International Journal on Artificial Intelligence Tools

Coupling Dynamic Programming with Machine Learning for Horizon Line Detection

Touqeer Ahmad
Dept. of Computer Science and Engineering, University of Nevada
Reno, 89557, NV, USA
sh.touqeerahmad@gmail.com

George Bebis
Dept. of Computer Science and Engineering, University of Nevada
Reno, 89557, NV, USA
bebis@cse.unr.edu

Emma Regentova
Dept. of Electrical and Computer Engineering, University of Nevada
Las Vegas, 89154, NV, USA
emma.regentova@unlv.edu

Ara Nefian
NASA Ames Research Center
Moffett Field, 94035, CA, USA
ara.nefian@nasa.gov

Terry Fong
NASA Ames Research Center
Moffett Field, 94035, CA, USA
terry.fong@nasa.gov

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

In this paper, we consider the problem of segmenting an image into sky and non-sky regions, typically referred to as horizon line detection or skyline extraction. Specifically, we present a new approach to horizon line detection by coupling machine learning with dynamic programming. Given an image, the Canny edge detector is applied first and keeping only those edges which survive over a wide range of thresholds. We refer to the surviving edges as Maximally Stable Extremal Edges (MSEEs). The number of edges is further reduced by classifying MSEEs into horizon and non-horizon edges using a Support Vector Machine (SVM) classifier. Dynamic programming is then applied on the horizon classified edges to extract the horizon line. Various local texture features and their combinations have been investigated in training the horizon edge classifier including SIFT, LBP, HOG, SIFT-LBP, SIFT-HOG, LBP-HOG and SIFT-LBP-HOG. We have also investigated various nodal costs in the context of dynamic programming

including binary edge scores, normalized edge classification scores, gradient magnitude and their combinations. The proposed approach has been evaluated and compared with a competitive approach on two challenging data sets, illustrating superior performance.

Keywords: Horizon Line Detection;Skyline Extraction;Sky Segmentation;Dynamic Programming

1. Introduction

Horizon line detection or sky segmentation is the problem of finding a boundary between sky and non-sky regions (ground, water or mountains) in a gray scale or color image. This has many applications including 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] port security and ship detection[19,20] and outdoor robot/vehicle localization.[10,11,12] Previous attempts to horizon line detection can be categorized into two major groups; (i) methods modeling sky and non-sky regions using machine learning[4,5,6,7,9,19] and (ii) methods employing edge detection.[3,21] Recently, some attempts[1,2] have been made to combine these two approaches by eliminating non-horizon edges using classification; however, these attempts also fall under the second category since 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.[7] 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.[6] 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.[19] is also based on the horizon boundary being linear and uses color and texture features (e.g., mean intensity, entropy, smoothness, uniformity etc.) to train various classifiers. Croon et al.[5] extended the features used in[19] 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.

Todorovic et al.[9] circumvent the assumption of the horizon boundary being linear in Ref. 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 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. Yazdanpanah et al.[22] 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 Ref. 5 and 19 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 Ref. 4, 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.[8] used horizon detection for UAV altitude estimation. By learning a transformation from the RGB space to 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 applicable 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.[3] 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, Ref. [3] has 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.[1] and Hung et al.[2] independently proposed extensions to the approach of Lie et al.[3] 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.[1] used SIFT descriptors [14] around the key-points and an SVM classifier whereas Hung et al.[2] used an SVM classifier with color information as well as the variance above and below a given point.

2. Background

In this section, we briefly review the method of Lie et al.[3] and point out its underlying assumptions. Lie et al. 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:

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

$\Psi(i,j)$ 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 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.

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

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:

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

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

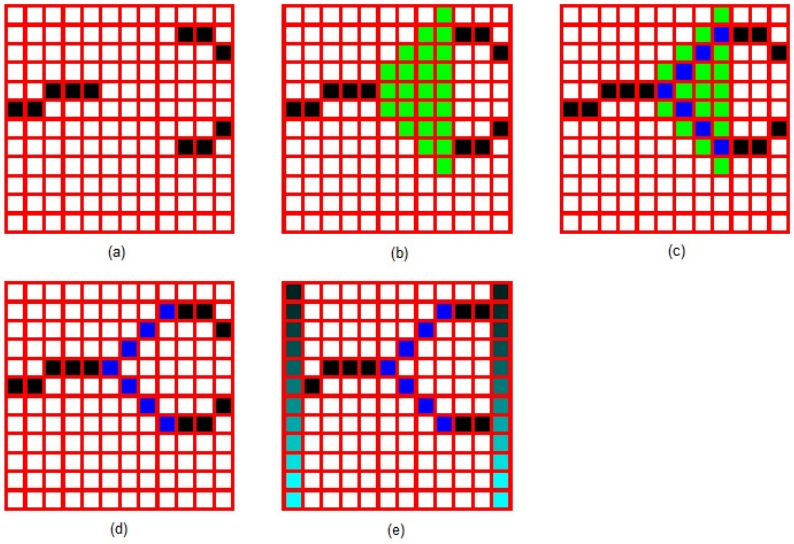

Fig. 1. Highlighting bias problem inherent to Lie et al. (a) edge (black) and non-edge (white) pixels, (b) a search window for node 5 in stage 5, (c) current node being connected to the nodes found in search window, (d) dummy nodes introduced with high costs (blue), (e) nodes in first and last stage being initialized with their vertical position (enforcing bias).

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. 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-(e), 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.

3. Classification of Horizon Edges

The complexity of the Lie et al.[3] approach depends on the number of edges detected in an image while its performance can be affected by nearby edges (e.g., due to

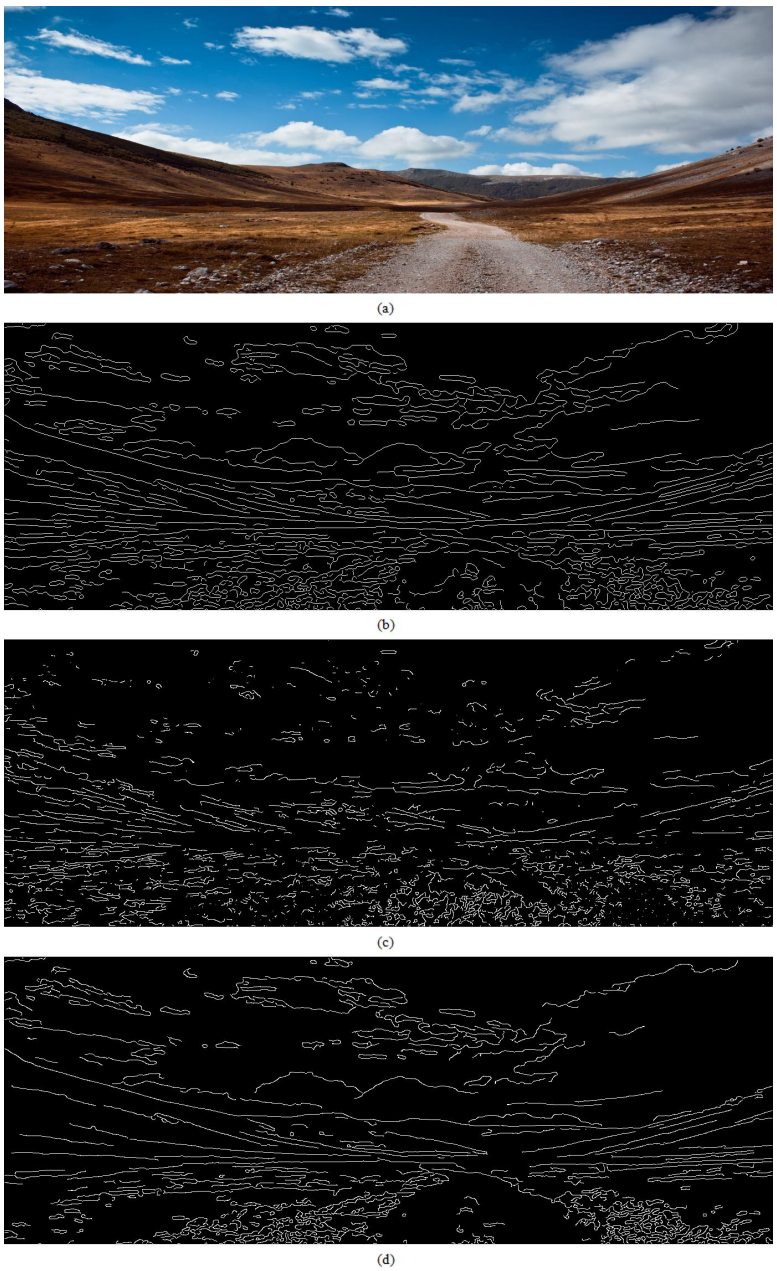

Fig. 2. Effect of MSEE: (a) input image, (b) output of the Canny, (c) discarded edges by MSEE and (d) survived edges i.e. MSEE image. Note the reduction in number of edges due to MSEE computation.

clouds) or edge gaps. We address these issues by detecting the most stable edges in an image (i.e., MSEEs), and applying classification using SVMs to eliminate non-horizon edges. To address the issue of edge gaps, we replace the binary edge map in the Lie et al.[3] approach with a continuous map based on gradient magnitude information or classification scores.

3.1. *Maximally Stable Extremal Edges (MSEEs)*

The idea of extracting MSEEs was inspired from the idea of extracting Maximally Stable Extremal Regions (MSER).[25] 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,

$$E(x,y) = \begin{cases} 1, & \text{if } \sum_{i=1}^{N} I(x,y)_i > k. \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

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. The computation of MSEE Image reduces the number of edges considerably while not damaging important edges (i.e., horizon edges). Figure 2 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.

3.2. *Learning to Classify Horizon Edges*

To further reduce the number of edges, an SVM classifier is applied on the MSEEs in order to eliminate non-horizon edges. To train the SVM classifier, we use both positive (i.e., horizon) and negative (i.e., non-horizon) examples by computing local features at horizon and non-horizon key point locations. To select the positive examples, we manually label the horizon line in each training image and choose a number of key points along it uniformly. Figure 3 shows a few images with ground truth information marked as red. The negative examples are selected randomly from non-horizon MSEE locations. Figure 4 shows an example of positive (red) and negative (blue) examples selected from a training image.

Fig. 3. Ground truth horizon lines; highlighted in red.

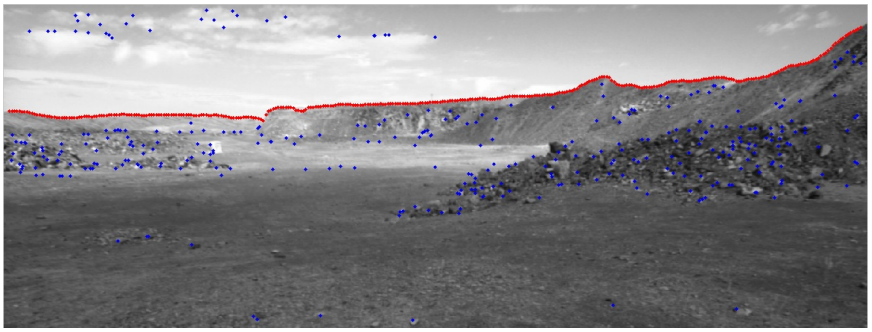

Fig. 4. Positive (red) and negative (blue) key point locations for a training image.

3.3. *Feature Extraction and Classifier Training*

To train an SVM classifier we take a 16×16 window around each +ive/-ive key point and compute a local descriptor. We have investigated three local descriptors: Scale Invariant Feature Transform(SIFT)[14], Local Binary Patterns(LBP)[13] and Histogram of Oriented Gradients(HOG).[15] We compute these descriptors using Vlfeat[16] which provides 128, 58 and 31 dimensional vectors for SIFT, LBP and HOG respectively. We have investigated the performance of individual descriptors as well as their combinations to train the SVM classifier. The combinations are formed by mere concatenation of the descriptor vectors for each training and testing instance. The feature combinations and their size are: SIFT-LBP (186 features), SIFT-HOG (159 features), LBP-HOG (89 features) and SIFT-LBP-HOG (217 features). We have found the SIFT-HOG combination as the best choice when compared with individual descriptors and other combinations as described in the results section. Figure 5 shows a flow diagram for the training phase of our proposed approach.

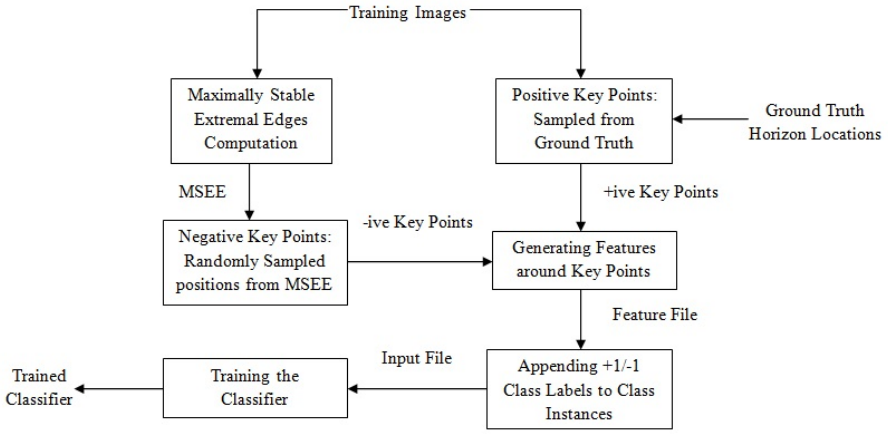

Fig. 5. Flow diagram for the training phase of the proposed approach.

4. Horizon Line Detection

4.1. *MSEEs Filtering*

During testing, the MSEE image $E(x, y)$ is generated for a given query image according to equation 4. To remove non-horizon MSEEs, each MSEE location is treated as a key point; a local descriptor is then computed at that location which is used to classify that MSEE location as belonging to the horizon or not. We refer to the resultant MSEE map as E_+. Assuming binary classifications, then E_+ can be expressed as follows:

$$E_+(x, y) = \begin{cases} 1, & \text{if } E(x, y) = 1 \,\&\, \text{Classifier}[D(E(x, y))] = 1. \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

where, D is the feature vector around at MSEE location $E(x, y)$. In addition to the reduction of edges caused by the MSEE step, as shown in figure 2, MSEE classification reduces the number of non-horizon edges significantly. Figure 6 shows an example to highlight the significant reduction of non-horizon edges for a query image.

4.2. *Graph Formulation*

Instead of using the output of the edge detector in the context of Dynamic Programming, we use the classified MSEE image E_+ to formulate the multi-stage graph $G(V, E, \Psi, \Phi)$. Therefore, equations 2 and 3 are modified accordingly:

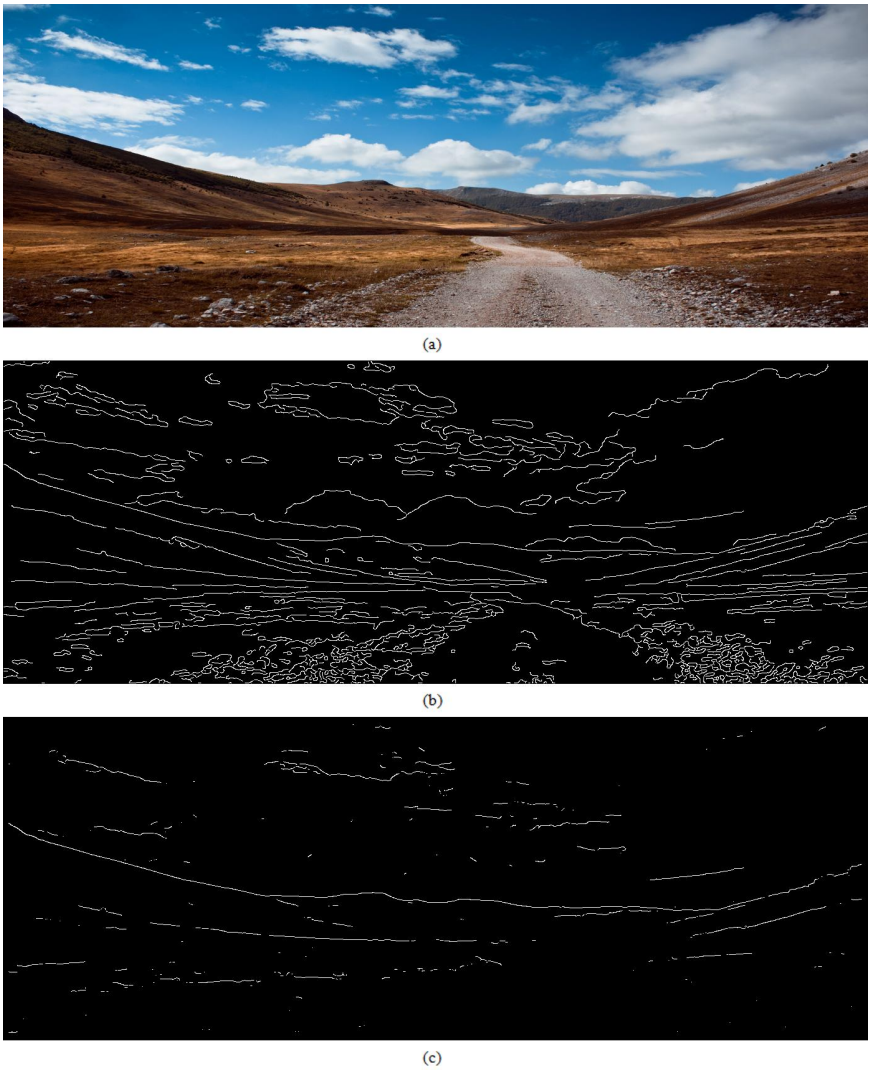

Fig. 6. Effect of trained classifier: (a) sample novel image, (b) corresponding MSEE and (c) MSEE_+ (c) images. Note the reduction in number of edges due to the classifier.

$$\Psi(i,j) = \begin{cases} l, & \text{if } E_+(x,y) = 1. \\ \infty, & \text{if } E_+(x,y) = 0. \end{cases} \tag{6}$$

$$\Phi(i,k,j) = \begin{cases} |i-k|, & \text{if } E_+(i,j) = E_+(k,j+1) = 1 \\ & \text{and } |i-k| \leq \delta \\ \infty, & \text{otherwise.} \end{cases} \tag{7}$$

Since, the number of candidate horizon edges are reduced considerably using the classified MSEE image, we do not enforce the bias that the horizon line in the upper half of the image. However, any kind of gaps are filled following the conventional method as explained in Section 2. Next, a source node and a sink node are added to the graph and linked with nodes in stages 1 and N assuming a zero cost. Finally, Dynamic Programming is applied to find the horizon line.

4.3. *Nodal Costs*

Lie et al.[3] have proposed using binary costs to encode whether a node in the multi-stage graph represents an edge pixel or not; dummy nodes are initialized to a high cost. Although we have reduced the number of non-horizon edges considerably using the classified MSEE image, we believe that using only binary costs to initialize the nodal costs is not enough. For example, Dynamic Programming might choose falsely classified horizon edges as part of the solution. In this paper, we have experimented with various nodal costs in order to provide additional information about the possibility of a positively classified MSEE location being part of the horizon line. Specifically, we have experimented with using the following information in defining the nodal costs:

- Gradient Information
- Classified Binary Edges
- Classified Edge Score
- Classified Edge Score + Gradient Information

4.3.1. *Gradient Information*

In contrast to Lie et al.[3] and others who have formulated the multi-stage graph using edges only, we propose building a dense multi-stage graph where each pixel in the image corresponds to a node which is connected to its neighbors in the next stage. To initialize the nodal costs, we use gradient magnitude information. When applying Dynamic Programming, the objective is finding a solution where the sum of gradient magnitudes is maximized. To ensure good continuity, we enforce the constraint that the difference between the gradient magnitudes of adjacent pixels is minimized. It should be noted that gradient magnitude based approach does not involve any kind of training.

Given a query image $Q(x, y)$, the gradient magnitude at each pixel of the image is computed as follows:

$$\nabla(x, y) = \Gamma[Q(x, y)] \tag{8}$$

where, Γ is the function which takes a gray scale image as input and returns the gradient magnitude image ∇. We have used the Canny edge detector in our experiments. Next, the difference of the gradient magnitude image is computed.

Since a node i in stage j can be connected to as many nodes in stage $j+1$, as defined by the δ parameter, several gradient magnitude difference images need to be generated. The equation below shows the gradient magnitude difference image assuming that we connect nodes at the same level between stages:

$$d\nabla(i,j) = |\nabla(i,j) - \nabla(i,j+1)| \tag{9}$$

Since we want to maximize the gradient magnitude while minimizing the difference of gradient magnitudes, we normalize the gradient magnitude and difference images between 0 and 1. The nodal costs $\Psi(i,j)$ of the multi-stage graph are defined as a weighted combination of these two images as shown in the equation below:

$$C_1(i,j) = \Psi(i,j) = w * d\nabla(i,j) + (1-w) * (1 - \nabla(i,j)) \tag{10}$$

where w is the weight parameter; in our experiments, we have set $w = 0.5$. The link costs may be initialized using equation 7; however, in our experiments we consider all link weights to be equal and have set them to zero due to using a small neighborhood (i.e. $\delta = 1$) due to the fact that we do not need to deal with gaps. Figure 7 shows the gradient magnitude image for a given image, the difference of gradient magnitude image, and their weighted combination.

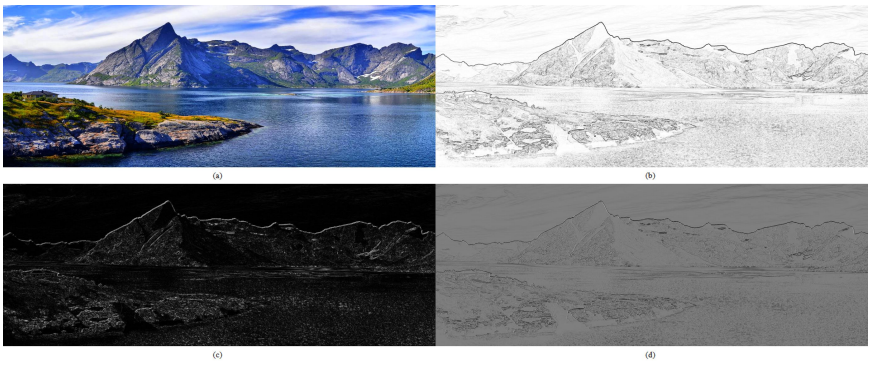

Fig. 7. Gradient Information;(a) original image,(b) gradient magnitude image,(c) difference of gradient magnitude Image and (d) combined magnitude and difference image.

4.3.2. *Classified Binary Edges*

This formulation is fairly similar to the one in Lie et al.[3], that is, we use binary edge information to define the nodal costs. The only difference between their approach and our approach is that the graph is formulated using the classified MSEE image. Therefore, equations 5 - 7 are used to initialize the graph and set the nodal/link

costs. Classification is performed using SIFT-HOG features as they outperform all other combinations that we have explored.

4.3.3. *Classified Edge Score*

Representing edge pixels using binary classification provides no information about the likelihood of an edge pixel being a horizon or non-horizon edge pixel. To enforce this information, we propose a two fold use of our classification framework. First, we use our classification results to distinguish between horizon and non-horizon edges as realized by equation 5 and second, to provide some confidence about and horizonness of an edge pixel. In this context, we normalize the raw classification scores between 0 and 1. The nodal costs are then initialized by the actual classification scores instead of setting all positively classified edges to a fixed cost. We have modified equation 6 to reflect this information where Ω is the classification score in the interval [0–1]. Since, Dynamic Programming is applied to find the shortest path in the graph, we have reversed the classification score values so that the smaller the classification score value is the more likely is that the pixel is a horizon pixel.

$$C_2(i,j) = \Psi(i,j) = \begin{cases} \Omega(E_+(x,y)), & \text{if } E_+(x,y) = 1. \\ \infty, & \text{if } E_+(x,y) = 0. \end{cases} \tag{11}$$

4.3.4. *Classified Edge Score + Gradient Information*

In this formulation, we have combined the classifier information with the gradient information. By combining equations 10 and 11, the nodal costs can be initialized as follows:

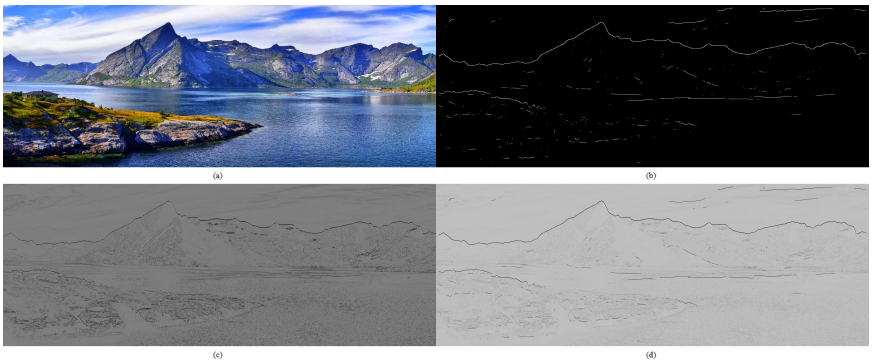

Fig. 8. Classifier edges and gradient Information;(a) original Image,(b) classified edge score,(c) gradient magnitude and difference of gradient magnitude and (d) gradient information combined with classified edge scores.

$$\Psi(i,j) = \begin{cases} w_2 * C_2(x,y) + (1-w_2) * C_1(x,y), & \text{if } E_+(x,y) = 1. \\ \infty, & \text{if } E_+(x,y) = 0. \end{cases} \quad (12)$$

where, w_2 is a weight parameter; we have set $w_2 = 0.5$ in our experiments. Figure 8 shows the gradient information image, classifier score image and their combination for a sample query image from our web data set.

The steps of our testing phase are shown in figure 9 .

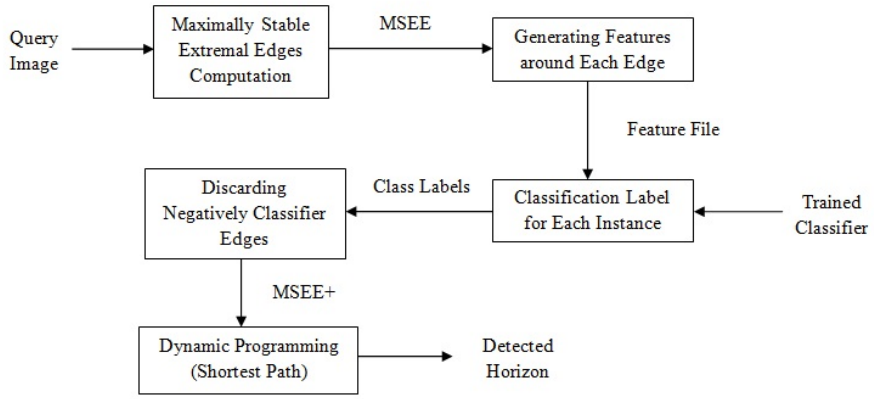

Fig. 9. Flow diagram for the testing pahse of the proposed approach.

5. Experiments and Results

5.1. *Data Sets*

We have experimented with two different data sets: the Basalt Hills data set and Web data set. The Basalt Hills data set consists of 45 images chosen from a larger data set based on a field study for outdoor robots. The Web data set consists of 80 mountainous images that have been randomly collected from the web. This data set is much more challenging than the Basalt Hills data set and includes various viewpoints, geographical and seasonal variations. 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.

5.2. *Edge Reduction Using MSEE*

To evaluate the effectiveness of MSEE in reducing the number of edges, we have computed the MSEE image for all the images in our data sets. Then, we have compared the original number of edges with the number of edges survived after

applying the MSEE approach. Table 1 shows the average percentage of reduction for each data set.

Table 1. Effect of MSEE

Data Set	Average % Reduction
Basalt Hills	66.37
Web	43.45

5.3. *Reduction of Non-Horizon Edges using Classification*

To evaluate the effectiveness of different features in classifying edge pixels as horizon or non-horizon pixels, we have performed a 5-fold cross-validation using the Basalt Hills data set. In each fold, the data set is divided into non-overlapping training (9 images) and test sets (36 images). Table 2 shows the percentage false positives and false negatives averaged over the five folds and the respective standard deviations. Figure 10 shows a graphical representation of the same results. Since, reducing the false negative rate is more important for horizon line extraction, we have chosen the classifier based on SIFT-HOG features for further evaluation, as highlighted in table 2. Out of the 5-folds we have chosen the classifier which resulted into the least false negative error for further experiments detailed in next subsection.

Table 2. Mean and standar deviations for % False Positive and False Negative Errors due to Various Features and their Combinations

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

5.4. *Horizon Line Extraction Using Different Nodal Costs*

In this section, we provide a comparison between the proposed nodal cost formulations presented in Section 4 and the approach of Lie et al.[3] In each case, the detected and true horizon lines are compared by calculating a pixel-wise absolute distance S between them as shown below. 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

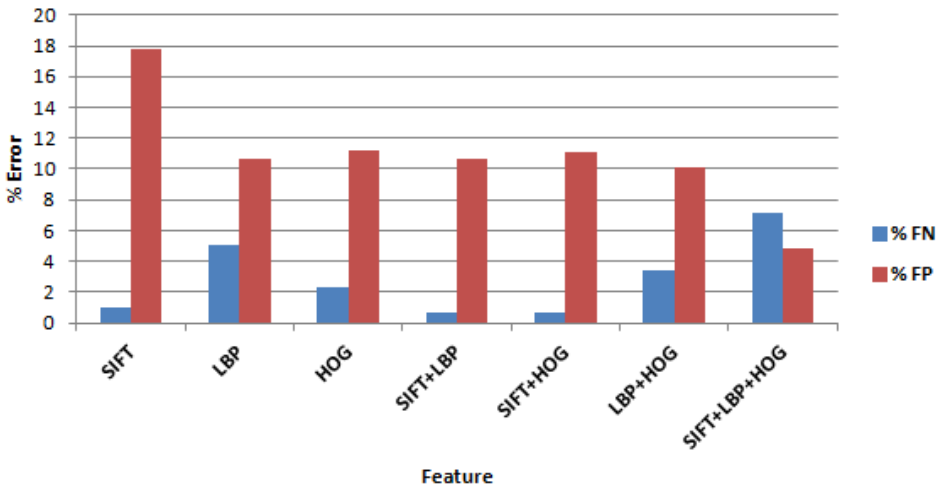

Fig. 10. Mean of % False Positive and False Negative rates when various features and their combinations are used for training the SVM classifier

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. Specifically, S is computed as follows:

$$S = \frac{1}{N} \sum_{j=1}^{N} |P_{d(j)} - P_{g(j)}| \tag{13}$$

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. For each method, we computer the average and standard deviation over all images in the data set listed as shown in table 3. Clearly, using edge classification scores based on SIFT+HOG features outperforms all other approaches. Our results reinforce the fact that using only edge information for the nodal costs is not enough.

Figure 11 shows several examples from each data set where our method has successfully detected the horizon line while Figure 12 shows examples where our method has failed to detect portions of the horizon line reliably. There are several reasons why the proposed approach does not perform well in some cases. One reason is due to the fact that we disallow nodes in some stage of the graph to connect with nodes in the same stage (i.e., only with nodes in the next stage). This is problematic when the horizon line has high slope (i.e., steep peaks). Allowing connections within the same stage will lead to more accurate horizon line detection at the expense of higher time requirements. Another reason is due to using a very small set of

training images (i.e., only 9 images from the Basalt Hill data set). This issue can be addressed by increasing the training set and making it more versatile. Finally, due to false negatives, a few horizon edges may be missed. Although gap filling tries to fill up the gaps, it does not always produce satisfactory results, especially when the gap is large. Similarly, false positives could lead to non-horizon consistent edges which might become part of the horizon line solution.

Fig. 11. Examples of horizon line detection using the proposed method; Balsat Hill data set [row 1 and 2], Web data set [rows 3 through 9]

6. Conclusion

We have presented a horizon line detection approach based on coupling dynamic programming with machine learning. In this context, we have investigated various local features and their combinations to reduce the number of non-horizon edges. Furthermore, we have considered dynamic programming for horizon line extraction and experimented with various nodal costs. Our experimental results show considerable improvements compared to the tradition edge-based horizon detection method

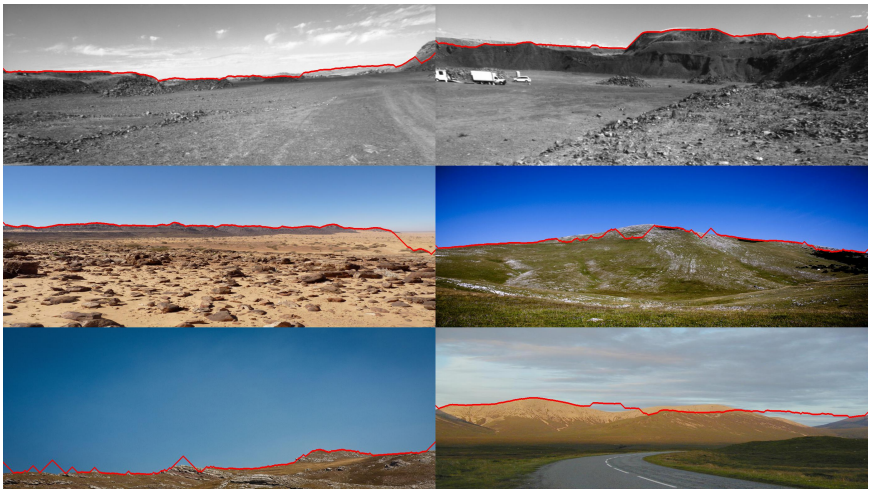

Fig. 12. Examples where the proposed approach misses portions of the horizon line.

Table 3. Average absolute errors between the detected and ground truth horizons for various choices of Nodal Costs used for Dynamic Programming

Nodal Costs	Basalt Hills		Web	
	Mean	Std. Dev.	Mean	Std. Dev.
Lie et al. (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

of Lie et al.[3] using two challenging data sets. For future work, we intend to explore horizon line as a localization cue in a GPS denied environments (e.g., planetary rover localization).

Acknowledgments

This work was supported by NASA EPSCoR under Cooperative Agreement No. NNX10AR89A.

References

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 gray-scale 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 *Proceedings of European Conference on Computer Vision*, 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. *International Conference on Machine Learning and Applications*, 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 Tech-*

nology 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.

(1983) 400–433.