A Machine Learning Approach to Horizon Line Detection Using Local Features

Touqeer Ahmad[1], George Bebis[1], Emma Regentova[2], and Ara Nefian[3]

[1] Department of Computer Science and Engineering, University of Nevada, Reno
sh.touqeerahmad@gmail.com,bebis@cse.unr.edu

[2] Department of Electrical and Computer Engineering, University of Nevada, Las Vegas
Emma.Regentova@unlv.edu

[3] Carnegie Mellon University and NASA Ames Research
ara.nefian@nasa.gov

Abstract. Planetary rover localization is a challenging problem since no conventional methods such as GPS, structural landmarks etc. are available. Horizon line is a promising visual cue which can be exploited for estimating the rover's position and orientation during planetary missions. By matching the horizon line detected in 2D images captured by the rover with virtually generated horizon lines from 3D terrain models (e.g., Digital Elevation Maps(DEMs)), the localization problem can be solved in principle. In this paper, we propose a machine learning approach for horizon line detection using edge images and local features (i.e., SIFT). Given an image, first we apply Canny edge detection using various parameters and keep only those edges which survive over a wide range of thresholds. We refer to these edges as Maximally Stable Extremal Edges (MSEEs). Using ground truth information, we then train an Support Vector Machine (SVM) classifier to classify MSEE pixels into two categories: horizon and non-horizon. Each MSSE pixel is described using SIFT features which becomes input to the SVM classifier. Given a novel image, we use the same procedure to extract MSSEs; then, we classify each MSEE pixel as horizon or non-horizon using the SVM classifier. MSEE pixels classified as horizon are then provided to a Dynamic Programming shortest path finding algorithm which returns a consistent horizon line. In general, Dynamic Programming returns different solutions (i.e., due to gaps) when searching for the optimum horizon line in a left-to-right or right-to-left fashion. We use the actual output of the SVM classifier to resolve ambiguities in places where the left-to-right and right-to-left solutions are different. The final solution, is typically a combination of edge segments from the left-to-right or right-to-left solutions. Moreover, we use the SVM classifier to fill in small gaps in the horizon line; this is in contrast to the traditional dynamic programming approach which relies on mere interpolation. We report promising experimental results using a set of real images.

1 Introduction

A key function of a mobile robot system is its ability to localize itself accurately [1]. This problem is more challenging in space missions due to lack of conventional localization methods (e.g., landmarks, maps, GPS etc.). Here, we investigate the problem of robot localization using the horizon line as a visual cue which can be found often in images captured by the rover's camera. Rover localization based on the horizon line consists of two main steps. First, the horizon line needs to be detected. Second, changes in the location and orientation of the rover need to be estimated. This can be done by matching the actual horizon line with a virtually generated horizon line from 3D terrain models such as DEMs. Our focus in this paper is on horizon line detection from rover 2D images.

Horizon line detection has many important applications such as ship detection, flight control and port security [2]. Recently, Baatz et al. [8] have demonstrated the idea of using the horizon line for visual geo-localization of images in mountainous terrain. In their approach, they match the horizon line extracted from RGB images with the horizon lines extracted from DEMs of a predefined large scale region.

Kim et al. [7] have proposed a Multistage Edge Filtering technique to find the horizon line in cluttered backgrounds for UAV navigation. First, they model the clutter first and then, using an iterative approach, they filter out the edges belonging to the clutter. The horizon line is then discriminated from the remaining candidates using length and continuity constraints. Fefilatyev et al. [2] have proposed a machine learning approach to find the horizon line by segmenting the sky and non-sky regions assuming that the horizon line is straight. Various color and texture features (e.g., mean intensity value of three color channels, entropy, smoothness, uniformity, third moment etc.) were used to train several classifiers. Although their approach has shown promising results, their underlying assumption that the horizon line straight is not general enough and is being violated often. McGee et al. [10] have presented a sky segmentation technique based on color where a linear separation between the sky and non-sky regions is found using SVMs. The main drawback of their approach is again the assumption that the horizon line is straight.

The horizon detection method of Ettinger et al. [11] suffers from the same assumption too. They model the sky and non-sky regions using Gaussian distributions and use Bayesian estimation to find the optimum boundary which separates the two distributions. In [12], Croon et al. have addressed this issue by training a classifier (e.g., shallow decision trees, J48 Implementation of C4.5 algorithm) using color and texture features. Their choice of decision trees is motivated by the computational efficiency achieved at run time to perform sky segmentation for static obstacle avoidance by Micro Air Vehicles (MAVs). They have extended the features used in [2] by introducing new features such as cornerness, grayness, and Fisher discriminant features. In comparison to [2], they used an extended database to train their classifier and a large number of features; hence their approach is more robust and capable of finding non-linear horizon lines. Todorovic et al. [13] extended their previous work [11] by eliminating the assumptions about the horizon line being linear and the sky/non-sky regions following a Gaussian distribution. In particular, they built prior statistical models for sky and non-sky regions using color and texture features. They argue about the importance of both color (Hue and Intensity) and texture features (Complex Wavelet Transform) due to the enormous variation in sky and ground appearances. A Hiddern Markov Tree model was trained using these features, yielding a robust horizon line detection algorithm.

Lie at al. [9] have presented a dynamic programming approach to find the horizon line using edges. They formulate the problem of horizon line detection as a multistage graph problem where each column of the edge image behaves as one stage of the graph. Their goal is to find a consistent shortest path extending from the left-most column of the image to the right-most column. The Sobel edge detector was used in their approach. The gaps due to edge detection are filled with dummy nodes using a fanout strategy based on interpolation. A gap tolerance of up to 30 pixels is allowed and a cost is associated with each dummy node based on the size of the gap.

In our approach, we employ Maximally Stable Extremal Edges (MSEE), that is, edges that survive a wide range of thresholds and are considered to be more stable. Our experimental results show that using MSEE eliminates non-horizon edges, reduces computational requirements, and preserves the accuracy of horizon line detection. To further eliminate non-horizon MSEEs, we use an SVM classifier which is trained using SIFT features computed at MSEE pixels. MSEEs pixels classified as horizon are post-processed using Dynamic Programming [9]. In the original approach [9], only one shortest path is found which extends from left-to-right. However, this is not always optimal due to the presence of gaps in the horizon line. Here, we compute the shortest path in both directions and calculate a compound score, based on SVM actual outputs, to resolve ambiguous segments (i.e., segments where the left-to-right and right-to-left paths do not overlap).

The rest of the paper is organized as follows: In section 2 and 3 we describe the main components of the proposed approach. Section 4 describes our experiments and results. The paper is concluded in section 5.

2 Horizon Line Learning

In this section, we describe the steps for learning to detect the horizon line in grayscale images.

2.1 Maximally Stable Extremal Edges (MSEEs)

The idea of extracting MSEEs was inspired from the idea of extracting Maximally Stable Extremal Regions (MSER) [14]. Given a gray scale image, we compute the edge image using the Canny edge detector 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,

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

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 $0.4 \times Th$. It was found through experimentation that $\sigma = 2$ and $k = 3$ were optimal choices. Although computing the MSEEs consumes extra time, we do save time later, when classifying edge pixels using SMV and extracting the horizon line using dynamic programming.

The computation of MSEE Image reduces the number of edges considerably while not damaging important edges (i.e., horizon edges). Figure 1 below shows a sample gray scale 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 maintains the edges belonging to the horizon line.

2.2 Ground Truth Labeling

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, we fill these horizon locations manually and save their pixel locations separately. Therefore, for each training image, there would be part of the horizon line which is in perfect alignment with MSEEs as well parts of the horizon line with no edge support or perfect alignment. In the figure below, we show one of the training images with the true horizon line (ground truth) superimposed on it. The portions of the horizon line for which there is edge support are shown in "red" while the ones without edge support are shown in "blue".

2.3 Feature Extraction and Classifier Training

We train an SVM classifier using WEKA[5]. Specifically, to train the classifier we use SIFT [4] descriptors computed at the MSEE pixel locations selected for training. The horizon MSEE locations are chosen every fourth pixel on the true horizon line; an equal number of non-horizon MSEE locations are chosen randomly for each training image. Figure 2 shows the locations of horizon and non-horizon MSEE pixels chosen for training for a given training image. Once the MSEE locations have been chosen, SIFT descriptors of size 128 are extracted from a window of size 16×16 around the chosen MSEE location. We use the MATLAB implementation of SIFT available from vlfeat[3]. Figure 3 shows a flowchart of the training phase of our algorithm.

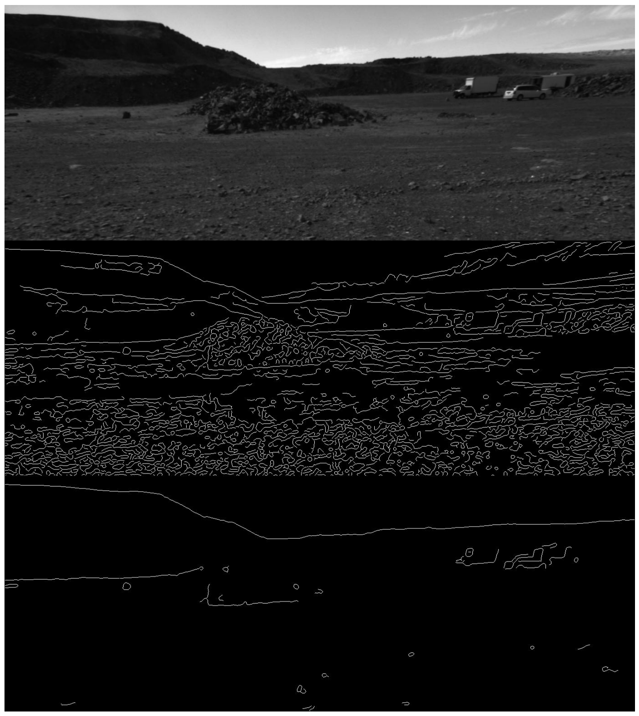

Fig. 1. Effect of MSEE: Gray scale sample image (top), Output of Canny edge detector (middle) and corresponding MSEEs (bottom).

3 Horizon Line Detection

3.1 Filtering MSEE Pixels

Given a novel image, we apply the SVM classifier to classify MSEE pixels into horizon and non-horizon pixels. MSEE pixels classified as non-horizon are discarded whereas MSEE pixels classified as horizon are post-processed using a dynamic programming based shortest path algorithm. The SVM classifier can be thought of as a mathematical binary function which returns 1 or -1 depending upon the feature vector around the MSEE pixel being classified. We refer to the binary image comprised of horizon classified edge locations as MSEE Positive and denote it as E_+. Each edge location (x, y) in E_+ indicates that an MSEE is present at that particular location which is classified as horizon. Mathematically,

$$E_+(x,y) = \begin{cases} 1, & \text{if } E(x,y) = 1\& \text{ Classifier}[E(x,y)] = 1. \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

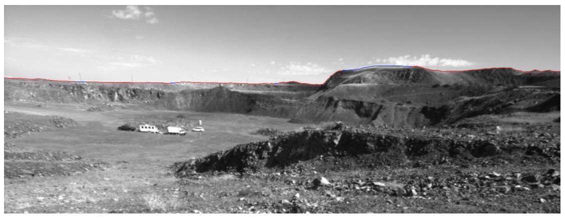

Fig. 2. Horizon line locations (ground truth) for a sample image. Red and blue segments emphasize the presence or abscence of MSEE edge support.

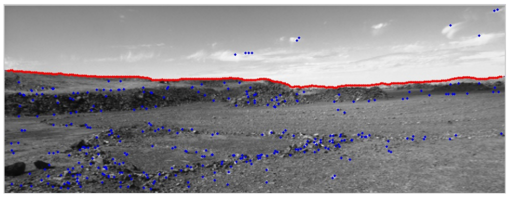

Fig. 3. MSEE locations for a training. Red and blue pixels correspond to the horizon and non-horizon MSEE locations.

3.2 Extracting Horizon Line Using Dynamic Programming

Lie et al. [9] have formulated the horizon line detection problem as a shortest path problem using a multi-stage graph. Their approach is straight forward and doesn't involve any preprocessing of the edge image. The edge image is directly fed to the dynamic programming algorithm which finds the shortest path between nodes S and T where S and T are dummy nodes added to the left of the leftmost column and the right of the rightmost column of the input binary image. In our implementation, we use the MSEE Positive image since it reduces the number of nodes per stage considerably by discarding non-horizon edges without affecting horizon edges. In particular, we show that the use of MSEE reduces the computational requirements of the dynamic programming algorithm while achieving similar or better accuracy. In our implementation, we use the same parameter values as in [9] (i.e., $\delta = 3$ and tolerance of gap (tog) $= 30$) where δ specifies the number of pixels to be searched in the next stage j for the current node in stage i. So, if kth node in stage i is under consideration; $\delta = 3$ pixel locations i.e. $k-1$, k and $k+1$ are checked in the next stage j.

3.3 Resolving Ambiguousness Due to Edge Gaps

In their approach, Lie et al. [9] only find the horizon line by processing the edge image in a left-to-right fashion. In our experiments, however, we have found that the left-to-right path is not always optimal since it might include incorrect

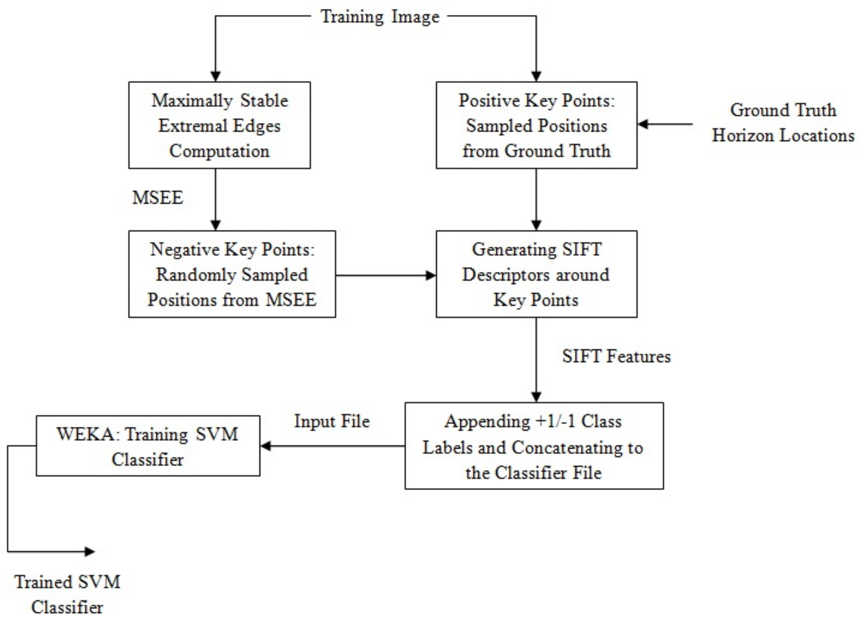
Training Image
Maximally Stable Extremal Edges Computation
Positive Key Points: Sampled Positions from Ground Truth
Ground Truth Horizon Locations
MSEE
Negative Key Points: Randomly Sampled Positions from MSEE
Generating SIFT Descriptors around Key Points
SIFT Features
Input File
WEKA: Training SVM Classifier
Appending +1/-1 Class Labels and Concatenating to the Classifier File
Trained SVM Classifier

Fig. 4. Flowchart diagram of the training phase of our approach

segments due to the presence of edge gaps. Here, we compute both the left-to-right and right-to-left shortest path solutions. Then, we find all horizon segments which are different in the two solutions (i.e., they do not overlap). To resolve these ambiguous horizon segments, we compute classification scores at each location based on the actual response of the classifier. The segment having the maximum compound score (i.e., product of classification scores) is selected to be included in the final solution.

Figure 4 below shows the flowchart of the testing phase of our approach. The dynamic programming component can be used to find a single solution)left-to-right) based on the method of Lie et al. [9] or it can be use to find both solutions (left-to-right and right-to-left) based on our approach. In the first case, the last step in the flowchart would be skipped. In the second case, however, further processing is required by using the classifier to calculate compound scores for the segments in the two solutions that do not overlap.

4 Experiments and Results

4.1 Data Set

Our data set is comprised of 10 gray scale images which have considerable scene, brightness and texture variations. The resolution of the images is 519 x 1388.

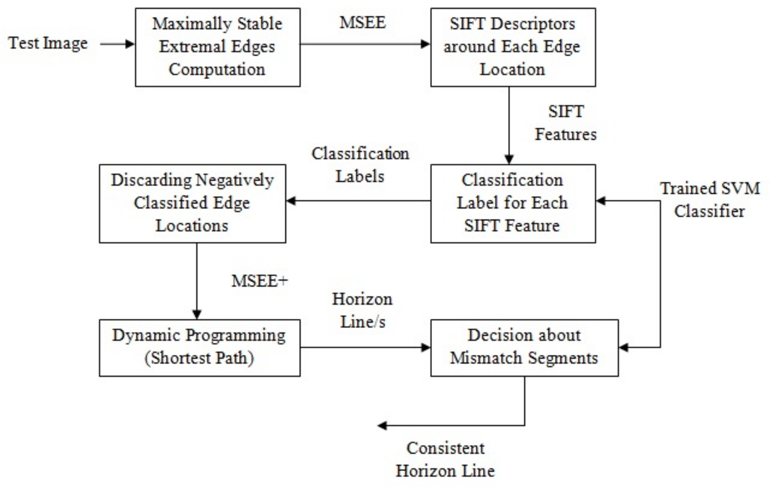
Test Image
Maximally Stable Extremal Edges Computation
MSEE
SIFT Descriptors around Each Edge Location
SIFT Features
Classification Label for Each SIFT Feature
Trained SVM Classifier
Classification Labels
Discarding Negatively Classified Edge Locations
MSEE+
Dynamic Programming (Shortest Path)
Horizon Line/s
Decision about Mismatch Segments
Consistent Horizon Line

Fig. 5. Flow Diagram of the Testing Phase and Dynamic Programming of Our Approach

We divide the data set in two equal subsets of 5 images each; we refer to them as $set1$ and $set2$. When using $set1$ for training, $set2$ is used for testing and vice versa. During training, we choose horizon and non-horizon examples from each image in the training set as mentioned in Section 2. During testing, we apply the classifier at each MSEE location in the test set.

4.2 Effect of MSEE on Horizon and Non-horizon Edges

Using MSEEs reduces the number of edges considerably which helps both the classification and dynamic programming steps. We have observed up to 92% reduction of non-horizon edges without affecting horizon edges. For each gray scale image in our data set, we apply the Canny edge detector implementation with $\sigma = 2$; the high/low threshold values are chosen by Matlab automatically. We compare the number of edges obtained using the above parameters with the edges sustaining the variation of lower/higher threshold while keeping σ fixed to 2. In our experiments, an edge is considered a MSEE if it survives over 3 different pairs of Th/Tl (i.e., stable edge). We have found that edges belonging to the horizon line are stable and that MSEE does not affect horizon edges while it reduces non-horizon edges remarkably.

Table 1 shows the number of edges detected using the MATLAB parameter (column2) versus MSEE (column3) for $\sigma = 2$. The percentage reduction in the number of non-horizon and horizon edges is shown in columns 4 and 5 of the table.

Table 1. % Reduction in number of Horizon and non-Horizon Edges due to MSEE

Image	Number of Edges (MATLAB)	Number of Edges (MSEE)	% Reduction Non-Horizon	% Reduction Horizon
1	38969	27428	29.1658	0
2	46877	31030	33.8055	0
3	40092	13873	65.3971	0
4	36909	11963	67.5879	0
5	43297	3289	92.4036	0
6	59088	16265	59.9088	0
7	47323	20313	57.0758	0
8	43810	26635	39.2034	0
9	48418	17558	63.7366	0
10	42333	12200	71.1809	0

4.3 (DP + Canny) versus (DP + MSEE)

In this section, we demonstrate the impact using MSEEs instead of all edges returned by the Canny edge detector. In this context, we compare the time and accuracy applying the dynamic programming approach using all Canny edges (i.e., DP + Canny) versus the MSEEs (i.e., DP + MSEE). To compare the accuracy of each approach, we compute the percentage of horizon line edges detected out of all (i.e., ground truth) horizon line edges. Since, ground truth horizon line edges are of two types as mentioned earlier (i.e., segments with edge support, shown in "red", and segments without edge support, shown in "blue"; see Figure 2) we distinguish between errors in these two types. Table 2 shows the comparison between (DP + Canny) and (DP + MSEE). Columns 2 and 3 show the running time of each approach in MATLAB. The total error percentage (%TErr) is the sum of the "red" error percentage (%RErr) and the "blue" error percentage (%BErr).

As it is evident from our results, (DP+MSEE) generally takes longer time than (DP+Canny). In terms of accuracy, (DP+MSEE) outperforms (DP+Canny) for challenging images such as images 7 and 10. The reason that (DP+MSEE) performs better is due to the fact that there are less non-horizon edges which allows the dynamic programming approach to find the correct path with higher probability. In the next section, we show that using the classifier to further reduce the number of non-horizon edges yields even higher accuracy.

4.4 (DP + MSEE$_+$)

As described in Section 3, for each test image first the MSEEs are computed. Then, the SIFT descriptors are calculated around each MSEE location and the SVM classifier is applied to classify MSEE edges as horizon or non-horizon. MSEE$_+$ contains only those MSEEs which were classified as horizon. We apply the dynamic programming approach using the MSEEs$_+$ and compute the shortest paths both from left-to-right and right-to-left. This is in contrast to the

Table 2. Comparing (DP+Canny) and (DP+MSEE) in terms of time and accuracy.

Image	**Time(sec)**		**Accuracy**					
			(DP+Canny)			**(DP+MSEE)**		
	(DP+Canny)	**(DP+MSEE)**	%TErr	%RErr	%BErr	%TErr	%RErr	%BErr
1	28.27	37.70	72.50	56.09	16.41	46.17	29.76	16.41
2	39.49	40.37	8.39	2.99	5.40	8.39	2.99	5.40
3	14.35	41.89	3.04	1.45	1.59	3.04	1.45	1.59
4	21.07	29.23	0.87	0.22	0.65	0.87	0.22	0.65
5	30.76	29.01	0.07	0.07	0	0.07	0.07	0
6	26.16	37.68	35.83	22.21	13.62	35.83	22.21	13.62
7	21.78	22.96	64.74	57.51	7.22	37.79	30.56	7.22
8	15.74	40.11	11.63	11.41	0.22	10.48	10.26	0.22
9	35.30	39.28	27.41	24.14	3.26	26.47	23.20	3.26
10	35.31	43.87	59.06	52.97	6.09	51.16	45.07	6.09

approach of Lie et al. [9] where the shortest path is computed from left-to-right using the Canny edges. Table 3 shows the horizon line detection errors in this case. For clarity, we report errors both for the left-to-right and the right-to-left solutions.

Table 3. Acquired Accuracy by Proposed Approach (DP + MSEE$_+$ + Compound Classifier Scoring)

Image	**left-to-right Path**			**right-to-left Path**			**Optimal Horizon Line**		
	%TErr	%RErr	%BErr	%TErr	%RErr	%BErr	%TErr	%RErr	%BErr
1	23.27	8.17	15.10	16.05	0.9482	15.10	16.05	0.9482	15.10
2	5.62	0.22	5.40	5.62	0.15	5.47	5.62	0.15	5.47
3	2.97	1.52	1.45	2.83	1.38	1.45	2.83	1.38	1.45
4	1.82	1.17	0.66	1.90	1.24	0.66	1.82	1.17	0.66
5	0	0	0	0	0	0	0	0	0
6	31.61	18.13	13.47	15.95	2.48	13.47	15.22	1.75	13.47
7	15.17	7.95	7.23	13.66	6.65	7.01	10.91	3.68	7.22
8	5.17	4.95	0.22	4.81	4.59	0.22	4.81	4.59	0.22
9	4.21	1.96	2.25	4.13	1.88	2.25	4.13	1.88	2.25
10	13.41	7.46	5.94	7.75	1.81	5.94	7.75	1.81	5.94

Comparing Tables 2 and 3, it is evident that the proposed approach outperforms both (DP+Canny) and (DP+MSEE).

4.5 Dealing with Ambiguous Segments

Using the left-to-right and right-to-left solutions, we identify those segments which do not overlap in the two solutions. To decide which of the two solutions to use for these segments, we use the actual response of the classifier to compute a compound score (i.e., product of classifier responses) for each of these segments. The score is normalized by the length of the pixels (edges) in the segment. Then, we choose the segment with the highest score. The last column of Table 3 shows the errors for the optimal solution where ambiguous segments are resolved based on the segment with the highest compound score. Figure 6 shows the optimal detected horizon lines imposed on the ground truth horizon lines for few images of our data set. The ground truth is shown in red and blue whereas deselected horizon is shown in green. Blue or red segments show the locations when proposed method have missed the ground truth horizon and so green color not hiding the red and blue.

As shown in Table 3, the total error percentage is lower for optimal horizon line, particularly for more challenging images such as images 6 and 7. Figure 7 shows several examples of ambiguous segments for images 1, 6 and 10. The ground truth is shown in red/blue and the solution found is shown in green. When the solution found perfectly overlaps with the ground truth, the red/blue colors are covered by green. It should be noted that segments belonging to right-to-left solution tend to have higher compound scores for our dataset. Extending the data set would surely produce more interesting cases.

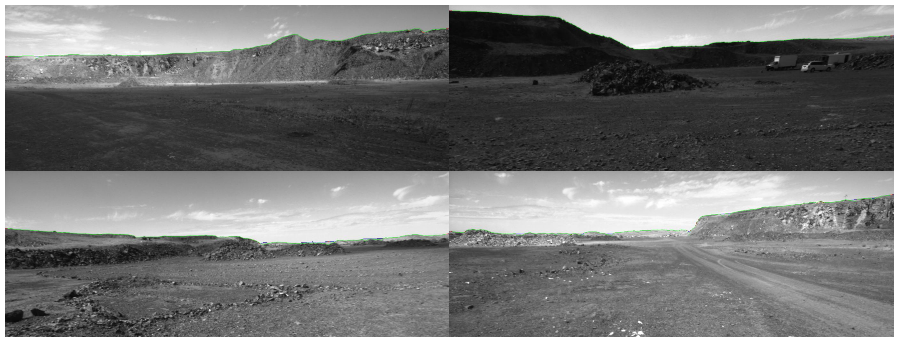

Fig. 6. Optimal Detected Horizon Lines Superimposed on Ground Truth Horizon Lines

5 Conclusion

We have presented a machine learning based horizon line detection algorithm using SIFT features. During training, we train an SVM classifier to classify MSEE pixels into two classes: horizon and non-horizon. During testing, MSEEs are detected and the SVM classifier is applied to identify those MSEEs that belong to the horizon line. Then, a dynamic programming algorithm is applied to find the horizon line. To deal with gaps, we apply the dynamic programming algorithm both in a left-to-right and right-to-left fashion. Segments which are

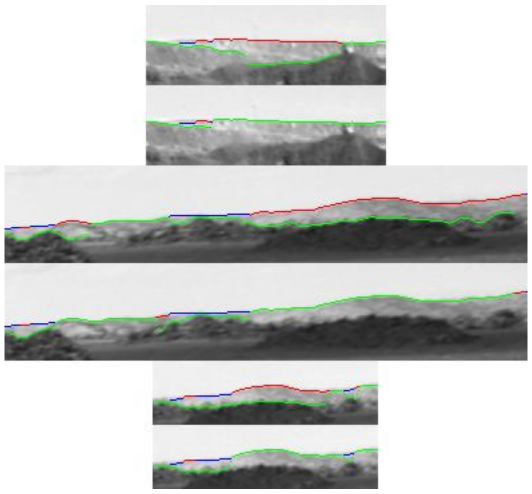

Fig. 7. Ambiguous segments in Left-to-Right (top) and Right-to-Left (bottom) paths computed for Images 1 (top 2 rows), 6 (middle 2 rows) and 10 (bottom 2 rows). Highlighting the alignment between ground truths, detected horizons and mismatch between left-to-right and right-to-left paths.

different in the two solutions are rectified by computing a compound score based on the actual responses of the classifier. For future work, we plan to investigate different local features such as WLD[15] and LBP[16] as well as single class classifiers such as Support Vector Data Description (SVDD)[17].

Acknowledgement

This material is based upon work supported by NASA EPSCoR under Cooperative Agreement No. NNX10AR89A.

References

1. Fabio Cozman and Carlos E. Guestrin: Automatic Mountain Detection and Pose Estimation for Teleoperation of Lunar Rovers. *ICRA*. 1997.
2. Sergiy Fefilatyev , Volha Smarodzinava, Lawrence O. Hall and Dmitry B. Goldgof: Horizon Detection Using Machine Learning Techniques. *ICMLA.*, 17-21, 2006.
3. http://www.vlfeat.org/index.html
4. D. G. Lowe: Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision.*, 68(2):91 - 110, 2004.
5. http://www.cs.waikato.ac.nz/ml/weka/
6. Stepan Obdrzalek and Jiri Matas: Object Recognition Using Local Afne Frames on Maximally Stable Extremal Regions *Toward Category-Level Object Recognition* LNCS 4170, 83-104, 2006.

7. 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.
8. Georges Baatz, Olivier Saurer, Kevin Koser, and Marc Pollefeys: Large Scale Visual Geo-Localization of Images in Mountainous Terrain *ECCV*, 2012.
9. Wen-Nung Lie, Tom C.-I. Lin , Ting-Chih Lin , and Keng-Shen Hung: A robust dynamic programming algorithm to extract skyline in images for navigation *Pattern Recognition Letters*, 26:221 - 230, 2005.
10. Timothy G. McGee, Raja Sengupta, and Karl Hedrick: Obstacle Detection for Small Autonomous Aircraft Using Sky Segmentation. In *International Conference on Robotics and Automation (ICRA'05)*, 2005.
11. Scott M. Ettinger, Michael C. Nechyba, Peter G. Ifju and Martin Waszak: Vision-Guided Flight Stability and Control for Micro Air Vehicles In *IEEE Int. Conf. on Intelligent Robots and Systems*, 2002.
12. G.C.H.E. de Croon, B.D.W. Remes, C. De Wagter, and R. Ruijsink: Sky Segmentation Approach to Obstacle Avoidance. In *IEEE Aerospace Conference*, 2011.
13. Sinisa Todorovic, Michael C. Nechyba and Peter G. Ifju: Sky/Ground Modeling for Autonomous MAV Flight In *International Conference on Robotics and Automation (ICRA'03)*, 2003.
14. J. Matas, O. Chum, M. Urban, and T. Pajdla: Robust wide baseline stereo from maximally stable extremal regions In *Proc. of British Machine Vision Conference*, pages 384-396, 2002.
15. Jie Chen, Shiguang Shan, Chu He, Guoying Zhao, Matti Pietikinen, Xilin Chen, and Wen Gao WLD: A Robust Local Image Descriptor In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.
16. T. Ojala, M. Pietikinen and D. Harwood: A Comparative Study of Texture Measures with Classification Based on Feature Distributions In *Pattern Recognition*, vol. 29, no. 1, pages 51-59, 1996.
17. David M.J. Tax and Robert P.W. Duin : Support Vector Data Description. In *Machine Learning*, vol. 54, no 1, pages 45 - 66, 2004.