Eukalyptus

(Koala Environment Simulator)

Design: Network communication

Jörg Conradt, Institute for Neuroinformatics, Zurich (INI / ETH)
Thabo Beeler, University of Applied Sciences, Rapperswil (HSR)

Revision History

Version	When	What	Who
0.1	12.04.2004	Initial version	Thabo Beeler
0.2	14.04.2004	Revised and approved	Jörg Conradt
0.3	24.04.2004	Added Model	Thabo Beeler
0.4	01.05.2004	Removed setRobotAngle M	Thabo Beeler
1.0	27.05.2004	Completely revised	Thabo Beeler
1.1	12.06.2004	Added the getPTAngles command Added the getBatteryVoltage command	Thabo Beeler
1.2	23.06.2004	Second revision	Jörg Conradt Thabo Beeler

Table of contents

Network structure

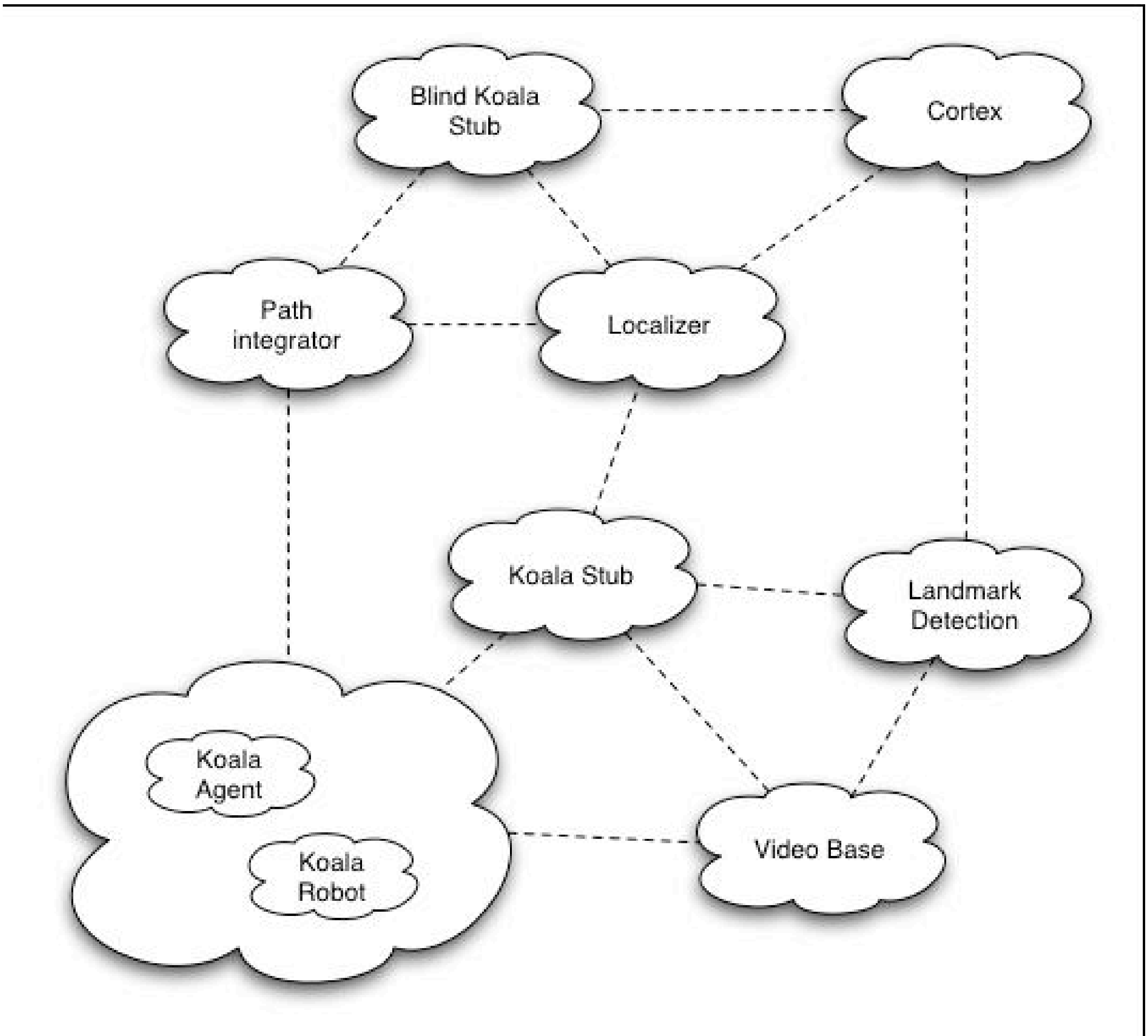

Figure 1: Network topology of the Koala/Eukalyptus project

It is essential that the Koala agent and the Koala robot are completely exchangeable.
To observe the Koala robot we need a Koala stub, which represents a local view on a remote Koala object (agent or robot).
To observe the quality of the path integrators and the cortexes, we need a blind Koala stub, which represents the estimated location. It is called blind, because it does not receive any sensor input.

Processes

From the preceding network structure we can distinguish three kinds of processes, which need to communicate:

- Sensors
- Cortex components
- Stubs (Remote control)

Sensors

Name	Type	Default Port	Communication partner
DriveControl	Actor Sensor (Active)	+10 +11	*From*: External Input, Motor Cortex *To*: PathIntegrator
Gyroscope	Sensor (Passive)	+31	*From*: Sensory Cortex *To*: Sensory Cortex
Whiskers	Sensor (Passive)	+32	*From*: Sensory Cortex *To*: Sensory Cortex
Magnetic Compass	Sensor (Passive)	+30	*From*: Sensory Cortex *To*: Sensory Cortex
Cameras	Actor Sensor (Active)	+20 +21	*From*: External Input, Motor Cortex *To*: Sensory Cortex, Landmark detection

Cortex components

Name	Default Port	Communication partner
Path integrator	+400	*From:* DriveControl *To:* Sensory Cortex
Localizer	+100	*From:* Path integrator, Gyroscope, Magnetic compass *To:* Sensory Cortex, DriveControl
Landmark detection	+300	*From:* Video base *To:* Sensory Cortex, Cameras
Video base	+200 - 207	*From:* Cameras *To:* Cameras, Landmark detection
Sensory Cortex	None	*From:* PathIntegrator, Gyroscope, Whiskers, Magnetic Compass, Cameras *To:* PathIntegrator, Gyroscope, Whiskers, Magnetic Compass
Motor Cortex	None	*From:* - *To:* DriveControl, Cameras

Stubs (Remote control)

These serve the purpose to observe remotely active sensors. So we can display the state of the koala robot or koala agent on a different computer. All the sensors mentioned above have stubs. While most stubs connect directly to their sensor, the DriveControlStub uses a PathIntegrator, which can run on a different computer.

Actor

Actors receive commands and perform these.

Active Sensors

Actively send data to the subscribed receivers. If these sensors need also to receive commands, they open two ports.

Passive Sensors

May be accessed remotely to query their state.

External Input

Input generated by external devices like Joysticks. For the robot, the input device should be transparent, so that it does not need to distinguish between Motor Cortex and External Input.

Motor Cortex

This is the part of the artificial intelligence of the robot that generates action commands. It is therefore connected to the actors.

Sensory Cortex

Requests and receives sensory inputs. It is therefore connected to the sensors (passive and active).

Path integrator

Receives the speed change commands from the DriveControl and integrates the location changment over time.

Localizer

The localizer merges the estimated location from the path integrator with additional sensor input from the gyroscope and the magnetic compass to provide a better location estimation.

Video base

The video base does some pre-processing on the perceived images from the two cameras.

Landmark detection

This process tries to detect landmark in the perceived images.

The Protocol

The whole communication is based on TCP/IP.
The syntax of the protocol is simple, plain text and straight forward. They are case insensitive.
The commands are **set**, **get**, or **reports**.

Syntax:

<Command> <Parameters>\n

Base Commands

Command	Parameters	Description	In/Out
OK		Acknowledge that a command has been received.	Out
Error	Command <string>	Error message. A command has not been accepted.	Out
Error	Format <string>	Error message. The format of the command was not understood.	Out

Koala

Command	Parameters	Description	In/Out
GetBatteryVoltage		Request the current battery voltage.	In
BatteryVoltage	V=<float>	Reports the current battery voltage in Volt.	Out

DriveControl

Command	Parameters	Description	In/Out
SetSpeed	L=<float>, R=<float>	Sets the speed of the wheels. In m/s	In
Speed	L=<float>, R=<float>, T=<double>	Reports the speed of the wheels. In m/s Sends a timestamp for synchronization purposes. The format is <sec>.<msec>	Out

Gyroscope

Command	Parameters	Description	In/Out
GetRobotAngle	G	Query the angle.	In
SetRobotAngle	G=<int>	Sets the angle. In degrees.	In
RobotAngle	G=<int>	Reports the angle. In degrees	Out

Magnetic compass

Command	Parameters	Description	In/Out
GetRobotAngle	M	Query the angle.	In
RobotAngle	M=<int>	Reports the angle. In degrees.	Out

Whiskers

Command	Parameters	Description	In/Out
GetWhiskerData		Query the whiskers.	In
WhiskerData	<int>,...,<int>	Reports the whisker data. The parameters are an array of the sensed distances.	Out

Cameras

Command	Parameters	Description	In/Out
SetPTAngles	PL=<int>, PR=<int>, TL=<int>, TR=<int>	Sets the pan/tilt angles of the visual system. In degrees. P stands for pan and T for tilt. L refers to the left camera, R to the right.	In
SetPTAngles	P=<int>,T=<int>	Sets the pan/tilt angles of the visual system. In degrees. P stands for pan and T for tilt. Sets both cameras to the same values.	In
SetPTAngles	DPL=<int>, DPR=<int>, DTL=<int>, DTR=<int>	Sets the pan/tilt angles of the visual system relative to the current position. In degrees.	In
SetPTAngles	DP=<int>,DT=<int>	Sets the pan/tilt angles of the visual system relative to the current position. In degrees. Move both cameras by the same delta.	In
GetPTAngles		Requests the current pan/tilt angles	In
PTAngles	PL=<int>, PR=<int>, TL=<int>, TR=<int>	Reports the current pan/tilt angles	Out

Landmark detection

Command	Parameters	Description	In/Out
Landmark	ID=<int>, angle=<int>, dist=<float>	Reports a detected landmark. ID is the identifier of the landmark, angle is the perceived angle (in degrees) and dist is the assumed distance in meters.	Out

Path integrator

Command	Parameters	Description	In/Out
GetLocation	PI	Request the currently estimated location.	In
SetLocation	PI x=<float>, y=<float>, orient=<int>, t=<double>	Set the location to these values. x = x-coordinate (in meters) y = y-coordinate (in meters) orient = orientation (in degrees) t = timestamp (in <sec>.<msec>)	In
Location	PI x=<float>, y=<float>, orient=<int>, t=<double>	Report the location to these values. See SetLocation PI for a description of the parameters.	Out

Localizer

Command	Parameters	Description	In/Out
GetLocation	LZ	Request the currently estimated location.	In
SetLocation	LZ x=<float>, y=<float>, orient=<int>, t=<double>	Set the location to these values. x = x-coordinate (in meters) y = y-coordinate (in meters) orient = orientation (in degrees) t = timestamp (in <sec>.<msec>)	In
Location	LZ x=<float>, y=<float>, orient=<int>, t=<double>	Report the location to these values. See SetLocation PI for a description of the parameters.	Out

Models

Figure 2 represents the inter-process communication messages, while figure 3 shows the stub/skeleton communication of the Eukalyptus environment. As one can see, we use exactly the same commands for both of them.
We display only the most important components in both figures.