Vanishing Point Detection for Road Navigation

Xinhe Feng xf459
Lina Qiu lq437

1 Motivation

Given a single image of an arbitrary road, that have clearly or not clearly delineated edges, or may not be well-paved, or have some obsoletes ahead, is it possible for a computer to find the vanishing point of the road? Vanishing point provides valuable information as to how certain image features should be interpreted. Many autonomous mobile robot navigation systems are built on the vanishing point. Besides, in many scenes there exist a number of straight lines which are mutually parallel in 3D, but under perspective projection these lines will meet at a common point known as the vanishing point. Once this point is identified, one can infer 3D structures from 2D features. Therefore, accurately and efficiently detecting vanishing points is of great significance in image recognition.

Traditional Hough Transform based road and vanishing point detection algorithms only work well on well-paved structure roads that have remarkable boundaries. If the road structure is not clear, the hough space of the image is likely to show peaks that represent other more obvious lines in the image space. Even though a road has delineated edges, other clear line components in the image can still bring significant noise to the detection. Knowing these drawbacks, we would like to try a more robust and more popular way of detecting the vanishing point and segmenting roads.

2 Introduction

Vanishing point (VP) is the convergence of the projection of two parallel road edges. Generally, it can be solved by finding the intersection point of the road edges detected by Hough Transform. But the traditional way of using Hough Transformation to find the vanishing point can be unstable and even inconsistent for unstructured roads without noticeable markings or borders. So we use a recently popular way to do vanishing point detection: texture analysis . For this project, we implemented some parts of the paper “Vanishing point detection for road detection”[1].

In our implementation, we try to make a texture orientation estimation first. This method works because the vanishing point is always associated with the main part of the road that contains some texture more or less. Then we perform a soft-voting(LASV) to select a vanishing point. At last, we use the detected vanishing point to find the road boundaries through texture orientation of pixels as well. The boundaries would on the other hand help to adjust the detected vanishing point. Since we do not use the color factor, this algorithm works alright under various illumination condition.

3 Approach

3.1 Texture orientation estimation

The texture orientation estimation relies on Gabor filters. Gabor filters is known as a reasonable model of simple cells in the Mammalian vision system. Therefore it is believed to be an effective model that can simulate how humans distinguish texture. The impulse response of Gabor filters is created by multiplying an Gaussian envelope function with a complex oscillation. Gabor[2] showed that these elementary functions minimize the space(time)-uncertainty product. By extending these functions to 2D it is possible to create filters which are selective for orientation.

Let $p=[x\ y]^T$ be the image pixel coordinates. The impulse response of a 2D Gabor filter g(x) is:

$$f(x, y, \omega, \theta, \sigma_x, \sigma_y) = \frac{1}{2\pi\sigma_x\sigma_y}\exp\left[\frac{-1}{2}\left(\left(\frac{x}{\sigma_x}\right)^2 + \left(\frac{y}{\sigma_y}\right)^2\right) + j\omega(x\cos\theta + y\sin\theta)\right],$$

Where σ is the spatial spread, ω is the frequency and the θ is the orientation of the filter.
Then to simplify the calculation and save the time, we split the 2D Gabor to two 1D Gabor filters:

$$f(x, \omega, \sigma) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(\frac{-x^2}{2\sigma^2} + j\omega x\right).$$

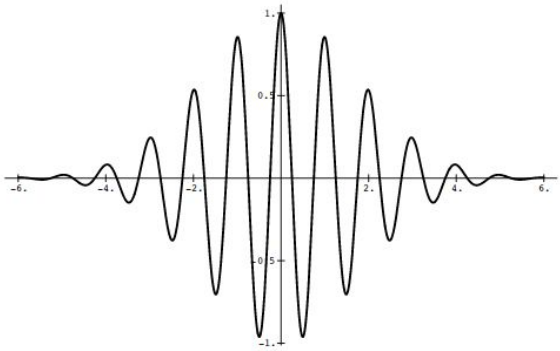
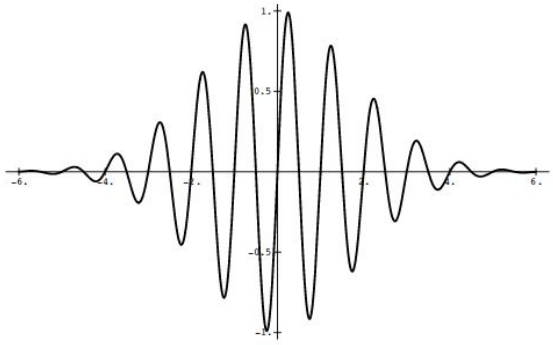


Figure 1: The even-symmetric component of 1D filter.



The odd-symmetric component on 1D filter

We consider 36 orientations to get enough details of the image. Though usually, less than 36 orientations would be considered (8 or 4) in the implementation for saving code running time. But we use C++ for our implementation, which in most time is faster than other programming languages, and the running time of using 36 orientations is reasonable, so we stick with 36.

The C++ code implementation of the core of Gabor filter is:

```
float a = x * cos(theta) + y * sin(theta);
float b = y * cos(theta) - x * sin(theta);
float oddResp = exp(-1.0f / 8.0f / sigma / sigma * (4.0f*a*a + b * b)) *
sin(2.0f*PI*a / lamda);
float evenResp = exp(-1.0f / 8.0f / sigma / sigma * (4.0f*a*a + b * b)) *
cos(2.0f*PI*a / lamda);
```

```
oddKernel.at<float>(x + k, y + k) = oddResp;
evenKernel.at<float>(x + k, y + k) = evenResp;
```

The following images show the original road picture and Gabor filter results for 36 different orientations. We choose 17 as the kernel size and calculate 36 orientations(equally spaced between 0 to 180 degrees):

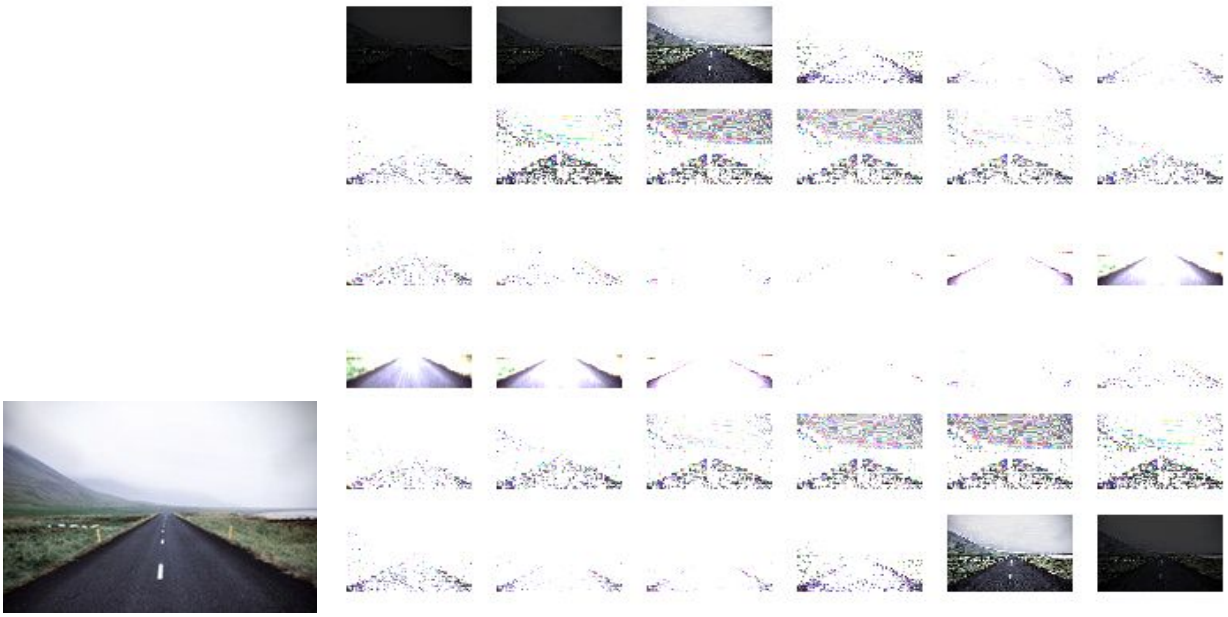


Figure 2: Original road image. ***36 filtered images for different orientations.***

We not only compute the texture orientation at each pixel, but also give a confidence to each estimation. Confidence value is used to select the candidate vp points. The brighter the pixel, the higher confidence the orientation estimation.

Let r1 > r2 > … >r36 be the ordered values of the Gabor response for the 36 considered orientation at pixel(x,y). The confidence in the orientation α is:

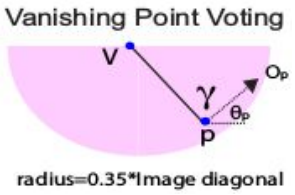
$$Conf(x,y)=1- Average(r5,...r15)/r1.$$

In our project, we discard the pixel whose confidence value is smaller than 0.35. The remaining pixels with high confidence values are selected to vote for the vanishing point location.

3.2 Voting Scheme

There are two voting strategy: hard-voting and soft-voting. The hard-voting strategy allows a pixel to vote for all candidate vanishing points ignoring the distance between them. But this can lead to a large offset in the estimation of the vanishing point because those points at the top of the images are more likely to receive larger votes. To eliminate the error, we choose the soft-voting strategy that the candidate vanishing point only receives the voting scores from its neighborhood. We set a constant distance value to filter unnecessary votes.

For each candidate point, we compute its likelihood by accumulating scores from a local neighborhood of pixels in a half-disk region.



In the experiment, we set the disk radius as the 0.35*image diagonal.

3.3 Vanishing Line

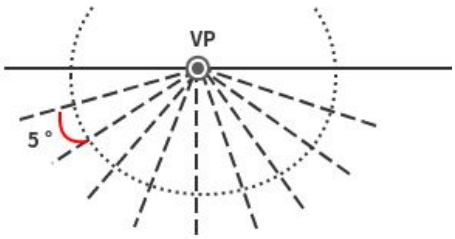
After getting the vanishing point on static road images, we also try to find the vanishing line of videos of road filming from moving view angles. In an road image π, when α is the ground plane and β is the horizon plane, then the vanishing line of α is the horizon line $\beta \cap \pi$. For different sets of lines parallel to the plane α, the vanishing point will lie on the vanishing line. The line represents the eye level of the observer.

So we can get the vanishing line by connecting several vanishing points. The vanishing line is detected by processing a sequence of images of a video. We store the coordinates of the vanishing point in each frame and then get the vanishing line by connecting all the vanishing points with lines. The vanishing points should lie on a straight smooth line if the height of the camera does not change.

3.4 Road boundary estimation

Road boundary estimation is another important part in autonomous navigation system. If we can decide an accurate vanishing point, it doesn't take us much effort to find the road boundaries because the road boundaries must intersect at the vanishing point. Therefore, we can perform a dominant edge detection method that is vanishing point constrained. We use Orientation Consistency Ration(OCR) theory to locate the road edge. That is, for each point, if the angle between the point's orientation and the line's direction is smaller than a threshold, this point is viewed to be orientationally consistent with the line. So combining the Gabor feature, the statement would be if there exist a line, all the pixels on the line should have similar Gabor responses and orientation even if the dominant angle is not aligned with the vanishing point ray.

In the experiment, we examine rays with 5 degrees spacing and evaluate how likely each ray is in fact a line (a road boundary) in the image.



For each candidate boundary line, we select 9 points on it and check how many orientations of these 9 points are aligned with the direction of the line. After the computation, we select one line with the highest OCR from each half (left and right) of the image respectively as our road boundaries.

4 Data

We chose 100 images with different background, illumination and road boundaries from the Internet. For the video test, we shot several videos ourselves but there were unavoidably so many cars, people and other buildings in the background that block the vanishing point of the road. So we found some good videos on the Youtube[3,4,5] and truncated several stable short video sequences to test our implementation.

5 Results

We test the system accuracy in recognizing the vanishing point and road boundaries in both static images and videos. We test the system on 100 general road images and 2 video sequences. The road images are variant in color, texture, illumination and ambient environment. The detection is good and robust in images, though a little unstable in video sequences.

5.1 Vanishing point estimation

To speed up the process, we downsampled the original image before all the calculation and estimation. The error is defined as the deviation between the human annotated and system estimated vanishing point. The results are shown in Figure 3. You can see that the results are reasonable.

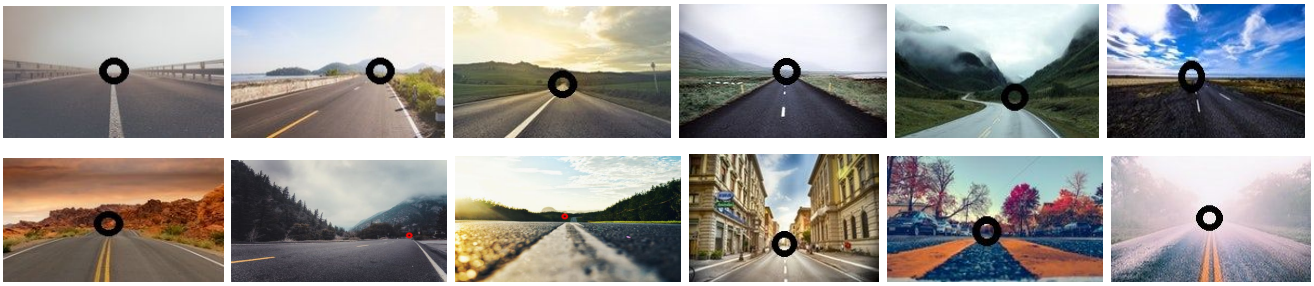
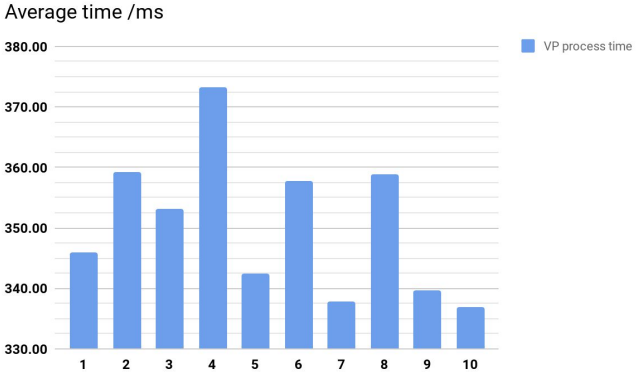


Figure 3: The results of vp detection

For some roads without remarkable boundaries like #6/8/12 in the above result images, the system can also give a good vanishing point. And for the curve road(#5) the vp point location is also acceptable.



For the processing time, we succeeded to make it run only 300-400 ms/frame in our system. The left chart shows the average time of processing 10 images 10 times. The time cost is low, which makes it possible to run in real-time systems. In

other words, it is practical to embed this algorithm in mobile autonomous navigation system to guide the steering.

Figure 4: Processing time

5.2 Vanishing line detection

The vanishing line detection works well on video sequences in which the recording camera is kept at the same height and the viewpoint is changed consistently. The following images show some snapshots of the processed video sequence. The location of the vanishing point is changing but still lying on the same line. You can see that the vanishing line detection is also effective in the video sequence for test.

Figure 5: Snapshots of a video sequence. You can see that the vanishing points(white circle) are lying on the same line.

5.3 Road boundaries detection and VP adjustment

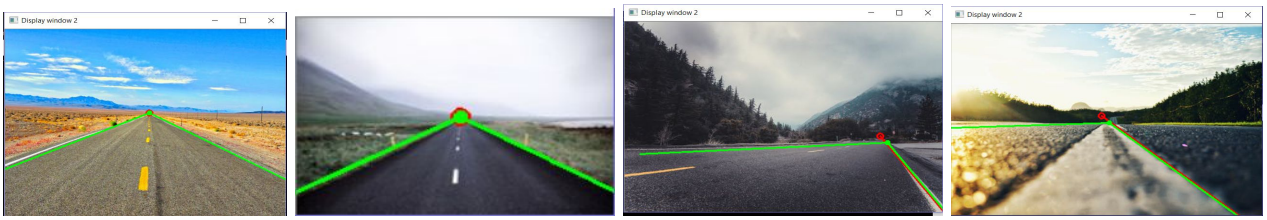


Figure 6: Results of road boundaries estimation and vp point adjustment

The red circle indicates the initial VP we detect, and the green circle indicates the adjusted VP. We can see that the adjusted VP is usually closer to the ground truth VP, so in return we can adjust the vanishing point location by the feedback of the road boundaries estimation. Here we have a realistic assumption that the ground truth VP lies on the dominant edge (one of the road boundaries) or its extension. We pick some neighbouring points of the initial VP from the dominant edge, and recalculate their OCRs. The one with the highest sum of OCRs from top 6 dominant edges is decided to be the adjusted VP. The two green lines represent the road boundaries of the adjusted VP.

From the last image, you can see that the detected right road boundary aligns with the center white line of the road which is wrong. This error is due to the high contrast between the white line and the road and the real road boundary is a bit blurred. This kind of error shows a drawback of the algorithm.

5.4 Failures

There are some situations that the algorithm cannot deal with. As we mentioned above, one significant requirement is that the vanishing point should be clear in the image, cannot be blocked by other objects. Also, if the image has some significant lines other than the real road boundaries, our implementation of road segmentation might sometimes be unstable.

6 Discussion

In this system, we first detect the location of the vanishing point by extracting Gabor features and using soft-voting scheme. Then we detect the vanishing line based on the location of the vanishing points. Finally we make a road region segmentation with the Gabor features and OCR theory.

In voting, only the pixels within a specific neighbor region and with high confidence can vote to the candidate points. This method can avoid the points from the top of the image obtaining large votes. In vanishing line detection, we simply connected the detected vanishing point based on the math theory.

In road boundary estimation, the line with the highest accumulating aligned Gabor responses (OCR) is chosen to be the road boundary line. And in return, the detected boundaries can adjust the initial vanishing point to be more accurate. This feedback mechanism increases the accuracy of the system.

Overall, we achieved the original goal and provided a robust solution.

7 Future work

In our experiment, the result sometimes is unsatisfied and unstable because of the background noise (obstacles before the VP). To eliminate the noise, we can do some pre-processing on the image before performing the Gabor filtering (this is not implemented in our code).

Besides, we can do some image erosion/dilation to eliminate small noise patches in the background. These noises are very common when there is forest or cloud in the background.
In addition, we can also perform Gabor filtering on different sample scales and then make the average of the results.This can be done by building a Gabor pyramids. We believe this can increase the accuracy at the cost of the processing time.

8 Reference

[1] Hui Kong, Jean-Yves Audibert,École Normale Supérieure and Jean Ponce. Vanishing point detection for road detection.
[2] *http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/TRAPP1/literat_gabor.html.*
[3] *https://www.youtube.com/watch?v=pqZTyosPRG0*
[4] *https://www.youtube.com/watch?v=V126LxPopg8*
[5] *https://www.youtube.com/watch?v=dGMJwzLYTyk*

Contribution List

Member

Xinhe Feng xf459
Lina Qiu lq437

Work	**Contributor**
Design and Planning	Xinhe Feng, Lina Qiu
Gabor Kernel Implementation	Xinhe Feng, Lina Qiu
Soft-voting Implementation	Xinhe Feng, Lina Qiu
Vanishing Line Detection	Lina Qiu
Road Boundary Estimation	Lina Qiu
Data Preparing	Xinhe Feng, Lina Qiu
Testing	Xinhe Feng, Lina Qiu
Report Writing	Xinhe Feng, Lina Qiu