

Michela Meister
“Detecting Drivable Surfaces in a Vehicle’s Path”
CS229 Final Report

INTRODUCTION:

Many autonomous driving programs rely on LIDAR sensors to detect drivable surfaces. However, LIDAR sensors can be expensive and do not have a very large range. Physical factors, such as dirt, the sensor’s mounting on the car, and light can cause additional problems (Dahlkamp et al.). Computer vision methods can greatly extend this range and can sometimes be cheaper than investing in a high-powered sensor.

In this project I propose a method that uses the video feed of a car-mounted camera to classify surfaces ahead as either drivable surfaces or obstacles.

I would like to thank Tao Wang of SAIL for mentoring me on this project.

METHODS:

Overview

We begin with the assumption that the ground directly in front of the vehicle, the “training area”, is drivable. We can then compare an image of this area of drivable ground to surfaces ahead of the vehicle in the current video frame. If these other surfaces look sufficiently similar to the training area, we can decide that they are also probably drivable. If they are sufficiently dissimilar, we classify these other surfaces as obstacles. These other surfaces could actually be drivable, and may just have a different appearance from the training area, but for the purposes of this project, we classify surfaces different from the training area as obstacles and surfaces similar to the training area as drivable.

Choosing Surface Patches to Compare

How can we compare the training area to other surfaces in the video frame? It would be inaccurate to simply crop a patch of size $m \times n$ from the training area and crop another patch of size $m \times n$ of a surface ahead of the vehicle. Because the second patch is a farther distance from the camera than the training patch, the second patch represents a much larger road area in 3D space than the training patch.

Instead, using the camera’s transformation matrix and GPS data on the vehicle’s movements, we create a perspective transform to warp the training area to its pixel representation i seconds ago when it was at a position d with respect to the vehicle. We find the perspective transform using the following method:

Let H be the quadrilateral that defines the training area in frame f at time t . The perspective transform is defined by H and H' , the pixel representation of H at time $t-i$. To find H' we have to find the pixel representation of each corner of H at time $t-i$. For each corner c of H represented by pixel $p(x,y)$ in video frame f , we use the camera’s transformation matrix to convert from pixel $p(x,y)$ to the 3D position $P(X,Y,Z)$ of c with respect to the camera at time t . Given GPS data on the vehicle’s movement, we can find the 3D position $P'(X',Y',Z')$ of c with respect to the camera at time $t-i$. Using the camera’s transformation matrix once again, we convert this 3D position to the pixel representation $p'(x',y')$ of c in video frame f' . The four values of $p'(x',y')$ found through

this process make up the corners of quadrilateral H' , which is used in conjunction with H to create the perspective transform (Szeliski, Ch 2).

Thus, since we can warp the contents of the training area to its representation at position d with respect to the vehicle, we can crop an $m \times n$ patch of warped training area and the $m \times n$ surface patch that is actually at position d with respect to the vehicle for comparison.

Patch Collection

Using the method described above, we collect a data set of 5000 pairs of patches, where each pair contained one patch of training area, the “training patch”, and one patch of surface area further ahead of the car, the “variable patch”. When warping the training area, we use time intervals of $i = 0.5, 1.0,$ and 1.5 seconds, to get variable patches at different distances from the car. Figure 1 and Figure 2 serve as examples of patch collection with $i = 0.5$. The trapezoid of green dots (H) at the bottom of Figure 1 defines the training area, which is warped to the rectangle of green dots (H') in Figure 2. The variable patch (Figure 3) is harvested by cropping H' from Figure 1, and the training patch (Figure 4) is harvested by cropping H from Figure 2.

Figure 1: Original Image



Figure 2: Warped Image

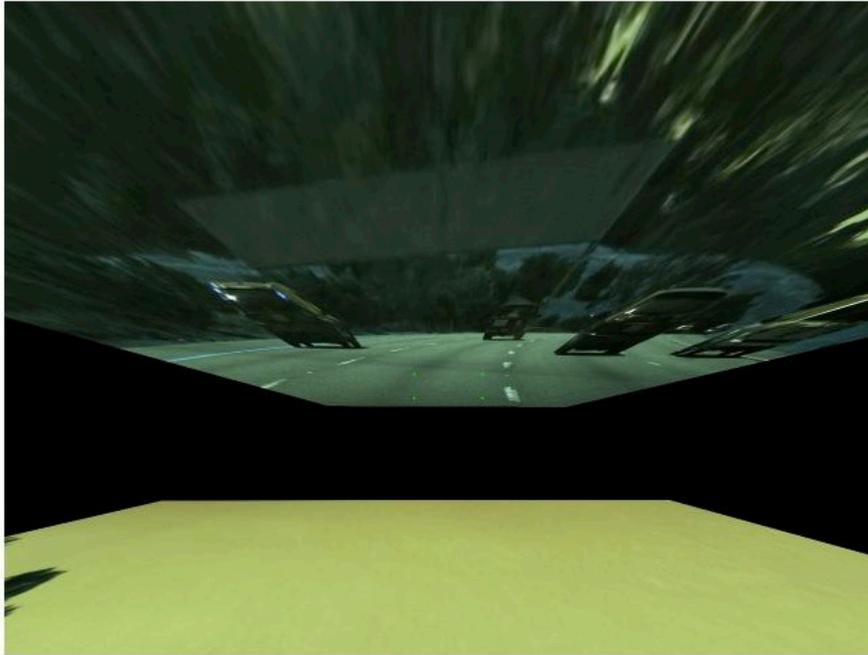


Figure 3: Variable Patch 1



Figure 4: Training Patch 1



We then manually label 1,000 pairs as “same” or “different”. While the patches in Figures 3 and 4 are labeled “same”, Figures 5 and 6 are labeled “different”, as Figure 5 shows the bottom of a car.

Figure 5: Variable Patch 2



Figure 6: Training Patch 2



Patch Comparison

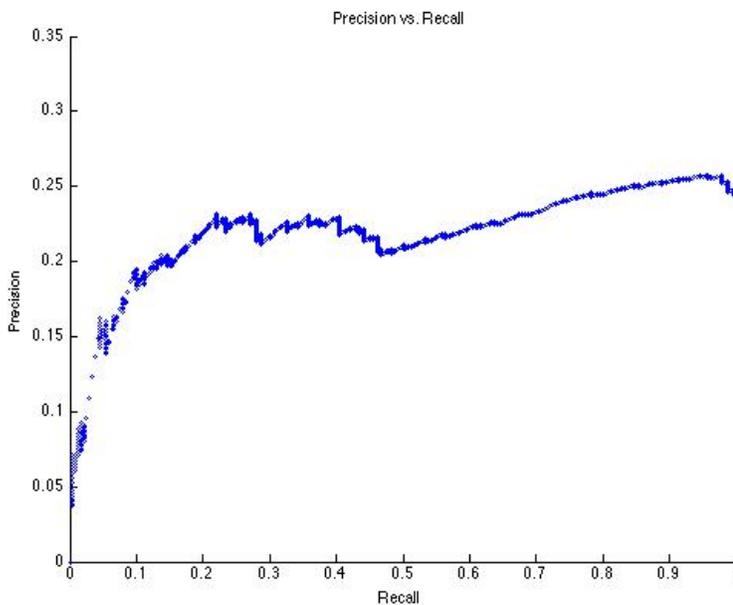
Given a training patch and a variable patch, we need to determine whether they are similar enough for the variable patch to be determined drivable. We used the sum of the distance-squared between corresponding pixels in a pair of patches as a metric for the similarity of the patches. A lower distance value means the patches are more similar, so

the variable patch is likely drivable, while a high distance value means the patches are less similar, so the variable patch is likely an obstacle.

For the 1000 manually-labeled pairs, we compare the distance values for each pair to their labels. We tried each pair's distance value as a threshold for similarity and calculated the precision and recall for each threshold, as plotted in Figure 7. A variable patch was considered a true positive if it was labeled an obstacle by both the threshold and the manual labeling, a false positive if it was labeled an obstacle by the threshold but not by the manual labeling, and a false negative if it was labeled drivable by the threshold but not by the manual labeling.

RESULTS:

Figure 7: Precision vs. Recall



When a threshold of 242 for the average distance-squared between two corresponding pixels is used, precision reaches a maximum of .25 and recall is .98. When a threshold of 198 for the average distance-squared between two corresponding pixels is used, precision is .24 and recall reaches a maximum of 1.0.

ANALYSIS & SOURCES OF ERROR:

Precision is always very low, however precision and recall increase together. A recall of 1.0 means that all obstacles are recognized, but false positives are a large problem.

There are many possible sources of error to explain the prevalence of false positives. First, there may be human error in the manual labeling of the patches. We label pairs as "same" if both patches were clear road surface, disregarding lane markings and shadows, and only label patches as "different" if the variable patch contained an obstacle, such as the car in Figure 5. However, while a white lane marking is not an obstacle, it is a

different color from plain road surface, and so the sum of the distance-squared between corresponding pixels for such a pair may suggest that the variable patch is an obstacle.

There also is error in warping the training area, which may result in error in the data set. After much debugging, we found that, because of the difference in height between the car's IMU and camera, points far ahead of the vehicle could not be reprojected perfectly to their locations at a prior time. Because of this error, some training patches may not have been pure road surface.

CONCLUSIONS:

We found a preliminary threshold for determining the similarity of two patches and thereby classifying unknown patches. However errors still need to be corrected in the warping of the training area and in the manual labeling of patches. The sum of the distance-squared between corresponding pixels in a pair of patches is a metric that can successfully recognize obstacles, but with many false positives. We must address lane markings and shadows, which alter the appearance of road surface but not the drivability, in order to decrease these false positives.

FUTURE WORK:

I chose this project, because it is my dream to build driver assistance technology. Data collection for this project was a huge challenge that taught me a lot about computer vision but which also required significant time, coming at the expense of applying more extensive machine learning techniques. In the following months I plan to continue this project, and with more time I plan to implement other techniques for classifying patches. I might experiment with different metrics, such as the peak color or intensity of a patch in order to decide the similarity of two patches. I am also considering using the histogram of oriented gradients to define the attributes of a patch. I could then use an SVM to classify patches given these attributes. Eventually I would like to extend the patch classification to the entirety of the vehicle's view.

REFERENCES:

H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun and G. Bradski: *Self-supervised Monocular Road Detection in Desert Terrain*.

Szeliski, Richard. *Computer Vision: Algorithms and Applications*. 3 Sept 2010 draft.