Lessons Learned from a Lightweight Autonomous Vehicle Implementation

Kuan Wang[1], Dawei Sun[1], Xiaojian Ma[1], Zhaoyuan Gu[1], Hao Zhao[1], Tianlei Zhang[1]
[1]Skyworks, Tsinghua University
{wangkuan15,sdw14,maxj14,gu-zy14,zhao-h13,ztl08}@mails.tsinghua.edu.cn

Abstract

Arguably, autonomous driving is one of the most promising computer vision applications. Yet we are still at very early stages towards true autonomous vehicles. From the perspective of laboratory-level research, we think there are two major issues: (1) autonomous driving involves non-trivial low-level mechanics, which is not straight-forward for most computer vision researchers; (2) even when restricted to the algorithm domain, knowledge about various sub-fields are needed (e.g. odometry and recognition).

Here we present a lightweight autonomous vehicle implementation, which can follow pre-defined routes based solely upon vision. Implementation details, qualitative results and intuitive observations are reported in these four aspects: (1) hardware solution; (2) visual odometry; (3) path planning; (4) visual recognition. Although it is out of our capability to do large-scale quantitative evaluations in the wild, we provide videos that clearly show the quality of our implementation. As a disclaimer, no particular theoretical contribution is reported or validated. Yet we expect autonomous driving researchers to find our empirical results comprehensive and informative.

1. Hardware Solution

Our hardware solution is illustrated by Fig 1. All four wheels are independently suspended on a fifth-scale chassis, powered by a single motor. A pure mechanical assistant steering system, connected to front wheels and a servo, controls the vehicle's heading direction. Both the motor and the servo receive pulse signals from an Arduino board.

In the first design (Fig 1 above), primary sensors are a set of BDStar C200-AT differential GPS system and a 1-line SICK TIM561 LIDAR. Two antennas of 10cm radius and the GPS processor are attached to a strip-shaped aluminum support. The support's design is a compromise between price, weight and strength (which is discarded in later designs). The LIDAR is mounted on a carbon fiber support with four stabilizing springs (which is proven unnecessary in later designs). We intend to use the Arduino board for computation taking solely GPS and LIDAR inputs, however, this design fails as the GPS system sees $\backsim 1m$ level drifts in all demo scenarios we present later (due to the influences of trees and buildings). We wonder whether this GPS system would be applicable in general city roads, but on our campus it is far from reliable. The LIDAR works in a reasonable manner.

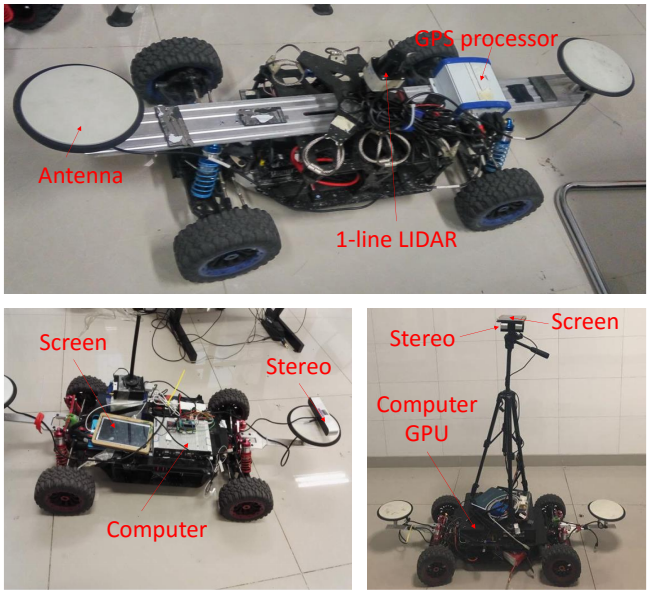

Figure 1. Hardware solution overview. Above: 1st version. Left bottom: 2nd version. Right bottom: 3rd version.

In the second version (Fig 1 left bottom), we replace those two mentioned (over-designed) supports with an acrylic board which proves to be strong and stable enough for all the high-level vision tasks we present later. A computer without GPU and a stereo of 12cm baseline are integrated for computation and visual inputs, respectively. A typical view of this stereo is illustrated by Fig 2a. Unfortunately, in this view geometry constraints are somewhat degraded (see the vanishing lines in Fig 2a) so that state-of-the-art visual odometry pipelines (specifically, ORB-SLAM and DSO) do not work, even after careful parameter tuning.

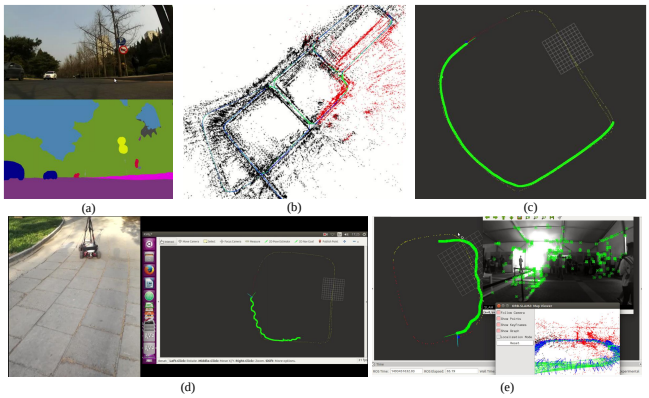

Figure 2. Software solution overview. (a) semantic segmentation. (b) stereo odometry. (c) path planning. (d) outdoor driving. (e) indoor driving. Better viewed in corresponding demo videos.

Our third design (Fig 1 right bottom) introduces a consumer tripod and a high-performance computer with Intel-i7 CPU/Nvidia GTX1070 GPU. Although primitive, we find the tripod essential for successful odometry. The motor and the computer are powered by two separated 12000 mAh Lithium batteries. Empirically, computation is always the endurance bottleneck in all our experiments.

2. Visual Odometry

We use ORB-SLAM for odometry out of its robustness and readability. Although the authors have provided a real-time road scene demo on a KITTI sequence, its generalization ability remains unclear. We provide a real-time evaluation with three loops in favor of this pipeline (https://www.youtube.com/watch?v=RYJRCowrcUg). This sequence shows challenging illumination change and clutter at around 4:00 and 6:20, which are handled properly. Interestingly, we refer readers to around 8:40, showing that when the third loop closes the last two loops are automatically aligned. We think this is a (clear) qualitative evidence that ORB-SLAM can automatically correct trajectory drifts in this kind of scenes.

Since the odometry is proven to be robust, we explore the possibility of using the corresponding map as a high-resolution navigation reference (hi-res compared with GPS). We extend the official ORB-SLAM so that the visual map (consisted of key-frames and sparse 3d point clouds) can be saved and loaded. Our implementation is at https://github.com/THUKey/ORB_SLAM2/tree/savemap, with which following a pre-defined route is made quite straight-forward. We expect odometry researchers to be interested in this function.

Besides, we experiment with stereo odometry on the same sequence but find the overlapping loops not successfully aligned (Fig 2b). This may be caused by the rolling shutter or stereo synchronization problems.

3. Path Planning

We implement a ROS driver for our vehicle which packages low-level mechanics into coordinate representations. This is designed to take GPS inputs, but compatible with visual navigation inputs after some geometric tweaking. For high-level path planning, a trivial preview point algorithm is implemented as demonstrated by Fig 2cde. Since these screenshots are not very informative, we refer readers to our outdoor demo (https://www.youtube.com/watch?v=R9m7T51l9VU) and indoor demo (https://www.youtube.com/watch?v=3VoUPGVuJZ0).

The outdoor demo shows a third-view point of the vehicle's behavior and its synchronized planning dynamics. Preview points are colored in red. In this experiment, we deliberately magnify the steering angle by 5 times, resulting a zigzag-style path, to show that the vehicle runs on its own instead of under manual control. The indoor demo shows odometry (on a pre-captured and loaded map), planning dynamics and first-view inputs simultaneously. This demo demonstrates that our solution could work robustly at the presence of a lot of dynamic objects.

4. Visual Recognition

We investigate whether semantic segmentation is applicable for an autonomous vehicle implementation like ours. This part of experiments are done on the 2nd-version hardware platform as shown by Fig 2a. Specifically, we exploit the pre-trained ResNet101-based model described in PSPNet. A video demo is provided at https://www.youtube.com/watch?v=ivr2Bp_q4mw. This clearly shows that a strong model properly trained Cityscapes generalizes well on our campus, successfully outlining hard categories like road signs and poles. However, it takes about 6 seconds to process one frame on our vehicle, far from real-time, and drains the battery quickly.

5. Conclusions and Acknowledgement

We comprehensively report a lightweight autonomous vehicle implementation, pointing out what works and what does not in a campus environment. Notably, we show that our vehicle can drive automatically according to pre-defined routes, using solely vision inputs. We believe our empirical results could serve as a design reference for any researcher who wants to build a laboratory-level autonomous vehicle with limited budgets. Besides, from the engineering side, we point to two potentially fruitful research directions: (1) efficient CNN-based semantic segmentation methods; (2) vision geometry algorithms for a lowly mounted camera. Sincerely we thank Guangzhou Chemi Inc. for providing us with key hardware equipment.