Enhancements of Vanishing Point Estimation in Road Scenes

Linh Nguyen

University of Wollongong

Follow this and additional works at: https://ro.uow.edu.au/theses1

Recommended Citation

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au
Enhancements of Vanishing Point Estimation in Road Scenes.

A thesis submitted in partial fulfilment of the requirements for the award of the degree

Master of Engineering by Research

from

University of Wollongong

by

Linh Nguyen

School of Electrical, Computer and Telecommunications Engineering

February 2018
This page intentionally contains only this sentence.
Statement of Originality

I, Linh Nguyen, declare that this thesis, submitted in partial fulfilment of the requirements for the award of Master of Engineering by Research, in the School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualifications at any other academic institution.

Linh Nguyen
25 February, 2018
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acronyms</td>
<td>VII</td>
</tr>
<tr>
<td>Abstract</td>
<td>VIII</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>X</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Research objectives</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Research contributions</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Thesis structure</td>
<td>4</td>
</tr>
<tr>
<td><strong>2 Literature Review</strong></td>
<td>7</td>
</tr>
<tr>
<td>2.1 Edge-based VPE methods</td>
<td>8</td>
</tr>
<tr>
<td>2.2 Texture-based VPE methods</td>
<td>12</td>
</tr>
<tr>
<td>2.3 Local orientation estimation methods</td>
<td>21</td>
</tr>
<tr>
<td>2.4 Deep learning-based VPE methods</td>
<td>27</td>
</tr>
<tr>
<td>2.5 Chapter summary</td>
<td>28</td>
</tr>
<tr>
<td><strong>3 Proposed VPE method</strong></td>
<td>31</td>
</tr>
<tr>
<td>3.1 Stage 1: Extracting VP features</td>
<td>32</td>
</tr>
<tr>
<td>3.2 Stage 2: Identifying VP search spaces</td>
<td>36</td>
</tr>
<tr>
<td>3.3 Stage 3: Calculating VP scores</td>
<td>41</td>
</tr>
</tbody>
</table>
## Contents

3.4 Chapter summary .................................................. 46

### 4 Experiments and Analysis

4.1 Image dataset .................................................. 50

4.2 Evaluation methods ............................................ 51

4.3 Analysis of voting algorithm parameters .................... 53

4.4 Analysis of voting functions ................................... 55

4.5 Analysis of histogram size for Bayesian classifier .......... 55

4.6 Comparison with other VPE methods ......................... 57

4.7 Chapter summary ................................................ 59

### 5 Conclusion

5.1 Thesis summary ................................................ 62

5.2 Future works .................................................. 63

5.3 Conclusion ..................................................... 64

**References** .................................................. 68
List of Figures

1.1 Examples of vanishing points in road scenes. ....................... 2

2.1 Example of the Hough transform. ............................... 9

2.2 Main stages in texture-based VPE methods. ....................... 12

2.3 Example of a Gabor filter bank with 8 orientations and 5 scales. . 13

2.4 Global voting scheme. The voting region \( R(v) \) of the candidate \( v \) includes all the pixels below \( v \). The vote score contributed by voter \( p \) is based on the angle difference \( \gamma \). .......................... 14

2.5 Local adaptive soft voting scheme. The voting region \( R(v) \) of the candidate \( v \) is the lower half-disk centered at \( v \) with the radius \( \rho \). The vote score contributed by voter \( p \) is based on the angle difference \( \gamma \) and the Euclidian distance between \( p \) and \( v \). ............... 16

2.6 Decomposition of a Gabor filter into a linear function of Haar-like box filters with \( c_i \) as the coefficients. .............................. 22

3.1 An overview of the proposed vanishing point estimation method. . 32

3.2 An example of the proposed VPE method. ......................... 33

3.3 Extracting edge pixels with non maximum suppression. ............. 35

3.4 Extracting local texture orientation and color edge maps. First row: Input images with texture orientation imposed. Second row: Color edge strength maps \( E \). Third row: Color edge maps \( B \). ............... 37
3.5 Example of detected straight line segments. The road border pixels are shown in red. 38
3.6 Defining VP candidate search spaces. First row: Input images. Second row: VP candidate search spaces. The ground-truth VPs are shown as red asterisks. 40
3.7 Examples of vanishing points located above the main road border in a road scenes. 40
3.8 Examples of VP voters optimization. First row: Input images with texture orientation imposed. Second row: Color edge maps $B$ as the initial voter space, with some spurious edge pixels marked. Third row: Optimized VP voter search space. 41
3.9 Computation of the VP vote score. Candidate $c$ collects vote scores from several voters $v$. The vote score depends on the Euclidean distance $d(v, c)$ and the angle difference $\Delta$. 42
3.10 Examples of VP vote score maps. First row: Input images. Second row: VP score maps with the estimated VP marked (red square). 46
4.1 Sample images from PLVP dataset with ground-truth VP annotated (red dot). 51
4.2 Sample images from the extension of PLVP dataset with ground-truth VP annotated (red dot). 52
List of Tables

4.1 VPE error for different values of $\tau$ and $\delta$ on the validation set.  54
4.2 Computation time (s) for different values of $\tau$ and $\delta$ on the validation set.  54
4.3 VPE error for different values of $g$ on the validation set.  54
4.4 Performance of different voting score functions on the validation set. The notable performances are given in bold.  56
4.5 VPE error performance on the validation set for different histogram sizes of the Bayesian classifier.  57
4.6 Performance of VPE methods on the test set.  58
**Acronyms**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP</td>
<td>Vanishing Point</td>
</tr>
<tr>
<td>VPE</td>
<td>Vanishing Point Estimation</td>
</tr>
<tr>
<td>2-D</td>
<td>Two Dimensional</td>
</tr>
<tr>
<td>3-D</td>
<td>Three Dimensional</td>
</tr>
<tr>
<td>PLVP</td>
<td>Pedestrian Lane Detection and Vanishing Point Estimation</td>
</tr>
<tr>
<td>LSD</td>
<td>Line Segment Detection</td>
</tr>
<tr>
<td>CHEVP</td>
<td>Canny-Hough Estimation of Vanishing Point</td>
</tr>
<tr>
<td>LASV</td>
<td>Local Adaptive Soft Voting</td>
</tr>
<tr>
<td>OLDOM</td>
<td>Optimal Local Dominant Orientation Method</td>
</tr>
<tr>
<td>gLoG</td>
<td>general Laplacian of Gaussian</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Networks</td>
</tr>
<tr>
<td>RGB</td>
<td>Red Green Blue</td>
</tr>
<tr>
<td>OOMP</td>
<td>Optimized Orthogonal Matching Pursuit</td>
</tr>
</tbody>
</table>
Abstract

A vanishing point is where the perspective projections of a set of parallel lines converge in the image plane. Vanishing point estimation (VPE) is an essential task in many applications including image-based camera calibration, 3D reconstruction and road detection. In image-based road detection, the approaches that utilize vanishing point are proven to be more efficient and robust than other solutions. However, estimating vanishing point from a single road image has been a demanding task, due to the variant nature of road scene and the computation restraint of real time application. The common drawbacks of the current methods are the lack of robustness in different road scenes and the slow processing time.

In this thesis, different VPE methods based on texture or edge features are investigated. This thesis proposes an efficient VPE method which is robust to different road scenes, and possesses fast processing time suitable for real-time applications. The proposed VPE method including three stages: extracting local texture orientation and edge maps, identifying the VP search space, and calculating VP scores with a voting scheme. Color tensor features are employed to extract texture orientation and edge maps. Based on these extracted map, the optimized search spaces for VP candidates and voters are identified using Hough transform and neighborhood pixels. The voting score function is defined with a novel weighting method and a simple Bayesian classifier to adaptively adjust the algorithm.
Experiments and analysis of the VPE methods are performed with the Pedestrian Lane Vanish Point Estimation dataset, which is extended to more than 4000 images. The results show that the proposed VPE method outperforms state-of-the-art methods in terms of accuracy, robustness, and processing time.
Acknowledgments

First and foremost, I would like to gratefully acknowledge the kind support of my main supervisor Assoc.Prof. Son Lam Phung, who has dedicated a great amount of time and effort to my study.

Special thanks also go to Prof. Abdesselam Bouzerdoum for his supportive supervision during my study.

I also offer my thanks to the staff of the School of Electrical, Computer and Telecommunications Engineering and my colleagues at the Video Audio Signal Processing labs for their valuable advices for my experiments.

Moreover, I would like to express my sincere gratitude to my beloved family and friends, who have encouraged and supported me during my studies and research projects.

Finally, a special remark is dedicated to my beloved wife, who has been offered her kind heart unconditionally throughout my research career.
Chapter 1

Introduction

Chapter contents

1.1 Research objectives ........................................ 2
1.2 Research contributions ................................... 3
1.3 Thesis structure ............................................. 4
1.1 Research objectives

A vanishing point (VP) is where the perspective projections of a set of parallel lines converge in the image plane (Figure 1.1). Vanishing point estimation (VPE) is an important step in many image processing applications, such as camera calibration [1], 3D reconstruction [2], and road detection [3]. In image-based road detection, the approaches that utilize vanishing point are proven to be more efficient and robust than other solutions [3, 4, 5]. However, estimating vanishing point from a single road image has been a demanding task. The image can be affected by many conditions such as different lighting and weathers. Objects such as vehicles and pedestrians also differ in each image. In addition, the roads have different types and structures. Currently, there are not many invariant features to define the road, especially in the images of unstructured scenes. Furthermore, the road detection applications generally require real-time processing, which places a constraint on the computational complexity of the vanishing point estimation algorithm.

Many studies, including edge-based and texture-based approaches, have been proposed, but the desired robustness and fast processing is yet to be achieved. Edge-based approaches rely on the presence of strong edges, such as clear road borders and lane markings. These approaches have low complexity, but they are easily affected by spurious edges in non-road regions and therefore are not so
1.2. Research contributions

effective with unstructured scenes. Texture-based approaches improve detection accuracy by employing texture orientations in pixel-wise voting schemes. However, they suffer from high complexity and slow speed. Furthermore, existing VPE approaches are mostly applied for intensity images, and only a few approaches utilize information from the color channels.

The objective of this research is to study the current techniques in VPE and develop a novel VPE method. The desired VPE method needs to be robust to different road scenes, while having fast processing time suitable for real-time applications.

In details, this thesis aims to address the following research questions:

1. What are the crucial image features for VPE? How do these features contribute to the VPE algorithm?

2. How to effectively extract these features for VPE from the road images taken in different road image conditions?

3. How to set up the parameters of VPE algorithm to achieve a consistent and accurate detection?

1.2 Research contributions

The main contributions of this research are:

- An investigation of different techniques for texture orientation and edge detection. Texture orientation and edge maps are essential features for most state-of-the-art VPE methods. The color invariant features [6] are studied because the color channels provide more information of the scene than the intensity channel, which is widely employed in conventional methods.

- A novel VPE method based on the pixel orientation and edge map with optimized search spaces of VP candidates and voters. The search spaces are
1.3. Thesis structure

reduced to provide faster computation time, while still containing essential pixels for accurate VP voting.

• A Bayesian classifier to adaptively adjust the voting algorithm, which improves the robustness of the method in different scenes. The classifier is built with histogram technique.

• An extended dataset based on the Pedestrian Lane Detection and Vanishing Point Estimation (PLVP) dataset [7]. More than 2000 images with ground-truth VP annotated are added to the dataset. With more than 4000 images, the extended dataset supports quantitative evaluation of VPE methods and the training process for the proposed classifier.

The publications arose from this Masters research thesis are listed as follows.


1.3 Thesis structure

The thesis is structured as follows:

• **Chapter 1** discusses the thesis objectives and highlights the research contributions and publications.

• **Chapter 2** presents a literature review on the VPE methods. In this chapter, the state-of-the-art edge based and texture-based VPE techniques are presented.
• **Chapter 3** proposes a novel VPE algorithm which optimizes the search spaces of VP candidates and voters, based on both texture and edge features. The VPE algorithm utilizes voting scheme, which includes a Bayesian classifier to adaptively adjust the voting function. The histogram technique for training the classifier is also addressed in this chapter.

• **Chapter 4** introduces the PLVP dataset and presents the experimental results and analysis. The experiments include the training process for the classifier and algorithm parameters, as well as the evaluation of the proposed VPE method along with the state-of-the-art method.

• **Chapter 5** provides a summary of research findings, future research directions, and the concluding remarks.
This page intentionally contains only this sentence.
Chapter 2

Literature Review

Chapter contents

2.1 Edge-based VPE methods .............................................. 8
   2.1.1 Canny-Hough estimation of vanishing point ................. 8
   2.1.2 Line segment detection based approach ..................... 10
2.2 Texture-based VPE methods ........................................ 12
   2.2.1 Global voting scheme approach ............................. 12
   2.2.2 Local adaptive soft voting approach ....................... 15
   2.2.3 Optimal Gabor filters and line voting approach .......... 19
2.3 Local orientation estimation methods ............................ 21
   2.3.1 Haar-like box filters ....................................... 22
   2.3.2 Generalized Laplacian of Gaussian filters ................. 24
   2.3.3 Color tensor features ...................................... 25
2.4 Deep learning-based VPE methods ............................... 27
2.5 Chapter summary ..................................................... 28
2.1 Edge-based VPE methods

This chapter presents a review of existing approaches to detect the VP. The edge-based and texture-based VPE approaches are reviewed, as well as the more recent approaches based on neural networks. The chapter is organized as follows. Section 1 analyses edge-based methods, while Section 2 surveys the texture-based methods that utilize Gabor filters. Section 3 studies different techniques for local orientation estimation in texture-based VPE. Section 4 reviews the VPE approaches using deep learning and neural networks.

2.1 Edge-based VPE methods

Edge-based VPE approaches are mostly applied for structured roads, which have clear borders and lane markings. These algorithms use the edge segments to detect the dominant straight lines in the image. Afterwards, the optimal VP is voted from the intersections of these lines. Traditional edge-based methods utilize the cascaded Hough transform to find the straight lines [3]. A more recent approach utilizes the Line Segment Detection (LSD) algorithm to construct an edge-based voting scheme [8].

2.1.1 Canny-Hough estimation of vanishing point

Wang et. al. proposed a VPE method named Canny-Hough Estimation of Vanishing Point (CHEVP) [3], which utilizes the Canny edge detection and cascaded Hough transform [9]. The cascaded Hough transform, or Hough transform in short, is employed to detect straight lines and curves in the image. With the Hough transform, each pixel in the image space is transformed to a line in the Hough space. The straight lines in the image are detected from the convergent points of the lines in Hough space. The basic of the Hough transform is illustrated in Figure 2.1. The line \( l : y = Ax + B \) is represented by the parameter point \( L = (A, B) \) in the Hough space. The points \( p_k = (x_k, y_k) \) contained by line \( l \) are transformed to the lines \( \zeta_k : a = \frac{y_k}{x_k} - \frac{1}{x_k}b \) in the Hough space, and these lines
2.1. Edge-based VPE methods

converge at point \( L \) presenting the line.

![Image of Hough transform](image)

Figure 2.1: Example of the Hough transform.

The CHEVP algorithm has three main steps. The first step extracts an edge map \( E \) of the image with the Canny method. The edge map is used in the next step instead of the image to reduce computation cost.

The second step detects the virtual straight lines of the image by applying the Hough transform on \( E \). The image is divided into horizontal sections to better represent the curve roads. Each section stores the line parameters in its own Hough space, and the straight lines detection is implemented separately based on the edge pixels in each section.

A voting scheme is used to detect the straight lines. Each edge pixel is transformed into a line in Hough space, and then the vote score of each point contained by this line is incremented by one. After the transformation, the accumulated Hough space is then normalized and filtered with a threshold to detect the peak points with high vote scores. The coordinates \((A_l, B_l)\) of the peak point \( L \) are the parameters of the detected line \( l : y = A_l x + B_l \) in the image space. The weight value \( s_l \) assigned to \( l \) is the vote score contributed to the point \( L \) in the accumulated space.

The final step estimates the optimal VP for each section from the intersections of these straight lines using another voting scheme. The intersection of lines \((l, m)\) in the image space is found as the line segment from two corresponding points.
2.1. Edge-based VPE methods

\((L, M)\) in the accumulated space. This line segment is then projected back to the image through the inverse Hough transform as the intersection of the processed pair \(i_{lm}(x_{lm}, y_{lm})\), where \((x_{lm}, y_{lm})\) is the root of the following equation:

\[
\begin{align*}
  y &= A_l x + B_l \\
  y &= A_m x + B_m.
\end{align*}
\]

(2.1)

For each intersection \(i = (x_i, y_i)\), the set of lines that converge at \(i\) is defined as \(R(i)\). The lines \(l\) and \(m\) belong to \(R(i)\) when their intersection coincides with \(i\), as \((x_{lm}, y_{lm}) = (x_i, y_i)\). The vote score of \(i\) is accumulated from the weight values \(s_l\) of the lines contained by \(R(i)\) as follows

\[\text{votes}(i) = \sum_{l \in R(i)} s_l.\]  

(2.2)

This step is repeated for each of the image sections. The intersection which has the highest score is determined as the vanishing point.

In term of performance, CHEVP has fast operation because the VP is voted from a limited number of intersections. The detection has adequate accuracy in simple structured scenes that have significant road borders or lane markers. These markers are detected as strong edges which contribute notable vote scores to the optimal VP in CHEVP algorithm. However, the method is not so accurate when the image lacks those dominant edges, as the case in unstructured scenes [4]. In addition, CHEVP is also affected by the interference from spurious edges in the non-road region of the image, especially in complex scenes [8].

2.1.2 Line segment detection based approach

Recently, She et. al. proposed an edge-based approach which utilizes the LSD algorithm on the Canny edge image [8]. The LSD algorithm, proposed in [10], builds up the line support regions from the edge pixels with the same edge
2.1. Edge-based VPE methods

gradient. The line segments are then validated from these regions with defined thresholds. LSD algorithm has been shown to be more accurate than the Hough transform in detecting straight lines of the image [10].

In the next step, a threshold is applied to the detected line segments to remove short segments and optimize the number of detected straight lines. A straight line \( l : A_mx + B_my + C_m = 0 \), detected with LSD is represented as the vector \((A_m, B_m, C_m)\). The line direction \( \theta_m \) is calculated as \( \theta_m = \arctan \frac{A_m}{B_m} \), and only the lines with more frequent directions are considered in the next step.

For a pair of lines \((l, m)\), the cross product \( p_{lm} = (p_{lm}^a, p_{lm}^b, p_{lm}^c) \) of the vectors \((A_l, B_l, C_l)\) and \((A_m, B_m, C_m)\) is calculated. The intersection of the lines \( l \) and \( m \) is detected as

\[
p_{lm} = (x_{lm}, y_{lm}) = \frac{(p_{lm}^a, p_{lm}^b)}{p_{lm}^c},
\]

where \( x_{lm} \) and \( y_{lm} \) denote the vertical and horizontal coordinates of \( p_{lm} \) in the image plane respectively.

These intersections are considered as the VP candidates, which collect vote score from the image pixels of the line segments. The vote score is based on the distance between the voting pixel and the candidate. In the next stage, a mean filter is used to optimize the vote score map. The VP is detected from the peak of the final vote score map.

With a pixel-wise voting scheme, the accuracy of this method does not solely depend on the dominant edges. However, because the vote score is only based on the position of the voting pixels, the method does not mitigate the interference of the spurious edges, especially in complex scenes.

In general, edge-based methods rely on finding the line segments which represent road borders. In unstructured scenes, e.g. country roads or pedestrian trails, this approach is not so effective because the road borders are hard to detect. Consequently, the recent VPE research has focused on the texture-based approach for unstructured scenes, which estimates the vanishing point from the
2.2. Texture-based VPE methods

texture orientation of image pixels.

2.2 Texture-based VPE methods

Texture-based VPE approaches typically include three basic stages. The first stage estimates the dominant texture orientation at the pixels using orientation filters. The second stage defines the possible VP candidates and their voting region of supporting pixels. The third stage accumulates the VP score for each candidate from the pixels in its dedicated voting region. The estimated orientations are utilized in both the second and third stage. Eventually, the candidate with highest vote score is considered as the VP. These stages are shown in Figure 2.2. This section mainly discusses the texture-based approaches which utilize Gabor filters for orientation estimation.

Figure 2.2: Main stages in texture-based VPE methods.

2.2.1 Global voting scheme approach

Gabor filters have been shown to be accurate in detecting the texture orientation of a grey image [11]. There are Gabor filters with different orientations, scales, and phases. Figure 2.3 shows an example of a Gabor filter bank of 8 orientations and 5 scales.

Rasmussen utilized a Gabor filter bank for texture orientation estimation[4]. The odd and even phases Gabor filters for orientation $\theta$, wavelength $\lambda$, having a
2.2. Texture-based VPE methods

Figure 2.3: Example of a Gabor filter bank with 8 orientations and 5 scales.

size of \( k \times k \) pixels with \( k = \left\lfloor \frac{10\lambda}{\pi} \right\rfloor \), are calculated as

\[
g_{\text{odd}}(x, y, \theta, \lambda) = \exp\left[-\frac{1}{8\sigma^2}(4a^2 + b^2)\right]\sin(2\pi a/\lambda), \tag{2.4}
\]

\[
g_{\text{even}}(x, y, \theta, \lambda) = \exp\left[-\frac{1}{8\sigma^2}(4a^2 + b^2)\right]\cos(2\pi a/\lambda). \tag{2.5}
\]

In (2.4) and (2.5), \( a = x \cos \theta + y \sin \theta, \quad b = -x \sin \theta + y \cos \theta, \quad \sigma = \frac{k}{s}, \quad (x, y) = (0, 0) \) is the kernel center. The square norm of the kernel is then normalized to 1 with the DC components removed. The total energy response \( C \) of a greyscale image \( I \) with the normalized Gabor filters at pixel \( p = (x, y) \) is calculated as

\[
C(x, y) = ||(g_{\text{odd}} \otimes I)(x, y)||^2 + ||(g_{\text{even}} \otimes I)(x, y)||^2. \tag{2.6}
\]

In (2.6), \( \otimes \) denotes the 2-D convolution. There are \( n = 72 \) orientations used: \( \{0, \frac{\pi}{n}, \frac{2\pi}{n}, \ldots, \frac{(n-1)\pi}{n}\} \). The filter wavelength is \( \lambda = 2^{\frac{1}{\log_2(\text{size})}} - 5 \), where \( \text{size} \) is the width of the input image. The texture orientation at pixel \( p \) corresponds to the filter which gives the maximum energy response at the pixel.

To set up the VP candidate region, the method assumes that the vanishing point locates within the image borders. Therefore, the VP candidate region is set as the entire pixels of the image.
2.2. Texture-based VPE methods

For the voting region, the method uses a global voting scheme as shown in Figure 2.4. A voting region \( R(v) \) is set for each VP candidate \( v \), including all the pixels below the candidate \( v \).

![Figure 2.4: Global voting scheme. The voting region \( R(v) \) of the candidate \( v \) includes all the pixels below \( v \). The vote score contributed by voter \( p \) is based on the angle difference \( \gamma \).](image)

The vote score collected from each pixel \( p \) in the voting region is calculated based on the angle \( \gamma \) between the direction \( (p, v) \) and the dominant orientation \( \hat{\theta}(p) \) as

\[
vote(p, v) = \begin{cases} 
1, & \text{if } |\gamma| \leq \frac{\pi}{2\pi}, \\
0, & \text{otherwise}.
\end{cases} \tag{2.7}
\]

The vote score of candidate \( v \) is accumulated as

\[
votes(v) = \sum_{p \in R(v)} vote(p, v). \tag{2.8}
\]

Finally, the candidate with the highest vote score is considered as the vanishing point of the image.

With a comprehensive pixel-wise voting based on texture orientation, this method shows adequate accuracy for unstructured scenes that have neither road borders nor lane markings [4]. However, the computational complexity of this
approach is high for three reasons. Firstly, there are a large number of complex Gabor filters involved in the orientation estimation step. Secondly, the entire pixels of the image are considered as the VP candidates. Finally, all the pixels below a candidate are counted in its voting region.

Another drawback of the global approach is the bias toward candidates in the higher section of the image. Because these pixels have larger voting regions, the non-road elements can have more negative impact on the vote score and reduce the accuracy of the voting process.

### 2.2.2 Local adaptive soft voting approach

Kong et. al. proposed Local Adaptive Soft Voting (LASV) approach in the voting scheme [12] to address the bias issue of the global approach. This method also uses a bank of Gabor filters to estimate the texture orientation. For an orientation $\theta$ and a radial frequency $\omega$, a Gabor filter is defined as

$$g_{\omega,\theta}(x, y) = \frac{\omega}{\sqrt{2\pi c}} e^{-\omega^2(x^2+y^2)/(8c^2)}(e^{i\omega \theta} - e^{-c^2/2}),$$

(2.9)

where $(x, y) = (0, 0)$ is the center of the kernel, $a = x \cos \theta + y \sin \theta$, $b = -x \sin \theta + y \cos \theta$, and $c = 2.2$ [11]. The method considers the filters with 5 scales $\omega = \omega_0 \times 2^k$ (with $\omega_0 = 2.1$ and $k = 0, 1, 2, 3, 4$), and $n = 36$ orientations evenly spaced in the range $[0, \frac{(n-1)\pi}{n}]$.

The complex response $C_{\omega,\theta}$ of a greyscale image $I$ at the pixel $p = (x, y)$ with the normalized Gabor filters $g_{\omega,\theta}$ is

$$C_{\omega,\theta}(x, y) = \text{Re}(I \otimes g_{\omega,\theta}(x, y))^2 + \text{Im}(I \otimes g_{\omega,\theta}(x, y))^2.$$

The response $R_\theta$ for orientation $\theta$ is the average of the complex response $C_{\omega,\theta}$ at all the scales $\omega$. Similar to [4], the texture orientation at pixel $p$ is the orientation corresponding to the filter that gives the maximum average complex response of
2.2. Texture-based VPE methods

all scales at the pixel.

The complex responses of $I$ at pixel $p$ are sorted as $r_1(p) > \cdots > r_{36}(p)$. The confidence in the maximum orientation is set as

$$\text{Conf}(p) = 1 - \frac{\text{Average}(r_5(p), \ldots, r_{15}(p))}{r_1(p)}. \quad (2.10)$$

The confidence values are normalized throughout the image to the range $[0, 1]$. The pixels with a confidence smaller than $T = 0.3$ are not considered in the voting region. Compared to the approach in [4], this approach reduces the computational complexity by optimizing the number of possible voters for each VP candidate.

In the voting scheme, the candidate pool is set as the entire pixels in the upper 90% section of the image. For each candidate pixel $v$, a voting region $R(v)$ is defined as the lower half-disk section of the Gabor response image centered at $v$ as in Figure 2.5. The radius $\rho$ of this half-disk is set at $0.35 \times L$ where $L$ is the diagonal of the image.

![Figure 2.5](image)

Figure 2.5: Local adaptive soft voting scheme. The voting region $R(v)$ of the candidate $v$ is the lower half-disk centered at $v$ with the radius $\rho$. The vote score contributed by voter $p$ is based on the angle difference $\gamma$ and the Euclidian distance between $p$ and $v$.

To calculate the vote score contributed by a pixel $p$ to the candidate $v$, at first the angle $\gamma$ between the direction $(p, v)$ and the dominant orientation $\hat{\theta}(p)$ at $p$ is calculated. The Euclidian distance $d(p, v)$ between $p$ and $v$ is then divided by the diagonal $L$ to find the ratio $d_{pv} = d(p, v)/L$. The vote score is defined as
2.2. Texture-based VPE methods

\[ \text{vote}(p, v) = \begin{cases} \frac{1}{1 + d_{pv} \gamma}, & \text{if } \gamma \leq \frac{5}{1 + 2d_{pv}} \\ 0, & \text{otherwise.} \end{cases} \tag{2.11} \]

Equation (2.11) means vote score is high if pixel \( p \) is close to candidate \( v \) and pixel \( p \) has an orientation similar to the direction \((p, v)\). The vote score of candidate \( v \) is accumulated as in (2.8). Eventually, the candidate that has the highest vote score is considered as the vanishing point.

In this method, the bias issue in the global approach [4] is solved, because the voting region only includes the pixels near the candidate, and the vote score emphasizes the contribution of the closer voters. The computational cost is reduced because there are fewer pixels in the voting region. However, the complexity is still very high due to the large number of Gabor filters involved. Also, the number of VP candidates is not improved from [4]. Another issue with the local approach is clutter, which happens when the local voting zone of the optimal VP does not include high-confident voters. In these cases, the vote score accumulated is not significant enough for the optimal VP to be considered accurately.

Next, we discuss some recent efforts to address the shortcomings of LASV algorithm.

- Bui et. al. proposed improvements for the computational cost of the LASV algorithm [13]. The technique filters the orientation map of the image with fixed thresholds to mitigate the effect of shadows and vertical objects. The number of voters is also optimized in the filtering process. In the voting scheme, the voters are scanned instead of the candidates. A triangular search zone is set upward from a voter \( p \) based on the local orientation at \( p \). Each VP candidate in this search zone is accumulated a vote score as in (2.11).

The revised method in [13] has faster running time compared to the method of [12], because the number of scanned pixels is reduced. However the cluttering issue of the LASV approach still affects the performance of this
2.2. Texture-based VPE methods

- Yang *et al.* utilized a Weber descriptor to calculate the voting confidence while using the same set of Gabor filters as in the LASV approach [12] for orientation estimation [14]. Inspired from the original Weber Local Descriptor [15], the approach defines the differential excitation of the pixels $p$ as follows

$$
\varepsilon(p) = \arctan \left( \frac{I(p) - \bar{I}(p)}{I(p)} \right),
$$

(2.12)

where $I(p)$ is the intensity of pixel $p$ and $\bar{I}(p)$ is the average intensity of all the neighbor pixels of $p$. The voting confidence of $p$ is calculated from $\varepsilon(p)$ as follows

$$
\text{Conf}(p) = \begin{cases} 
\sqrt{\varepsilon(p)}, & \text{if } \varepsilon(p) \geq 0 \\
0, & \text{otherwise.}
\end{cases}
$$

(2.13)

Afterwards, a threshold is applied to filter out the low-confident pixels.

In the next step, a Gabor-based approach similar to the method of [12] is employed to calculate the local orientation on the remaining pixels. Finally, these high-confidence pixels are considered in a line voting scheme motivated by the approach in [16].

The Weber descriptor in equation (2.13) is effective in filtering the spurious edges from the off-road objects, and is shown to offer better accuracy for the voting scheme [14]. However, the complexity of the Gabor filters is still an issue with this approach. The line voting approach of this method is discussed further in the next section.
2.2. Texture-based VPE methods

2.2.3 Optimal Gabor filters and line voting approach

The methods proposed in [4] and [12] both use a large set of Gabor filters, which leads to high complexity and slow operation. Addressing this issue, Moghadam et al. introduced Optimal Local Dominant Orientation Method (OLDOM) and a line voting scheme [16]. This section discusses the original method presented in [16] and the recent enhanced method proposed in [17].

Firstly, to estimate the prominent orientation at each pixel, four Gabor filters with the orientations \( \theta \in \{0, \pi/4, \pi/2, 3\pi/4\} \) are defined. For each orientation \( \theta \), a Gabor filter is calculated as

\[
g_{\theta}(x, y) = \frac{\omega}{\sqrt{2\pi c}} e^{-a^2(4a^2+b^2)/(8c^2)}(e^{i\omega} - e^{-c^2/2}),
\]

where \((x, y) = (0, 0)\) is the kernel center, \( a = x \cos \theta + y \sin \theta \), \( b = -x \sin \theta + y \cos \theta \) and \( c = \pi/2 \). The radial frequency is \( \omega = 2\pi/\lambda \), where \( \lambda = 4\sqrt{2} \) is the spatial frequency.

For a greyscale image \( I \), from the convolution \( \hat{I}_{\theta} = I \otimes g_{\theta} \), the Gabor energy response of \( I \) with the orientation \( \theta \) at the pixel \( p \) is calculated as follows

\[
E_{\theta}(p) = \sqrt{\text{Re}(\hat{I}_{\theta}(p))^2 + \text{Im}(\hat{I}_{\theta}(p))^2}.
\]

The Gabor responses are calculated in (2.15) for four orientations \( \theta \in \{0, \pi/4, \pi/2, 3\pi/4\} \) and sorted into \( E_1(p) > E_2(p) > E_3(p) > E_4(p) \), with the orientations \( \theta_1(p), \theta_2(p), \theta_3(p), \theta_4(p) \) respectively. The joint activity of these four Gabor responses defines the two vectors \( S_1(p), S_2(p) \) with their angles \( \phi_1(p), \phi_2(p) \) as follows

\[
\begin{cases}
\|S_1(p)\| = E_1(p) - E_4(p) \\
\phi_1(p) = \theta_1(p)
\end{cases}
\]
2.2. Texture-based VPE methods

\[
\begin{cases}
\|S_2(p)\| = E_2(p) - E_3(p) \\
\phi_2(p) = \theta_1(p).
\end{cases}
\] (2.17)

The response vector \(V(p)\) is determined as

\[
V(p) = \sum_{i=1}^{n} S_i(p) e^{i\phi_i}.
\] (2.18)

The dominant orientation at pixel \(p\) can be estimated as follows:

\[
\hat{\theta}(p) = \arg(V(p)).
\] (2.19)

Subsequently, a line voting scheme is implemented to estimate the vanishing point. From each pixel \(p\) with orientation \(\hat{\theta}(p)\), a line \(r_p = (p, \hat{\theta}(p))\) is rendered upward following the direction of \(\hat{\theta}(p)\). The line terminates at its intersection with the borders of the image, and \(D_p\) is the Euclidean distance between \(p\) and the intersection. Each line is assigned the weight value of \(\sin(\hat{\theta}(p))\) to reduce the interference of the horizontal edges. Also, to mitigate the bias issue of the global approach [4], each pixel \(z\) contained by \(r_p\) is assigned a distance function \(D_p(z)\), which is calculated from the Euclidian distance \(d(p, z)\) between \(p\) and \(z\) as

\[
D_p(z) = e^{-d_{pz}^2/2\delta^2},
\] (2.20)

where \(\delta^2 = 0.25\) and \(d_{pz} = d(p, z)/D_p\). The formula (2.20) means the pixels closer to \(p\) receive more contribution than other ones along the line.

Eventually, each pixel \(z\) of the line \(r_p\) is credited a vote score determined by

\[
vote(p, z) = \sin(\hat{\theta}(p))D_p(z).
\] (2.21)

After rendering the lines for all voters, the pixel with the highest accumulated score is considered as the vanishing point.
2.3. Local orientation estimation methods

The method proposed in [16] has lower complexity in orientation estimation step compared to the method in [4] and [12], since there are only four Gabor filters involved. However, rendering the virtual lines from a large number of voters leads to more computational cost. Furthermore, the accuracy of this method is affected by the interference of the spurious edges, shadows, and strong illumination [17].

Recently, Shi et al. proposed an enhanced method based on OLDOM technique [17]. In this method, the orientation confidence at pixel \( z \) is defined as follows:

\[
\text{Conf}(p) = \begin{cases} 
1 - \frac{E_4(p)}{E_1(p)}, & \text{if } E_1(p) > E_0 \\
0, & \text{otherwise},
\end{cases}
\]

(2.22)

where \( E_0 \) is a set threshold, \( E_1(p) \) and \( E_4(p) \) are the Gabor responses at pixel \( p \) as mentioned in (2.16) and (2.17). The texture orientations are only calculated for the pixels with the confidence higher than a desired threshold \( C_0 \). These pixels are considered as the voters in the same line voting scheme of OLDOM [16]. The joint activity of the Gabor energy responses in (2.16) and (2.17), as well as the voting function in (2.21), are replaced with trained look-up tables to achieve faster operation.

To reduce the computation cost, this method reduces the number of voters based on the orientation confidence. The implementation of look-up tables also improves the overall processing time. However, the issues with the spurious edges and shadows are still not addressed in this method.

2.3 Local orientation estimation methods

In general, the Gabor-based algorithms for texture orientation estimation have high complexity. This issue hinders the use of the texture-based VPE methods in real-time applications. This section investigates the texture-based approaches that utilize different orientation estimation algorithms other than Gabor filters.
2.3. Local orientation estimation methods

2.3.1 Haar-like box filters

In [18], Miksik approximated the Gabor filter bank with a set of Haar-like box filters. A Gabor kernel is decomposed into a linear function of various Haar-like box filters [19] as in Figure 2.6.

A single Haar-like box filter is formed as follows

\[
H_{\text{single}}(x, y) = \begin{cases} 
1, & \text{if } x \in [x_0, x_0 + w - 1] \text{ and } y \in [y_0, y_0 + h - 1] \\
0, & \text{otherwise.} 
\end{cases} 
\] (2.23)

where \((x_0, y_0)\) are the Euclidean coordinates of the top left corner; \(w\) and \(h\) are the width and height of the white box respectively.

A vertically symmetric Haar-like box filter is calculated as

\[
H_{\text{vertical}}(x, y) = \begin{cases} 
1, & \text{if } x \in [x_0, x_0 + w - 1] \text{ and } y \in [y_0, y_0 + h - 1] \\
1, & \text{if } x \in [x_c - x_0 - w + 1, x_c - x_0] \text{ and } y \in [y_c - y_0 - h + 1, y_c - y_0] \\
0, & \text{otherwise.} 
\end{cases} 
\] (2.24)

where \((x_c, y_c)\) is the center of white box number \(c\). The horizontally symmetric box filter is calculated in a similar manner as in (2.24).

In this method, a set of single and symmetric Haar-like box filters is calculated. A dictionary \(D = \{h_1, h_2, \ldots, h_N\}\) is defined afterwards, where \(h_i\) are the column vectors formed by reshaping the calculated filters.

A Gabor filter for scale \(\omega\) and orientation \(\theta\) defined in (2.9) is approximated as


2.3. Local orientation estimation methods

follows

\[ g_{\omega,\theta} \approx \hat{g}_{\omega,\theta} = \sum c_i h_i, \]

(2.25)

where \( c_i \) are the coefficients found with Optimized Orthogonal Matching Pursuit (OOMP) algorithm [20]. The composition of single and symmetric boxes consisted in \( D \) is based on the orientation, scale, and phase of the Gabor kernel.

Consequently the convolution of the greyscale image patch \( P \) with the Gabor filter is estimated as

\[ C_{\omega,\theta} \approx \sum \alpha_i (h_i \times P), \]

(2.26)

where \( \alpha_i \) is normalized from \( c_i \) with the DC component removed. The texture orientation is estimated from the responses \( C_{\omega,\theta} \) of 5 scales and 36 orientations in a similar manner as in LASV method [12].

In the voting step, superpixels are implemented to optimize the candidate pool. The image is divided into equal size \( k \times k \) square zones called superpixels. A histogram of orientation is calculated for the pixels in each superpixel. The superpixels which have the maximum histogram count higher than a trained threshold are included in the candidate pool. The remaining steps in the voting scheme are similar to the LASV method [12].

The main contribution of the orientation estimation algorithm in [18] is the lower complexity of computing the energy responses in (2.26) compared to the Gabor-based algorithm [4, 12]. Also, employing the superpixels reduces the number of VP candidates in the voting scheme and improve processing time. However, this method has to sacrifice the accuracy to achieve fast operation, because the Haar-like filters and superpixels are based on approximation. In addition, the LASV algorithm [12] used in the voting step is still affected by the cluttering issue, which is not addressed in this method.
2.3. Local orientation estimation methods

2.3.2 Generalized Laplacian of Gaussian filters

In [21], Kong et al. implemented generalized Laplacian of Gaussian (gLoG) filters in estimating the pixel orientation. For an orientation $\theta$, with the vertical and horizontal scales $\sigma_x, \sigma_y$ respectively, a gLoG filter [22] is defined as follows

$$G(x, y) = G_0 e^{-(ax^2 + 2bxy + cy^2)},$$

(2.27)

where

$$a = \frac{\cos^2 \theta}{2\sigma_x} + \frac{\sin^2 \theta}{2\sigma_y},$$

$$b = -\frac{\sin 2\theta}{4\sigma_x^2} + \frac{\sin 2\theta}{4\sigma_y^2}$$

(2.28)

$$c = \frac{\sin^2 \theta}{2\sigma_x} + \frac{\cos^2 \theta}{2\sigma_y}.$$

The values of $\sigma_x$ and $\sigma_y$ are calculated as

$$\begin{cases}
\sigma_x, \sigma_y \in \mathbb{N}, \sigma_x > \sigma_y, \\
\sigma_x \in [\sigma_x^{\text{min}}, \sigma_x^{\text{max}}]; \sigma_y \in [\sigma_y^{\text{min}}, \sigma_y^{\text{max}}], \\
\sigma_x^{\text{min}} = \sigma_y^{\text{min}} + 1, \\
\sigma_x^{\text{max}} = 8, \sigma_y^{\text{min}} = 2
\end{cases}$$

(2.29)

For each orientation $\theta_i$ chosen from $n$ orientations evenly spaced in $[0, \frac{(n-1)n}{n}]$, a set $F_i$ of gLoG kernels is generated based on all the combinations of $\{\sigma_x, \sigma_y, \theta_i\}$. The response map $E_i$ of the greyscale image $I$ for the orientation $\theta_i$ is calculated as

$$E_i = \sum_{G_l \in F_i} |G_l \otimes I|.$$ 

(2.30)

The orientation which has the maximum response at a pixel $z$ is estimated as the dominant orientation $\hat{\theta}(p)$.

In the next step, the method detects the blob structures of the image using the
2.3. Local orientation estimation methods

calculated gLoG kernels. The voting region in this method includes all the pixels in the blob structures below the VP candidate. The vote score contributed by pixel \( p \) in the voting region of candidate \( v \) is defined as

\[
vote(p, v) = \begin{cases} 
\exp(-|\gamma| \times d_{pv}), & \text{if } |\gamma| \leq \delta_0 \\
0, & \text{otherwise}
\end{cases}
\]  

(2.31)

where \( \gamma \) and \( d_{pv} \) are the parameters discussed in (2.11). The threshold \( \delta_0 \) is selected to remove the impact of the pixels that have the texture orientations different from the direction \( (p, v) \). Eventually, the candidate with maximum vote score is considered as the vanishing point.

The gLoG filters provide the accuracy comparable with the Gabor filters in estimating texture orientation, while the blob detection provides optimizations for the voting region [21]. The voting scheme uses a global adaptive approach to address the issue of cluttering in LASV [12]. However, the gLoG filters function in (2.27) still has high complexity. The blob detection also adds more computation cost to the algorithm. Furthermore, the global approach in the voting scheme increases the number of voting pixels and leads to the overall slow operation.

2.3.3 Color tensor features

From a single image, the information for VPE is sparse. Meanwhile, most of the available methods only consider the intensity image and do not adopt the information from color channels. Addressing this concern, Le et al. utilized the color tensor based features for orientation estimation in a texture-based approach [23]. The technique also uses color edges of the image to determine the VP voters and optimize the voting scheme.

Weijer et al. proposed the color tensor based approach in calculating the invariant features of a color image [24]. For an image \( I \) with three color channels \( I_k (k = 1, 2, 3) \), using the Gaussian smoothing filters \( g \), the elements of the color
2.3. Local orientation estimation methods

tensor structure are computed as

\[
T_{xx} = g \otimes \left[ \sum D_{k,x} \circ D_{k,x} \right], \\
T_{yy} = g \otimes \left[ \sum D_{k,y} \circ D_{k,y} \right], \\
T_{xy} = g \otimes \left[ \sum D_{k,x} \circ D_{k,y} \right],
\] (2.32)

where \( D_{k,x} \) and \( D_{k,y} \) are the derivatives of \( I_k \) along the horizontal and vertical direction, respectively. In (2.32), \( \otimes \) denotes the 2-D convolution and \( \circ \) stands for the Hadamard product. The color texture orientation map \( O \) of the image is calculated in [24] as follows

\[
O = \frac{1}{2} \arctan\left( \frac{2T_{xy}}{T_{xx} - T_{yy}} \right). 
\] (2.33)

The derivative energies of the pixels in their orientations are calculated as

\[
\Lambda = \frac{1}{2}(T_{xx} + T_{yy} + \sqrt{(T_{xx} - T_{yy})^2 + 4T_{xy}^2}).
\] (2.34)

The color edge map \( E \) is then calculated from \( \Lambda \) using an adapted Canny edge detection algorithm [25].

The color edge map \( E \) is used to optimize the voting region of a pixel wise voting scheme. For a candidate \( v \), only the edge pixels in \( E \) which locate below \( v \) are included in the voting process. The vote score of the voter \( p \) to the candidate \( v \) is calculated using (2.11), based on the orientation map \( O \). The candidate which accumulates highest vote score is detected as the vanishing point.

In this method, the number of voters is significantly reduced because only the edge pixels are involved. The orientation map \( O \) calculated in (2.33) shows comparable accuracy with fast operation because of the low complexity of the Gaussian smoothing filters [24]. While using the same vote score as LASV [12], the color-based method uses the color edge map to optimize the voting region. However, as all the pixels in the image are chosen as VP candidates, the running
time of the algorithm is still considerably high [23].

2.4 Deep learning-based VPE methods

The texture-based VPE methods are proven to have adequate robustness in various scenes. However, these methods still rely on heuristics and use fixed parameters, as well as voting functions. For complex environments, these less adaptable approaches struggle to detect the VP consistently. To address this issue, some recent studies of VPE are based on deep learning approach.

In [26], Borji proposed a VPE method utilizing convolutional neural networks (CNN). There are two deep networks employed, including AlexNet and VGGnet. A data-driven learning approach is implemented, in which the CNN is trained in an end-to-end manner in order to estimate the VP coordinates.

For the implementation of deep neural networks, a large amount of training data is essential. This method uses 37497 images (resized to 300 × 300 pixels) of various weather and road conditions, with the ground truth VP marked manually. In this method, the training images are considered as $n \times n$ grid map ($n = [10, 20, 30]$), in which the grid cell which contains the VP has the label 1.

AlexNet and VGGNet are trained to map an input image into the grid cell containing VP, which is one of the $n \times n$ grids. The output layers thus have $n \times n$ neurons. The training process is stopped after 40 epochs.

The main advantage of this approach is that it is data-driven and does not have fixed parameters and heuristic features. However, the deep networks result in a very high computational complexity due to the nature of CNN. Also, both the outputs of these models are a grid cell with the VP, therefore the exact coordinates of the VP are still approximated and not as reliable as traditional approaches. Another disadvantage of this approach is that it does not consider the scenes when the VP is not in the image.

Recently, Shuai et al. [27] provided some enhancements to the AlexNet ap-
proach. The network output in this method is the exact coordinates of the VP. There are three main ideas of their method. Firstly, the ReLU activation function of the AlexNet is replaced by the \( \tanh \) activation function. Secondly, more fully connected layers are added to the networks. Finally, the loss function is designed as the square of Euclidean distance of the network output and the ground truth VP.

The training data of [27] has 5548 images, each resized to 224 \( \times \) 224 pixels. The ground truth VP is annotated manually. The network is trained to map an input image into one pixel of the 224 \( \times \) 224 grid, which means there are 50176 neurons in the output layer.

Although the method in [27] has configured the AlexNet to better suit the VPE problem, it adds a large number of neurons in the output layers. The training process thus needs more epochs to converge, and adds difficulty to the end-to-end training. The customized AlexNet is still a deep network which requires slow processing time in practical applications.

### 2.5 Chapter summary

Several approaches have been proposed for VPE. The shortcomings of the current solutions are summarized as follows:

- The estimation techniques based solely on the edge segments are not quite effective for the unstructured scenes where the road borders are hard to detect.

- The texture-based techniques based on Gabor filters have high complexity due to the reliance on the large number of filters. Addressing this issue, the method utilizing Haar-like box filters sacrifices the accuracy for faster performance, but it is still not sufficient for real-time applications.

- The voting schemes of the texture-based methods consider most of image
pixels as the VP candidates. This leads to the high computation cost in the voting scheme. The approach with LASV tries to reduce the size of the voting region, but it leads to cluttering issue when the VP locates near the pixels with the low orientation confidence.

- The deep learning approaches employ deep CNNs and large datasets to train. While these approaches do not rely on heuristic parameters, the computation cost of the deep neural networks is very high.

From the above reviews, the promising directions in VPE research can be listed as follows:

- Investigate texture orientation estimation algorithms with the requirements of low complexity and robustness in various image conditions. In this research, we focus on the color-based features, such as color texture orientation and color Canny edge map.

- Utilize the low computational cost edge features to enhance the current texture-based methods. The main focus of our research is to optimize the search space for VP candidates and voters based on both edge-based and texture-based features.

- Apply an adaptive voting scheme of VPE methods. The threshold parameters and voting function of the methods can be determined via a learning process to achieve fast and consistent performance.
This page intentionally contains only this sentence.
Chapter 3

Proposed VPE method

Chapter contents

3.1 Stage 1: Extracting VP features ................. 32
   3.1.1 Color tensor features .................. 33
   3.1.2 Texture orientation and color edge maps .... 35

3.2 Stage 2: Identifying VP search spaces .......... 36
   3.2.1 Vanishing point candidates ............... 37
   3.2.2 Vanishing point voters ................ 39

3.3 Stage 3: Calculating VP scores ................. 41
   3.3.1 VP voting with weighting method .......... 42
   3.3.2 Bayesian classifier for the propose adaptive voting function 44
   3.3.3 VP score map and post-processing .......... 45

3.4 Chapter summary .................................. 46
3.1 Stage 1: Extracting VP features

In this chapter, we propose a novel VPE algorithm for color input images. Compared to intensity images, color images have more information from the color channels and are utilized in a wider range of applications. We employ color tensor based approach to extract VP features and propose a voting scheme to determine the VP.

The VPE method proposed in this chapter consists of three stages (Figure 3.1): i) extracting important features for VPE from the input image, ii) identifying VP search spaces, and iii) calculating VP scores for detection.

![Diagram of vanishing point estimation method]

Figure 3.1: An overview of the proposed vanishing point estimation method.

An example of the proposed VPE method applied on an input image is illustrated in Figure 3.2, in which the results of each above mentioned stage are also available.

3.1 Stage 1: Extracting VP features

In this stage, we extract local texture orientation and color edge maps using the color tensor features, which are proven to be invariant to color coordinate transformations [24]. This characteristic of color tensors makes the proposed method robust to different color spaces of the input image. For this thesis, the RGB color space is considered due to its popularity, so that the proposed VPE method can be implemented in a wide range of applications. The method is still applicable for any other color spaces.
3.1. Stage 1: Extracting VP features

3.1.1 Color tensor features

Given an RGB input image, color tensors are applied to find local texture orientations and color edges. We utilize the shadow-shading quasi-invariant method [24] for color tensor calculation to mitigate noises and interferences from complex lighting conditions in road scenes.
3.1. Stage 1: Extracting VP features

Let $I_k$, where $k = 1, 2, 3$, be the image color channels, and $g$ be a 2-D Gaussian filter. As in [6], the color tensor structure of the image is given by

$$T = \begin{pmatrix} T_{xx} & T_{xy} \\ T_{xy} & T_{yy} \end{pmatrix},$$  \hspace{1cm} (3.1)$$

where

$$T_{xx} = g \otimes \sum_{i=1}^{3} D_{k,x} \circ D_{k,x},$$  

$$T_{yy} = g \otimes \sum_{i=1}^{3} D_{k,y} \circ D_{k,y},$$  \hspace{1cm} (3.2)$$

$$T_{xy} = g \otimes \sum_{i=1}^{3} D_{k,x} \circ D_{k,y}.$$  

In (3.2), $D_{k,x}$ and $D_{k,y}$ are the derivatives of $I_k$ along the horizontal and vertical directions respectively, $\otimes$ denotes the 2-D convolution and $\circ$ denotes the Hadamard product. For the shadow-shading quasi-invariant method, the derivative $D_{k,x}$ and $D_{k,y}$, are calculated with Gaussian standard deviation filters $d_k (k = 1, 2, 3)$ [28]. In this stage, we use the same filter size $g \times g$ for $d_k$ and $g$.

For the color tensor matrix $T$, we find the eigenvalues and eigenvectors as functions of $T_{xx}, T_{yy}$ and $T_{xy}$ using simple algebra. The eigenvalues of $T$ are given by

$$\Lambda_1 = \frac{1}{2} \left[ T_{xx} + T_{yy} + \sqrt{(T_{xx} - T_{yy})^2 + 4T_{xy}^2} \right],$$  

$$\Lambda_2 = \frac{1}{2} \left[ T_{xx} + T_{yy} - \sqrt{(T_{xx} - T_{yy})^2 + 4T_{xy}^2} \right].$$  \hspace{1cm} (3.3)$$

The eigenvectors are $v_1 = (\cos \Theta_1, \sin \Theta_1)$ and $v_2 = (\cos \Theta_2, \sin \Theta_2)$, which have
the directions \( \Theta_1 \) and \( \Theta_1 \) calculated as

\[
\Theta_1 = \frac{1}{2} \arctan\left( \frac{2T_{xy}}{T_{xx} - T_{yy}} \right), \\
\Theta_2 = \Theta_1 + \frac{\pi}{2}.
\]  

(3.4)

3.1.2 Texture orientation and color edge maps

The local texture orientation and color edge maps are calculated from the eigenvectors and eigenvalues. In (3.3) and (3.4), \( \Theta_1 \) represents the dominant local orientation, and \( \Lambda_1 \) represents the gradient energy along the dominant orientation [24]. The texture orientation map \( O \) is then calculated as

\[
O = \Theta_1 = \frac{1}{2} \arctan\left( \frac{2T_{xy}}{T_{xx} - T_{xy}} \right).
\]  

(3.5)

**Color Canny edge detection.** Note that \( \Lambda_1 \) contains thick edges that include the edge pixels and the neighbor pixels, and the aim is to find the edge pixels only. The Canny method [25] is applied on the gradient energies \( \Lambda_1 \) to find the edge pixels.

![Figure 3.3: Extracting edge pixels with non maximum suppression.](image)

First, we perform non-maximum suppression on \( \Lambda_1 \) to find the pixel with the maximum gradient energy (edge strength) in an edge. For computation efficiency, we employ a technique without interpolation. Each 3 \( \times \) 3 grid of pixels is divided into 8 sections, see Figure 3.3. If the orientation of pixel \( l \) is in the \( [-\frac{\pi}{8}, \frac{\pi}{8}] \) range,
3.2. Stage 2: Identifying VP search spaces

we compare the edge strength of $I$ with that of the pixels included in this range ($m$ and $n$ in Figure 3.3). If $I$ has the maximum edge strength, we keep the pixel. Otherwise, we consider it as a background pixel by setting the edge strength of $I$ to zero. The same approach is applied for other ranges of $I$, in which other pairs of pixels are considered. Eventually, the edge strength map $E$ is obtained after non-maximum suppression.

Second, we normalize the edge strength of $E$ by dividing its maximum value and then apply an edge threshold $\alpha$ (with $0 \leq \alpha \leq 1$). Pixels with normalized edge strength larger than $\alpha$ are included in the resulted color edge map $B$. Examples of the extracted feature maps in this section are provided in Figure 3.4. We will use $O$, $B$ and $E$ in the next stage.

Compared to traditional texture-based VPE methods, the approach using color tensors utilizes a Gaussian filter to calculate the orientation map, instead of complex Gabor filters. The proposed method therefore has significant faster computation time in this stage. However, the size $g$ of the Gaussian filter must be optimized, so that the accuracy of the orientation map can be maintained.

3.2 Stage 2: Identifying VP search spaces

To estimate the vanishing point, a pixel-wise voting scheme is employed. For an input image, a set of pixels are defined as the VP candidates, and for each candidate another set of pixels are defined as its VP voters. These set of pixels are considered as VP search spaces. Let $\mu$ be the size of the VP candidate search space, $\psi$ be the size of the VP voter search space, the complexity of the voting scheme can be calculated as $\Omega = \mu \times \psi$. Without any optimization, $\mu$ and $\psi$ are equal to the number of pixels in the input image. Therefore, $\Omega = (H \times W)^2$, where $H$ and $W$ are respectively the height and width of the input image.

In this section, we propose two new strategies to optimize VP search spaces and reduce the value of $\Omega$, based on the local texture orientation and color edge
3.2. Stage 2: Identifying VP search spaces

Figure 3.4: Extracting local texture orientation and color edge maps. First row: Input images with texture orientation imposed. Second row: Color edge strength maps $E$. Third row: Color edge maps $B$. 

maps calculated in the previous stage.

3.2.1 Vanishing point candidates

Our strategy is based on the observation that intersections of straight lines provide a good initial estimate of the vanishing point, because most pixels of the main road borders or road clues are included in those straight lines, see Figure 3.5. Hence, we construct the search space for VP candidates based on the intersections of these straight lines.

**Straight Line Segments.** We apply the Hough transform [9] on the edge strength map $E$ in the input image. Instead of the (binary) edge map $B$, the edge strength map $E$ is employed to take into account weak edges, which in unstructured scenes may contain important road clues (such as vehicle marks on the road).

The edge pixels $p_k(x_k, y_k)$ in $E$ are transformed to the lines $\zeta_k : a = \frac{y_k}{x_k} - \frac{1}{x_k} b$ in
3.2. Stage 2: Identifying VP search spaces

Figure 3.5: Example of detected straight line segments. The road border pixels are shown in red.

Hough space. The vote score of each point in Hough space is initially set to zero. For each point \((A, B)\) contained by the line \(\zeta_k\), the vote score is increased. The score is accumulated from all the transformed lines. The resulted Hough space is then normalized. The line segments shorter than a threshold \(\xi\) are filtered to reduce the computation cost. The peak points with higher vote scores are then detected. In Hough space, the coordinates \((A_l, B_l)\) of the peak point \(L\) are the line parameters of the detected line \(l : y = A_l x + B_l\) in the image space. The vote score \(s_l\) of point \(L\) in the accumulated space is the pixel count \(s_l\) assigned to the line \(l\). This value is also the length of the line segment from \(l\) that presents in the image.

We consider only the line segments within the image for this step. Furthermore, the segments that have vertical and horizontal angles are removed to reduce the computation time. The horizontal lines do not contribute to VP, and the vertical lines often come from background object such as trees or lamp posts, which produce interferences to the VPE process.

The intersection of a pair of lines \((l, m)\) in the image is found by pairing the corresponding points \((L, M)\) in the accumulated Hough space. This line segment is then projected back to the image through the inverse Hough transform by
3.2. Stage 2: Identifying VP search spaces

solving following equation:

\[
\begin{align*}
  y &= A_l x + B_l \\
  y &= A_m x + B_m, 
\end{align*}
\]

(3.6)

**Intersection points neighborhood.** The neighbor pixels of these intersections are included in the VP candidates search space. These pixels are shown to have high probability to include the image VP, based from the results in [3] and the concept of proximity and connectedness [29].

Centered on each intersection, we construct a square patch of size

\[ R = \tau \times L, \]

(3.7)

where \( L \) is the image diagonal and \( \tau \) is the search threshold. The VP candidate search space is the union of all square patches. The square shape, instead of other shapes, is used because it is more computation-efficient to process. The search region is proportional to the size of the input image, and is regulated by \( \tau \) whose value is determined using a training set.

Figure 3.6 provides examples of the VP candidate search space, with the ground-truth VP superimposed. The ground-truth VPs are included in the VP candidates of all the images in Figure 3.6. The number of VP candidates \( \mu \) is significantly smaller than the total number of pixels in the input images, therefore the complexity \( \Omega \) of the voting scheme is reduced. The search threshold \( \tau \) should be optimized so the best VP is not excluded from the VP candidates, and the value of \( \mu \) is as low as possible.

### 3.2.2 Vanishing point voters

To increase speed and accuracy, it is also necessary to optimise the VP voters. To define VP voters, we first consider edge pixels (represented by \( B \)) located below a
3.2. Stage 2: Identifying VP search spaces

VP candidate. This approach is based on the fact that VP is always located above the main road borders in normal perspective of vehicle cameras or pedestrians. Some examples are shown in Figure 3.7

However, the voter space still contains spurious edge pixels in the non-road regions which will influence the vote score. For example for the images in Figure 3.8, the edge pixels detected from the trees are considered in the voter space and can interfere the VP voting process. To reduce this interference, we rely on the observation that road borders contain edge pixels of significantly similar orientations, and spurious edges include pixels of non-uniform orientations. The orientation map $O$ is utilized to construct the histogram of local orientations for
the input image. The orientation range $[0, \pi]$ is divided into $n = 18$ equal size bins, and the orientations belonging to the low count bins are identified. Edge pixels with local orientations included in these low count bins are removed from the VP voter search space. Similar to the VP candidates, we also remove the pixels of vertical and horizontal orientations in the VP voter search space.

![Figure 3.8: Examples of VP voters optimization. First row: Input images with texture orientation imposed. Second row: Color edge maps $B$ as the initial voter space, with some spurious edge pixels marked. Third row: Optimized VP voter search space.](image)

Some examples of the optimized VP voter search space are presented in Figure 3.8. It is notable that the reduced number of voter in the search space $\psi$ also resulted in lower value of the complexity $\Omega$, which leads to faster computation time in the voting scheme.

### 3.3 Stage 3: Calculating VP scores

In this section, we calculate a VP score map from the VP candidates and voters search spaces, based on texture orientation features. Instead of using a pre-defined voting function, we improve the robustness of the voting scheme by integrating
3.3. Stage 3: Calculating VP scores

a simple Bayesian classifier to adaptively adjust the voting function.

3.3.1 VP voting with weighting method

In general, each VP candidate collects vote scores from its VP voters. Consider a VP candidate \( c \) and a VP voter \( v \). Let \( f(v, c) \) be the vote score contributed by voter \( v \) to candidate \( c \). The accumulative VP score for VP candidate \( c \) is calculated as

\[
S(c) = \sum_v f(v, c). \tag{3.8}
\]

The candidate with the highest accumulative VP score \( S(c) \) is considered to be the image vanishing point.

The key task now is to define the voting function \( f(v, c) \). Let \( O(v) \) be the local orientation at \( v \). Let \( \phi(v, c) \) be the angle of the line connecting \((v, c)\). Let \( \Delta(v, c) \) be the angle difference, \( \Delta(v, c) = O(v) - \phi(v, c) \). Let \( d(v, c) \) be the Euclidean distance between \( v \) and \( c \). To improve consistency with different image sizes, we use the normalized distance \( \bar{d} = d(v, c)/L \), and the normalized angle difference \( \bar{\Delta} = \Delta(v, c)/\delta \), where \( \delta \) is the angle threshold. The computation of these variables is illustrate in Figure 3.9.

![Figure 3.9: Computation of the VP vote score. Candidate \( c \) collects vote scores from several voters \( v \). The vote score depends on the Euclidean distance \( d(v, c) \) and the angle difference \( \Delta \).](image.png)
3.3. Stage 3: Calculating VP scores

The VP score that the voter $v$ contributes to the VP candidate $c$ depends on the relative position of $(v, c)$. In Figure 3.9, this relative position is represented by the normalized distance $\bar{d}$ and the angle difference $\bar{\Delta}$. Therefore, we separately study the contribution of $\bar{d}$ and $\bar{\Delta}$ to the voting function $f(v, c)$, represented by weight functions $w_1(\bar{d})$ and $w_2(\bar{\Delta})$ respectively. The proposed voting function in this chapter is formed as the product of these two weight functions:

$$f(v, c) = \begin{cases} w_1(\bar{d}) \times w_2(\bar{\Delta}) & \text{if } \bar{\Delta} \leq 1, \\ 0 & \text{otherwise}, \end{cases}$$

(3.9)

To determine the weight functions $w_1(\bar{d})$ and $w_2(\bar{\Delta})$, we consider the following observations in [12]:

- The voting function should be high if the local orientation at $v$ has a similar direction as the line connecting $(v, c)$. In other words, the VP score is higher when the angle difference $\Delta$ is lower.

- A voter $v$ closer to candidate $c$ should have a higher confidence score. Therefore, the VP score increases when the distance $d$ decreases.

Based on the above observations, $w_1(\bar{d})$ and $w_2(\bar{\Delta})$ should be decreasing functions. We then propose the following weighting functions$^1$:

$$w_1(\bar{d}) = \exp (-\bar{d}),$$
$$w_2(\bar{\Delta}) = \frac{1}{1 + \bar{\Delta}}.$$  

(3.10)

Hence, the voting function in (3.9) can be calculated as

$$f_1(v, c) = \begin{cases} \frac{\exp (-\bar{d})}{1 + \bar{\Delta}} & \text{if } \bar{\Delta} \leq 1, \\ 0 & \text{otherwise}. \end{cases}$$

(3.11)

The impact of $w_1(\bar{d})$ is the main concern of the next section.

---

$^1$An analysis for these weighting functions is given in Section 4.4.
3.3. Stage 3: Calculating VP scores

3.3.2 Bayesian classifier for the propose adaptive voting function

For the texture-based methods (reviewed in Chapter 2), the voting functions with a similar form as (3.11) are able to provide adequate accuracy in most scenes. However, this approach is not robust to the false road clues in the road region, such as obstacles or tree shadows. For such images, those false clues tend to accumulate higher VP score to their candidates than a dominant road borders (long edges), whose vote scores should be emphasized because they are the main contributors to the correct VP.

On the other hand, the approaches that mitigate the distance factor \( w_1 \tilde{d} = 1 \) are able to emphasise the contribution of the dominant edges \([4]\). The proposed voting function in (3.9) following this approach can be calculated as

\[
    f_2(v, c) = 1 \times w_2(\tilde{\Delta}) = \frac{1}{1 + \tilde{\Delta}}. 
\]

(3.12)

However, this approach tends to favor the VP candidates at the higher region of the image and shows inconstancy when the ground-truth VP is located at the lower region of the image \([12]\).

To summarize, for some images \( f_1(v, c) \) is a better voting function than \( f_2(v, c) \), and for other images \( f_2(v, c) \) is a better voting function than \( f_1(v, c) \). To adaptively select \( f_1(v, c) \) or \( f_2(v, c) \), we propose to adopt a simple classifier based on the Bayesian decision rule for minimum cost \([30]\).

The abovementioned issues in both of the voting functions in (3.11) and (3.12) are related to the dominant edges. Therefore, for the classifier we consider the longest line segment detected in the Stage 2 (Section 3.2.1). For each input image, two features are computed. Let \( s \) be the longest line segment detected from the input image. The first feature is the the length of \( s \), normalized against the image diagonal \( L \):

\[
    \tilde{I} = \frac{|s|}{L}. 
\]

(3.13)
The second feature is the angle between $s$ and the vertical axis, normalized by the angle threshold $\delta$:

$$\tilde{\gamma} = \frac{\angle s}{\delta}. \quad (3.14)$$

Note that these two features are already detected in Section 3.2.1. Using a training set of input images and labeled vanishing points, we estimate the class-conditional probability density functions $p_1(\tilde{l}, \tilde{\gamma})$ and $p_2(\tilde{l}, \tilde{\gamma})$ that $f_1(v, c)$ or $f_2(v, c)$ performs better than the other, respectively. For a new test image with features $\tilde{l}$ and $\tilde{\gamma}$, the voting function $f_1(v, c)$ is used if

$$\frac{p_1(\tilde{l}, \tilde{\gamma})}{p_2(\tilde{l}, \tilde{\gamma})} \geq \beta, \quad (3.15)$$

where $\beta$ is a threshold.

The class-conditional pdfs can be estimated in different ways, e.g. see [30]. In this chapter, we use a 2-D histogram approach with a total of $N^2$ bins. $\tilde{l}$ and $\tilde{\gamma}$ are quantized into $N$ bins of each dimensions of the histogram. Initially, the values in the histogram bins are set at zeros. We run the proposed VPE using solely $f_1(v, c)$ or $f_2(v, c)$ on a training image, whose $\tilde{l}$ and $\tilde{\gamma}$ are calculated with VPE. The error performance is evaluated for both cases. If using $f_1(v, c)$ provides more accuracy, the corresponding histogram bin $p_1(\tilde{l}, \tilde{\gamma})$ is updated and vice-versa. We prefer $f_1(v, c)$ in the event of equal accuracy. A large set of images is dedicated for training this classifier.

### 3.3.3 VP score map and post-processing

The VP score map $S$ calculated after the voting scheme normally contains multiple local maximums, which can locate close to the global maximum of the score map. In the final step we apply mean filter technique to $S$ to emphasize the local maximums that locates close to others, smooth the VP score map, and eventually improve the consistency.
3.4 Chapter summary

Generally, the VP candidates located near other high-score candidates have the higher probability to be the VP, following the concept of proximity and connectedness [29]. Therefore, a mean filter is used to accumulate more VP score for such candidates. The filter has similar size to the square patch proposed in Stage 2.

Finally, we detect the candidate with the highest accumulative VP score in $S$ as the VP of the input image. Some examples of the VP score maps and detected VPs are provided in Figure 3.10. An example of the proposed VPE method applied on an input image is illustrated in Figure 3.2.

![Figure 3.10: Examples of VP vote score maps. First row: Input images. Second row: VP score maps with the estimated VP marked (red square).](image)

3.4 Chapter summary

A new method is proposed for vanishing point estimation in unstructured road scenes. The texture orientations and the edge features of the input image are extracted using color tensor analysis. The search spaces of vanishing point candidates and voters are optimized based on line detection and neighborhood pixel strategies. An adaptive voting scheme is proposed for the VPE algorithm in unstructured road scenes. The VP voting score includes proposed weight functions, which are adjusted adaptively using a Bayesian classifier. After post-processing,
the VP candidate with highest VP score is chosen as the VP of the input image.
This page intentionally contains only this sentence.
Experiments and Analysis

Chapter contents

- 4.1 Image dataset ............................................ 50
- 4.2 Evaluation methods .................................. 51
- 4.3 Analysis of voting algorithm parameters ............. 53
- 4.4 Analysis of voting functions .......................... 55
- 4.5 Analysis of histogram size for Bayesian classifier .... 55
- 4.6 Comparison with other VPE methods ................. 57
- 4.7 Chapter summary ........................................ 59
4.1 Image dataset

In this chapter, firstly the image dataset and experiment methods are described in Section 4.1 and Section 4.2. Secondly, the selection of the algorithm parameters is discussed in Section 4.3. Thirdly, a survey of the proposed voting functions is provided in Section 4.4. Forthly, the configurations of the Bayesian classifier are analysed in Section 4.5. Finally, an evaluation of the proposed VPE method in comparison with state-of-the-art methods is presented in Section 4.6.

4.1 Image dataset

The experiments are conducted using the Pedestrian Lane Detection and Vanishing Point Estimation (PLVP) dataset [7]. This benchmark dataset originally contains 2000 images of unmarked pedestrian lanes that are taken under various imaging conditions: indoor, outdoor, sunny, cloudy. The images also include different lane types and lighting conditions (day, night, shadows). The ground-truth vanishing point for each image in the dataset has been annotated manually by volunteers to enable the quantitative measurement. The dataset is available to download at www.uow.edu.au/~phung/plvp_dataset.html. Sample images are provided in Figure 4.1.

In this research, we extend the PLVP dataset [7] with 2500 images of unmarked pedestrian lanes, taken under various imaging conditions. Some sample images of the dataset are provided in Figure 4.2. The ground-truth vanishing points of those images are also marked manually by volunteers, using the MATLAB tool provided with PLVP dataset.

For the experiments, the extended dataset is separated into three different sets of images: the training set, the validation set and the test set. The training set contains 3500 images and is dedicated for training the Bayesian classifier. The validation set including 500 images is used to evaluate the trained classifier as well as other parameters of the method. The test set has 500 images and is utilized in the comparison of the proposed VPE method with other methods.
4.2 Evaluation methods

The performance of the VPE methods on the dataset is evaluated based on three metrics. First, the estimation error of the method for an image is calculated as the Euclidean distance between the detected VP $p_d$ and the ground-truth VP $p_g$, normalized by the diagonal length $L$ of the image:

$$e_{vp} = \frac{\|p_d - p_g\|}{L}.$$  \hspace{1cm} (4.1)

The mean value $\bar{e}$ calculated for all tested images is used to compare the detection accuracy of VPE algorithms.

Second, the stability of a VPE algorithm is measured as the standard deviation $std$ of the estimation error, calculated for all the tested images. A VPE algorithm which provides a lower $std$ value is more stable to different road scenes.
4.2. Evaluation methods

Figure 4.2: Sample images from the extension of PLVP dataset with ground-truth VP annotated (red dot).
4.3 Analysis of voting algorithm parameters

Third, the processing speed of a VPE algorithm is measured as the average time $\bar{t}$ to process an image in the tested set. Because most of the applications employing VPE require real-time processing, a lower value of $\bar{t}$ is preferable.

To train the classifier and select the algorithm parameter, we use the error $\bar{e}$ as the main criterion. To evaluate VPE methods, a combination of three parameters $\bar{e}$, $std$ and $\bar{t}$ is used.

Consistent with the reviewed literature, all tested images are resized to $140 \times W$ pixels for landscape orientation images, or $H \times 140$ pixels for portrait orientation ones, where $H$ and $W$ are respectively the height and width of the resized images. The aspect ratio of the original images is unchanged. All experiments are conducted in MATLAB on a PC with a 3.4 GHz CPU and 8 GB of RAM.

4.3 Analysis of voting algorithm parameters

The proposed method utilizes a number of processing stages with a number of hyper-parameters. To reduced the complexity of the optimization progress, certain parameters are chosen empirically as follows:

- In Section 3.1.2, the edge threshold $\alpha$ is set as $\alpha = 1$.

- In Section 3.2.1, the threshold $\xi$ is set as $\xi = 5$.

- In Section 3.3.2, the threshold $\beta$ is set as $\beta = 1$.

Two key parameters need to be selected in the voting scheme: the search threshold $\tau$ in Section 3.2.1, and the angle threshold $\delta$ in Section 3.3.1. Initially, the size of the Gaussian filter $g$ in (3.2) is set as $g = 17$ based on the results in [7]. Lower values of $\tau$ and $\delta$ are preferred to reduce the search space and computation time. We compare the error performance of the VPE method with different values of $\tau$ and $\delta$ on 500 images of the validation set. The voting function $f_1(v, c)$ in (3.11) is used in this experiment.
4.3. Analysis of voting algorithm parameters

Table 4.1: VPE error for different values of $\tau$ and $\delta$ on the validation set.

<table>
<thead>
<tr>
<th>$\tau$</th>
<th>$\delta$</th>
<th>$\pi/180$ (1°)</th>
<th>$\pi/36$ (5°)</th>
<th>$\pi/18$ (10°)</th>
<th>$\pi/12$ (15°)</th>
<th>$\pi/9$ (20°)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.02</td>
<td>$\pi/180$ (1°)</td>
<td>0.1007</td>
<td>0.0942</td>
<td>0.0819</td>
<td>0.0825</td>
<td>0.0863</td>
</tr>
<tr>
<td>0.04</td>
<td>$\pi/36$ (5°)</td>
<td>0.0941</td>
<td>0.0754</td>
<td>0.0722</td>
<td>0.0732</td>
<td>0.0783</td>
</tr>
<tr>
<td>0.06</td>
<td>$\pi/18$ (10°)</td>
<td>0.0931</td>
<td>0.0715</td>
<td>0.0698</td>
<td>0.0701</td>
<td></td>
</tr>
<tr>
<td>0.08</td>
<td>$\pi/12$ (15°)</td>
<td>0.0923</td>
<td>0.0739</td>
<td>0.0712</td>
<td>0.0764</td>
<td></td>
</tr>
<tr>
<td>0.10</td>
<td>$\pi/9$ (20°)</td>
<td>0.0963</td>
<td>0.0795</td>
<td>0.0740</td>
<td>0.0763</td>
<td>0.0791</td>
</tr>
</tbody>
</table>

Table 4.2: Computation time (s) for different values of $\tau$ and $\delta$ on the validation set.

<table>
<thead>
<tr>
<th>$\tau$</th>
<th>$\delta$</th>
<th>$\pi/180$ (1°)</th>
<th>$\pi/36$ (5°)</th>
<th>$\pi/18$ (10°)</th>
<th>$\pi/12$ (15°)</th>
<th>$\pi/9$ (20°)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.02</td>
<td>$\pi/180$ (1°)</td>
<td>0.0416</td>
<td>0.0442</td>
<td>0.0475</td>
<td>0.0647</td>
<td>0.0858</td>
</tr>
<tr>
<td>0.04</td>
<td>$\pi/36$ (5°)</td>
<td>0.0501</td>
<td>0.0554</td>
<td>0.0606</td>
<td>0.0830</td>
<td>0.1027</td>
</tr>
<tr>
<td>0.06</td>
<td>$\pi/18$ (10°)</td>
<td>0.0791</td>
<td>0.0876</td>
<td>0.0935</td>
<td>0.1104</td>
<td>0.1359</td>
</tr>
<tr>
<td>0.08</td>
<td>$\pi/12$ (15°)</td>
<td>0.1293</td>
<td>0.1334</td>
<td>0.1422</td>
<td>0.1709</td>
<td>0.1975</td>
</tr>
<tr>
<td>0.10</td>
<td>$\pi/9$ (20°)</td>
<td>0.1611</td>
<td>0.1691</td>
<td>0.1775</td>
<td>0.2161</td>
<td>0.2286</td>
</tr>
</tbody>
</table>

Table 4.1 and 4.2 present the estimation error and computation time of the method on the training set for different values of $\tau$ and $\delta$. In Table 4.2, lower values $\tau < 0.06$ and $\delta < \pi/18$ give faster computation time, but these values lead to high VPE errors (> 0.07) as shown in Table 4.1. Larger values $\tau < 0.06$ and $\delta < \pi/18$ result in a higher VPE error with slower computation time.

The search threshold $\tau = 0.06$ and angle threshold $\delta = \pi/18$ are selected. We use these values to analyse different sizes of the Gaussian filter as shown in Table 4.3. The value $g = 13$ provides the lowest VPE error.

Table 4.3: VPE error for different values of $g$ on the validation set.

<table>
<thead>
<tr>
<th>Gaussian filter size $g$</th>
<th>7</th>
<th>11</th>
<th>13</th>
<th>17</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPE error</td>
<td>0.0865</td>
<td>0.0761</td>
<td>0.0691</td>
<td>0.0695</td>
<td>0.0801</td>
</tr>
</tbody>
</table>

From the results in this section, the parameters $g = 13$, $\tau = 0.06$, and $\delta = \pi/18$ are selected in the other experiments in this chapter.
4.4 Analysis of voting functions

In this section, we provide an analysis of different voting functions, which include different versions of the weight functions $w_1(\tilde{d})$ and $w_2(\tilde{\Delta})$ in (3.9). The analysis is based on the average error $\tilde{e}$ on 500 images of the validation set. The values of $\tau$, $\delta$, and $g$ are selected as mentioned above. The voting functions are either the combinations of $w_1(\tilde{d})$ and $w_2(\tilde{\Delta})$, or the voting functions proposed in the reviewed materials. The proposed voting function $f_1(v, c)$ in (3.11) and $f_2(v, c)$ in (3.12) are included in the analysis.

The results in Table 4.4 indicate the following:

- The proposed weight functions $w_1(\tilde{d}) = \exp(-\tilde{d})$ and $w_2(\tilde{\Delta}) = 1/(1 + \tilde{\Delta})$ give lower errors than the other weight functions.

- In each voting function groups, the functions $f$ that contain $w_1(\tilde{d}) = \exp(-\tilde{d})$ or $w_2(\tilde{\Delta}) = 1/(1 + \tilde{\Delta})$ tend to give lower errors than other combinations.

- The proposed voting function $f_1(v, c) = \frac{\exp(-\tilde{d})}{1+\tilde{\Delta}}$ performs better than the other voting functions proposed in state-of-the-art methods ([4], [12] and [21]). Also, $f_1(v, c)$ has the lowest average error in this analysis and second lowest std.

- The proposed voting function $f_2(v, c) = 1/(1 + \tilde{\Delta})$ is the best voting function in its category (in which the contribution of the distance $\tilde{d}$ is not considered).

To sum up, the results in this experiment support the selection of the voting functions in Section 3.3.1.

4.5 Analysis of histogram size for Bayesian classifier

In this chapter, we compare different histogram sizes for the Bayesian classifier described in (3.15). Note that a higher histogram size provides a finer pdf estimation, but requires more storage and training samples. In this experiment, 3500
4.5. Analysis of histogram size for Bayesian classifier

Table 4.4: Performance of different voting score functions on the validation set. The notable performances are given in bold.

<table>
<thead>
<tr>
<th>Group</th>
<th>Voting function</th>
<th>$\bar{e}$</th>
<th>$std$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f = w_1(\bar{d})$</td>
<td>$\frac{1}{1+d}$</td>
<td>0.0567</td>
<td>0.0766</td>
</tr>
<tr>
<td></td>
<td>$\frac{1}{1+d^2}$</td>
<td>0.0578</td>
<td>0.0767</td>
</tr>
<tr>
<td></td>
<td>$\exp(-d)$</td>
<td>0.0539</td>
<td>0.0708</td>
</tr>
<tr>
<td></td>
<td>$\exp(-d^2)$</td>
<td>0.0541</td>
<td>0.0714</td>
</tr>
<tr>
<td>$f = w_2(\bar{\Delta})$</td>
<td>$\frac{1}{1+\Delta}$</td>
<td>0.0658</td>
<td>0.0907</td>
</tr>
<tr>
<td></td>
<td>$\frac{1}{1+\Delta^2}$</td>
<td>0.0663</td>
<td>0.0917</td>
</tr>
<tr>
<td></td>
<td>$\exp(-\Delta)$</td>
<td>0.0664</td>
<td>0.0916</td>
</tr>
<tr>
<td></td>
<td>$\exp(-\Delta^2)$</td>
<td>0.0664</td>
<td>0.0916</td>
</tr>
<tr>
<td>$f = w_1(\bar{d}) w_2(\bar{\Delta})$</td>
<td>$\frac{\exp(-d)}{1+\Delta}$</td>
<td>0.0535</td>
<td>0.0709</td>
</tr>
<tr>
<td></td>
<td>$\frac{\exp(-d)}{1+\Delta^2}$</td>
<td>0.0538</td>
<td>0.0712</td>
</tr>
<tr>
<td></td>
<td>$\frac{\exp(-d^2)}{1+\Delta^2}$</td>
<td>0.0536</td>
<td>0.0706</td>
</tr>
<tr>
<td></td>
<td>$\frac{\exp(-d)\exp(-\Delta)}{(1+d)(1+\Delta)}$</td>
<td>0.0540</td>
<td>0.0712</td>
</tr>
<tr>
<td></td>
<td>$\frac{\exp(-d^2)\exp(-\Delta^2)}{1+\Delta}$</td>
<td>0.0538</td>
<td>0.0712</td>
</tr>
<tr>
<td>$f = w_1(\bar{d}) + w_2(\bar{\Delta})$</td>
<td>$\frac{1}{1+d} + \frac{1}{1+\Delta}$</td>
<td>0.0586</td>
<td>0.0815</td>
</tr>
<tr>
<td></td>
<td>$\frac{1}{1+d^2} + \frac{1}{1+\Delta^2}$</td>
<td>0.0585</td>
<td>0.0810</td>
</tr>
<tr>
<td></td>
<td>$\frac{1}{1+\Delta} + \frac{1}{1+\Delta^2}$</td>
<td>0.0580</td>
<td>0.0805</td>
</tr>
<tr>
<td></td>
<td>$\frac{1}{1+d} + \exp(-\Delta)$</td>
<td>0.0589</td>
<td>0.0815</td>
</tr>
<tr>
<td></td>
<td>$\frac{1}{1+d} + \frac{1}{1+\Delta}$</td>
<td>0.0569</td>
<td>0.0783</td>
</tr>
<tr>
<td></td>
<td>$\frac{1}{1+d^2} + \exp(-\Delta^2)$</td>
<td>0.0580</td>
<td>0.0805</td>
</tr>
<tr>
<td></td>
<td>$\exp(-d) + \frac{1}{1+d}$</td>
<td>0.0579</td>
<td>0.0805</td>
</tr>
<tr>
<td></td>
<td>$\exp(-d) + \exp(-\Delta)$</td>
<td>0.0569</td>
<td>0.0783</td>
</tr>
<tr>
<td></td>
<td>$\exp(-d^2) + \exp(-\Delta^2)$</td>
<td>0.0573</td>
<td>0.0787</td>
</tr>
<tr>
<td></td>
<td>$\exp(-d^2) + \exp(-\Delta)$</td>
<td>0.0577</td>
<td>0.0803</td>
</tr>
<tr>
<td></td>
<td>$\exp(-d) + \exp(-\Delta^2)$</td>
<td>0.0566</td>
<td>0.0778</td>
</tr>
<tr>
<td>$f(\bar{d} \bar{\Delta})$</td>
<td>$\frac{1}{1+d\bar{\Delta}}$ [12]</td>
<td>0.0591</td>
<td>0.0820</td>
</tr>
<tr>
<td></td>
<td>$\frac{1}{1+d\bar{\Delta}^2}$</td>
<td>0.0592</td>
<td>0.0815</td>
</tr>
<tr>
<td></td>
<td>$\exp(-d \bar{\Delta})$ [21]</td>
<td>0.0578</td>
<td>0.0806</td>
</tr>
<tr>
<td></td>
<td>$\exp(-d \bar{\Delta}^2)$</td>
<td>0.0582</td>
<td>0.0809</td>
</tr>
</tbody>
</table>
images of the training set are used to construct four different Bayesian classifiers with $32^2$, $64^2$, $128^2$, and $256^2$ bins. Table 4.5 shows the VPE errors on the validation set. For comparison purpose, the VPE error obtained when using only VP voting function $f_1(v, c)$ (i.e. no Bayesian classifier) is also shown.

The results in Table 4.5 indicate that the Bayesian classifier needs at least $64^2$ bins to improve VPE accuracy compared with using the voting function $f_1(v, c)$ alone. In addition, the best result is achieved with the $256^2$ histogram bins. Therefore, in the subsequent experiments, a histogram size of $256^2$ bins was used.

To sum up, the results in Table 4.4 and Table 4.5 indicate that the average error of the method is greatly improved with the implementation of the voting function, reducing from $\bar{e} = 0.0641$ ($f = 1$) to $\bar{e} = 0.0501$ (voting function using Bayesian classifier with $256^2$ bin histogram).

Table 4.5: VPE error performance on the validation set for different histogram sizes of the Bayesian classifier.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>$\bar{e}$</th>
<th>std</th>
</tr>
</thead>
<tbody>
<tr>
<td>$32^2$ bins classifier</td>
<td>0.0534</td>
<td>0.0708</td>
</tr>
<tr>
<td>$64^2$ bins classifier</td>
<td>0.0527</td>
<td>0.0705</td>
</tr>
<tr>
<td>$128^2$ bins classifier</td>
<td>0.0515</td>
<td>0.0651</td>
</tr>
<tr>
<td>$256^2$ bins classifier</td>
<td>0.0501</td>
<td>0.0636</td>
</tr>
<tr>
<td>No classifier, $f_1(v, c)$ only</td>
<td>0.0535</td>
<td>0.0709</td>
</tr>
<tr>
<td>No classifier, $f_2(v, c)$ only</td>
<td>0.0637</td>
<td>0.0712</td>
</tr>
</tbody>
</table>

### 4.6 Comparison with other VPE methods

The proposed VPE method was compared with the edge-based method (Wang et al. [3]) and three texture-based methods (by Kong et. al. [5], Moghadam et. al. [16], and Phung et. al. [7]). The methods of Wang et al., Kong et. al, and Moghadam et. al. are described in Section 2.1.1, Section 2.2.2 and Section 2.2.3 respectively. The method of Phung et. al is the enhanced version of the method discussed in Section 2.3.3. In this section, we utilize the tested values for the
4.6. Comparison with other VPE methods

parameters $\tau$ and $\delta$, as well as the $256^2$ histogram trained in previous experiment. The evaluation is performed on 500 images of the test set. The MATLAB code for the methods of Kong et. al and Phung et. al are provided by the corresponding authors, while the methods of Wang et al. and Moghadam et al. are implemented based on the related literature.

Table 4.6: Performance of VPE methods on the test set.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\bar{e}$</th>
<th>$std$</th>
<th>$t$ (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge-based [3]</td>
<td>0.1211</td>
<td>0.1771</td>
<td>0.0249</td>
</tr>
<tr>
<td>Kong et al. [5]</td>
<td>0.0811</td>
<td>0.1039</td>
<td>2.9982</td>
</tr>
<tr>
<td>Moghadam et al. [16]</td>
<td>0.2009</td>
<td>0.1032</td>
<td>0.5942</td>
</tr>
<tr>
<td>Phung et al. [7]</td>
<td>0.0710</td>
<td>0.0949</td>
<td>0.5739</td>
</tr>
<tr>
<td>Proposed method</td>
<td>0.0503</td>
<td>0.0638</td>
<td>0.1315</td>
</tr>
</tbody>
</table>

Table 4.6 presents the performance of the above VPE methods on the test set. Firstly, in term of detection accuracy, the average error $\bar{e} = 0.0503$ of the proposed method is significantly lower than that of the state-of-the-art methods, of which the lowest error is $\bar{e} = 0.0503$ by the method of Phung et al. [7]. The improved accuracy of the proposed method can be attributed to the optimization of the VP candidate and voter search spaces, as well as the voting functions. The results also show that the texture-based methods in [5] ($\bar{e} = 0.0811$) and [7] ($\bar{e} = 0.0710$) are more accurate than the edge-based method in [3] ($\bar{e} = 0.1211$), with the exception of the OLDOM method in [16] ($\bar{e} = 0.2009$). The resulted errors of the OLDOM and edge-based methods are very high (>0.1) on the test set.

Secondly, in term of detection stability, the standard deviation of the proposed method ($std = 0.0638$) is about 67% times smaller than that of the method in [7] ($std = 0.0949$), which is the best result among the state-of-the-art methods. This comparison shows that the proposed method is more stable than the current VPE solutions. The results also confirm that the texture-based methods (best $std = 0.0949$) offer better stability than the edge-based method ($std = 0.1771$).

Thirdly, in term of processing speed, the proposed method requires shorter time ($t = 0.1315$ s) than other texture-based methods, of which the fastest is
\( \bar{t} = 0.5739 \text{ s} \) achieved by \([7]\). The difference is about 4.4 times, which is significant with the real-time applications. This improvement in the computation efficiency can be attributed to the selection of a smaller number of VP candidates and voters compared to the other texture-based methods. The edge-based CHEVP method has the shortest computation time (0.0249 s), however, as mentioned above this method does not have high accuracy and stability on the tested images.

Figure 4.3 provides some visual results of the VPE methods on the test images. In most cases, compared with the other methods, the vanishing points detected with the proposed method (white squares) are closer to the ground-truth vanishing points (red asterisks).

### 4.7 Chapter summary

In this chapter, we present the experimental results and analysis of our proposed VPE method. The database used in the experiment is described, and the parameters for the VPE method are selected using a validation set. Different configurations of the voting function and Bayesian classifier are evaluated to determine the optimal configuration for the proposed method. The state-of-the-art VPE methods are compared with the proposed method on the test set of PLVP dataset. The results show that the proposed VPE method outperform the other solutions in term of accuracy, robustness and processing time.
Chapter 5

Conclusion

Chapter contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Thesis summary</td>
<td>62</td>
</tr>
<tr>
<td>5.2 Future works</td>
<td>63</td>
</tr>
<tr>
<td>5.3 Conclusion</td>
<td>64</td>
</tr>
</tbody>
</table>
A vanishing point is considered as an essential image feature in many applications, such as road detection, camera calibration, and 3D reconstruction. In order to fast and accurately detect the VP, we propose a system that includes different image processing techniques as well as a Bayesian classifier to adaptively adjust the parameters of the algorithm. In this thesis, we investigate color tensors features to extract the texture orientation and color edge maps. We propose new enhancements to the VP voting process, including the optimized VP search space and the adaptive voting function based on Bayes’ theorem. These enhancements address the drawbacks in computation time and robustness of state-of-art VPE methods.

In this chapter the research activities presented in the thesis are summarized. The chapter is organized as follows: Section 5.1 outlines the research contributions of the thesis; Section 5.2 provides the future research directions; Section 5.3 draws the conclusion for the thesis.

### 5.1 Thesis summary

The research activities described in the thesis are summarized as followed:

- We provide the literature review of existing methods for VPE, as well as image processing techniques to extract texture orientations and edge features. State-of-the-art methods concerning both intensity images as well as color images are reviewed. The color tensor features are also investigated.

- We propose a VPE method including three stages: extracting local texture orientation and edge maps, identifying the VP search space, and calculating VP scores with a voting scheme. Color tensor features are used to extract texture orientation and edge maps. Based on these extracted maps, the search spaces for VP candidates and voters are optimized using Hough transform, straight line and intersection detection and neighborhood pixel
technique. A novel weighting method was proposed to define the voting score function, and a simple Bayesian classifier is employed to adaptively select the voting function.

- We analyse the parameters utilized in the proposed algorithm and evaluate the proposed VPE method along with several existing methods. The PLVP dataset of pedestrian lane images is extended to support the training process of the Bayesian classifier.

5.2 Future works

Although the experimental results validate many advantages of the proposed VPE method, there are a few aspects in the method that can be improved as follows:

- There are a number of hyper-parameters of the proposed method are chosen empirically, which might lead to inconsistency when the method is implemented on different datasets.

- Because the VP voter search space in Section 3.2.2 locates below its respective VP candidate, the method might fail to correctly detect the best VP in extreme scenes, such as uphill roads or U-shape roads.

- The approach of adaptively choosing the voting function improves the accuracy and robustness of the proposed method. However, the results could be better to match the additional computation cost of the voting function and the Bayesian classifier.

Possible research directions from this thesis can be summarized as follows:

- Improve the texture orientation extraction and edge detection using optimized color tensor analysis or different image processing techniques.

- Evaluate the proposed line detection technique using Hough transform along with other solutions.
5.3. Conclusion

- Study optimization technique to determine the algorithm parameters, including the implementation of neural networks with data-driven approach.
- Utilize the proposed VPE method in a lane segmentation system to evaluate the impact in practical applications.

5.3 Conclusion

An effective 3-stage system is proposed for VPE in road scenes. Stage 1 employs color tensors to calculate the texture orientations and the edge features of the input image. Stage 2 proposes new strategies to identify an optimized search space of VP candidates and voters. Stage 3 provides a robust approach to calculate the VP scores, which includes a simple Bayesian classifier. The proposed method is optimized using the training images from the extended PLVP dataset. Different VPE methods are evaluated on the test set selected from the dataset. Overall, the proposed VPE method yields significant improvements in computation time and accuracy over several state-of-the-art approaches.
References


References


References


