Scene understanding to aid in limited-bandwidth remote driving

Project Progress Report - CS 231

Hayk Martirosyan
Brady Jon Quist

February 23, 2015

1 Introduction

Progress on autonomous driving promises to make our transportation infrastructure more efficient. In particular, huge efficiency gains could be made in the transportation of goods by using vehicles without humans in them. These vehicles would have a fraction of the components of passenger vehicles, and could be designed without crumple zones, seats, steering wheels, infotainment, airbags, door and window controls. Instead, they could consist of a simple electric drive-train, a sensor cluster, and a cargo bay. In this type of logistics infrastructure, the hardware is lighter and cheaper, there are no human drivers, and the system can be intelligently automated and running 24/7, 365.

However, fully autonomous vehicles are not yet ready to fulfill this vision. Existing solutions operate well in pre-mapped regions and good conditions, but break down in adverse weather and in the presence of unexpected events or unknown terrain. One alternative which acts as a gateway to this ultimate goal is to use remotely-driven vehicles. The hardware remains the same, but the vehicle is operated by a human in a driving simulator at a remote location. This approach retains many of the efficiency benefits of the autonomous version while keeping the flexibility and judgement of human drivers.

One of the key challenges with this approach is maintaining a robust communication stream that enables remote driving. In order to reliably operate a remote vehicle, the driver must have a low-latency, high-fidelity, and wide-coverage stream of audio and video. Existing infrastructure has good coverage in many areas, but bandwidth can be very limited. To accommodate this, we propose that many parts of a typical stream have no importance to driving capability. For example, the driver does not care about details of the sky, trees, or objects very far away. A nearby object moving quickly across the view, however, is extremely important.

We propose to rank the components of a vehicle's stereo vision stream in terms of their importance to remote driving capability, with the goal of enabling limited-bandwidth transmission of the stream.

2 Technical

As this project focuses on the Computer Vision component of this problem, we will primarily use stereo camera simulations and real world data that are publicly available and are well calibrated [1][2][4][5]. This will alleviate some of the potential difficulties associated with the setup (e.g., alignment, syncing the image frames from two separate cameras, etc.) and allow us to get further in the computer vision aspects of our proposal.

There are several methods to be considered for extracting information from the image that are relevant to bandwidth reduction.

Lane detection[1] - Localizing the road and lane markers is extremely important to understanding the scene. The road vanishing point is the single most important point of the image, and naturally segments the image. The triangle formed by the lane markers identifies the immediate terrain that the vehicle is crossing and the driver should be paying attention to.

Stereo disparity map - Calculating a depth map of the scene from the stereo cameras is extremely useful for understanding, because we can draw conclusions about how important various features are by their distance from the vehicle. We can tie this depth data with recognition methods to identify features.

Image segmentation - Segmenting the image into known regions provides an easy way to enable variable compression. These objects might include the sky, road, grass, median, trees, buildings, etc. Based on the object classification, we could determine the compression level and frame rate that different portions of the video need to be transmitted at.

Feature recognition - More advanced feature detection for entities like vehicles, pedestrians, and animals would highlight key areas to pay attention to. This could potentially enable transmission of feature metadata instead of pixel data, to be reconstructed in the form of augmented reality for the remote driver.

Visual flow - Going beyond single-image methods, investigating the temporal differences of the images in the stream can tell us information about the movement of objects. We could say, for example, that objects moving quickly across the screen are extremely important to watch for.

Edge detection - General edge detection can be important for compression, with minimal required knowledge of the scene. In general, regions of the stream without significant edges are much less likely to be important to transmit in high-resolution.

We are exploring various combinations of these methods to determine good approaches for limited-bandwidth transmission. Ideally, we will combine 2D and 3D understandings of the scene to develop an algorithm for ranking the image stream by importance. If time permits, we would like to implement the image compression as a means of testing how useful the 3D point estimation and image segmentation/classification are for reducing the transmission bandwidth. It would be ideal for this whole process to work in real-time, but recognize that to do so may be beyond the scope of this project.

3 Milestones Achieved So Far

All data used so far comes from the synced and rectified images provided by [1]. Our software uses Python 3.4 and OpenCV 3.0.

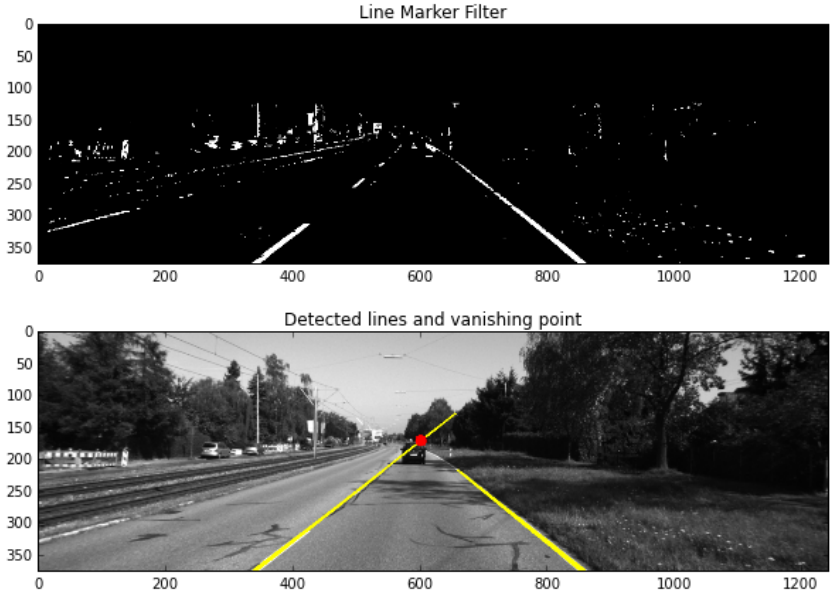
Our initial effort focused on the detection of lane markers and their vanishing point, which we judged the most important information to know. Our approach is to identify the lane marker lines and calculate the vanishing point as their projected intersection.

We initially tried Canny edge detection followed by a Hough transform for line detection of the lane markers. We found that the results often included spurious lines aligned with the outline of a row of trees or other irrelevant features. To overcome this, we implemented a special line marker detection filter as described by Nieto [3]. We further adapted it to include a binary erosion algorithm to eliminate noise. Note that we also removed the top 1/3 of the image because it should never be needed to detect the lanes.

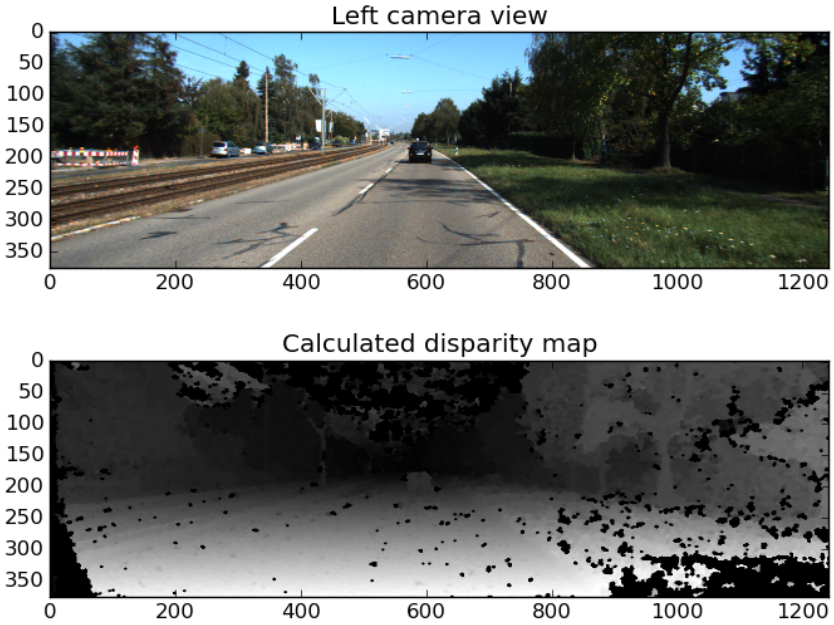
We then explored line detection using this special lane marker filter. Our initial experiences showed lines connecting dense regions of scattered points. To combat this, we switched to a

[1] As part of our efforts, we created a tutorial for lane detection using OpenCV 3.0 and Python 3.4. It can be viewed online at `http://nbviewer.ipython.org/github/hmartiro/project-mvq/blob/master/python/notebooks/lane_detection.ipynb`

probabilistic Hough method as implemented in OpenCV. We defined a minimum line length and maximum line gap that greatly reduced the number of noisy lines. These parameters can be reasonably estimated ahead of time. In order to determine the vanishing point of the road, we implemented our own RANSAC algorithm. Our implementation looked at the intersection of two randomly selected lines and found the number of detected lines which passed within a specified distance to that intersection. We have found this method to be robust to various types of images and that it reliably estimates the vanishing point.



We have also explored the creation of disparity maps from our data, which can potentially be very useful for compression. We take stereo images from a color camera pair, and feed them into OpenCV's semi-global block matching algorithm[2]. We tuned several parameters including the window size, disparity count, correspondence uniqueness ratio to find a good fit for our images. Finally, we use a closing algorithm with a small elliptical kernel on the disparity map to fill in gaps and create a smoother image.



[2]OpenCV StereoSGBM `http://docs.opencv.org/java/org/opencv/calib3d/StereoSGBM.html`

4 Future Milestones

1. 3D point estimation
    - Modify parameters/algorithms to determine how far away the objects are and the angle to target to determine compression level.
2. Image segmentation/classify
    - Using the vanishing point and initial line estimates, detect lane and road boundaries to localize the lane/road to create a region where very little compression should occur.
3. Video compression (time permitting)
    - Compress the video based on the 3D scene detection and image segmentation to test the performance and benefits of the the two preceding items.
4. Make the whole process work in real time (moonshot - time permitting)
    - Make 3D scene detection, image segmentation/classification, and compression work in real time so that a vehicle could be driven remotely.

References

[1] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[2] Albert S. Huang, Matthew Antone, Edwin Olson, David Moore, Luke Fletcher, Seth Teller, and John Leonard. A high-rate, heterogeneous data set from the darpa urban challenge, 2010.

[3] Marcos Nieto. Lane markings detection and vanishing point detection with opencv. `https://marcosnietoblog.wordpress.com/2011/12/27/lane-markings-detection-and-vanishing-point-detection-with-opencv/`.

[4] Tobi Vaudrey, Clemens Rabe, Reinhard Klette, and James Milburn. Differences between stereo and motion behavior on synthetic and real-world stereo sequences. In *23rd International Conference of Image and Vision Computing New Zealand (IVCNZ '08)*, pages 1–6, 2008.

[5] Andreas Wedel, Clemens Rabe, Tobi Vaudrey, Thomas Brox, Uwe Franke, and Daniel Cremers. Efficient dense scene flow from sparse or dense stereo data. In *10th European Conference on Computer Vision (ECCV '08)*, pages 739–751, Berlin, Heidelberg, 2008. Springer-Verlag.