IT Systems Engineering
Universität Potsdam

Bachelor's Thesis

Racing Line Detection - Recording and Comparison of Racing Lines

Ideallinienerkennung - Aufnahme und Vergleich von Ideallinien

by

Tim Oesterreich

Potsdam, June 2016

Supervisor

Prof. Dr. Christoph Meinel,
Philipp Berger, Patrick Hennig

Internet-Technologies and Systems Group

Disclaimer

I certify that the material contained in this dissertation is my own work and does not contain significant portions of unreferenced or unacknowledged material. I also warrant that the above statement applies to the implementation of the project and all associated documentation.

Hiermit versichere ich, dass diese Arbeit selbständig verfasst wurde und dass keine anderen Quellen und Hilfsmittel als die angegebenen benutzt wurden. Diese Aussage trifft auch für alle Implementierungen und Dokumentationen im Rahmen dieses Projektes zu.

Potsdam, July 7, 2016

(Tim Oesterreich)

Kurzfassung

deutsche Zusammenfassung...

Abstract

//TODO

Motor racing is all about getting around a track as fast as possible. One of the most important and most driver-influenced steps in achieving best times is the racing line, escpecially hitting the ideal racing line through corners. Our race analysis system collects different car-based data to evaluate and compare the driving style of drivers which helps them to gain an advantage over the competitors.

This paper especially focuses on the recording and comparison of racing lines, using different methods, like Visual Odometry and lane detection and use the extracted data to compare and analyse the driving behaviour of different drivers.

Contents

1. Introduction

1.1. Motivation

The trajectory around a track that allows a given vehicle to traverse the circuit in the minimum amount of time is called the ideal racing line. Needless to say, in a race this is the path every driver wants to take. In reality, however, this isn't an easy task. In order to achieve best times, a racing driver has to remember exactly which corner of the track comes next, how fast they can take this corner, see how the car is positioned on the track and process many more information.

1.2. Project Scope

1.3. Problem Introduction/Description/Context

2. Related Work

2.1. Racing Line calculation in Video Games

- vamos racing simulator

3. Algorithm/Concept

As this Bachelor Thesis tackles two problems, the algorithms can be devided into two categories as well. The first category consists of racing line recording concepts and the second one of ways to compare these recordings.

3.1. Recording

3.1.1. Visual Odometry

Visual Odometry (VO) is a concept that most camera-based robots use for navigation. It uses two consecutive camera frames to calculate the rotation and translation of the camera between them. The advantage over traditional tracking systems like GPS is, that it isn't dependent on any external sources, like satellites or radio towers, to determine the position of the object. Also it is capable of much higher polling rates than most GPS receivers, which normally poll at 1 Hz, so they get 1 positional update per second. The potential speed of Visual Odometry is linked to the framerate of the camera. That way, a camera that records at 30 frames per second enables Visual Odometry to estimate a position 30 times a second. The crude algorithm behind VO determines and stores recognisable features in one frame, using a feature detection algorithm. At first we need two consecutive, recitifed camera images. It is important that the images are rectified, because otherwise the edges of the image would be curved and produce wrong results. Our soulution uses the Harris corner detection method for determining features. Finding corners in an image is a very important prerequisite for the final position determination, because it allows us to calculate direction vector to another point. This other point being the same feature in the following camera frame, where the Harris detector is used, as well. Having two sets of corners from either image we can track each feature from one frame to another.

- Lucas-Kanade?
- Jacobian?

Now that we have an end point to every (possible) starting point, we can construct a set of directional vectors. Removing outliers (i.e. vectors that are much longer or go in a vastly different direction than most vectors) an approximation translation and rotation the camera conducted during the two frames can be calculated. Outlier removal is important, because otherwise the movement of other cars on the track would cause the algorithm to think the camera was moving in a different direction than it actually was. If the taken path includes loops it is possible to optimize the result by using loop detection. If any features are redetected at a later point in time and the current position is close to the position the features were redetected, the path can be streched in a way that it connects to the earlier path. This can reduce errors that accumulated due to small estimation errors in the previous steps.

3.1.2. Lane Detection

Lane Detection uses markings on a road to determine its lateral boundaries in realtion to the recording camera. It is often used in autonomous driving and driving assistance systems, because the position of the car in between lane markings gives a lot of information about the direction the car is traveling. In our case, not only can we determine a traveling direction, but also the position on the lane. By tracking the distance to the left and right lane, a deviation to the middle of the lane can be calculated. If the driven track is know or the camera is capable of determining a scale (e.g. by using a stereo camera), the deviation can even be expressed as a metric length, which is helpful for extracting additional information, like speed and acceleration, later on.

- possibility to improve recording results on known track
- possibility to determine accurate horizontal position on track

3.1.3. Recording OBD2 Data

Definition OBD II (On-board diagnostics 2) is an interface which all cars build after 1996 in the USA or after 2003 in the EU, respectively, have to have built-

in. It implements the SAE J1962 standard which provides standardized PIDs that return specific car-based sensor data, like the current speed, current motor revolution (rpm), throttle position and more. A data request is possible about 10 times per second. This gives us a good basis for further detailed comparisons combined with the racing line recording.

3.2. Comparing

3.2.1. Visual Comparison of Racing Line with help of sensor data

A fairly simple, but effective way to find out where time is lost on the track is the visual comparison of racing lines from different drivers. The idea is to overlay the previously recorded lines over the race track. To actually determine at which point a driver was faster than the other one, we need an obvious optical representation of the speed at any given point on the track and the possibility to receive further information on demand. In our system, the speed is represented by coloring in the racing line on a gradient scale reaching from green (standing still) to red (>250 km/h). Besides speed data, different values, like acceleration, brakeing behaviour or lap time, can be displayed in a similar fashion. That way the fastest line for a given corner can be determined and the slower driver can find out, why their line was inferior.

- overlay racing lines with different lap times
- highlight relevant locations, like corners, showing detailed information about throttle and brake behaviour and curvature through the corner
- fastest way through corner is tradeoff between distance travelled and curvature, with low curvature allowing the driver to carry more speed to exit of the corner

3.2.2. Mathematical Calculation of ideal line given a certain track

Besides comparing two driver-generated racing lines, it is also possible to calculate a fast, almost ideal racing line.

- calculating smooth curves around an edged representation of the track using Bézier curves
- naively not an optimized track, but fair approximation
- simulating a chain of springloaded masses -> ideal line calculated by itterating multiple times and taking the line that uses the least force to keep masses in bounds of track (//this seems quite complex, not really sure how this works yet)

3.2.3. Mathematical Comparison of two paths

- convert pair of consecutive points into list of vectors; each vector is distance between points and angle to x-axis
  - double dx = endPoint.X - startPoint.X;
  - double dy = endPoint.Y - startPoint.Y;
  - double magnitude = Math.Sqrt((dx * dx) + (dy * dy));
  - double direction = Math.Atan2(dy, dx) * (180 / Math.PI);
- combine vectors that have the same direction, by generating a new vector with the direction of the old ones and the sum of their lenghts. This indicates a straight, which is not that interesting for comparison
- Now we can compare the angle for each section of a corner

4. Implementation

Most of the image processing steps were implemented using OpenCV, an open-source C++ computer vision library.

4.1. Recording

4.1.1. Visual Odometry

4.1.2. Sensordata

The OBD-II dongle uses a bluetooth connection to communicate with our analysis software. Via this connection we can open a serial port to write and read data from. The ELM 327 standard determines the formatting of the request and of the return value. The request consists of the required mode and a Parameter ID (PID), which stands for one specific data value. In mode 1 the dongle returns the current values, which is what we need. Other modes, for example, return the values since the last engine failure or information about the car.

4.2. Comparison

4.2.1. Google Maps

- calculate GPS coordinates from VO points
- Polyline Overlay
- hidden Polyline for calculations and hover function vs visible, colored Polyline

5. Evaluation

5.1. Visual Odometry to record Racing Lines

- Pro:
  - Many Datapoints (in theory up to Framerate positions per second)
  - Capable of detecting fairly slight movements
  - No need for external sensors (apart from camera which already exists)
  - works in every environment (especially where there is no or poor GPS signal)
- Con:
  - uses fairly expensive operations
  - in praxis mobile-range computers will only reach up to 10 fps
  - no constant speed (slows down in areas with many feature points (like forests))
  - fairly error prone
  - rounding errors and inaccurate calculations add up and distort the racing line

5.2. Usefulness of Racing Line Comparison

- professional drivers can achieve similar lap times with very different racing lines
- however amateur and intermediate drivers can learn a lot from more skilled drivers

6. Future Work

6.1. Open Research Questions

6.2. Extensions

7. Conclusion

References

A. Appendix