Sky Pixel (Region) Detection and Segmentation Using a Computer Vision Algorithm With Python and OpenCV

Saurabh Varughese M. Kovoor

School of Science and Technology,

Sunway University,

Selangor,

Malaysia

20017604@imail.sunway.edu.my

Abstract — This paper presents the development of a computer vision program used to automatically extract or segment the sky region, separating it from the of the image, for a dataset of outdoor images from the few selected SkyFinder dataset. The proposed approach is based on finding the optimal sky-ground border line based on the image's gradient information and edges obtained using the Sobel operators and the maximum energy or rate change of intensity values. From there, the optimal horizon line can be used to identify and segment the sky and ground regions of the image. Afterwards, a binary mask of each of the segmented or processed dataset images will be exported in the PNG format and stored clearly in its appropriate folders for the dataset considered (1093, 4795, 8438 and 10870) and the accuracy of the segmentation will be measured by comparing the mask with the ground truth image (mask) based on intersection-over-union (IOU) and recorded in the command line along with computed average segmentation accuracy for all the processed images. Hence, the Python programming language and the OpenCV library were used for the development of the program. Another main Python library or package that was used for the main algorithm of image segmentation was NumPy. This paper starts by providing an overview or introduction to the project, explaining the problem statement, project's purpose, and the differences between the proposed approach and the status quo or existing algorithms. Then, it will proceed to present an overview of the literature review conducted to understand the problem domain and the existing solutions to this problem before progressing to the methodology employed and the proposed system's design and finally the presentation of its results and discussion.

Keywords— *OpenCV, Python, sky region detection and segmentation, Intersection over Union (IOU)*

I. INTRODUCTION

A. Problem Statement

For various real-world applications involving outdoor photos and video frames, the sky is an important subject matter that is to be identified and extracted for further processing purposes, including for classification problems and content-based image manipulation. This is owing to the fact that the sky relays significant information about the outdoor and surrounding environment. From the literature review performed, it is evident that this step of sky detection is a crucial part of the preprocessing step for various essential outdoor-based computer vision and image processing applications whether it is for outdoor image interpretation problems, such as horizon estimation, robot navigation, image geolocalization,

acquisition and understanding, weather forecasting, solar exposure prediction, among others [7].

Additionally another problem domain where sky detection is essential is its role as a background detector for generating 3D depth maps. Hence, after the sky region has been recognised in a certain image or video frame, information and features can be extracted from it, such as for deducing weather and illumination conditions of the environment. Additionally, this essential knowledge regarding the sky condition, helps in constraining the search domain for implementing image processing and recognition of higher complexity, such as for obstacle detection or path planning systems in autonomous or unmanned aerial or surface vehicles.

These are the descriptions of certain areas and problem domains regarding the sky that increase the demand for improved sky detection and segmentation algorithms with increased speed, robustness or applicability to various images, and accuracy.

B. Goals and Objectives

Therefore, the main goal of this project is to develop a program or algorithm that automatically segments the sky region or detects and classifies the sky pixels in an outdoor image or video frame, separating it from the non-sky regions in the rest of the image. These outdoor images are retrieved from four Skyfinder dataset (1093, 4795, 8438 and 10870). Then, after processing each image, a binary mask is generated with the region of interest (pixel with intensity value 1 or 255) being the sky region and the non sky (pixel with intensity value of 0) for each image and stored in the results folder and subsequently within the datasets name folder, with a standard naming convention. Simultaneously, the accuracy of processing each image in the sample will be calculated by comparing it with its ground truth mask image and the overall average accuracy of the algorithm is also calculated based on IOU.

Some sub-goals or objectives that are striverd to be achieved through this project include developing a sky detection and segmentation algorithm that is:

- More accurate, resulting in lesser misclassifications and pixels are correctly identified and classified as sky or non-sky.
- More rapid, allowing for more number of images or video frames to be analysed in a shorter amount of time.
- More robust, allowing the proposed methodology to function on various images with different weather conditions and simultaneously be able to identify all the boundaries of the sky regions in the image and differentiate them from the rest of the image while involving minimal user intervention.

Therefore, this report will consist of three sections, an overview of the literature review performed to understand the existing systems and methods proposed, explaining the methodology of our proposed approach, as well as the experimental outcome and discussion of the results.

C. Differences Between the Proposed Algorithm and Existing Algorithms

Applicability and Robustness to Handle Bright and Dark Sky Images. Most of the current sky segmentation systems experience difficulty in handling dark sky images or images with unfavourable conditions. These algorithms are typically simple and low complexity approaches which aim to identify the horizon line in the image through an edge detection method and subsequently identify the sky and non-sky regions. For instance, the algorithm proposed by Shen and Wang [9] checks if the image contains dark sky or no sky regions, instead focuses on processing bright sky images.

Therefore, as a modification to that, our proposed approach still handles dark sky images, although it does not process it directly. We first assume that images from a dataset folder are taken of the same subject, location and positioning, but possibly at different times of the day or conditions of the surrounding (variations in illumination, sky colour, weather, cloudiness). Then, when processing the images from the dataset, we first check if it contains a dark or bright sky. If it contains a bright sky image, the processing and segmentation will proceed as

usual. However, for a dark sky image, it will be stored in an array list which will be handled once all the bright sky images are processed. Afterwards, we use the most accurate binary mask obtained from segmenting a bright sky image, and apply that on the skipped dark sky image. Doing this ensures that the dark sky images are still handled and to ensure the program performance and accuracy remains high and optimal. These dark sky images would otherwise require a more complex approach to process, which is both time and resource intensive.

Sky Region Segmentation Based on Detecting and Finding the Horizon Line. In contrast to a majority of the existing systems that utilise a pixel-wise classification technique using machine learning or deep learning classifiers, the proposed approach in this project is through finding the optimal horizon line that most effectively separates the in contrast to ML classification of individual pixels. The benefits, disadvantages, and differences between both these approaches are explained further in the literature review section.

To put it simply, the approach proposed in this project has a low complexity as it does not utilise ML models and classifiers, can produce results rapidly with an average processing time of 1-1.5 seconds per outdoor image, and does not require the preparation of a training data set or training time to build the model as the sky segmentation process is performed based on the information and features found within the image without using priori knowledge or heuristics.

Region of Interest is the Sky Region. Our project's problem domain is concerned with identifying and segmenting the sky region in an image as opposed to some systems, such as Shen and Wang's [9] approach which aims to find the non-sky or ground region for their problem domain of the development of sensing and understanding of the surrounding environment for autonomous outdoor ground robot navigation systems. Thus our algorithm is implemented in such a way that the sky portion of the image is the region of interest, and this is displayed in the resultant binary mask where the sky region is labelled with 255 and the non-sky region is labelled with 0.

Uses Intersection-Over-Union (IOU) and Program Execution Time Metric for Evaluation. In regards to the evaluation metrics to test the performance of the system, the proposed approach presented in this paper was tested in terms of the IOU accuracy metric, which was not seen employed from reviewing the existing literatures and systems. This metric is essentially a common evaluation metric, to assess or measure the performance of an object detection or segmentation algorithm, for this case, a sky-region segmentation program. It essentially measures the overlap or similarity between the resultant mask with the ground truth to determine the accuracy of the segmentation. The manner in which this evaluation metric was implemented will be explained in the methodology section.

Another metric that is prioritised is the program execution time, in that the proposed design and implementation should have a low execution time for processing and segmenting the sky regions in each individual image. By having a low execution this improves its applicability for real-time time and resource-intensive application, such as in embedded systems.

II. LITERATURE REVIEW

The following section will present a summary of the in-depth reviews of some of the existing algorithms and papers regarding sky detection/segmentation and their general workflow towards solving this problem as well as the evaluation metrics used to measure their performance, as was thoroughly described in the first comparative literature review paper. Firstly, a summary of an additional paper which was not covered in the initial paper will be presented, as it is a significant existing algorithm which inspired the design and methodology of our proposed algorithm. After that, an overview or summary will be provided of the review conducted on systems, algorithms, and literature sources related to this project topic. Most of these were more deeply explained and explored in the previous literature review comparative study paper, so one may check that for more in-depth information.

A. Sky Region Detection in a Single Image for Autonomous Ground Robot Navigation [9]

This study by Shen and Wang [9] presents their proposed algorithm of sky detection and segmentation contained within a singular image with its gradient information and using the energy function optimization. One special aspect of their proposed model is its applicability to both colour and grayscale images. The main purpose of their algorithm is for the problem domain of the development of sensing and understanding of the surrounding environment for autonomous outdoor ground robot navigation systems. Based on their literature review performed, they understood the importance of detecting colour information and differences in many existing sky region detection techniques.

First they convert the colour image into a grayscale image, then using the Sobel operator, the gradient image is computed. The Sobel operator consists of two operators, one in the horizontal and vertical directions, which will be convolved on the grayscale image, to return two gradient images, which will be combined to form a gradient magnitude image that is applied to an energy optimisation function.

The authors have designed this energy optimisation function, J_n, based on their review of several literature sources, and considering that the originally obtained colour image can only be divided into two outcomes, sky and ground region and the pixels in both regions comprise of their RGB components. The energy optimisation function, J_n, and can be denoted as the following:

$$J_n = \frac{1}{\gamma \cdot |\Sigma s| + |\Sigma g| + \gamma \cdot |\lambda_1^s| + |\lambda_1^g|} \quad (1)$$

Here, in function (1), Σs and Σg represent the covariance matrices of the pixels consisting of RGB values for sky and ground regions respectively. So, these two functions can be denoted as the following:

$$\Sigma s = \frac{1}{N_s} \sum_{(y,x) \in sky} (I^s(y,x) - \mu^s)(I^s(y,x) - \mu^s)^T \quad (2$$

$$\Sigma g = \frac{1}{N_g} \sum_{(y,x) \in ground} (I^g(y,x - \mu^g)(I^g(y,x) - \mu^g)^T \quad (3)$$

In functions (2) and (3) Ns and Ng represent the total number of pixels in the sky and ground regions of the image, respectively, where as μ^s and μ^g represent the average RGB values in those two regions, respectively. From function (1), λ_i^s and λ_i^g denotes the Eigen values, where i can be equivalent to 1, 2, or 3, and it is for the aforementioned matrices whereas the |·| denotes the determinant, measuring the pixel distribution's variance or volume for each of the two regions. In equation 1 the sum of the Eigen values is considered to handle images or video frames that lose colour information, allowing the determinants for the two matrices to become singular. Additionally, γ is used for considering the homogeneity of the sky region and its value is set to 2. For function (1), the goal is to maximise or optimise it to minimise the intra-class variance between ground and sky pixel distributions, allowing for a more significant distinction and optimal segmentation result.

Hence, for the next step, they propose a parameterisation technique to select the parameters that will optimise the energy function. This starts by defining their sky border position function b(x), which signifies the separation line between the two regions where W and H are the width and height of the input image.

$$sky = \{(x,y) \mid 1 \le x \le W, 1 \le y \le b(x)\} \quad (4)$$

$$ground = \{(x,y) \mid 1 \le x \le W, b(x) < y \le H\} \quad (5)$$

A threshold parameter, t, is used to compute this sky border position function, b(x), allowing for the sky and ground regions to be calculated following the above functions 4 and 5. Therefore a function should be implemented for computing this value.

After that, the next function is to calculate the optimal sky border position function using the previous threshold value. Hence, after obtaining the threshold t, an algorithm can be used to determine the b(x). The combination of equations (4) and (5) allows for the sky and ground segmentation results based on the t value to be calculated and subsequently Jn(t) to be estimated. Hence, another algorithm can be used for determining the energy optimisation function

which uses a singular t parameter, that can be calculated with the following function:

$$t = thresh\_min + \frac{threshmax - threshmin}{n-1} x\,(k - 1) \quad (6)$$

Therefore, for this threshold function there are three parameters to determine the threshmax, threshmin, and n values. Based on their experimental study and from following the definition of the Sobel operator, they designate the threshmax value to 600, threshmin value to 5 and the n or sampling search step value to 5 as a balance between the search precision and computation complexity of the program. That concludes, the author's main preliminary sky region segmentation approach.

Following that they perform post-processing or refinement to the segmented image where they start by identifying if the image contains no sky regions and to avoid sky region detection and segmentation in that case. Simply put, this is done through defining a border-ave function with a predefined threshold, where if the detected sky region in an image only occupies a minute section of the image or lesser than the threshold, then, this image is considered not containing a sky region. After that, for the processing, refinement of the segmentation is performed by recalculating the sky border position function, b(x), from before. This is done through calculating the Mahalanobis distance between the pixels in the original sky region and the refined sky and ground region clusters' centre.

To observe the performance of their proposed algorithm which was implemented in C++, they applied it to a dataset of 680 colour images. Then for quantitative evaluation, as the ground truth image, they created separate corresponding images where the sky region is segmented by manually drawing or labelling the horizon lines. The quantitative evaluation or evaluation metrics they used were calculating the common sky and ground region pixel sets using functions (7) and (8), and then determining the precision of the sky and ground region detection, Ps and Pg, and the whole image segmentation precision, Pt using functions (9), (10) and (11).

$$sky_{common} = \{(x,y) |\ (x,y) \in sky \wedge (x,y) \in sky_{bench} \wedge 1 \leq x \leq W \wedge 1 \leq y \leq H\} \quad (7)$$

$$ground_{common} = \{(x,y) |\ (x,y) \in ground \wedge (x,y) \in ground_{bench} \wedge 1 \leq x \leq W \wedge 1 \leq y \leq H\} \quad (8)$$

$$Ps = |\frac{skycommon}{skybench}| \times 100\% \quad (9)$$

$$Pg = |\frac{groundcommon}{groundbench}| \times 100\% \quad (10)$$

$$Pt = \frac{Ps \cdot |skybench| + Pg \cdot |groundbench|}{|skybench| + |groundbench|} \times 100\% \quad (11)$$

From the authors' results, they find that their average precision rate is 95% when applied on the test set and the processing time is approximately 150ms for a single 640x480 image on a laptop with a single core. They found that their proposed system's performance was superior to the existing systems they were comparing against and it achieved. Nonetheless, the system possesses limitations in dealing with complex images or dark sky images and it is only fitted for the purpose of robot navigation where the sky region has smooth borders and is evident.

To conclude this paper inspired our approach for detecting the sky region based using the image's gradient/edges and energy or rate change of intensity value information. So, some information from this journal article was useful for our implementation, such as the development of the energy optimisation and threshold determination functions.

A. Overview of Existing Systems and Literature Review

1. Challenges, Criteria, and Assumptions

From reviewing the existing systems and the literature sources, it was made evident the challenges that exist in this problem domain and certain criteria or assumptions that have to be prioritised or maintained during the program's operations. Firstly, the shape of the horizon line can affect the segmentation accuracy where many ground objects can alter the shape of a straight horizon line to a non-linear line that is harder to detect and. Additionally, some foreground objects may split the

sky into disjointed parts, making it difficult to identify and segment small minuscule sky section separately [3]. Secondly, the illumination and lighting of the environment affects the accuracy where in a brighter image the sky and non-sky are easier to segment and conversely for darker images. Thirdly, similar to before, the colour of the sky and non-sky regions and in conjunction the weather, affect the detection accuracy, where grey skies with less contrast and colour are more difficult to segment than blue skies with colour and contrast. Conclusively, the designed program should be robust enough to accurately process images of different conditions in terms of horizon line shape, illumination, and sky colour. Another common assumption that is made for most designed programs is that the camera is upright and the sky region is most likely at the top part of the image and the non-sky at the lower part.

2. Types of Sky Detection and Segmentation Algorithms

Next, from reviewing the current algorithms and proposed solutions, there are typically two main categories of sky segmentation algorithms.

One is the detection of the horizon line in the image that separates the sky and non-sky region, and subsequently using that to determine and distinguish the two regions. This horizon line is usually identified using an edge detection technique, such as Canny edge detection, Sobel, Roberts Cross, Prewitt, Laplace etc., and then followed by a post-processing of the edges to improve segmentation accuracy. This separating line or edge is identified through identifying or analysing a fast colour, illuminance, contrast or gradient change or energy in the image. Hence, energy in this case is the rate change of a certain feature in the image whether it is the colour, illuminance, contrast and gradient, and the location with the highest energy is most likely the sky-ground border. Additionally, with the earlier assumption of the top part of the image most likely being the sky region, the areas above this border are considered sky and below are considered non-sky. This approach is performed by Shen and Wang [9] and our designed system uses this approach as well. Its benefit is that it has low complexity as no machine learning classifier or prior training is required and the image is processed based on the features present in the image rather than classifying every individual pixel. Thus, this methodology possesses lesser computational requirements as well and can run on any normal computer and so it can be used for real-world applications and in embedded systems that possess low computing resources. However, the downside is that if the sky and non-sky regions appear similarly, it is difficult to make clear distinctions such as in a dark sky image or cloudy and smog-filled image.

The second form of sky segmentation found is a pixel-wise classification where every individual pixel in the image is analysed and determined based on its features (colour, area and illuminance) and neighbouring pixels and then classified to either belong in the sky or non-sky region. Thus, this scene or superpixel-level matching uses machine learning (ML) and deep learning (DL) models to classify every individual pixel. A benefit of this method is its robustness to apply to various different images and conditions and its performance in classifying the sky and non-sky regions based on the trained ML classifier. However, a disadvantage to this approach is that a dataset has to be prepared to build the ML classifiers using the training and testing datasets. So, a significant amount of training time is required and a dataset of images and their corresponding ground truths. Additionally, this ML method has a risk of being overfitted to the training dataset, affecting its generalisation and applicability and robustness to segment and process a wide range of images and environmental conditions. Hence, when unfavourable conditions are applied the distinction between the regions are more obscure and the algorithm may fall short in terms of accuracy and misclassification rate (MCR) tends to increase [1].

3. Summary of Literature Reviews

This next section summarises the approach proposed by different literature sources and journal articles that were reviewed.

Sravanthi and Sarma [6] utilised a conventional method to the sky segmentation problem using a canny-based edge detection that identifies the horizon line, followed by Hough transform to isolate the desired features. Along with that they proposed

another approach that utilises the K-means clustering ML model to process the HSV colour channels to classify groups in the image based on a cluster's centre point, followed by seed-based region growing using the fast marching method. After that, post-processing is performed using morphological operations and the sky region is extracted by tracing the boundaries to a binary mask.

Wang et. al. [1] proposed a method based on the hybrid probability model, that utilises gradient, colour, and location information to classify individual pixels as part of the sky or non-sky regions. Hence, their methodology falls under the pixel-wise classification method.

Subsequently, the methodology proposed by Laungrungthip et. al. [8] aims to find the sky region by extracting the colour channels of the image, applying Canny edge detection algorithm to find the edges and boundary lines, applying morphological closing algorithm to close the gaps in the edge lines, and finally using the FloodFill algorithm to identify pixels in every sky regions. Hence, it uses the colour and brightness information of the image for the segmentation. Thus, it is clear to see that the overall workflow of the method is simple and falls under the detection of horizon line category.

Khan [2] presented a methodology that is a simple pixel-based classification that utilises offline trained ML classifiers. So, they compared the performances of the Random Forests, Multi-layer Perceptron, Radial Basis Function and the Bayesian Network classifiers to determine the best performing model in detecting and segmenting the sky region. The feature that they prioritise for their ML models is only the sky colour, so by limiting the number of features they aim to increase the processing speed of this pixel based algorithm.

Yazdanpanah et. al. [4] proposed an approach that combines the results of two ML classifiers, namely, the K-means clustering followed by the Neural Network (NN) classification. Therefore, this hybrid ML model approach is seen as a pixel-wise classification of individual pixels, whether they are part of the sky or non-sky region.

Song et al [5] presented their unique approach for sky detection and segmentation in adverse weather conditions (e.g., haze, fog, and mist) by probing the density of the haze. Therefore, they used several characteristics or haze-specific features from the image to segment the sky region that reflect the perceptual hazy density and scene depth. Then, they used these features for using two support vector machine (SVM) classifiers and a similarity measurement.

Finally, the previously explained methodology by Shen and Wang [9] detect and segment the sky region in a singular image based on its gradient information retrieved through combining the edges obtained from the sobel operators, and using the energy function optimization to determine the optimal border separating the sky and non-sky regions. Hence, their approach falls under the category of horizon line detection. However, for their problem domain they are interested in finding the ground or non-sky region, so that is their region of interest rather than finding the sky region as in our objective.

III. METHODOLOGY AND PROPOSED APPROACH

A. Introduction

The program which was designed produces two categories of output, one being the resultant binary mask from processing and segmenting the input outdoor image in a PNG format and the performance metrics of the segmentation processing, which include the accuracy rate (intersection over union, IOU) and the execution time. Overall the initial goal is for the program to segment the sky region in outdoor images with a minimum overall accuracy of 85%. These performance metrics, results and outcome of the implementation will be explained further in the following section, Results and Discussion.

The program consists of 1 Python (.py) file, sky_segmentation.py that acts on a folder of images (Images). Within this Images folder, there will be more folders, which are images from the same dataset or camera of the Skyfinder dataset. For now, there are

four dataset of images that is used to study the performance of the program, 1093, 4795, 8438 and 10870. Hence, one assumption we maintained for the development is that images within a folder or dataset, taken by the same camera, will be of the same subject, location and positioning, but possibly at different times of the day or conditions of the surrounding (variations in illumination, sky colour, weather, cloudiness). These folders contain between 600-1000 images each, so a subset of images (e.g., 20, 50, 100 images, etc.) will be randomly sampled from each dataset folder and used for the execution of the program to ensure the program does not require an extensive execution time to produce a result. This number of samples can be chosen by the user after executing the file and entering the positive integer at the prompt. Ideally, to avoid errors, enter a number not less than 8. After executing the sky_segmentation.py file, a Results folder will be generated in the same directory, containing folders signifying the dataset name, and within that will hold the resultant binary mask (PNG format) for each of the processed and segmented images, with the sky region being the region of interest in the binary mask. Simultaneously, the performance metric results of the processing will be outputted in the Python terminal, including the accuracy values and the program execution time.

Overall, the sky region detection and segmentation approach that is proposed in this paper, is built on the basis of separating the sky and land regions, through finding gradient information of the image and subsequently the horizon line that possesses the highest energy change. This is identified through finding the most prominent edge or one which possesses the highest energy or most rapid change in intensity. Edges are essentially part of the image where there is a rapid change of the image intensity, caused by a change in illumination and colour within an image. Hence, the energy in this case represents a maximisation problem, where the position with the highest energy or maximum or most rapid change in intensity value, is aimed to be identified. Thus, this point in which there is a rapid change or high energy could signify the border between the sky and land regions. As mentioned in the literature review earlier, it is based on and inspired by the approach proposed by Shen and Wang [9]. The implementation of our approach to segmenting the sky regions through processing the gradient and energy information of the image will be further explained in section 3-6 of the following section.

B. Contents of sky_segmentation.py and Methodology

1. Creating Results Directories and Looping Through Input Images' Folders

The program will then create the Results directory to store the programs outputs, the aforementioned resultant binary mask. This is done using the makedirs method of the os library and error handling is applied simultaneously. So, the file structure consists of the Results directory in the same directory as this code, and underneath that is the respective folders to hold each of the folders for each dataset which holds the resultant binary masks.

The sampleSize attribute signifies the number of images per folder/dataset to sample for the program execution or to be segmented and processed. Hence, user input is accepted for this value, so they can choose the number of images they wish to sample from each dataset by entering the value at the prompt. So, one can reduce this for faster execution time or increase for testing more images. Ideally to prevent errors, this should be set to at least 8. After that, before starting the main program, the tracking of the execution time begins by recording the time at start time (time.time()). This will be used for calculating the total duration for processing the sampleSize number of images.

Two nested for loops are created, where the first iterates over the dataset folders (folderlist) in the Images directory. Then, a nested for loop within the previous, will iterate over the images within the directories (imagelist). This will allow for processing of specific images and folder/dataset to be conducted at every cycle. For each cycle of the first for loop, a folder within the Results directory for the dataset is created, the image names are sampled from the dataset based on the sample size set, the skippedDarkImages and imageAccPairs list is initialised.

2. Checking if Day or Night Image, brightOrDark()

For the second for-loop it iterates over every image within a dataset folder. So, it starts by reading the image as in the randomly generated sample list, imagelist, from the dataset, and reading its corresponding ground truth mask from the Masks folder within the same directory as the program. Then, the image is checked if it contains a bright or dark sky using the user-defined brightOrDark function, which is a modification that was added to ensure the program will only process and segment bright sky images, instead of dark sky images which will produce poor results if it was to be directly processed using this algorithm. Hence, if the algorithm was to be directly applied on the dark sky image, the performance would bring down the program's overall accuracy rate and performance metrics.

Instead, a modification was performed to ensure that dark sky images are segmented effectively, with the assumption that images within a dataset are from the same camera, of the same subject and positioning, we utilise the best performing binary mask obtained from segmenting a bright sky image, and apply that on the dark sky image. This improves the performance recorded for the algorithm and allows dark sky images to also be considered, which would otherwise require a high complexity program that may need significantly more execution time to process these dark sky images. Thus, this system's low execution time and high performance can be beneficial for its applicability in real-world, real-time and time-sensitive applications.

The brightOrDark function, essentially applies median blur on the input image, and checks to see if the image has an average intensity value above the threshold, 90. If so, the image will be considered a bright sky image, else it is considered a dark sky image. The median blur separates the two extremes of images average intensity, as the score/average intensity for dark images are lowered more significantly than for bright images. The threshold was determined by a separate algorithm that was designed which processed all the images within a dataset or from the same camera, and plotted the average intensity value on a histogram to obtain the valley point, which would separate the bright and dark images. So, from each of the histograms, it was observed that performing the median blur produced significant distinction between the two extremes of images. Thus, from the four histograms (for four datasets), it can be seen that the common valley point or threshold that effectively split the two types of images was 90, hence that was chosen as the threshold for this function.

Therefore, after determining the condition of the image, for bright sky images, the normal operation of sky detection and segmentation is employed whereas for dark sky images, it is not segmented or processed. Instead, it will be appended to a list, SkippedDarkImages, that will be handled at the end or after the bright sky images for the dataset sample is processed.

3. Converting the Input Image to Grayscale and Creating Gradient Image, returnGradientImg()

For the main algorithm that is performed on the bright sky images, it starts by performing the returnGradientImg user-defined method which is used for preprocessing the input image before subsequent processing. It converts the image to a grayscale image. Then, using two sobel operators, a horizontal and vertical one, will be used to extract the horizontal and vertical edges in the image, respectively. The two edges are combined using the pythagorean theorem function (np.hypot) to combine the two edges to form the gradient magnitude image, gradientImg. A gradient image created with the two sobel components is used for edge detection within the image as it clearly shows the edges in the image as well as the distinction and boundaries between subjects in the image, in this case of the sky and ground regions. Allowing us to easily identify and segment the two regions separately. Subsequently, in this approach, this gradient or edge information in the image along with finding the energy values or the maximum rate change of intensity values will be used to find the sky-ground border to clearly label the sky and ground regions of the image.

4. Calculating the Border Optimal Points, borderOptimisation()

After that, the gradient magnitude image and the input image is used for the borderOptimisation function, which is used to find and compute the optimal sky border position in the input image, which will be used to separate the sky and ground (non-sky) regions. It does this by first finding an appropriate threshold or parameter value which maximises the energy optimisation function to find the most optimal sky border position points. For this three parameters are required or initialised which are the minimum, maximum threshold and the sampling search step value. From reviewing Shen and Wang's journal article [9], from their experimental study, they found that threshold values exceeding 600, produced an energy optimisation function value which remained approximately constant. Therefore, this was chosen for the maximum threshold value and similarly the minimum threshold of five was chosen as that is where the graph began when extrapolated. To balance between search accuracy/precision and computation program and time complexity, a moderate search step size of 5 was chosen.

The program then determines the number of sample points to search or is in the search space based on this search step and a function utilising the max and min threshold values. These three parameters will be used for computing the optimal threshold value. The list of optimal border points and the energy function values are initialised.

Then, a for loop is declared which iterates for every sample point in the search space. Hence, at every loop, the aforementioned function used to find the threshold value, which is used to compute the optimal sky border function value, is returned. This function was implemented based on function/equation (6).

5. Returning Border Point Positions, returnBorderPosition()

The returned threshold value along with the gradient image is entered into the next user-defined function, the returnBorderPosition, which returns a list of border points signifying the points of the sky-land horizon border. The returnBorder function itself only returns the possible border points, based on the entered threshold value and input gradient image, but it does not necessarily have to return the optimal points. Only when a computed optimal threshold value parameter, will it return possible optimal border points. So, the function starts by creating and initialising a skyBorder array that is the size of the input image's length and filling it with an intensity value of the size of the width. Then, a for loop is declared, for the length of the skyBorder array. For every loop or every column in the image, using the np.argmax function, it computes and finds the index value that possesses the highest intensity value on the vertical axis of the gradient image, for that particular column, which signifies the point that is most likely part of the boundary line, given that it is more than the optimal threshold. Therefore, if the positioning is more than 0 it would be considered as part of the sky or horizon line and will be returned to the caller function, borderOptimisation.

6. Finding the Optimal Border Point Corresponding to Highest Energy Value, energyOptimisation()

Therefore, returning to the borderOptimisation function, it obtains the list of border position points, which together with the input image will be entered into the energyOptimisation or cost function which will check the energy difference at the border point, to find the point with the highest energy value/change in intensity and determine if it is indeed the optimal point in the border line. Hence, if the energy value is more than the currently found maximum energy value, the maximum value will be updated, and the border point will be recognised to be the optimal border point. How this energyOptimisation or cost function operates is, it starts by creating a general skyMask, using the entered initial border points and the input image, and is used to create a binary mask using the createMask function, where the confirmed true sky and land regions are identified. The operation and workflow of this createMask function is explained more thoroughly in the next section. Nonetheless, this sky-land border is not perfect or has optimal points to ensure that it is as close to the true border as possible.

After that, the ground and sky regions of the image are determined and a mask for each is created and converted to a 1-dimensional array. Subsequently, the energy optimisation function is declared and

implemented based on (1) which takes in information (covariance and Eigen values) from the ground and sky region. This energy optimisation function requires five parameters. First, the gamma value represents the homogeneity of the sky region and is set at 2. Then, the covariance matrices of the ground and sky region, which was obtained using the cv2.calcCovarMatrix function and then np.linalg.det to find the determinants from its array output to be used for the energy optimisation function. After that, the Eigen values for the sky and ground regions were obtained using the np.linalg.eig function and then np.linalg.det to find the determinants from its array output to be used for the energy optimisation function. After entering the five parameters into this energy optimisation function, it will return the computed energy value to its caller function borderOptimisation. Hence, as mentioned before, based on this retrieved energy value, it is used to determine whether a particular border point, is the most optimal, by comparing this energy level with the current max, and if the obtained energy level is higher, this denotes the corresponding border point, is the optimal one, and it will be returned.

Finally, after getting all the border points that are optimal in an image, denoting the optimal points of the sky-ground border that effectively split differentiate the two regions, this array will be returned to the main program which will create and draw a binary mask based on these optimal border points, that ideally resemble the image subject with the sky region and non-sky/ground region clearly identified.

7. Create the Binary Mask, createMask()

Therefore, the next step of the program is to create the binary mask where there are two regions, the sky region, which is the region of interest and has pixel 1 as well as the non-sky region with a pixel value of 0. So, in the final resultant mask, the ground and its protruding objects will appear black, whereas the sky regions will appear white, clearly showing the distinction between the two regions. This function uses the list of optimal border points and the image as the inputs to draw this binary mask separating the two regions. It creates a blank binary mask (0, 1) the size (length x width) of the input image. Then it checks the border point list, it checks its x and y coordinate, and assigns the intensity value of 255 to it and the pixels below the border point on the binary mask. Therefore, based on the obtained border points, in the initial resultant mask, the ground regions are labelled with the intensity value of 255 whereas sky regions are labelled with 0. This will be inverted in order to make the sky region the region of interest (pixel value 255) and conversely for the non-sky region, using the cv2.bitwise_not(mask) function. Finally, the mask is returned to the caller.

8. Post-Processing the Mask, postprocessing()

It is possible to directly apply the algorithm up till now on the test or dataset images. However, it was noticed that there exists gaps within the ground or non-sky region of the image. Hence, a simple post-processing is applied to the resultant mask to fill in these gaps and make it more suitable and akin to the actual image's ground region. For this, the postprocessing() method is called and the mask is inputted, then it is inverted so that the ground region is the region of interest and then applied with the morphological operation closing which dilates and erodes the mask with a 20x20 square kernel. From the experimental study, this kernel was seen to be the most performant size for it, as it resulted in a more "connected" ground region of the image and highest final segmentation accuracy.

After that, the resultant mask is saved to the Results directory, within its dataset's folder, and assigned the name of its initial image followed by "_mask" and saved in the PNG format.

9. Measuring Accuracy of the Segmentation Using Intersection-Over-Union, IOU, calculateIOU()

After obtaining the final mask through the previous sky identification and segmentation steps, For this it will be compared with its corresponding ground truth image contained in the Masks directory, where the sky and non-sky regions have already been labelled correctly. This ground truth mask was also obtained from the Skyfinder dataset.

The accuracy of the segmentation is measured using the Intersection-Over-Union, IOU method, which is a

common evaluation metric, used to assess or measure the performance of an object detection or segmentation algorithm, and for this case, a sky-region segmentation program. The way it works is with the ground truth image/mask and the segmented region (resultant final mask), it computes the ratio of the overlap and union area between the images. Hence, the IOU value can range from 0 to 1, 0 being there is no intersection or similarity between the two images and 1 where the resultant mask and the ground truth are exactly and perfectly alike. So, the goal is to achieve an IOU accuracy that is higher, which signifies a better prediction or estimation of correct sky region segmentation by the program.

Therefore, this was implemented in the program with the calculateIOU function, where it accepts the two masks, the ground truth and post-processed resultant, then computes the area or number of non-zero value pixels in the images. Then, the number of non-zero value pixels in an intersection and union between the two masks is computed. After that the IOU accuracy value is calculated by dividing the area of intersection over union. This accuracy value is multiplied by 100 to obtain its value in percentage for results presentation purposes.

10. Handling Skipped Dark Images

The next step of the program is to handle the previously skipped dark-sky images. For this, the list of the skippedDarkImages are checked whether it is empty, if not, it will proceed with the steps to handle those images, else it will skip this step. First, it will display the skipped image's names, then it will sort the contents of the imageAccPairs list which consists of the names of the processed/segmented images and their segmentation accuracies in descending order of accuracy to obtain the name and accuracy of the best performing bright sky mask. The mask is then used for each of the dark sky images, as it is saved to the Results directory and dataset folder as that image's mask and the new image accuracy pair is appended to the imageAccPairs list. It then calculates and stores the average segmentation accuracy for the folders' images to compute the total average accuracy of the program later on.

11. Displaying Program Performance Metrics

After the for loop has completed and images from all the dataset folders in the Images directory have been processed and segmented (according to the preset sample size), the results of the program are displayed. First, stops the tracking of the program execution time and calculates the duration by minusing the end time and previously recorded start time. It then displays the average segmentation accuracy for each folder of images, while calculating the total accuracies. This total accuracy is divided by the number of folders/datasets to compute the average total accuracy of the program, which signifies the average segmentation accuracy for each image and this value is displayed to the user in the Python terminal. After that, the total program execution time and the average execution time for a single image is computed and displayed.

IV. RESULTS AND DISCUSSION

For this program there are two ways in which we measure its performance or ways the experimental setup is conducted, which include measuring the program's intersection-over-union accuracy in segmenting dataset image's sky and non-sky regions successfully and the time taken for the program execution. Both of these performance metrics are essential and requirements for this program and its problem domain, where the program should possess. Hence, first, the program shall possess sufficient accuracy (initial goal, more or equal to 85%) in order to identify, label and segment sky and ground regions and pixels of the image correctly (IOU).

The total average accuracy of the program after segmenting all the images, measures how often the program is able to segment these regions in images correctly and accurately. As for the program execution time, it should be kept to a minimum (initial goal, less than 5s), in order to make it suitable and applicable to be used for real-world, real-time and time intensive applications and systems.

On that note, the following is the record of the results of evaluating the algorithm's accuracy and execution time.

A. Intersection-Over-Union (IOU) Accuracy

The following table displays the average IOU accuracy for segmenting images from each folder/dataset and all datasets calculated for when 100 images are sampled from each dataset.

Table 1

Average Intersection-Over-Union (IOU) Accuracy for 100 image sample

Dataset	Average IOU Accuracy (%)
1093	**96.4439**
4795	**73.8626**
8438	**95.6482**
10870	**94.3072**
Total Average	**90.0655**

The following table displays the average IOU accuracy for segmenting images from each folder/dataset and all datasets calculated for when 400 images are sampled from each dataset.

Table 2

Average Intersection-Over-Union (IOU) Accuracy for 400 image sample

Dataset	Average IOU Accuracy (%)
1093	**96.4528**
4795	**79.4985**
8438	**95.4293**
10870	**93.5934**
Total Average	**91.2435**

The following table displays the average IOU accuracy for segmenting images from each folder/dataset and all datasets calculated for all images from each dataset.

Table 3

Average Intersection-Over-Union (IOU) Accuracy for all images in the datasets

Dataset	Average IOU Accuracy (%)
1093 (870 images)	**97.3208**
4795 (725 images)	**77.5446**
8438 (1060 images)	**96.0027**
10870 (1790 images)	**93.3193**
Total Average	**91.0468**

From the results obtained it is clear to see that the segmentation accuracies do not vary much between when a 100, 400, and all image sample is taken as the total average is around 90-92% (90.0655% for 100 image sample, 91.2435% for 400 image sample, 91.0468% for all images in datasets). However, a smaller sample can be tested to reduce the overall program execution time and to perform quick tests or evaluations of the program's performance.

The results also show that the initial goal to achieve a minimum of 85% average accuracy has been fulfilled. From the explanation about IOU given in the methodology section, it is clear that an accuracy closer to 100% denotes that the resultant mask appears closely similar to the ground truth mask and the converse is less similar. Hence, the goal is to achieve a higher IOU value.

B. Program Execution Time

The following table displays the total and average program execution time for segmenting images of all datasets calculated for when 100 images are sampled from each dataset.

Table 4

Program execution time for 100 image sample

	Program Execution Time (s)
Total (for 100 sample images per folder)	**439.9618**
Average for each image	**1.0999**

The following table displays the total and average program execution time for segmenting images of all datasets calculated for when 400 images are sampled from each dataset.

Table 5

Program execution time for 400 image sample

	Program Execution Time (s)
Total (for 400 sample images per folder)	**2203.9621**
Average for each image	**1.3775**

The following table displays the total and average program execution time for segmenting images of all datasets calculated for when all images are sampled from each dataset.

Table 6

Program execution time for all images in the dataset

	Program Execution Time (s)
Total (for all images in the datasets, 4445 images)	**6936.4225**
Average for each image	**1.5605**

From the results obtained it is clear to see that the execution time of this approach is admirable and fulfils our initial goal, achieving an average program execution time between 1-1.6 seconds (1.0999s for 100 image sample, 1.3775s for 400 image sample, 1.5605s for all images in the datasets). It is clear to see that a larger sample requires more total program execution time. Nonetheless, as a smaller sample results, such as 100, results in a similar average accuracy, it can be opted for quick evaluations and tests of the system. This low program execution time reflects on the less time complexity of the approach and its applicability for various real-world applications that function in real-time and where program execution time is a key requirement.

C. Limitations of the Program

1. Challenge in Handling Images With Partial Sky Regions

From viewing the accuracy results, the accuracy of sky segmentation is relatively low for images from the 4795 dataset. This is because the images from that camera contain a partial sky region, and not a full length sky region across the length of the input image, from the first column to the last. Thus the horizon line is typically disjointed in these images.

The image only has a partial sky region where only a small region in the left-corner is part of the sky and the rest are objects protruding from the ground. Hence, it is difficult for the program to estimate the optimal border points to determine the sky-ground border and so there will be columnal streaks of gaps between the identified ground region.

This was found to be improved after applying morphological closing. Hence, after performing the post-processing and selection of the 20x20 filter kernel for the morphological closing operator this slightly more optimal result of (73-80%) was obtained. Hence, the addition was mainly useful for groups of images which would otherwise produce a mask with gaps in the non-sky region. Thus, morphological closing can be used to close those gaps. This was especially prevalent for the images from the 4795 dataset.

2. Inability to Handle Small Sky Regions in the Image

From viewing this approach's methodology, it essentially functions by finding the sky-ground

border with optimal border points, and as seen in the createMask function, the mask is created based on those points where pixels above the border are region of interest and labelled 255, signifying sky region detected and below are 0, signifying ground region. Thus, this prevents the program from being able to identify small regions of sky that appear after the occlusion of the sky region by foreground obstacles or objects that segment the sky into disjointed parts.

For instance, this occurs after a sky region is blocked by tree branches or leaves, leaving behind minuscule sky regions. So, these regions are not identified by the proposed segmentation algorithm. However, to handle these challenges, a more complex approach may need to be taken, such as classifying each individual pixel as a non-sky or sky pixel, such as with the use of trained machine learning classifiers. This may increase the complexity of the program and the overall execution time, which we want to minimise to allow it to apply for real-world applications. Besides that, a machine learning approach would require a substantial amount of time to train the classifiers and to prepare the training and testing datasets beforehand. Therefore, seeing as the sample dataset images and ground truth masks do not have such minuscule sky regions, this low complexity approach to identifying the optimal sky-ground border and subsequently the two regions was opted.

3. Challenge in Directly Handling Dark Sky Images

Another challenge the program encounters is that it is not able to directly process dark sky images with the segmentation algorithm and produces optimal results or an IOU accuracy that is minimum 85%. This is because in these dark sky images, there is lesser distinction or colour and intensity value difference between the sky and ground regions. Thus the sky-ground border has a lower energy value or change in intensity value that can be identified to generate optimal border points to differentiate the sky and land regions.

Therefore in order to prevent this detection and segmentation performance and accuracy affect or bring down the overall algorithm performance and accuracy, a modification was performed on the approach, where when an image is read, it checks if it contains a dark or bright sky using the brightOrDark function. Then, if the image has a dark sky it will be appended to a skippedDarkImages will be processed later and skipped for now. Once all the bright sky images are processed and the resultant masks are obtained, the best performing resultant mask, or the one with the highest IOU segmentation accuracy, will be used to extract and segment the dark sky images in the skippedDarkImages list. This ensures that dark sky images are still handled by the program, even though it is not directly processed. This is possible and valid, with the assumption that the images in a particular dataset are taken at the same location, positioning and subject and only differ in terms of time of day conditions of surrounding (variations in illumination, sky colour, weather, cloudiness).

To directly classify dark pixels in an image, a more complex method would have to be employed such as a machine learning classifier, capable of classifying each pixel as part of the sky or non-sky region based on certain features and the neighbourhood of pixels. However, as mentioned previously such an approach may increase the complexity of the system, requiring more computation time, and if it is concerned with machine learning, substantial training time and training datasets will be required. So, our approach of detecting the bright or dark sky and performing the subsequent processing, has low complexity and execution time, making it suitable for real-world applications.

REFERENCES

[1] C.-W. Wang, J.-J. Ding and P.-J. Chen, "An Efficient Sky Detection Algorithm Based on Hybrid Probability Model," Proceedings of APSIPA Annual Summit and Conference 2015, pp. 919-922, 2015.

[2] R. Khan, "Learnt Horizon Filter: A Machine Learning Approach," Sci.Int.(Lahore), vol. 29, no. 1, pp. 125-130, 2017.

[3] F. Schmitt and L. Priese, "Sky Detection in CSC-Segmented Color Images," Proceedings of the Fourth International Conference on Computer Vision Theory and

Applications, vol. 2, no. 5, pp. 101-106, 2009.

[4] A. P. Yazdanpanah, E. E. Regentova, A. K. Mandava, T. Ahmad and G. Bebis, "Sky Segmentation by Fusing Clustering with Neural Networks," 9th International Symposium on Visual Computing, pp. 27-38, 2013.

[5] Y. Song, H. Luo, J. Ma and Z. Chang, "Sky Detection in Hazy Image," Sensors, vol. 18, no. 4, pp. 1-18, 2018.

[6] R. Sravanthi and A. Sarma, "Efficient Horizon Line Detection using Clustering and Fast Marching Method" International Journal of Innovative Technology and Exploring Engineering (IJITEE), vol. 9, no. 4, pp. 2913-2918, 2020.

[7] R. P. Mihail, S. Workman, Z. Bessinger and N. Jacobs, "Sky Segmentation in the Wild: An Empirical Study," 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1-6, 2016.

[8] N. Laungrungthip, A. McKinnon, C. Churcher and K. Unsworth, "Edge-Based Detection of Sky Regions in Images for Solar," 23rd International Conference Image and Vision Computing New Zealand, pp. 1-6, 2008.

[9] Y. Shen, Q, Wang, “Sky Region Detection in a Single Image for Autonomous Ground Robot Navigation,” Int. J. Adv. Robot. Syst., vol. 10, no. 3, pp. 23-44, 2013.