Sky Region Detection using Computer Vision Algorithm

Lee Jia Qian
BSc (Hons) in Computer Science
School of Engineering and Technology
Sunway University
Selangor, Malaysia
19117613@imail.sunway.edu.my

***Abstract*— This project goal is to develop a computer vision algorithm which will automatically identify the pixels that belong to the sky region. The sky region detection system is developed using computer vision techniques which is applying Canny edge detection to obtain sky region and flood fill to create mask to draw skyline.**

Keywords*— Sky Region Detection, Computer Vision, Canny Edge Detection, Flood Fill

I. INTRODUCTION

The objective of this project is to create a computer vision algorithm capable of automatically detecting pixels that represent the sky region in outdoor images. Identifying the sky region is a crucial initial step in various applications, including weather forecasting [1], solar exposure detection [2] and ground robot navigation [3].

The algorithm is inspired by previous work and is designed to enhance the contrast between the sky and the rest of the image. It first applies the Canny edge detection algorithm to extract edges and then employs a flood fill method to obtain the mask of the sky region. A day or night detection function is utilized to determine the time of the image and if it is a night image, the algorithm enhances the image using the HSV color space.

The accuracy of the algorithm is evaluated on multiple datasets and the results demonstrate promising performance. The report also discusses the strengths and weaknesses of various existing approaches for sky region detection, highlighting the effectiveness of the proposed method in handling complex scenes and challenging weather conditions.

II. LITERATURE REVIEW

In recent times, edge detection methods have gained popularity in identifying the sky region, owing to their ability to distinguish between the sky and non-sky region. Computer vision, a scientific field that emulates human visual perception, focuses on enabling computers to comprehend visual data [4]. Machine learning and computer vision aim to empower computers to sense, comprehend and respond based on past and present data [5]. This literature review examines and evaluates current computer vision algorithms specifically designed for sky region detection in outdoor images, regardless of day or night conditions, with a particular focus on edge-detection algorithms. The goal is to determine the most effective and accurate algorithm for identifying the sky region and the subsequent section will analyze the strengths and weaknesses of different edge-detective approaches.

The existing sky region detection approaches with a specific focus on the method proposed by Jiang et al. [6]. Their approach simplifies the segmentation process by directly identifying the sky region without complex preprocessing stages, such as edge detection and binarization. The method utilizes light contribution and RGB intensities to accurately estimate the sky region by identifying the furthest pixel. While the technique shows promise in test images, there is a need for additional measures to prevent misidentification of neighboring scene pixels as part of the sky region. The approach excels in segmenting various scene types, including distant and fuzzy scenes, while effectively handling interference from nearby objects. The adaptability of thresholds and template scale allows adjustment for different scenarios, including handling plane contrails effectively. Overall, the method exhibits improved segmentation results without the need for preprocessing, as supported by experimental data.

Furthermore, Zhao et al. [7] introduce a novel approach for precise sky detection in complex settings, based on the concept of “border points”. These border points are identified in each image column using gradient and color information and they serve as the division between the sky and non-sky regions. By gathering these border points, the entire sky region can be accurately identified. The proposed approach successfully handles various sky conditions and is more reliable and computationally efficient compared to mathematical models and machine learning methods. Visual inspection is used to evaluate the extraction of sky region and the results show the effectiveness of the proposed strategy.

Moreover, another study made by Song and others [8] proposes a novel approach for reliable sky detection in adverse weather conditions. They use a multiple classifier framework to classify images into high confidence sky regions, high confidence non-sky regions and uncertain regions. Their approach involves identifying sky features using color and gradient information, employing two unbalanced SVM classifiers for sky and non-sky region and post-processing to refine the results. They introduce new criteria to enhance the accuracy of sky detection. The evaluation metrics include detection rate and misclassification rate, and the proposed approach outperforms other methods in terms of detection accuracy under various weather and illumination conditions, with an average misclassification rate of 9.63%.

In addition, a study made by Laungrungthip and others [2] proposes an edge-based detection approach to accurately identify the region in an image for solar image for solar exposure assessment. Their method involves obtaining a color plane, using Canny edge detection to locate boundaries, applying morphological closing to fill gaps in edges and employing the flood fill method to identify sky pixels within the sky region. Different regions of the sky are identified by

adjusting for varying weather conditions and enhancing the contrast between the sky and the rest of the image. The approach's parameters, such as threshold and element sizes, need to be carefully selected for reliable image processing and accurate sky detection. The paper provides the results of the sky segmentation without comparison and the output shows the discovered sky region transformed into binary form.

Lastly, Luo and Etz [10] propose a physical model-based approach for sky detection that utilizes a multilayer backpropagation neural network for color classification, region extraction and sky signature validation. The method involves three stages which are neural network color categorization, region extraction with refinement and region growth as well as physics-based sky trace model sky signature validation. By applying dynamic range adjustment (DRA) normalization and adaptive thresholding, the system effectively distinguishes the sky from other blue-colored regions. The approach successfully identifies true positive sky regions with a low false positive rate of 2% and does not rely on picture orientation data. However, it may encounter challenges in accurately identifying gloomy skies or images with polarizers. Despite these limitations, the algorithm's progressive desaturation effect ensures its independence from the actual horizon, making it an effective and accurate sky detection technique.

In conclusion, there are various approaches to sky region detection, each offering distinct methods and advantages. The first approach directly focuses on segmenting the sky region without prior processing, leading to improved segmentation performance even in images with varying background complexities. The second approach introduces the concept of borders, enabling the identification of complexities. The second approach introduces the concept of borders, enabling the identification of complex sky regions within each image column, even in the presence of dividing objects. The third approach utilizes a multiple classifier architecture and region-level characteristics to overcome challenges posed by adverse weather conditions, resulting in higher detection accuracy and lower misclassification rates compared to existing methods. The fourth approach involves extracting color channels, locating boundaries, filling gaps, and using image processing parameters to determine the sky region. Lastly, the model-based strategy demonstrates effectiveness and accuracy in sky recognition, with a low false positive rate and the potential to handle off-blue sky colors. Overall, these approaches collectively offer reliable and precise sky detection techniques applicable in various scenarios.

III. PROPOSED ALGORITHM

The proposed method for sky region detection was closely related and inspired by the work of Laungrungthip and other [2]. The method will extract blue plane from input images where it enhances the contrast between the sky and the rest of the image. This method also uses Canny edge detection, morphological closing, and flood fill.

This proposed method will read the dataset folder and the content inside to get the input images and extract the blue plane of the image. A canny edge detector is used to get edges and morphological closing operation is applied to close the gap between edges. Next, the flood fill algorithm is used to fill the connected region to obtain the mask. Then, a day or night detection function which identified the image as a day or night image, will return the day or night value and input into the figure title. Besides that, if the image was night scene, it will enhance the night images for better detection of the skyline. Next, the accuracy will be calculated for each dataset and the whole proposed system. Finally, the system will output or save the figure with the original image, predicted mask and the skyline.

A. Edge Detection Algorithm

According to Fisher and others [10], Canny edge detection algorithm operates through multiple stages to detect edges in an image. Initially, the image is smoothed using Gaussian convolution, reducing noise and preparing it for edge detection.

Then, a simple 2D first derivative operator, similar to the Robert Cross operator, is applied to the smoothed image. This step highlights regions with high first spatial derivatives, emphasizing potential edges. These edges are represented as ridges in the gradient magnitude image. The algorithm then tracks along these ridges and retains only the pixels on the ridge top, discarding others, to create a thin line as the output. This process is called non-maximal suppression.

To ensure the continuity of edges and avoid fragmenting noisy edges, the tracking process involves hysteresis controlled by two thresholds, T1 and T2, where T1 is greater than T2. Tracking begins at point on a ridge with a magnitude higher than T1 and continues in both directions until the ridge magnitude falls below T2. This hysteresis approach prevents the formation of fragmented edges from noisy regions.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

The gradient calculation is performed using the above formula in a 3 by 3 convolution mask where Gx is horizontal axis and Gy is vertical axis [11].

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

The above formula is used to find the gradient strength and direction [11].

The canny function in the proposed method performs the Canny edge detection algorithm on the input image to identify prominent edges and boundaries. The process involves extracting the blue channel from the input image. Then, the Canny edge detection is applied to the blue channel, using 16 as low threshold and 186 as the high threshold, which detect edges within the intensity range of 16 to 186 as suggested by Laungrungthip and other [2]. A 17 by 17 structural element is created using "cv2.getStructuringElement" function and is used for morphological closing operation to close small holes and smooth the edges.

B. Flood Fill

The fill hole function is using flood fill algorithm where it first obtain the height (h) and width (w) of the input image

using img.shape[:2]. Then, a mask is created which additional 2 rows and columns from the obtained height and width. The flood fill algorithm is then performed with the input image and the mask from point (100, 100) with value 255 using "cv2.floodFill()". The output is then inverted and using a 'or' operation to combine the original and with the inverted image [12]. The function will convert to RGB and invert again. Then, a dilation operation with 7 by 7 kernel and morphological closing operation with 7 by 7 rectangular structure element is applied to try removing some of the detected cloud.

C. Skyline Detection

This function takes the predicted mask as the input, which is obtained from applying edge detection and flood fill algorithms. This function uses Canny edge detection to draw the skyline from the predicted mask. Furthermore, a for loop is created to calculate the bounding box of the contour then the if statement is used to make a clean skyline by filtering some of the clouds.

D. Day or Night Function

This function is designed to determine whether the input image represents a day or night scene based on the number of white pixels present in the image. First the image will be converted to gray scale. Then Otsu thresholding method is used from converting the grayscale to binary image to automatically calculate the optimal threshold value. Next, it will count the number of white pixels in the binary image and the threshold is set to 20. If the number of white pixels is more than the set threshold it will return "Day Time" else, it will return "Night Time". This function will later be used to determine the time of the image and use to obtain the accuracy.

E. Accuracy Function

The accuracy function is used to calculate the accuracy of detecting the sky and non-sky region using mean square error (MSE).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2$$

The above is the formula of MSE where n is the total number of observations, yi is the actual value while $\hat{y}$i is the prediction value [13].

The accuracy is calculating the square different by comparing the output image and the expected output mask and square it to remove any negative value. Then, using "np.mean()" to find the mean squared difference. The accuracy score is calculated with 1 divided by 1 + MSE.

F. Night Image Enhancement Function

This function is to perform enhancement on night image using HSV color space by first converting the image from BGR to HSV color. Then, the image is split into three channels which are hue, saturation and value. The hue and saturation channel are adjusted to adjust color and decrease saturation. Moreover, the adjusted channel will then convert to uint8 and merge back together and convert back from BGR to HSV to get the enhanced image.

G. Main Function

This function is designed to process image datasets and compare the results of the proposed method with the expected ground truth. It takes three arguments which are "datasets", the name of the folder, "file", the name of the file and "expected_dic", a dictionary containing the actual mask value.

The function will first read the image from the specified dataset and a copy of original image as well as the actual mask. The dayOrNight function is called to determine the time of the image, if it is a night image, the night enhancement algorithm will be applied. Following by the Canny and flood fill function is used to obtain the predicted mask. Then the skyline will use the predicted mask as the input parameter to draw out the horizon using the skyline detection function.

Besides, it will save original image, predicted mask and the skyline in a figure into a new folder call result under the same directory as the algorithm file after visualization. "calAccuracy" function is called twice to compare the predicted mask and the actual mask as well as the predicted skyline and actual skyline. Next, it returns the two values which are for mask accuracy and skyline accuracy.

H. Interpreter

This is the main script for calculating average mask and skyline accuracy for each dataset and overall system accuracy. The system will first run through all values in the datasets list to calculate each image mask accuracy and skyline accuracy then append it into a list called "acc_mlist" and "acc_slist" respectively. The comprehension for loop is used as an error handling to remove none type object in both lists to calculate the mean of the specific dataset.

Furthermore, it will calculate the mean value of mask and skyline lists for each dataset and then take the mean to output and append it into another list call "avg_mask_acc" and "avg_skyline_acc". Then, same as before it all calculates the mean of both newly called lists to get the overall system accuracy of predicted mask, predicted skyline and print in the console.

IV. RESULT AND DISCUSSION

A. Experimental Setup

The experimental setup involves evaluating the proposed method on multiple datasets. The datasets used for evaluation are "623", "684", "9730", "10917" sky finder dataset. Each dataset contains different sky conditions and day or night images. The actual mask of the image is stored in a dictionary list that matches with the dataset name. Then the main script will loop through each dataset from the datasets list and processes all the images in the folder one by one. The proposed method is applied to the images. After processing an image, the predicted image mask will compare with the actual image mask to calculate the accuracy. After all the images are processed, the total average accuracy of mask and skyline is calculated to get the overall system accuracy of both predicted mask and skyline.

B. Results

The table shown below is the performance metrics all the tested datasets using Canny edge detection algorithm for mask and skyline:

Datasets	Mask Accuracy (%)	Skyline Accuracy (%)
623	91%	99%
684	86%	98%
9730	91%	98%
10917	95%	100%
Total Average	**91%**	**99%**

Table 1: Accuracy of Proposed Sky Detection Method

According to Laungrungthip and others [2], the optimal range of upper and lower threshold for the test images is between 186 – 217 and 16 – 61. Therefore, the upper and lower threshold is set to 16 and 186 respectively.

C. Discussion of the Results

One result of each datasets are shown in figures below:

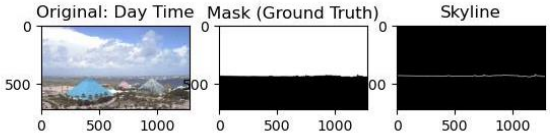

Figure 1: Result from dataset 623

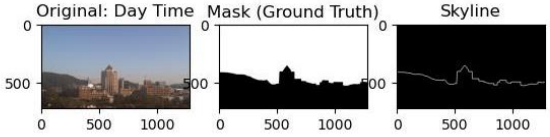

Figure 2: Result from dataset 684

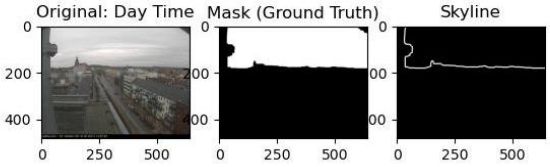

Figure 3: Result from dataset 9730

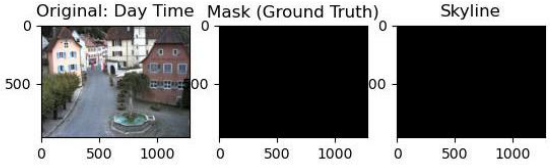

Figure 4: Result from dataset 10917

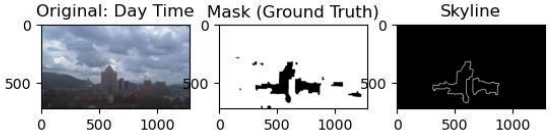

Figure 5: Result from dataset 684 (Complex Cloud)

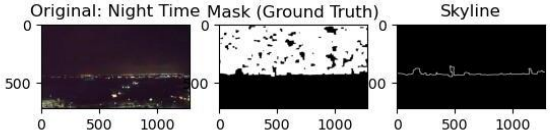

Figure 6: Result of night image from dataset 632

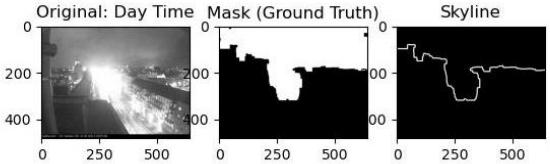

Figure 7: Result of night image from dataset 9730

Dataset of 684 has the lowest accuracy while 10917 dataset has the highest mask accuracy. Some of the day image have trouble extracting the mask because of the complex cloud as shown in Figure 5 but the skyline detection is filtered to remove the cloud. Besides that, another problem encounter during the extraction of the mask is night images is the enhanced night image have hard time to extract the mask and the skyline extract is sometime connected to the sky region as shown in Figure 6. Moreover, result shown in Figure 7 is a night image, but it mislabelled as day image because of the blown-out streetlight causing the algorithm to recognise as a day image and having trouble extracting the mask. Other than that, the computational cost for the system is high as it is doing morphological operations and enhancing night images then saves all the images into a figure from the dataset folder into a new result folder. Additionally, the skyline accuracy overall quite accuracy because the of contours is the only part to compare and get a 99% accuracy for the overall system.

V. CONCLUSION

This paper introduces an algorithm for detecting the sky region, which utilizes the Canny edge detection algorithm with some slight adjustments to obtain the sky region mask and the skyline. However, it is worth noting that the proposed method might not perform well in images heavily impacted by fog and bright blown out streetlight. As future research, efforts will be made to enhance the algorithm's accuracy and adaptability to handle such challenging scene.

REFERENCES

[1] B. Zhao, X. Li, X. Lu, and Z. Wang, "A CNN–RNN architecture for multi-label weather recognition," *Neurocomputing*, vol. 322, pp. 47–57, 2018, DOI: https://doi.org/10.1016/j.neucom.2018.09.048

[2] N. Laungrungthip, A.E. McKinnon, C.D. Churcher and K. Unsworth, "Edge-Based Detection of Sky Regions in Images for Solar Exposure Predicition," *2008 23rd International Conference Image and Vision Computing New Zealand,* 2008. DOI: 10.1109/IVCNZ.2008.4762101

[3] Y. Shen and Q. Wang, "Sky Region Detection in a Single Image for Autonomous Ground Robot Navigation," *International Journal of Advanced Robotic Systems*, vol. 10, no. 10, p. 362, Jan. 2013, DOI: https://doi.org/10.5772/56884.

[4] S. Xu, J. Wang, W. Shou, T. Ngo, A.M. Sadick and X. Wang, "Computer Vision Techniques in Construction: A Critical Review," *Archives of Computational Methods in Engineering,* vol. 28, pp. 3383-3397, 2021, DOI: 10.1007/s11831-020-09504-3

[5] A.S. Khan. and S. Al-Habsi, "Machine Learning in Computer Vision," *Procedia Computer Science,* vol. 167, pp. 1444-1451, 2020, DOI: 10.1016/j.procs.2020.03.355

[6] B. Jiang, W.X. Zhuang, X.L. Ma, Y. Ru, H.Q. Meng and L. Wang, "Method for Sky Region segmentation," *Electronics Letters*, vol. 51, no. 25, pp. 2104-2106, 2015.

[7] Z. Zhao, Q. Wu, H. Sun, X. Jin, Q. Tian and X. Sun, "A Novel Sky Region Detection Algorithm Based on Border Point," International Journal of Signal Processing, Image Processing and Pattern Recognition, vol. 8, no. 3, pp. 281-290, 2015, DOI: 10.14257/ijsip.2015.8.3.26

[8] Y. Song, H. Luo, J. Ma, B. Hui and Z. Chang, "Sky Detection in Hazy Image," *Sensor 2018*, vol. 18, no. 4, 2018, DOI: 10.3390/s18041060

[9] J. Luo and S.P. Etz, "A Physical Model-Based Approach to Detecting Sky in Photographic Images," *IEEE Transactions on Image Processing*, vol. 11, no. 3, 2002, DOI: 10.1109/83.988954

[10] R. Fisher, S. Perkins, A. Walker, and E. Wolfart, "Feature Detectors - Canny Edge Detector," *The University of Edinburgh HIPR2*, 2003, Available:

https://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm#:~:text=How%20It%20Works.
[11] IndianTechWarrior, "Canny Edge Detection for Image Processing," *IndianTechWarrior*, 2021, Available: https://indiantechwarrior.com/canny-edge-detection-for-image-processing/
[12] S. Mallick, "Filling holes in an image using OpenCV (Python / C++)," *LearnOpenCV*, 2015, Available: https://learnopencv.com/filling-holes-in-an-image-using-opencv-python-c/.
[13] P. Campoy *et al.*, "Computer Vision Onboard UAVs for Civilian Tasks," *Journal of Intelligent and Robotic Systems*, vol. 54, no. 1–3, pp. 105–135, 2008, DOI: https://doi.org/10.1007/s10846-008-9256-z.