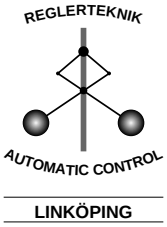
ENSYM Project Oriented Studies of spring 98 - team 1

Fredrik Tjärnström, Mattias Duppils, Patrik Haslum, David Byers, Gundars Kulups, Dan Lawesson

Department of Electrical Engineering
Linköping University, S-581 83 Linköping, Sweden
WWW: `http://www.control.isy.liu.se`
Email: `fredrikt@isy.liu.se`

January 15, 1999



Report no.: LiTH-ISY-R-2094
Submitted to (Nothing)

Technical reports from the Automatic Control group in Linköping are available by anonymous ftp at the address `ftp.control.isy.liu.se`. This report is contained in the compressed postscript file `2094.ps.Z`.

ENSYM Project Oriented Studies of spring 98 - team 1

Fredrik Tjärnström, Mattias Duppils, Patrik Haslum,
David Byers, Gundars Kulups, Dan Lawesson

February 15, 1999

Abstract

The report is description of the ENSYM Project Oriented Studies (POS) of spring 1998. The project goal was to control a toy car around a not beforehand given track as fast as possible.

1 Introduction

The goal of the ENSYM Project Oriented Studies (POS) of spring 1998 was to produce control software to maneuver a radio controlled car along a race track as fast as possible, while avoiding obstacles and competitor cars. The autonomous cars were to compete in a racing event, with a track layout that was unknown before the race.

1.1 Available equipment

The equipment available for the project consisted of two (toy) RC cars with their transmitters connected to a Pentium 75 computer, a camera to detect the cars and 486 PC computer dedicated to image processing. The project goal was to create a system that could run a car autonomously around an arbitrary race track. The area covered by the camera was limited to a square with a side length of about 3.7 m, which set the limits of the racing arena. See Figure 1.

To make the cars visible for the camera, they had been equipped with a LED. Images from the camera are processed to find bright spots which are interpreted as objects. The position of the spots are given as coordinates on a 256×256 grid. This gives a resolution of about 1.45 cm, to be compared with a length of 21.8 cm and a width of 13 cm for the cars, see Figure 2.

Each team connects via TCP/IP to the Pentium 75 computer, which acts as a server. It delivers coordinates and executes commands to steer the car (left, right, forward, backward and percent of maximum motor power). Commands are pulse width modulated (PWM) by the power control box which interfaces the server to the transmitters (see Figure 3).

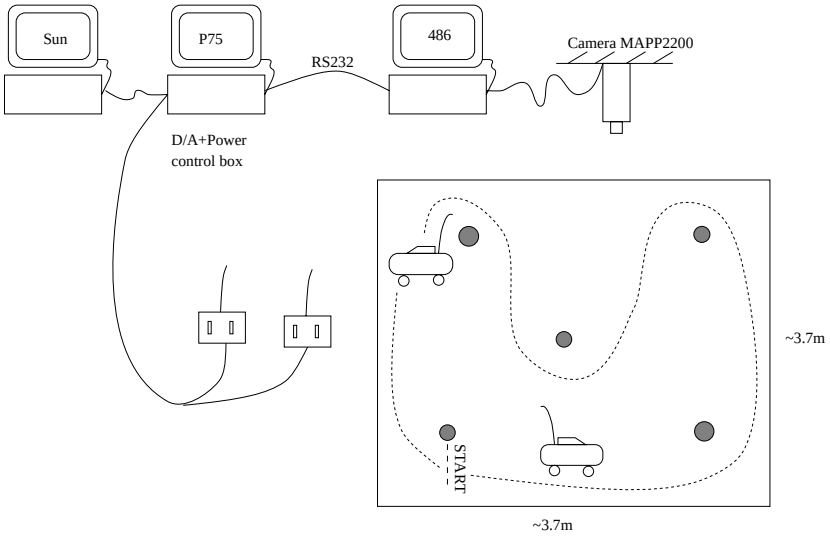


Figure 1: Equipment available for the car race

2 What we did and how we did it

This chapter contains a description of the system and some notes on how it was developed. The implemented system is presented as well as some ideas that did not work out.

Figure 3 is a diagram of the system divided into logical components.

Power Control Box

The power control box is a device which translates the power level signal provided by the computer to a power signal to the car. The car is directly controlled by a forward / backward / off switch on a transmitter. The power control box uses pulse width modulation (PWM) in order to alternate the power to the car. Ideally, a high pulse frequency could be low-pass filtered by the car engine, but for some unknown reason frequencies higher than 11Hz are not usable in practice.

Camera

The camera is positioned in the ceiling above the racing track, and connected to a PC via a standard serial cable. The actual camera system MAPP2200 is developed and manufactured by Integrated Vision Products (IVP), a spin-off company from Linköping University. It has one bit resolution only, but the image processing can identify the center in a cluster of bright points. The result is a list of the coordinates of all bright clusters plus a time stamp, and is sent from the server in the format $[x_1 y_1 \ldots x_n y_n t]$. The snap shot frequency is about

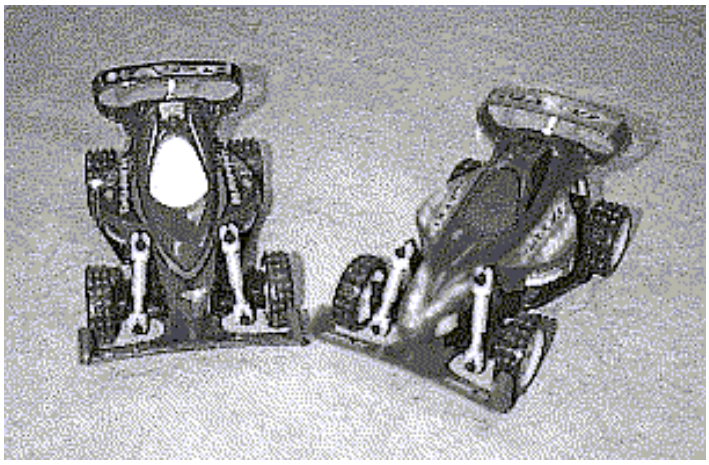


Figure 2: Picture of the toy cars

10Hz, which is sufficient for a vehicle driving at low speed. However, the camera system turned out to be very sensitive to daylight from the windows, reflected on the floor.

Car

There are two radio controlled cars of brand Asahi, model Havoc Fastlane, colored red and blue. The cars have gear boxes which can be set high or low, though only low gear was used. As described in the section about the power control box, the car control signal is pulse width modulated with a rather low frequency. This makes the movement of the car non-smooth, because the wheels slip every time a new pulse arrive. The extent of this slipping depends on the speed, at higher speed there is less slipping since the wheels are already turning fast when a pulse arrive. High frequency PWM would be a solution to this problem but that would require modifications to the hardware.

Radio link

The radio links, one for each car, send control signals to the cars. The transmitters employ amplitude modulation (AM) with carrier frequencies 27MHz and 40MHz, respectively. The 27MHz radio link was disturbed by the environment, which caused the corresponding car not to follow given commands exactly.

Vehicle identifier

The purpose of the vehicle identifier is to identify the coordinate in the coordinate list that corresponds to the controlled car, and constitutes an interface between the image data sent by the server and the vehicle observer. The identi-

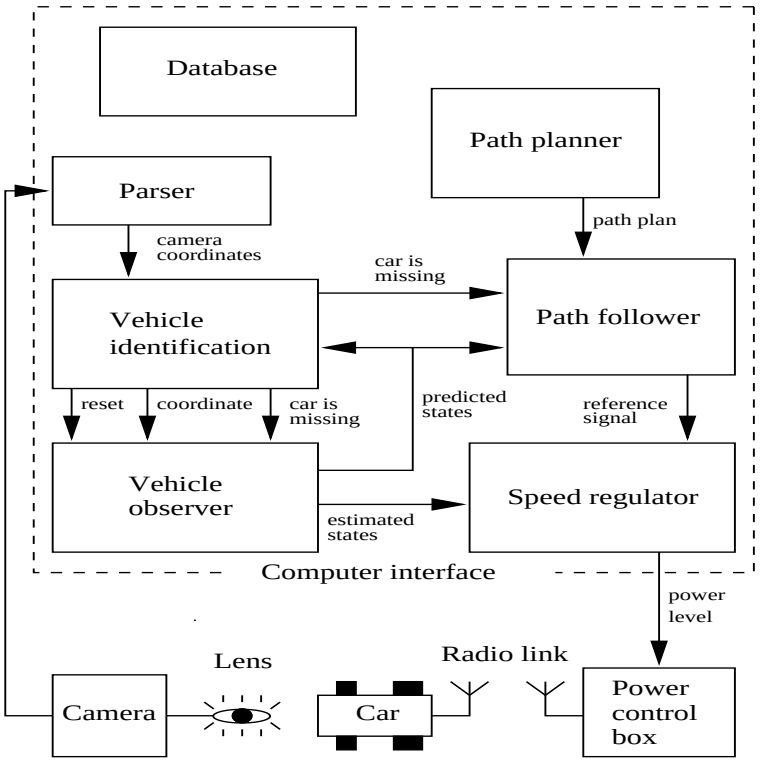


Figure 3: The components of the system

fier can be seen as a camera filter, removing unrealistic or multiple coordinates from the image data.

The identifier algorithm, which can be seen at Figure 5 in appendix, uses memory in order to increase the detection probability. It maintains a car locked state (locked) and keeps track of the number of consecutive samples where no car has been found (counter). When the car status turns from locked to unlocked, a reset signal (do_reset) is sent to the vehicle observer.

Vehicle observer

The vehicle observer estimates a lot of physical quantities associated with the motion of the car. Position, velocity and acceleration are all estimated, using a time-varying Kalman filter based on a random walk model for the car. This type of model does not very accurately describe the actual dynamics of the system, but in most cases it still produces good estimates of the current system state. It has therefore been used to estimate e.g. the motion of airplanes and boats from measurements of position.

Other physical quantities like turning radius, angle and angle velocity are

estimated from the position and the velocity. We had some problems in estimating the angle of the car when running at low speed. The reason for this is that since the estimates are filtered, we also get some time delay in them, so when the car was actually stopped we could still get estimates that were non-zero. This made the estimated angle change, even when it should not. To get rid of this problem we added update rules, so that the angle is not updated when the speed is too low. This works quite well, but introduces a problem when the car moves slowly for a long period of time since then the angle is never updated event though it should be. This causes problems for the path follower which needs good estimates of the cars angle and position. Because of this, it should certainly be possible to improve control of the car by a little more effort spent on making good estimates.

Speed regulator

The speed regulator tries to adjust the control (power) signal to the car so that it keeps a (by the path follower) desired absolute speed. For the first version, we used a PID-regulator with anti-windup. This resulted in a very slow response to large changes in the reference signal, in particular when the direction was reversed. We then modified the regulator to get a more linear relation between the control signal sent and the absolute speed of the car, and with this approach we got a much faster response in actual car speed. Since the linearization was made in an ad hoc manner it should be possible to improve it using measurements on the car.

Path follower

The path follower is the component that takes a plan and a state of the world as input and provides a desired behavior as output. This component is on top of the control hierarchy at runtime; it is the part of the system that decides where to go and how to get there.

Path planner

The path planner is the component that provides a path plan to the path follower. In our first attempt at path planning, paths were curves, represented as (rather dense) sequences of points. The idea was to state the problem as an initial curve representing the track and refine it through small iterative modifications, to a curve that the car could follow. In particular, turns needed to be smoothed out, as the turning capabilities of the car are limited. To each point of the final curve, the maximum speed at which it could be followed at that point was computed, based on measurements of the cars turning radius at different speeds.

This idea did not work, mainly for two reasons: First, representing paths as curves does not allow paths in which the car reverses direction of travel; it can only drive forwards. With the limitations on space and maneuverability, this

meant that most instances of the planning problem in this representation were not solvable. Second, even in the few cases where it was possible to generate a path, the car could not follow it. The representation of curves as sequences of points allow for very little deviation in following the path, which due to the (relatively long) feedback delay and the imprecise steering was not possible to achieve.

An alternative approach would be to construct a graph of connected, convex, obstacle free regions, and search this for a "good" path, using some heuristic measure on the relative difficulty of maneuvering through a region. This was not tried due to lack of time.

Database

Database system facilities have been integrated into our system. Race data is recorded in real-time into the database. Recorded data could be used in the future for such purposes as recovery algorithm improvement and keeping track of our car in multi-car race. However, only the recording part has been implemented so far. In every cycle, coordinates of the car are recorded together with acceleration vector, predicted turning radius and some coefficients from the vehicle observer and speed regulator. Afterwards data can be retrieved for given coordinates and acceleration vector. All records where these parameters are in given range are retrieved from the database. The average of the rest of the parameters for those records is calculated and returned to the control application. The database server is implemented on an NT workstation and is accessed through TCP/IP connection. Underlying the server is the main memory database system AMOS. It allows very fast data access for small to medium data sets. To improve search speed some assumptions have been made. The idea is that search is done on a range of values and average of data in the resulting data set is returned. Therefore, instead of storing exact values of search parameters, rounded values are stored. This allows using hash table index and to avoid expensive range searches.

2.1 The path follower

Maneuvering the car along the track is divided in two parts: Planning a path and following it. The later is the responsibility of the path follower. As part of path following a limited form of replanning is used, but the overall path planning has to be done manually.

2.1.1 The definition of a path

A path is represented as a sequence of waypoints, each waypoint consisting of a pair of coordinates and a rectangular area. That a waypoint is not simply a point is because it is in general not possible to hit a point exactly, due to the imprecision in the camera and in the rest of the physical system. Therefore, the waypoint is considered to be reached when the car has come within the

acceptance area. The purpose of coordinates is to have something that is easy to aim for. Following a path thus comes down to steering towards the current waypoint and changing to the next as soon as the car enters the acceptance area associated with the waypoint.

2.1.2 Preplanning

In the implemented system, preplanning has to be done manually. Before the race, a path in the form of a sequence of waypoints with associated acceptance areas is entered, using a graphical tool. To make the system work correctly the creator of the plan has to be aware of many aspects of the path following algorithm, including obstacle avoidance and replanning.

2.1.3 Path following

Path following is the task of making the car follow the path defined in the preplanning stage. As described above, this amounts to heading towards the current waypoint until the corresponding acceptance area is reached, and then switch to the next point in the path. Since both camera and car have their imperfections, the path follower must also be capable of dealing with deviations from the plan.

The path follower operates somewhat like a hybrid automaton, switching between different modes of control. Steering is affected directly by the path tracker, while drive commands are mediated by the speed regulator. An overview of the control states and switches is shown in Figure 4.

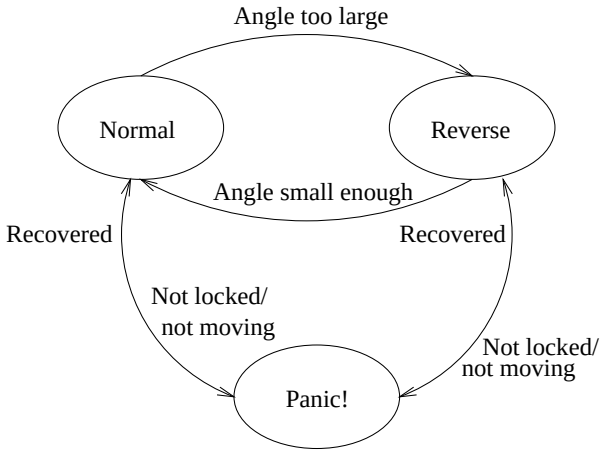


Figure 4: Control states of the path follower

In **normal** mode, the controller drives forward at "march speed", steering towards the current waypoint. Should the angle between the cars current heading and the direction of the waypoint become to great, the car is not likely to manage the turn without running into a wall or obstacle; in this case the

controller switches to reverse mode. In reverse mode, the controller sets the desired speed to negative march speed, and steers "inversely" towards the current waypoint. When the angle towards the waypoint becomes small enough, the controller switches back to normal mode.

The exception to this simple mode switching is the **panic** mode. Panic mode is entered if the controller has set a non-zero speed reference signal but the car has not moved for a number of samples (this often indicates the car has run into an unseen obstacle) or if more than a fixed number of samples pass without the identifier locking onto the car (which is usually the result of the car being lost in a corner of the world). Whatever the reason for entering the mode, the controller in panic mode tries to relocate the car by making a complete halt, then driving at high speed for a fixed time, first in one direction and if that does not work in the other.

Because steering is not very precise, if the path plan is too constrained it may still happen that the car finds itself in a position where the current waypoint is not directly reachable; there is an obstacle in the way. This is handled by a limited form of runtime replanning. The planner places a temporary waypoint on the most convenient side of the obstacle. Replanning is done periodically, as part of the process of switching waypoints. This navigation and planning can be seen as a separate process, running concurrently with the controller.

2.2 Integration

Integration of the path follower with the speed regulator and the vehicle observer was timeconsuming. The interface between the different parts is simple: When the vehicle observer has done its computation on a sample from the camera, it calls the path follower which issues new commands in reaction to the new information. The commands, steering and a speed reference signal, are passed to the speed regulator part which computes the proper power level for the power control box.

Simple as the interface is, it was still nontrivial to make the different components work together as one unit. Changing the reference signal too quickly makes the speed regulator oscillate, but moving to slowly introduces a lot of noise - due to quantization errors in the camera - that makes the data from the vehicle observer less accurate. The location is still correct, but estimates of velocity and direction become unusable.

3 Results

We have divided the results and conclusions into three sections. One covering aspects of racing, i.e. the actual project work, and one discussing environmental issues like hardware and course administration. The last section gives a summary of our experiences of the course.

3.1 Racing

The task of the project was to make a radio controlled car follow a given track. The system we built managed to do this quite well, at least with only one car on the track, but with more time it could be surely improved on many points. We designed the system to be able to deal with competing cars as well but because of the problems with the 27 MHz radio link, the cars could not race against eachother at the same time, so the performance our system in a multi-car race was not really tested.

3.1.1 Things that worked

There were a lot of practical problems with the system that had to be overcome. In our opinion we managed to reach reasonable results with the following difficulties.

We achieved very good estimates of all of the interesting physical quantities, e.g., position, velocity and turning radius, at medium to high speed. We had more problems getting accurate estimates at low speed, which is due to the fact that the grid the camera uses is too sparse and that the radio signal to the motor in the car operates at too low a frequency. The sparseness in the grid gives a high quantization error in the measurements of the position of the car. Because of the low frequency in the radio signal to the car, the wheels skid and a lot of power from the motor is lost due to friction. This gives the car a jerky movement, and a lot of errors are induced from this non-smooth behavior. These errors will directly influence the other estimates, since they are based on the position measurements.

In the high to medium speed range we got good results in trying to regulate the speed to constant level. There was a big improvement in the regulator when we tried a simple linearization of the control signal. The modified control signal gave a much faster response on the actual speed than the none-modified one.

The guidance algorithm in its final version worked very well, surprisingly so considering the uncertainty of sensor information and the imprecision of control. With a reasonably welldesigned plan it managed even to complete the "five point M" track used in the final race. By adding temporary subgoals and performing a limited form of replanning, it is capable of recovering from situations where an obstacle has come in between the car and its next goal, although it will not always choose the "smartest" route.

Sometimes avoiding walls and obstacles can become contradictory, as staying away from a wall might force the car too close to an obstacle. To make the correct judgments in such narrow situations, some trimming is necessary with the current implementation. Two different impulses - to go far away from the wall and the obstacle, respectively - have to be weighted to result in a correct path. After trimming the algorithm worked correctly on all tracks. No track was followed correctly on the first attempt, making the need for trimming evident.

An important contributing factor to the robustness of the guidance algorithm is most likely the fact that avoiding collision with walls or obstacles has a high

priority. Following a collision, the car behaves in a way that is difficult to model and predict, which reduces the quality of the estimates drastically, and leads to problematic situations which may be very difficult to recover from. Driving slowly but safely simply pays off better than driving fast and recklessly.

We also think that our group mostly did a good job of keeping the work running and fixing things in time. There was a lot of work during the last days before the race, but we think that this is a quite a common problem in group projects.

3.1.2 Things that can be improved

At the moment the predictions of the physical quantities could be better, at least if one wants to look a couple of samples ahead. In the short time range there are also problems with the one step ahead prediction of quantities like turning radius and angle velocity.

In trying to separate our car from other objects, i.e. other light sources on the track, we tried to just continue driving and keeping track of our own car by making good predictions. This was harder than we thought, so we need to find a better solution to this if we want to race against other cars.

As indicated above there will be problems when one tries to keep the car running at a low constant speed due to low frequency signaling from the radio transmitter. This low-speed problem is more or less impossible to solve with the equipment available.

The systems planning capabilities are currently limited to choosing a route around the nearest obstacle in the way of the car. The overall route plan must be entered manually. The generation of this plan should be automatic, from information about the location of obstacles and the layout of the track.

Such a plan could consist of a connected chain of regions. If the guidance algorithm is modified to maneuver using the boundaries of the current region as "walls", it should under normal circumstances be possible to avoid replanning all together. Since circumstances are not always normal, replanning capabilities are still a desirable feature. As the area in which the car moves is relatively small and the number of obstacles not too large, it may well be possible to make the planning algorithm efficient enough to be usable also for replanning, in real time.

As mentioned in the previous section, the obstacle avoidance sometimes comes into conflict with avoiding walls. The track-specific trimming that is needed should be handled automatically. This could be done either by constructing a trimming module, or - preferably - by integrating different avoidance behaviors to one robust unit.

The guidance algorithm currently does not consider the movements of the other car. This could be improved in increments, from adding the simplest form of reaction - to break when the other car gets in the way - to arbitrarily complex strategies for overtaking, outmaneuvering, etc.

3.2 Organization and hardware

Apart from racing specific experiences we encountered several other aspects of project work.

3.2.1 Things that worked

Several things worked quite well during the project. We were allowed access to the project room as much as we needed. The supply of batteries and other things we needed was also satisfactory. The service that Andrey provided was extraordinary. He was available at any time and always most helpful. The image processing equipment worked as expected without posing any severe problems, and so did the network communication. The course administration did a good job, considering the fact that this was the first time the course was given.

3.2.2 Things that can be improved

There where two major problems with the equipment. Firstly, the transmission of pulses from the computer to the car was done at too low frequency. This led to a very jerky motion for the car at low speed. A frequency of about 10 Hz is far to little for this application. What one wants is a pulse-width that is so short that the equalization of the speed is done inside the motor and not on the floor. Secondly, there were some problems with the camera. The car could be seen as a dot on the computer-screen, but we did not get any coordinates from the image processing equipment. This problem only occurred at corners and along some of sides of the racing arena. Another thing that seemed strange was that we sometimes got the coordinate $(0, 0)$ from the camera when the car was somewhere in the middle of the arena.

The organization of the course was mainly very good, but we would like to point out some things that could have been done differently and perhaps better. The lectures were all interesting and gave some idea about problem areas that we could run into during the development of the project, but in the end they did not give much help in solving the given task. If this is good or bad we do not know, since we are not really sure about the goals of the course, but it also means that we could have started working with the project the same day we had our introductory meeting. The project became quite big and therefore it would have been nice to get started with the project already in the beginning of February, working in parallel with the project and going to lectures.

3.3 Experiences of the course

Working with this project has been a lot of fun and very interesting, but it feels like more time is needed to reach the initial goals of the course. Therefore we think it would be better to give this course more time (and academic credits) to fulfill the goals of making the cars compete against each other and implement things that comes closer to our research areas. If the project is not supposed to reach these levels the course administration must aim a little bit lower and

not get too disappointed with our results. Our recommendation is to make this course bigger, because of the fun and interest we had working with it.

In spite of the fact that the project team was divided in two, working on different parts of the system and problem, the "interdisciplinarity" purpose of the POS course worked out well. To design working solutions to each subproblem, and to integrate them into a working system, some knowledge of the problems and methods of all other parts was necessary.

The project itself has also given us some useful experiences. When working against the physical reality, with all its uncertainties and shortcomings, theories are put to a very concrete test, and difficulties that are otherwise easy to ignore have to be dealt with. It also showed that seemingly very simple systems may be very hard to model accurately, and that using simple models of complex systems may, in spite of their inaccuracy, provide good overall results.

Appendix

Identifier Algorithm

Figure 5 shows a flow chart of the identifier algorithm.

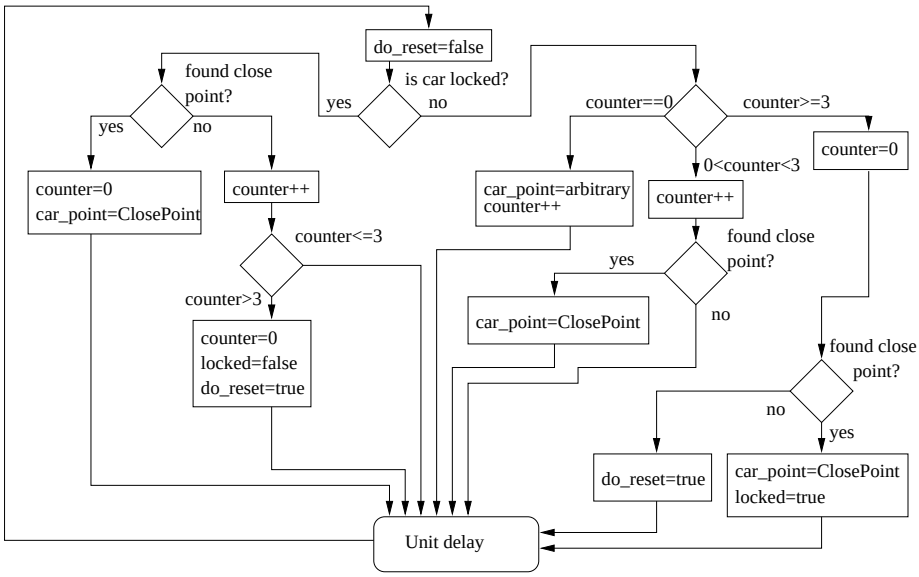


Figure 5: A flowchart of the identifier algorithm

User Manual

This user manual contains a description of how to start the system.

Path editing

First the track has to be entered to the system. This step results in a textfile named path.txt.

Figure 6: The track editor

- Start the track editor with the command
  java TrackEdit
- The track editor allows the user to enter obstacles, waypoints and acceptance areas. See Figure 6.
- Save the track by clicking the **Write** button.

Run time system

After setting up the hardware, which includes starting the computers and switching on the car, the system can be started by the command **testrun**. This invokes a script that starts the software components.

The run time system displays two windows (see Figures 7 and 8). One enables the user to change parameters of the system and the other displays information about the state of the system.

The first 8 parameters in Figure 7 are assumptions of noise levels for the Kalman filter. Lambda is the forgetting factor used in the adaptive car dynamics estimator. The parameter v_ref specifies a reference velocity, and is not used in the final system where the path follower is responsible for setting the desired speed. The 6 last parameters (K through T) are used by the speed regulator.

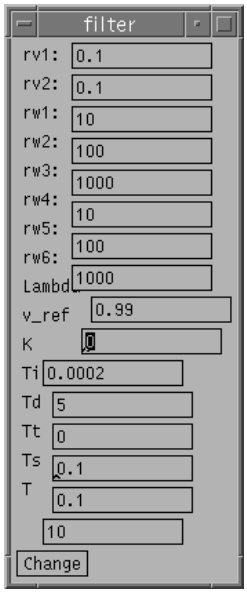


Figure 7: Parameter editor

Figure 8: Run time system monitor