



DARPA introduction

Joost Westra



Introduction

- Darpa & Tekkotsu
- Project work
- Previous work
- Practical
- RoboLab



Why Darpa?

- Dutch ARchitecture Project for Aibos
- Problems with the old (German) code
- Creating your own code
- Preventing the same problems



Why Tekkotsu?

- Great support
- Great documentation
- Supplied with JAVA user interface
- Solutions for low level control
- Event driven
- Handling concurrency in good way
- Not too specific



Project work

- People work short time (high turnover)
- Long term project
- RoboCup & other research
- Short startup time



Requirements

- Strong modular approach
- Future research (extendibility)
- Easy to understand (clarity)



No waterfall approach

- requirements changes frequently on a large project
- implementation step hindered by previous steps
- All testing and debugging must be done at the end



Iterative approaches

- Rapid prototyping
 1. Identify the initial user requirements.
 2. Develop a prototype.
 3. Use and evaluate the prototype.
 4. Revise the prototype.
 5. Start again with point 3.



Iterative approaches

- Extreme programming
 - Short releases
 - Keep people happy
 - Every member contributor to the project
 - Extensive testing
- Pair programming
- Refactoring
- Milestones



Coding style

- Uniform
- Tab is 4 spaces wide
- Shorter and clear implementation

```
int updateOutputs() {  
    outputs = 1;  
}
```



Guard against multiple inclusions

```
// IncludeGuard.h  
#ifndef INCLUDEGUARD_H  
#define INCLUDEGUARD_H  
// Body of header file here...  
#endif // INCLUDEGUARD_H
```



Naming

- Java naming conventions

- class:

```
class FrenchVanilla : public IceCream {
```

- object identifier:

```
FrenchVanilla myIceCreamCone(3);
```

- Function:

```
void eatIceCreamCone();
```



Documentation

- Inline code documentation
- Review reports
 - Small
 - Choices made during implementation
 - deviated from specifications
 - Other remarks
 - In the code
- Theoretical reports
 - worthy for publication



Testing!!

- Make tests before programming
 - Think about it
- Test at least daily
- Do not commit untested code!!
 - At least make it compiling!
- Test code on the AIBO!!!



Versioning

- Different branches for each team
- Always at least one stable version
- Maybe different branches for tasks
 - Don't need maze solver for soccer
 - Even better if module could be used and is not hindering



Version Control

- Code is stored on a server
- All changes are stored
 - Go back to old revision
- Checkout
- Update
- Commit
- Conflict



SVN over CVS

- Directory versioning
- True version history
 - Renaming and copying are supported
- User friendly
 - TortoiseSVN in windows



Editor

- Eclipse
- Use your own favorite editor
- Do not litter the SVN with project files or binary files
- Compiling is done in command line, no special configuration settings needed



What is in Tekkotsu?

- Low level solutions
- Motion
 - Walking
 - Looking at
- Gui interface
 - JAVA
- Vision
 - Color segmentation
 - Region detectoin



New Tekkotsu Version (3.0)

- Some improvements
 - Line detection
 - Color tool
- Not always needed
- From scratch -> new version



What is added?

- Documentation
 - Vision
 - Generator
 - Worldstate
- Calibration tool



Vision explained

- More practical approach
- Recommended by Tekkotsu site
- How to extend for object recognition
- Processing on: raw/segmented/jpeg
- Vision pipeline uses lazy evaluation



Old Color Calibration Tool

- Default tool very limited
- Mistake -> start all over
- No reloading off image classification
- Uses only U and V channels
- Colors change if angle changes
- Overlap in different colors



Color Calibration Tool

- Ported from software project
- Uses multiple images
- Uses Y,U,V Channels
- Image are classified using polygons
- Mistake -> delete mistake
- Reloading image classification
- Possible to add more images
- Automated overlap handling
- Still room for improvement



Detection Generator

- Behavior can listen to CMVisionObject
- Better to create a filter
- Filter listens to CMVisionObject
- Behavior can subscribe to filter
- GoalDetectionGenerator



Detection Generator

- Prevent duplication of code (modular)
- Easy to make new behavior
- Step-by-step guide



World State

- Shared memory region
- Direct access: `state->yourVariableName`
- Event throwing
 - Behaviors reacting to changed world state
- Step-by-step guide
 - Creating and throwing new events
 - Extending the world state
 - Listening to events
- Demo behavior



World state

- No dynamic memory allocation
 - No vectors ect.
 - Can be a problem (3APL for example)
- No pointers
 - If no dynamic memory, not a big problem
- Only accessible from default threads
 - Communication possible true main thread



What more is added?

- Auto color calibration
- Tic-Tac-Toe
 - Internal representation
- Graphical behavior tool
- Communication
- Reasoner
 - 3apl style -> 4apl
- Maze solver
- Flag detection

Behavior Maker

The screenshot displays the BehaviorMaker v1.0 software interface. The central workspace shows a state machine diagram with three states: a green start state labeled "Walk(x|50,ty|0,(a)|0,(n)|0)", a white state labeled "Sound((Sound|BarkMed.wav)", and a red state labeled "Turn(x|0,ty|0,(a)|-0.5,(n)|0)". Transitions are as follows: "Walk" to "Sound" on "VisualTargetClose((Visual|PinkBall))"; "Sound" to "Turn" on "LostTarget((Visual|PinkBall),(D...))"; "Turn" to "Walk" on "SeeVisualTarget((Visual|PinkBall))" and "LostTarget((Visual|PinkBall),(Delay|1))".

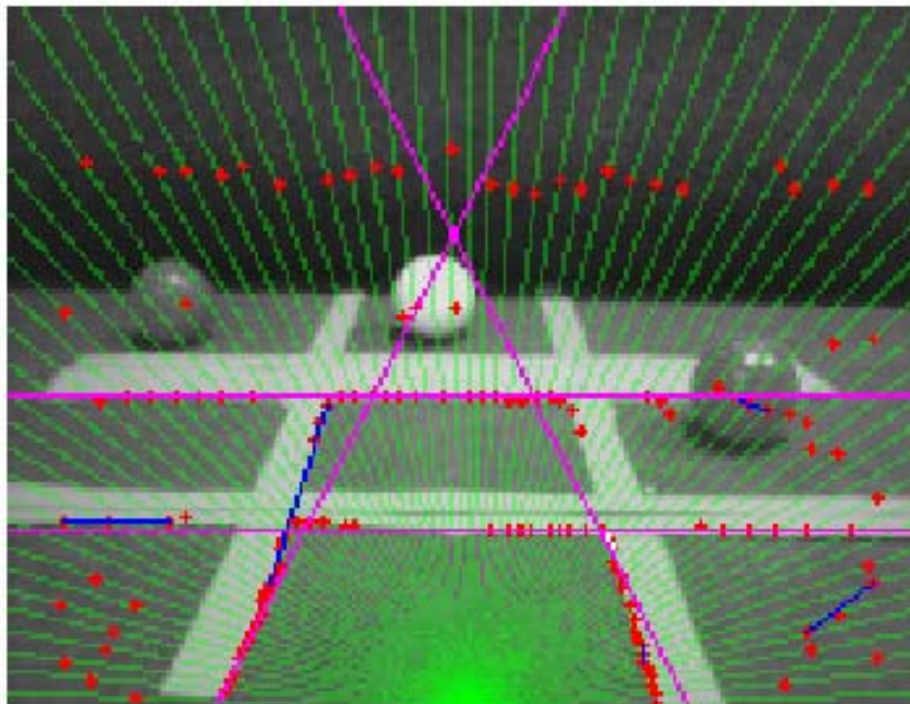
On the right side, there are three panels:

- Behaviors:** A list of behaviors including Sound, Walk, Turn right, Turn left, Strafe right, Strafe left, Walk to target, and Wait. "Turn right" is currently selected.
- Parameters:** Two input fields with "Set" buttons. The first contains "0" and the second contains "dummy".
- Conditions:** A list of conditions including Completion, Event, Visual Target Close, See Visual Target, Time Out, and Lost Target. "See Visual Target" is currently selected.

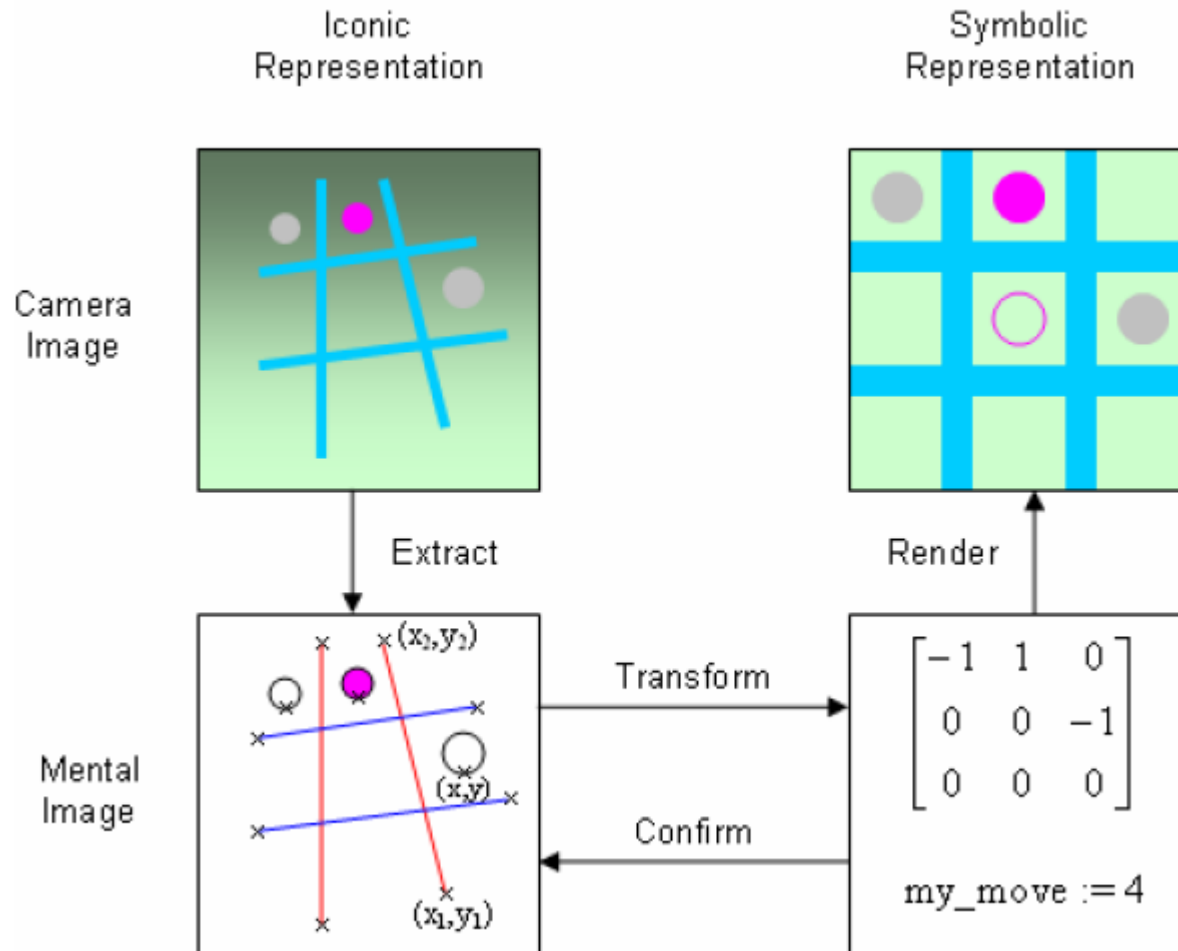
At the bottom of the interface, there are several control buttons: "run Machine!", "make Meta.behavior", "make Stop behavior", "Load FSM", "make Start behavior", "remove Stop behavior", "Save FSM", "reset all", "add Condition", and "remove Condition".

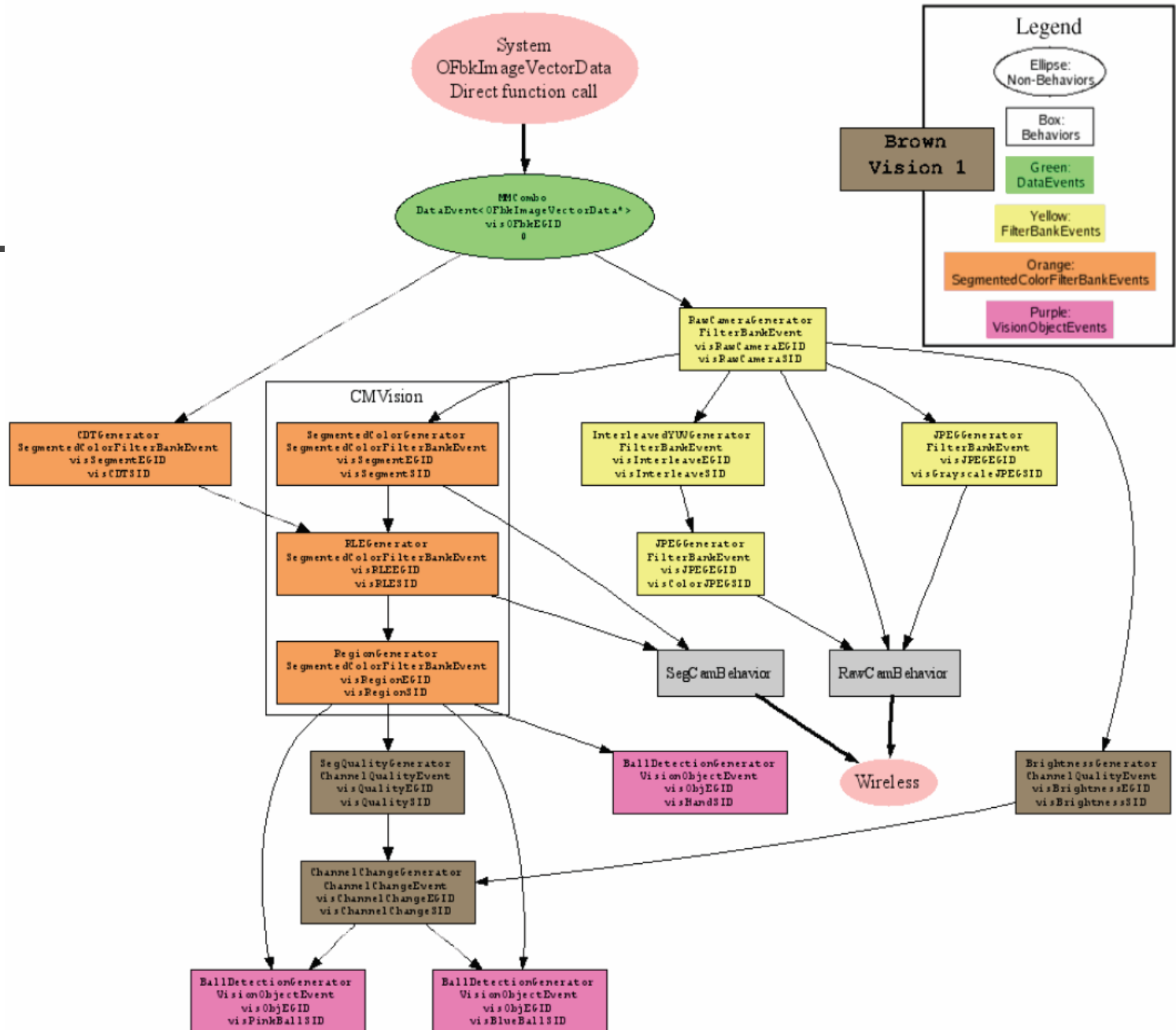
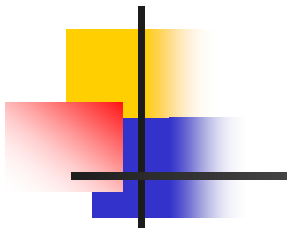
At the bottom left, it says "Condition created from 0 to 2". At the bottom right, it says "connected to AIBO".

Tic-Tac-Toe



Tic-Tac-Toe







Practical

- Read documentation!
- Use matching ip-adress
- Use own SVN account
- Robot safety
- Erase Memory stick, do not Format!
- Recommend using windows with cygwin
 - Problems with memory card writers
- Installation Script



Cygwin

- Cd /cygdrive/c/
- Ls
- Tab completion
- Install dir in SVN



Basic commands and files

- Project\make newstick
- Project\make update
- ftpupdate
- Tools\mon\ ./ControllerGUI [ip-adress]
- \project\ms
 \open-r\system\conf\wlandflt.txt
- C:\cygwin\CygwinTeamNaam.bat



Svn

- <https://svn.cs.uu.nl:12443/repoman/info/darpa2>
- <https://svn.cs.uu.nl:12443/repos/darpa2/>
- 2007 prefix
- Create account
 - Send account name:
joostwestra@gmail.com



Standard problems

- Vision related problems
- Communication problems
 - Use right ip!
- Compile problems
 - Version control



Course Format

- Likely to change
- Small number of students



Short Demo

- Configuring cygwin
- Compiling
- Opening GUI

The logo for RoboLab features a stylized graphic on the left consisting of overlapping colored squares (yellow, red, blue) and a black crosshair. To the right of this graphic, the word "RoboLab" is written in a blue, sans-serif font.

RoboLab

- Motion capture lab.
 - Be careful-> expensive equipment
 - Needs to be dark
 - Some "acting"
- Robolab
 - Need light
 - Maybe new location



Questions?



To the lab!

CGN-C104



Extra

- Telnet connection
 - Used for debugging
 - telnet ip-adress 59000
- Docs in SVN/docs/AIBO2006docs/