University of Nevada, Reno

Horizon Line Detection: Edge-less and Edge-based Methods

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering.

by

Touqeer Ahmad

Dr. George Bebis/Thesis Advisor

December, 2014

THE GRADUATE SCHOOL

We recommend that the thesis prepared
under our supervision by

TOUQEER AHMAD

Entitled

Horizon Line Detection: Edge-less and Edge-based Methods

be accepted in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

George Bebis, Ph.D. , Advisor

Mircea Nicolescu, Ph.D. , Committee Member

Yantao Shen, Ph.D. , Graduate School Representative

David W. Zeh, Ph.D., Dean, Graduate School

December, 2014

Abstract

Planetary rover localization is a challenging problem due to unavailability of conventional localization cues e.g. GPS, architectural landmarks etc. Horizon line (boundary segmenting sky and non-sky regions) finds its applications for smooth navigation of UAVs/MAVs, visual geo-localization of mountainous images, port security and ship detection and has proven to be a promising visual cue for outdoor robot/vehicle localization.

Prominent methods for horizon line detection are based on faulty assumptions and rely on mere edge detection which is inherently a non-stable approach due to parameter choices. We investigate the use of supervised machine learning for horizon line detection. Specifically we propose two different machine learning based methods; one relying on edge detection and classification while other solely based on classification. Given a query image; an edge or classification map is first built and converted into a multi-stage graph problem. Dynamic programming is then used to find a shortest path which conforms to the detected horizon line in the given image. For the first method we provide a detailed quantitative analysis for various texture features (SIFT, LBP, HOG and their combinations) used to train an Support Vector Machine (SVM) classifier and different choices (binary edges, classified edge score, gradient score and their combinations) for the nodal costs for Dynamic Programming. For the second method we investigate the use of dense classification maps for horizon line detection. We use Support Vector Machines (SVMs) and Convolutional Neural Networks (CNNs) as our classifier choices and use raw intensity patches as features. Dynamic Programming is then applied on the resultant dense classifier score image to find the horizon line. Both proposed formulations are compared with a prominent edge based method on three different data sets: City (Reno) Skyline, Basalt Hills and Web data sets and outperform the previous method by a high margin.

Dedication

To Saba and Romaisa.

Acknowledgements

First of all I would like to thank my advisor Professor George Bebis for his kind guidance throughout the course of my MS, helping me understand research material and proof reading my documents; presentations, reports, papers and thesis and providing his valuable feedback and remarks. I am grateful to Dr. Mircea Nicolescu and Dr. Yantao Shen for being members of my thesis committee.

I would like to say thanks to my parents especially my dad Professor (retd.) Muhammad Ikram for always supporting me to pursue my Ph.D. and helping me stay focused on my studies by all means possible. I am thankful to my lovely wife Saba and my beautiful daughter Romaisa for always being on my side throughout this journey and being the source of my smiles and happiness.

Declaration

I declare that all the work in this thesis is solely my own, except where attributed and referred to other authors. Some material from this thesis has been previously published and others are under review: please refer to Appendix for details.

Contents

1 Introduction 1

2 Previous Work 4

2.1 Literature Review 4

2.2 Lie et al. [3] 6

3 Method 1: Edge-based Horizon Line Detection 9

3.1 Learning the Horizon Line 9

3.1.1 Maximally Stable Extremal Edges (MSEEs) 10

3.1.2 Ground Truth Labeling and Key Points Selection . . . 11

3.1.3 Texture Features and Classifier Training 11

3.2 Horizon Line Detection 12

3.2.1 Filtering MSEE Pixels 13

3.2.2 Graph Formulation for Dynamic Programming 14

3.3 Proposed Nodal Costs 15

3.3.1 Gradient Information 15

3.3.2 Classified Binary Edges 16

3.3.3 Classified Edge Score 17

3.3.4 Classified Edge Score + Gradient Information 17

4 Method 2: Edge-less Horizon Line Detection 18

4.1 Pixel Classification 19

4.2 Horizon Line Detection 20

4.2.1 Dense Classifier Score Image (DCSI) 20

4.2.2 Reduced Dense Classifier Score Image (mDCSI) 20

4.2.3 Horizon Line Detection as a Shortest Path Problem. . 21

5 Experimental Details 23

5.1 Data Sets and Quantitative Evaluation 23

5.2 Results for Method1 24

5.2.1 Effect of MSEE 24

5.2.2 Reduction in Horizon Candidate Edges due to Different Features 24

5.2.3 Best Nodal Cost. 25

5.3 Results for Method2 26

5.3.1 Quantitative Evaluation 26

5.3.2 Comparing Classifiers 27

5.4 Discussion and Some Visual Results 28

5.4.1 Failure of Lie et al. [3] 28

5.4.2 Failure of Proposed Approaches 30

6 Conclusions 33

List of Figures

2.1 Steps of Horizon Detection by Lie et al. [3] 8

3.1 Effect of MSEE: Input Image (row1), Output of the Canny (row2), Discarded Edges by MSEE (row3) and survived edges (row4) i.e. MSEE Image. 10

3.2 Ground Truth Horizon Lines; highlighted in red. 11

3.3 Positive (red) and negative (blue) key point locations for one of the training image. 11

3.4 Flow diagram for the training pahse of the first proposed approach. 12

3.5 Flow diagram for the testing pahse of the first proposed approach. 13

3.6 A sample novel image (row1), corresponding MSEE (row2) and MSEE₊ (row3) Images. Note the reduction in number of edges due to the classifier. 14

4.1 Main steps of the training/testing phases of second proposed horizon line detection approach. 19

4.2 Sample test images(row1), respective DCSIs (row2), mDCSIs (row3), shortest path solutions (row4, highlighted in red), and detected horizon lines (row5, highlighted in red). 21

5.1 Mean of % False Positive and False Negative Errors when various Features and their combinations used for training the SVM Classifier 26

5.2 Test images (column1), corresponding SVM-DCSIs (column2) and corresponding CNN-DCSIs (column3). 28

5.3 Examples illustrating: [row 1] missing the horizon line or parts of it due to edge gaps (Lie et al.), and [row 2] detecting the true horizon line using our approach 2 (SVM). 29

5.4 Zoomed sub-images of the left column images of Figure 5.3 . . 29

5.5 Examples illustrating: [row 1] missing parts of the horizon line due to the assumption that the horizon line is close to the top of the image (Lie et al.), and [row 2] detecting the true horizon line using the proposed approach 2 (SVM). 30

5.6 [row 1] effect of not allowing node connections within the same stage; [row 2] solution obtained by allowing node connections within the same stage. 31

5.7 Examples illustrating the inability of the proposed method to find a good solution due to the lack of sufficient training data. 32

5.8 Sample results illustrating our horizon line detection approach: City data set(row1), Basalt Hills data set (row2) and Web data set (row 3 through 5). Detected horizon lines are highlighted in red/green. 32

List of Tables

5.1 Importance of MSEE 24
5.2 % FP and %FN Errors due to Various Features 25
5.3 Approach1: Average Absolute Errors 27
5.4 Approach2: Average Absolute Errors 27

Chapter 1

Introduction

In computer vision community, problem of segmenting an image into sky and non-sky regions is termed as horizon line detection or skyline extraction problem. Earlier methods for horizon line detection either rely on edge detection as preprocessing step [3, 21, 28, 29] or address the problem using supervised (classification) [5, 6, 7, 9, 19] or unsupervised (clustering) [4] machine learning techniques. Edge based methods suffer due to edge ambiguities and result into gaps e.g. due to clouds parts of horizon might be occluded and hence missed by the edge detector or otherwise non-horizon edges might be included into solution horizon due to bias induced towards specific solutions [3]. Machine learning based attempts mostly model sky and non-sky regions and are based on the faulty assumption that horizon is a linear boundary [6, 19]. Moreover the non-sky regions can vary a lot and it is very hard for machine learning algorithms to generalize well across all these variations in the test sets [4, 8]. For example a non-sky region could be comprised of water, mountains, plains or mixture of all i.e. significant color and texture changes.

In this work, we have proposed two different machine learning based horizon line detection approaches; where instead of modeling sky and non-sky regions we model horizon and non-horizon regions. It is important to note that horizon boundary is more consistent as compared to non-sky regions, which is why methods trained on horizon generalize much better as compared to others trained on sky, non-sky regions. Moreover earlier methods are based on faulty assumptions e.g. horizon being a linear boundary [6, 19] or horizon being present in the upper half of the image [3] which is generally not the case. Our first proposed method relies on edge detection and classification whereas the second addresses the problem in a pure classification framework. Both approaches formulate the resultant edge/classification map as a multistage graph problem and find a shortest path using Dynamic Programming (DP) which conforms to the detected horizon in the given query image.

For the first approach, given a query image; Canny edge detector is applied first with a range of thresholds to keep only those edges which survive over a wide range. The surviving edges are termed as Maximally Stable Extremal Edges (MSEEs). The number of edges is further reduced significantly by classifying the MSEEs into horizon and non-horizon edges using a trained classifier. Dynamic programming is then applied on horizon classified edges using any of the available nodal costs. We investigate the suitability of various local texture features and their combinations as feature choices for the trained horizon classifier. Specifically, we explore SIFT [14], LBP[13], HOG[15] and their combinations SIFT-LBP, SIFT-HOG, LBP-HOG and SIFT-LBPHOG as features to train an SVM [26] classifier. We evaluate various nodal costs for dynamic programming e.g. binary edge scores, normalized classification scores for horizon classified edges, gradient information and their combination. The proposed formulation is compared with an earlier prominent approach which relies only on edge information and is based on faulty assumption of horizon being present in the upper half of the image [3]. The results are reported for two challenging data sets (125 images in total) i.e. Basalt Hills data set (45 images) and Web data set (80 images) comprised of mountainous images captured during an outdoor robot exploration and randomly chosen from the web with considerable viewpoint, scenic and weather changes.

In the second proposed approach we use machine learning and Dynamic Programming to extract the horizon line from a classification map instead of an edge map. The key idea is assigning a classification score to each pixel, which can be interpreted as the likelihood of the pixel belonging to the horizon line, and representing the classification map as a multi-stage graph. Using DP, the horizon line can be extracted by finding the path that maximizes the sum of classification scores. In contrast to conventionally used edge maps which are typically binary (edge vs no-edge) and contain gaps, classification maps are continuous and contain no gaps, yielding significantly better solutions. We use normalized intensity patches as feature choice for this method and train SVM [26] and CNN [27] classifiers.

Both of our formulations allow us to remove certain assumptions which are common to earlier methods such as the horizon is close to the top of the image or that the horizon forms a straight line. The purpose of these assumptions is to either bias the DP solution or simplify the underlying segmentation problem but they fail to produce good results when not valid. We demonstrate our second approach on three different data sets (138 images in total) i.e. City Skyline data set (13 images), Basalt Hills data set (45 images) and Web data set (80 images) and again compare it with a prominent traditional approach based on edge maps [3]. For both approaches although our training set is comprised of a very small number of images from the same location, our results illustrate that our methods generalize well to images acquired under different conditions and geographical locations.

Following this chapter we provide a brief literature review of earlier horizon detection methods and provide details about a prominent horizon line detection method by Lie et al.[3] relying solely on edge detection. In chapter 3 we present our first proposed approach which is based on edge detection and edge based classification. Chapter 4 gives the details about the second proposed approach using classification maps instead of edge maps. Chapter 5 provides the experimental details and results for different data sets for both proposed methods and their comparisons against Lie et al. [3] formulation. Thesis is concluded in chapter 6 with concluding remarks about the presented methods and listing of the future work.

Chapter 2

Previous Work

2.1 Literature Review

Horizon/Sky line detection or sky segmentation is the problem of finding a boundary between sky and non-sky regions (ground, water or mountains) given a gray scale or color image. Previous attempts to horizon line detection can be categorized into two major groups; methods modeling sky and non-sky regions either by some machine learning algorithms both supervised and unsupervised approaches [4, 5, 6, 7, 9, 19] and methods relying on edge detection as the essential pre-processing step [3, 21, 28, 29]. Recently some attempts [1, 2] have been made to combine these two ideas to refine the edges by training classifiers but broadly these attempts also fall under the second category as they are not possible without edges being detected. Most of the earlier methods to horizon detection suffer from the assumption of horizon boundary being linear and hence are limited. Horizon line or sky segmentation has many applications e.g. smooth navigation of small unmanned aerial vehicles (UAVs) [4, 7, 8] and micro air vehicles (MAVs) [5, 6, 9], visual geo-localization of mountain images [17, 18, 28, 29], port security and ship detection [19, 20] and outdoor robot/vehicle localization [10, 11, 12] to name a few.

McGee et al. [7] proposed a sky segmentation approach to find a linear boundary between sky and non-sky regions based on an SVM classifier trained only on the color information. They use sky segmentation as an obstacle detection tool for small scale UAVs. Their underlying assumption of horizon line being linear violates very often and probably is acceptable only for UAVs navigation. Ettinger et al. [6] proposed a flight stability and control system for micro air vehicles (MAVs) which relies on their horizon detection method. They model the sky and non-sky regions by Gaussian distributions and try to find an optimized boundary segmenting them. Their proposed model is based on two assumptions; horizon line is linear and horizon line separates the image into two regions of significantly different appearance i.e. sky and non-sky. So, their approach is also limited by the assumption of horizon being a linear boundary. Fefilatyev et al. [19] is also based on horizon being linear and uses color and texture features such as mean intensity, entropy, smoothness, uniformity etc. to train various classifiers. Croon et al. [5] extend the features used by [19] by including corner-ness, grayness and Fisher discriminant features to train shallow decision trees. They train decision trees to make the sky segmentation fast for MAVs obstacle avoidance. This approach is able to detect non-linear horizon lines.

Todorovic et al. [9] circumvent their assumption of horizon being linear in [6] by building priors for sky and non-sky regions based on color and texture features. Unlike their earlier work, they focused on both color (Hue, Intensity) and texture (Complex Wavelet Transform) features to model the priors due to great appearance variations in sky and non-sky regions. They trained a Hidden Markov Tree model based on these color and texture features which resulted in a robust horizon detection approach capable of detecting non-linear horizon boundaries. [22] have proposed a fusion based approach where they combine the outputs of NN classifier with k-means clustering. They use mean intensity and texture features similar to [5, 19] to train their classifier. Although their approach demonstrates reasonable results but their system is based on various heuristics and parameters which would not generalize very well to other data sets under different conditions.In [4], Boroujeni et al. presented a clustering based horizon detection approach for robust UAV navigation. They assume the presence of a dominant light field between sky and ground region which they try to locate using intensity and k-means clustering. In general their assumption about the light field does not hold. They themselves identified cases where their proposed method needed modifications for cluster detection process. Thurrowgood et al. [8] used their horizon detection approach for attitude estimation of UAVs. Using training data they learn transformation from RGB space to a single dimension where an optimum threshold is searched to segment sky/non-sky regions based on histograms and priors about sky and non-sky region distributions. The proposed approach is limited only to UAV navigation due to their assumption of sky and ground pixels being equi-probable.

Most prominent method belonging to second category is that of Lie et al. [3] where they address horizon detection as a graph problem. This method relies on edge detection and assumes an almost consistent edge boundary exists between sky and non-sky regions. The detected edge map is formulated as a multi-stage graph problem where each column of the image becomes a stage of the graph and each edge pixel becomes a node. A shortest path is then found extending from left most column to right most column by using Dynamic Programming (DP). The assumption of horizon being consistent edge boundary is seldom true due to environmental conditions e.g. clouds and results into edge gaps. To address this issue [3] use the δ and tolerance-of-gap(tog) as parameters to fill up the gaps which arise from edge detection to find a consistent horizon line. Also they assume horizon being present in the upper half of the image and induce bias to find a shortest path in this region.

2.2 Lie et al. [3]

This section briefly details the approach by [3]. Since, we also formulate our approaches as a multi-stage graph problem it seems necessary to provide the details of original approach by [3] so that we can point out the problems with underlying assumptions by [3] and identify various similarities and differences of our proposed approaches.

Lie et al. [3] formulate the horizon line detection problem as a multi-stage graph problem and then solve it by DP to find the shortest path extending from left to right. Their approach is based on the assumption that a full horizon line exists in the given image from left to right and lies in the upper half of the image. For a given query image of size $M \times N$, edge detection is applied first to get a binary edge map I containing 1 as edges and 0 as non-edges. This edge map is used to formulate an $M \times N$ multi-stage graph $G(V,E,\Psi,\Phi)$ where each edge pixel from the edge map becomes a node with a constant lower cost l associated with it and each non-edge pixel has a higher cost typically ∞.

$$\Psi(i,j) = \begin{cases} l, & \text{if } I(x,y) = 1 \\ \infty, & \text{if } I(x,y) = 0. \end{cases} \tag{2.1}$$

where, $\Psi(i,j)$ is the node cost associated with vertex i in stage j i.e. v_{ij}. The graph can be visualized as an N stage graph; where there are N stages (columns) and each stage has M nodes (rows) in it. As edge detection may cause gaps due to presence of clouds in the image or edges being ignored due to thresholding; they propose a gap filling approach to deal with these gaps.

For a given node i in stage j its neighborhood in next stage $j + 1$ is defined by δ parameter i.e. number of nodes to which i could be connected in stage $j + 1$. The edges from i to its neighbors are associated with costs equal to the vertical absolute distances from it as shown in the equation below.

$$\Phi(i,k,j) = \begin{cases} |i-k|, & \text{if } I(i,j) = I(k,j+1) = 1 \\ & \text{and } |i-k| \leq \delta \\ \infty, & \text{otherwise.} \end{cases} \tag{2.2}$$

If a node i in stage j is not able to be connected to any node in stage $j + 1$ with in δ neighborhood; a search window is defined by parameters δ and tolerance-of-gap (tog). Specifically an edge node is searched in this search window and once such an edge node is found the gap filling is performed by introducing dummy nodes in between node i in stage j and node k found within the search window j+tog. They associate a high cost with these dummy nodes introduced by gap filling step.

Once the gaps are filled with high cost dummy nodes; the nodes in stage 1 and N are assigned an increasing cost associated with the vertical positions of the nodes according to the equation below.

$$\Psi(i,j) = \begin{cases} (i+1)^2, & \text{if } j = 1 \text{ or } j = N \\ \Psi(i,j), & \text{otherwise.} \end{cases} \qquad (2.3)$$

This enforces the assumption of horizon line being present in the upper half of the image and hence biasing towards a shortest path present in the upper half. Next two nodes; a source s and a sink t are added to the left of the left most stage (i.e. stage 1) and to the right of the right most stage (i.e. stage N) respectively. A zero cost is associated with these two nodes. The s node is connected with all the nodes in stage 1 and all the nodes in stage N are connected to node t. The edges for these links are associated with zero costs. A shortest path is then found extending from node s to t by Dynamic Programming which conforms to the detected horizon boundary.

Figure 2.1 shows a drawing depicting the steps of Lie et al. [3] for a sample small size image. An edge map is shown in 2.1-(a) where black and white rectangles represent edge and non-edge pixels. A search window is shown for the edge node in stage j = 5 for parameter choices δ = 1 and tog = 4 in figure 2.1-(b). With in the search window j+tog two edge nodes are discovered which are then connected to node j by introducing the dummy nodes highlighted in blue in figure 2.1-(c,d). The nodes in stage 1 and N are set to an increasing cost associated with their vertical position reflected by increasing intensity in figure 2.1-(e). Two nodes s and t would now be introduced as described above and DP would be applied on this graph. As clear from figure 2.1-(e) there exists two equal paths in the above sample graph (ignoring source (s) and sink (t) nodes); but DP formulation would find the upper path due to the assumption of horizon being present in the upper half and stages 1 and N being initialized with bias to this assumption. It is probable that the true horizon line was actually the lower one and the upper edge segment on the right was only due to some clouds.

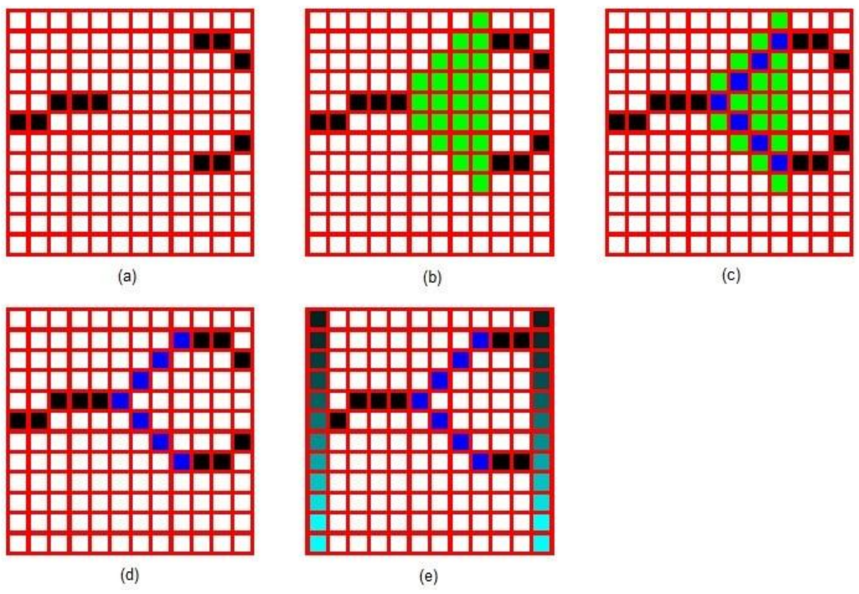

Figure 2.1: Steps of Horizon Detection by Lie et al. [3]

We would show various specific examples in the experiment section where this method failed to detect a portion of horizon due to this bias associated with the location of horizon.

Chapter 3

Method 1: Edge-based Horizon Line Detection

3.1 Learning the Horizon Line

In the first approach we propose a machine learning based framework where number of edges are reduced considerably first by MSEE image and then by using a trained SVM classifier. Different components of our framework are presented next.

3.1.1 Maximally Stable Extremal Edges (MSEEs)

The idea of extracting MSEEs is inspired from extracting Maximally Stable Extremal Regions (MSER) [25]. Given a gray scale image, we compute the edge image using the Canny edge detector [30] with sigma (σ) parameter being fixed to a chosen value while varying the low and high thresholds. This results in the generation of N binary images assuming N combinations of parameter values, call them I_1 to I_N. An edge at pixel location (x,y) is considered stable if it is detected as an edge pixel for k consecutive threshold values. The image comprised of these stable edges is referred to as Maximally Stable Extremal Edge Image and denoted as E. Mathematically,

$$E(x,y) = \begin{cases} 1, & \text{if } \sum_{i=1}^{N} I(x,y)_i > k. \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

In our experiments, we varied the high threshold of the Canny edge detector, Th, between 0.05 and 0.95 with a step of 0.05; the lower threshold Tl was set 0.4 × Th. It was found through experimentation that σ = 2 and k = 3 were optimal choices. The computation of MSEE Image reduces the number of edges considerably while not damaging important edges (i.e., horizon edges). Figure 3.1 shows a sample image, the output of the Canny edge detector and the MSEEs. As it can be observed, the number of edges has remarkably been reduced in MSEE while maintaining the edges belonging to the horizon line.

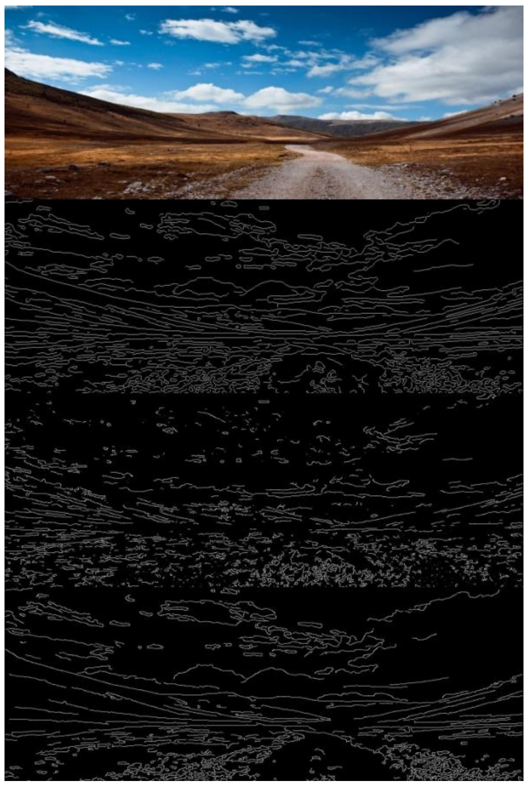

Figure 3.1: Effect of MSEE: Input Image (row1), Output of the Canny(row2), Discarded Edges by MSEE (row3) and survived edges (row4) i.e. MSEE Image.

3.1.2 Ground Truth Labeling and Key Points Selection

To train the SVM classifier, we manually label the horizon line pixels in the training images using the MSEEs. Since some portion of the true horizon line might not be detected as edges or the edge detector's output might not match the true horizon line perfectly and there may be edges which are not horizon pixels but strong enough to survive different parameter choices; a careful manual labeling of horizon pixels is required. We use these manually labeled horizon pixel indices for comparing the detected horizons in experiment section. Figure 3.2 shows few images from Web data set with ground horizons marked as red.

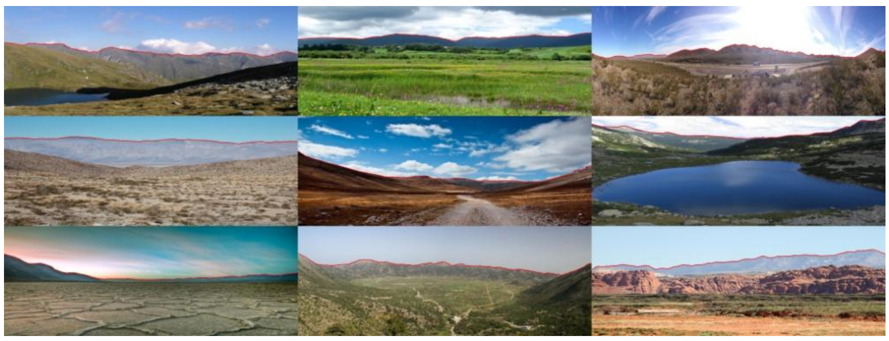

Figure 3.2: Ground Truth Horizon Lines; highlighted in red.

The positive and negative key points to train the classifier are chosen uniformly and randomly from the ground truth horizon location and MSEE non-horizon edge locations respectively for all the images in the training set. Figure 3.3 shows the locations of positive and negative key points for one of the training images from Basalt Hills data set.

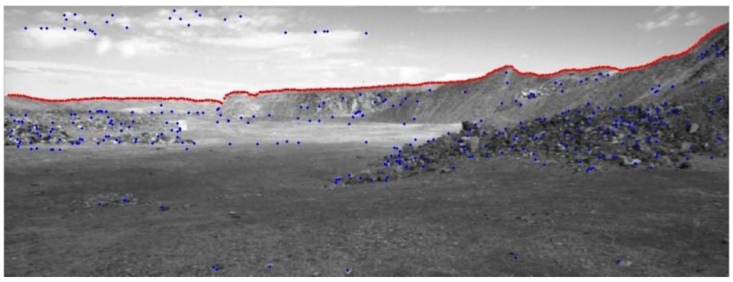

Figure 3.3: Positive (red) and negative (blue) key point locations for one of the training image.

3.1.3 Texture Features and Classifier Training

To train an SVM classifier we take a 16×16 window around each +ive/-ive key point and compute the chosen descriptor. We have investigated three texture descriptors as feature choices to train the SVM classifier namely Scale Invariant Feature Transform (SIFT) [14], Local Binary Patterns(LBP)[13] and Histogram of Oriented Gradients(HOG)[15]. We also explore their combinations with each other and then all of them combined as feature choices for our classifier. We use the implementation of these descriptor available from vlfeat[16] which provides 128, 58 and 31 dimensional vectors for SIFT, LBP and HOG respectively. We investigate individual descriptors as well as their combinations as the feature choices to train the SVM classifier. The combinations are formed by mere concatenation of the descriptor vectors for each training and testing instance. The feature sizes for each combination: SIFT-LBP, SIFT-HOG, LBP-HOG and SIFT-LBP-HOG are 186, 159, 89 and 217 respectively. We have found SIFT-HOG combination as the best choice when compared with individual descriptors and other combinations as described in the results section.

Figure 3.4 shows a flow diagram for the training phase of our first proposed approach.

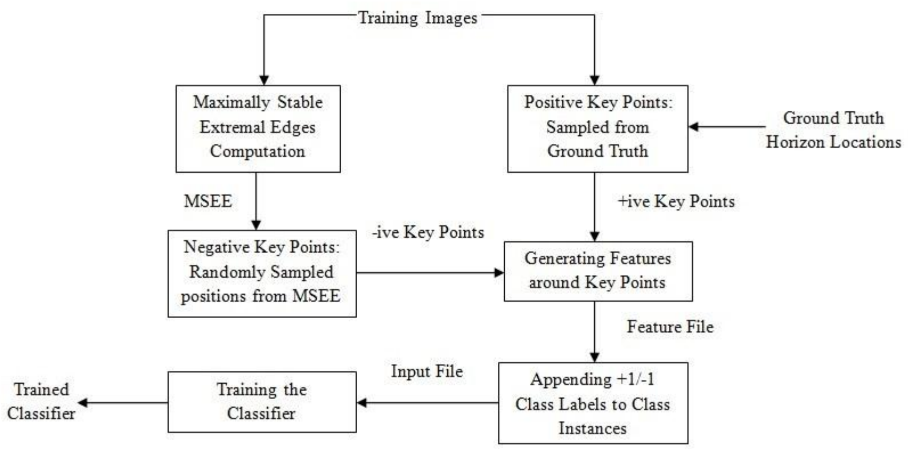
Training Images
Maximally Stable Extremal Edges Computation
Positive Key Points: Sampled from Ground Truth
Ground Truth Horizon Locations
MSEE
+ive Key Points
-ive Key Points
Negative Key Points: Randomly Sampled positions from MSEE
Generating Features around Key Points
Feature File
Input File
Appending +1/-1 Class Labels to Class Instances
Training the Classifier
Trained Classifier

Figure 3.4: Flow diagram for the training phase of the first proposed approach.

3.2 Horizon Line Detection

This section describes various steps involved for the detection of horizon line in a given query image. Figure 3.5 shows various steps of the test phase of our first proposed approach.

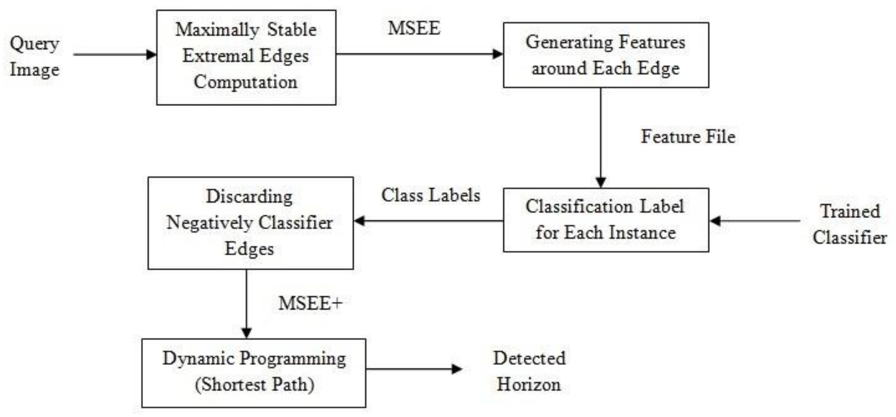
Query Image
Maximally Stable Extremal Edges Computation
MSEE
Generating Features around Each Edge
Feature File
Classification Label for Each Instance
Trained Classifier
Class Labels
Discarding Negatively Classifier Edges
MSEE+
Dynamic Programming (Shortest Path)
Detected Horizon

Figure 3.5: Flow diagram for the testing phase of the first proposed approach.

3.2.1 Filtering MSEE Pixels

On the testing side, MSEE image $E(x,y)$ is generated for a given query image according to equation 3.1 . Each of the edge pixels in MSEE image is treated as a key point; the chosen descriptor around it is computed and then classified as horizon or non-horizon by the trained classifier. The resultant edge image comprising of only horizon classified edges is named E_+. If the classifier is assumed to be a binary function assigning 1/0 labels to the input edge pixel then mathematically E_+ can be written as,

$$E_+(x,y) = \begin{cases} 1, & \text{if } E(x,y) = 1\& \text{ Classifier}[D(E(x,y))] = 1. \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

where, D is the chosen descriptor (or concatenated descriptor) vector around the edge pixel from $E(x,y)$. In addition to the reduction caused by MSEE as shown in figure 3.1, horizon pixel classification further reduced the number of horizon candidate edges significantly. Figure 3.6 shows an example to highlight the significant reduction in number of horizon candidate edges for a query image.

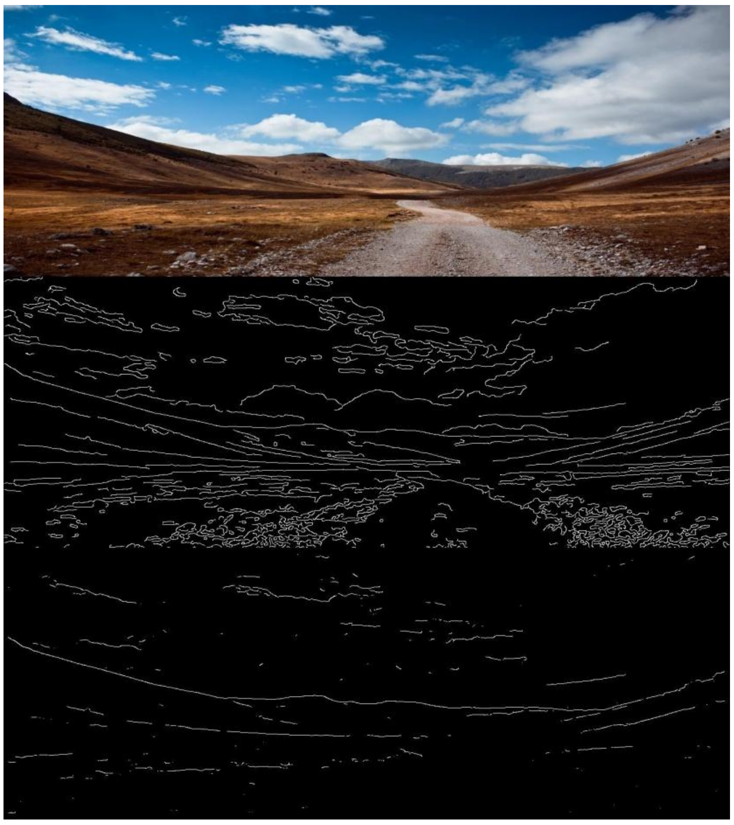

Figure 3.6: A sample novel image (row1), corresponding MSEE (row2) and MSEE$_+$ (row3) Images. Note the reduction in number of edges due to the classifier.

3.2.2 Graph Formulation for Dynamic Programming

To apply Dynamic Programming, instead of using the output of edge detector; we use this horizon classified edge image i.e. E_+ to formulate the multi-stage graph $G(V,E,\Psi,\Phi)$. So, the equations 2.1 and 2.2 are modified accordingly.

$$\Psi(i,j) = \begin{cases} l, & \text{if } E_+(x,y) = 1. \\ \infty, & \text{if } E_+(x,y) = 0. \end{cases} \tag{3.3}$$

$$\Phi(i,k,j) = \begin{cases} |i-k|, & \text{if } E_+(i,j) = E_+(k,j+1) = 1 \\ & \text{and } |i-k| \leq \delta \\ \infty, & \text{otherwise.} \end{cases} \tag{3.4}$$

Since, the number of candidate horizon edges is reduced considerably due to the use of MSEE and the trained classifier; we do not enforce the bias towards horizon solutions in the upper half so equivalent to equation 2.3 is skipped in our formulation. However, any kind of gaps are filled following the conventional method as explained in Chapter 2. Next two nodes i.e. a source and a sink are added, essential links between them and nodes in stages 1 and N are established with zero cost and DP is used to find the horizon line.

3.3 Proposed Nodal Costs

Lie et al. [3] proposed the use of edges where they use binary costs to encode the information in the multi-stage graph about a node being an edge pixel or not and then initialize the dummy nodes to high costs. Although we reduce the number of horizon candidate edges considerably by the use of MSEE and trained classifiers; using only the information about pixels being edge or non-edge is not enough to initialize the nodal costs as it is possible for DP to choose falsely classified horizon edges as part of the solution. We propose the use of various nodal costs that provide further evidence about the positively classified edges for being true horizon edges. Specifically, we have investigated the following nodal costs.

3.3.1 Gradient Information

We use the information due to gradient magnitudes and difference of gradient magnitudes to initialize the node costs of our multi-stage graph. Unlike Lie et al.[3] and others who formulate the multi-stage graph based only on edges we propose here to make a dense multi-stage graph where each pixel would be a node of the graph and be connected to it neighbors in the next stage.

However, the nodes are initialized according to the gradient information. As gradient magnitudes are used as an intermediate part of edge detection the DP should find a solution where the sum of the gradient magnitude is maximized but for continuity we should also enforce that the difference between the gradient magnitudes of two adjacent neighbors should be minimized. We should also note that gradient based approach does not involve any kind of training and is presented here to establish that using just the constant low and high costs for edge and non-edge pixels (horizon classified edges in case of Ahmad et al. [1] and Hung et al. [2]) are not enough to find the accurate horizons.

In gradient based approach; given a query image $Q(x,y)$ the gradient magnitude for each pixel of the image is computed. Mathematically,

$$\nabla(x, y) = \Gamma[Q(x, y)] \tag{3.5}$$

Where, Γ is the function which takes a gray scale image as an input and returns the corresponding gradient magnitude image ∇. Next, the difference of the gradient magnitude image is computed. Since, a node i in stage j can be connected to as many nodes as defined by the δ neighborhood; one should generate as many gradient difference images where the difference should be taken with the node in next stage to which the current node is being connected. The equation 3.6 below shows the making of difference of gradient mask for connections at the same level.

$$d\nabla(i, j) = |\nabla(i, j) - \nabla(i, j+1)| \tag{3.6}$$

Since, we want to maximize the gradient magnitude while minimizing the difference of gradient magnitude we normalize the magnitude and difference images between 0 and 1. The nodal costs Ψ(*i,j*) of the graph are then set as a weighted combination of these two images depending upon to which node in the next stage the current node is being connected; identified by the sub-script k in equation 3.7 below.

$$C_1(i, j) = \Psi(i, j) = w * d\nabla_k(i, j) + (1 - w) * (1 - \nabla(i, j)) \tag{3.7}$$

where w is the weight assigned to the difference of magnitude and the gradient magnitude image; we use a value of 0.5 hence equally weighing both. Next, the links costs may be initialized the way in equation 3.4; however for our experiments we consider all the link weights to be equal and set them to zero since we are considering a small neighborhood i.e. δ = 1.

3.3.2 Classified Binary Edges

The 2nd formulation that we investigate is fairly similar to Lie et al. [3] and others [1, 2] i.e. we use the binary edge information as the nodal cost. The only difference is that we apply the DP on the graph formulated by reduced number of edges due to MSEE and trained classifier. So eventually equations 3.2 through 3.4 are used to initialize the graph and set the node/link costs. We use SIFT-HOG classified edges as we would show in the result SIFT-HOG concatenation outperforms all other feature choices.

3.3.3 Classified Edge Score

As described earlier using a fixed low cost for edge pixels provides only partial information and no confidence at all to compare two positively classified nodes where one might be misclassified. To enforce this knowledge in our dynamic programming formulation we propose a twofold use of the classifier; first to distinguish between horizon and non-horizon edges as realized by equation 3.2 and second to provide a confidence about and edge pixel of horizon-ness. We normalize the raw scores provided by the classifier between 0 and 1 without using any thresholding. The node costs are then initialized by the actual classification scores instead of initializing all positively classified edges to fixed low cost. The equation 3.3 would be altered to reflect this information where Ω is realized as a classifier which returns a value between [0–1]. Since, we want to find a shortest path through DP we assume that the values have been reversed so a smaller value reflects an edge pixel is more probable to be a horizon pixel.

$$C_2(i,j) = \Psi(i,j) = \begin{cases} \Omega(E_+(x,y)), & \text{if } E_+(x,y) = 1. \\ \infty, & \text{if } E_+(x,y) = 0. \end{cases} \tag{3.8}$$

3.3.4 Classified Edge Score + Gradient Information

In this formulation we combine the classifier information with the gradient information. By fusing equations 3.7 and 3.8 we get a new initialization for the nodal costs; where, w_2 is a scalar and we use 0.5 values to weight both the scores equally.

$$\Psi(i,j) = \begin{cases} w_2 * C_2(x,y) + (1 - w_2) * C_1(x,y), & \text{if } E_+(x,y) = 1. \\ \infty, & \text{if } E_+(x,y) = 0. \end{cases} \tag{3.9}$$

Chapter 4

Method 2: Edge-less Horizon Line Detection

Our second proposed approach applies DP on a classification map that associates with each pixel a classification score which can be interpreted as the confidence of the pixel being part of the horizon line. Most importantly, it does not rely on edge detection, therefore, it does not require performing gap filling or introducing dummy nodes as is required by Lie et al. [3] and also by our first proposed method. Moreover, again we do not force the nodes in stages 1 and N to be associated with their vertical positions since the assumption of the horizon line being present in the upper half of the image could be violated (e.g., due to the rover moving on a peak and looking towards the horizon). The resulting DCSI is used to form an $M \times N$ multi-stage graph without any node initialization. Once we have introduced the source/destination nodes s/t and decided on the value of δ, any shortest path finding algorithm can be used to find the path that maximizes the sum of classification scores. We will later show that the number of nodes per stage can be significantly reduced by only considering the pixels with the m highest classification scores where m is a parameter; we refer to this reduced map as mDCSI map. Using fewer nodes per stage does not affect accuracy while it speeds up computations considerably. Figure 4.1 illustrates the main steps of our second proposed approach.

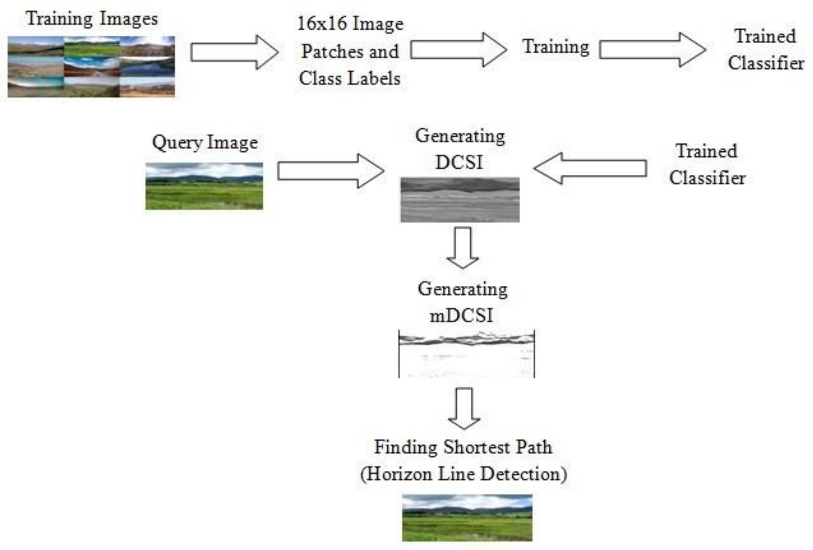

Figure 4.1: Main steps of the training/testing phases of second proposed horizon line detection approach.

4.1 Pixel Classification

For classification, we have experimented with two classifiers: SVM [26] and a CNN [27]. Each classifier is trained using horizon and non-horizon image patches from a set of training images where the horizon line has been extracted manually (ground truth) as detailed in section 3.1. Specifically, for each training image, we select N points uniformly from the ground truth; an equal number of points is randomly selected from non-horizon locations as shown in figure 3.3. We take a 16 × 16 normalized image patch around each sampled point and the resulted 256-D vector is used for training the classifiers. It should be mentioned that we are using pixel intensities as features here as compared to texture features that have been used for first method. The decision is partially biased due to the fact we are training a CNN classifier [27] which is expected to find features itself instead of hand crafted features. For a fair comparison between SVM and CNN classifiers, normalized pixel intensities are used as features for both. The pixel intensities are normalized between -1 and 1. For the CNN classifier, we use an architecture comprising of 2 Convolution(C)-Sub-sample(S) layers. The first C-S layer is comprised of 4 levels with a convolution(C) mask of 5 × 5 and a sub-sampling(S) mask of 2×2. The second C-S layer is comprised of 8 levels with a C mask of 3×3 and an S mask of 2 × 2. For the SVM classifier, we use a linear kernel as it was found to be equally good as the RBF and polynomial kernels. We have only used 9 images for training the classifiers with 343 positive (horizon) and 343 negative (non-horizon) examples extracted from each image. Figure 3.3 shows an example of horizon (red) and non-horizon (blue) training samples. It is important to note that for both approaches; same number of training images and same key points are used, the difference being the feature and classifier choices.

4.2 Horizon Line Detection

4.2.1 Dense Classifier Score Image (DCSI)

Once the classifiers have been trained, the DCSI can be generated for a given test image. For each pixel location in the test image, a 16×16 patch of pixel intensities around the pixel is extracted. The normalized intensities are then used to form a 256-D vector $V(x,y)$, which is fed to the classifier. The classification score is then associated with that pixel location. Classification scores are normalized in the interval [0, 1]; the resultant scores form the DCSI which is denoted as $D(x,y)$. In essence, $D(x,y)$ can be interpreted as a probability map which reflects the likelihood of a pixel belonging to the horizon line. Figure 4.2 shows the DCSIs for two sample images.

$$D(x, y) = \Gamma(V(x, y)) \tag{4.1}$$

4.2.2 Reduced Dense Classifier Score Image (mDCSI)

Although the full DCSI can be used for horizon line detection, we have found that keeping only the m highest classification scores in each column does not compromise accuracy while reducing computations. This is because the highest classification scores are typically concentrated within a small band around the horizon line. We refer to the reduced DCSI as *mDCSI*. The multi-stage graph corresponding to the mDCSI contains fewer vertexes; as a result, fewer paths need to be considered when searching for the shortest path which results in considerable speedups. In our experiments, we have found that by keeping the highest 50 classification scores yields accurate horizon line detections.

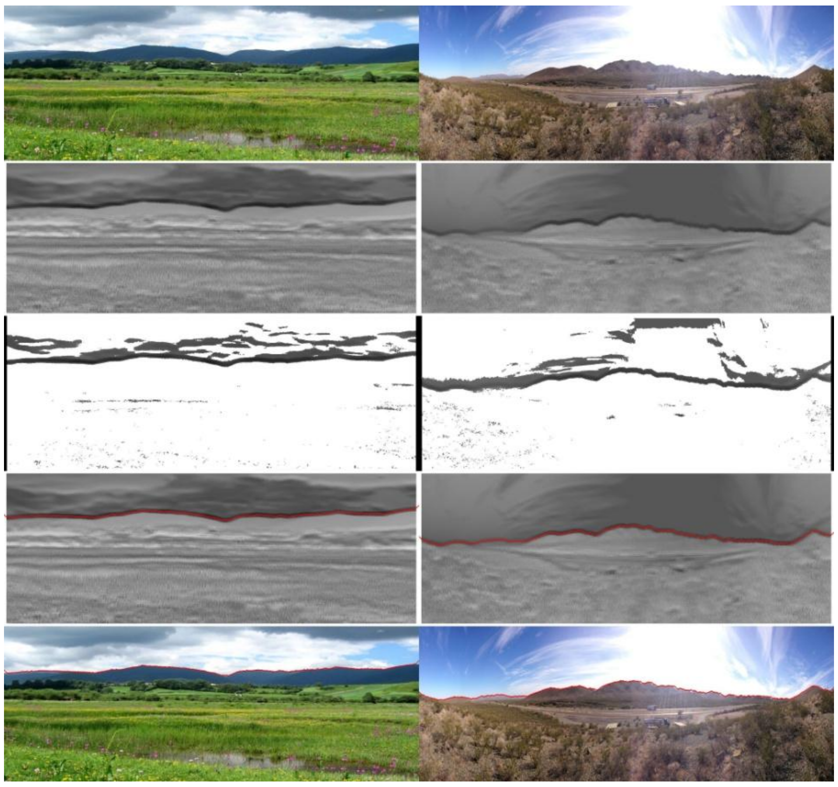

Figure 4.2: Sample test images (row1), respective DCSIs (row2), mDCSIs (row3), shortest path solutions (row4, highlighted in red), and detected horizon lines (row5, highlighted in red).

From an implementation point of view, the mDCSI is computed as follows:

$$mD(x,y) = \begin{cases} D(x,y), & \text{if } x \in L(m)_y \\ l, & \text{otherwise; } l < 0. \end{cases} \tag{4.2}$$

where, mD corresponds to the mDCSI and D corresponds to the DCSI. If the x-th pixel (node) in column (stage) y belongs to the list $L(m)_y$ of indices corresponding to the highest m classification scores, the classification score from $D(x,y)$ is used; otherwise, the score is set to a low score l where l is smaller than the smallest score returned by the classifier. Figure 4.2 shows examples of the respective mDCSIs.

4.2.3 Horizon Line Detection as a Shortest Path Problem

In earlier approaches e.g. [1, 2, 3], the edge map (or classified edge map) was used to form the multi-stage graph; in these approaches, gap filling is an essential step in extracting the horizon line. Since this approach does not rely on edge maps, we do not need to worry about gap filling. In our approach, the mDCSI is used to create an $M \times N$ graph $G(V,E,\Psi,\Phi)$ with node costs initialized to mD(x,y). So, equation 2.1 can directly transform into,

$$\Psi(i,j) = mD(x,y) \tag{4.3}$$

Since the resulted graph is a dense graph, each node i in stage(column) j is connected to three nodes i, $i-1$ and $i+1$ in stage(column) $j+1$ (i.e., $\delta = 1$). These connections are considered as edges with zero costs similar to first approach. This is in contrast to conventional approaches ([3]) where the absolute difference between the positions of nodes in two stages is

$$\Phi(i,k,j) = \begin{cases} 0, & \text{if } |i-k| \leq \delta \\ \infty, & \text{otherwise.} \end{cases} \tag{4.4}$$

Since, the horizon line might not always appear in the upper half of the image, we do not set the nodes in stages 1 and N proportional to their vertical position. Two dummy nodes, s and t, are introduced to the left of stage 1 and to the right of stage N respectively. The edge weights from s to every node in stage 1 and from every node in stage N to node t are set to zero. The shortest path in this graph can be found using DP.

Chapter 5

Experimental Details

5.1 Data Sets and Quantitative Evaluation

To evaluate the performance of our proposed approaches, we have experimented with three different data sets: the City data set, the Basalt Hills data set and Web data set. The City data set consists of 13 images of a small city (Reno) surrounded by mountains. The Basalt Hills data set is a subset of a data set which was generated by placing cameras on an autonomous robot navigating through Basalt Hills [23]. We have chosen 45 images from this data set with considerable viewpoint and scene changes. The most challenging of our data sets is the Web data set which consists of 80 mountainous images that have been randomly collected from the web. This data set includes various viewpoints, geographical and seasonal variations. Our training set in both approaches consists of only 9 images from Basalt Hills data set. The resolution of all images in our data sets is 519×1388. In both proposed approaches same 343 positive and 343 negative key points are chosen from each training image. The positive key points are chosen uniformly from the ground truth horizons whereas the negative key points are chosen randomly from the non-horizon edge locations belonging to both sky (edges due to clouds etc.) or non-sky regions.

To quantitatively evaluate the performance of the proposed approaches, we have manually extracted the horizon line (ground truth) in all the images of our data sets. To evaluate the proposed approaches, the detected and true horizon lines are compared by calculating a pixel-wise absolute distance S between them. For each column, the absolute distance between the detected and ground truth pixels is computed and summed over the entire number of columns in the image. The resultant distance is normalized by the number of columns in the image, yielding the average absolute error of the detected horizon line from ground truth. Since nodes in a particular stage are not allowed to be connected to nodes in the same stage, the true and detected horizon lines are bound to have the same number of columns/stages in the image/graph. Hence, there is a one-to-one correspondence between the pixel locations in the true and detected horizon pixel locations:

$$S = \frac{1}{N}\sum_{j=1}^{N} |P_{d(j)} - P_{g(j)}| \qquad (5.1)$$

where $P_{d(j)}$ and $P_{g(j)}$ are the positions (rows) of the detected and true horizon pixels in column j and N is the number of columns in the test image. We have evaluated our first proposed approach on the Basalt Hill and Web data sets whereas the second method is tested over all three data sets. Both methods are compared for all the images against the tradition edge based approach of Lie et al. [3]

5.2 Results for Method1

5.2.1 Effect of MSEE

As a first step to our proposed approach we compute Maximally Stable Extremal Edge (MSEE) Images for all the images in our data set. We compare the number of edges survived after MSEE with the number of edges found by Canny edge detector [30] of Matlab and see considerable reduction in the number of candidate horizon edges which are further reduced by the classifier. Table 5.1 shows the average percentage reduction for each image of both data sets used for this approach.

Table 5.1: Importance of MSEE

Data Set	Average % Reduction
Basalt Hills	66.37
Web	43.45

5.2.2 Reduction in Horizon Candidate Edges due to Different Features

We investigate various texture descriptors and their combinations as the feature choices to train our SVM classifier. We perform a 5-fold validation on the Basalt Hills data set where for each fold the data set is divided into non-overlapping train (9 images) and test sets (36 images). Since, we have the ground truth horizons at our disposal we know which edges belong to true horizon and which do not. Table 5.2

shows the percentage false positive and false negative errors averaged over the five folds of training and the respective standard deviations. Figure 5.2.2 shows a graphical view of the same information. Since, false negative error is of more importance we choose the classifier based on SIFT-HOG combination for further evaluation, highlighted in table 5.2 .

Table 5.2: % FP and %FN Errors due to Various Features

Feature	%FN		%FP	
	Mean	**Std. Dev.**	**Mean**	**Std. Dev.**
SIFT	1.0224	0.7890	17.8090	5.6089
LBP	5.0332	8.3747	10.6366	8.0770
HOG	2.3285	2.3498	11.2331	5.6616
SIFT+LBP	0.6915	1.1827	10.6065	5.8949
SIFT+HOG	**0.6624**	**0.7436**	**11.0801**	**5.1797**
LBP+HOG	3.3647	3.6415	10.0737	6.394
SIFT+LBP+HOG	7.1887	8.1177	4.8302	4.0438

5.2.3 Best Nodal Cost

We compare our proposed formulations 3 against the state of the art horizon detection method based on edges and DP i.e. Lie et al. To compare the detected horizon lines found by each method with the ground truth horizons,

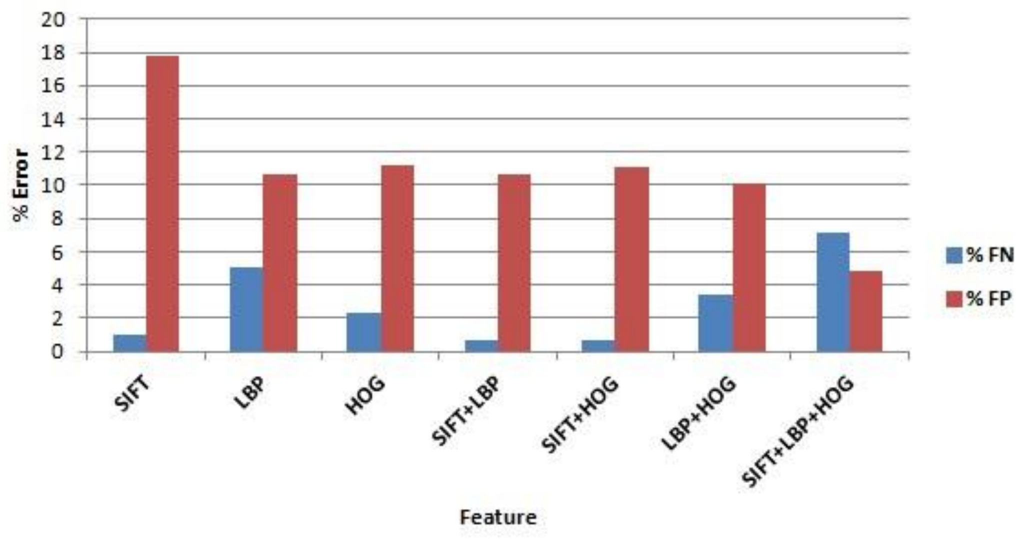

Figure 5.1: Mean of % False Positive and False Negative Errors when various Features and their combinations used for training the SVM Classifier.

We compute an average pixel wise absolute error using 5.1. Since, in all of our formulations we do not allow nodes to be connected within the same stage, there exists a one-to-one mapping between the pixels of the detected (d) horizon and the ground (g) truth horizon. For each of our formulation and Lie at al. [3] we compute the average and standard deviation over all images in the data set listed in the table 5.3. Clearly, SIFT+HOG Scores outperforms all others strengthen our understanding that using only edge information is not enough for the nodal costs.

5.3 Results for Method2

5.3.1 Quantitative Evaluation

For the second proposed method the results have been computed for all three data sets mentioned earlier and same absolute average error is computed as listed in equation 5.1. Table 5.4 shows the average absolute error for all the images in each data set, both for the SVM and CNN classifiers. For comparison purposes, we also provide results based on the method of Lie et al. [3]. It is interesting to note that although our methods were trained using a very small number of images from the same data set, they generalize very well to images from other data sets, such as the Web data set which is very different from the training data set.

Table 5.3: Approach1: Average Absolute Errors

Nodal Costs	Basalt Hills		Web	
	μ	σ	μ	σ
Lie et al. [3] (Edges)	5.5548	9.4599	9.1500	17.9195
Gradient Info.	3.9908	6.3530	11.8641	26.8084
SIFT+HOG Edges	0.5783	1.0227	**0.8698**	**1.0366**
SIFT+HOG Scores	**0.4124**	**0.8120**	0.9704	1.5698
SIFT+HOG Scores + Gradient	0.4358	0.8124	1.3016	3.9814

Table 5.4: Approach2: Average Absolute Errors

Data Set	Proposed Approach				Lie et al. [3]	
	SVM-mDCSI		CNN-mDCSI			
	μ	σ	μ	σ	μ	σ
City	0.7244	0.1777	1.2129	2.3597	6.2342	10.8206
Basalt Hills	1.0101	0.2887	0.7573	0.2295	5.5548	9.4600
Web	1.2854	1.1988	1.4121	1.4860	9.1500	17.9196

5.3.2 Comparing Classifiers

Comparing the two classifiers used in our experiments for second approach, the CNN classifier outperforms the SVM classifier for the Basalt Hills data set while SVM outperforms CNN on the other two data sets. This indicates that the features found by the CNN classifier might not generalize well to different data sets. Figure 5.2 shows some representative DCSI results using the SVM and CNN classifiers. It is worth noting that the CNN classifier provides a crispier DCSI, having a narrower band around the true horizon line as compared to the DCSI produced by SVM. It might be possible to further improve our best results by combining the SVM and CNN classifiers but we have not experimented with this idea.

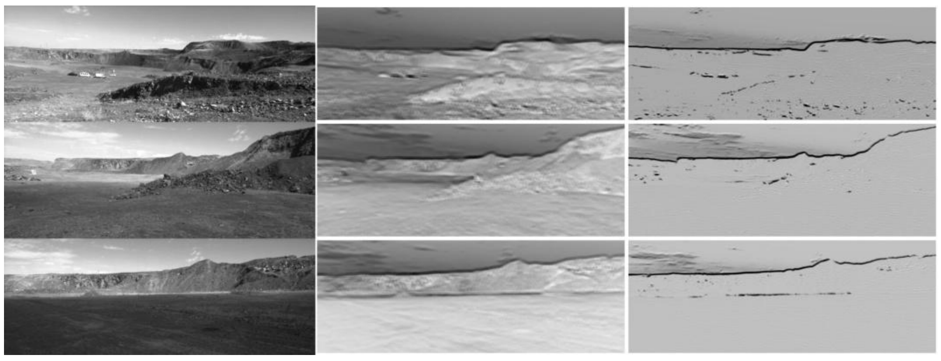

Figure 5.2: Test images (column1), corresponding SVM-DCSIs (column2) and corresponding CNN-DCSIs (column3).

5.4 Discussion and Some Visual Results

5.4.1 Failure of Lie et al. [3]

Our experimental results illustrate that the proposed approaches outperform the traditional approach of [3] based on edge maps. In particular, both the average error and standard deviation of the traditional approach are much higher than the proposed approaches based on SVM or on both SVM and CNN classifiers in second approach. To better illustrate the performance of the traditional approach, we have identified specific examples where it fails to detect the true horizon line or it misses parts of it. The main reason for this is due to the presence of big gaps in the edge map. This might happen due to different reasons, for example, horizon edges might not be strong enough or stronger edges might exist close to the horizon line due to various environmental effects such as clouds. Although Lie et al. have proposed a gap filling approach by introducing dummy nodes with high costs, this does not always work well, for example, when gaps are long and edges from clouds are close to the horizon line. In these cases, it is likely that the DP approach might find a low cost path by taking an alternative path. Figure 5.3 (row 1) shows two examples where the method of Lie et al. has failed to find a good solution due to edge gaps and the presence of clouds; the proposed method (approach2) was able to find the true horizon line with high accuracy in both cases (row 2). Figure 5.4 shows zoomed sub-images of the left column images of 5.3 (i.e., Lie et al.) for better visualization.

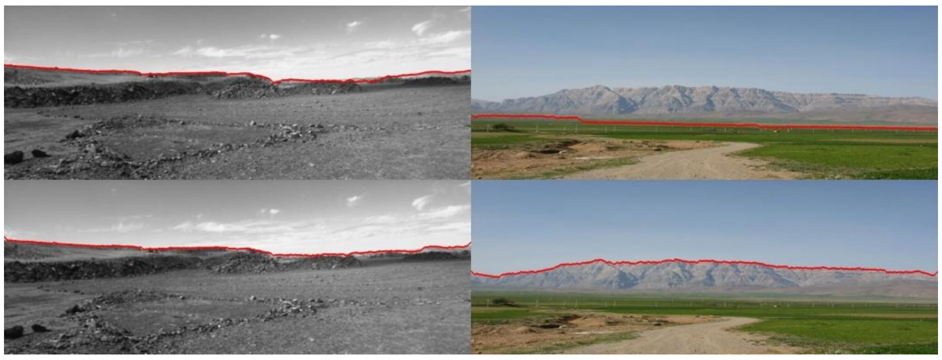

Figure 5.3: Examples illustrating: [row 1] missing the horizon line or parts of it due to edge gaps (Lie et al.), and [row 2] detecting the true horizon line using our approach 2 (SVM).

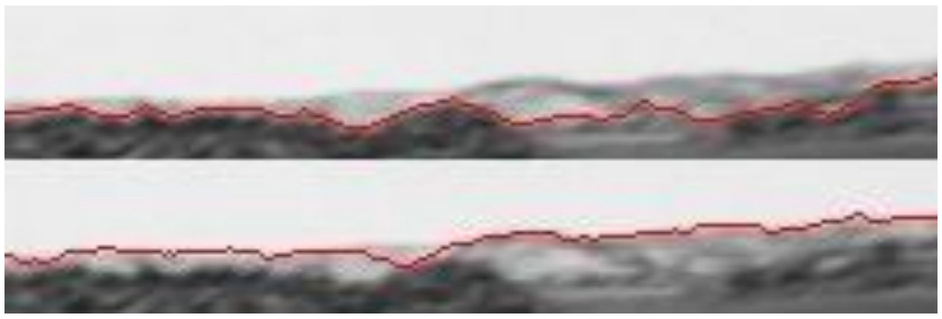

Figure 5.4: Zoomed sub-images of the left column images of Figure 5.3

Another reason affecting the performance of Lie et al. is the underlying assumption of the horizon boundary is close to the top of the image. When clouds are present in an image, a portion of the true horizon may be missed if the true horizon line is below the clouds due to the bias introduced towards solutions closer to the top of the image. Figure 5.5 shows some examples where the approach of Lie et al. has found solutions consisting of both horizon line segments as well as cloud edge segments. Our approach, on the other hand, was able to find the correct solution for these cases as it does not make such assumptions.

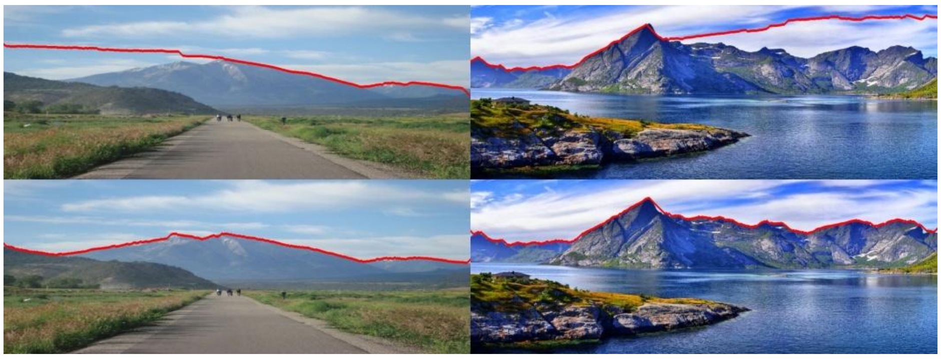

Figure 5.5: Examples illustrating: [row 1] missing parts of the horizon line due to the assumption that the horizon line is close to the top of the image (Lie et al.), and [row 2] detecting the true horizon line using the proposed approach 2 (SVM).

5.4.2 Failure of Proposed Approaches

Attempting to better understand why the proposed approach (approach2) sometimes finds poor solutions, we have identified two main reasons. First, disallowing nodes in some stage to connect with nodes in the same stage but only with nodes in the next stage. In the multi-stage graph formulation of Lie et al., a node at stage j is only allowed to be connected to nodes at stage j+1 which is problematic when the horizon line has high slope (i.e., steep peaks). Figure 5.6 (row 1) shows an example due to this issue. This issue can be easily rectified by allowing the nodes in some stage to be connected both with nodes in the next stage as well as nodes in the same stage. Figure 5.6 (row 2) shows the solution obtained by allowing connections within the same stage. Allowing connections within the same level will of course increase time requirements as it increases the number of paths that need to be explored.

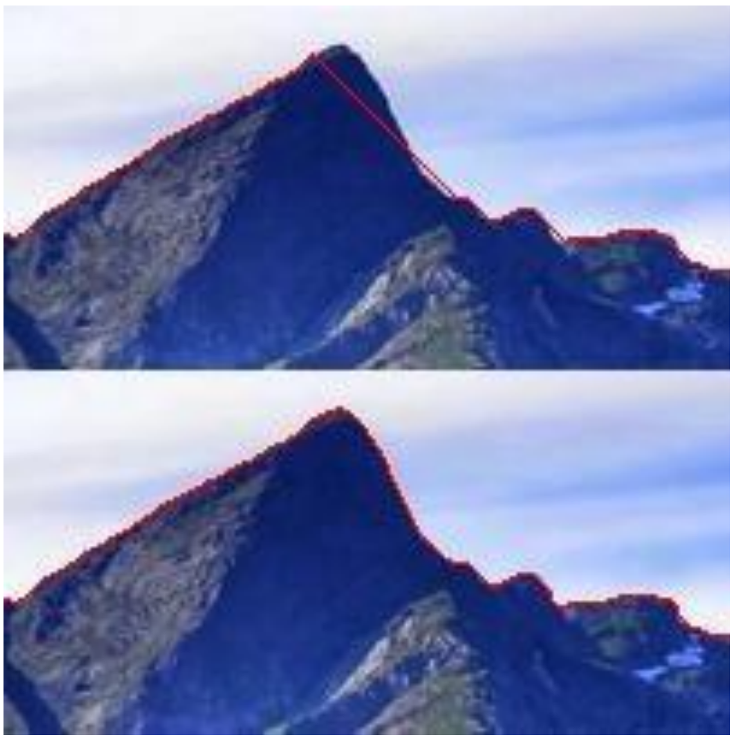

Figure 5.6: [row 1] effect of not allowing node connections within the same stage; [row 2] solution obtained by allowing node connections within the same stage.

The second most important reason affecting the performance of the proposed approach is due to using a very small set of training images (i.e., only 9 images from the same data set). Figure 5.7 shows examples where our method has failed to find good solutions.This is due to the fact that our second approach totally relies on the classification whereas the first one uses edges or gradient information along with classification evidence. This issue of generalization can be addressed by increasing the train set and making it more versatile. By carefully analyzing our results on the Web data set, our second approach failed to find a good solution in 9 out of the 80 images due to using a small training set. Removing these images from the data set improves the average error of our approach using the SVM classifier from 1.2854 to 0.9227 and reduces the variance from 1.1988 to 0.3637.

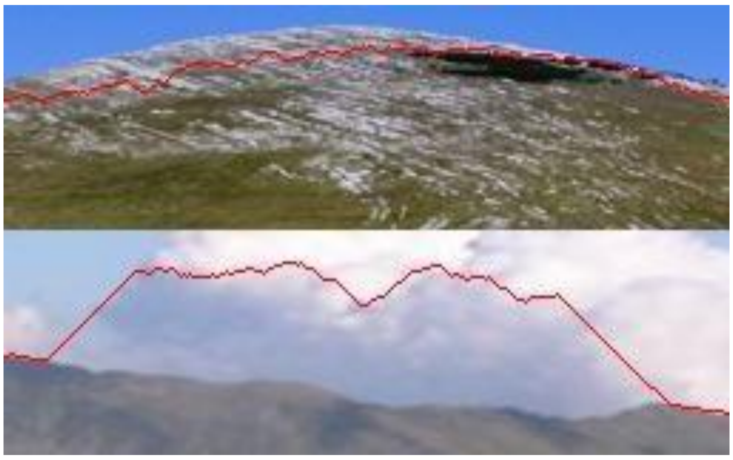

Figure 5.7: Examples illustrating the inability of the proposed method to find a good solution due to the lack of sufficient training data.

Figure 5.8 shows some representative results of our horizon detection approach (approach2) using images from all three data sets.

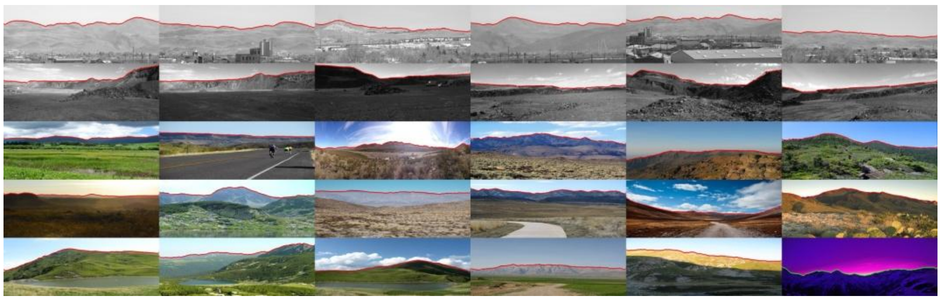

Figure 5.8: Sample results illustrating our horizon line detection approach: City data set(row1), Basalt Hills data set (row2) and Web data set (row 3 through 5). Detected horizon lines are highlighted in red/green.

Chapter 6

Conclusions

In this thesis we have proposed two novel horizon detection approaches based on supervised machine learning. The first approach relies on edge detection but unlike conventional methods uses trained classifier to reduce the number of horizon candidate edges considerably. Further classifier is used to provide evidence about the likelihood of an edge being horizon or non-horizon. We have investigated various texture features and nodal costs in the framework of our proposed method. Our second proposed approach does not rely on edge detection as a pre-processing step. The key idea is classifying each pixel as horizon or non-horizon and applying DP on the classification map to extract the horizon line.

Both proposed approaches do not make any assumptions about the horizon being a straight line or being close to the top of the image which are common in conventional methods. The proposed approaches use a very small number of images to train the horizon classifiers and outperform traditional approaches based on edge maps on three challenging data sets. For future work, we plan to investigate the suitability of the proposed horizon detection methods for pose estimation and localization of planetary rovers and UAVs.

Appendix

Publications

1.**Touqeer Ahmad**, George Bebis, Emma Regentova, Ara Nefian and Terry Fong: *Coupling Dynamic Programming with Machine Learning for Horizon Line Detection*. International Jounal on Artificial Intelligence Tools, 2014. [**Submitted**]

2.**Touqeer Ahmad**, George Bebis, Monica Nicolescu, Ara Nefian and Terry Fong: *An Edge-less approach to Horizon Line Detection*. IEEE International Conference on Multimedia and Expo (ICME'15), Torino, Italy, June 29-July 03, 2015. [**Submitted**]

3.**Touqeer Ahmad**, George Bebis, Emma Regentova, Ara Nefian and Terry Fong: *An Experimental Evaluation of Different Features and Nodal Costs for Horizon Line Detection*. 10th International Symposium on Visual Computing (ISVC'13), Las Vegas, USA, December 8-10, 2014.

4.**Touqeer Ahmad**, George Bebis, Emma Regentova and Ara Nefian: *A Machine Learning Approach to Horizon Line Detection Using Local Features*. 9th International Symposium on Visual Computing (ISVC'13), Crete, Greece, July 29-31, 2013.

5.Ali Pour Yazdanpanah, Emma E. Regentova, Ajay Kumar Mandava, **Touqeer Ahmad** and George Bebis: *Sky Segmentation by Fusing Clustering with Neural Networks*. 9th International Symposium on Visual Computing (ISVC'13), Crete, Greece, July 29-31, 2013.

Bibliography

[1] T. Ahmad, G. Bebis, E. Regentova and A. Nefian, A Machine Learning Approach to Horizon Line Detection using Local Features, *Proceedings of 9th International Symposium on Visual Computing (ISVC)*. 2013.

[2] Y. Hung, C. Su, Y. Chang, J. Chang and H. Tyan, Skyline Localization for Mountain Images, *Proceedings of International Conference on Multimedia and Expo (ICME)*. 2013.

[3] W. Lie, T. C.-I. Lin , T. Lin , and K.-S. Hung, A robust dynamic programming algorithm to extract skyline in images for navigation, in *Pattern Recognition Letters*, **26**(2)(2005.)221–230.

[4] Nasim S. Boroujeni, S. Ali Etemad and Anthony Whitehead: Robust Horizon Detection Using Segmentation for UAV Applications. *Proceedings of IEEE 2012 Ninth Conference on Computer and Robot Vision*,
2012.

[5] G. C. H. E. de Croon, B. D. W. Remes, C. De Wagter, and R. Ruijsink: Sky Segmentation Approach to Obstacle Avoidance. *IEEE Aerospace Conference*, 2011.

[6] Scott M. Ettinger, Michael C. Nechyba, Peter G. Ifju, and Martin Waszak: Vision-Guided Flight Stability and Control for Micro Air Vehicles. *Proceedings of International Conference on Intelligent Robots and Systems(IEEE/RSJ)*, 2002.

[7] Timothy G. McGee, Raja Sengupta, and Karl Hedrick: Obstacle Detection for Small Autonomous Aircraft Using Sky Segmentation. *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2005.

[8] Saul Thurrowgood, Dean Soccol, Richard J. D. Moore, Daniel Bland, and Mandyam V. Srinivasan: A Vision Based System for Altitude Estimation of UAVs. *Proceedings of International Conference on Intelligent Robots and Systems(IEEE/RSJ)*, 2009.

[9] Sinisa Todorovic, Michael C. Nechyba, and Peter G. Ifju: Sky/Ground Modeling for Autonomous MAV Flight. *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2003.

[10] Vishisht Gupta and Sean Brennan : Terrain Based Vehicle Orientation Estimation Combining Vision and Inertial Measurements. *Journal of Field Robotics*, **25**(3):181 - 202, 2008.

[11] Nghia Ho and Punarjay Chakravarty: Localization on Freeways using the Horizon Line Signature. *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2014.

[12] Steven J. Dumble and Peter W. Gibbens: Efficient Terrain-Aided Visual Horizon Based Attitude Estimation and Localization. *Journal of Intelligent and Robotic Systems*, 2014.

[13] T. Ojala, M. Pietikainen, and T. Maenpaa: Multiresolution grayscale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence(TPAMI)*, **24**(7):971 - 987, 2002.

[14] D. G. Lowe: Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision(IJCV)*, **68**(2):91 - 110, 2004.

[15] Navneet Dalal and Bill Triggs: Histograms of Oriented Gradients for Human Detection. *Proceedings of Computer Vision and Pattern Recognition(CVPR)*, 2005.

[16] http://www.vlfeat.org/index.html

[17] Georges Baatz, Olivier Saurer, Kevin Koser, and Marc Pollefeys: Large Scale Visual Geo-Localization of Images in Mountainous Terrain *ECCV*, 2012.

[18] W. Liu and C. Su: Automatic Peak Recognition for Mountain Images *Advanced Technologies, Embedded and Multimedia for Human-centric Computing*, 2014.

[19] Sergiy Fefilatyev, Volha Smarodzinava, Lawrence O. Hall and Dmitry B. Goldgof: Horizon Detection Using Machine Learning Techniques. *ICMLA.*, 17-21, 2006.

[20] E. Gershikov, T. Libe and S. Kosolapov: Horizon Line Detection in Marine Images: Which Method to Choose? *International Journal on Advances in Intelligent Systems*, **6**(1-2):79 - 88, 2013.

[21] Byung-Ju Kim, Jong-Jin Shin, Hwa-Jin Nam and Jin-Soo Kim: Skyline Extraction using a Multistage Edge Filtering *World Academy of Science, Engineering and Technology 55*, 2011.

[22] A. P. Yazdanpanah, E. E. Regentova, A. K. Mandava, T. Ahmad and G. Bebis: Sky Segmentation by Fusing Clustering with Neural Networks. *Proceedings of 9th International Symposium on Visual Computing (ISVC)*. 2013.

[23] A. V. Nefian, X. Bouyssounouse, L. Edwards, T. Kim, E. Hand, J. Rhizor, M. Deans, G. Bebis and T. Fong: Planetary Rover Localization within Orbital Maps. *Proceedings of International Conference on Image Processing(ICIP)*. 2014.

[24] T. Ahmad, G. Bebis, E. Regentova, A. Nefian and T.Fong: An Experimental Evaluation of Different Features and Nodal Costs for Horizon Line Detection, *Proceedings of 10th International Symposium on Visual Computing (ISVC)*. 2014.

[25] J. Matas, O. Chum, M. Urban, and T. Pajdla: Robust Wide Baseline Stereo from Maximally Stable Extremal Regions, *Proceedings of British Machine Vision Conference*, pages 384-396, 2002.

[26] C. Cortes and V. Vapnik: Support-vector networks., *Machine Learning*, **20**(3):273 - 297, 1995.

[27] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient based learning applied to document recognition., *PIEEE*, **86**(11):2278–2324, 1998.

[28] L. Baboud, M. Cadik, E. Eisemann and H. -P Seidel: Automatic Phototo-Terrain Alignment for the Annotation of Mountain Pictures. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[29] E. Tzeng, A. Zhai, M. Clements, R. Townshend and A. Zakhor: UserDriven Geolocation of Untagged Desert Imagery Using Digital Elevation Models. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops(CVPRW)*, 2013.

[30] J. Canny: A computational approach to edge detection., *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **8**(6):679– 698, 1986.