ISSN 0280-5316
ISRN LUTFD2/TFRT--5762--SE

Online Terrain Classification for Team Caltech in DARPA Grand Challenge

Lisa Nyström

Department of Automatic Control
Lund University
November 2005

Department of Automatic Control **Lund Institute of Technology** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER THESIS *Date of issue* November *Document Number* ISRNLUTFD2/TFRT--5762--SE
Author(s) Lisa Nyström	*Supervisor* Richard Murray at Caltech, USA Anders Rantzer at Automatic Control in Lund *Sponsoring organization*

Title and subtitle
Online Terrain Classification for Team Caltech in DARPA Grand Challenge (Terrängklassifiering för Team Caltech i DARPA Grand Challenge).

Abstract
The US Department of Defense wants one third of ground military forces to be automated by 2015. By using self-driving military vehicles, American troops can be kept out of harm's way in the battlefield.[1] To speed up the development, the US Department of Defense decided to arrange a desert race for autonomous vehicles, DARPA Grand Challenge. Team Caltech is entering the Challenge with the vehicle Alice. She is equipped with sensors that scan the surroundings and a GPS that gives Alice her location. The GPS signal is the only type of communication that is allowed during the race. This master's thesis describes the work that Team Caltech did to prepare for DGC. The thesis also describes the work that I did as a member of Team Caltech: administration of the testing and on-line terrain classification. Testing is a critical part in the development of an autonomous vehicle. As time is limited, it is essential to test the right things and the right types of terrain. To improve the efficiency and effectiveness of the testing, I developed a test matrix for different terrain types. This matrix helped us to make sure that we were testing different types of terrain. Having developed the matrix, I was also responsible for the documentation of the testing. The vehicle acts differently depending on the type of terrain. Thus, it is valuable that Alice can differentiate the current terrain type. I have written a program in C++ that can distinguish and classify different terrain types. The C++ program was not implemented and tested in real conditions, because time was too short. However, it will be useful for the future.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280-5316		*ISBN*
Language English	*Number of pages* 40	*Recipient's notes*
Security classification		

First of all, I want to thank my mentor at Lund Institute of Technology Anders Ranzer and my mentor at California Institute of Technology Richard Murray for giving me this great opportunity. I had a great time, learned a lot and gained valuable experience. I would also like to thank my co-mentor Lars Cremean for being patient and explaining everything to me.

I would also like to thank my team members in Team Caltech that I have been working together with towards the common goal to win the DARPA Grand challenge 2005. The team consisted of individuals with different skills and backgrounds, which gave me an experience of both the benefits that can be achieved from differences and the struggles to overcome differences.

I want to thank Kristofer for helping me with proofreading.

Lisa Nyström 2005-11-03 Lund

Gotcha Chart

I spent my first week at Caltech in meetings. Some days, we had meetings from 9 in the morning to 9 in the evening. We discussed the GOTCHA chart, i.e. Goals, Objectives, Technical CHallenges and Approach. This first chapter gives the reader information about Team Caltech's GOTCHA chart.

Goal

Team Caltech's overall goal is to[2]:

- build, test and document a vehicle that wins the 2005 DARPA Grand Challenge
- qualify for the race
- perform publishable research on state-of-the-art problems in robotics
- effectively manage the "signal to noise" ratio of a large project

As a member of Team Caltech, I helped the team to achieve these goals by:

- developing tools for making the test system more effective and being responsible for the documentation of the testing.
- developing a tool for online terrain classification.
- helping out where help was needed, e.g. writing code for different modules, mechanical work on BOB3, helping out with the testing, preparing the mock qualification course and sponsor presentations.

Team Caltech's goals were broken down in spirals. Each spiral constitutes 4 weeks' work and contains the overall goals for that period. To achieve the four-week-goals, the objectives have to be completed. The objectives are weekly.[4] The difference between objectives and goals is that goals are what should be achieved, whereas objectives are explicit things that need to be done.

[2] www.teamcaltech.edu, 2005-10-25

[3] BOB is last year's race vehicle

[4] For our approach, objectives and spiral log on to: http://gc.caltech.edu/wiki/index.php/Project_GOTChA_Summer_2005

Team Caltech listed a number of technical difficulties that needed to be solved, to enable to complete the race. Some of these technical difficulties are listed below:[5]

1. Robust vehicle capable of surviving testing without damage

2. Very aggressive schedule between beginning of summer, Preliminary Implementation Review (PIR)[6], Critical Implementation Review (CIR)[7] and race

3. Many untested technologies under consideration; could serve as a distraction

4. High level decision making and learning not yet planned

5. High speed technical driving in rough terrain not yet demonstrated; likely to be hard and risky

6. System still susceptible to component and interface failures, resulting in substantial downtime

7. Large team, with diverse background, skills, experience, and availability

[5] www.teamcaltech.edu, 2005-11-09

[6] For the PIR, researcher and sponsors were invited for a presentation of the current state of the project and the plans for the future.

[7] For the CIR, researchers and sponsors were invited for further discussion and feedback.

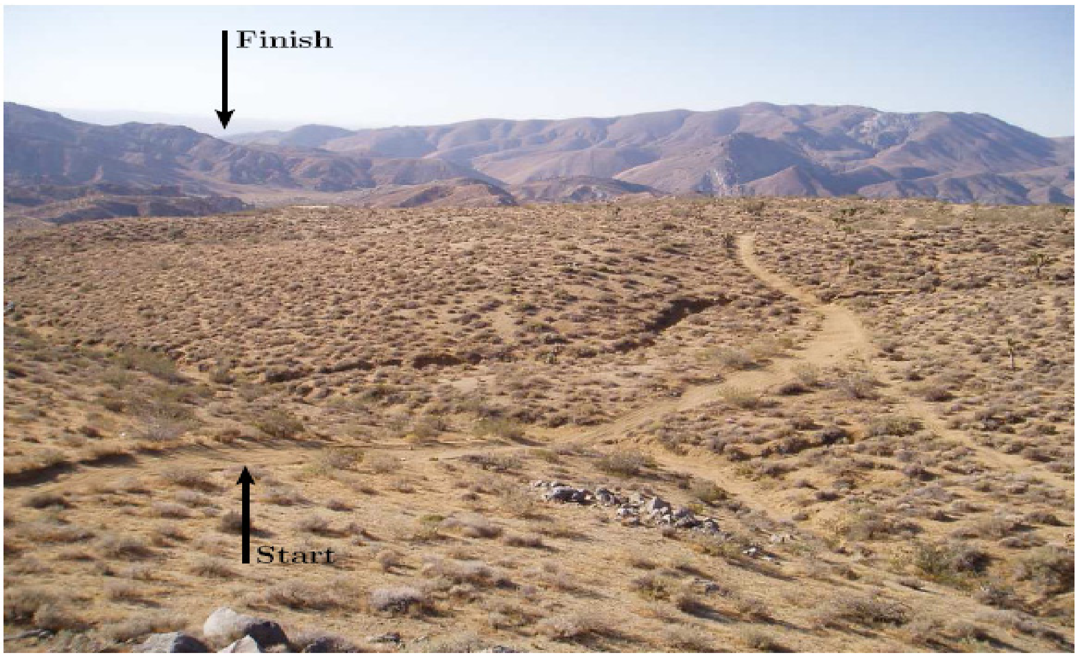
1 Introduction

This chapter introduces the DARPA Grand Challenge, the National Qualifying Event, Team Caltech and the race vehicle Alice.

1.1 DARPA Grand Challenge (DGC)

DGC is a desert race between fully autonomous ground vehicles. DGC is sponsored by the Defense Advanced Research Projects Agency (DARPA). The ultimate goal is to build unmanned self-driving military vehicles and thus keep American troops out of harm's way in the battlefield[8]. They want one third of ground military forces automated by 2015. The purpose with arranging DGC is to speed up the development of autonomous vehicles.

The course is about 150 miles, with a 10 hour time limit. There are a couple of important rules that has to be followed: the vehicle must stay within a certain corridor throughout the race; moving obstacles (handled by DARPA) must be avoided; the vehicle must not intentionally touch any other competing vehicle; and GPS signals are the only signals allowed between the participating teams and the vehicle.[9]



Picture 1.1 Is Alice capable to complete this course?

8 www.darpa.com, 2005-11-12

9 www.darpa.com, 2005-11-12

In the 2004 DGC, the first year of the event, none of the autonomous vehicles came even close to meet the challenge. Year 2004 the prize money was USD 1 million. For this years competition, Dr. Tony Tether, the director of DARPA, announced that the prize money had been increased to USD 2 million[10]. To participate in the race, your vehicle first has to go through the semifinal, which is called the National Qualifying Events (NQE). The NQE consists of a technical inspection and an assault course (fig. 1.2).

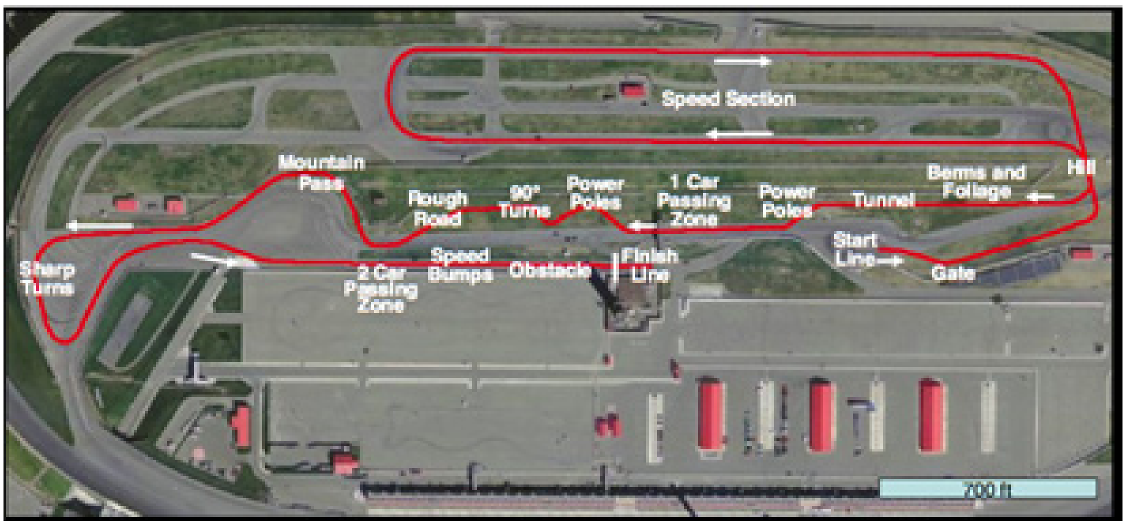


Figure 1.1 This is the 2005 NQE course, which had to be completed to make it to the final race

1.2 Team Caltech

Team Caltech consists mostly of undergraduates and is divided in three race teams: vehicle team, planning team and terrain team. These three teams have three distinctly defined areas of responsibility. *The vehicle team* is responsible for locating and attaching every component to the vehicle; for performing vehicle upgrades; and for implementing the power system. *The terrain team* is responsible for the sensors (e.g. LADAR units, cameras, radar) that scan the surrounding area and detect obstacles. *The planning team* writes software that decides which direction to turn and how fast to go. Each race team has a graduate student as a team coordinator, a professor as an advisor, and a number of undergraduate students that are responsible for the majority of the work. In total, Team Caltech consists of 36 members from the four schools Lund Institute of Technology (4), Princeton (1), Virginia Tech (1) and Caltech (30)[11].

Team Caltech has been working on the project since March 2003, with a budget of $ 160 000[12]. Team Caltech's main sponsors are: California Institute of Technology, Sportsmobile and Ford.

[10] www.darpa.com, 2005-11-13

[11] www.teamcaltech.edu 2005-11-13

[12] www.caltech.edu 2005-11-16

1.3 Wiki – Team Caltech's home page

The Wiki is the location for almost all project documentation. Everything from contact information to the team members, notes from the meetings, papers and thesis from team members, information about how the different modules work and how to use different tools and more.

1.4 Bugzilla

Bugzilla is a free independent "Bug-Tracking System" web browser that helps individuals or groups of developers to keep track of outstanding bugs in their product effectively[13]. It is difficult to keep track of all the documentation in a big project. Therefore, Bugzilla was a useful administrative tool. All the things that need to be done become bugs in Bugzilla. When a bug is modified, solved or accepted, an automatic email is sent out to all the people involved in the bug. If someone has spare time, that person can always go to Bugzilla to find out what needs to done. In the same way, if someone else has too much to do, that person can search for team members that have time to spare, and assign those team members a bug.

1.5 Subversion

To keep track of all the source code, Team Caltech used a versioning system called Subversion. Subversion can handle different versions of a file and remember its history, which makes it possible for several people to work on the same files. Subversion only updates the changes between each revision instead of copying all the files; which saves a lot of disk space.

1.6 Meeting

Monday was the meeting day. First we met all together, than we held separate meetings in the different race teams. After that, we all met up again to talk about cross team issues that had come up under the race team meetings.

1.7 Alice

Alice is the name of the vehicle that Team Caltech competed with in the DGC 2005. She is a Ford E350 van that has been highly modified to become capable off-road (fig 1.3).

Alice has two types of sensors: terrain sensors and state sensors. The state sensors are GPS and IMU. The GPS provides precise inertial position data (but imprecise velocity data). IMU provides precise velocity data (but imprecise inertial position data). Combined[14], these data sources produced state reading that is precise overall[15].

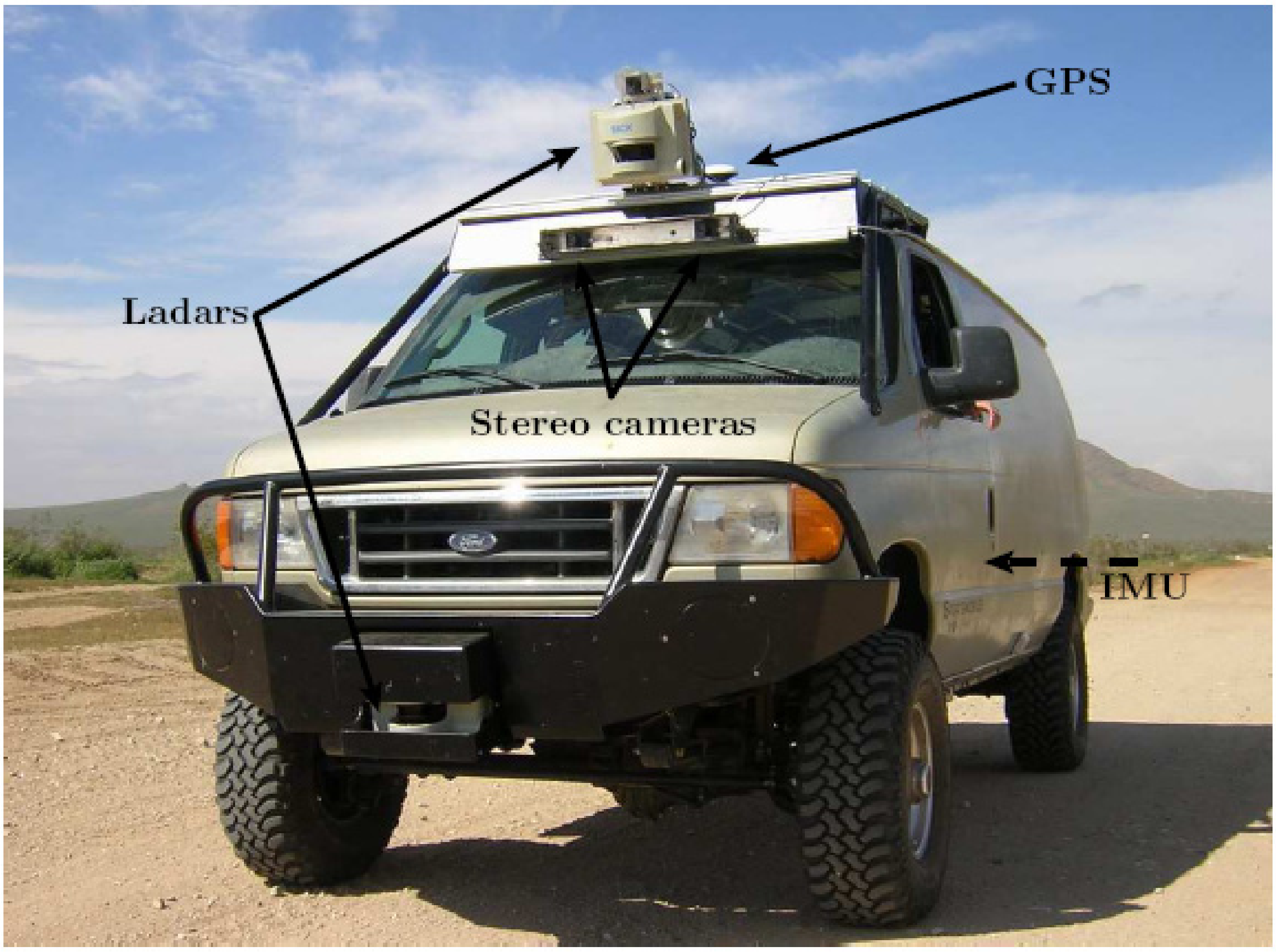
---

[13] www.caltech.edu , 2005-10-25

[14] In this case through a Kalman filter.

[15] Kogan Dimitri: Real Time Path Planning Via Non Linear Optimzation Methods (2005), Pasadena

Alice has a range of terrain sensors that tells her what the surroundings look like. Alice is equipped with two pairs of stereovision (that when they are combined give her the depth of the surroundings), three ladar units (that uses a spinning mirror that scans a laser beam that reads the distance it hits), a road following color camera, and more.



Picture 1.2 Alice's main sensors

2 Planning architecture

This chapter gives an overview of the different modules and how they work together.

The programming at Team Caltech was divided in modules. Each module has an owner, and I was responsible for the Dynamic Bumpiness Sensors (DBS). Having one person responsible for each module, we made sure that all the modules were given sufficient attention. This is a short description of the most important modules, and the ones that I have been working on. Except for the DBS, I also wrote code for RDDF PathGen and TrajFollower.

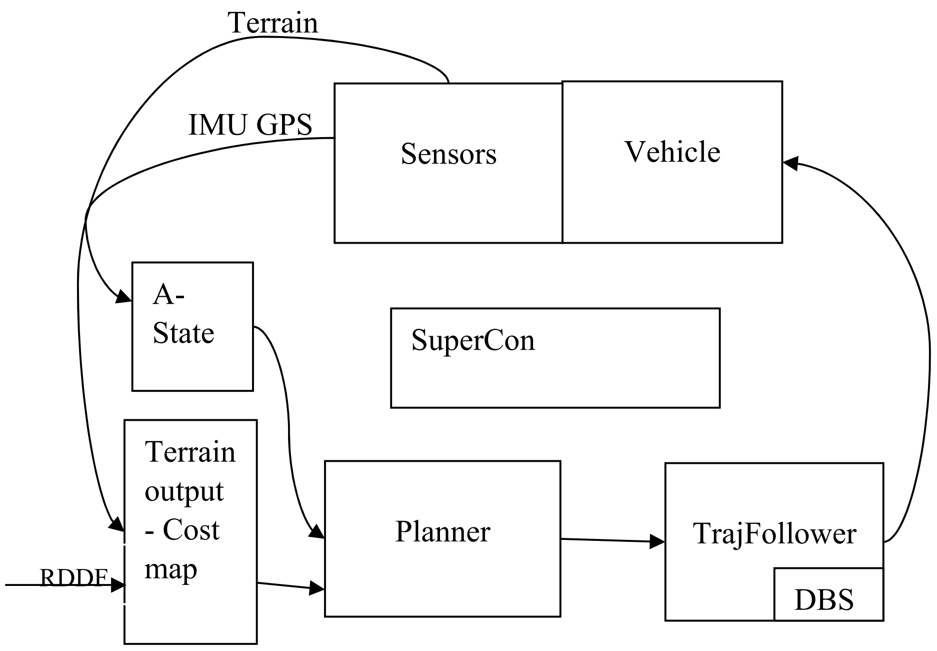


Figure 2.1 An overview of the planning architecture.[16]

[16] The figure is based on a figure on www.teamcaltech.edu and Kogan's thesis "Real Time Path Planning Via Non Linear Optimzation Methods"

2.1 Cost map –the output from terrain team

The terrain team uses the information from the terrain sensors. The information from the sensors is translated to an elevation map. The elevation map is a topographic description of the surrounding. After that, the topographic information is translated into a cost map. A cost map assigns areas with different topographic properties different costs. The cost can be translated to speed. For example, an area with a big rock gets the speed set to zero, whereas flat area is assigned maximum speed.

2.2 RDDF

The vehicle must stay within a certain corridor throughout the race. This corridor is provided by DARPA through a Route Description Data File (RDDF). This file contains information not only about the corridor width and length, but also about the maximum allowed speed. A waypoint in a RDDF consists of a northing and an easting coordinate; a maximum radius of the distance; and a maximum allowed speed. The corridor is provided by DARPA two hours before the race.

RDDF PathGen is a module that generates the middle of this corridor. Suppose that there were no obstacles on the path through the race, Alice would not need sensors. In this scenario, she could simply rely on the path generated by RDDF PathGen.

2.3 Dynamic Bumbines Sensors (DBS)

DBS is a class that acts on state data and provides the output online terrain classification and speed cap. Terrain classification is important because the vehicle acts differently depending on what ground the vehicle is traversing. The information about the terrain class is used by the TrajFollower. The speed cap information is integrated in the cost map. The DBS assigns a higher cost (i.e. lower speed) when the vehicle is traversing rough terrain that has not been detected by the sensors, and thus not prior integrated in the cost map.

2.4 Planner

The Planner receives data from the A-state and the cost map. The Planner processes this data based on numerical optimization to determine the best route and speed. The output is a trajectory. The trajectory is sent by the planner to the TrajFollower.

2.5 TrajFollower

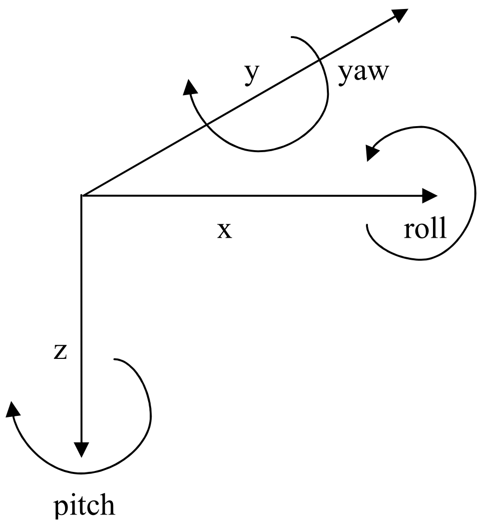
The TrajFollower component receives the trajectory from the Planner, and information about the current vehicle state from the State reader. Based on the state information, the TrajFollower corrects the vehicles route according to the trajectory by a PID-controller and commanding the steering, throttle and brake (via the actuators). The specifications for TrajFollowing are that it must be capable of controlling Alice such that she can follow a path with <±0.2m accuracy in her perpendicular distance from the path, and her actual speed must be within +0%, -5% of the commanded speed[17].

2.6 SuperCon

There are a lot of different scenarios that can appear, and it is not possible to be prepared for everything. The sensors do not give a 100% accurate image of the surrounding and for example it is almost impossible for the sensors to detect a thin fence. If something goes wrong, for example if Alice gets stuck in a fence (like last year!), then SuperCon sets in. SuperCon paints an obstacle on the map where Alice got stuck, tells Alice to reverse and start up from the beginning.

2.7 A-state

The GPS and the IMU combined through a Kalman filter and give the vehicle state. The vehicle state contains information about the North (x), East (y), Z down (z), pitch, roll, yaw and the first and second derivatives of each. When a module needs to know the position of the vehicle, the module calls A-state and A-state returns the vehicle state with a minimum latency.



Figur 2.2 The coordinate system that describes Alice position.

[17] www.teamcaltech.edu

3 Testing

To win the DGC, the equipment that Team Caltech has developed for Alice must be tested and configured. This chapter describes how I improved the effectiveness and efficiency of the testing.

3.1 Problem description and Approach

The test time is limited and risky. Therefore, it is of high priority to make sure that Team Caltech is testing the right things in the right way.

My first testing experience was fun, but frustrating. The purpose of my first testing session was to teach all the new team members to safety drive Alice. The safety driver is the person who sits in the front seat and is ready to pull the plug if Alice is in danger. The team was divided in four groups. My group was the last one to test drive. We drove to Santa Anita around 4 o'clock in the afternoon. Santa Anita is approximately 15 minutes from Campus. When we arrived at Santa Anita, something was wrong with Alice. Therefore, it took 4 hours before the group prior to mine could start. Situations like this were typical, and could have been avoided if planned better.

There have been several problems like this during the summer. Not only that we have had problems being efficient, we also really bad luck with the testing during the summer. At my first desert testing, the air condition crashed, and Alice had to go to the mechanical workshop to get repaired.

Picture 3.1 Testing at Elmirage

We have three main test areas: Santa Anita, which is a parking lot close to Campus; Elmirage (pic 3.1); and Stoddard Wells (fig 3.1), which is real desert terrain, two hours from campus.

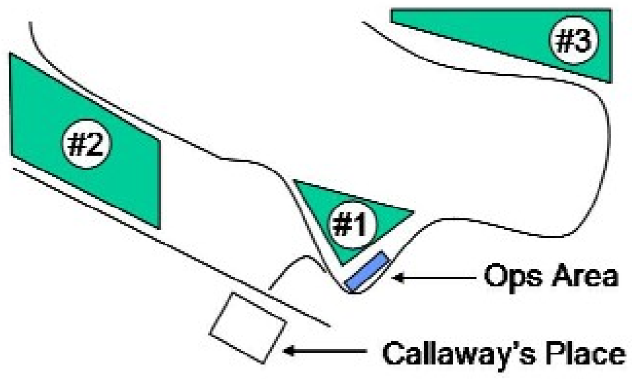


Figure 3.1 Test Area #1: Lakebed test area
Test Area #2: Dirt road/trail area; whoop-de-doos and other rough terrain.
Test Area #3: Additional desert terrain

There are a lot of preparations, both before and during a test. This requires a great deal of planning: Who is available? Who needs to be there? What modules do we need to test? Who are going to be safety drivers?

The people that are going on a test drive have to meet several times to plan the testing. The night before the test, the test team has to go through a checklist and to make sure that everything is prepared. Moreover, all the modules are run in simulations before the test team leaves for the desert. Thus, the team does not have to waste time fixing bugs in the desert. One important rule is: “Do not leave the shop unless we know that the things we want to test are working in a controlled environment”. This is a rule that should be fairly easy to follow. None the less, Team Caltech lost a great deal of time because the rule was not followed. Even though there are rules for the testing, rules are not useful unless the rules are also followed.

To win the race, we would have needed to drive a distance of about 150 miles in a time limit of 10 hours and we need to be able to handle different types of terrain. By watching the movie from last year’s DGC, I divided the terrain in different classes. I also managed the wiki documentation for the testing, and helped out with the preparations for the testing. The matrix that I developed was also helpful in developing the speed strategy for the race.

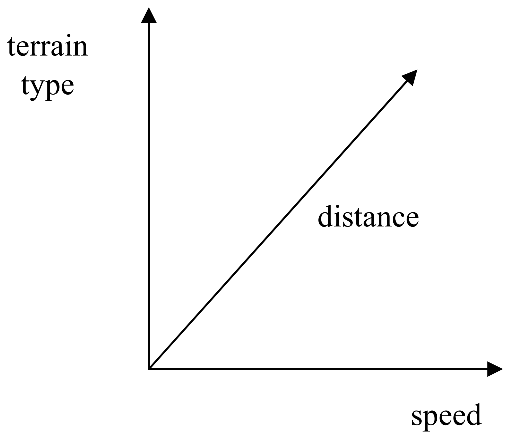


Figure 3.2 To win the race we need to drive a distance of approximately 150 miles within a time limit of ten hours. Moreover we need to handle different types of terrain.

3.2.1 Test Matrix

The main idea with the test matrix is to divide driving into regimes and push envelope in three axes: terrain/RDDF type ("difficulty"), speed and distance.

These requirements should flow down to configuration of sensors and configuration of modules. The sensors and modules configuration was divided in four groups:

- Module configurations
- Cost map configurations
- Planner Module configurations
- Sensor contributions

By dividing the testing into these four groups, we made sure that we tested all the different configurations of sensors and modules.

As a first step in my project, I made a table of the different types of terrain. I watched the movie from the last year course run, which helped me to understand what different types of terrain and what kinds of different situations we needed to handle.

I started to check every part of the movie, and to categorize the different parts depending on the rolling, the winding, the width and the terrain roughness (tab 3.1).

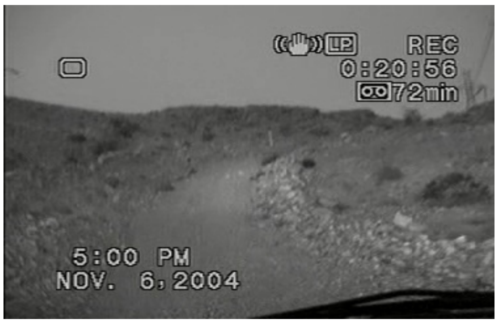
Each type (width, rolling, winding and terrain roughness) is divided in four levels of difficulty, 0 is the easiest terrain type and 3 is the most difficult terrain type. The difficulty level of these four types determines how demanding it will be for Alice to traverse the terrain.

Paved road **Dirt straight road, mild/medium**

Dirt straight road, medium/extreme

Winding (almost no rolling)

Rolling (almost no winding)



Climbing or descending from mountain mild/medium

Specials medium/extreme

Climbing or descending from mountain medium/extreme

Picture 3.2 We need to handle different types of terrain and a distance of 175 miles in a 10 hours limit

Table 3.1. Difficulties that describe the terrain with a scale from 0 to 3.

Type	Width	Rolling	Winding	Terrain roughness
0	wide open	no rolling	no winding	paved road
1	>3 Alice	mild	mild	Mild
2	2-3 Alice	medium	medium	Medium
3	<2 Alice	extreme	extreme	Extreme

As a second step, I tried to categorize the result of the almost 150 different terrain types in 8 "bigger" groups, though it would not be clear to put all 150 in a table. The results are shown in the tables below (tab 3.2). It also includes the test area, the difficulty (average/max), and the expected distance of each terrain type of the total course.

The test matrix in table 3.2 is used to keep track of our testing and to make sure that we cover the expected terrain types. Information about the testing was administered through the Wiki. The following information I filled in after each test session:

- How many autonomous miles
- The max speed
- Terrain type
- What went wrong
- What kind of test (modules, sensors and so on)

In the beginning of the summer, Team Caltech set up goals and requirements: a weekly schedule of what needed to be done to win the DGC. In the beginning, the table helped the team to see what needed to be added or adjusted in test plan. It gave us a good overview of how close we were to reach the goal and what further improvements needed to be done. It was a very useful tool for making decisions.

Table 3.2 I divided the total course in 8 different terrain types.

Terrain type	Distance (miles)	Percents (%)	W	R	win	tr	difficulty	test area
Paved road	12	9.1	1	≤1	≤1	-	0.17	Santa Anita
Dirt straight road, mild/medium	44	33.3	1	-	-	1-2	0.21	Stoddard Wellsloop
Dirt straight road, medium/extreme	43	32.5	≤2	-	-	3-4	0.38	Stoddard Wells
Winding (almost no rolling)	4	3	≤2	≤1	2-3	>2	0.58	Camp rock road
Rolling(almost no winding)	9	6.8	≤2	2-3	≤1	>2	0.58	
Climbing or descending from mountain (both rolling and winding) mild/medium	11	8.3	≤2	2-3	2-3	2-3	0.78	Dagget ridge
Climbing or descending from mountain (both rolling and winding)medium/extreme	8	6.1	≤3	3	3	3	0.96	
Specials	1	0.8	X	X	X	3	1	
s1, Tunnels							1	Victorville
s2, water crossing							1	Mojave River
s3, ditches (unexpected)							1	
s4, rough off road terrain							1	
Total	132[18]							

3.2.2 Speed Strategy Based on the Test Matrix

In last DGC 2004, no team came even close to crossing the finish line. The winning vehicle traveled less than 10 miles before breaking down. Caltech's BOB managed to cover 1.3 miles before BOB got stuck in a fence. Year 2005, Caltech's Alice had been able to cover 31 autonomous miles on easy dirt road without stopping. Based on this experience, Team Caltech's race strategy was to drive as slow as possible, but within the ten hours time limit.

[18] The total mileage should have been 175 miles. However, parts of the movie from last years course run was of poor quality.

Knowing what terrain types to expect was here very useful. I calculated the expected time to complete the race course based on different speeds and the time limit of ten hours. The result showed that a maximum speed of 30 miles per hour would be sufficient to cover the course in 10 hours. Alice is allowed to drive at maximum speed on easy terrain. The maximum speed is the basis for calculating the speed on the more difficult terrain types. The maximum speed that DARPA allows is 40 miles per hour.

Table 3.3 Based on the strategy to drive as slow as possible but make sure to complete the race within the limit of 10 hours, the maximum speed was set to 30 miles per hour.

Data values are yellow	Total (mi)	175	1 mph (m/s)	1 mile (km)
Calculated values are orange	Total (km)	281.6	0.44704	1.609344

Terrain type (codes)	Min (mph)	Max (mph)	Avg (mph)	Expected (%)	Expected (mi)	Time (hours)
Paved road (P)	20	30	25	9.1	16	0.64
Dirt straight road, mild/medium(D1)	20	30	25	33.3	58	2.33
Dirt straight road, medium/extreme(D2)	15	25	20	32.6	57	2.85
winding(w)	15	25	20	3	5	0.26
rolling	15	25	20	6.8	12	0.60
climbing or descending from mountain,mild/medium(M1)	10	20	15	8.3	15	0.97
climbing or descending from mountain,medium/extreme(M2)	5	10	7.5	6.1	11	1.42
specials(S)	3	8	5.5	0.8	1	0.25
totals	3	30		100	175	9.32

4 Online Terrain Classification

This chapter gives a description of online terrain classification, which is a part of the Dynamic Bumpiness Sensors (DBS)

4.1 Problem Description and Approach

The course covers for example the terrains; pavement, gravel, rocky and sand. Tunnels, slopes and other obstacles occur on the course (fig 4.1). The terrain sensors describe the surrounding as height differences and distances. However the ground is a problem in itself. Depending on the terrain type, Alice behaves differently. For example, Alice might climb a rocky slope with ease, but slide on a sandy slope with the same gradient. Knowing how Alice behaves on different terrains, and knowing the current terrain, her control and planning strategy can be adapted. With such knowledge, Alice could be enabled to traverse the race course in a safer and more efficient way.[19]

Picture 4.1 Different types of terrain that can occur on the race

[19] Brooks, Ignemma, Kart, Dubowsky, "Vibration-based Terrain Analysis for Mobile Robots" ICRA 2005

I divided the terrain classification problem into three steps. First, I collected state data, both by gathering existing logs with state data and by logging new data when test driving Alice in different terrain types. Second, I loaded the state data to Matlab and analyzed the data by looking at the frequency domain. Third, based on my analysis, I wrote a program in C++ that gives the terrain classification as output (fig 4.2).

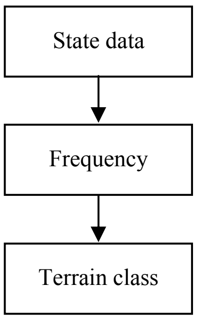


Figure 4.1 The program reads in state data and translates it to the frequency domain and gives terrain classification as output

4.2 Theory

4.2.1 Fourier transformation

In a frequency analysis, the mathematical tool Fourier transforms is used to transform a signal from the time domain to the frequency domain. The Fourier transforms is defined for time continuity as well as for time discrete signals.

The continuous Fourier transform is defined as:

$$f(\nu) = \mathcal{F}_t[f(t)](\nu) = \int_{-\infty}^{\infty} f(t)\, e^{-2\pi i \nu t}\, dt.$$

The discrete Fourier transform is defined as:[20]

$$F_n \equiv \sum_{k=0}^{N-1} f_k\, e^{-2\pi i n k/N}.$$

In the time domain, time is given on the x-axis and the value that is studied is given on the y-axis. A frequency study has frequency on the x-axis and effect on the y-axis.

Swiftly varying signals in the time domain correspond to high frequencies, and slowly varying signals correspond to low frequencies. We were hoping that this could be useful in determining the terrain type.

[20] www.mathworld.wolfram.com/FourierTransform.html den 2005-11-03

When working in the frequency domain it is important to have knowledge about The Nyquist theorem. The theorem states that when sampling a wave, at least two samples must be made during each period to be able to reconstruct the original wave. For example, to measure a 20 Hz wave, you have to sample with a frequency of at least 40 Hz. If you sample at a lower frequency, the alias effect will occur. Under the alias effect, a different curve is depicted than the actual curve.

4.2.2 Previous research about online terrain Classification

So far, most research on terrain classification has focused on how to avoid obstacles, such as cliffs and tunnels, in the surrounding. However, this type of research rarely considers the ground as a problem in itself.

In 2002, Brooks and Dubowsky got the idea to researched terrain classification based on vibration analysis. The vibrations were measured with an accelerometer.

In April 2005, a Massachusetts Institute of Technology (MIT) research team used the vibration based method to distinguish sand, mars and gravel. The team observed vibrations in an offline training phase. Based on these observations, the team calculated the voltage in the time domain and than calculated the Power Spectral Density. After that, further analysis was made in the frequency domain. The analysis of the observations was saved in a memory for the online phase. During the online phase the vibrations were measured and a classifier voted for each terrain type. In this case, there were three terrain types. There were three classifiers: one to separate sand from gravel, one to separate gravel and mars and one to separate mars from sand. In the online phase it could separate sand with 97% reliability and gravel with 98% reliability and mars with more than 80% reliability. In approximately 30% of the cases, the classifiers gave the response: Unknown. The classifiers were programmed to prefer Unknown to giving a wrong answer.[21]

A different type of online classification is base on wheel sinkage into deformable terrain; friction angels; and soil cohesion. This type of online classification has been developed at MIT to determine terrain traversability[22].

[21]Brooks, Ignemma, Kart, Dubowsky, "Vibration-based Terrain Analysis for Mobile Robots" ICRA 2005

[22] Brooks, Ignemma, Kart, Dubowsky, "Vibration-based Terrain Analysis for Mobile Robots" ICRA 2005

4.3 System design analysis

4.3.1 Data Collection

I collected state data by running the module A-state during a test run with Alice. A-state receives the information from the GPS about the position of the vehicle and the Inertial measurement unit (IMU) measure the acceleration of the vehicle, i.e. how much the vehicle is vibrating. A-state takes the GPS and the IMU and combines them through a Kalman filter. The output is state data and has the format of: time, north(x), east(y), down (z) velocity north, velocity east, velocity down, acceleration north, acceleration east, acceleration down, roll, pitch, yawing, roll rate, pitch rate, yaw raw, roll acceleration, pitch acceleration, yaw acceleration.

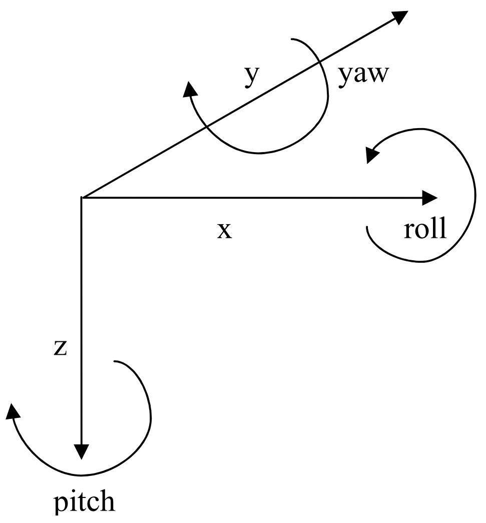


Figure 4.2 The coordinate system that describes Alice position; x-axis is the north direction, y-axis is the east direction and z down describes the down direction. By logging data with A-state you receive information about, x, y, z, roll, pith, yaw, and first and second derivatives of each parameter.

No one on Team Caltech had prior experience of collecting and analyzing data in frequency plan. Therefore, I had to figure out how to get hold of data from prior test runs. I found a file with information on test run on pavement and dirt road. However, the data on the file was sampled at different frequencies, which made the analysis difficult.

I calculated the sampling by measure the time T between 2 measurements (time is measure in million seconds). The sampling frequency is than, f_s=1/T. I tried to cut out the bad parts of the existing state data for example the parts with very high- and very low frequency and when Alice stood still.

```
1125631142659411    3778447.599    403875.9782    -110.7192235    1.959111453    -3.666690501    0.06280033722    -3.662847292    2.759586052    2.518255066    -0.03328724576  -0.001228294347 -
1.169814611    -0.001525878906 -0.006103515625 0.002288818359  0.003051757812  -0.004577636719 0
1125631142661022    3778447.631    403875.9061    -110.7180914    1.952220667    -3.651510029    0.08286618063    -3.873230049    5.894292708    3.208815766    -0.03329436629  -0.001571139338 -
1.169760779    0.001525878906  -0.01678466797  0.002288818359  0.001525878906  -0.006866455078 0.0007629394531
1125631142662792    3778447.664    403875.8338    -110.7168768    1.951141464    -3.654475761    0.08670075792    -4.187443868    4.201028939    2.886720463    -0.03321929038  -0.001917808741 -
1.169707131    0.006866455078  -0.02059936523  0.002288818359  0.002288818359  -0.006103515625 0
1125631142664490    3778447.696    403875.7615    -110.7153809    1.958002249    -3.657738459    0.1060085517    -1.203433378    7.488131323    4.354262015    -0.0330237051    -0.002321726629 -
1.169653217    0.0114440918    -0.02212524414  0.001525878906  -0.0007629394531       -0.007629394531 0
1125631142666066    3778447.728    403875.689    -110.713702    1.948478277    -3.668370208    0.103228789    -6.348279915    5.145616659    3.190345518    -0.03266968427  -0.002767505139 -
1.169595206    0.02212524414    -0.02212524414  0.0007629394531 0.003814697266  -0.006866455078 -0.0007629394531
1125631142667806    3778447.761    403875.6162    -110.7120614    1.966168292    -3.680586452    0.1019638653    -4.168676508    3.769630029    2.060598839    -0.03222959506  -0.003120936832 -
1.169513632    0.02365112305    -0.01525878906  0.003051757812  0.003051757812  -0.004577636719 0
1125631142669518    3778447.793    403875.5433    -110.7104875    1.957843956    -3.687359554    0.09335733303    -0.9268715102    0.0778730255    0.4262890333    -0.03168651407  -0.003453341832
-1.169429926    0.02899169922    -0.01525878906  0.002288818359  0.001525878906  -0.0007629394531       0
1125631142671282    3778447.826    403875.4701    -110.7090851    1.966267376    -3.691552268    0.09460008783    -4.762214927    4.134760056    2.427353581    -0.03115392644  -0.003645159427 -
1.169364905    0.02822875977    -0.0129699707   0.001525878906  0.003051757812  -0.005340576172 0
```

Figure 4.3 A sample of state data received from logging data with A-state. Here are the 11 first columns, from time to roll.

There were inconsistencies in the data that I was unable to explain. It even looked like the vehicle incurred more pitch and roll on paved-road then rough terrain. I could not figure out what the problem could be, and asked my mentor for help. Eventually, the student responsible for the A-state told me that the data collected from dirt road was of poor quality due to a problem with the A-state.

Having rejected the existing data due to poor quality, I decided to collect a new sample of data. With this sample, I hoped to make sure that a fourier transforation (fft)-analysis in Matlab is an adequate way to compare different types of terrain. I collected data:

- when the vehicle stood still
- when the team was pushing Alice from the side, and the vehicle incurred a lot of roll
- when the team was pushing Alice from the front, and the vehicle incurred a lot of pitch

The data was sampled at 40 Hz. According to the Nyquist theorem, the Fourier transformation returned information in the spectrum 0-20 Hz. The new data showed differences that could be expected and analyzed. It gave an indication of what frequencies to expect. The frequency seemed to vary between 0-5 Hz.

4.3.1.1 How much data is sufficient to determine the road-type?

The objective is to classify the terrain as fast as possible without misinterpreting information. If the classification is based on information received during a short period of time, the lowest frequencies will not be recognized. At the same time, the DBS functions as a feedback loop, i.e. the information is received subsequent to the actual event. It is important to respond fast, for the information to be useful and current.
I discussed with the team members that were going to use my output, i.e. the TrajFollower and the Mapper. Together, we decided to experiment with a one second decision interval.

4.3.1.2 Deciding what terrain types to emphasize

After a meeting with the student responsible for the TrajFollower, the following criteria for evaluating what terrain types that are most important to be able to distinguish between were decided upon:

a) Likeliness that the terrain type will appear
b) Effect that the terrain type will have on the dynamics of the vehicle

In a desert race, sand is obviously a terrain type that fulfills both criteria. Sand is both certain to appear, and sliding affects safety of the vehicle. Other than sand, the following terrain types fulfill the criteria:

- Paved-road
- rocky
- dirt-road
- bumpy

Paved-road is a man built flat asphalt road. Dirt-road is a man built gravel road. Rocky is flat terrain with gravel intermixed with bigger stones. Bumpy is terrain with frequent bumps that can put the vehicle in danger if traversed at high speed.

To analyze the characteristics of the different terrain types, I collected approximately 30 seconds of state data from each terrain. I could not get data on sand, because it is too risky. Moreover, when I collected the data, the speed was not constant.

4.3.2 Analyzing the data

It is not easy to find an equation that describes the different types of terrain.

One solution would be for me to analyze data for relevant terrain types, and for me to set up suitable criteria (or equation) for the different terrain types. After that, the criteria would be plugged in to the computer. Thus, the computer would recognize the terrain type once the criteria were met.

Another solution would be to plug in data for a certain terrain type; let the computer find an equation that the computer finds suitable; and hope that the computer will recognize the data that the computer receives when the vehicle is driving on similar terrain. I tried that by the least square method.

4.3.2.1 Analyzing the data manually

I loaded the data I collected to Matlab. For each state parameter e.g. altitude, roll, and pitch I plotted in both the time domain and the frequency domain. The transform from time domain to frequency domain is easy to do. In Matlab there is a function called fast fourier transform[23] (fft) that transform from time domain to the frequency domain. For each state parameter, I calculated the mean value, the standard deviation and minimum and maximum values for the total area under the fft. The total area under the fft includes all the vehicle vibrations. When I analyzed, I also calculated the same parameters for intervals of the area under the fft. The state data variables that showed to be most useful were pitch, roll, and first- and second derivatives down. I have put together the most important data in the tables below.

Table 4.1

Area under the fft pitch <40				
Terrain type	Mean Value	Standard deviation[24]	Min Value	Max Value
Paved-road	0.33	0.08	0.26	0.43
Dirt-road	1.43	0.86	1.10	2.77
Rocky	2.67	0.59	2.33	2.87
Bumpy	4.03	1.22	1.26	7.47

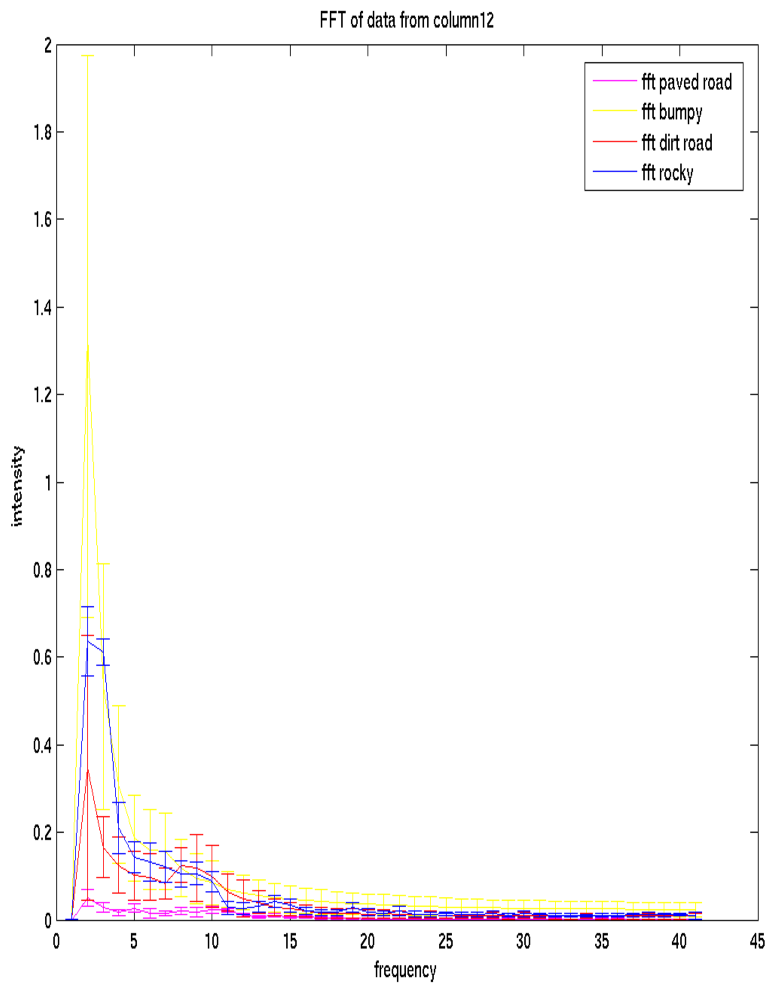
Area under the fft roll for 5-8				
Terrain type	Mean value	Standard deviation	Min Value	Max Value
Rocky	0.95	0.10	0.93	0.95
Dirt-road	0.54	0.50	0.26	1.44

[23] Fast fourier transform is the same transform as fourier transform, it is just another way to calculate that reduces the numbers of calculations.

[24] $s = \sqrt{\frac{\Sigma(x-\overline{x})^2}{n-1}}$

Area under the fft pitch for <3				
Terrain type	Mean value	Standard deviation	Min Value	Max Value
Rocky	1.25	0.33	1.09	1.39
Dirt-road	0.51	0.60	0.11	1.39

Area under the fft down derivatives for <40				
Terrain type	Mean value	Standard deviation	Min Value	Max Value
Rocky	25.74	0.92	25.00	27.10
Dirt-road	18.73	0.63	13.72	34.23



Plot 4.1 I plotted the pitch in the frequency domain. For each second measured, 40 data points from the time domain are transformed through the fft, and results in 40 point in the frequency domain. I collected data for about 30 seconds of each terrain type so each curve in the plot above shows the mean frequency curve of the 30 curves and the standard deviation. The x- axis shows the frequency 0-20 Hz.

It was not possible to separate all the terrain types with certainty. The characteristics of a certain terrain type had a big variance. I believe that the variance to some extent was an effect of speed variations.

I plotted how the speed varies (by calculating √ (north nvelocity^2 + east velocity^2)). Intuitively, the higher the speed, the more vibrations. However, it was difficult to come to this conclusion from the analysis that I made from my data. High speed can explain some of the variance. This is especially obvious on asphalt and dirt road. However, when driving on rocky and bumpy, where the road was varying, driving at low speed could result in high vibrations. This could be an effect of letting a human drive, as humans slow down when the terrain gets rougher.

To isolate the speed effect, I divided the frequency with the speed:

(frequency /speed) =wave/seconds/ (m/s) = wave/meter

The equation above resulted in a worse result compared to not considering the speed.

4.3.2.2 Supervised Learning by the Least Square Method

The idea with supervised learning is to let the computer do the work. Here, I will try this method to separate two terrain types from each other by using the Least Square Method. I choose to start my experiment to try separate paved road (p) and bumpy road (b). These terrain types have the biggest differences of the area under the pitch. By analyzing manually, I could separate them with 100 % accuracy.

The Least Square Method states that:

Min $\mathbf{x}$: $||M\mathbf{x}-\mathbf{y}||$, $\mathbf{x} = (M^TM)^{-1}M^T*\mathbf{y}$

In the matrix M, I read in state data of the pitch in the frequency domain in Matlab. Then I read in a column vector y. For each row in the matrix with paved-road data, I put 1 in the vector y. For each row the matrix with bumpy terrain, I put -1 in the column vector y. Each row in the matrix M, contains data from one second of respective terrain. One second constitutes 40 sampling points. That resulted in the equation:

$$\begin{array}{ccccc}
\mathbf{M[60*40]} & & \mathbf{x\ [1*40]} & & \mathbf{y[1*40]} \\
\begin{matrix}
p_{1,1} & p_{1,2} & \cdots & p_{1,40} \\
p_{2,1} & p_{2,2} & \cdots & p_{2,40} \\
p_{3,1} & p_{3,2} & \cdots & p_{3,40} \\
\vdots & \vdots & . & \vdots \\
b_{58,1} & b_{58,2} & \cdots & b_{58,40} \\
b_{59,1} & b_{59,2} & \cdots & b_{59,40} \\
b_{60,1} & b_{60,2} & \cdots & b_{60,40}
\end{matrix} & * & \begin{matrix}
x_{1,1} \\ x_{1,2} \\ \vdots \\ x_{1,38} \\ x_{1,39} \\ x_{1,40}
\end{matrix} & - & \begin{matrix}
1 \\ 1 \\ \vdots \\ -1 \\ -1 \\ -1
\end{matrix}
\end{array}$$

Interpreting the method, you are searching for the x that for each row in M constitute a linear combination that tend to equal 1 if the row correspond to paved-road and tend to equal -1 if the row correspond to bumpy terrain. In this way, we were hoping to be able to recognize different surfaces just by examining this linear combination.

By calculating $\mathbf{x} = (M^TM)^{-1}M^T*\mathbf{y}$ and saving the vector x, the vector x should than be able to recognize and separate these types of terrain. To try this, I read in a new row from the matrix M, containing paved-road and multiplied with x. If the method would have worked, the result would have been close to 1. I tried this for different rows. The result is shown below:

Table 4.2

Paved-road	x:
M1	-0.017
M2	0.13
M3	0.4132

The computer was unable to separate paved–road from bumpy-road. I realized that it would be difficult to enable the computer to separate some of the terrain types that are more similar. On top of that, I had hoped that the computer (eventually!) would be able to distinguish not only two, but five different terrain types.

4.3.3 Choosing what variables to implement

As shown in the previous chapter, I analyzed the data variables pitch, roll and first- and second derivatives down. However, I chose not to base the implementation on the pitch alone. First of all, pitch and roll were superior to the other indicators. These two indicators constitute a dynamic couple. Therefore, it is sufficient to base the analysis on either roll or pitch. Second, a numeral variable implementation would have been an inappropriately complex solution.

Based on the data I collected and analyzed, the area under the pitch was the best overall indicator to separate terrain types.

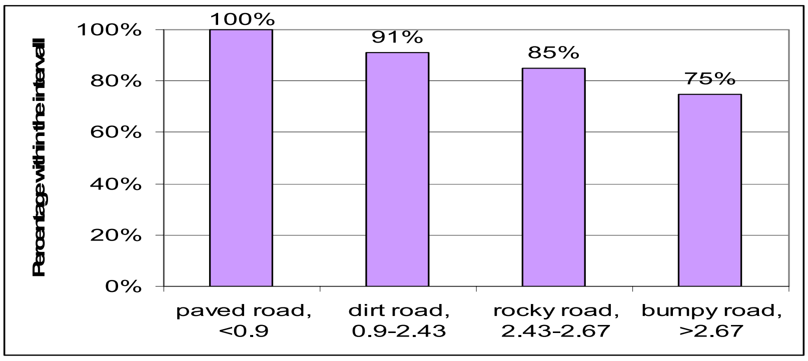


Figure 4.4 Based on the data that I collected and analyzed, the area under the pitch was the best overall indicator to separate terrain types. The area under 0.9 area unit corresponds to paved-road, the area units between 0.9-2.43 corresponds to dirt-road, the area units between 2.43-2.67 corresponds to rocky road and the area units that are bigger than 2.67 corresponds to bumpy-road.

4.4. Implementation

Matlab is a powerful tool for analyzing. However, all code in Alice is programmed in C++. Therefore, I had to translate the code from Matlab to C++.

My prior programming experience primarily involves JAVA. There are obvious similarities between JAVA and C++, as they both are object oriented, which gives a good structure. The main difference is that C++ is faster than JAVA. Speed was the selling point when Team Caltech decided to use C++, as it is highly important to avoid lag.

My program is based on the module RDDF PathGen. I copied all the code from this module, and created a new program for the online terrain classification. It took quite some time to get started with programming, because I had to delete the information in RDDF PathGen that I did not need, and keep the information that is useful for my program. Having cleaned up the program, I compiled and ran the program. RDDF PathGen already contains a state reader that read in the state data to my program. Therefore, I merely had to create a vector with 40 slots. Next step was to transform the state data in my vector to the frequency domain. It would have been impossible to write a function for this transformation. Luckily, there are plenty of online libraries on the internet. I download the online library called the fastest fourier transformation in the west (www.fftw.org). This library helped me to transform a vector with 40 points of state data in the time domain to a vector with 40 points of state data in the frequency domain. Than I sat up different rules that were based on the area under the fft pitch, resulted in the output a number for each terrain type.

The terrain classification was not used at the race, because my client TrajFollower was sufficiently developed to receive the information. At the time, TrajFollower emphasized road following abilities, which have to be reliable before terrain classification can be useful. However it will be useful in the future.

4.5 Solutions and future work

4.5.1 Terrain Classification

The poor results of the least square method surprised me. The computer was unable to separate paved–road from off-road. These two terrain types have distinctively dissimilar frequency curve. I realized that it would be difficult to enable the computer to separate some of the terrain types that are more similar. On top of that, I had hoped that the computer (eventually!) would be able to distinguish not only two, but five different terrain types.

I expected that there would be a difference in frequency between different terrain types. However, all the terrain types capped at 2 Hz. This can be an effect of how the bumpers were configured. The intensity showed a big difference on different terrain types.

The data that the analysis is based on was collected with a human driver. I would expect the data to differ if Alice was driving autonomous. First of all, Alice does not drive in the same manner as a human driver. Second, the speed was not constant when the data was collected. Speed affects the data. Intuitively, I would expect a higher frequency when driving at a higher speed. Within the same terrain type, I would expect a human driver to slow down in more rough areas, and speed up in more flat areas. Therefore, the data might be skewed as an affect of different speed. To get as reliable information as possible, both speed and load has to be considered. When the data was collected, three or four heavy Americans were sitting in the car. This obviously affects the data variables. Intuitively, the more load, the less bumpiness.

For the future, I still believe that autonomous vehicles should take the ground into consideration. If terrain classification facilities are developed, these provide an additional sense for the vehicle. When collecting data, it is important to bear in mind what information that is useful. For example, if the major hazard is sliding, the ability to separate sand from the other terrain types might be sufficient. It is not obvious that gravel has to be separated from rocky, if these two terrain types affect the dynamics of the vehicle similarly. It might be sufficient to enable Alice to separate asphalt from the rest of the terrain types. It all depends on what the information is going to be used for.

4.5.2 Test Matrix

Last year they only tested the vehicle twice. This summer we drove more than 300 miles autonomous, which is a huge progress. However, I still think we need to put more effort to get the testing more efficient.

Team Caltech decided to set a deadline three weeks prior to the race day. Modules that were not working properly at that point were not to be used for the race. The motive for the deadline was to have enough time for integrated testing. When deciding upon what modules to use, the information in the test matrix could have been useful. The test matrix described what modules that had been tested and how these modules had performed. However, the test matrix was not made use of. Instead, the deadline was disregarded, and the decision what modules to use was taken one week prior to the race. To disregard the deadline, a motivation should have been required. However, nobody was required to make such a motivation, and kept working on the modules in disregard of the deadline.

That the deadline was disregarded is even more frustrating knowing that it was quite obvious what modules that were going to be used at the race. Letting people keep working on and test modules that are not going to be used at the race should be avoided for two main reasons. First of all, a decision on what modules to use would have enabled more manpower to work on the modules that were actually going to be used. Second, testing was a bottleneck; therefore testing modules that are not going to be used, reduced the test time available for modules that were going to be used. Testing of such modules also reduced the time available for integrated testing. Such testing is essential to avoid sub optimization.

Team Caltech had problems with sub optimization last year as well. However, the problem has not been addressed. I am not surprised that Team Caltech is struggling with such problems: the team is strongly decentralized and each student can choose to work on whatever module he finds interesting. However, it should be mentioned that Caltech interest in DGC is two folded: developing a superior autonomous vehicle and letting students work and learn the things that each student finds interesting. These goals are not easily combined, and are often conflicting. For example, even though Team Caltech had a fully functional planner, one of the students started to develop a new planner using a different approach. At one point, Team Caltech had four different planners. This kind of work was a waste of time, given that other modules were in screaming need of assistance.

Another problem with the testing has been described in the chapter 3.2. In the chapter, I critizied the testing for not being sufficiently effective and efficient. At Caltech, the manager was responsible for making sure that the modules that were to be tested were in fact ready to be tested. However, the testing decisions were made ad hoc, and not rational decisions based on the available information. To solve this problem, each test group should have been assigned lower level managers; e.g. packing list manager, module running manager and test driver manager. Utilizing this kind of decentralized management, the test managers would have been relieved of some duties, and enabled to spend more time deciding what modules to test.

The test matrix was often disregarded. Some of the test managers did not recognize the importance of documentation. These test managers did not even respond when I asked them to provide the necessary information. When I discussed the issue with these managers, they explained that it is a waste of time to test Alice in rough terrain before having tested sufficiently on easy dirt road. However, as I explained to them, testing in rough terrain was not the issue; rather, Alice could have been tested on pavement at Santa Anita 15 minutes from Caltech instead of at dirt-road at Stoddard Wells 2 hours away. Decisions could have been made. Generally speaking, I wanted the test managers to gather information, whereupon rational decision could be taken.

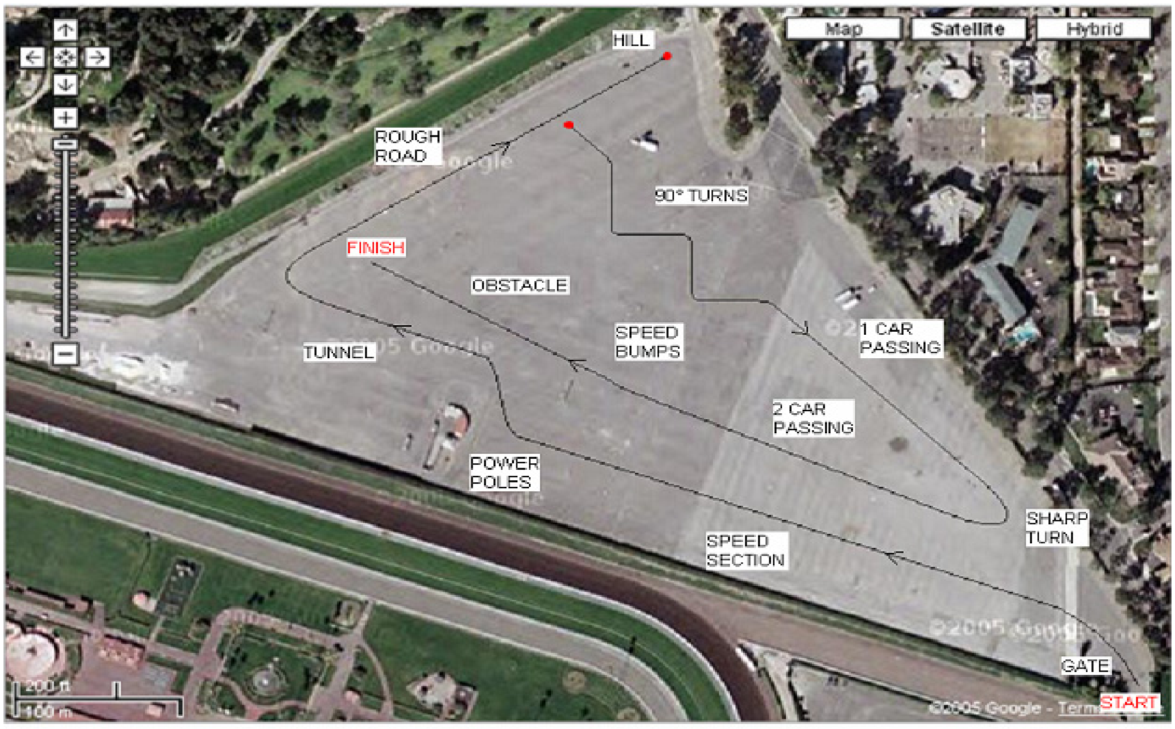
5 Team Caltech and Our Way through the DGC

The first step to the race is to pass the National Qualifying Events (NQE). A great deal of preparations needs to be done both for the NQE and the race. I helped to prepare for the NQE. This chapter describes the preparations for the NQE, and how Alice performed in the NQE and the final race.

5.1 National Qualifying Events

August 23, DARPA announced the NQE course. As a member of Team Caltech's NQE set-up team, I set up a mock NQE course at Santa Anita. This involves planning the route, making an RDDF file of the course, building obstacles and moving the obstacles to and from Santa Anita.

Picture 5.1 I am watching Alice in the middle of the night. Alice needs to look her best for the NQE.



Picture 5.2 The mock NQE course that I and Laura Fishman planned at Santa Anita

In the first run at the NQE, Alice was somewhat out of shape. Alice ran into the hay balls, than she lost her GPS in the tunnel and got stuck. Having fixed the bugs, Alice did very well at the rest of NQE and was finally ranked among the top 10 teams.

This year, 13 teams completed the course perfectly, which is an improvement compared to last year when only 2 teams completed the NQE course. There were a lot of different vehicles that competed in the NQE. There were vehicles with 8 wheels and there was also an autonomous motorbike. 23 teams qualified to the final race.

Picture 5.3 US Berkeley's bike in the semifinals. The vehicle next to him is TerraMax, one of the biggest vehicles in the competition.

5.2 The race day

The final race was held the October 8. Years of hard work and lots of money invested was at stake. This was the day we hoped to get paid for all the hard work: becoming famous and 2 million dollars richer...

Picture 5.4 Caltech showing there pride, it says GO ALICE on there back, and Caltech on the stomach.

6 am the first vehicle took off. Alice started at 10 am. Having completed just about 8 miles of autonomous driving, Alice passed under a power-line. This power-line caused a GPS-error. Without the GPS, Alice did not get information about the corridor that defined her route. By that time, Alice had also lost two LADAR units, and was therefore limited in her terrain sensing capabilities. With a bit of bad luck, Alice was near by concrete barriers at the same time as the GPS error occurred. Alice managed to force the concrete barriers and drove straight at the media that was covering DGC (pic. 5.4).

Here is a stepwise description of Alice's race[25]:

- Four minutes into the race, the two mid-range LADAR units stopped functioning. The reason is still not known, but does not appear to be a previously seen failure mode (which we detect and reset). The remaining LADAR units where the long range (35 meter) and short range (3 meter).
- Coming onto the straightaway next to the berm, Alice passed under a set of power lines that disrupted its GPS signal

[25] http://team.caltech.edu/

- The software properly recognized the condition and stopped to let Alice reacquire the signal. This was triggered by the GPS-reported position being more than 2 meters off of our current (IMU-based estimate).
- The GPS receiver reacquired the signal, but with very high error estimates (a condition we had never seen before). The covariance weight that we used for fusing the measurement was consequently high and this caused a low convergence of the state estimate to the true position.
- The slow convergence of the position caused the state estimation module (astate) to believe that the filter had converged, and so we indicated to the supervisory controller (superCon) that it was safe to begin moving.
- Upon receiving the command to begin moving, the supervisory controller cleared the map of the terrain in front of it since that map was taken from an unknown position (due to the large error in detected position). At this point, the long range LADAR was pointed above the concrete barriers that we would eventually hit and the state estimate was not yet converged.
- Alice began moving forward, pointing straight down what it thought was the corridor and accelerating to approximately 10 mph.
- The short range LADAR saw the concrete barriers, but did not have time to stop the vehicle before we went over the barrier (still on course, according to Alice)
- DARPA e-stop paused the vehicle (right in front of the media)

Picture 5.5 Alice last seconds in the DGC

Bibliography

Web pages

http://wikipedia.org/wiki/DARPA_Grand_Challenge

http://team.caltech.edu/

www.darpa.com

Litterature:

Brooks, Ignemma, Kart, Dubowsky, "Vibration-based Terrain Analysis for Mobile Robots" ICRA 2005

Hägglund, Tore: Reglerteknik (2001) Lund

Kjellander, Henrik Arbiter and Simulations for Team Caltech in the DARPA Grand Challenge

Kogan Dimitri: Real Time Path Planning Via Non Linear Optimzation Methods (2005), Pasadena

Spanne, Sven: Konkret analys (1997) Lund