Sky Detection Project Report

Mengxu Pan

Reflection

- Research and Idea

Originally I planned to use color thresholding combined with edge detection for the detection. The idea behind choosing these two techniques were straightforward:
1. Sky is usually luminated and has some kind of uniform color (hues of blue probably).
2. Sky and the other objects under the sky are usually clearly sperated by some kind of edges. Mountains, buildings and other objects usually have more complex edges than sky, which should have relatively few edges. For cloudy sky, if I can add blur to the image, it should reduce the edges created by cloud further. By adding some kind of filter excluding the area with complex edges, the areas with few or no edges should be sky.
Combing the above two techniques should be separate sky from other objects relatively efficiently.

-Implementation

The technique I ended up using was color thresholding only.
Partially due to the unfamiliarity with OpenCV libraries and the numbers plugged in those functions, I wasn't able to use Canny Edge Detection effectively on top of the results produced by color threshholding.
I also found it difficult to find effective threshold numbers for particular filters.
How to adjust the threshhold numbers dynamically when applied to different pictures were another challenge that I couldn't solve.
I'm hoping to learn more techniques to tackle these problems for future projects while getting familar with OpenCV.
This project was a fun trial and error process.

-Results

The current implementation works well on mostly blue-ish sky. If it is a clear blue sky, the current threshholds works pretty well.
But if it is dawn or dusk cloudy sky, it doesn't work as effectively. On one testing picture with snowy mountains with cloudy sky, the mask falsely detected all the areas with snow as sky because the snowy areas has blue-ish hue that's within our filter range.
Although the current implementation is very simple and not effective for all kinds of sky, I had to learn OpenCV funtions and produce a relatively effective results in a short amount of time. I'm happy with the current result and consider this project as a fun learning process.

Documentation

The chosen technique for sky detection involves utilizing the HSV (Hue, Saturation, Value) color space. The rationale behind this choice is that the HSV color space is particularly effective for color-based image segmentation. In the HSV space, the hue component represents the color information, the saturation component represents the vividness or intensity of the color, and the value component represents the brightness. By isolating the value channel and applying histogram equalization, we aim to enhance the contrast of the image, making it easier to identify the sky region based on its characteristic color.
The subsequent steps involve defining a color range corresponding to the sky, creating a binary mask through thresholding, and refining the mask using morphological operations. The final result is obtained by masking the original image, leaving only the non-sky regions.

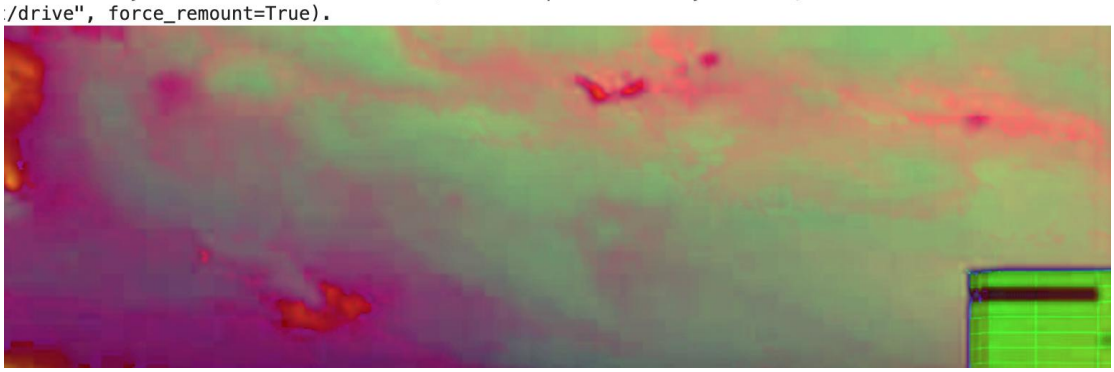
Implementation steps:

-Read the input image (img) using OpenCV.
-Convert the image to the HSV color space (hsv_img).

```
# import image
img = cv2.imread('/content/drive/MyDrive/CV/sky.jpeg')

hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
cv2_imshow(hsv_img)
```

```
rive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/conten
/drive", force_remount=True).
```



-Equalize the histogram of the value channel (v) to enhance the brightness.

-Merge the modified channels back into an HSV image (mer_img).

-Define lower (iLow) and upper (iHigh) bounds in HSV for detecting the sky color.

-Create a binary mask based on the color range (rng_img).

```
h, s, v = cv2.split(hsv_img)
v = cv2.equalizeHist(v)
mer_img = cv2.merge((h, s, v))

iLow = np.array([100, 43, 46])
iHigh = np.array([124, 255, 255])

rng_img = cv2.inRange(mer_img, iLow, iHigh)
cv2_imshow(rng_img)
```

-Perform morphological operations (erosion and dilation) to refine the mask.

[5]:
```
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (10, 10))
erode_img = cv2.erode(rng_img, kernel)
dlt_img = cv2.dilate(erode_img, kernel)
cv2_imshow(dlt_img)
```

-Threshold the mask to create a binary mask (mask).

-Invert the binary mask (notmask).

-Use the inverted mask to extract the non-sky regions from the original image

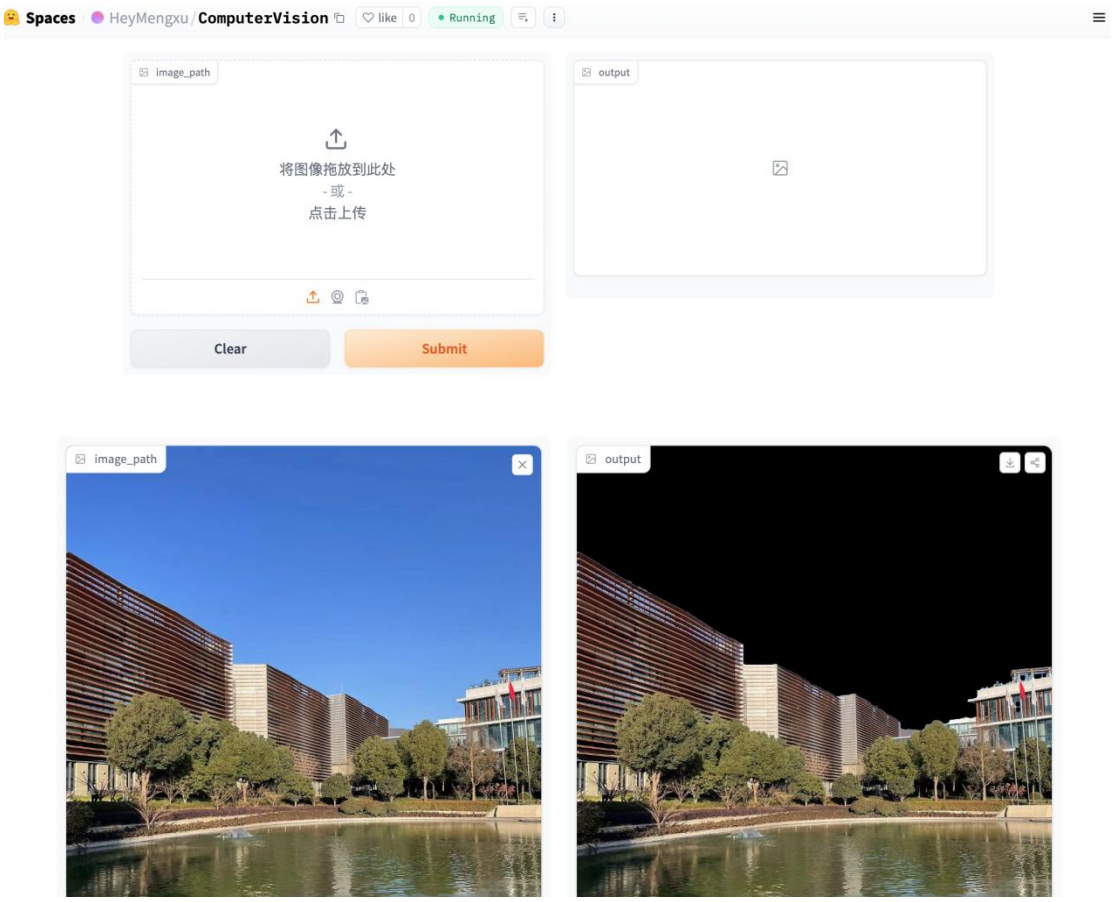
-Convert the result image to RGB format, as Gradio expects RGB.

[6]:
```
ret, mask = cv2.threshold(dlt_img, 150, 255, cv2.THRESH_BINARY)
notmask = cv2.bitwise_not(mask)
frontpic = cv2.bitwise_and(img, img, mask=notmask)

cv2_imshow(frontpic)
cv2.waitKey(0)
```

Demo using Gario and Hugging Face:

https://huggingface.co/spaces/HeyMengxu/ComputerVision



Challenges and Problem Solving

Tuning Color Range: Setting the appropriate HSV color range for sky detection required experimentation. Adjusting the lower and upper bounds involved trial and error to achieve satisfactory results.

Morphological Operations: Determining the size and shape of the structuring element for morphological operations posed a challenge. Fine-tuning was necessary to achieve a balance between noise reduction and preserving relevant details.

Thresholding: Selecting an optimal threshold value for creating the binary mask involved considerations for image-specific characteristics. Iterative adjustments were made to find a value that worked well across various images.