# Fault Tolerant Control and Localization for Autonomous Driving: Systems and Architecture

*ByungHyun Shin*

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2016-83
http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-83.html

May 13, 2016

Acknowledgement

Thank you to Professor Francesco Borrelli and Ashwin Carvalho for their support and guidance throughout this project. The consistent feedback was key to building and achieving what we wanted to. Thank you to Professor Trevor Darrell for advising the project and reading the report. I am also grateful for the help of Amy Lee and Alexandre Beliaev in their support throughout the writing process. And of course thank you to the UC Berkeley EECS staff and Master of Engineering staff and instructors for this past year.

# Fault Tolerant Control and Localization for Autonomous Driving: Systems and Architecture

by

ByungHyun Shin

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

UNIVERSITY OF CALIFORNIA AT BERKELEY

May 2016



Author .....
Department of Electrical Engineering and Computer Science
May 13, 2016

Certified by .....
Francesco Borelli
Associate Professor
Thesis Supervisor

Certified by .....
Trevor Darrell
Professor
Thesis Supervisor

Accepted by .....

**Fault Tolerant Control and Localization for Autonomous Driving: Systems and Architecture**

by

ByungHyun Shin

Submitted to the Department of Electrical Engineering and Computer Science
on May 13, 2016, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

## Abstract

Fault tolerant localization is a critical component of autonomous driving technology. In order to perform any control and perception of objects around its environment, a vehicle must possess an accurate estimate of its own location in the environment despite potential sensor failures. This paper describes the system and architecture developed to peform fault tolerant localization on a prototype test vehicle using a differential GPS and INS coupled system, a camera, and laser rangefinder sensors. A ROS-based system was developed to allow modularization and interfacing with other modules such as perception and control. The system permits parameter tuning and automated testing for both software-in-the-loop and hardware-in-the-loop, as well as interchanging of different algorithms for the localization itself. The framework for developing and testing new localization algorithms has been built in order to test a particle-filter based implementation that provides an accurate estimate of the vehicle's location despite failures in GPS, camera, or laser rangefinder.

Thesis Supervisor: Francesco Borelli
Title: Associate Professor

Thesis Supervisor: Trevor Darrell
Title: Professor

# Acknowledgments

Thank you to Professor Francesco Borrelli and Ashwin Carvalho for their support and guidance throughout this project. The consistent feedback was key to building and achieving what we wanted to. Thank you to Professor Trevor Darrell for advising the project and reading the report. I am also grateful for the help of Amy Lee and Alexandre Beliaev in their support throughout the writing process. And of course thank you to the UC Berkeley EECS staff and Master of Engineering staff and instructors for this past year.

# Contents

# Chapter 1

# Introduction

The problem of localization is essential to solve within the context of autonomous driving in order to realize widespread adoption of self-driving vehicles. One of the many tasks that an autonomous vehicle must constantly execute, estimating its precise position relative to its surroundings relies on various sensor measurements and signal processing techniques. Furthermore, to achieve widespread adoption, localization must be reliable and sufficiently accurate in the presence of potential sensor failures or faults. As one of three capstone teams in the umbrella project of fault tolerant control in autonomous driving, the localization team worked to develop the methods and systems an embedded computer running on a vehicle can execute to achieve this task in real time. In addition to developing new algorithms to achieve centimeter-level accuracy that is robust against sensor failures and missing information, the team has built the modularized software infrastructure that can run these techniques on the vehicle's embedded computer.

Several methods exist to perform localization for self-driving vehicles, as this is an active area of research in the robotics community. Offline map-building using cameras and/or laser rangefinder sensors followed by online localization against the map is an increasingly common approach, although often requiring expensive sensors and computational resources [8], [7], [1], [17]. Extensive progress has been made in the field of simultaneous localization and mapping (SLAM), often using just one or two cameras and/or a laser rangefinder sensor [19], [20], [16]. While this project did

Figure 1-1: The prototype test vehicle was a Hyundai Grandeur. It is equipped with a camera, GPS/INS system, six laser rangefinders, and radar sensors.

not implement these methods which require unhindered access to a camera and its images, future work building upon this project may utilize these newer methods with an additional camera sensor to achieve higher performance. Our approach utilizes a combination of an RTK GPS/INS system, an out-of-the-box camera system that produces polynomial estimates of lane markings in front of the vehicle, and Ibeo Lux laser rangefinder sensors and an iterative closest points (ICP) matching algorithm [12], [13].

The first part of this paper briefly describes the methodology and overall system built by the team and how the various aspects fit together for running real-time on a vehicle (Fig. 1-1).

In addition, this paper describes in detail the author's technical individual contribution, namely the software architecture that was designed and implemented to run our localization algorithm as well as the infrastructure built to test the system. Our capstone team work breakdown consists of the following: theoretical development of a global positioning system (GPS) localization, theoretical development of a particle filter using both GPS and lane information from camera measurements, an exploration of laser rangefinder localization methods and implementation of the iterative closest point algorithm, and a development of the software architecture that runs these algo-

rithms. Our project involved several parallel processes in which we developed, tested, and refined localization algorithms in one programming language, while also building software that implemented the algorithm and was ready for deployment on a vehicle. For a more theoretical treatment of developing the localization algorithms, see Wu [18] and Schindler [15]. Leo Li's paper describes in detail an additional algorithm that we are used to improve our algorithm using laser rangefinder sensors, and also explains the data interfacing required for testing in simulation and in real time [9]. This paper explores how the software to actually run these algorithms on an embedded computer was designed and built, including the Robot Operating System (ROS) architecture and testing infrastructure.

## 1.1 Project Specifications

Before starting the design process, several requirements were established to understand the goals and metrics to be used in evaluating the project and for making design decisions. The first and primary goal of this capstone project was to develop localization software that could actually be deployed on the test vehicle, which could produce an estimate of the vehicle's location readily accessible by other modules of the autonomous driving platform and which was accurate enough to use with a controller that could then perform path planning. Second, the goal of this project was to develop this localization in a fault tolerant manner able to handle failures in sensors. Even with failures of GPS or camera for instance, the vehicle should be able to produce a decent estimate of its position. While a specific quantitative target was not set for the localization accuracy, an accuracy target on the order of decimeters was good for a localization estimate that could interact with other modules of the deployable autonomous driving platform.

## 1.2 Methodology

A particle filter algorithm was developed to perform localization. By using position and heading information from a coupled differential GPS/INS unit, lane information from a front-facing camera on the car, and a fused point cloud of the surrounding environment using six laser rangefinder sensors located on the vehicle, a global estimate of the vehicle's position and heading are produced. The process model uses a hybrid kinematic bicycle/dynamic car model which uses the best model based on the current driving, e.g. high speed turns vs. low speed straight driving. The implementation of the particle filter and process model are discussed in [18] and [15].

# Chapter 2

# Individual Technical Contribution

## 2.1 Software Architecture

Building upon preliminarily developed algorithms in MATLAB using just global positioning satellite (GPS) data, Wu and Schindler have tested new algorithms that use additional sensor information and are able to provide accurate enough estimates despite sensors that may fail for period of time, or that provide low-quality information. They has developed an algorithm that uses GPS data as well as visual camera measurements of lane markings to output an estimate of the car's real-time location. Using MATLAB allows faster development and iterations of the new techniques, leveraging some previously built infrastructure of the laboratory that allows for easy testing, evaluation, and visualization of results.

The software infrastructure needed to deploy the autonomous driving technology was designed and built to run on Robot Operating System (ROS).

ROS is often referred to as a "meta operating-system" - while it is different from the functions of a traditional operating system such as Microsoft Windows, it is a framework that allows various tools and libraries to be used in an easy-to-develop manner for application to robotics. Using ROS, "nodes" can be built which, while running their own separate processes and calculations, are provided standardized methods of communication and interaction amongst themselves.

The capstone team has been building the ROS infrastructure that is needed to

implement, test, and ultimately run the new localization algorithms that the MATLAB sub-team develops. While the laboratory already owns MATLAB code that can unpack test data to the correct format, read it in, feed it to the localization algorithm, record output data, and evaluate the performance, any such analogous functionality must be built from scratch in ROS. Leo Li's paper describes our team's work in developing part of the ROS architecture that loads (post-test) data output by the many sensors on the car, picks the relevant information needed by the localization algorithms, and sends this to the correct programs that need it. He also explains the process of translating the GPS algorithm MATLAB code to a Python program, which may be called in a ROS environment to be actually used. In addition to the data interfacing and translating of the original GPS algorithm, our team has been able to decide upon, and implement, a structure of ROS programs which will be able to take as inputs GPS and camera sensor data, and publish the estimated position of the car to whatever other programs in ROS may need the information. This may include perception modules that attempt to identify various moving objects around the vehicle, and control modules that use the location to plan specific maneuvers and send commands to actuators (steering wheel, acceleration pedal).

Because ROS provides several ways that nodes can communicate with each other, our team first decided on how our localization module could communicate within itself and interface with outside modules. One method of asynchronous communication is the publisher/subscriber method: one node "publishes" information to a certain "topic," and another, separate node can then "subscribe" to the same "topic" to receive the first node's output information. This method requires that the subscriber node constantly receive any and all information that is published on the topic, however frequent (or infrequent) this may be. The other method that our team considered was the server/client model: one "server" node holds relevant information, and is in a "sleep" mode that, until called upon, does nothing. A "client" node can then make a request to the server requesting certain information, at which point the server wakes up, fulfills the information request, and then returns to the idle, sleeping state. The advantage of this method is that, unlike the publisher/subscriber model, this is an

on-demand model that provides information to nodes that need it when they need it, instead of having a constant flow of published information. After further research and preliminary implementations of both systems with simple published data, our team decided to implement the publisher/subscriber model to implement localization. While the on-demand model is desirable, there appeared to be a high overhead cost of servers fulfilling a client's request. This meant that, as more and more clients (perception modules, control modules, etc.) make requests of the server that contains the current location information of the car, the server would take longer to respond, resulting in irregular, and possibly fatally slow, responses. The publisher/subscriber model allows the publication of location information at a fixed sample rate, guaranteeing the timely arrival of location information to any and all nodes that need it.

The publisher/subscriber model is ideal for embedded systems that contain several modules that may need to communicate, but by virtue of hardware differences cannot be synchronized against a clock like traditional threads. The GPS, for instance, may record measurements at highly variable frequencies, depending on whether it can capture a good enough signal. The camera and laser range finders, on the other hand, though running at a fixed rate, may run at two different frequencies that cannot easily be up/down-sampled to match each other. Furthermore, in the case of the laser range finder sensors, data can sometimes be delayed as it communicates over a difference protocol (Ethernet) than all other sensors. By allowing publisher nodes to publish messages containing sensor readings or data calculations as they become available, subscribers listening for such messages can process them as they come without trouble.

This model also allows communication between disjoint modules. The localization module is one of many modules necessary to achieve autonomous or semi-autonomous driving. A perception module uses data published from the localization module to classify static and dynamic objects, a path planning module needs to subscribe to the location to plan a trajectory and corresponding actions to actuators (gas pedal/brake, steering), and in the future, modules may be needed for vehicle-to-vehicle communi-

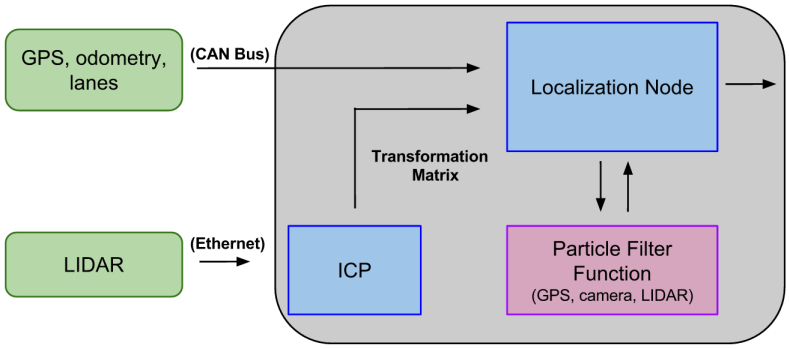


Figure 2-1: The ROS architecture.

cation. ROS provides the framework for these distributed systems to be built, abstracting away specific hardware configurations. We implemented a localization node which subscribes to the data interfacing node described in detail in [9]. The data interfacing reads in historical test data and feeds our localization node GPS and lane measurement data, simulating how data would come in during real-time autonomous vehicle driving tests. The localization node takes the data, calls the particle filter program translated from MATLAB [18],[15], and then publishes the calculated location information to the "location"' topic. This program, like the other programs developed in ROS, is then able to run on the vehicle's embedded computer, allowing real tests to be run. Our localization node can be configured to run at a fixed frequency, able to be changed by the user as a parameter [15], and which determines the rate at which the particle filter runs. This means that if sensor measurements are received by the localization node at a faster rate than it is running the particle filter, then it only keeps the latest measurements received and throws away old measurements. Because sensors generally run at a much higher rate than the particle filter needs to run, this is necessary to prevent accumulation of massive amounts of data that are not needed.

By incorporating laser rangefinder data into our algorithm, we can further provide for reliability of the localization module while improving its overall accuracy. Because of the flexibility of our ROS architecture, a complete redesign of our nodes and the way they communicate is unnecessary. The laser rangefinders collectively output data that can be sent in the same manner as previous sensor measurements: packaged in a ROS message and sent over a specific topic. Scan matching is performed in real

time using an iterative closest point approach [9]. A separate ROS node is developed to subscribe to the laser rangefinder data, which comes in the form of a collection of three dimensional points (a point cloud), which calculates the estimated movement between two time steps. This estimated change in pose is then published and received by our original localization node. The particle filter is then called, inputting this change in vehicle pose along with the GPS and lane measurements. While the implementation of the particle filter needs to be changed, instead of fundamental changes to the algorithm, the new sensor measurements are easily integrated, compared to the process model of how much the algorithm believes the vehicle should have moved between scans, and updating each particle accordingly (Fig. 2-1). While all nodes for interacting in real-time with the laser rangefinder data has been implemented, the performance of the algorithm using all three sensors had not been tested at the time of writing.

### 2.1.1 Fault Tolerance

In order to accommodate for the faulty sensors [18], the localization node passes in to the specific algorithm flags along with the sensor data to indicate whether a particular sensor's readings have not come in. This allows partial data to be sent to the particle filter [15]. If, for instance in the case of the merged particle filter, GPS measurements have not arrived for a few seconds, then the localization node can only give to the particle filter lane measurements. While this may result in a worse estimate than having had GPS measurements available, it is far better than waiting for the next GPS measurement to send all data at once.

## 2.2 Testing

### 2.2.1 Methodology

After development of the algorithms in MATLAB, testing was further conducted in three parts. In order to ensure proper translation of all algorithms into Python

for compatibility with ROS, the algorithms were tested with the same data used to develop and refine the algorithms using Python test scripts. Further, the overall software running in ROS was tested by simulating online measurements using prerecorded historical data. After continual improvements with both methods of testing, field tests were conducted to perform online localization in real time running on the actual vehicle.

While development and tuning of the localization algorithm was done in MATLAB, testing after translation of the code into a Python program was needed to ensure that all aspects of the algorithm were preserved while undergoing changes in data structure and use of libraries specific to Python. While the stochastic nature of the algorithm prevented a deterministic comparison of output values of the algorithm between MATLAB and Python, a test script was developed that ran through several historical datasets of vehicle test runs and recorded the difference between the algorithm's estimation of the location and the continuous differential GPS measurement of the vehicle location. By comparing the first order statistics of this error with those of the MATLAB output, a high degree of confidence was established in the accuracy of the Python algorithm with respect to the MATLAB code.

Before conducting field tests, software-in-the-loop testing is used to evaluate the performance of the particle filter and make necessary improvements. This allows for fast iterations of inexpensive tests to ensure the fault tolerance and safety of the software without needing to conduct actual online tests with the vehicle. In order to perform software in the loop testing, additional ROS nodes and scripts were developed that simulates real-time data. By publishing historical data on the same topic used by the ROS node performing the state estimation, past test runs can effectively be run in almost the exact same manner as it had been actually produced when running the vehicle. For every test run of the vehicle, all sensor data is recorded in a standardized format. A node was built to read this historical data and publish the GPS pose information, lane coefficients, and other necessary data (i.e. odometry) one time step at a time on the correct topics. This allows for automated testing with no modification of the main localization algorithm node, which only sees incoming data in real time

and cannot distinguish whether it is historical data being replayed or real time data.

While the CAN bus provides most sensor data collected by the vehicle, including the GPS estimated pose, INS odometry, and the lane polynomial coefficents from the camera, the LIDAR scans are sent over a separate bus via Ethernet. Two separate programs were created to read the CAN bus historical data and publish to the relevant topics, and to read the LIDAR historical data file and publish point cloud scans to its respective topic. These separate simulator nodes allow testing of the algorithm using real situations, allowing for a wide variety of past situations to be tested and reviewed without field tests.

The final localization algorithm requires over thirty parameters to be set. Several of these are arguments that may be modified by the user (the tester) to fit his/her testing needs. These include the type of model used to predict the vehicle dynamics, the different sensors that may be used, and the frequency at which estimation should run. Other parameters describe specifically the parameters used for algorithm and which have been tuned using past test run data. These include the variance of the noise of every sensor as well as constants and parameters governing the actual dynamics models. These parameters are all stored on the ROS parameter server, which means that all ROS nodes running on the embedded systems may access these values. In the future, this may be useful to change tuned parameters in real-time as the algorithms "learn" the parameter values best tuned for the particular real-time driving situation it is in.

Hardware in the loop The final step of testing involves hardware in the loop. By running actual field tests on the vehicle running our localization algorithm, the performance and robustness can be evaluated to make sure that the algorithm can handle completely new situations.

### 2.2.2 Results

Extensive software in the loop testing was performed on several different vehicle tracks using historical data. While one problem is the lack of a "ground truth" measurement with which to compare our location estimate, the differential GPS signal is used to

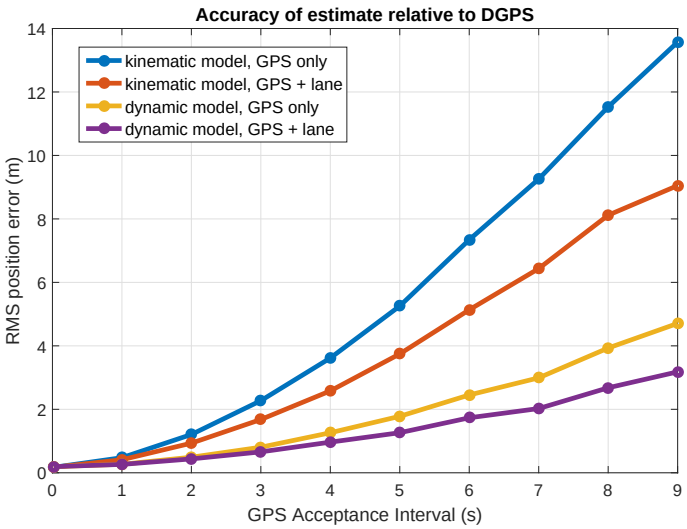


Figure 2-2: The average RMS error of the position estimate relative to differential GPS measurements. We can see that the algorithm using the dynamic car model and multiple sensors has the best accuracy.

measure error. For the purposes of testing fault tolerance, this comparison against a continuous GPS was enough to determine how well the algorithm was performing. By parameterizing the rate at which the algorithm accepts GPS measurements, we are able to demonstrate fault tolerance of the localization algorithm. Figure 2-2 shows the performance of the algorithm. The average RMS difference between the estimate and the measured differential GPS signal diverges the longer the GPS signal is not acquired - by only using an internal dynamics model and lane information, the estimate of its global position is off. However, for periods of time where GPS is accepted every two seconds or less, the error of the algorithm is on the order of decimeters.

While the computational limits of our algorithm were not tested on the embedded system, it was found that the full system was able to run in real time with all sensors active at 50 Hz with 100 particles. In particular, there was no trouble in processing the LIDAR point clouds in real time and estimating the transformation between scans with the ICP algorithm.

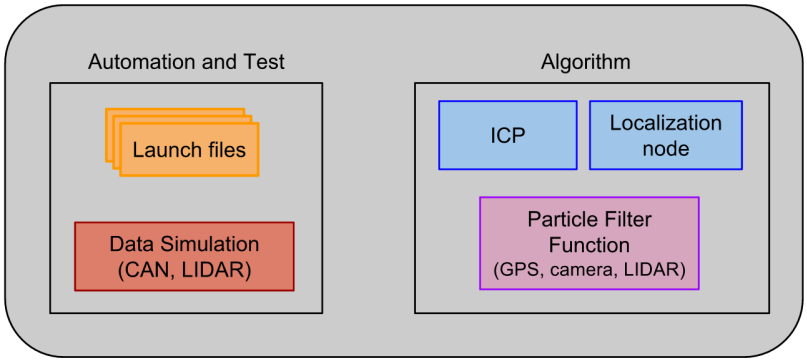


Figure 2-3: The final software package developed by the capstone team consists of the algorithm and the programs needed to run it, as well as the testing and automation files that allow parametrization and automatic testing.

## 2.3 Discussion

This project set out to build a fault tolerant localization software package that could be deployed on a test vehicle and produce as accurate an estimate of the vehicle's state as possible. In order to properly test control techniques and validate other research at the Model Predictive Control Lab, a functioning localization module was needed that could interface with various functional modules of the autonomous driving system. Furthermore, the goal of fault tolerance was a focus for the umbrella project that included perception and simulation. In order to build an autonomous driving system that was robust against potential sensor or actuator failures, the localization capstone team built a system that could produce an accurate estimate even when the GPS, camera, or laser rangefinder sensors went down. The software package that was built for use by the lab is a highly modular component that can communicate with other modules and which can be run in an automated fashion with user parametrization (Fig. 2-3). This will allow future work to build on top of this project instead of starting from scratch.

Future work can utilize the infrastructure built by this project to improve localization for autonomous driving. One direction to pursue is the improvement of the algorithm by taking advantage of the laser rangefinders. While the algorithm currently makes use of the sensors by scan matching and using a noisy estimate of the displacement of the vehicle between scans, it can be improved to utilize offline maps

that are created from historical laser rangefinder runs. By building a point cloud map of the environment prior to online testing, and then by comparing online rangefinder data to the stored map, the localization algorithm performance has the potential to drastically improve. Furthermore, utilizing additional sensors on board the vehicle, such as radar, to improve the fault tolerant nature of the system will help provide an accurate estimate in many different situations. Additionally, integration with other modules of the autonomous driving platform developed in the lab will not only be useful for full realization of autonomous driving but also for finding additional faults and failure cases of the localization platform. Furthermore, as more field tests are performed in a variety of different driving scenarios, increasingly automated testing infrastructure may greatly help the performance of the localization module. Being able to perform many software in the loop tests automatically could pave the way for learning of the many parameters governing the algorithm.

## 2.4 Conclusion

The Localization team has successfully implemented a software module to estimate a vehicle's pose using GPS/INS, camera, and LIDAR sensors. A particle filter-based algorithm was used to combine sensor measurements and provide robust performance with inconsistent or infrequent GPS information. The software was built on the ROS platform, allowing for simple integration with other modules that require pose information.

While further testing and work is required to build an extremely high performance localization method, this project successfully built a ROS-based system that runs on the actual vehicle, and which can interact with other modules to provide an accurate estimate of the vehicle current location at all times. By building a redundant system with multiple sensors, the capstone team was able to account for reasonable failures in the system. With the software and testing infrastructure in place, this work can be built upon to improve the algorithm as well as to build other modules of the autonomous driving platform that require this first step of localization.

# Chapter 3

# Project Context

As the automobile industry faces disruptive changes and challenges from autonomous driving, several contextual issues may inform the decisions that players developing self-driving technology need to make. While our product, localization software, is just one segment of the autonomous driving technology package, understanding the broader industry and surrounding social issues not only helps drive its development but clarifies ways to commercialize and realize value for society. This paper outlines the state of the automobile industry and how autonomous driving technology may fit in, the need to adopt technology strategies to break into and establish oneself in the market, and suggestions for navigating the difficult and growing ethical dilemmas facing the industry. Examining the current automobile industry can inform how well-positioned autonomous driving technology is to enter it, and how profitable the industry may be. The automobile manufacturing industry is a mature industry with a steady growth average at 3.5% per year projected over the next 10 years [14]. The main resources, suppliers, for car manufacturers are: auto parts and accessories, automobile engine parts, iron and steel mills, and tire manufacturers. After manufacture, cars are sold to the end consumer through the car and automobile sales industry. Fig. 3-1 illustrates this value chain. We believe that our product, localization software specifically, fits in more closely with the car manufacturers' end of the value chain because it requires well-developed sensors with their software packages and configuration information of the cars which our algorithm will be deployed upon.


Figure 3-1: The automobile industry value chain

In addition to the value chain, PorterâĂŹs Five Forces model helps illuminate how profitable the industry currently is (Fig. 3-2). The power of suppliers is weak, as the supplier market for the automobile manufacturing industry is very fragmented. Furthermore, the power of car buyers is relatively weak. Car buyers in general do not collectively have leverage over manufacturers. However, as there do not exist consolidated channels where large volumes of orders are bought, customersâĂŹ buying power is not necessarily strong. The threat of substitutes is weak, as many places require car travel for moving between places. Rivalry, however, is strong. While there are a limited number of car manufacturers in each market segment, the product is often hard to differentiate from those of competitors within each respective market segment, potentially increasing the level of competition. Due to the complexity of design and demand for resources, the barrier of entry has been high for the automobile industry. Thus threats from new entrants have been low, though exceptions have occurred, such as Tesla. Electric vehicles may not be the only potential breach point for the automobile industry, however.

With autonomous driving technology, big technology companies like Google and Apple are seriously thinking of entering the market with a strong offering of tech features [4]. Established players in the luxury market segment such as Mercedes-Benz, BMW, and Audi already have advanced driver assistance systems incorporating aspects of autonomous driving technology. From our Five Forces analysis, the profitability of the automobile industry may be considered high, and the introduction of this new technology could bolster this even further for the players that beat out the increasing competition. Facing such a big industry and complex technology, there are two aspects to consider when deciding upon technology strategy. First is the strategy of entering different segments of the market, and second is the question of how to

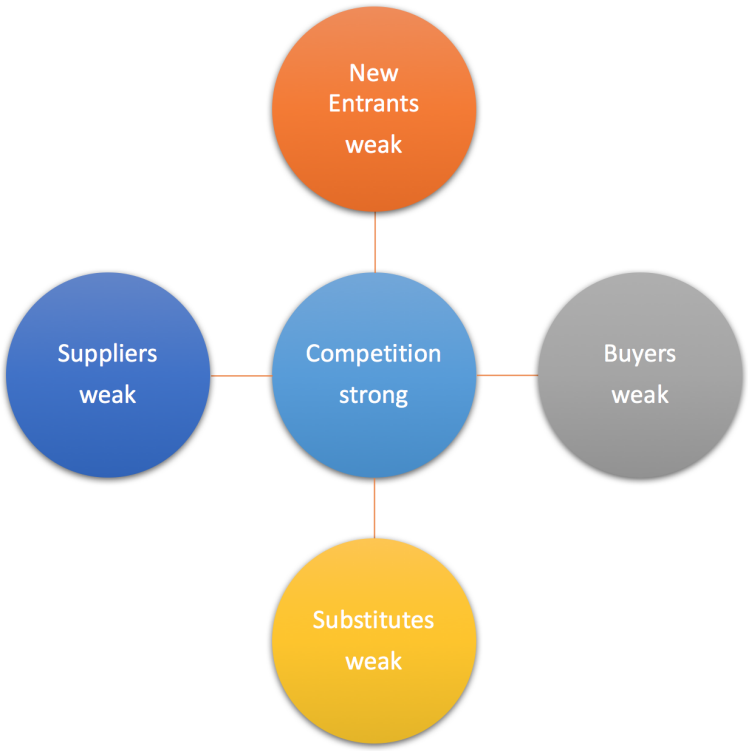


Figure 3-2: Porter's Five Forces analysis of industry profitability.

differentiate oneself with respect to competitors.

The automobile industry can be separated into two main components: the trucking industry with an annual revenue of $24 billion [14] and the sedan (passenger cars) industry with an annual revenue of $527 billion. The trucking industry represents roughly five percent of the automobile industry but its need for autonomous driving technology is arguably stronger, based on the fact that trucking is a highly repeated process and with long duration. Furthermore, trucks spend a majority of traveling time on highways, which is a much simpler situation compared to urban streets for autonomous driving technology. Targeting this market first would allow a new entrant to break into the market and gain customers. After establishment in the truck industry and gaining enough experience, the company can approach the sedan market segment, starting with the high-end. In general, more expensive luxury cars tend to be better manufactured with higher quality equipment such as sensors and computational power, which are key to our localization software. With further algorithm development and time, the robustness of the autonomous driving technology product

can be improved until it can be offered in lower-end market segments, where sensor equipment and computational power are relatively limited.

Potential competitive advantages are proprietary technical barriers as well as adaptable software architectures. In the case of our capstone team, highly precise localization technology that is robust against sensor failures may serve to distinguish against competitors in terms of performance. Additionally, developing high-level software architectures that are flexible and adaptive to various hardware configurations on different automobiles is crucial to build a large customer base with different car model needs [2]. A car manufacturer developing localization software as part of their autonomous driving package should have a clear go-to-market strategy and methods of maintaining a competitive advantage in order to be successful.

Corroborated by the numerous public debates on the topic, a third key aspect of autonomous driving requiring consideration is its ethical component. Currently, both US and European legislation are discussing liability in autonomous driving [5],[3],[10], and chapters on ethics have already found their way into autonomous driving books [11]. Here, we focus on one central question, namely, who should be held responsible for accidents involving autonomous cars?

To start with, we notice that questions of liability are often difficult to answer even if only manually driven cars are involved in an accident. The involvement of autonomous cars just makes the analysis more difficult. [6] presents an ethical analysis of this question, in which the authors consider holding (i) the manufacturer or (ii) the person using the autonomous car responsible for potential accidents. Both options show important caveats. According to NHTSA (2015) human failure leads to over 90% of car crashes, which suggests that overall traffic safety is likely to benefit from mature autonomous driving technology. For this reason, it is clear that any liability system should not discourage manufacturers from developing this new technology. Conversely, it seems evident that the person using an autonomous car cannot be held liable for something upon which he or she cannot take action. Hevelke and Nida-Rumelin argue, however, that by the very choice of using an autonomous car, a person actually chooses to expose anyone on the road to a potential threat, hence

taking on some amount of responsibility. Consequently, they propose a notion of shared responsibility among all users of autonomous cars, based on which one could justify a special, mandatory insurance for the use of autonomous cars.

In our opinion, the introduction of a particular type of insurance policy would be one viable option practically addressing the liability issue. Without necessarily assigning a party at fault, it would recognize the fact that, despite leveraging the most advanced autonomous driving technology, unforeseeable events may still occur which cannot necessarily be attributed to specific agents. We advocate, however, that once this new technology reaches a certain level of maturity, the manufacturer should be held responsible for providing some sort of standardized safety guarantee on its autonomous driving system. Such a hybrid solution of insurance as well as safety standards would provide the advantage that manufacturers do carry a reasonable amount of responsibility for accidents without being discouraged from pushing the development of this new technology forward.

In taking a solution to a highly complex and technical problem to a broader commercial business that provides tangible value to society, we must understand the solution's surrounding situation. Autonomous driving technology has the potential to radically disrupt a mature automobile industry, and with a well-planned technology strategy to break into such a large, albeit pervasive, customer base, we can establish and maintain a sustainable competitive advantage. As the technology matures, regulations and standards may need adoption to deal with a potential shift in liability of accidents. Industry players will need to keep abreast of these various contextual issues in order to successfully, and profitably, commercialize autonomous driving technology.

# Bibliography


[1] Ian Baldwin and Paul Newman. Road vehicle localization with 2d push-broom lidar and 3d priors. In *Robotics and automation (ICRA), 2012 IEEE international conference on*, pages 2611–2617. IEEE, 2012.

[2] Christian Berger and Bernhard Rumpe. Autonomous driving-5 years after the urban challenge: The anticipatory vehicle as a cyber-physical system. *arXiv preprint arXiv:1409.0413*, 2014.

[3] Heather Bradshaw-Martin and Catherine Easton. Autonomous or âĂŸdriverlessâĂŹ cars and disability: a legal and ethical analysis. *European Journal of Current Legal Issues*, 20(3), 2014.

[4] Joshua Dowling. Google and apple will get into the car business, says top auto industry executive. *News Corp Australia Network*, 2015.

[5] Jeffrey K Gurney. Driving into the unknown: Examining the crossroads of criminal law and autonomous vehicles. 2014.

[6] Alexander Hevelke and Julian Nida-Rümelin. Responsibility for crashes of autonomous vehicles: an ethical analysis. *Science and engineering ethics*, 21(3):619–630, 2015.

[7] Henning Lategahn, Markus Schreiber, Jens Ziegler, and Christoph Stiller. Urban localization with camera and inertial measurement unit. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 719–724. IEEE, 2013.

[8] Jesse Levinson and Sebastian Thrun. Robust vehicle localization in urban environments using probabilistic maps. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4372–4378. IEEE, 2010.

[9] Xiang Li. Name tba. Master's project, University of California, Berkeley, Department of Industrial Engineering and Operations Research, May 2016.

[10] Gary E Marchant and Rachel A Lindor. Coming collision between autonomous vehicles and the liability system, the. *Santa Clara L. Rev.*, 52:1321, 2012.

[11] Markus Maurer, J Christian Gerdes, Barbara Lenz, and Hermann Winner. *Autonomes Fahren: technische, rechtliche und gesellschaftliche Aspekte*. Springer-Verlag, 2015.

[12] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. Comparing icp variants on real-world data sets. *Autonomous Robots*, 34(3):133–148, 2013.

[13] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. Comparing ICP Variants on Real-World Data Sets. *Autonomous Robots*, 34(3):133–148, February 2013.

[14] Brandon Ruiz. Global car automobile manufacturing. *IBISWorld Industry Report*, 2015.

[15] Kilian Schindler. Name tba. Master's project, University of California, Berkeley, Department of Industrial Engineering and Operations Research, May 2016.

[16] Thomas SchoÌĹps, Jakob Engel, and Daniel Cremers. Semi-dense visual odometry for ar on a smartphone. In *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*, pages 145–150. IEEE, 2014.

[17] Markus Schreiber, Carsten Knoppel, and Ulrik Franke. Laneloc: Lane marking based localization using highly accurate maps. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 449–454. IEEE, 2013.

[18] Jian Wu. Name tba. Master's project, University of California, Berkeley, Department of Mechanical Engineering, May 2016.

[19] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems Conference (RSS)*, pages 109–111, 2014.

[20] Ji Zhang and Sanjiv Singh. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2174–2181. IEEE, 2015.