

# Resource Efficient Mountainous Skyline Extraction using Shallow Learning

Touqeer Ahmad\*, Ebrahim Emami†, Martin Čadík‡, George Bebis†

\*Vision and Security Technology Lab, University of Colorado at Colorado Springs, USA  
touqeer@vast.uccs.edu

†Department of Computer Science and Engineering, University of Nevada, Reno, USA  
ebrahim@nevada.unr.edu, bebis@cse.unr.edu

‡ Faculty of Information Technology, Brno University of Technology, Czech Republic  
cadik@fit.vutbr.cz

**Abstract**—Skyline plays a pivotal role in mountainous visual geo-localization and localization/navigation of planetary rovers/UAVs and virtual/augmented reality applications. We present a novel mountainous skyline detection approach where we adapt a shallow learning approach to learn a set of filters to discriminate between edges belonging to sky-mountain boundary and others coming from different regions. Unlike earlier approaches, which either rely on extraction of explicit feature descriptors and their classification, or fine-tuning general scene parsing deep networks for sky segmentation, our approach learns linear filters based on local structure analysis. At test time, for every candidate edge pixel, a single filter is chosen from the set of learned filters based on pixel’s structure tensor, and then applied to the patch around it. We then employ dynamic programming to solve the shortest path problem for the resultant multistage graph to get the sky-mountain boundary. The proposed approach is computationally faster than earlier methods while providing comparable performance and is more suitable for resource constrained platforms e.g., mobile devices, planetary rovers and UAVs. We compare our proposed approach against earlier skyline detection methods using four different data sets. Our code is available at [https://github.com/TouqeerAhmad/skyline\\_detection](https://github.com/TouqeerAhmad/skyline_detection).

## I. INTRODUCTION

Skyline detection is the problem of finding a path that extends from left-most column of an image to the right-most and divides an image into sky and non-sky regions. Finding skyline is a challenging vision problem due to non-linear nature of the skylines, variations in the sky and non-sky regions due to geographical terrains and weather conditions. Skyline detection serves as the underlying method for many practical applications and have been investigated for navigation and localization of Unmanned Aerial Vehicles (UAVs) [1]–[9], planetary rover and vehicle localization [10]–[17], augmented reality and tourism applications [18]–[20], geolocation of mountain and desert images [18], [21]–[25], marine security and ship detection [26]–[29].

Sky segmentation and skyline detection are two related yet distinct problems. In a general scene parsing sense, sky segmentation is equivalent to semantically identifying all the pixels belonging to the sky region where the sky region may or may not extend from left-to-right and may be present in a small portion of the image. In the case of the skyline extraction problem, a skyline is thought of as a **visible** boundary which

extends from **left-to-right** and divides the input image into two main regions i.e. sky and non-sky. Depending upon the specificity of the non-sky region, the skyline could refer to the mountainous skyline, sea-shore skyline or city skyline. While any general scene-parsing network can be fine-tuned for sky segmentation as demonstrated in [30], the two problems are not inter-changeable, unless the assumptions for skyline hold true.

Recent efforts for skyline detection can be grouped into two main categories. The first group of approaches employs supervised learning to discriminate between skyline and non-skyline pixels by either using feature descriptors or directly discriminating based on the pixel intensities [31]–[37]. Some approaches belonging to this category use edge information while others solely rely on discrimination power of the trained classifier. These approaches generally address skyline detection as a shortest-path problem and incorporate dynamic programming (DP) which was first utilized for skyline detection problem in [38]. The second type of approaches address the skyline detection in the context of sky-segmentation problem and generally rely on Convolutional Neural Networks (CNNs) to segment out the sky regions [19], [39], [40]. There have also been some attempts to fine-tune semantic segmentation networks for sky segmentation [30]. While both types of methods perform reasonably well for skyline detection, they are computationally expensive solutions e.g., in some cases, single or multiple feature descriptors around each pixel need to be extracted before passing them through the trained classifier (Support Vector Machines/Convolutional Neural Networks) or directly employing dense deep networks which are inherently quite expensive. Due to expensive computational and memory requirements, such approaches are not very suitable for resource-constrained platforms e.g., mobile devices, UAVs and planetary rovers. To address these issues, in this paper we present a skyline detection approach which builds upon a recently introduced **shallow learning** framework [41] targeted for **resource-constrained** devices and provides a reasonable trade-off between computational cost and performance.

The rest of the paper is structured as follows: Section II provides the details of earlier work on skyline detection methods relying on dynamic programming (DP) and establishes the

foundation for our proposed approach and its comparisons. In Section III, we describe the shallow learning framework employed for our proposed skyline detection approach. Experimental details and results are provided in Section IV. The paper is concluded with pointers for future work in Section V.

## II. SKYLINE DETECTION USING DYNAMIC PROGRAMMING

In this section, we outline the details of earlier skyline detection methods which address the underlying problem as graph-search and rely on dynamic programming to solve it. Since our proposed approach is also based on dynamic programming, it is important to describe the details of these methods from comparison view-point and building the basis of the proposed approach.

### A. Skyline Detection as Graph Search Problem

Lie *et al.* [38] are the first one to formulate skyline detection as a multi-stage graph search problem. Given an  $M \times N$  image, the image is first passed through an edge detector to compute a binary edge map  $E(i, j)$  where 1/0 imply the presence/absence of an edge pixel:

$$E(i, j) = \begin{cases} 1, & \text{if } (i, j) \text{ is an edge pixel,} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

They then used the edge map  $E(i, j)$  to build an  $M \times N$  multi-stage graph  $G(V, E, \Psi, \Phi)$  where each pixel in the map corresponds to a graph vertex  $v_{ij}$ ; a low cost  $l$  is associated with edge pixels (vertices) while a very high cost (i.e.,  $\infty$ ) is associated with non-edge pixels as shown below:

$$\Psi(i, j) = \begin{cases} l, & \text{if } E(i, j) = 1, \\ \infty, & \text{otherwise.} \end{cases} \quad (2)$$

$\Psi(i, j)$  is the cost associated with vertex  $i$  in stage  $j$  (i.e.,  $v_{ij} \in V$ ). It should be noted that use of  $\infty$  reflects a node with a high numeric cost. They have further used a gap filling process to address edge-gaps. Given a node  $i$  in stage  $j$ , its neighborhood in the next stage  $j + 1$  is defined by a  $\delta$  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:

$$\Phi(i, k, j) = \begin{cases} |i - k|, & \text{if } E(i, j) = E(k, j + 1) = 1 \\ & \text{and } |i - k| \leq \delta, \\ \infty, & \text{otherwise.} \end{cases} \quad (3)$$

If a node  $i$  in stage  $j$  cannot be connected to any node in stage  $j + 1$  within  $\delta$  distance, a search window is defined using two parameters:  $\delta$  and tolerance-of-gap (tog). If an edge node  $k$  is found within the search window, gap filling is performed by introducing dummy nodes between node  $i$  in stage  $j$  and node  $k$  within the search window  $j + \text{tog}$  and a high cost is associated with such dummy nodes. More details for Lie *et al.* can be found in [38] or [36].

### B. Skyline Detection using Supervised Learning and Edges

In a series of work [30]–[36], Ahmad *et al.* extended the skyline detection approach of [38] by incorporating supervised machine learning techniques. Interested readers are requested to please consult their original papers [30]–[36].

In a simple extension of the binary edge map approach of Lie *et al.*, to ensure good continuity, Ahmad *et al.* [32], [34] used the gradient information. They enforced the constraint that the difference between gradient magnitudes of adjacent pixels is minimized. The gradient magnitude at each pixel of the input image  $I(i, j)$  is computed as follows:

$$\nabla(i, j) = \Gamma[I(i, j)], \quad (4)$$

where  $\Gamma$  is a function which takes a gray scale image  $I$  as input and returns the gradient magnitude image  $\nabla$ . Next, the difference of the gradient magnitude image  $d\nabla(i, j)$  is computed:

$$d\nabla(i, j) = |\nabla(i, j) - \nabla(i, j + 1)|. \quad (5)$$

The normalized (i.e., between 0 and 1) gradient magnitude and gradient difference images are combined through a weighted average:

$$G_r(i, j) = w_1 * d\nabla(i, j) + (1 - w_1) * (1 - \nabla(i, j)), \quad (6)$$

where  $w_1$  is the weight parameter. Then they used the weighted average  $G_r$  as the nodal cost:

$$\Psi(i, j) = G_r(i, j), \quad (7)$$

whereas, the link costs may be initialized using equation (3). In [31], [34], Ahmad *et al.* considered Maximally Stable Extremal Edges (MSEEs) and classification as a way to filter out non-skyline edges. They first applied Canny edge detector using various thresholds and kept only stable edges over a range of thresholds which they called MSEE edges. The MSEE edge map  $E_m(i, j)$  is further filtered by classifying each MSEE pixel  $(i, j)$  as belonging/not-belonging to the skyline using the trained classifier:

$$C(i, j) = \begin{cases} 1, & \text{if } (i, j) \text{ pixel is classified as skyline,} \\ 0, & \text{otherwise.} \end{cases} \quad (8)$$

In [31], [32], [34], authors experimented with various texture features and their combinations for classification and found the SIFT-HOG combination yielded the lowest false negative rate. The edge map  $E_+(i, j)$ , comprising of the horizon classified MSEE edges, was used to define the nodal costs in the context of DP. Specifically, the edge map comprising of positively classified MSEE edge points can be expressed as follows:

$$E_+(i, j) = \begin{cases} 1, & \text{if } E_m(i, j) = 1 \text{ and } C(i, j) = 1, \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

Authors have incorporated the classification information into nodal costs in two ways: (i) by using the binary costs

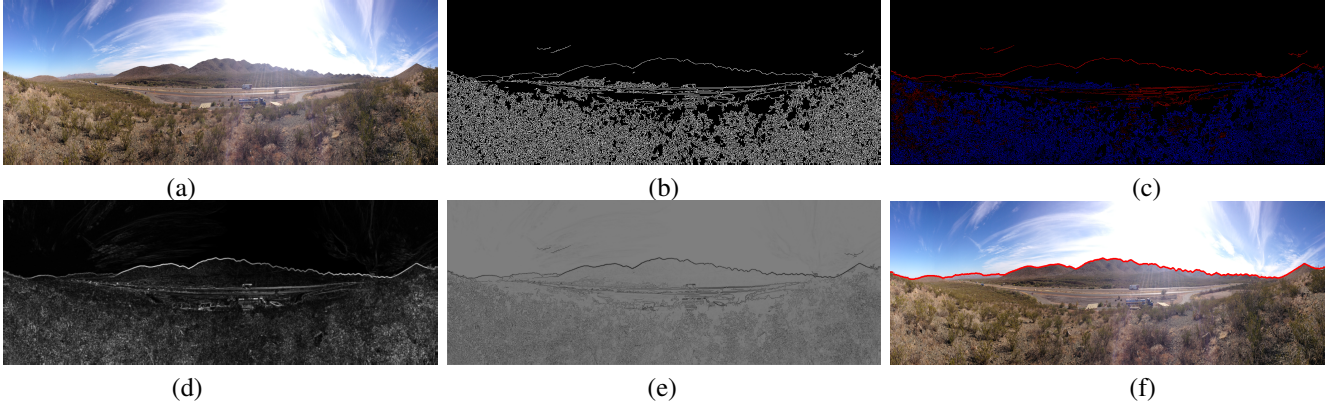


Fig. 1. Proposed Skyline Extraction Approach: (a) input image, (b) output of the Canny edge detector, (c) predicted score for each pixel belonging to skyline using selected linear filter based on pixel's structure tensor (brighter red and blue intensities respectively reflect more and less confidence for pixel belonging to the skyline), (d) gradient strength estimated as part of the structure tensor, (e) weighted predicted skyline score combined with gradient strength, (f) found skyline by dynamic programming overlaid on the original query image in red.

based on  $E_+(i, j)$  and (ii) by using the normalized classification scores  $S(i, j)$  for these edges. For each case, the nodal-cost (2) changes accordingly:

$$\Psi(i, j) = \begin{cases} l, & \text{if } E_+(i, j) = 1, \\ \infty, & \text{otherwise.} \end{cases} \quad (10)$$

$$\Psi(i, j) = \begin{cases} S(i, j), & \text{if } E_+(i, j) = 1, \\ \infty, & \text{otherwise,} \end{cases} \quad (11)$$

whereas, by using  $E_+(i, j)$  the link-cost (3) adapts accordingly:

$$\Phi(i, k, j) = \begin{cases} |i - k|, & \text{if } E_+(i, j) = E_+(k, j + 1) = 1 \\ & \text{and } |i - k| \leq \delta, \\ \infty, & \text{otherwise.} \end{cases} \quad (12)$$

They further combined classification scores with gradient information and adjusted the nodal-cost as below:

$$\Psi(i, j) = w_2 * S(i, j) + (1 - w_2) * G_r(i, j), \quad (13)$$

where  $w_2$  is a weight parameter.

### C. Skyline Detection using Dense Classification Score Images

In the second set of their work [33], [35], [36], Ahmad *et al.* investigated to exclude the edge-detection and generated a dense classification score image (DCSI) which reflects the likelihood of a pixel belonging to the skyline. The resultant DCSI is then used directly to initialize the nodal-cost:

$$\Psi(i, j) = S(i, j). \quad (14)$$

The difference between Eqs. (11) and (14) is that in the former, classification score is used to initialize only those nodes which are MSEE edges and have been positively classified as skyline edges whereas in the latter, all the nodes have been initialized with the normalized classification scores without using any edge information. In the former case SIFT-HOG

features were extracted around  $(i, j)$  location while in the latter normalized pixel intensities were used.

For this work, they have investigated both Support Vector Machine (SVM) and Convolutional Neural Network (CNN) classifiers and resultant dense classification score images are identified by SVM-DCSI and CNN-DCSI respectively. For computational improvement, they further experimented by retaining only the best m-scores per stage (column) of the multi-stage graph which they termed as SVM-mDCSI and CNN-mDCSI. In extended work [36] they investigated the fusion strategies by combining MSEE edges, Canny edges or gradient information with their DCSI images.

## III. SHALLOW LEARNING

Our shallow learning is inspired and based on the BLADE framework introduced by Getreuer *et al.* [41] which is a generalization of popular super-resolution approach RAISR [42]. BLADE has been demonstrated to perform well for various computational photography applications including denoising [43], demosaicing and image stylization [44]. Unlike earlier applications, we adopt this shallow learning framework for classification and explore its applicability for mountainous skyline extraction problem. Below, we briefly describe the inference and training steps of BLADE for a sample image restoration problem i.e., denoising and then extend these steps for our skyline extraction problem.

### A. Spatially Varying Inference

Let  $\mathbf{z}$  and  $\mathbf{u}$  be input (i.e., noisy image) and output (i.e., denoised image) gray-scale images respectively. A pixel at  $i$ -th spatial location is specified by  $\mathbf{z}_i$ . Let there be a set of  $K$  linear finite impulse response (FIR) filters denoted by  $\mathbf{h}^1, \dots, \mathbf{h}^K$ . The coefficients of these filters are learned during training phase. At inference time, a spatially varying filter of size  $n \times n$  is selected from the learned filter bank and applied on patch

TABLE I  
AVERAGE ABSOLUTE ERROR FOR EXISTING AND PROPOSED APPROACHES

Approach	Basalt Hills				Web			
	$\mu$	$\sigma$	$min$	$max$	$\mu$	$\sigma$	$min$	$max$
<b>Edges Only [38]</b>	5.55	9.46	0.53	49.31	9.15	17.92	0.38	93.02
$G_r$ ([32], [34])	3.99	6.35	0.18	31.33	11.86	26.81	0.15	121.48
<b>SIFT+HOG Edges</b> ([32], [34])	0.57	1.02	0.00	3.58	0.87	1.03	0.43	7.05
<b>SIFT+HOG Scores</b> ([32], [34])	0.41	0.81	0.00	3.08	0.97	1.57	0.38	12.19
<b>SIFT+HOG Scores + <math>G_r</math></b> ([32], [34])	0.43	0.81	0.00	3.08	1.30	3.98	0.38	34.95
<b>SVM-mDCSI</b> ([33], [36])	1.01	0.29	0.62	1.76	1.28	1.20	0.37	6.21
<b>CNN-mDCSI</b> ([33], [36])	0.75	0.23	0.42	1.28	1.41	1.49	0.27	10.79
<b>SVM-DCSI+<math>G_r</math></b> ([36])	0.60	0.29	0.17	1.31	4.86	15.98	0.14	98.46
<b>SVM-DCSI+MSEE Edges</b> ([36])	0.73	0.32	0.48	2.07	0.85	0.89	0.35	5.05
<b>SVM-DCSI+Canny Edges</b> ([36])	0.77	0.35	0.48	2.07	0.78	0.76	0.35	4.84
<b>Proposed</b>	1.45	1.55	0.24	5.69	7.85	32.64	0.29	203.34

extracted around the central pixel  $\mathbf{z}_i$ . In vector notation, the inference can be written as a dot-product,

$$\hat{u}_i = (\mathbf{h}^{s(i)})^T \mathbf{R}_i \mathbf{z}, \quad (15)$$

where,  $\mathbf{R}_i$  extracts a patch centered around pixel  $i$  rearranged as vector,  $\mathbf{h}^{s(i)}$  is the filter selected from the filter bank for the  $i$ -th pixel based on its structure tensor,  $(\cdot)^T$  denotes the vector transpose operator and  $\hat{u}_i$  is the inferred pixel value.

For computational photography and stylization tasks such as studied in [41], [43], [44], the above operation is conducted for each pixel in the input image. For the skyline extraction problem, we first perform edge detection on the input image and then extract the gray-scale patches only around pixels detected as edges. The inferred pixel value  $\hat{u}_i$  reflects the score of  $i$ -th pixel belonging to the skyline.

### B. Learning the Filter Bank

To learn the set of FIR filters  $\mathbf{h}^1, \dots, \mathbf{h}^k$ , training set comprising of input image and output target pairs is employed. Depending on the underlying application,  $\mathbf{z}$  and  $\mathbf{u}$  are the appropriate image pairs e.g., for learned denoising,  $\mathbf{z}$  and  $\mathbf{u}$  would be noisy and noise-free images respectively. The filter set is learned by minimizing the objective function comprised of  $L^2$  loss and quadratic regularization term i.e.,

$$\arg \min_{\mathbf{h}^1, \dots, \mathbf{h}^k} \sum_{k=1}^K (\mathbf{h}^k)^T \mathbf{Q} \mathbf{h}^k + \sum_{s(i)=k} (u_i - (\mathbf{h}^k)^T \mathbf{R}_i \mathbf{z})^2, \quad (16)$$

where,  $\mathbf{Q}$  determines the regularization and encourages the learned filters to be smooth. The minimization described in (16) can be solved independently for each filter and amounts to a multivariate linear regression with regularization listed below:

$$\mathbf{h} = (\mathbf{Q} + \mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}. \quad (17)$$

All the training examples belonging to a specific filter are accumulated in an  $(\mathbf{n}^2 + 1) \times (\mathbf{n}^2 + 1)$  gram matrix  $\mathbf{G}$ ,

$$\mathbf{G} \leftarrow \mathbf{G} + \begin{pmatrix} \mathbf{R}_i \mathbf{z} \\ u_i \end{pmatrix} \begin{pmatrix} \mathbf{R}_i \mathbf{z}^T u_i \end{pmatrix}. \quad (18)$$

Once the training samples are accumulated in  $\mathbf{G}$ , the respective matrices  $\mathbf{A}^T \mathbf{A}$  and  $\mathbf{A}^T \mathbf{b}$  for (17) can be accessed:

$$\begin{pmatrix} \mathbf{A}^T \mathbf{A} & \mathbf{A}^T \mathbf{b} \\ \mathbf{b}^T \mathbf{A} & \mathbf{b}^T \mathbf{b} \end{pmatrix} \leftarrow \mathbf{G}. \quad (19)$$

To learn the filters for skyline detection, we use positive training examples extracted along the ground truth skyline boundary and equal number of negative training examples extracted from random edge locations not in the vicinity of the skyline. We use two specific intensity levels as target  $u_i$  for positive and negative examples. At inference time, after normalization,  $\hat{u}_i$  reflects the confidence score of an edge-pixel belonging or not belonging to the skyline.

### C. Filter Selection using Structure Tensor

Following BLADE [41], we use local structure tensor to select the filter for a given pixel both during training and inference. Structure tensor is a way to provide local gradient estimate by employing Principal Component Analysis (PCA) of the gradients of  $i$ -th pixel's local neighborhood  $P_i$  by minimizing:

$$\arg \min_a \sum_{j \in P_i} w_j^i (\mathbf{a}^T \mathbf{g}_j)^2, \quad (20)$$

where  $\mathbf{g}_j$  is the gradient at  $j$ -th pixel and  $w_j^i$  is the spatial weighting e.g., coefficients of a Gaussian kernel. Instead of computing structure tensor using only Luma channel as done in [42], we adopt computing it jointly using all color channels as in [41]. Using a  $2 \times 3$  Jacobian matrix of color-wise gradients,

$$\mathbf{G}_j = [\mathbf{g}_j^R \quad \mathbf{g}_j^G \quad \mathbf{g}_j^B], \quad (21)$$

the unit vector  $\mathbf{a}$  is found by minimizing (20) and replacing  $\mathbf{g}_j$  with  $\mathbf{G}_j$ :

$$\sum_j w_j^i \|\mathbf{a}^T \mathbf{G}_j\|^2 = \mathbf{a}^T \left( \sum_j w_j^i \mathbf{G}_j \mathbf{G}_j^T \right) \mathbf{a} = \mathbf{a}^T \mathbf{T}_i \mathbf{a}. \quad (22)$$

The spatially weighted structure tensor  $\mathbf{T}_i$  for  $i$ -th pixel can be written as:

$$\mathbf{T}_i = \sum_c \sum_j w_j^i \begin{bmatrix} g_{x,j}^c g_{x,j}^c & g_{x,j}^c g_{y,j}^c \\ g_{x,j}^c g_{y,j}^c & g_{y,j}^c g_{y,j}^c \end{bmatrix}, \quad (23)$$

where  $c \in [R, G, B]$  the three color channels and  $[g_{x,j}^c, g_{y,j}^c]^T = \mathbf{g}_j^c$ . The three features computed from the eigen analysis of this  $2 \times 2$  matrix  $\mathbf{T}_i$  serve as the indices for feature selection after their respective quantization into pre-specified bins. The three features being computed are: (i)





Fig. 2. Examples of good skyline detections by the proposed approach. Detected skylines are overlaid in red.

*orientation* as the angle of the dominant eigenvector reflecting orientation of the gradient, (ii) *strength* as the square root of the dominant eigenvalue  $\sqrt{\lambda_1}$  reflecting gradient magnitude and (iii) *coherence* characterizing the amount of anisotropy of local structure defined as:

$$\frac{\sqrt{\lambda_1} - \sqrt{\lambda_2}}{\sqrt{\lambda_1} + \sqrt{\lambda_2}}, \quad (24)$$

where  $\lambda_1 \geq \lambda_2 \geq 0$  are the eigenvalues of  $\mathbf{T}_i$ .

#### D. Proposed Skyline Detection Approach

Unlike Ahmad *et al.*, we do not rely on extracting feature descriptors (and their combinations) and then training SVM/CNN classifiers. Instead, for every given pixel, a linear filter is chosen from the learned filter bank based on its structure tensor and applied (dot product) to the patch around the central pixel. Further, we generate the prediction only for the pixels detected as edges by the edge-detector. In spirit, our approach is closer to that of SIFT-HOG classification score described in (11), however we use Canny edge detector instead of MSEE edges. Using Eqs. (15), (2) and (11), the nodal-cost can be described as:

$$\Psi(i, j) = \begin{cases} (\mathbf{h}^{s(k)})^T \mathbf{R}_k \mathbf{z}, & \text{if } E(i, j) = 1, \\ \infty, & \text{otherwise,} \end{cases} \quad (25)$$

where,  $k$  and  $(i, j)$  correspond to the same central pixel in two different formulations. Although (25) can be directly used to solve the shortest path problem through dynamic programming

to find the skyline, we have found through experimentation that complementing the normalized prediction score with gradient strength improves the results. Since gradient magnitude  $\sqrt{\lambda_1}$  is already computed as part of structure tensor analysis, we do not encounter any computational overhead and combine the normalized prediction score with the normalized gradient strength:

$$\Psi(i, j) = v * (1.0 - (\mathbf{h}^{s(k)})^T \mathbf{R}_k \mathbf{z}) + (1 - v) * (1.0 - \sqrt{\lambda_1(i, j)}), \quad (26)$$

where  $v$  is a weight parameter. Both normalized gradient strength  $[0, 1]$  and normalized prediction score  $[0, 1]$  are subtracted from 1.0 as we are trying to solve a minimization problem through dynamic programming. Figure 1 demonstrates the steps involved in our proposed skyline detection approach.

## IV. EXPERIMENTAL SETUP

### A. Data Sets for Skyline Detection

There are several public data sets annotated for the problems of mountainous skyline detection and visual geo-localization which are briefly described below.

*Basalt Hills and Web:* The Basalt Hills data set is a subset of a larger data set captured by placing stereo cameras on an autonomous robot and navigating it through the Basalt Hills in California [45] to mimic a planetary-rover environment. Ahmad *et al.* have manually generated ground truth for 45 images from this data set and have used for their experiments. They have used 9 out of these 45 images as their training set



Fig. 3. Examples of faulty skyline detections by the proposed approach. Detected skylines are overlaid in red.

TABLE II  
AVERAGE ABSOLUTE ERROR FOR PROPOSED APPROACH

Approach	Basalt Hills (Full)				Web (73)			
	$\mu$	$\sigma$	$min$	$max$	$\mu$	$\sigma$	$min$	$max$
Proposed	1.33	1.45	0.23	5.69	0.79	0.68	0.29	4.66

TABLE III  
IMPACT OF FILTER SIZE ON SKYLINE DETECTION PERFORMANCE

Filter Size	Basalt Hills (Full)				Web (73)			
	$\mu$	$\sigma$	$min$	$max$	$\mu$	$\sigma$	$min$	$max$
$7 \times 7$	1.3334	1.4542	0.23	5.69	0.7919	0.6864	0.29	4.66
$9 \times 9$	1.3402	1.4559	0.23	5.69	0.7921	0.6610	0.29	4.33
$11 \times 11$	1.3419	1.4547	0.23	5.69	0.7830	0.6366	0.29	4.45
$13 \times 13$	1.3557	1.4639	0.23	5.69	0.7887	0.6691	0.29	4.54
$15 \times 15$	1.3649	1.4593	0.23	5.69	0.7941	0.6872	0.29	4.68

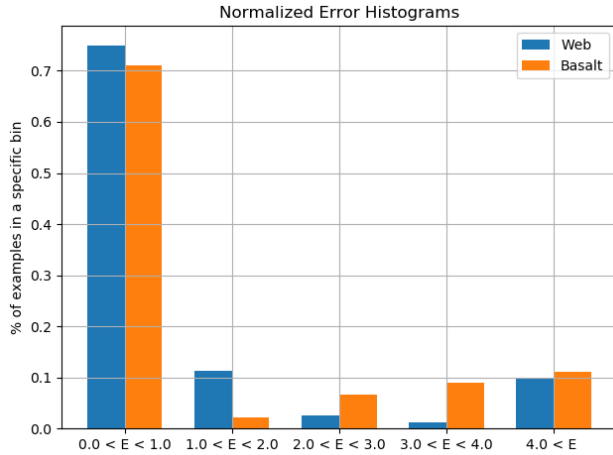


Fig. 4. Normalized histograms for Basalt Hills and Web data sets. In each case, for about 90% of the images we get an average absolute pixel error of less than 4.0 pixels which corresponds to a very good detection.

and remaining 36 along with the web set as their test set. The web data set is comprised of 80 images gathered from the web for which Ahmad *et al.* manually generated the ground truth skyline boundaries and used in their evaluations.

**CH1 and CH2:** CH1 [23] and CH2 [22] are visual geo-localization data sets which contain a total of about one thousand images with ground truth GPS location and FOV information. These datasets do not provide the ground truth camera orientation information and sky/non-sky segmentation is only available for 203 images belonging to CH1 data set.

**GeoPose3K:** Brejcha and Čadík [46] introduced the GeoPose3K data set that provides over three thousand images

with ground truth GPS, orientation and semantic labels<sup>1</sup>. This is probably the biggest data set available for mountainous geo-localization. However, for skyline extraction, this is an extremely challenging data set as many images do not have visible skylines and it may be difficult for a human annotator to properly mark the ground truths for such images without any additional information e.g., relying on Digital Elevation Models (DEMs) or Google Earth. We feel this data set is suitable for camera orientation and GPS estimation where the visible portions of the terrain can be used to align with the DEMs, but it is challenging for skyline extraction approaches which have the underlying assumptions of skyline being visible and extending from left-to-right. The data set is also suited for evaluating general sky segmentation approaches as demonstrated in [30], where specifically designed learning-based skyline detection approaches performed poorly compared to scene parsing networks fine-tuned for sky segmentation.

## B. Evaluation Metric

To quantitatively evaluate the performance of the proposed approach and compare against earlier approaches, we calculate the average pixel-wise absolute distance ( $A_{err}$ ) between the detected and ground truth skylines:

$$A_{err} = \frac{1}{N} \sum_{j=1}^N |P_{d(j)} - P_{g(j)}|, \quad (27)$$

where  $P_{d(j)}$  and  $P_{g(j)}$  are the positions (rows) of the detected and ground truth skyline pixels in column  $j$  and  $N$  is the number of columns in the test image.

<sup>1</sup>Semantic labels are synthesized from a Digital Elevation Model.

TABLE IV  
IMPACT OF TRAINING SET ON SKYLINE DETECTION PERFORMANCE

Training Set	Test Set					
	CH1		Basalt Hills (Full)		Web	
	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$
CH1	—	—	1.33	1.45	7.85	32.64
Basalt Hills (Full)	109.84	144.34	—	—	14.62	54.81
Web	101.87	140.18	1.27	1.39	—	—

### C. Results

**Base Results:** Unless otherwise stated, we use 203 images from the CH1 [23] data set to learn the filter bank, where by default we used a filter size of  $7 \times 7$  and 6, 3 and 16 quantization bins for strength, coherence and orientation computed from structure tensor analysis. We use Web and Basalt Hills data set to report our base results. For Basalt Hills data set, we first report results for 36 out of 45 images to be consistent with other reported results for earlier methods and then separately report the results for all 45 images. Table I shows mean ( $\mu$ ) and standard deviation ( $\sigma$ ) of the evaluation metric for our proposed approach against other methods. For all of the compared methods, we take the reported numbers from the individual papers which are specifically noted along with the approach’s name.

As can be seen from table I, our proposed approach performs poorer than other learning-based approaches where dedicated descriptors and non-linear classifiers are employed, whereas it outperforms other edge-based approaches which do not have any learning component. As reported in [41], [43], the BLADE framework is well-suited for resource constrained platforms e.g., mobile devices. Inheriting from BLADE, our skyline extraction approach provides a good trade-off between performance and computation and is suitable for applications with limited on-line memory and compute resources e.g., augmented reality, visual geo-localization and navigation of UAVs and planetary rovers. To better understand the performance of our approach, we plot the normalized histograms of the average absolute error for Basalt and Web data sets in figure 4. It is clear that for only 10% of the images in each data set, we get an average pixel error of 4.0 or more. Figure 2 shows some of the good skyline detection results for the Web data set, whereas some failure cases of the proposed approach from the same data set are shown in figure 3. As mentioned before, Ahmad *et al.* reported results for 36 out of 45 images for Basalt Hills data set. For a direct comparison, in table I, we also reported the error for our approach for the same 36 images, however, for completeness table II reports results for all 45 images identified as **Basalt Hills (Full)**. For Web data set, our approach resulted into an error of 4.0 pixels or more for 7 out of 80 images; excluding these images from the error computation results in mean and standard deviation which are directly comparable to the existing computationally intensive learning based methods. The error metric for these 73 web images are also reported in table II identified by the column **Web (73)**. We have identified number of reasons for

partial or full failure cases. Firstly skyline or portion of skyline are not strong edges and are missed by the edge detector and subsequently a prediction score is not generated for such pixels. Secondly the employed training data is limited which caused some of the feature bins to be not learned very well. At test time if a pixel falls to such a bin based on its structure tensor, the resultant prediction score might not be accurate. Finally there exist a shortest path in the image extending from left-to-right which is less expensive than the skyline path and hence been found by dynamic programming as the solution (e.g, middle image in figure 3).

**Filter Size:** Since filter size is directly related to the computational overhead, following [41], [43] we have used a filter size of  $7 \times 7$  for our base experiments. We conducted an experiment to evaluate the impact of filter size on skyline detection performance. Table III shows the performance of skyline detection on **Basalt Hills (Full)** and **Web (73)** as the filter size is varied. As evident from table III, there is no noticeable effect on skyline detection due to change in the filter size.

**Data Set Variation:** We have demonstrated results where CH1 data set has been used as a training set and Basalt Hills and Web data sets have been used as the test sets. To study the affect of the training set, we conducted an experiment where each of the three data sets serves as a training set while the remaining two are used as the test sets. Table IV demonstrates the results for this experiment. As expected the performance on the test set is proportional to the versatility and number of images in the used training set, e.g., in case of Basalt as a training set, the performance of skyline detector on Web set is poorer than when CH1 data set is used as training set. Similarly, on CH1 data set the performance is relatively better when trained on Web set compared to when trained on Basalt set. The overall skyline detection performance on CH1 data set is poor compared to Basalt and Web data which is understandable as CH1 data set is more challenging compared to the other two.

**GeoPose3K:** As mentioned earlier the GeoPose3K dataset [46] is more suited for general scene parsing approaches where sky segmentation is the underlying objective instead of skyline detection. Nonetheless, we provide the comparison of our proposed approach on GeoPose3K in table V where we report the numbers for existing approaches from [30] (Table III in that paper). In [30], authors fine-tuned general scene-parsing deep networks for sky segmentation using CH1 dataset and provided a comparison between scene-parsing and specific skyline detection methods using 2895 images from



TABLE V  
COMPARISON OF THE PROPOSED APPROACH ON GEOPOSE3K DATASET

Approach	Accuracy	Absolute Error	
		$\mu$	$\sigma$
FCN8s-Pascal [30]	0.9551	32.161	57.510
FCN16s-Pascal [30]	0.9539	32.888	58.193
FCN32s-Pascal [30]	0.9520	33.534	57.588
FCN8s-SiftFlow-g [30]	0.9491	34.975	53.334
FCN8s-SiftFlow-s [30]	0.9563	31.399	55.052
Horizon-ALE-CH1 [30]	0.9411	43.959	86.038
Horizon-DCSI-CH1 [30]	0.8743	99.742	160.252
SegNet-CH1 [30]	0.8437	114.893	99.021
FCN8s-SiftFlow-s-CH1 [30]	0.9486	37.947	69.435
FCN8s-Pascal-CH1 [30]	0.9432	41.596	71.707
<b>Proposed</b>	0.8652	105.712	164.224

the GeoPose3K dataset. Following [30] we also report the performance of our proposed approach on the same 2895 images in table V where we used CH1 dataset to learn the filter bank. In addition to the pixel-wise absolute distance, we also report the segmentation accuracy to be consistent with [30]. As expected our approach is out-performed by scene parsing approaches adapted for sky segmentation while comparable to specifically designed skyline detection methods e.g., Horizon-DCSI-CH1.

#### D. Resource Comparison

In table VI, we report the memory foot-print and the inference time for our proposed approach and compare it against selective existing methods. Specifically, we compare against top-performing deep networks (table V) adapted for sky-segmentation [30] and a representative approach designed for skyline detection i.e., Horizon-DCSI [33]. For each approach we report an average inference time for an image size of  $519 \times 1388$  pixels using a consistent single CPU environment (Processor: 1.8 GHz Intel Core i5, Memory: 8GB 1600 MHz DDR3). It should be noted that we do not include the time taken to load the coefficients/weights of a model. As clear from VI, each of the deep networks adapted for sky segmentation has a memory foot-print of more than 500 megabytes whereas the average inference time for specifically designed skyline detection approach is more than 20 seconds. Compared to these methods, our approach provides a middle ground where the memory requirement for our  $7 \times 7$  filter bank is just 57KB and inference time is the best, collectively rendering our approach best suited for resource constrained mobile devices where memory, inference time and battery life are of great concern. We should note that both Horizon-DCSI and our proposed approach can further benefit from GPU implementation as there are several identical operations being performed for every pixel in the image.

#### V. CONCLUSIONS

Earlier skyline detection approaches are based on supervised or deep learning and are unsuitable for resource constrained devices. In this paper, we have presented a computationally efficient and faster skyline detection approach which is based on the shallow learning framework specifically designed for

TABLE VI  
COMPARISON OF REQUIRED COMPUTATIONAL RESOURCES OF THE PROPOSED APPROACH AGAINST OTHERS

Approach	Memory (MB)	Inference Time (s)
FCN8s-Pascal	513	15.39
FCN16s-Pascal	514.3	15.31
FCN32s-Pascal	519.4	16.16
FCN8s-SiftFlow	514	15.65
Horizon-DCSI	0.0022	20.45
<b>Proposed (<math>7 \times 7</math>)</b>	0.057	10.59

mobile and resource constrained devices. Our approach provides a good trade-off between performance and computations, as it outperforms non-learning skyline detection methods while comparable to supervised and deep learning based methods. We have provided a quantitative comparison of our proposed approach against earlier relevant skyline detection methods using four publicly available data sets and established performance metrics. We conducted experiments to study the affect of training data and filter size. Further we provided a resource comparison (in terms of memory foot-print and inference-time) of our approach against existing ones. In future work, we would investigate to improve the performance of our approach while maintaining the same computational foot-print. To this end, we would like to understand reasons behind the failure cases in addition to ones identified above. In our base experiments, the performance of the proposed approach on CH1 data set is rather poor which is related to less versatility of the training set (i.e., Web and Basalt data sets). This would be another dimension of our exploration where we aim to remedy the failure rate by generating ground truth for CH2 data set. Additionally we intend to isolate samples from GeoPose3K where humans can annotate skylines without additional information so that methods designed specifically for skyline detection can be correctly evaluated.

#### ACKNOWLEDGEMENTS

This work was supported by project no. LTAIZ19004 Deep-Learning Approach to Topographical Image Analysis; by the Ministry of Education, Youth and Sports of the Czech Republic within the activity INTER-EXCELENCE (LT), subactivity INTER-ACTION (LTA), ID: SMSM2019LTAIZ. Computational resources were partly supplied by the project e-Infrastruktura CZ (e-INFRA LM2018140) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

#### REFERENCES

- [1] 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 (CRV)*, pp. 346–352, 2012.
- [2] 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 (ICRA)*, 2005.
- [3] S. Thurrowgood, D. Soccol, R. J. D. Moore, D. Bland, and M. V. Srinivasan, "A vision based system for attitude estimation of uavs," in *IEEE International Conference on Intelligent Robots and Systems*, pp. 5725–5730, 2009.



- [4] B. Grelsson, M. Felsberg, and F. Isaksson, "Highly accurate attitude estimation via horizon detection," *Journal of Field Robotics*, 2015. 1
- [5] J. Hou and B. Li, "An improved algorithm for horizon detection based on ostu," in *Proceedings of International Conference on Intelligent Human-Machine Systems and Cybernetics*, pp. 414–417, 2015. 1
- [6] L. Di, T. Fromm, and Y. Chen, "A data fusion system for attitude estimation of low-cost miniature uavs," *Journal of Intelligent & Robotic Systems*, vol. 65(1), pp. 621–635, 2012. 1
- [7] G. C. H. E. d. Croon, B. D. W. Remes, C. D. Wagter, and R. Ruijsink, "Sky segmentation approach to obstacle avoidance," in *IEEE Aerospace Conference*, pp. 1–16, 2011. 1
- [8] 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 International Conference on Intelligent Robots and Systems (IEEE/RSJ)*, 2002. 1
- [9] 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 (ICRA)*, 2003. 1
- [10] E. Boukas, A. Gasteratos, and G. Visentin, "Localization of planetary exploration rovers with orbital imaging: a survey of approaches," in *International Conference on Robotics and Automation Workshops (ICRAW)*, 2014. 1
- [11] F. Cozman and E. Krotkov, "Automatic mountain detection and pose estimation for teleoperation of lunar rovers," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, pp. 2452–2457, 1997. 1
- [12] F. Cozman, E. Krotkov, and C. Guestrin, "Outdoor visual position estimation for planetary rovers," *Autonomous Robots*, vol. 9(2), pp. 135–150, 2000. 1
- [13] V. Gupta and S. Brennan, "Terrain based vehicle orientation estimation combining vision and inertial measurements," *Journal of Field Robotics*, vol. 25(3), pp. 181–202, 2008. 1
- [14] N. Ho and P. Chakravarty, "Localization on freeways using the horizon line signature," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2014. 1
- [15] S. J. Dumble and P. W. Gibbens, "Horizon profile detection for attitude determination," *Journal of Intelligent & Robotic Systems*, vol. 68(3), pp. 339–357, 2012. 1
- [16] S. J. Dumble and P. W. Gibbens, "Efficient terrain-aided visual horizon based attitude estimation and localization," *Journal of Intelligent & Robotic Systems*, vol. 78(2), pp. 205–221, 2015. 1
- [17] 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. 1
- [18] L. Baboud, M. Čadík, 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. 1
- [19] L. Porzi, S. R. Buló, 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. 1
- [20] J. Brejcha and M. Čadík, "Camera orientation estimation in natural scenes using semantic cues," in *2018 International Conference on 3D Vision (3DV)*, pp. 208–217, 2018. 1
- [21] Y. Chen, G. Qian, K. Gunda, H. Gupta, and K. Shafique, "Camera geolocation from mountain images," in *Proceedings of 18th International Conference on Information Fusion*, pp. 1587–1596, 2015. 1
- [22] O. Saurer, G. Baatz, K. Köser, L. Ladický, and M. Pollefeys, "Image based geo-localization in the alps," *International Journal of Computer Vision (IJCV)*, vol. 116, no. 3, p. 213–225, 2016. 1, 6
- [23] G. Baatz, O. Saurer, K. Köser, and M. Pollefeys, "Large scale visual geo-localization of images in mountainous terrain," in *European Conference on Computer Vision (ECCV)*, 2012. 1, 6, 7
- [24] W. Liu and C. Su, "Automatic peak recognition for mountain images," in *Advanced Technologies, Embedded and Multimedia for Human-centric Computing*, 2014. 1
- [25] E. Tzeng, A. Zhai, M. Clements, R. Townshend, and A. Zakhori, "User-driven geolocation of untagged desert imagery using digital elevation models," in *Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2013. 1
- [26] S. Fefilat'ev, V. Smarodzinava, L. O. Hall, and D. B. Goldgof, "Horizon detection using machine learning techniques," in *International Conference on Machine Learning and Applications (ICMLA)*, pp. 17–21, 2006. 1
- [27] E. Gershikov, T. Libe, and S. Kosolapov, "Horizon line detection in marine images: Which method to choose?," *International Journal on Advances in Intelligent Systems*, vol. 6(1,2), 2013. 1
- [28] W. Kruger and Z. Orlov, "Robust layer-based boat detection and multi-target-tracking in maritime environments," in *Proceedings of International Waterside Security Conference*, 2010. 1
- [29] X. Kong, L. Liu, Y. Qian, and M. Cui, "Automatic detection of sea-sky horizon line and small targets in maritime infrared imagery," *Infrared Physics & Technology*, vol. 76, pp. 185–199, 2016. 1
- [30] T. Ahmad, P. Campr, M. Čadík, and G. Bebis, "Comparison of semantic segmentation approaches for horizon/sky line detection," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017. 1, 2, 6, 7, 8
- [31] T. Ahmad, G. Bebis, E. Regentova, and A. Nefian, "A machine learning approach to horizon line detection using local features," in *9th International Symposium on Visual Computing (ISVC)*, 2013. 1, 2
- [32] 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 *10th International Symposium on Visual Computing (ISVC)*, 2014. 1, 2, 4
- [33] T. Ahmad, G. Bebis, M. Nicolescu, A. Nefian, and T. Fong, "An edge-less approach to horizon line detection," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 2015. 1, 2, 3, 4, 8
- [34] 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*, vol. 24, no. 4, pp. 1–19, 2015. 1, 2, 4
- [35] T. Ahmad, G. Bebis, M. Nicolescu, A. Nefian, and T. Fong, "Fusion of edge-less and edge-based approaches for horizon line detection," in *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, 2015. 1, 2, 3
- [36] T. Ahmad, G. Bebis, M. Nicolescu, A. Nefian, and T. Fong, "Horizon line detection using supervised learning and edge cues," *Computer Vision and Image Understanding*, vol. 191, 2020. 1, 2, 3, 4
- [37] Y. Hung, C. Su, Y. Chang, J. Chang, and H. Tyan, "Skyline localization for mountain images," in *Proceedings of International Conference on Multimedia and Expo (ICME)*, 2013. 1
- [38] W. Lie, T. C.-I. Lin, T. Lin, and K.-S. Hung, "A robust dynamic programming algorithm to extract skyline in images for navigation," *Pattern Recognition Letters*, vol. 26(2), pp. 221–230, 2005. 1, 2, 4
- [39] L. Porzi, S. R. Buló, and E. Ricci, "A deeply-supervised deconvolutional network for horizon line detection," in *ACM Conference on Multimedia*, 2016. 1
- [40] R. Verbeekas and A. Whitehead, "Sky and ground detection using convolutional neural networks," in *International Conference on Machine Vision and Machine Learning (MVML)*, 2014. 1
- [41] P. Getreuer, I. Garcia-Dorado, J. Isidoro, S. Choi, F. Ong, and P. Milanfar, "Blade: Filter learning for general purpose computational photography," in *2018 IEEE International Conference on Computational Photography (ICCP)*, 2018. 1, 3, 4, 7
- [42] Y. Romano, J. Isidoro, and P. Milanfar, "Raisr: Rapid and accurate image super resolution," *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 110–125, 2017. 3, 4
- [43] S. Choi, J. Isidoro, P. Getreuer, and P. Milanfar, "Blade: Filter learning for general purpose computational photography," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018. 3, 4, 7
- [44] I. Garcia-Dorado, P. Getreuer, B. Wronski, and P. Milanfar, "Image stylization: From predefined to personalized," *IET Journal of Computer Vision*, 2020. 3, 4
- [45] A. Nefian, X. Bouysseounouse, L. Edwards, T. Kim, E. Hand, J. Rhizor, M. Deans, G. Bebis, and T. Fong, "Planetary rover localization within orbital maps," in *2014 IEEE International Conference on Image Processing (ICIP)*, 2014. 5
- [46] J. Brejcha and M. Čadík, "GeoPose3K: Mountain landscape dataset for camera pose estimation in outdoor environments," *Image and Vision Computing*, vol. 66, no. October 2017, p. 1–14, 2017. 6, 7