31385 Autonomous Robot System

Laserscanner

(rev 1)

Objective

The objective of this exercise is to give an introduction to the laserscanner used as obstacle detector

When you have finished this exercise you will be able to

- Use the laser scanner server (ulmsserver) with a simulated scenario
- Monitor laser scanner data using a graphic client (qclient).
- Use the laser scanner server (ulmsserver) with the auzoneobst plugin for object detection.

Introduction to the laserscanner.

The laserscanner on the smrs is a Hokuyo with an angular resolution of 0.36 degrees, a scanfield of 240 degrees and a range of 4 m.

The laser is handled by the server 'ulmsserver' and a number of suitable plugins. In this exercise the plugin auzoneobst is used. It limits the scanfield to 180 degrees so that the smr does not see itself. The scanfield is split into 9 equal zones of 20 degrees each and the plugin returns the distance to the nearest obstacle in each zone i.e. it returns 9 distance values.

You need to a server set-up as shown on figure 1.

When simulating a laser scanner, an additional port is opened in the simulator for the laser scanner data.

The laser scanner server 'ulmsserver' connects to this port as shown on the figure.

The ulmsserver further need the robot position to make calculations on the laser scan. This information is provided by the MRC. When the MRC *is started* it will look for a laser scanner server.

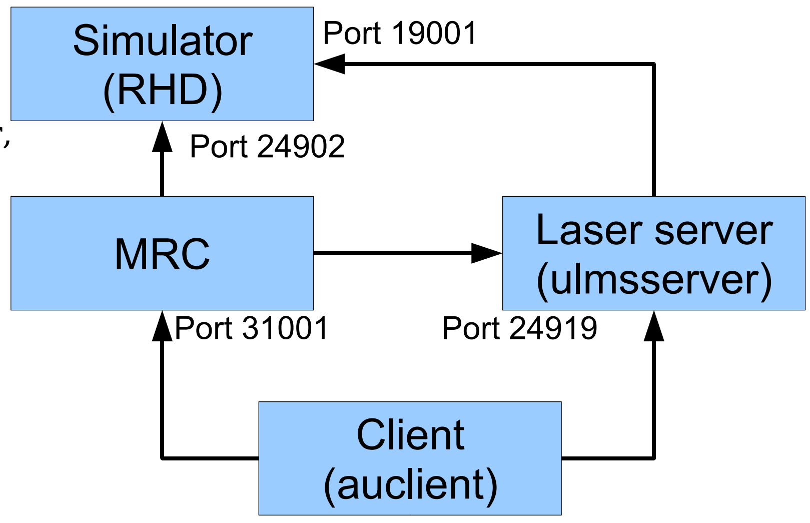

Figure 1: Server set-up for MRC and laser scanner simulation

Simulator

Start a terminal window and go to the sim subdirectory and start the simulator:

```
  simserver simconfig.385world.xml
```

You should now see the simulated environment.

Laser server

Start the laser scanner server (in the sim directory):

```
  ulmsserver
```

(The `ulmsserver.ini` is modified with the port numbers from figure 1)

write the command zoneobst and see that it returns 9 distances.

MRC

Start the MRC in (in the sim directory):

```
  mrc -s0 -t1
```

Monitoring client

To start some action a client is needed.

and start a client (in the sim directory):

```
  qclient
```

In the **client ,** make the robot go forward into the scene, e.g:

```
>> mrc1 fwd 1.6
>> mrc1 turn 70
```

The 'mrc1' tells the client that the commands are for the MRC, the rest of the line after the 'mrc1' is send to the MRC connection.

The robot should move, and the laser scan display should reflect the movement.

The x,y position of the robot should be updated at the bottom-left of the scan image (if not, then probably the MRC failed to connect to the ulmsserver. The ulmsserver needs to be started before the MRC).

Try some display options from the on-line help:

```
>> disp help
```

The '`disp`' command takes some options, e.g. '`scale`' and '`pos`', as shown in the help tekst. Try:

```
>> disp scale=3.0 pos=0.25
```

The pose history is shown as a red line. The green laser scan is actually shown for the last 5 scans received from the laser scanner (`disp scan=5`).

Try to change some of these display settings (e.g. `disp posehist=1000`).

The displayed image is in an image pool in the client, and can be saved to a disk fil. Try the command:

```
>> poollist
```

This command should show that there is an image called 97.

Save this image:

```
>> poolget img=97 savepng=foo.png
```

The saved image(s) can then be viewed by e.g.:

```
  eog foo.png
```

Running the Simulator with the Square Program.

The simulator may also be used with the square program. First start the simulator and the laserserver as above then the square program should connect to the simulator when started. The square program may be started from its own directory.

Simulator Configuration.

The simulation environment is mainly defined by the files:

385world.world and 385world.png.

The 385world.world decides which png-files that define the obstacles and the lines on the floor and the starting position of the smr:

```
# create a robot
smr
(
name “smr0”
pose [x y z th]
}
```

where pose decides the starting pose of the smr.

The blue lines in the 385world.png file decide the obstacles in the world and the black lines give the linepositions.

The png-files are created from the odg-files using libreoffice draw and may be modified. It is important not to change the outer frame as it decides the size of the world. The modified files should be exported as png-files with the resolution suggested by the program.

Zoneobst

Start the simulator from the sim directory

```
simserver simconfig.385world.xml
```

Start the laserserver from the sim directory:

```
ulmsserver
```

issue the `zoneobst` command in the laserserver. The laserserver vil return the 9 zone distance values in the display.

Put the smr looking towards a wall with a distance of 45 cm from the laser to the wall. Issue the cmd and note the values of zone 3, 4 and 5.

Put the smr looking parallel to the wall with the wall to the left and a distance of 45 cm from the laser to the wall. Issue the cmd and note the values of zone 0,1 and 3.

Put the smr looking parallel to the wall with the wall to the right and a distance of 45 cm from the laser to the wall. Issue the cmd and note the values of zone 5,6 and 8.

Calculate the theoretical values and compare with the measured

Using zoneobst with MRC.

The zoneobst plugin may be used with mrc. The laserserver must be started befor mrc is started. The first command in the smrcl script should be:

laser “scanpush cmd='zoneobst' “

this will make the laserserver run the zoneobst cmd at every scan an return the values in the variables $l0 to $l8.

Make a smrcl-script (documentation in smrcl.pdf) that starts the zoneobst cmd (as above) and logs the return variables ($l1 to $l8)
Look at at logfile 'log'
The smrcl commands 'log' and 'wait' may be useful.

Using zoneobst with square.

The zoneobst plugin may be used with the square program. The laserserver must be started before the square program is started.

The line

```
    len=sprintf(buf,”push t=0.2 cmd='mrcobst width=0.4'\n”);
```

should be changed to

```
 len=sprintf(buf,”scanpush cmd='zoneobst'\n”);
```

this will make the laserserver run the zoneobst cmd at every scan an return the values in the array laserpar

Make the changes described above to the square program and log the values in the laserpar array in a file.