An Edge-Less Approach to Horizon Line Detection

Touqeer Ahmad*, George Bebis*, Monica Nicolescu*, Ara Nefian†, Terry Fong†

*Department of Computer Science and Engineering
University of Nevada, Reno
{ahmad, bebis, monica}@cse.unr.edu
†NASA Ames Research Center
{ara.nefian, terry.fong}@nasa.gov

***Abstract*—Horizon line is a promising visual cue which can be exploited for robot localization or visual geo-localization. Prominent approaches to horizon line detection rely on edge detection as a pre-processing step which is inherently a non-stable approach due to parameter choices and underlying assumptions. We present a novel horizon line detection approach which uses machine learning and Dynamic Programming (DP) 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 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. Using classification maps instead of edge maps allows for removing certain assumptions 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 bias the DP solution but they fail to produce good results when they are not valid. We demonstrate our approach on three different data sets and provide comparisons with a traditional approach based on edge maps. Although our training set is comprised of a very small number of images from the same location, our results illustrate that our method generalizes well to images acquired under different conditions and geographical locations.**

I. Introduction

Horizon 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. This has many applications including smooth navigation of small unmanned aerial vehicles (UAVs) [7], [22], [27] and micro air vehicles (MAVs) [9], [10], [28], augmented reality [25], visual geo-localization and annotation of mountain/desert imagery [5], [6], [19], [20], [29], port security and ship detection [11], [12], outdoor robot/vehicle localization [13], [14], [26] and autonomous vehicle navigation [24]. Previous attempts to horizon line detection can be categorized into two major groups; (i) methods modeling sky and non-sky regions using machine learning [7], [9], [10], [11], [22], [28] and (ii) methods relying on edge detection [16], [18]. Recently, some attempts [2], [15], [25] have been made to combine these two approaches by eliminating non-horizon edges using classification; however, these attempts also fall under the second category as horizon detection is effectively based on edges. It should be mentioned that earlier methods to horizon detection suffer from the assumption that the horizon boundary is linear and hence are limited.

McGee et al. [22] proposed a sky segmentation approach to find a linear boundary between sky and non-sky regions using an SVM classifier trained only on color information. They use sky segmentation as an obstacle detection tool for small scale UAVs. Their underlying assumption of the horizon boundary being linear is violated very often and probably is acceptable only for UAVs navigation. Ettinger et al. [10] proposed a flight stability and control system for micro air vehicles (MAVs) which relies on horizon detection. They model the sky and non-sky regions by Gaussian distributions and try to find an optimum boundary segmenting them. Their model is based on two assumptions: first, the horizon boundary is linear and second, the horizon line separates the image into two regions of significantly different appearance (i.e., sky and non-sky). However, these assumptions might not always be true. The approach of Fefilatyev et al. [11] is also based on the horizon boundary being linear and uses color and texture features such as mean intensity, entropy, smoothness, uniformity etc. to train various classifiers. Croon et al. [9] extended the features used in[11] by including cornerness, grayness and Fisher discriminant features to train a shallow decision tree classifier. Their approach has been tested in the context of MAVs obstacle avoidance and is able to detect non-linear horizon boundaries. In [19] Liu et al. have proposed a sensor fusion approach to estimate horizon using textured Digital Elevation Model (DEM), airport model, GPS, AHRS and vision sensors. Their objective was to estimate an accurate linear horizon boundary from an aircraft in low visibility conditions and the approach does not generalize to non-linear horizons.

Todorovic et al. [28] circumvent the assumption of the horizon boundary being linear in [10] 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 between sky and non-sky regions. Using a Hidden Markov Tree model they built a robust horizon detection approach capable of detecting non-linear horizon boundaries. [30] proposed a fusion-based approach where they combine the outputs of a Neural Network (NN) classifier with K-means clustering. They use the mean intensity and texture features, similar to [9], [11], to train the NN classifier. Although their approach demonstrates reasonable results, their system is based on various heuristics and parameter settings that might not generalize well to different data sets. In [7], a clustering based horizon line detection approach for robust UAV navigation was presented. The main assumption is the presence of a dominant light

field between sky and ground regions which they detect using intensity information and K-means clustering. In general, the assumption about the light field does not hold and they have identified cases where their method requires modifications for the clustering process to produce good results. Thurrowgood et al. [27] used horizon detection for UAV attitude estimation. By learning a transformation from the RGB space and a single dimensional space, an optimum threshold can be found to segment sky/non-sky regions based on histograms and priors about sky and non-sky regions. Their approach is limited only to UAV navigation due to the assumption that sky and ground pixels are equi-probable.

The most prominent method belonging to the second category is that of Lie et al. [18] where horizon detection is formulated as a graph search problem. Their approach relies on edge detection and assumes a consistent edge boundary between sky and non-sky regions. The detected edge map is represented as a multi-stage graph where each column of the image becomes a stage of the graph and each edge pixel becomes a node. The shortest path is then found extending from the left-most column to the right-most column using DP. The assumption that the horizon boundary is a consistent edge boundary is rarely true in practice due to environmental conditions (e.g., clouds) and edge gaps. To address the issue of gaps, [18] have proposed a gap-filling approach which highly depends on the choice of certain parameters. Moreover, they assume that the horizon line is present in the upper half of the image which biases the shortest path solution to be in that region.

Recently, Ahmad et al. [2] and Hung et al. [15] independently proposed extensions to the approach of Lie et al. [18] by introducing a classification step to eliminate non-horizon edges. The graph is then built using edge pixels classified as horizon pixels. This is performed by training a classifier using features from key-points taken from horizon and non-horizon locations. Ahmad et al. [2] used SIFT features [21] around the key-points and an SVM classifier whereas Hung et al. [15] used an SVM classifier with color information as well as the variance above and below a given training point. In addition to training a classifier, Ahmad et al. [2] reduce the number of edges considerably by applying a series of thresholds for Canny edge detector and keeping only those edges for further processing which are strong enough to survive for various thresholds. Somewhat relevant is the approach by Porzi et al. [25]. Similar to Ahmad et al. [2] they first reduce the number of edges using a threshold for the Sobel detector and then use trained Random Ferns to further classify remaining edge pixels into contour and non-contour edges. Ahmad et al. [2] extended their work in [3], [1], [4] where they investigated various textural features and nodal costs respectively for training the SVM classifier and Dynamic Programming.

In this paper, we extend the approach of Lie et al. [18] by extracting the horizon line not from an edge map, which typically contains many gaps and provides no confidence about the likelihood of an edge point belonging to the horizon line, but from a classification map. The key idea is classifying each pixel in an image (or a region of interest) as belonging to the horizon line or not; we refer to the classification map as Dense Classifier Score Image (DCSI). This is performed by training a classifier using both horizon and non-horizon pixels using a small set of training images. The resulting classification map contains no gaps but also each pixel is associated with a classification score which provides information about the likelihood of the pixel belonging to the horizon line. Representing the classification map as a multi-stage graph, the horizon line can be extracted by finding the path that maximizes the sum of classification scores using DP. We have experimented with Support Vector Machines(SVM) [8] and Convolutional Neural Networks(CNNs) [17] and normalized pixel intensities for features. Using classification maps instead of edge maps also allows us to remove certain assumptions such as that the horizon line is close to the top of the image or that the horizon is a straight line. The purpose of these assumptions is to bias the DP solution but fail to produce good results when they are violated which is commonly the case. We demonstrate our approach on three different data sets and provide comparisons with a traditional approach using edge maps.

It should be noted that we have intentionally chosen to compare the proposed approach against Lie et al. [18] instead of recent extensions such as Ahmad et al. [2] and Hung et al. [15]. This is because these approaches employ machine learning to reduce the number of non-horizon edges which yields a smaller graph and speeds-up Dynamic Programming. Since our emphasis here is on dealing with edge gaps rather than speeding-up computations, we have decided to use Lie et al. [18] for comparison purposes as it is more straightforward to implement. Also these approaches [2], [15] can result into increased number of gaps due to misclassification.

The rest of paper is organized as follows. In section 2, we review the approach of Lie et al. [18]. Section 3 describes the steps of our proposed approach. Section 4 presents our data sets, experimental results. Finally, section 5 provides our conclusions and directions of future research.

II. Background

In this section, we briefly review the method of Lie et al. [18] and point out its underlying assumptions. Lie et al. [18] formulate the problem of horizon line detection as a multi-stage graph problem and use DP to find the shortest path extending from left to right. Their approach is based on the assumptions that a full horizon line exists in the image from left to right and that it lies in the upper half of the image. Given an image of size $M \times N$, edge detection is performed first to compute a binary edge map I where 1 implies the presence of an edge pixel and 0 a non-edge pixel. This edge map is used to build an $M \times N$ multi-stage graph $G(V, E, \Psi, \Phi)$ where each pixel in the edge map corresponds to a graph vertex; a low cost l is associated with edge pixels while a very high cost (i.e., ∞) is associated with non-edge pixels as shown below:

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

$\Psi(i,j)$ is the cost associated with vertex i in stage j (i.e., v_{ij}). The graph can be visualized as an N (columns) stage graph where each stage contains M nodes (rows). To deal with edge gaps, they propose a gap filling approach. Given a node i in stage j, its neighborhood in the next stage $j+1$ is defined by a δ parameter, that is, the number of nodes to

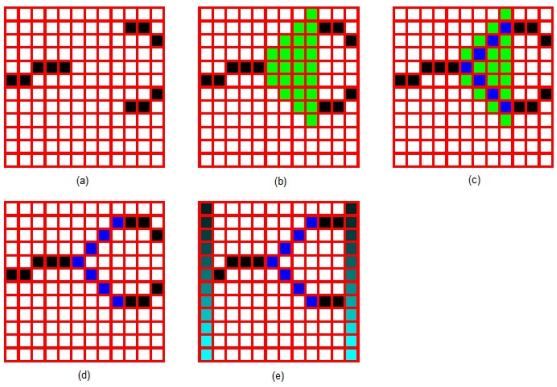

Fig. 1. Steps of Horizon Detection by Lie et al.

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 distance from it as shown in the equation below.

$$\Phi(i,k,j) = \begin{cases} |i-k|, & \text{if } I(i,j) = I(k,j+1) = 1 \\ & \text{and } |i-k| \leq \delta \\ \infty, & \text{otherwise.} \end{cases} \quad (2)$$

If a node i in stage j cannot be connected to any node in stage $j+1$ with in δ neighborhood, a search window is defined using 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 between node i in stage j and node k found within the search window j+tog. A high cost is associated with the dummy nodes introduced by the gap filling step.

Once the gaps are filled with high cost dummy nodes, the cost of the nodes in stage 1 and N is increased based on the vertical position of the nodes according to the equation below:

$$\Psi(i,j) = \begin{cases} (i+1)^2, & \text{if } j = 1 \text{ or } j = N \\ \Psi(i,j), & \text{otherwise.} \end{cases} \quad (3)$$

This enforces the assumption that the horizon line is present in the upper half of the image and hence biasing the DP solution 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 each one of them. The s node is connected with all the nodes in stage 1 and all the nodes in stage N are connected to node t. A shortest path is then found extending from node s to t using DP which conforms to the detected horizon boundary.

Figure 1 illustrates the steps of Lie et al. [18] for a sample image. An edge map is shown in Figure 1-(a) where black and white rectangles represent edge and non-edge pixels. A search window is shown in Figure 1-(b) for the edge node in stage $j = 5$ using $\delta = 1$ and $tog = 4$. Within the search window j+tog, two edge nodes are discovered which are then connected to node j by introducing dummy nodes as shown in Figure 1-(c,d) (highlighted in blue). The nodes in stage 1 and N are set to a higher cost associated with their vertical position; this is reflected by an increasing intensity in Figure 1-(e). Two nodes s and t are then introduced, as described above, and DP is applied on this graph. As it is clear from Figure 1-(5), there exist two equal paths in the above sample graph (ignoring source (s) and sink (t) nodes); however, DP will select the upper path due to the assumption of the horizon line being present in the upper half. However, it might be possible that the true horizon line is actually the lower one and that the upper edge segment was only due to some clouds. Also, if the gap happen to be bigger than the chosen parameter (tog) values, DP could miss parts of the actual horizon.

We provide specific examples in the experimental results section, illustrating how this method fails to detect a portion of horizon line due to this assumption.

III. Proposed Approach

The proposed approach does not rely on the assumptions of [18] or the assumption that the horizon curve is linear; moreover, the horizon line could be anywhere in the image. Our only assumption is that the horizon line extends from left to right. Specifically, our 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. Moreover, we do not force the nodes in stages 1 and N to be associated with their vertical position 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 2 illustrates the main steps of the proposed approach.

A. Pixel Classification

For classification, we have experimented with two classifiers: SVM[8] and a CNN [17]. 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). 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. 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 have experimented with different features in the past (e.g., SIFT features similar to [2]), however, normalized pixel intensities seem to work better and are faster to compute. 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

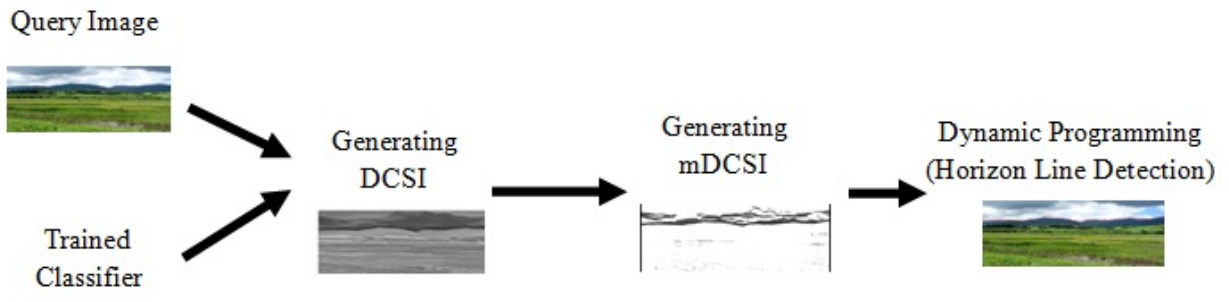

Fig. 2. Main steps of the proposed horizon line detection approach.

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 shows an example of horizon (red) and non-horizon (blue) training samples.

B. 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 shows the DCSI for a sample query image.

$$D(x, y) = \Gamma(V(x, y)) \tag{4}$$

C. 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 vertices; as a result, fewer paths need to be considered when searching for the shortest path which

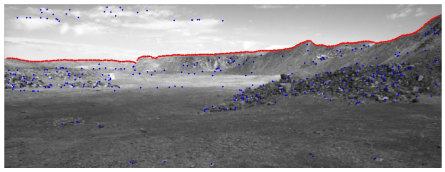

Fig. 3. Locations of horizon (red) and non-horizon (blue) training samples.

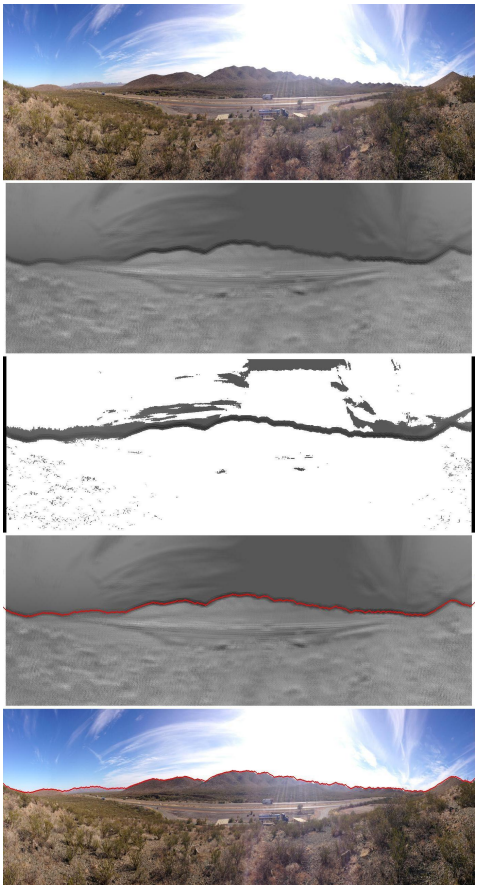

Fig. 4. Sample test image (row1), DCSI (row2), mDCSI (row3), shortest path solution (row4, highlighted in red), and detected horizon line (row5, highlighted in red).

results in considerable speedups. In our experiments, we have found that by keeping the highest 50 classification scores yields accurate horizon line detections. From an implementation point of view, the mDCSI is computed as follows:

$$mD(x, y) = \begin{cases} D(x, y), & \text{if } x \in L(m)_y \\ l, & \text{otherwise; } l < 0. \end{cases} \tag{5}$$

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 shows example of the respective mDCSI.

D. Horizon Line Detection as a Shortest Path Problem

In earlier approaches eg [2], [15], [18], 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 our 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).

$$\Psi(i, j) = mD(x, y) \tag{6}$$

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. This is in contrast to conventional approaches where the absolute difference between the positions of nodes in two stages is used as an edge cost.

$$\Phi(i, k, j) = \begin{cases} 0, & \text{if } |i - k| \leq \delta \\ \infty, & \text{otherwise.} \end{cases} \tag{7}$$

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 positions. 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 Dijkstra's algorithm.

IV. Experimental Results and Comparisons

A. Data Sets and Quantitative Evaluation

To evaluate the performance of the proposed approach, 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 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 consists of only 9 images from Basalt Hills data set. The resolution of all images in our data sets is 519×1388.

To quantitatively evaluate the performance of the proposed approach, we have manually extracted the horizon line (ground truth) in all the images of our data sets. To evaluate the proposed approach, 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:

$$S = \frac{1}{N} \sum_{j=1}^{N} |P_{d(j)} - P_{g(j)}| \tag{8}$$

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. Figure 5 shows some representative results of our horizon detection approach using images from our data sets. Table 1 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. [18]. It is interesting to note that although our method was trained using a very small number of images from the same data set, it generalizes very well to images from other data sets, such as the Web data set which is very different from the training data set. Next, we provide a detailed analysis of our results.

Our experimental results illustrate that the proposed approach outperforms the traditional approach of [18] based on edge maps. In particular, both the average error and standard deviation of the traditional approach are much higher than the proposed approach based on SVM or CNN classifiers. 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 6 (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 was able to find the true horizon line with high accuracy in both cases (row 2). Figure 7 shows zoomed sub-images of the left column images of 6 (i.e., Lie et al.) for better visualization.

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 8 shows some

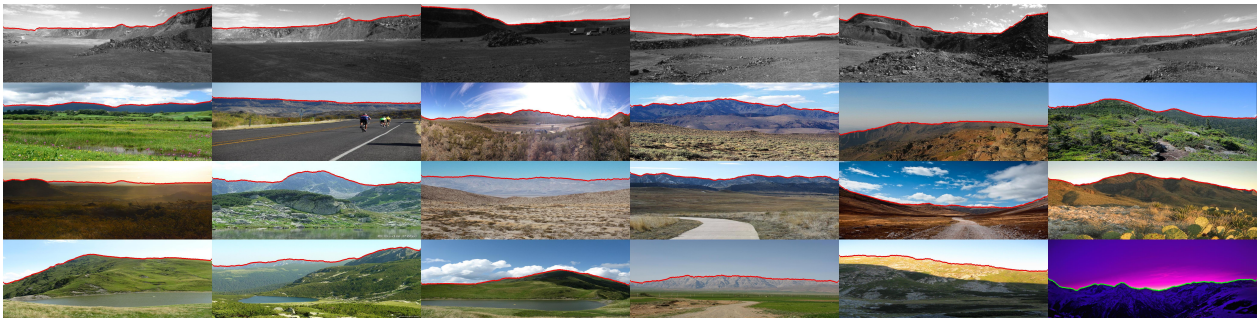

Fig. 5. Sample results illustrating our horizon line detection approach: Basalt Hills data set (row1) and Web data set (row 2 through 4). Detected horizon lines are highlighted in red/green.

TABLE I. AVERAGE ABSOLUTE ERRORS

Data Set	Proposed Approach				Lie et al. [18]	
	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

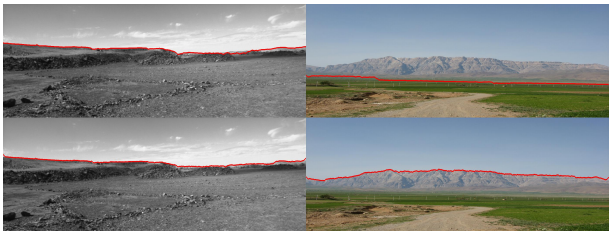

Fig. 6. Examples illustrating: [row 1] missing the horizon line or parts of it due to edge gaps (Lie et al.), and [row 2] detecting the true horizon line using our approach (SVM).

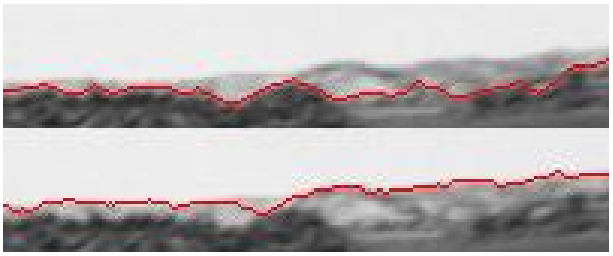

Fig. 7. Zoomed sub-images of the left column images of Figure 6

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.

Comparing the two classifiers used in our experiments, 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 9 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

Fig. 8. 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 (SVM).

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.

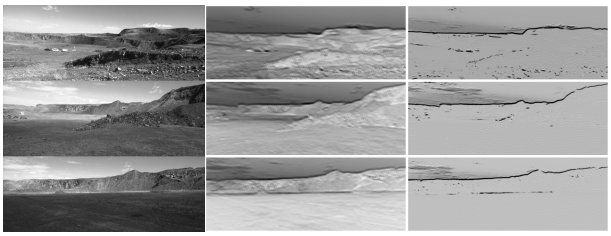

Fig. 9. Test images (column1), corresponding SVM-DCSIs (column2) and corresponding CNN-DCSIs (column3).

Attempting to better understand why the proposed approach 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 10 (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 10 (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.

Fig. 10. [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 11 shows examples where our method has failed to find good solutions. This issue can be addressed by increasing the train set and making it more versatile. By carefully analyzing our results on the Web data set, our 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.

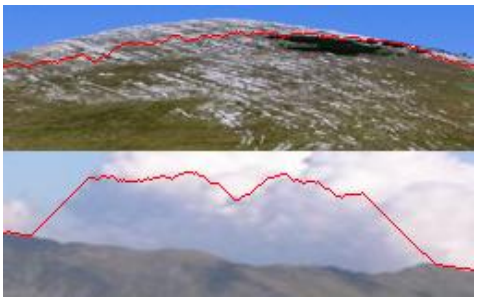

Fig. 11. Examples illustrating the inability of the proposed method to find a good solution due to the lack of sufficient training data.

V. Conclusion

We have proposed a novel horizon detection approach which 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. The proposed approach does not make any assumption about the horizon being a straight line or being close to the top of the image. The proposed approach uses a very small number of images to train the horizon classifiers and outperforms traditional approaches based on edge maps on three challenging data sets. For future work, we plan to perform extensive experiments using much larger training and test sets and different types of features. Moreover, we plan to combine classification scores with gradient magnitude information to improve the DP solutions. Our long term goal is to investigate the suitability of the proposed horizon detection method for pose estimation and localization of planetary rovers and UAVs.

Acknowledgements

This work is supported by NASA EPSCoR under Cooperative Agreement No. NNX11AM09A, and in part by NSF PFI.

References

[1] T. Ahmad, G. Bebis, M. Nicolescu, A. Nefian, and T. Fong. Fusion of edge-less and edge-based approaches for horizon line detection. In *Proceedings of 6th IEEE International Conference on Information, Intelligence, Systems and Applications (IISA)*, 2015.

[2] T. Ahmad, G. Bebis, E. Regentova, and A. Nefian. A machine learning approach to horizon line detection using local features. In *Proceedings of 9th International Symposium on Visual Computing*, pages 181–193, 2013.

[3] T. Ahmad, G. Bebis, E. Regentova, A. Nefian, and T. Fong. An experimental evaluation of different features and nodal costs for horizon line detection. In *Proceedings of 10th International Symposium on Visual Computing*, pages 193–205, 2014.

[4] T. Ahmad, G. Bebis, E. Regentova, A. Nefian, and T. Fong. Coupling dynamic programming with machine learning for horizon line detection. *International Journal on Artificial Intelligence Tools (IJAIT)*, 24(4), 2015.

[5] G. Baatz, O. Saurer, K. Koser, and M. Pollefeys. Large scale visual geo-localization of images in mountainous terrain. In *Proceedings of 12th European Conference on Computer Vision*, pages 517–530, 2012.

[6] L. Baboud, M. Cadik, E. Eisemann, and H.-P. Seidel. Automatic photo-to-terrain alignment for the annotation of mountain pictures. In *Computer Vision and Pattern Recognition (CVPR)*, 2011.

[7] N. S. Boroujeni, S. A. Etemad, and A. Whitehead. Robust horizon detection using segmentation for uav applications. In *IEEE 2012 Ninth Conference on Computer and Robot Vision*, pages 346–352, 2012.

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

[9] G. C. H. E. de Croon, B. D. W. Remes, C. D. Wagter, and R. Ruijsink. Sky segmentation approach to obstacle avoidance. In *IEEE Aerospace Conference*, pages 1–16, 2011.

[10] S. M. Ettinger, M. C. Nechyba, P. G. Ifju, and M. Waszak. Vision-guided flight stability and control for micro air vehicles. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, pages 2134–2140, 2002.

[11] S. Fefilatyev, V. Smarodzinava, L. O. Hall, and D. B. Goldgof. Horizon detection using machine learning techniques. In *Proceedings of 5th International Conference on Machine Learning and Applications*, pages 17–21, 2006.

[12] 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).

[13] V. Gupta and S. Brennan. Terrain-based vehicle orientation estimation combining vision and inertial measurements. *Journal of Field Robotics*, 25(3):181–202, 2008.

[14] N. Ho and P. Chakravarty. Localization on freeways using the horizon line signature. In *International Conference on Robotics and Automation*, 2014.

[15] Y. Hung, C. Su, Y. Chang, J.Charig, and H. Tyan. Skyline localization for mountain images. In *Proceedings of International Conference on Multimedia and Expo*, pages 1–6, 2013.

[16] B. Kim, J. Shin, H. Nam, and J. Kim. Skyline extraction using a multistage edge filtering. *World Academy of Science, Engineering and Technology*, 2011.

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

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

[19] C. Liu, Y. Zhang, K. Tan, and H. Yang. Sensor fusion method for horizon detection from an aircraft in low visibility conditions. *IEEE Transactions on Instrumentation and Measurement*, 63(3):620–627, 2014.

[20] W. Liu and C. Su. Automatic peak recognition for mountain images. In *Advanced Technologies, Embedded and Multimedia for Human-centric Computing*, pages 1115–1121, 2014.

[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[22] T. G. McGee, R. Sengupta, and K. Hedrick. Obstacle detection for small autonomous aircraft using sky segmentation. In *Proceedings of International Conference on Robotics and Automation*, pages 4679–4684, 2005.

[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. In *International Conference on Image Processing*, 2014.

[24] A. M. Neto, A. C. Victorino, I. Fantoni, and D. E. Zampieri. Robust horizon finding algorithm for real-time autonomous navigation based on monocular vision. In *International Conference on Intelligent Transportation Systems*, 2011.

[25] L. Porzi, S. R. Bulo, P. Valigi, O. Lanz, and E. Ricci. Learning contours for automatic annotations of mountains pictures on a smartphone. In *ACM/IEEE International Conference on Distributed Smart Cameras*, 2014.

[26] S. J. Steven and P. W. Gibbens. Efficient terrain-aided visual horizon based attitude estimation and localization. *Journal of Intelligent & Robotic Systems*, 2014.

[27] S. Thurrowgood, D. Soccol, R. J. D. Moore, D. Bland, and M. V. Srinivasan. A vision based system for attitude estimation of uavs. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, pages 5725–5730, 2009.

[28] S. Todorovic, M. C. Nechyba, and P. G. Ifju. Sky/ground modeling for autonomous mav flight. In *Proceedings of International Conference on Robotics and Automation*, pages 1422–1427, 2003.

[29] E. Tzeng, A. Zhai, M. Clements, R. Townshend, and A. Zakhor. User-driven geolocation of untagged desert imagery using digital elevation models. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2013.

[30] A. P. Yazdanpanah, E. E. Regentova, A. K. Mandava, T. Ahmad, and G. Bebis. Sky segmentation by fusing clustering with neural networks. In *Proceedings of 9th International Symposium on Visual Computing*, pages 663–672, 2013.