messages (did you install the rgb.txt as instructed when building player? Did you install libltdl-dev before building player? If not fix and re-configure and recompile player and then stage) subarchitectures/nav.sa + tools Besides CAST itself the code that is relevant when using the nav.sa is

```
subarchitectures/nav.sa/
tools/hardware/robotbase
tools/hardware/laser
tools/hardware/utils
```

Build it in the standard cmake style, i.e. go into the BUILD directory and make sure that these are set to compile

Components

The following section will describe the components inside the nav.sa. The most important files in nav.sa are found in {{{subarchitectures/nav.sa/src/slice/subarchitectures/nav.sa/src/c++/components/ }}} The first is where you find the NavData.ice file that defines the structures used inside the SA. The latter contains the components that do processing.

SlamProcess This component has as its main task to keep track of the position of the robot. It does so either in SLAM mode in which case it estimates the map as well as the pose of the robot, or in localization mode where a map is given and only the robot pose is estimated. This component will write/overwrite a NavData::RobotPose object with information about

the current robot pose to working memory. It will also write the current NavData::LineMap to working memory.

Needs: LaserServer, RobotbaseServer running

OdomLocalizer This is a dummy component that translates Robotbase::Odometry from the RobotbaseServer into NavData::RobotPose and writes this to working memory. The idea is that you can use this in cases where you for one reason or another do not want to run the SlamProcess to do real localization/SLAM but still need a NavData::RobotPose updated when the robot is moving.

Needs: RobotbaseServer running

NavGraphProcess This component builds a navigation graph consisting of nodes (NavData::FNode) and edges (NavData::AEdge) connecting these. The nodes represent free space where the robot has been and the edges connectivity between them. Together the nodes and the edges model the free space and how to move in it. The nodes can be of two types, normal nodes and gateway nodes. The latter are doors and these divide the graph into topological units (ideally roughly corresponding to rooms). It detects the doors using the laser scanner. It looks for openings of a certain width that the robot is in, i.e. it can only detect a door if it is in it.

Needs: LaserServer, RobotbaseServer running and NavData::RobotPose (from SlamProcess or OdomLocalizer)

NavControl This component handles the motion of the platform. If you want to move the robot you should write a struct NavData::NavCommand to the nav.sa WM. The enum NavData::CommandType tells what command types that are available. It uses the navigation graph produced by the NavGraphProcess to know how to move from A to B.

Needs: LaserServer, RobotbaseServer running, NavData::RobotPose (from SlamProcess or OdomLocalizer) and NavData::NavGraph (from NavGraphProcess)

TranslationProcess The task of this component is to listen for messages/events of type NavData::NavCommand and translate these into structures of type NavData::InternalNavCommand which is what the NavControl component is controlled by.

Needs: nothing

DisplayNavInPB This component display the robot when it is moving around, the line map, the navigation graph with object, etc

Needs: peekabot

CureDataLogger This component can log scan, odometry and world position to file

Needs: LaserServer, RobotbaseServer running and optional NavData::RobotPose (from SlamProcess if you want the world position)

Tutorial: nav.sa**Part I : Getting to know what is there**

Hooking up to the robot with Player/stage

Stage To start things off fire up stage by opening a new terminal window and

```
cd subarchitectures/nav.sa/config/stage_models/cas_floor7
player cogxp3.cfg
```

This should open up a window with a model of the 7th floor of CAS. It looks much like the 5th floor where you are probably now.

Attach a joysick to the computer (which cannot be a Mac at this point, sorry GJ!). Run the cast-file

```
subarchitectures/nav.sa/config/tutorial/first-glance-1.cast
```

If you look in the system log (with "dmesg" or "tail /var/log/messages") you should see something similar to this

```
[14700.596136] usb 5-1: new low speed USB device using uhci_hcd and address 8
[14700.806095] usb 5-1: configuration #1 chosen from 1 choice
[14700.869442] input: Logitech Logitech Cordless RumblePad 2 as /devices/pci0000:00,
[14700.892134] input,hidraw0: USB HID v1.10 Gamepad [Logitech Logitech Cordless Rumb
```

You should now be able to drive the robot around inside stage using the gamepad. Things should run just fine without a gamepad but you cannot drive the robot around.

Using the gamepad To move the robot using the gamepad you always need to keep key "5" pressed. You move the robot with the left joystick. If you simultaneously press key "6" you will force the translation speed to be 0 (ie will not listen to the forward/backward motion of the joystick). Key "8" does the same for rotation. These last two buttons are handy when you want to drive straight or turn on the spot.

5 dead switch (all control requires this to be pressed)

6 set translation speed to 0

8 set rotation speed to 0

left joystick : move around

Player with the real robot Now to do the same as above but with the real robot all you need to do is to change the configuration file for player. You are provided with a config file that should work

```
subarchitectures/nav.sa/config/player_cfg/botx.cfg
```

This file looks like

```
driver
(
  name "p2os"
  provides ["odometry::position2d:0"
            "sonar:0"
            "power:0"
            "bumper:0"
           ]
  port "/dev/ttyUSB11"
  pulse "2"
)

driver
(
  name "urglaser"
  provides ["laser:0"]
  port "/dev/ttyACM0"
)
```

Before you firing up player you need to connect the hardware to the computer that will run player. When you connect the usb-serial adapter connecting the robot to the computer you should see something like this in the system log (with "dmesg" or "tail /var/log/messages")

```
[14653.876305] usb 5-1: new full speed USB device using uhci_hcd and address 5
[14654.033388] usb 5-1: configuration #1 chosen from 1 choice
[14654.037772] p12303 5-1:1.0: p12303 converter detected
[14654.054593] usb 5-1: p12303 converter now attached to ttyUSB0
```

The key here is the device name, in this case ttyUSB0. It should match the one in the configuration file.

When you connect the laser scanner to the computer you should see something like

```
[14667.556288] usb 5-1: new full speed USB device using uhci_hcd and address 6
[14667.720818] usb 5-1: configuration #1 chosen from 1 choice
[14667.724529] cdc_acm 5-1:1.0: ttyACM0: USB ACM device
```

again it is the device name (ttyACM0) that is important and should match the config file for player.

Now we are ready to start. Close player if it is running and then do

```
player subarchitectures/nav.sa/config/player_cfg/botx.cfg
```

This should result in a printout like this

```
Registering driver
Player v.2.1.1
```

```
* Part of the Player/Stage/Gazebo Project [http://playerstage.sourceforge.net].
* Copyright (C) 2000 - 2006 Brian Gerkey, Richard Vaughan, Andrew Howard,
* Nate Koenig, and contributors. Released under the GNU General Public License.
* Player comes with ABSOLUTELY NO WARRANTY. This is free software, and you
* are welcome to redistribute it under certain conditions; see COPYING
* for details.
```

```
listening on 6665
Added file watch 4
Added file watch 5
Listening on ports: 6665
```

If you hook up a gamepad and run the same cast file as above, i.e. subarchitectures/nav.sa/config/first-glance-1.cast, you should be able to move the real robot around.

Displaying stuff in peekabot Being able to see what is going on inside the nav.sa is quite helpful, for this purpose we use peekabot (<http://www.peekabot.org>). Repeat the same exercise as above but now with the following cast file

```
subarchitectures/nav.sa/config/tutorial/first-glance-2.cast
```

and also with peekabot running. You should see the robot on the screen, the laser scan should be displayed and the robot should move inside peekabot as you drive.

Building a map Now let us build a map!

We will build a line based map and the navigation graph. You will need peekabot started and you can chose between using the real robot or stage like you did before.

Run the system with the cast file from

```
subarchitectures/nav.sa/config/tutorial/build-map.cast
```

As you drive the robot around you should see nodes and edges forming a graph and you should see wall popping up. Each node is marked with a circle. The nodes are space 1m apart approximately. The color of the nodes changes with the id of the area. Try moving through a door to see that the robot detects it and displays it on the screen. Notice that the door detector only detects doors that the robot passes through and that it has no way of knowing that it really is a door or not. This means that if obstacles/people happen to form an opening that has the right width and the robot drives there it will think that it found a door. You can set the min and max distances for a door with the options `-min-door-width` and `-max-door-width` for `NavGraphProcess`.

As the system is running you should see three files being updated. A file called `tmpmap.graph` contains the data in the graph. The file `tmpmap.metric` contains the line map in a hard to parse format and `robotpose.ccf` contains the current pose of the robot.

If you want to keep the map you made you can copy the files `graph/metric` to files with other names.

Notice how changing the values inside the `robotpose.ccf` file will affect the map building. This file tells the system where the robot starts. If you delete the file it will be interpreted as the start pose is 0,0,0. (The file format for `robotpose.ccf` is for a 6D representation so x,y,z and three angles.) A good rule is to make sure that the `robotpose.ccf` file is removed before you start building a map, this way you will know that the position from where you started is 0,0,0 with the x-axis being defined by the direction the robot was facing.

Localizaiton In localization mode the robot will use an existing map to stay localized. For this to work you obviously need a map first. To test this build map like above. Make sure that the robot is not moved after the system is closed and do not touch the `robotpose.ccf` file. I should now contain the pose of the robot in the map. Edit the cast file so that you give the `SlamProcess?` an additional argument `"-m <mapfile>"`, if you have not changed the name of the map file you should thus be adding `"-m tmpmap.metric"`. You can do the same for the `NavGraphProcess?` to make it read the graph file, e.g. `"-m tmpmap.graph"`. The line map will not be extended in localization mode but the graph can be extended. It will simply add to it.

As long as you do not move the robot after stopping the system the `robotpose.ccf` file should do the trick and you should be able to continue as you were. However, if you want to use an old map, move the robot when the system is not running or otherwise make the `robotpose.ccf` file not contain the right information you need to supply the system with the information about the starting position of the robot.

Task: Build a map. Shut down the system, move the robot and make sure that you know how to initialize it again in the map.

Moving about autonomously The motion of the robot is taken care of by the component NavControl. It takes commands of type NavData::InternalNavCommand. You should NOT write these commands directly to the working memory. Instead write commands of type NavData::NavCommand and let the Translation process create the internal commands and manage the task queue, the priorities etc

Makes sure you have player or stage running as well as peekabot. Run the cast file

```
subarchitectures/nav.sa/config/tutorial/test-navctrl.cast
```

You should see three more object popping up in peekabot, one big green circle and two markers, one grey and one green. Drag the grey marker to a desired location of the robot. Have look here if you do not remember how to do this. Then drag the green marker inside the big green circle. The robot should start moving toward the grey marker.

Take a look inside the PeekabotControl component to see how it was done.

Part II : Change events

Task 1: Write a component that listens for changes to the following object types in working memory, NavData::RobotPose, NavData::TopologicalRobotPose, NavData::FNode, NavData::AEdge, NavData::NavGraph, NavData::NavCommand and see how these events come when you move the robot around.

Task 2: Modify the !DisplayNavInPB component so that it displays the status of the current NavCommand in peekabot as a text string on the ground as it is executed. It should update the string when the status of the command changes.

Part III : Move autonomously

Task: Write a component that makes the robot drive autonomously between the nodes in a graph. It should stop at a node and for nodes with odd id turn to 0 degs in the world frame, and for nodes with even id to turn 45 degrees relative to the position it arrived at the node with.

When you reach a node you should also write an NavData::ObjObs to the working memory pretending that you made an object observation. Notice what happens in peekabot and experiment with the optional flags in struct. Notice that you need to set the time. Use the getCASTTime() function for this

Part IV : Person following

Task: Write a component that hooks up to the LaserServer and gets the RobotPose from working memory. Find the closest point around the robot (watch for outliers) and assume that this is a person. Hook up to the RobotbaseServer (see NavControl for an example) and send motion commands directly to it. The commands should be designed so that the robot maintains a distance of 1m from the person (or whatever was closest) and looks in the direction of him/her.

To avoid getting conflicting commands from NavControl you either have to make sure that it is not running or write an object of type !NavData::InhibitNavControl to the working memory. As long as such an object is in the working memory NavControl will not send any commands.

Now do the same using the NavData::NavCommands, i.e. do not talk directly to the RobotbaseServer. Notice that the NavControl component is not particularly fond of going to places which are already occupied by something.

The Vision Subarchitecture

Vision at this point only provides object detection and tracking.

Prerequisites

You will need to have OpenCV installed. It is used for image capture and for various lower level image processing.

First make sure You uninstall any previous OpenCV version (e.g. via ubuntu package manager or in case You installed from source by doing make uninstall in the installation directory).

Then get the svn sources with

```
svn co https://opencvlibrary.svn.sourceforge.net/svnroot/opencvlibrary/trunk/opencv
```

The newest version is needed because only that supports the latest libdc1394 version 2.x for image capture. Also install the ubuntu package libdc1394-22-dev. And also libdc1394-utils is highly recommended.

IMPORTANT: The configure script is outdated! Use the cmake file instead.

```
mkdir BUILD
cd BUILD
ccmake ..
  type 'c'
  set OPENCV_CONFIG_FILE_INCLUDE_DIR /usr/local/include/opencv
  type 'c' 'g'
make install
```

And do not use

```
./configure
make install
```

cmake will output something like

```
...
-- Video I/O:
--   DC1394 1.x:
--   DC1394 2.x:          1   <===== This is what we want so see!
--   FFmpeg:
--   GStreamer:          0
--   UniCap:
--   V4L/V4L2:           1/1
--   Xine:                0
...
```

Caveats

- Never have two components on one machine doing OpenCV-based visualisation (i.e. calling `cvNamedWindow()`, `cvShowImage()`). CAST components are threads, i.e. share memory space. Having two OpenCV event loops screws things up and crashes the system.
- In case the kernel module for the firewire Express Card is not loaded automatically, do

```
sudo modprobe pciehp
```

- In order to use firewire cameras as non-root, do

```
sudo chmod a+rw /dev/raw1394 /dev/video1394/*
```

- if you get an error

```
libdc1394 error: Error: Failed to allocate iso bandwidth
libdc1394 error: Error: Failed to setup DMA capture
```

call `dc1394_reset_bus` from the command line. This is contained in the package `libdc1394-utils`.

Video Servers Video servers are the sources of images in the CogX system. They also provide camera parameters for each image. There can be any number of them and each one can handle any number of video inputs. Currently there are two implementations: `OpenCVImgSeqServer` and `OpenCVLiveServer`.

VideoServer base class

The base class for all video servers. Note that You will never instantiate one of those, but there are some CAST command line parameters that are common for all derived video servers:

- `-videoname <string> ..` a string ID specifying this servers name, e.g. `video1`. Video names must be unique in the system.
- `-camids <int list> ..` a list of integer IDs that specifies which camera this video server handles, e.g. `0` or `"1 2 3"`. Camera IDs must be unique in the system. If more than one ID, enclosing quotes are needed.
- `-camconfigs <string list> ..` a list of config files for the cameras, e.g. `config/logitech.cal` or `"config/logitech1.cal config/logitech2.cal"`. These calibration files follow the `.ini` file format and contain internal and external camera and distortion parameters. You can create calibration files with e.g. `camcalb`.

OpenCvImgSeqServer

Reads stored image sequences, where images are stored as individual files (e.g. img000.jpg, img001.jpg, ... img999.jpg). Supports all common file formats.

CAST command line parameters are

- `-files <string list> ..` a list of filename template strings, e.g. `subarchitectures/vision.sa/config/te ferns/tea%03d.jpg` or `"test/img%03d-left.jpg test/img%03d-right.jpg"`. If more than one string, enclosing quotes are needed.
- `-start <int> ..` number of the first frame.
- `-end <int> ..` number of the last frame, must be `>= start`.
- `-framerate_ms <int> ..` frame rate in milliseconds

OpenCvLiveServer

Captures live images from USB or Firewire cameras. Currently image size is fixed to 640x480.

CAST command line parameters are

- `-devnums <int list> ..` a list of integer device numbers, e.g. `0` or `"0 3"`. Specifies which of the attached video cameras to use. E.g. device `0` corresponds to `/dev/video0`.

Video Clients To get data from a video server you need to implement a client.

VideoClient base class

The base class for all video clients. Note that You will never instantiate one of those, but there are some CAST command line parameters that are common for all derived video clients:

- `-videohost <string> ..` hostname (or IP address) of the machine the server is running on, e.g. `localhost`, `192.168.1.4`. Defaults to `localhost`.
- `-videoname <string> ..` name of the video server to connect to

VideoViewer

Just a simple component to display current camera images. Also good as a template for components doing image processing.

The single CAST command line parameters is

- `-camid int ..` ID of the camera to use

ObjectDetectorFERNS The FERNS based planar object detector by Vincent Lepetit et al. <http://cvlab.epfl.ch/software/ferns/index.php>, which estimates a homography for a planar surface from detected keypoints on the surface texture. The component takes DetectionCommands and returns VisualObjects. These currently contain a label, the 2D contour of the detected planar surface as well as a 2D bounding box. Note that the homography can be used to calculate a proper 3D pose, which is however not implemented at this time.

The detector expects a file foo.jpg.roi containing four corner points file for each foo.jpg model file. Use e.g. tools/genroi/genroi for generating the .roi file. Points should be added in clockwise manner or the learning process will fail.

For the first run the detector will train its models using the image and ROI and save foo.jpg.detector_data and foo.jpg.tracker_data files. Training will take several minutes per model. For subsequent runs the detector will read (and decompress) the generated data files, which takes approx. 10 seconds per model.

The models for detection can get quite large, e.g. 20 MB for the tea.jpg.detector_data (tracker_data always is small). You might want to play with the parameters of the detector, in order to tune model size vs. detection rate. Parameters for the detector are passed via its constructor method. In method ObjectDetectorFERNS::setupFERNS()

```
detectors[i] = planar_pattern_detector_builder::build_with_cache(
    model_images[i].c_str(),
    &range,
    400,
    5000,
    0.0,
    32, 7, 4,
    30, 12,
    10000, 200);
```

These are the default values chosen by the original authors of the detector. The parameters of the constructor can be seen in ObjectDetectorFERNS/ferns_demo-1.0/planar_pattern_detector_builder.h:

```
static planar_pattern_detector * build_with_cache(const char * image_name,
    affine_transformation_range * range,
    int maximum_number_of_points_on_model,
    int number_of_generated_images_to_find_stable_points,
    double minimum_number_of_views_rate,
    int patch_size, int yape_radius, int number_of_octaves,
    int number_of_ferns, int number_of_tests_per_fern,
    int number_of_samples_for_refinement,
```

```
int number_of_samples_for_test,
    char * given_detector_data_filename = 0,
    int roi_up_left_u = -1, int roi_up_left_v = -1,
    int roi_bottom_right_u = -1,
int roi_bottom_right_v = -1);
```

Setting e.g. `maximum_number_of_points_on_model` to 40 instead of 400 and `number_of_generated_images_to_find_stable_points` to 500 instead of 5000 reduced the model size for `tea.jpg.detector_data` from 20 MB to 2 MB and model loading time from 10 to 2 seconds, without noticeable decrease of detection performance (on the stored tea image sequence)

Note that in order to have one display where everything in the system is visualised, `ObjectDetectorFERNS` can output its detected objects also to `nav.sa WM` where `peekabot` will pick it up and add to its display. In order to do so, You have to set option `FERNS_OUTPUT_TO_NAV` to "on" in `ccmake`. Of course You then also need to have option `BUILD_SA_NAV` turned on (and the other various `nav.sa` bits).

CAST command line parameters are

- `-videohost <string> ..` hostname (or IP address) of the machine the server is running on, e.g. `localhost`, `192.168.1.4`. Defaults to `localhost`.
- `-videoname <string> ..` name of the video server to connect to
- `-camid int ..` ID of the camera to use
- `-models <string list> ..` list of model image filenames, e.g. `"config/experiment1/tea.jpg config/experiment1/poster.jpg"`. The base of the model image filename become the labels of the detected objects, e.g. `tea` and `poster` in this case.
- `-mode <string> ..` specifies one of two modes the component can run in: `DETECT_ONLY` only does keypoint-based detection. `DETECT_AND_TRACK` does detection first and then tries to track using a template based tracker. If the tracker fails (e.g. too much motion), it falls back to detection.
- `-displaylevel <int> ..` specifies how much should be displayed: 0 - no display window, 1 - display bounding box, 2 - display actual surface boundary, 3 - display all the keypoints

Stereo

Stereo Server Provides 3D point clouds, calculated from two video server images. The stereo server thus is a video client.

Prerequisites

The stereo server uses the GPU, so You need to have nvidia CUDA installed (and needless to say an nvidia graphics card). Get CUDA from

http://www.nvidia.com/object/cuda_get.html

Choose Your OS, 32/64 Bit, and CUDA version. Note that You need the correct version of CUDA that matches Your graphics card driver. I am using ubuntu 8.10 and use the nvidia driver supplied by "Hardware Drivers", which is "NVIDIA accelerated graphics driver (version 180) [Recommended]". This matches CUDA version 2.1 (using CUDA version 2.2 leads to errors of some things being "not yet implemented" that are required by the GPU stereo code). NOTE: You will have to choose "Linux Version: ubuntu 8.04" to be offered CUDA 2.1. If You choose "Linux Version: ubuntu 8.10" You will only be offered the latest CUDA 2.2.

So You would e.g. get the files `cuda-toolkit_2.1_linux64_ubuntu8.04.run` `cuda-sdk-linux-2.10.1215.2015-3233425.run`. If You have a newer nvidia driver installed (e.g. 185.something), You might be able to use CUDA 2.2. But I did not test that, as I assume the standard Cogx ubuntu 8.10 32 bit laptop).

Installation of CUDA toolkit and sdk are simple. Just `sudo sh thefile.run`. The toolkit is installed into `/usr/local/cuda`. After that make a directory `/usr/local/cuda/sdk` and install the SDK there. Do a `sudo make there`. NOTE: You can ignore the errors

```
make[1]: Entering directory
`/usr/local/cuda/sdk/projects/convolutionSeparable'
/usr/include/bits/stdio2.h(35):
error: identifier "__builtin_va_arg_pack" is undefined
```

These errors only affect some of the example projects (and could be resolved by downgrading gcc to 4.2 which You don't really want to do). But the libraries we are actually interested in are already built ok before the example projects.

You then need `/usr/local/cuda/bin/` in Your `$PATH`, so the nvidia `nvcc` can be found.

Building

At the moment the CUDA things can not be built from within a cmake file, so You have to build externally. Go to `tools/hardware/stereo/src/c++/components/gpustereo` and do a `make` there. I hope to resolve this issue soon.

CAST command line parameters are (from `subarchitectures/vision.sa/config/stereoviewer.cast`)

```
...
COMPONENT CPP VideoServer OpenCvImgSeqServer --videoname video \
--camids "0 1" \
--files "subarchitectures/vision.sa/config/test-stereo/img%03d-L.jpg \
subarchitectures/vision.sa/config/test-stereo/img%03d-R.jpg" \
--start 0 --end 1 --framerate_ms 30
```

```

...
COMPONENT CPP stereoserver StereoServer --stereoname stereo \
--videoname video --camids "0 1" \
--stereoconfig subarchitectures/vision.sa/config/test-stereo/stereo.ini \
--display
...

```

- `-videohost <string>` .. hostname (or IP address) of the machine the server is running on, e.g. localhost, 192.168.1.4. Defaults to localhost.
- `-videoname <string>` .. name of the video server to connect to
- `-camids <"int int">` .. IDs of the left and right camera
- `-stereoconfig <string>` .. stereo calibration file, in the format generated by SVS (Small Vision System) IMPORTANT: SVS outputs an .ini file, where the '=' are missing. Proper .ini format is with '='. The stereo server requires such a proper .ini file with '=', so You have to add the '=' by hand after running the SVS calibration program.
- `-display` .. turn on display of left, right and disparity image

Other Stuff

Dummy Driver The DummyDriver component is just a simple component continuously sending detection commands to the visual WM, where they are picked up by ObjectDetectorFERNS. In order not to flood the WM, DummyDriver only asks for new detections, once it received a detection result (where the confidence level of the detected object can actually be 0, indicating that the object was in fact not detected).

CAST command line parameters are


- `-labels <string list>` .. list of objects to look for, e.g. "tea poster"

Invited talk: Giorgio Metta

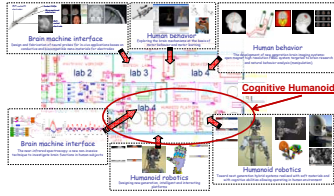
iit
 Cognitive Humanoids Laboratory @iit
 Giorgio Metta



iit
 The Italian Institute of Technology



iit Robotics, Brain and Cognitive Sciences Department



iit Our main focus

- The main focus of the Cognitive Humanoids Laboratory activities is in the implementation of biologically sound models of cognition in robots of humanoid shape
- This has the two-fold aim of
 - furthering our understanding of brain functions
 - realizing robot controllers that can learn and adapt from their mistakes

iit By means of...

- Reverse engineering:
 - Study and be inspired by biological systems
- Models:
 - Robots and mathematical/control models
- Global approach:
 - Sanity check by implementing everything on a real physical platform, complete systems, real feedback

iit Our approach

Guiding Philosophy

- Cognition cannot be hand-coded
- It is necessarily the product of a process of embodied development
- Initially dealing with immediate events
- Increasingly acquiring a predictive capability

COGNITION = PREDICTION

iit Why?

- Purposeful manipulatory actions are founded on predictions of physical events and the effects of one's own actions



From Claes von Hofsten and Kerstin Rosander

iit Main themes

- Mindware
 - Architecture, machine learning, and development
 - Embodied active perception
 - Computational motor control
- Bodyware
 - Mechatronics design
 - Tactile, haptic & force sensing
 - Microelectronics
 - New and future technologies

iit Two main platforms

iit What I'm going to tell you

- Older examples of our research (to 2003)
 - Mirror neurons models
 - Motor based action recognition
- Newer examples (2004-2008)
 - Mechatronics
 - Humanoid robots
- Directions (2009)
 - More mechatronics
 - New cognitive behaviors

iit F5 canonical neurons looking at objects

A B

Observation + action Observation only

iit Mirror Neurons looking at others

The neuron is activated by "seeing" someone else's hand performing a manipulative action and while the monkey is performing the same action

A B

The type of action seen is relevant

From: Fadiga, L., L. Fogassi, V. Gallese, and G. Rizzolatti. Visuo-motor Neurons: ambiguity of the discharge or "motor" Perception? International Journal of Psychophysiology, 2000, 35, p. 165-177.

iit Objects come to existence because they are manipulated

Which edge should be considered?
 Maybe some cruel grad-student glued the cube to the table.
 Color of cube and table are poorly separated
 Cube has misleading surface pattern

With Paul Fitzpatrick

iit Into object affordances...

With Paul Fitzpatrick

iit The qualitative "geometry" of poking

Estimated probability

Direction of movement [deg]

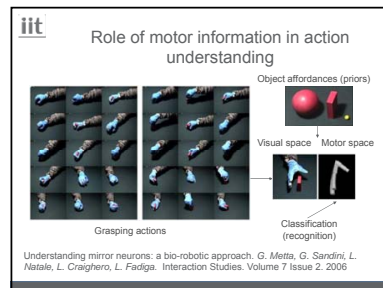
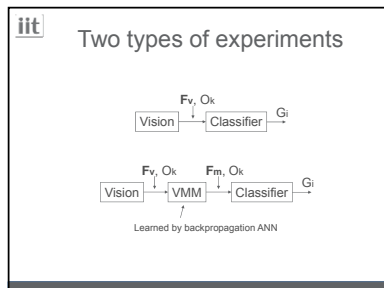
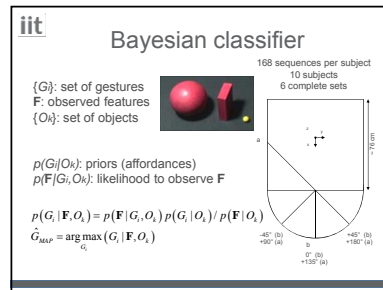
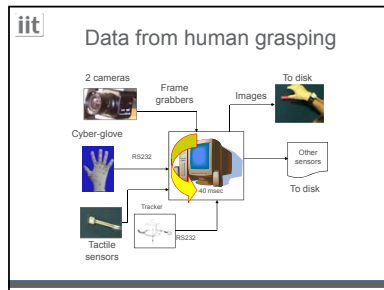
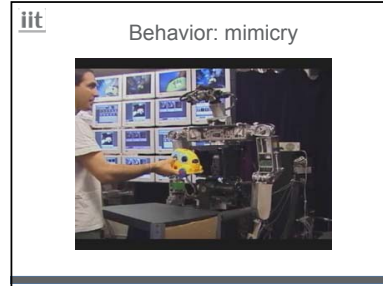
iit Interpreting observations

Invoking the object's natural rolling affordance Going against the object's natural rolling affordance

Demonstration by human

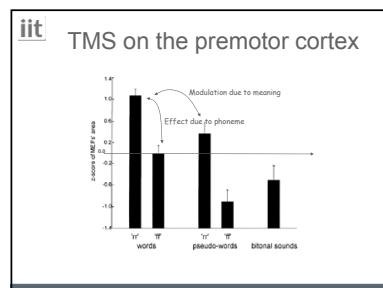
Mimicry in similar situation

Mimicry when object is rotated




iit Some results...

	Exp. I (visual)	Exp. II (visual)	Exp. III (visual)	Exp. IV (motor)
Training				
# Sequences	56	24	64	24
# of view points	1	1	4	1
Classification rate	100%	100%	97%	98%
# Features	5	5	5	15
# Modes	5-7	5-7	5-7	1-2
Test				
# Sequences	8	96	32	96
# of view points	1	4	4	4
Classification rate	100%	90%	80%	97%



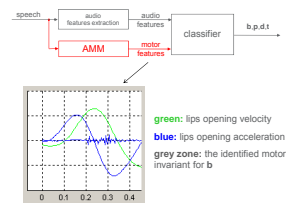
iit **Data collection**



- 9 speakers, 74 (pseudo)words and syllables
- magnetic tracking of tongue, lips and teeth
- ultrasound imaging of tongue
- video of face
- laryngography of vocal folds

The mirror system and speech: with L. Fadiga, L. Craighero & the Contact people

iit **As before...**



green: lips opening velocity
blue: lips opening acceleration
grey zone: the identified motor invariant for b

iit **The iCub: quick summary**


The **iCub** is the humanoid baby-robot designed as part of the **RobotCub** project

- The iCub is a **full humanoid robot** sized as a three and half year-old child.
- The total height is **104cm**.
- It has **53 degrees of freedom**, including articulated hands to be used for manipulation and gesturing.
- The robot will be able to **crawl and sit** and autonomously transition from crawling to sitting and vice-versa.
- The robot is **GPL/FDL**: software, hardware, drawings, documentation, etc.

iit **Degrees of freedom**


- Head: vergence, common tilt + 3 dof neck
- Arms: 7 dof each
 - Shoulder (3), elbow (1), wrist (3)
- Hands: 9 dof each ▶ 19 joints
 - 5 fingers ▶ underactuated
- Legs: 6 dof each
 - Hip (3), knee (1), ankle (2)
- Waist: 3 dof

$\Sigma = 53 \text{ dof}$ (not counting the facial expressions)




iit **Sensorization**

- For each joint:
 - Position (some absolute, some incremental):
 - Magnetic absolute position sensors
 - Encoders
 - Hall-effect sensors
 - Torque/tension
 - Limb level, but work in progress to add joint level torque sensing
 - Current consumption
 - Temperature (monitor, safety)
 - Safe operation (but we have a disclaimer)



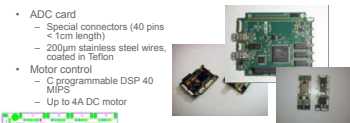
iit **In addition...**

- Cameras
 - Pointgrey Dragonfly firewire cameras
- Force/torque sensors
 - Custom development: 6 axial
- Microphones, speaker
 - Standard condenser electret miniature microphones
 - Pinnae
- Gyroscopes, linear accelerometers
 - Xsense: Mtx
- Tactile sensors, skin, fingertips
 - Capacitive sensors (more later)



iit **Custom electronics**

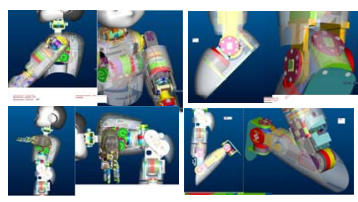
- ADC card
 - Special connectors (40 pins < 1cm length)
 - 200µm stainless steel wires, coated in Teflon
- Motor control
 - C programmable DSP 40 MIPS
 - Up to 4A DC motor



80x30mm
Motorola DSP56F807 (5680x family)
MAC instructions
PWM generation
ADC
Digital I/O
Can bus
C programmable

58x42mm

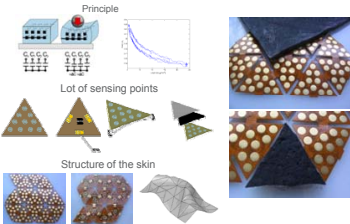
iit



iit In the pipeline...

- Force control: joint level sensors, SEL or strain gauges based sensing
- Skin/tactile sensors: almost everywhere on the robot surface
- Robot general improvements: e.g. zero-backlash everywhere, better control electronics, higher resolution position sensors

iit The skin (ideas...)

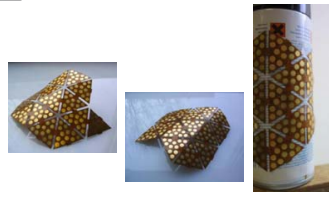


Principle

Lot of sensing points

Structure of the skin

iit




iit Some initial results




iit Fingertips

- Capacitive pressure sensor with 12 sensitive zones
- Natural round shape, 14.5 mm long and 13 mm wide, for iCub
- Embedded electronics: twelve 16 bits measurements of capacitance (either all 12 taxels independently at 50 Hz or an average of the 12 taxels at about 500 Hz)



iit More examples...



With Peter Ford-Domney (NISE-EM, Lyon)

With Auke (Spent, Ludovic Righetti, Sarah Trötschel (EPFL))

With a lot of students

@ RobotCub summer school 2008

With VstLab (IST Lisbon)

iit External projects

- The Cognitive Humanoids Laboratory participates in several externally funded projects, namely:
 - RobotCub, humanoid design & cognition
 - ITALK, development of language
 - Poeticon, structure of action
 - CHRIS, safety and cognition
 - eMorph, VLSI for vision
 - RoboSkin, skin technology
 - Viactors, actuation technology

iit People (platform)


- Giulio Sandini, David Vernon: guidance & mentoring
- Lorenzo Natale, Francesco Nori: Software, testing, calibration
- Marco Maggiali, Marco Ranzazza: firmware, DSP libraries, tactile sensing
- Francesco Becchi, Paolo Pino, Giulio Maggiali, Gabriele Careddu: design and integration
- Roberto Puddu, Gabriele Tabbita, Walter Fancelli: assembly
- Nikos Tsagarakis, William Hinojosa: legs and spine, force/torque sensors
- Bruno Bonino, Fabrizio Laroba, Claudio Lotini: electronics and wiring
- Luciano Pitters, Davide Deleplane: wiring
- Mattia Salvi: CAD maintenance
- Stefano Sicari: managing quotes, orders and spare parts
- Giovanni Stelli: hand sensors
- Ricardo Beira, Luis Vargas, Miguel Praca: design of the head and face
- Paul Fitzpatrick & Alessandro Scato: software middleware
- Alberto Parmiggiani: joint level sensing
- Alexander Schmitz: fingertips
- Nestor Nave: small Harmonic Drive integration
- Ravinder Dahiya: FET-PVDF tactile sensors
- Lorenzo Jamiohi: fingertips
- Daniel Roussey: construction
- Ludovic Righetti: simulation and initial torque specification

iit


People (scientific)

- The Cognitive Humanoids Laboratory @ IIT
 - Lorenzo Natale, Francesco Nori
 - Marco Maggiali, Marco Randazzo, Ugo Pattacini
 - Lorenzo Jansson, Gerena Ivaldi, Alessandra Scutti
 - Matteo Fumagalli, Arjan Gijsberts, Andrew Danks
 - Ryo Saegusa, Alexander Schmitz, Ravinder Dahiya
 - Alberto Farnigiani, Chiara Bartolozzi
 - Tourik Bentaleb, Biso Maurizio
 - Carlo Ciliberto, Francesco Raa
 - Massimiliano Izzo, Vishwanathan Mohan
 - Madia Salvi, Alessandro Scio
- Giulio Sandini: Italian Institute of Technology
- Luciano Fadiga, Lalla Craighero: University of Ferrara
- Claes von Hofsten, Kerstin Rosander: Univ. of Uppsala
- Jose' Santos-Victor, Alex Bernardino, Manuel Lopez: IST Lisbon
- Claudio Castellini: University of Genoa
- Peter Dominey: INSERM, Lyon
- Paul Fitzpatrick: once at MIT, now University of Genoa

iit



The iCub humanoid robot:
an open platform for research in embodied cognition



Giorgio Metta
Italian Institute of Technology
CogX summer school,
Stockholm March, 2009

Summary of the talk

- Overview of the hardware
- The software architecture
- Some examples of code

Collaborations
the RobotCub Consortium

U. of Geneva Claudio Geronzi Giorgio Metta David Vernon Lucrezia Nanni Francesco Nori Paul Fitzpatrick	ISRO/IST Paolo Dario Luca Colletti Giovanni Staudin	U. of Zurich Raffaello Farnetoli Gabriel Gomez	U. of Uppsala Claes Van Hatalow Karim Rosenbhar	U. of Ferrara Luigi Di Paolo Luigi Giuffridè Andrey Obyedkov
U. of Hertfordshire Martin Donohue Christopher Nahriv Markus Buehner Nahem Assaf Mizra	IST - Lisbon João Santos-Vitor Alexandre Bernardino Ricardo Araujo Miguel Praça	U. Salford Dariusz Czerniewski John Gray Nick Trigoniadis	EPFL Auke Ijspeert Benoît Delpierre Ludovic Righetti	Telerobot S.r.l. Francesco Biondi David Coriani

Further links:
the RobotCub international advisory board

MIT Rodney Brooks	ATR Gordon Cheng
Yusuo Kumiyoshi University of Tokyo	Juergen Konczak University of Minnesota
NICT Hideki Kazima	NICT (GRL)

The Hardware

The iCub: quick summary


The iCub is the humanoid baby-robot designed as part of the RobotCub project

- The iCub is a **full humanoid robot** sized as a three and half year-old child.
- The total height is **104cm**.
- It has **53 degrees of freedom**, including articulated hands to be used for manipulation and gesturing.
- The robot will be able to **crawl and sit** and autonomously transition from crawling to sitting and vice-versa.
- The robot is **GPL/FDL**: software, hardware, drawings, documentation, etc.

Degrees of freedom


- Head: 3 dof eyes + 3 dof neck
- Arms: 7 dof each
 - Shoulder (3), elbow (1), wrist (3)
- Hands: 9 dof each
 - 5 fingers, underactuated (19 joints)
- Legs: 6 dof each
 - Hip (3), knee (1), ankle (2)
- Waist: 3 dof

$\Sigma = 53 \text{ dof}$ (not counting the facial expressions)



Sensorization

- For each joint:
 - Position (some absolute, some incremental):
 - Magnetic absolute position sensors
 - Encoders
 - Hall-effect sensors
 - Torque/tension
 - Limb level, but work in progress to add joint level torque sensing
 - Current consumption
 - Temperature (monitor, safety)





In addition...

- Cameras
 - Paintgrey Dragonfly firewire cameras
- Force/torque sensors
 - Custom development: 6 axial
- Microphones, speaker
 - Standard condenser electret
 - miniature microphones
 - Pinnac
- Gyroscopes, linear accelerometers
 - Xsense: Mtx
- Work in progress: tactile sensors, skin, fingertips
 - Capacitive sensors (more later)



Integration: custom electronics

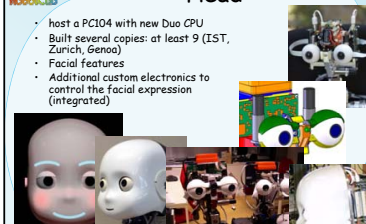
- Motor control
 - C programmable DSP 40 MIPS
 - Up to 4A DC motor/2 Brushless
- PC104 board
 - 2 firewire
 - 4 can bus interfaces


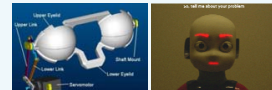
80x30mm
Motorola DSP56F807 (5680x family)
P1016 generation
ADC
Digital I/O
Can bus
C programmable

Head

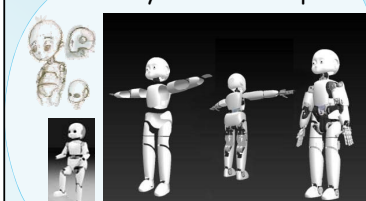
- host a PC104 with new Duo CPU
- Built several copies: at least 9 (IST, Zurich, Genoa)
- Facial features
- Additional custom electronics to control the facial expression (integrated)



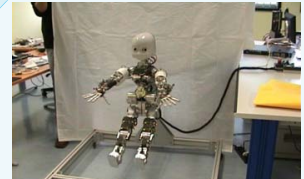
Facial expressions

Body cover: concept



The robot yoga!




Preprogrammed movements


Work in progress: more sensors

The skin


Principle

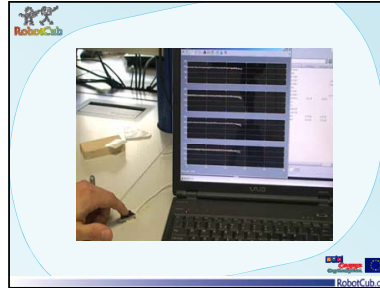
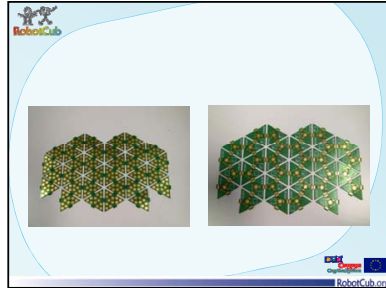


Lot of sensing points



Structure of the skin





Fingertip

3D CAD → Electrode fabrication → Complete prototype

Outline → Silicon cover → Electronics

6-axial force/torque sensor

- Semiconductor strain gauges
- On board signal conditioning, sampling, and calibration
- Digital output: CAN bus

By Nikos Tsagarakis and Darwin Caldwell
Electronics: Claudio Lorini

open source

Wiki
CVS
Part lists
Drawings

Promoting the iCub

- RobotCub Open Call
 - 31 participants, 6 winners will receive a copy of the iCub free of charge
- Further development
 - FP7 project ITALK: 4 iCub's will be built, language
- Collaborations
 - Univ. of Karlsruhe: new and longer legs
- Simulator:
 - Both Open Source and as a model in Webots



Examples:

Multimodal Saliency-Based Bottom-Up Attention: A Framework for the Humanoid Robot iCub

James Blanton¹, Alessio Liguori¹, Alessio Bonarino¹, James Hertzberg¹, Gianantonio "Robi" Polini¹

Presented at ICRA'08, May 21, Pasadena, US

¹ Italian Space Agency, Space Robotics Programme
© University of Göttingen, Germany

The Software Architecture

Let's start from the Hardware

The sad fate of most robot software

- Writing software is difficult and time consuming
- Our software tends to die with our projects
- Sad! Software collaboration speeds things up

Hardware Diversity

- Research groups that all use a **specific robot** (Khepera, Pioneer, AIBO, ...) often form a **natural software community**
 - But each alone is a **small subset** of robotics
- Groups developing **new robots face obstacles**
 - Differences in sensors, actuators, bodies...
 - Differences in processors, operating systems, libraries, frameworks, languages, compilers...
 - Big barriers to software collaboration

the PC

- Constant hardware flux
 - Parts change rapidly
 - Interfaces change slowly
- Lots of software grew and evolved alongside the changing hardware
 - Parts change rapidly
 - Interfaces change slowly
 - "Modularity" is rewarded

A Modular Approach

- Robot software is notoriously **hardware-specific** and **task-specific**
- Both hardware and target tasks **change quickly**, even within the lifetime of one project
- Our humanoid robots are far more complex than one person can build and maintain, both in terms of hardware and software
- They need to be **modular**

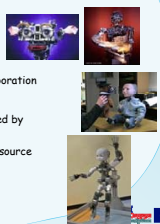
Modular Approaches

- Modular approaches to robotics:
 - **Player/Stage** (mobile robotics)
 - Robot control (Khepera, Pioneer), simulator
 - **Orocos** (industrial robotics)
 - Real-time control, kinematics library, other libs
 - **YARP** (humanoid robotics)

SOURCE: Chad Jenkins, June 11, 2009, Workshop Introduction Robotics 2009 Workshop on Modular Foundations for Control and Perception

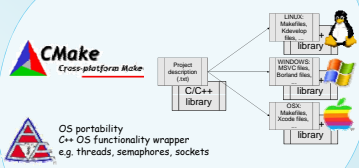
Yet Another Robot Platform

- YARP is an open-source middleware for humanoid robotics
- History
 - An MIT / Univ. of Genoa collaboration
 - Born on Kismet, grew on COG
 - With a major overhaul, now used by RobotCub consortium
 - Exists as an independent open source project
 - C++ source code



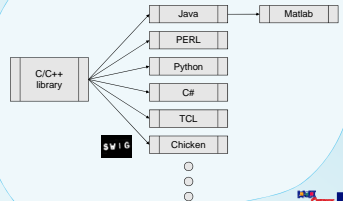
IDE/OS Portability

CMake
Cross-platform Make



OS portability
C++ OS functionality wrapper
e.g. threads, semaphores, sockets

Language Portability (SWIG)



What is YARP for?

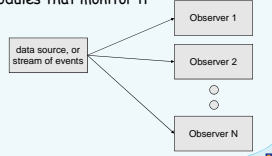
- Factor out **details of data flow between programs** from program source code
 - Data flow is very specific to robot platform, experimental setup, network layout, communication protocol, etc.
 - Useful to keep "algorithm" and "plumbing" separate
- Factor out **details of devices used by programs** from program source code
 - The devices can then be replaced over time by comparable alternatives; code can be used in other systems

What is YARP for?

- Factor out **details of data flow between programs** from program source code
 - Data flow is very specific to robot platform, experimental setup, network layout, communication protocol, etc.
 - Useful to keep "algorithm" and "plumbing" separate
- Factor out **details of devices used by programs** from program source code
 - The devices can then be replaced over time by comparable alternatives; code can be used in other systems

the Observer pattern

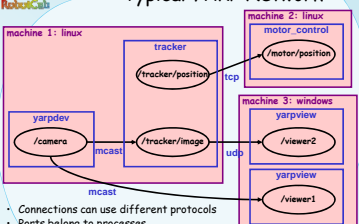
- Data source knows nothing about identity of modules that monitor it



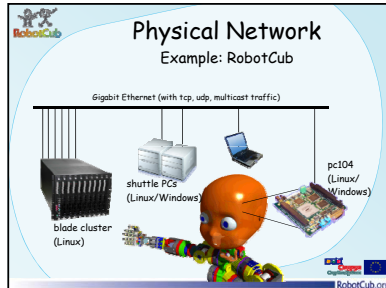
YARP Ports

- We follow the **Observer** design pattern.
- Special "Port" objects deliver data to:
 - Any number of observers (other "Port"s) ...
 - ... in any number of processes ...
 - ... distributed across any number of computers/OSes ...
- using any of several underlying communication protocols with different technical advantages, streaming or RPC
- This is called the YARP Network

Typical YARP Network



- Connections can use different protocols
- Ports belong to processes
- Processes can be on different machines/OS



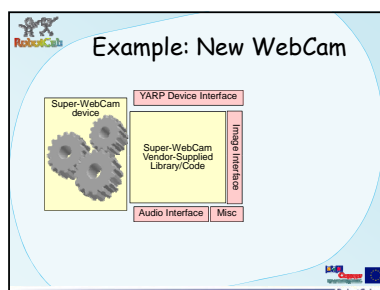
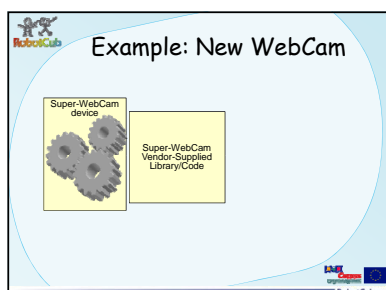
- Why is all this useful?**
- We've separated out most of the **plumbing**
 - We get to change it **dynamically** (handy)
 - More importantly, we have better **modularity**
 - Programs can be **moved around** as load and OS/device/library dependencies dictate
 - Fundamental protocol for communication can be **changed** without affecting programs
 - Better chance that your code can be **used by others** (even just within your group)

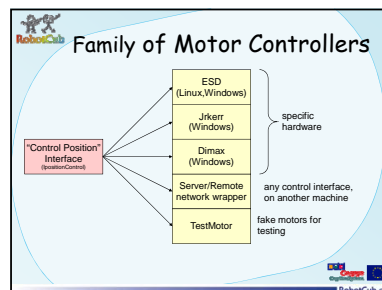
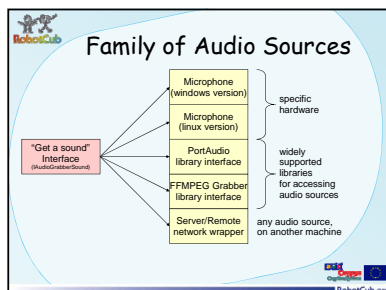
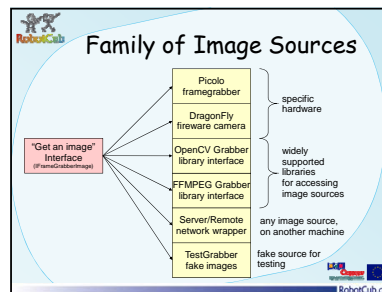
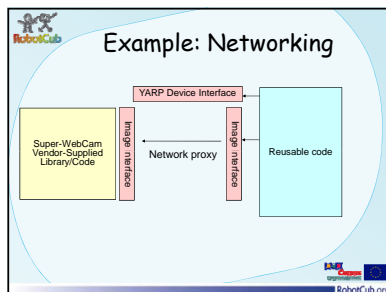
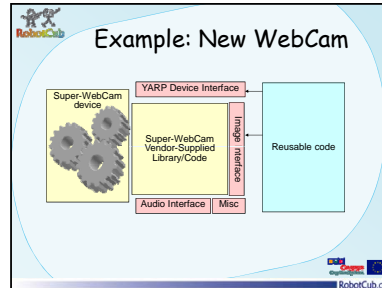
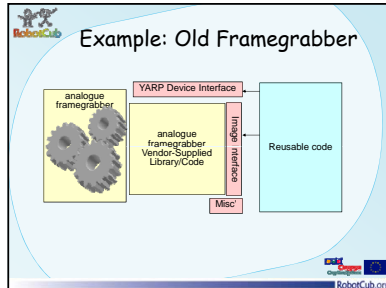
- What is YARP for?**
- Factor out **details of data flow between programs** from program source code
 - Data flow is very specific to robot platform, experimental setup, network layout, etc.
 - Useful to keep "algorithm" and "plumbing" separate
 - Factor out **details of devices used by programs** from program source code
 - The devices can then be replaced over time by comparable alternatives; code can be used in other systems

- What is YARP for?**
- Factor out **details of data flow between programs** from program source code
 - Data flow is very specific to robot platform, experimental setup, network layout, etc.
 - Useful to keep "algorithm" and "plumbing" separate
 - Factor out **details of devices used by programs** from program source code
 - The devices can then be replaced over time by comparable alternatives; code can be used in other systems

- YARP Devices**
- Basic idea: if you view your devices **through well thought out interfaces**, the impact of device change can be minimized.
 - There are three separate concerns related to devices in YARP:
 - Defining interfaces for **device families**
 - Implementing **specific drivers** for particular devices
 - Implementing **network wrappers** for interfaces

- YARP Devices**
- **New devices** come out all the time - needs to be easy to connect them to existing code
 - YARP needs a **minimal "wrapper" class** to match **vendor-supplied library** with relevant interfaces that capture common capabilities
 - YARP encourages separating configuration from source code
 - Devices and communications remain distinct concerns





Why is this useful?

- Allows **collaboration** between groups whose robots have different devices
- Makes **device changes** less painful
- Devices and communications are orthogonal features
 - Can **switch** from remote use of device to local use and vice versa without pain
 - Local use can be **very efficient**, just an extra virtual method call

Publication of software

- The **literature** of a research community both expresses its ideas, and **aids in their evolution**
 - Published ideas are read, evaluated, and built upon
 - Useful advances get published
- **Publication of software** can speed progress
 - Facilitates **evaluating and comparing** approaches
 - Brings **new research topics** into reach

Examples of Code

A (very) simple example: read data to/from a port

```

[on terminal 1] yarp server
[on terminal 2] yarp read /read
[on terminal 3] yarp write /write /read
    
```

```

$ yarp write /write /read
Port write listening at tcp://127.0.0.1:10012
yarp: Sending output from /write to /read using tcp
Added output connection from "/write" to "/read"
hello yarp
1 2 3

$ yarp read /read
Port read listening at tcp://127.0.0.1:10002
yarp: Receiving input from /write to /read using tcp
1 2 3
    
```

How do we get this?

Let's now write a simple "relay" executable which takes whatever comes from a port and forwards it to another one.

```

int main(int argc, char *argv) {
    Network yarp;
    BufferedPort<Bottle> inPort;
    inPort.open("/relay/in");

    BufferedPort<Bottle> outPort;
    outPort.open("/relay/out");

    while (true) {
        cout << "waiting for input" << endl;
        Bottle "input" = inPort.read();
        if ("input"==NULL) continue;

        Bottle& output = outPort.prepare();
        output->input;
        cout << "writing " << output.toString().c_str() << endl;
        outPort.write();
    }
    return 0;
}
    
```

Connect the new module to our network

```

yarp disconnect /write /read
yarp connect /write /relay/in
yarp connect /relay/out /read
    
```

how the network grows

It is easy to add, for example, another reader... Processes can run on different machines, with different OS

```

yarp connect /relay/out /read
yarp connect /relay/out /read2
...
    
```

iCub ports

- YARP ports are the interface to the robot
- For example:
 - /icub/cam/left
 - /icub/cam/right
 - /icub/head/state:o
 - /icub/head/command:i
 - /icub/head/rpc:i
 - /icub/right_arm/state:o
 - ...
- see http://eris.liralab.it/wiki/iCub_joints for a more complete list

Example of modularity: The ODE Simulator

Replace a full robot: opens the same ports as the robot, just replace `icub` -> `icubSim` in the port names

http://eris.liralab.it/wiki/Simulator_README

Read sensors from the robot

```

yarpview -name /view/right
yarpview -name /view/left

$ yarp read /read2
yarp: Receiving input from
icubSim/head/state:o using tcp
0.0 0.0 0.0 5.900092 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 5.900092 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0

$ yarp read /read1
yarp: Receiving input from
icubSim/head/state:o
using tcp
-32.0 0.0 0.0 -10.0 0.0 0.0
-32.0 0.0 0.0 -10.0 0.0 0.0
-32.0 0.0 0.0 -10.0 0.0 0.0
-32.0 0.0 0.0 -10.0 0.0 0.0
-32.0 0.0 0.0 -10.0 0.0 0.0
-32.0 0.0 0.0 -10.0 0.0 0.0
    
```

Controlling the motors

To control the motor it is enough to send messages to the robot ports
 However, this becomes easily tedious
 We can use YARP's support for devices to simplify this task
 Each part of the robot (or sensory system) can be seen as a YARP device implementing a given set of interfaces

Device and interface creation

```

Property options:
options.put("device", "remote_controlboard");
options.put("local", "test/right_arm");
options.put("remote", "/icub/right_arm");
// create a device
PosDriver robotDevice(options);

IPositionControl *pos;
IEncoders *enc;

robotDevice.view(pos);
robotDevice.view(enc);

// The process will create the following
// (local) ports:
// test/right_arm/state:
// test/right_arm/command:
// test/right_arm/rpc:

And will automatically connect them to
the server, on the following (remote):

/cubSim/right_arm/state:
/cubSim/right_arm/command:
/cubSim/right_arm/rpc:
    
```

an (almost) working example...

```

int np=0;
pos->getAxis(&np);
Vector tmp_speed, command;
tmp.resize(np, 0.);

// set accelerations and speeds
pos->setRefAccelerations(accelerations.data());
pos->setRefSpeeds(speeds.data());

int times=0;
while(true) {
if (times%2)
// set command to a certain value
else
// set command to another value

pos->positionMove(command.data());

Time::delay(5); // wait some time
// read encoders
enc->getEncoders(encoders.data());
printf("%d %d %d %d %d %d\n", encoders[0]...);
}
    
```

For a list of interfaces and their methods

http://eris.lirilab.it/yarpdoc/d4/dd6/namespacenyarp_1_1dev.html

Details on IPositionControl:
http://eris.lirilab.it/yarpdoc/d4/d03/classyarp_1_1dev_1_1IPIC.html

or IEncoders:
http://eris.lirilab.it/yarpdoc/d4/d57/classyarp_1_1dev_1_1IEncoders.html

Image Processing Example

```

RobotPort imageCubPos@igbo -> imagePort;
imagePort->open("ImagePos@igbo");
// read an image
imageCubPos@igbo->image = imagePort.read();
//do something with the image, for example cycle through all pixels
for (int i=0; i<image->width(); i++)
for (int j=0; j<image->height(); j++) {
    PixelRgb pixel = image->pixel(i,j);
    // very simple test for blue/white
    // make some blue level exceeds red and green by a certain factor
    if (pixel.blue > 1.2 * (pixel.red + pixel.green) * 255) {
        // blue is c
        // green is g
        // red is r
    }
}
if (c>g) {
    // blue is c
} else {
    // green is g
}
printf("Best guess at blue target: %d %d %d", r, g, b);
    
```

Thanks for your attention!

More resources:

Online resources:

- RobotCub home page: <http://www.robotcub.org>
- YARP home page: <http://yarp.cnr.it>
- Cub software repository and wiki: <http://eris.lirilab.it/cub>
- RobotCub wiki: <http://wiki.ros.org/robotcub>

Some papers:

- G. Metta, G. Sandini, D. Vernon, L. Natale and F. Nori. The iCub Humanoid Robot: an open platform for research in embodied cognition. Workshop on Performance Metrics for Self-Organizing Systems, National Institute of Standards and Technology, Washington DC, USA, August 13-21, 2008.
- P. Fitzpatrick, G. Metta, and L. Natale. "Twenty-08 Long-Lived Robot Genes". Robotics and Autonomous Systems, Volume 56(2), pp. 23-45, Elsevier (2007).
- S. D'Agellier, L. Righetti, L. Natale, F. Nori, G. Metta and A. Tognetti. A modular, bio-inspired architecture for movement generation for the humanoid robot iCub. IEEE RAJEROB2 International Conference on Biomedical Robotics and Biomechatronics (BioRob) 2008, October 29-31, Scottsdale, Arizona, USA.
- J. Hornstein, M. Lopez, J. Santos-Victor, F. Lacort, S. Edelkamp. Small-scale architecture for humanoid robots: building auto-actor agent based on the ARFF. Proc. IEEE RSJ International Conference on Intelligent Robots and Systems, IROS, Beijing, China, Oct 9-13, 2006.
- J. Ruscio, M. Lopez, A. Bernardino, J. Hornstein, J. Santos-Victor, S. Edelkamp. Multi-modal Sensory-Based Behavior of iCub. A Framework for the Humanoid Robot iCub. 2008 IEEE International Conference on Robotics and Automation (ICRA), May 19-23, 2008, Pasadena Conference Center, Pasadena, CA, USA.
- Metta, G., Sandini, G., Vernon, D., Caldwell, D., Tognetti, N., Ballo, S., Santos-Victor, J., Egeert, A., Righetti, L., Cagnella, G., Saffin, G., and Beccati, F. 2009. The iCub robot project - an open framework for research in embodied cognitive humanoid robotics. IEEE RAJEROB2 International Conference on Intelligent Robots, December.

People

- Lorenzo Natale, Francesco Nori: Software, testing, calibration
- Marco Maggiali, Marco Randozzo: Firmware, DSP libraries, tactile sensing
- Francesco Becchi, Paolo Pino, Giulio Maggiali, Gabriele Corradu: design and integration
- Gabriele Tabbita, Walter Fancello: assembly
- Nikos Tsagaraki, William Hinojosa: legs and spine, force/torque sensors
- Bruno Bonino, Fabrizio Lorusso, Claudio Lorini: electronics and wiring
- Luciano Pittero, Davide Bellepiane: wiring
- Mattia Scivo: CAD maintenance
- Alberto Zolezzi: managing quotes, orders and spare parts
- Giovanni Shellini: hand
- Ricardo Beira, Luis Vargas, Miguel Proaia: design of the head and face
- Paul Fitzpatrick & Alessandro Scalzo: software middleware
- Alberto Farnagioni: joint level sensing
- Alexander Schmitt: fingertips
- Nestor Nova: small Harmonic Drive integration
- Saverio Dabry: FET-PVDF tactile sensors
- Lorenzo Jamone: fingertips
- Daniel Rousay: construction
- Ludovic Righetti: simulation and initial torque specification

Invited talk: Marc Hanheide

Universität Bielefeld Technical Faculty – Applied Informatics

Learning by Interacting

Marc Hanheide
CogX Summer School
KTH Stockholm, March 2009

Universität Bielefeld Technical Faculty – Applied Informatics

Thanks for inviting me nevertheless!

The Bielefeld conspiracy [usenet, wikipedia]:
The story goes that the city of Bielefeld (population 330,000) in the German state of North Rhine-Westphalia does not actually exist. Rather, its existence is merely propagated by an entity known only as THEM, which has conspired with authorities to create the illusion of the city's existence.

The theory posits three questions:
1. Do you know anybody from Bielefeld?
2. Have you ever been to Bielefeld?
3. Do you know anybody who has ever been to Bielefeld?

Universität Bielefeld Technical Faculty – Applied Informatics

Who am I?

Universität Bielefeld Technical Faculty – Applied Informatics

Applied Informatics

<http://aiweb.techfak.uni-bielefeld.de>

- Pattern Recognition
- Human-Robot Interaction
- Interaction & Dialog Management
- Developmental Robotics
- System Architectures Research

Universität Bielefeld Technical Faculty – Applied Informatics

Our Research in a Nutshell

- Applied Informatics Group
- Focus is "Cognitive Interaction Technology"
 - ... in robotics
 - ... in cognitive architectures

Universität Bielefeld Technical Faculty – Applied Informatics

CoR-Lab

<http://www.cor-lab.org>

- Cognition & Robotics
- ASIMO
- Fundamental research in
 - Cognitive Robotics and Learning (Jochen Steil)
 - Cognitive Systems Engineering (Sebastian Wrede)
 - Hybrid Society (Britta Wrede)

Universität Bielefeld Technical Faculty – Applied Informatics

CITEC

<http://www.cit-ec.org>

- all/most converges into the Cluster of Excellence "Cognitive Interaction Technology"
- 4 "columns"
 - Motor Intelligence
 - Attentive Systems
 - Situational Communication
 - Memory & Learning
- Biology, Psychology, Sports, Informatics, Linguistics, Physics, Mechatronics

Universität Bielefeld Technical Faculty – Applied Informatics

Cognitive Interaction Technology?

- Cognitive:
 - cognitive foundations
 - biological/neural principles
 - linguistic cognitive models
- Interaction:
 - studies
 - interactive learning and tutoring
 - multi-modal and multi-cue interaction
- Technology:
 - architecture (software & systems)
 - interaction behaviors and abilities

Universität Bielefeld Technical Faculty – Applied Informatics

Search Recipes | Submit a Recipe | Top 100 Recipes
 Home >> Cocktails >> Vampire Gimlet
Cocktail Recipe: Vampire Gimlet
 1 1/2 oz Vodka
 1 oz Rose's Concentrated Lime Juice
 1 oz Peppermint Schnapps
 Combine ingredients with ice, shake, strain into cocktail glass.
 © bartweiser.com

Scenario: "cocktail assistant"

- object manipulation
 - objects
 - action
 - task oriented
- assistance
 - step-wise instructions
 - guidance to object positions
 - supervision and correction
- „ego-vision“ system



Universität Bielefeld Technical Faculty – Applied Informatics

Scenario: The curious bi-manual robot



- Mixed-initiative Interaction
- Grasping of learned objects
- Platform: PA-10 Bi-manual

Universität Bielefeld Technical Faculty – Applied Informatics

Applications with BIRON

- mobile home tour robot
 - actively controlled sensors
 - focus on interaction
 - embodiment

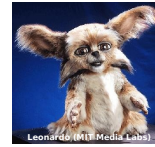


Learning by Interacting



Beyond Machine Learning

- “socially guided machine learning“ [Breazeal]
- Learning is not only
 - collect data sets
 - learn a model (classifier)
 - test the model (classifier)
- but can also
 - start from very generic abilities
 - exploit attention/curiosity/interaction
 - testing, application, and learning are highly intertwined
- The particular learning algorithms are not discussed today!



Teaching by Interaction

- When a tutor communicates the learning stuff, he relates things to the human behavior
- A tutor communicates by
 - speech, gesture,
 - context, history of actions
- Aligned teaching
 - perception and storage of information is guided by communication



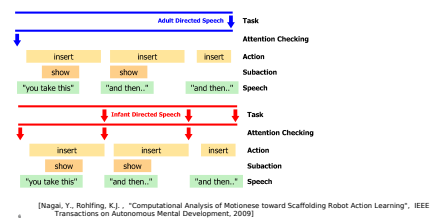
Exemplary Videos - Child-directed



Exemplary Videos - Adult-directed

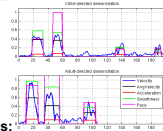


Attention, Teacher's Gesture and Actions



Motionese to Emphasize Actions

- Analyze measurable features
 - average velocity
 - average acceleration
 - smoothness (path/dist)
 - relation of lengths between action and preceding pause
- Characteristics of child-directed actions:
 - More and longer pauses
 - Lower velocity
 - Trajectories (2D) more straight, less smooth

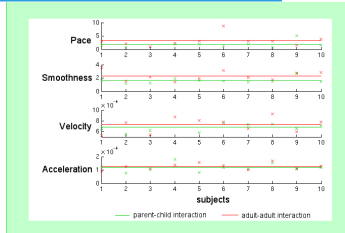


Contrast: Child - Adult

- Speed
- Acceleration
- Pace: Motion Time / Pauses
- Roundness: Trajectory / Distance
- Acoustic Events



Results: Teaching Children and Adults

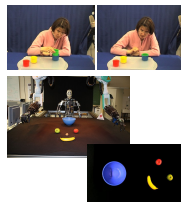


Acoustic Packaging – Origin

- Proposed and termed **acoustic packaging** by Hirsh-Pasek and Golinkoff (1996)
 - Language helps to divide a sequence of events into units
 - Prerequisite: synchrony between language and events
- Study by Brand and Tapscott (2007)
 - Co-occurring infant-directed-speech and motion can help infants to segment the action into smaller units

Acoustic Packaging – Aiding Feedback and Learning in Human Robot Interaction

- Scenario
 - Task demonstration in tutoring situations
 - Human robot interaction
- Goals
 - Generate feedback during tutoring
 - Acoustic packages form early units for further learning processes
 - Finding a common and extensible representation

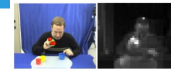


Speech Segmentation – Phoneme Recognizer

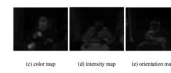
- ESMERALDA
 - Toolkit for tasks needed to build a speech recognizer
 - Incremental Speech Recogniser (ISR)
- Acoustic Model
 - Monophemes from existing acoustic model
 - Training Data: Verbomobil Corpus (lexicon size: ca. 6800)
- n-gram Language Model
 - Training data: Verbomobil corpus phonetically transcribed
 - models phonotactics

A first computational attention model of attention

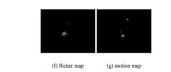
Preparing an Action: Objects are Salient (initial state)



During an Action: Faces are Salient (social information, feedback)



After an Action: Objects are Salient (goal state)



Motivation – Adult vs. Infant Directed Speech

Adult directed speech

- No apparent correlation between A/V segments
- High verbal specification



Infant directed speech

- Highly correlated A/V segments

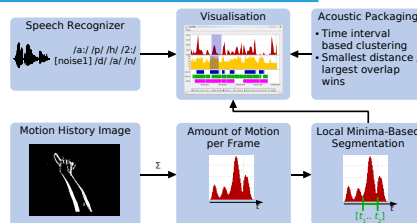


(Schillingmann et al., ICMI 2009)

Requirements

- Vision and speech cues
- Segmentation for speech and vision
- A timestamp concept
 - Accuracy to a certain degree
 - Online / offline usage
- An architecture with the ability to integrate
 - different processing modules
 - operating with varying time steps

Approach – Current State



Universität Bielefeld Technical Faculty – Applied Informatics

Screenshot

Universität Bielefeld Technical Faculty – Applied Informatics

Architecture

- Uses XCF / ActiveMemory
 - Decoupling
 - Persistence
- Modules communicate high level hypotheses
 - XML + binary attachment
 - Interval timestamps

Universität Bielefeld Technical Faculty – Applied Informatics

Learning by interacting

- **Pros of learning by interacting**
 - "what to learn" is explicit
 - implicit task-dependency
 - iterative refinement (aligned representations)
 - intuitive (situated)
 - not necessarily the developmental way
- **Challenge: architectural**
 - pick up IDI concepts: let information generation drive system behavior
 - concurrent behaviors
 - mixed-initiative
 - facilitate learning
- **Challenge: HR-interaction**
 - interaction patterns
 - no longer a pure machine-learning issue
 - appropriate interactive behaviors
 - feedback perception

Universität Bielefeld Technical Faculty – Applied Informatics

A robot learning by interacting

- mobile robot BIRON
- flexible & customized
- platform and scenario since 2001

Universität Bielefeld Technical Faculty – Applied Informatics

Knowledge about environments for mobile robots

- Where is the living room?
- Where can I cook?
- Where are the mugs?
- How to get to the bath room?
- How does the dining room look alike?
- What is a living room?

Where should a robot get this knowledge from?

Universität Bielefeld Technical Faculty – Applied Informatics

Learning by Interacting

- home robots learn
 - one system
 - multi tasks
 - flexibility (everyday environment)
- how do they do it?
 - from demonstration
 - experience & (self-) exploration
 - by interaction

Universität Bielefeld Technical Faculty – Applied Informatics

The Home Tour

- task-oriented use case
- home tour: naive user showing a yet uninstructed robot her/his own living environment
 - spatial representation
 - rooms
 - functional areas (cooking area, private area)
 - objects (cups, coffee machine)
 - "fluent" interaction without exhaustive training
 - autonomous, "non-tweaked" system
- flexible and generic learning-enabling behaviors

Universität Bielefeld Technical Faculty – Applied Informatics

"Hello"

- speech recognition
- voice direction
- detection of
 - face
 - leg
- person tracking (reviewed later on)

Excursion: Looking into people's faces

- **Faces are in particular relevant for interaction**
 - identity
 - emotion
 - feedback cues
- **Perception is difficult**
 - changing light
 - active camera



26

Active Appearance Model (AAM)

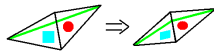
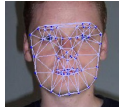
- **image processing approach (Cootes, Edwards and Taylor 1998)**
 - object description via parameter vector
 - interpretation by synthesis with a generative
- **model:**
 - shape model
 - texture model / grey level model
- **search method**



27

Texture Model

- texture: pixel intensities of an object
- create „shape-free“ object textures:
 - Delaunay Triangulation
 - image warping to the mean shape



28

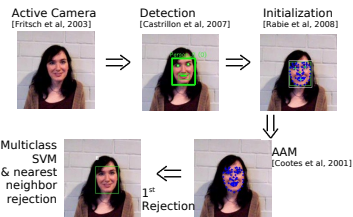
Search Method

- find \vec{a} for a new image by minimization of the grey level difference (reconstruction error):

$$d \vec{g} = \vec{g}_{image} - \vec{g}_{model}$$
- difficult optimization problem!
- idea: pre-compute linear relation: $d \vec{a} = R \cdot d \vec{g}$
- displace model parameters for training examples: vary and size, position and rotation \vec{a}
- find R using *Multivariate Linear Regression*

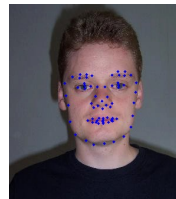
29

Analyzing Faces using AAM Techniques



30

Shape Model



- set of training images with landmarks
- feature point correspondence in all images
- image + shape = appearance

31

Appearance Model

- use PCAs to build shape and texture model
- combination of shape and texture model yields appearance model
- third PCA on all combined vectors
- description of an appearance via *appearance parameter* \vec{a}
- several normalizations are necessary



32

Fitting to new Images

1. initial estimation of the parameters \vec{a}
2. use \vec{a} to generate shape and texture
3. calculate texture error: $d \vec{g} = \vec{g}_{image} - \vec{g}_{model}$
4. predict parameter modifications: $d \vec{a} = R \cdot d \vec{g}$
5. test new parameters: $\vec{a} := \vec{a} - k \cdot d \vec{a}$ where $k \in \{1.5, 0.5, 0.25, \dots\}$
6. select best \vec{a}
7. iterate, until the texture error $d \vec{g}$ does not improve any more

33

Universität Bielefeld Technical Faculty – Applied Informatics

Example

- left: fitting successful
- right: convergence into local minimum

Universität Bielefeld Technical Faculty – Applied Informatics

So much for Faces

- **Faces are just one feedback channel...**
 - Gestures with head, limbs, posture,...
 - Prosodic information also conveys e.g. emotional states
 - explicit verbal statements
- **let's continue teaching the robot something about the apartment**

Universität Bielefeld Technical Faculty – Applied Informatics

"This is the living room"

- **Topological mapping (Human augmented mapping)**
- **real world**
 - changing furniture
- **features mixed initiative**
 - learning by interacting
 - robot's curiosity
 - novelty detection on geometric features

Universität Bielefeld Technical Faculty – Applied Informatics

"Biron, follow me"

- **intelligent following**
- **obstacle avoidance**
- **using person tracking**
- **real world**
 - narrow door passages (path planning and obstacles avoidance)
 - interactive error recovery

Universität Bielefeld Technical Faculty – Applied Informatics

"This is a mug"

- **object attention**
- **real world**
 - based on skin-color tracking
 - skin-color based tracking has its limits
 - Alternatives: **3D body tracking**

Universität Bielefeld Technical Faculty – Applied Informatics

"This is a chair"

- **Camera orientation as implicit feedback**

Universität Bielefeld Technical Faculty – Applied Informatics

3D Body Tracking

- **One Camera:**
 - Model based on Cylinders
 - Matching 3D Model und 2D Features
 - Generic Motion Model (no pre-defined motion classes)
 - Particle Filtering Approach: Adaptable to Processor Power
- **Enables Tracking of 3D Body Motion by a Mobile Robot**

[Schmidt, J., Kwielek, B., & Fritsch, J. Kernel Particle Filter for Real-Time 3D Body Tracking in Monocular Color Images Proc. of Automatic Face and Gesture Recognition, 2006, 567-572]

Universität Bielefeld Technical Faculty – Applied Informatics

Initialization and Recovery


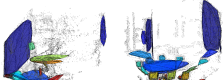
- **robust face detection and tracking**
 - learn and update human's appearance model:
 - skin color (mixture of gaussians in RG color space)
 - shirt color (color histogram)
- **constrained pose estimation (only 5 DOF)**
 - allow less body movements
 - head and hands position
 - torso color segmentation
 - use pose for initializing tracking
 - use pose for error recovery

Universität Bielefeld Technical Faculty – Applied Informatics

Not only humans are of interest...

3-D scene Perception

- Scene description by planar surfaces
- Sensor: Swissranger ToF Camera
- Modeling of human-made environment independent from:
 - Colors, textures, view point, specific furniture, objects, ...



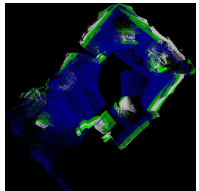



[Beuter, N., Swadzba, A., Schmidt, J., "Simultaneous Tracking And Scene Reconstruction For Robot Perception", Workshop for Cognitive Humanoid Vision: IEEE-RAS, 2008]

Universität Bielefeld Technical Faculty – Applied Informatics

3D scene perception

- integrate with SLAM

Universität Bielefeld Technical Faculty – Applied Informatics

Wrap-Up

- Enable technology to learn by interaction
- Equip robots with perceptual capabilities to facilitate interaction and learning
- Target at closed-loop, "social machine learning"

Find/investigate an architectural paradigm focusing on

- multimodal fusion
- learning & memorization
- flexible w.r.t. system and platforms

Universität Bielefeld Technical Faculty – Applied Informatics

Memory & Memory Processes „brains in machines and developers“

Marc Hanheide
CogX Spring School
KTH, Stockholm, March 2009

Universität Bielefeld Technical Faculty – Applied Informatics

Memory and Cognitive Systems

- **Cognitive systems without memory?**
- Memory is a common feature of most cognitive systems
- **Our ambition**
 - design systems around memory architectures to support
 - life-long learning and adaptation
- **Objectives**
 - be informed by cognitive foundations in nature/humans
 - Allows easier binding of distributed information and knowledge
 - Allows shared consolidation of compact representations
 - Gives additional hints for functional decomposition
- **Memory is not passive, but an active process...**

Universität Bielefeld Technical Faculty – Applied Informatics

Cognitive Foundations: Memory Processes

Silke Klinger

- A very simplified view
- but quite accepted in cognitive psychology

Universität Bielefeld Technical Faculty – Applied Informatics

Temporal Processes

Universität Bielefeld Technical Faculty – Applied Informatics

Temporal Processes – Working Memory

(Baddeley, 2003)

Universität Bielefeld Technical Faculty – Applied Informatics

LONG-TERM MEMORY SYSTEMS

	PROCEDURAL MEMORY	PERCEPTUAL MEMORY	SEMANTIC MEMORY	EPISODIC MEMORY
Encoding and consolidation	Basal ganglia, motor-related areas	Primary and association cortex	Posterior sensory cortex	Cerebral cortex, limbic structures
Storage	Basal ganglia, motor-related areas	Primary and association cortex	Prefrontal sensory cortex	Cerebral cortex (mainly association areas), limbic regions
Retrieval	Basal ganglia, motor-related areas	Primary and association cortex	Posterior sensory cortex	Frontotemporal cortex (right), limbic regions

(Tulving & Markowitsch)

Universität Bielefeld Technical Faculty – Applied Informatics

A principle processing model – SPI model

- model according to Tulving (e.g. 2001)
 - Encoding in serial
 - Storage in parallel
 - Retrieval is independent
- **SPI-model**

Figure 1. A schematic rendering of a model of process-specific, episodic, serial, parallel, independent, SPI among three large memory systems – perceptual (PRS), semantic and episodic.

Universität Bielefeld Technical Faculty – Applied Informatics

SPI model (technically) revisited: Object recognition

- **Encoding in serial**
 - object's appearance (perceptual)
 - object's class (semantic)
 - object's role in an action (episodic)
- **Storage in parallel**
 - we store all this, not to compute it again
 - representation (data types) are different
 - reference between representations is crucial
- **Independent retrieval**
 - other processes shall have access

Figure 1. A schematic rendering of a model of process-specific, episodic, serial, parallel, independent, SPI among three large memory systems – perceptual (PRS), semantic and episodic.

Universität Bielefeld Technical Faculty – Applied Informatics

A Memory for Constructing Intelligent Systems

Two-fold view on “Memories”:

- a cognitive foundation**
 - facilitate learning by interacting
 - close coupling of perception and action based on acquired knowledge
- a constructive paradigm for software development**
 - flexible and evolving processing strategies
 - decomposition strategy into memory processes
 - provide a loosely coupled component model of processing for collaborative development

• how to support the cognitive model?

Universität Bielefeld Technical Faculty – Applied Informatics

History of Memory Architectures

- Working Memory in the focus of “traditional” cognitive architectures**
 - ACT
 - SOAR
- in Bielefeld (in European projects)**
 - VAMPIRE: “Visual Active Memories and Interactive RETrieval”
 - COGNIRON: “COGNitive Robot companioN”
 - iTalk: “Integration and Transfer of Action and Language Knowledge in robots”
 - historically termed “Active Memories”
- related**
 - CoSy, CogX

Universität Bielefeld Technical Faculty – Applied Informatics

Active Memory Approach A Cognitive Systems Viewpoint

- Memory is generally useful in cognitive interaction systems**
 - Store and recall shared multi-modal information
 - Cope with unreliable, outdated information
 - Support learning based on memory content / dynamics
 - Support transparent data fusion processes
- However, there may be different, specialized types of memory in cognitive interaction systems**

taken mainly from VAMPIRE view

Universität Bielefeld Technical Faculty – Applied Informatics

Active Memory Approach A Software Engineering Viewpoint

- But, memory can be useful for software integration, too**
 - Implicit collaboration
 - Shared representations
 - Implicit representation of system state
- Introduces temporal dimension in otherwise transient EDA**
 - Allows bootstrapping of event-driven components
- The active memory is a generic service in an IDI architecture**

taken mainly from VAMPIRE view

Universität Bielefeld Technical Faculty – Applied Informatics

Information-Driven Integration in a Nutshell Manifesto

Focus on the information generated in a system to facilitate efficient software integration in cognitive systems and the construction of loosely coupled, modular architectures with particular support for memory and learning.

Universität Bielefeld Technical Faculty – Applied Informatics

Information-Driven Integration (IDI) in a Nutshell Methodology

- Event-driven architecture (EDA)
- Service-oriented principles (SOA)
- Tuplespace concepts (TS)

Combined in information-driven integration approach!

Universität Bielefeld Technical Faculty – Applied Informatics

IDI Architecture Information-oriented Representation

- XML Documents**
 - Encode information and knowledge
 - XML is a hierarchical data type
 - instead of pre-defined stubs and skeletons (IDL or slide definitions)
 - binary attachments
- IDI Approach**
 - Self-contained and coarse grained
 - Reuse of common XML locations
- Benefits**
 - Unifies variety of message structures
 - Facilitates generic components and algorithms
 - XML Schema support for validation

Universität Bielefeld Technical Faculty – Applied Informatics

IDI Architecture Uniform Access and Extensibility

- XPath for tree navigation**
- IDI Guidelines**
 - Ignore additional content (must-ignore policy)
 - Be location independent (partial specialization)
- Benefits**
 - Declarative
 - Extensibility
 - Facilitates reuse / loose coupling

XPath	Explanation
*/CENTER	Associative access to nodes
*/RECT[w]	Conditional lookup with predicate
...	...

Universität Bielefeld Technical Faculty – Applied Informatics

IDI Architecture Events as Atoms of Communication

- Events**
 - Created by Informers (objects)
 - Represent state change
 - Polymorphic event hierarchies
- Event notifications**
 - Eventually received by interested listeners
 - Extend DM by metadata and binary attachments
- Benefits**
 - Map well to domain events
 - Loose coupling

Universität Bielefeld Technical Faculty – Applied Informatics

IDI Architecture Reactive Programming with Subscriptions

- Subscription**
 - Allows listeners to express interest (subscribe) to specific events
 - Subscription contains filters that evaluate event notification against conditions
 - If a subscription matches, its callback (domain object) is notified
- Client-side matching**
- Benefits**
 - Dynamic registration
 - Expressiveness

```

Subscription
XPathFilter
Condition:
Type: Object
Center position:
(160<x<480) and (120<y<360)
Reliability: >=0.9
    
```

```

<OBJECT[REGION(CENTER(@x>160 and @x<480 and @y>120 and @y<360))]
    
```

Universität Bielefeld Technical Faculty – Applied Informatics

IDI Architecture Matching and Transformation

- Subscriptions composed of filter and transformation functions**
- Message Transforming Functions (MTF)**
 - Extension to filtering
 - May be stateful
 - May modify content
- Benefits**
 - More powerful conditions
 - Modeling of pre- and post-conditions
 - Pre-processing
 - Bound to local state

[Lütkabohle, I., Schaefer, J., Wrede, S., "Facilitating Re-Use by Design: A Filtering, Transformation, and Selection Architecture for Robotic Software Systems". Software Development and Integration in Robotics, 2009]

Universität Bielefeld Technical Faculty – Applied Informatics

Memory Approach: Memory as a Tupespace

- SPI: Independent (associative) retrieval**
- Informed by Tupespaces [Gelernter]**
- Virtual shared memory**
- Tuples (atomic datatypes)**
- High-level actions (in, out, read, ...)**
- Benefits**
 - Abstract level of component interaction
 - Allows modeling of concurrent access
- Active Memory**
 - No tuples but memory elements

```

<-GENERATOR
  include: "ROBOT_DETECTORS"
  InstanceID="0323456789"
  <CLASS: Cup<-CLASS
  <REGION Image="office_122"
  <COORDS x="235" y="245"
  w="65" h="60" />
  </REGION>
</OBJECT>
    
```

Universität Bielefeld Technical Faculty – Applied Informatics

Memory Service: Functions

- Shared repository**
 - Concurrent access
 - Reliable storage
 - Transaction support
- High-level actions**
 - Insert, replace, remove, query, ...
 - XPath for selection
- Memory events**
 - Inform about modifications

Universität Bielefeld Technical Faculty – Applied Informatics

Perceptual Consolidation and Tracking: Hypothesis Anchoring (Binding Perception)

- SPI: Serial Encoding, parallel storage**
- inherently asynchronous processing flow**
- person anchoring is one concurrent behavior**

Universität Bielefeld Technical Faculty – Applied Informatics

Hypotheses Anchoring in Memories

- map percepts to episodes**
- perception is uncertain**
 - miss-classifications, ...
- but...**
 - existence in memory is a binary feature
 - uncertainty needs to be expressed explicitly
- hypotheses concept**

Universität Bielefeld Technical Faculty – Applied Informatics

Hypotheses Anchoring

- generic fusion process based on distance measures**
- specialized function**
 - match
 - insert
 - update

Memory Processes: Our Objectives revisited

- identify the consensus of generic memory processes
 - anchoring (binding)
 - forgetting
 - consolidation and reliability treatment
- provide a framework to support memory architecture specifically on different cognitive layers
 - reactive perception-action loops
 - long-term consolidation (exploiting interaction as well)
 - associative look-up of knowledge (e.g. information-oriented selection)

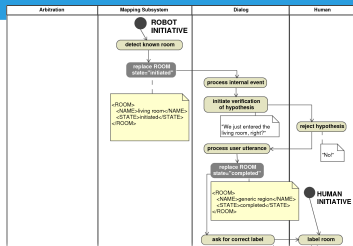
26

Mixed-initiative Learning of Room



27

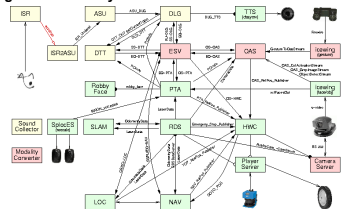
Room learning - Robot's Initiative



28

How to illustrate a system's architecture?

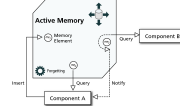
-Coupling before Memory Architecture



29

Learning by Interaction: space-based collaboration

- Beyond perception consolidation
 - interactive learning is supported by the concepts of memories
 - exchange of information between components drives the system
- exploit space-based interaction strategies on system level
- facilitate interactive negotiations of representation
 - picking up concepts of dialogic grounding [Clark] and alignment [Pickering & Garrod]
 - introduce a state in memory elements

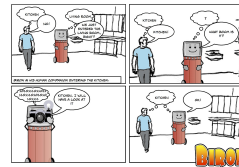


- An examples:
 - interactive human-augmented mapping

30

Mixed initiative using the IDI approach

- "negotiation" about aligning representations
- hypothesis generation



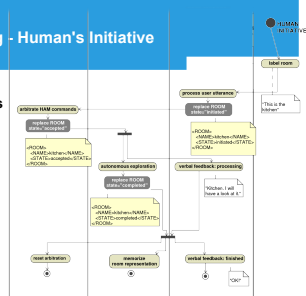
- independent of event source (informer)
 - user (via dialog)
 - system (via novelty detection)

- insertion of "memory element" drives the robot's learning
- it's basically hypothesis validation/clarification

31

Room learning - Human's Initiative

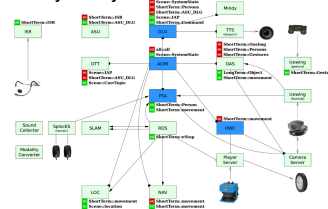
- error treatment by forgetting processes
- different forgetting schemes based on states



32

The problem of architecture drawings

-Decoupling achieved by Memory Architecture



33

Universität Bielefeld Technical Faculty – Applied Informatics

A more Functional View

- Final Software Architecture of the COGNIRON project of BIRON
- "Active Memory" being introduced in existing architecture

Universität Bielefeld Technical Faculty – Applied Informatics

A revised system architecture

- memory and non-memory interaction patterns
- Memory layers
 - Long-Term (LT)
 - Scene(=episodic) (SC)
 - Short-Term (ST)
- Arbitration based on memory contents
 - ACMI: rule-production system
 - HWA: grant access through HAL

Universität Bielefeld Technical Faculty – Applied Informatics

Technological Background

- DI: Information-driven Integration Framework (historically XCF)
 - <http://xcf.sf.net>
- Basic features:
 - Event-driven middleware for distributed processing
 - facilitates generic memories and processes
 - Unified information encoding (XML+Attachments)
 - Unified access and modification (XPath, XQuery)
 - Persistence (Oracle (Sleepycat) DBXML)
- Drawback: No explicit real-time support
- Bindings: C++, Java, Matlab, Python (partially)
- Environment: Primarily Linux, Win32 (partially)
- Main dependencies: Xerces, XQilla, Spread, Ice (partially)

Universität Bielefeld Technical Faculty – Applied Informatics

Comparison with CAST, let's discuss

Active Memory	CAST
conceptually one memory space (with content-based filtering)	several working memories
binding by dedicated generic memory spaces	explicit architectural binding support
very generic representation and access (XML + attachments)	defined data types (slice definitions)
only few managed components with specific interface → easy legacy integration	sub-architecture interfaces with clearly defined processing strategies
...	...


Universität Bielefeld Technical Faculty – Applied Informatics

Wrap-Up

- Memory-centered architectures
 - are picking up cognitive foundation
 - can serve as a generic principle to the construction of cognitive (robotic) systems
- Open challenges
 - balance between
 - close coupling for learning
 - loose coupling for software engineering

Universität Bielefeld Technical Faculty – Applied Informatics


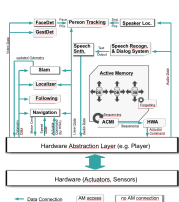


Memory & Memory Processes „brains in machines and developers“



Marc Hanheide
CogX Spring School
KTH, Stockholm, March 2009

Universität Bielefeld Technical Faculty – Applied Informatics


Why System Interaction Analysis (SiNA)?

Universität Bielefeld Technical Faculty – Applied Informatics

Why SiNA?

- complex systems
- evaluation of single components does not take into account interaction conditions
- evaluation profits from real interaction situations
- evaluation has to take into account system level and interaction level at the same time
- evaluation has to be interdisciplinary



Universität Bielefeld Technical Faculty – Applied Informatics

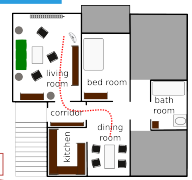
What is SiNA?

- What is SiNA?
 - method to analyze the data acquired in user studies in an integrated manner
 - data-driven
 - based on traditional interaction analysis
 - applies task analysis methods
 - applicable different scenarios and robots
- allows to answer the questions:
 - What do users do?
 - What does the robot do?
 - What happens within the system?

Universität Bielefeld Technical Faculty – Applied Informatics

The evaluation story

- script
 1. first contact
 2. teaching a room
 3. teaching an object
 4. guiding the robot
 5. teaching a second room
 6. teaching a second object
 7. good bye




tasks: greeting, location learning, object learning, following, farewell

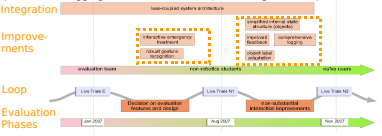
home tour

Universität Bielefeld Technical Faculty – Applied Informatics

Evaluation-implementation-cycle



- Systemic Evaluation
 - does our (implementation of the) interaction model suffice?
 - early involvement of users in task-oriented testing
 - evaluation-driven (re-)implementation: iterative process
 - important: „debugging“ on interaction level (not on segfault level)




Universität Bielefeld Technical Faculty – Applied Informatics

Interdisciplinary Systemic Interaction Analysis

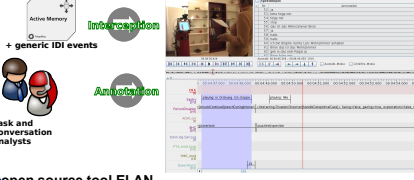
it's not only about robots... but humans as well!

- motivation:
 - bringing together conversational analysis, technical evaluation, and task analysis
 - interdisciplinary analysis with one corpus
 - structured analysis of human and robot behavior
- major questions:
 - what does the human do? (task analysis)
 - and why? (conversation analysis)
 - what does the robot do? (task analysis)
 - and why? (system analysis based on IDI introspection)



Universität Bielefeld Technical Faculty – Applied Informatics

Interdisciplinary Systemic Interaction Analysis



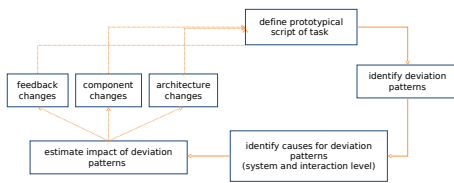
- open source tool ELAN
- unified view on system and interaction process

Technological Background

- **exploit event bus to intercept whole system dynamics**
 - event content is XML with binary attachments
 - XSLT or proprietary transformation of messages into ELAN format
 - record all/selected events to persistent storage (file/memory server)
- **exploit log4cxx/log4j**
 - towards richer and specific event (customizable)
 - logging integrated in EDI framework → logging events on bus

9

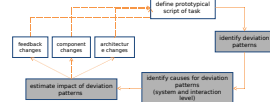
SInA Cycle



11

How does SInA work?

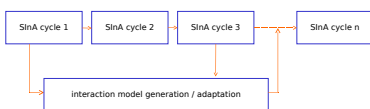
- **Step 2: Deviation patterns**
 - cases that deviate from the prototypical script are grouped
 - patterns not coincidences
 - patterns are similar in what the users do, what happens within the robot, and what the robot does
 - patterns are sorted by robot function (e.g., speech recognition, person perception, navigation)



12

How does SInA work?

- **Iterative process**



14

Exemplary ELAN session



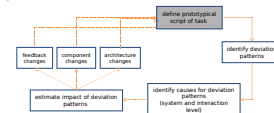
- **manual annotations:**
 - speech, gestures
- **interception logs:**
 - speech phrases
 - gestures
 - speech output
 - arbitration commands
 - screen character
 - self localization
 - tracked persons
 - dialog states
 - ...

10

How does SInA work?

- **Step 1: Prototypical interaction script**

- based on interdisciplinary video analysis (data driven)
- includes real-world restrictions
- includes subtasks, appropriate verbal commands and other behaviors

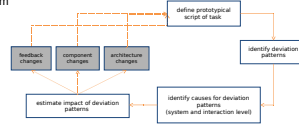


12

How does SInA work?

- **Step 3: System changes**

- scheduled based on the estimated impact
- kind of change depends on the source of the deviation
- short- and long-term



14

What are the advantages of SInA?

- **closed loop design (evaluation and implementation)**
- **learning from real-world problems**
- **scenario-, task-, and robot-independent**
- **comparability between systems and tasks**
- **allows for integration of other methods (e.g., Conversation Analysis)**
- **enables integrated and structured interdisciplinary analysis**

14

Let's look at some insights and examples

Do studies with "representative" samples of humankind



Complete Wrap-up

Many more videos

The obligatory last slide

The Graduate School Cognitive Interaction Technology at Bielefeld University, Germany offers

Research Grants for PhD Students and Postdoc Researchers

The Center of Excellence Cognitive Interaction Technology (CITEC) at Bielefeld University has been established in the framework of the Excellence Initiative as a research center for intelligent systems and cognitive interaction between humans and technical systems. CITEC's focus is directed towards motion intelligence, attentive systems, situated communication, and memory and learning. Research and development are directed towards understanding the processes and functional constituents of cognitive interaction, and establishing cognitive interfaces that facilitate the use of complex technical systems.

The Graduate School Cognitive Interaction Technology invites applications from outstanding young scientists, in the fields of robotics, computer science, biology, physics, sports sciences, linguistics or psychology, that are willing to contribute to the cross-disciplinary research agenda of CITEC. The international profile of CITEC fosters the exchange of researchers and students with related scientific institutions. For PhD students, a structured program including taught courses and time for individual research is offered. The integration and active participation in interdisciplinary research projects, which includes access to first class lab facilities, is facilitated by CITEC. For more information, please see: <http://www.cit-ec.org>

visit <http://www.cit-ec.org>
deadline: 22 March 2009