Report of Humbodt University Berlin (Team Name: Humboldt Heroes)

Prof. H. D. Burkhard, Dr. M. Ritzschke, Dr. F. Winkler,
Dr. M. Werner, A.Georgi, U. Dueffert, H. Myritz

14th September 1999

Contents

1 Overview **2**

2 Strategy **2**

3 Algorithm **3**

3.1 Walking, and Posture control 3
3.2 Vision (Color recognition and how to make color table) 4
3.2.1 Generating of CDTs with a colour simulation tool 4
3.3 Localization 9
3.4 General behaviors 9

4 Agent Architecture (or how the architecture is constituted of objects.) **12**

5 Appendix **12**

1 Overview

The team members include students as well as members of the teaching stuff from the Institute of Informatics at the Humboldt University.
They represent the groups of Artificial Intelligence, Responsive Computing, and Signal Processing, respectively.
It was the aim of the project to combine the skills of these disciplines to program football playing legged robots.

An underlying idea was to use the experiences from the simulation league for the general structure of the robot software. We still think that this concept is realistic. But the restricted time forced us to use a very simple reactive approach for the RoboCup 1999.

Our general research interests can be described as follows:

We are using methods from Distributed AI for knowledge processing and control, based on mental models of deliberation. We are interested in the development of skills on higher level decision protocols using methods from Machine Learning, especially from Case Based Reasoning.
We are specifically interested in developing normal consensus protocols, collision avoidance protocols and would like to develop new models of faults, e.g., the opposing soccer team would be considered as a new type of a fault.

We are interested in novel algorithms for image processing and their implementation in embedded systems. We would like to apply parallel computing structures for image processing using the pixel-bit parallelism principles of distributed arithmetic. Scalable resolution allows simultaneous suppression of noise, sharpening of discontinuities and labelling of important data.

2 Strategy

We have experience in the design of agent architectures for the simulation league, where we have used belief-desire-intention approaches. There we have developed a couple of basic skills (e.g. for kicking, dribbling, ball interception). The choice of desires and intentions is based on utility calculations, which lead to individual plans on the base of the available skills. Cooperation is performed using the known behaviors of team mates.
We already have some experience with robots where robots need to cooperate to execute a specific task. Our approach is based on CORE (COnsensus for

REsponsiveness) middleware which was developed by our group for a cluster of workstations. In this project we will add an extra level of difficulty by modeling hostile faults. We also plan to use sophisticated search techniques such as Tabu Search and Simulated Annealing to develop a strategy and decide on particular moves.

Furthermore, we have some experience in hardware/software co-design and implementing low level signal processing procedures for recognition tasks both in hardware and in software.

We got some experience in implementation of effective control methods from experiments with the minirobot system KHEPERA. So we could make comparisons between decision- rules-controller and fuzzy-logic-controller.

3 Algorithm

We have distinguished four main parts which we call Cortex, Brain, Body, and Communication. Messages are passed between these modules according to the underlying control structure.

3.1 Walking, and Posture control

The general idea is to transmit the plans computed by the Brain to the Body and performed it by the available skills. The Body controls the movement of the legs in order to turn, move, kick etc. Additionally, there exists a direct information flow between Cortex and Body for immediate actions, e.g., for keeping track of the ball. This imposes some rudimentary layered architecture.
Realization: We did not develop our own skills. This was a real drawback since the usage of the available skills (LE2 module) caused several problems. The main disadvantage was a missed possibility to interrupt a movement in process. Another disadvantage was a lack of movements' precision.
We tried to overcome these disadvantages by several means:

- To interrupt a movement, we insert an interception function between LE2 module and the robot module. Its task was to intercept outgoing commands and to report success to the LE2 module.

- To improve the real-time behavior, we used a priority queue to transmit commands to the body. Sent but not yet executed command that became superfluous are throw away.

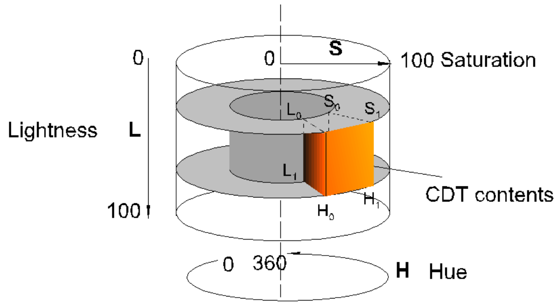


Figure 1: CDT-contents in the HLS space

- Critical parts of a movement are executed in the step-wise mode. That shall increase the movement's accuracy.

However, the creation of a real-time walking and posture control is a mayor objective to become an appropriate competitor in the next competition.

3.2 Vision (Color recognition and how to make color table)

The Cortex uses the Color Detection Engine to identify the objects in the image by common procedures of image-processing to find the object parameters, e.g. position, width, center-point. The control of head motions is subject of the Cortex.

3.2.1 Generating of CDTs with a colour simulation tool

It is well known, there are some possibilities to describe the colour in a picture. Common colour spaces are for instance RGB, YUV (the PAL/European standard for colour television broadcasting) and HLS(Fig.1), where H is for the Hue (the H value is a degree value through colour families), L for Lightness (1 = white, 0 = black) and S fore Saturation (that is the degree of strength of a colour - greater is S, the purest is the colour).

The robot-eye use the YUV space and so we can analyze the YUV-values in a robot-image. But we want to create CDTs for a wide range of lightness/darkness and of saturation, because the robot sees different colours in the pictures if he looks from different viewpoint to the same objects. On the other side we found in our testing-period, that every robot had from the same viewpoint under the same light conditions little different YUV-values.

Therefore we realize a way to develop our CDT's with four steps:

Step 1:

Shooting session to get some images from all relevant objects (ball, goals, player-dress, landmarks), we use different viewpoints and - if enough time, all our robots.

Step 2:(Fig.2)
Analyzing of the YUV values and transformation via RGB in the HLS space. Our tool allows us to use the mouse for moving a reticule over the object. With the help of the statistical componente we get for H,L,S the mean value and the standard deviation.

Step 3:
Now we use a second tool (Fig.3) to simulate possible combinations of HLS around the mean values of H,L,S - with the help of random numbers of the constant distribution like radio noise. Every generated HLS-point (we use ¿ 1000 points) will be transformated in the YUV space and the resulting borderlines of YU and YV plane built our CDTs, which we can save in a file(Fig.4).

Step 4:
Checking: We check the quality of our CDTs with a further program using originally robot-images.
This tool allows - if it is necessary - manually corrections of the CDTs.

C:\Aperios\bilder\neu_pic\Hon1y.pgm

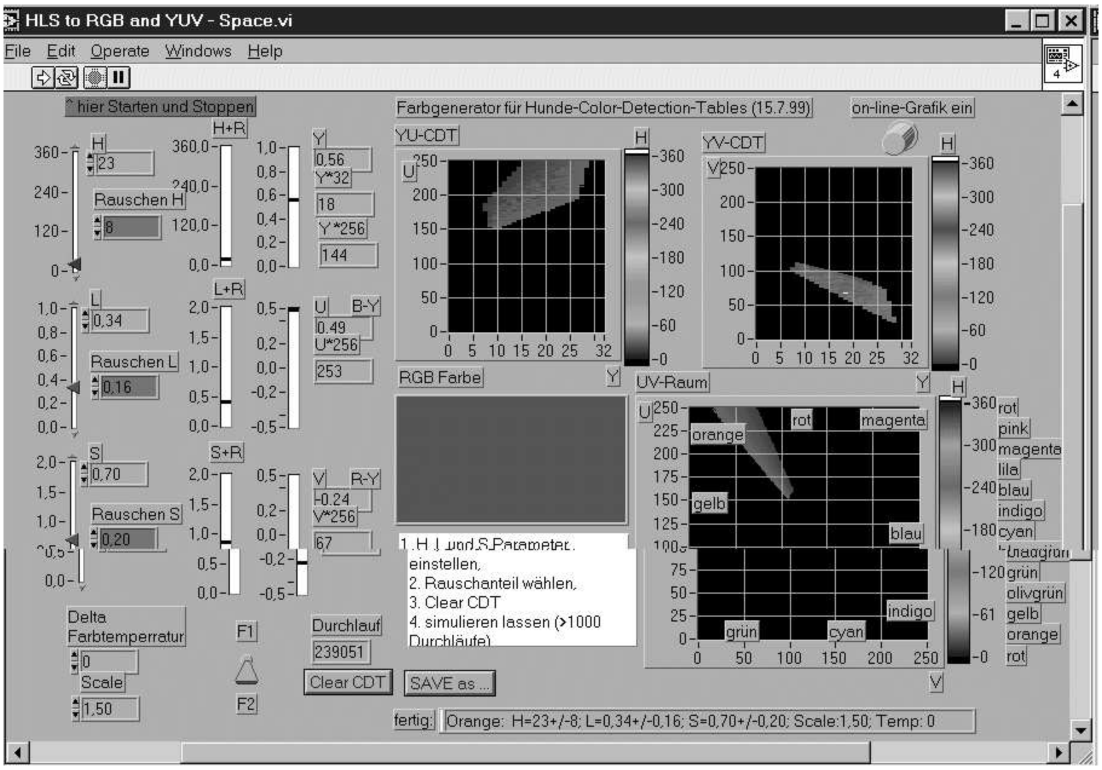
Y 137
U 205
V 76
Y copy 137
U copy 77
V copy -52

Scale 1,30

Cursor 0 36,15 18,26 137,00

R 224,78
G 112,75
B 31,54
H 25,21
L 0.39
S 0,75

H mean 24,28
H 2*sigma 1,99
L mean 0,39
L 2*sigma 0,10
S mean 0,73
S 2*sigma 0,10



H 23

Rauschen H 8

L 0,34

Rauschen L 0,16

S 0,70

Rauschen S 0,20

Delta Farbtemperratur 0

Scale 1,50

Y 0,56

Y*32 18

Y*256 144

U B-Y 0,49

U*256 253

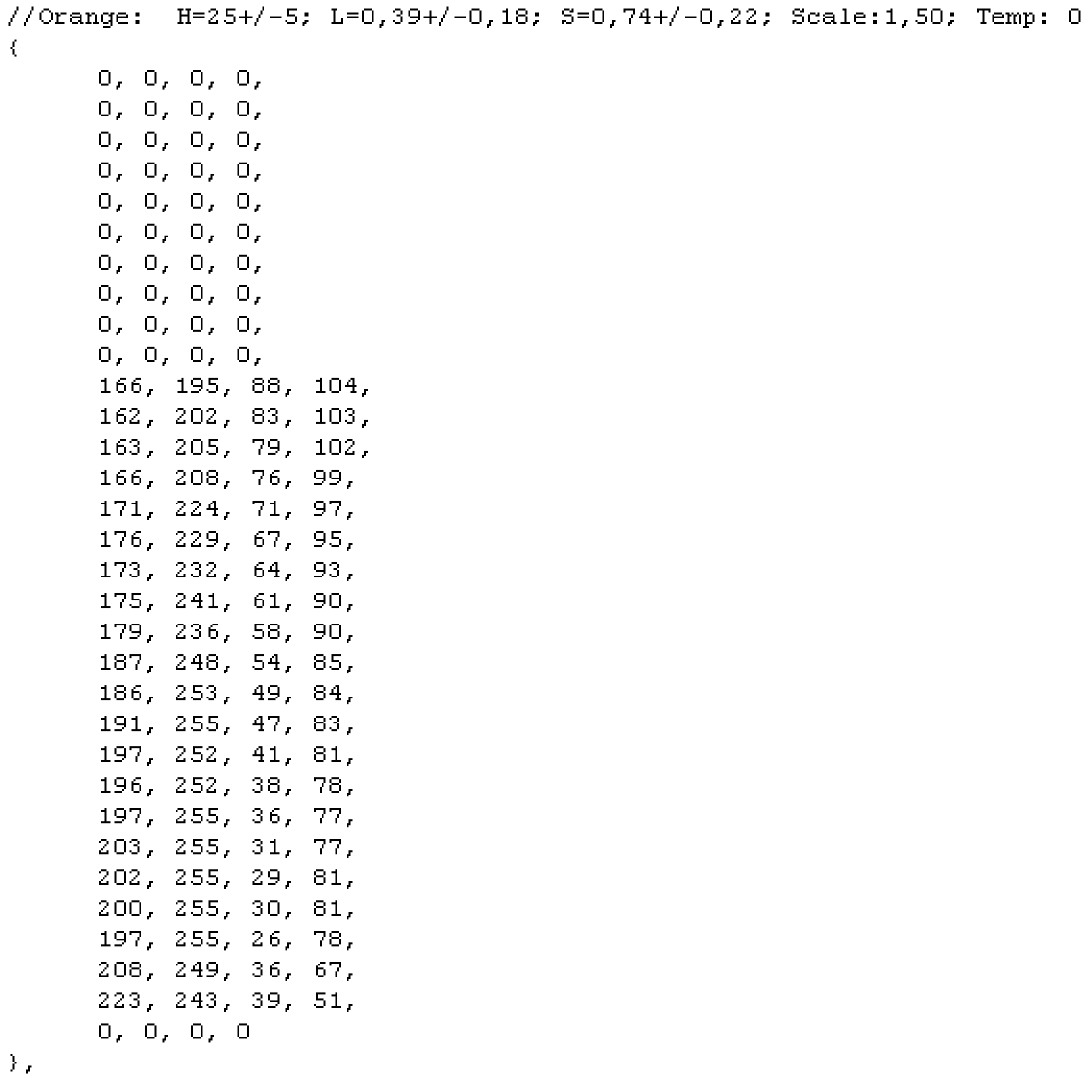
V R-Y -0.24

V*256 67

1. H, L und S Parameter einstellen,
2. Rauschanteil wählen,
3. Clear CDT
4. simulieren lassen (>1000 Durchläufe)

Durchlauf 239051

fertig: Orange: H=23+/-8; L=0,34+/-0,16; S=0,70+/-0,20; Scale:1,50; Temp: 0



```
//Orange:  H=25+/-5; L=0,39+/-0,18; S=0,74+/-0,22; Scale:1,50; Temp: 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, 0, 0, 0,
      0, 0, 0, 0,
      0, 0, 0, 0,
      166, 195, 88, 104,
      162, 202, 83, 103,
      163, 205, 79, 102,
      166, 208, 76, 99,
      171, 224, 71, 97,
      176, 229, 67, 95,
      173, 232, 64, 93,
      175, 241, 61, 90,
      179, 236, 58, 90,
      187, 248, 54, 85,
      186, 253, 49, 84,
      191, 255, 47, 83,
      197, 252, 41, 81,
      196, 252, 38, 78,
      197, 255, 36, 77,
      203, 255, 31, 77,
      202, 255, 29, 81,
      200, 255, 30, 81,
      197, 255, 26, 78,
      208, 249, 36, 67,
      223, 243, 39, 51,
      0, 0, 0, 0
},
```

Figure 4: The generated CDT

3.3 Localization

The robot had to stop each time he tries to localize himself on the field. Then he turns his head in a predefined way to a 180 degree angle, all seen flags and goals are stored.
The dog can now calculate his own position and body direction with these informations.
For more details please look at appendix.

The calculations are most of the times correct, only if the number of seen flags is very low or the distance to a flag or a goal is quite wrong, than the calculation could went wrong very hard.
A hard time of seconds must passed until the next localization should be executed.

3.4 General behaviors

Idea: The software architecture of the Brain is oriented on mental modelling of agents (BDI), which is used by AT Humboldt in the virtual RoboCup. It transforms the received data into an internal world representation ("belief"). It identifies possible options ("desire") and commits for useful plans ("intention").

Actually, we did not finish the work on this concept for RoboCup. Instead, we used a very simple reactive approach:

Look for the ball

Run to ball

Search and position for opponent goal

Go with the ball to opponent goal

Kick if you are near the opponent goal

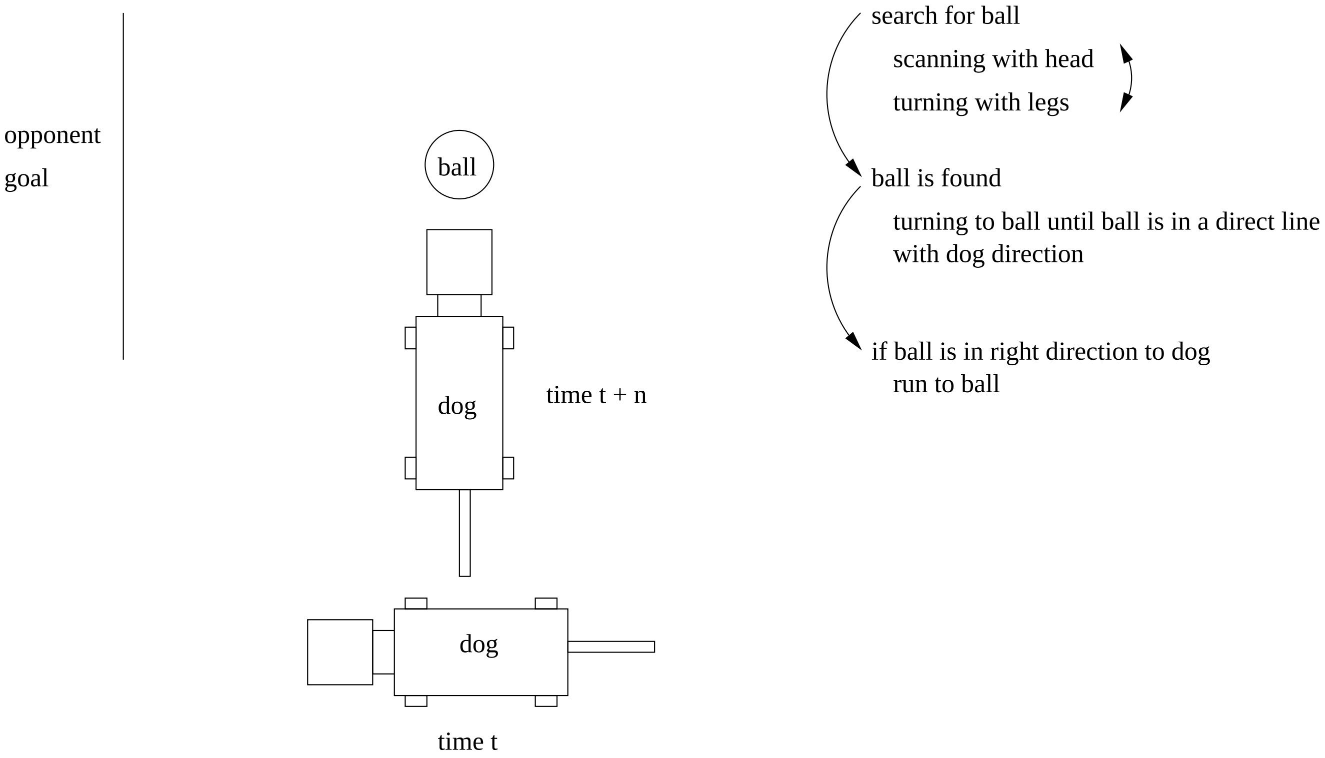


Figure 5: Search for Ball and turn to ball

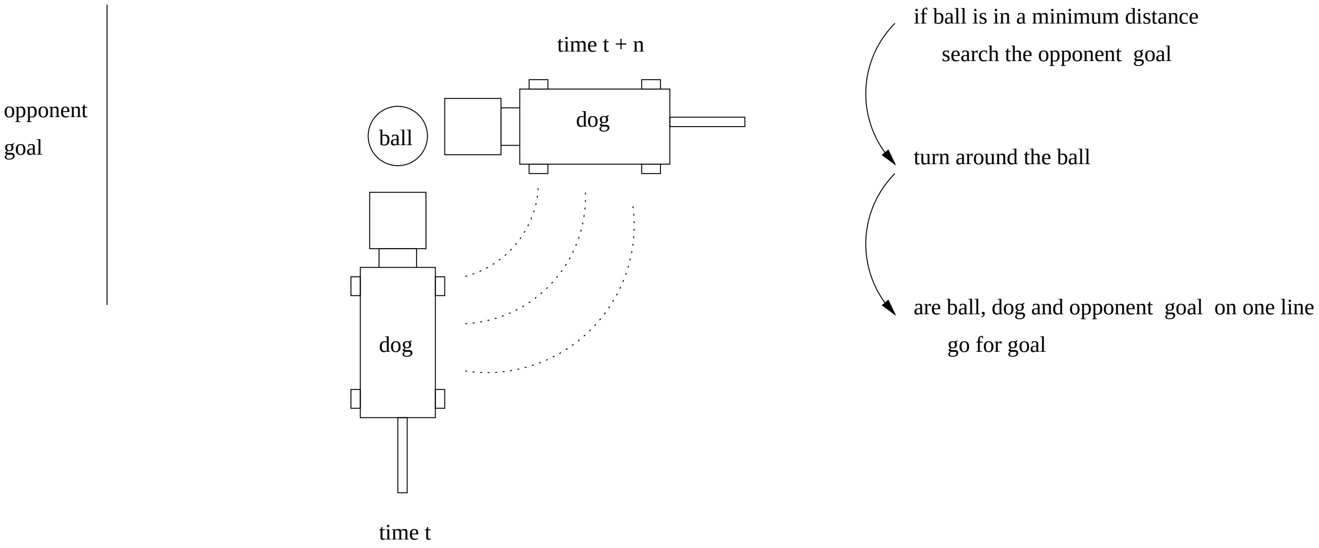


Figure 6: Go to ball and turn around ball

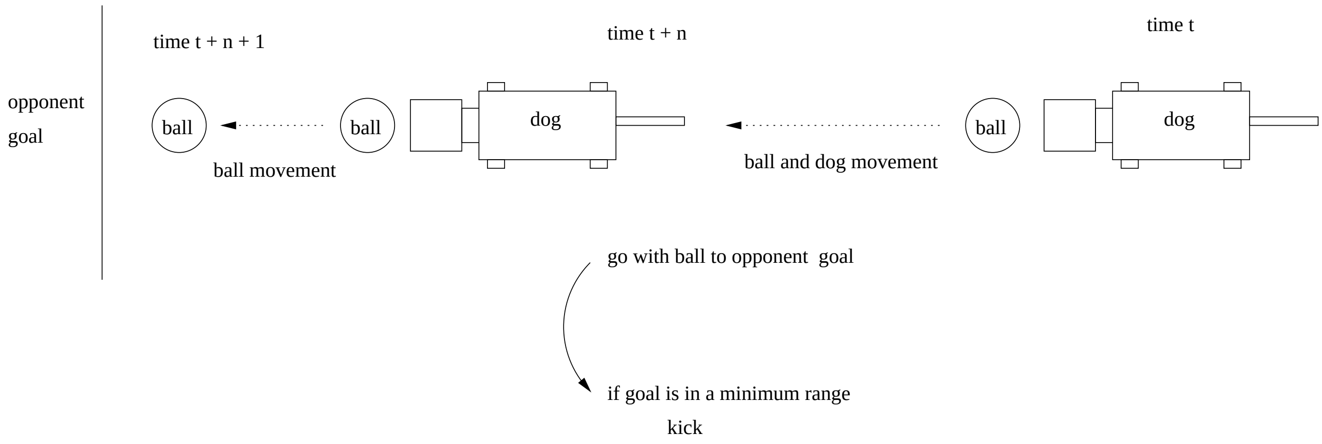


Figure 7: Go with ball and kick

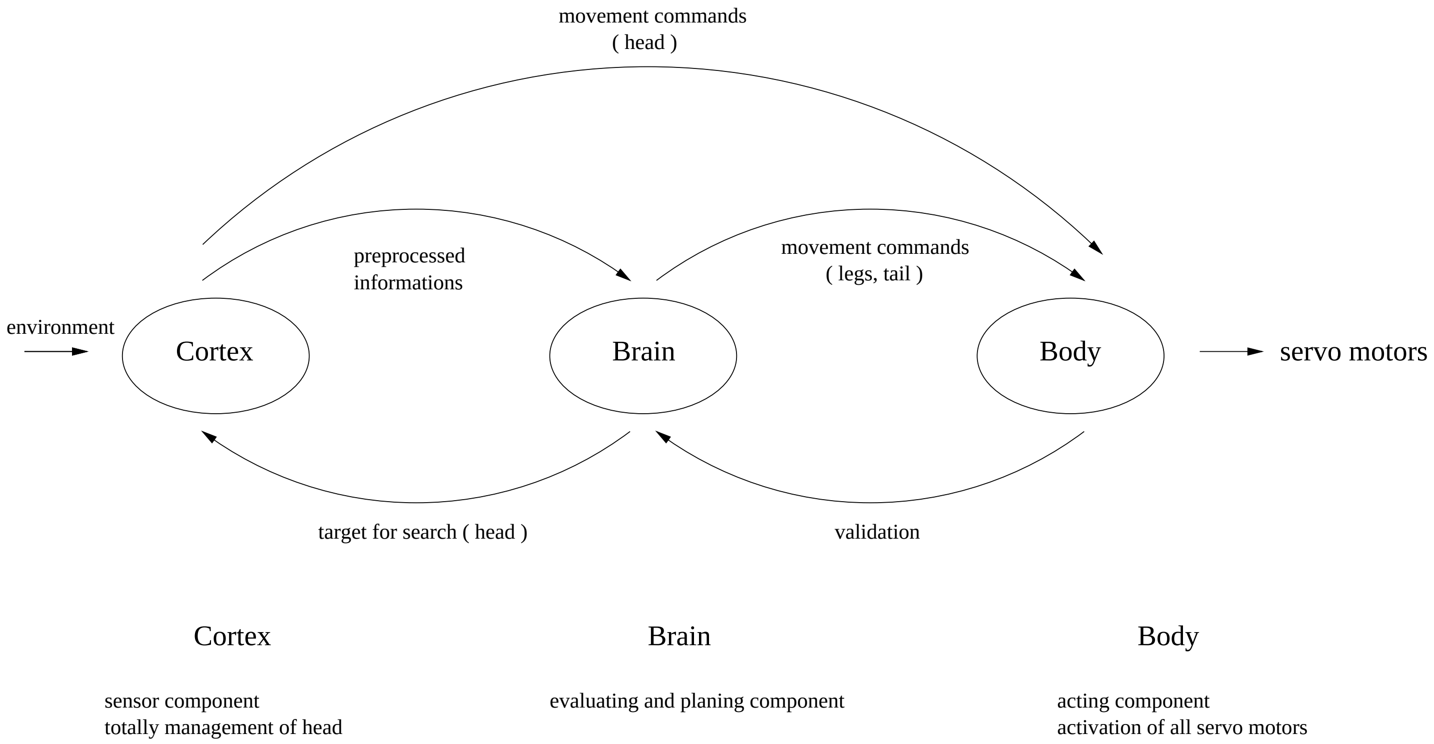


Figure 8: Agent architecture (without comunication, because it was not supported)

4 Agent Architecture (or how the architecture is constituted of objects.)

We have distinguished four main parts which we call Cortex,
Brain, Body, and Communication (Fig.8). Messages are passed between these modules according to the underlying control structure.

5 Appendix

algorithm for computation of own bodydirection (parts):

```
double Field::calculateOwnBodyDir()
// Shortcuts: theFirstBestFlag: f1, theSecondBestFlag: f2
SeenObject* f1 = theFirstBestFlag;
SeenObject* f2 = theSecondBestFlag;
```

```
// Shortcuts: seen angle of f1: beta, seen angle of f2: alpha
double beta = f1->dir();
double alpha = f2->dir();

// Shortcut: seen angle between f1 and f2: gamma
double gamma = normalizeAngle( beta - alpha );

if( gamma < 0 ) // skip flags f1 and f2
f1 = theSecondBestFlag;
f2 = theFirstBestFlag;
beta = f1->dir();
alpha = f2->dir();
gamma = -gamma;

Vector vectorFromf1Tof2 =
theFlagPositions[ f2->id() ] - theFlagPositions[ f1->id() ];
// Shortcut: length between the flags f1 and f2: c
double c = vectorFromf1Tof2.length();

double sinEpsilon = ( f2->dist() / c ) * sin( gamma );
double sinDelta = ( f1->dist() / c ) * sin( gamma );

// calculating bodyDir by two different ways
double firstBodyDir = normalizeAngle( PI - ( epsilon + beta ) );
double secondBodyDir = normalizeAngle( delta - alpha );

// calculating average value of both body directions
double bodyDir = ( firstBodyDir + secondBodyDir ) / 2;
if( absolute( bodyDir - firstBodyDir ) ¿ PI/2 )
bodyDir = normalizeAngle( bodyDir + PI );


// turning bodyDir to the angle of the line between the flags f1 and f2
bodyDir = normalizeAngle( bodyDir + vectorFromf1Tof2.angle() );

return bodyDir;
```