การแบ่งพื้นที่ภาพที่มีสีเดียวกันด้วยปริภูมสี HSV

รายชื่อสมาชิก

1. B6403218 นายพีรณัฐ สังขวาสี
2. B6414313 ไชยนันต์ สีขจร

เสนอ

อาจารย์ ผศ.ดร.ปรเมศวร์ ห่อแก้ว

รายงานนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาตรี

สำนักวิชาวิศวกรรมศาสตร์

สาขาวิชาวิศวกรรมคอมพิวเตอร์

ภาคเรียนที่ 2 ปีการศึกษา 2566

มหาวิทยาลัยเทคโนโลยีสุรนารี

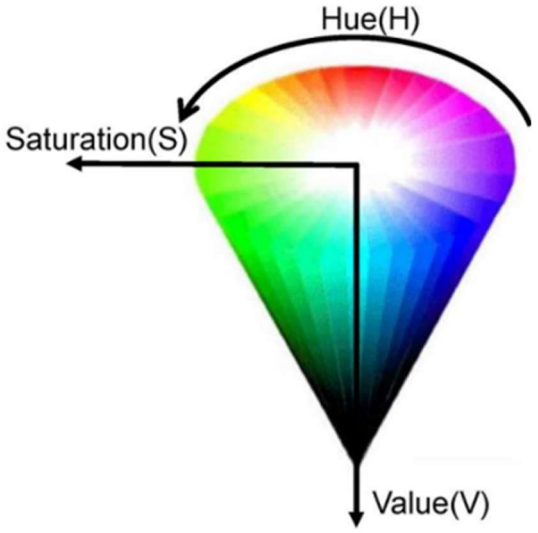
วัตถุประสงค์

1. เพื่อศึกษาการแบ่งพื้นที่ของภาพที่มีสีเดียวกันที่ด้วยปริภูมิสี HSV
2. เพื่อแบ่งภาพภูเขาออกจากพื้นที่ธรรมดา
3. เพื่อทดสอบการใช้ Otsu's method ในการหาค่าแบ่งระดับ

เนื้อหาที่เกี่ยวข้อง

1. **ปริภูมิสีแบบ HSV**

ปริภูมิสีแบบ HSV จะแบ่งค่าด้วย เนื้อสี(Hue), ความอิ่มตัว(Saturation), ค่าเทากลาง(Value)



โดย H 1 ค่า (สีเดียว) สามารถมีได้หลาย S, V ทำให้ในทาง Computer Vision สามารถแบ่งพื้นที่ที่มีสีเดียวกันจากค่าเทากลางและค่าความอิ่มตัวได้

2. **Otsu's Method**

เป็นการเลือกระดับค่า Threshold จากข้อมูลของความถี่แต่ละค่า โดยคำนวณหาค่า Threshold จากพิสัยของค่าที่ต้องการจากตารางแจกแจงความถี่เพื่อหาว่า 1) ค่าใดที่ทำให้ความแปรปรวนภายในกลุ่มน้อยที่สุด (intra-class variance) 2) ค่าใดที่ทำให้ความแปรปรวนระหว่างกลุ่มมากที่สุด (inter-class variance)

Intra-class variance

$$\sigma_{\omega}^2 = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

Inter-class variance

$$\sigma_b^2 = \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2$$

โดย

$$\omega_0(t) = \sum_{i=0}^{t-1} \mathrm{p(i)} \qquad \omega_1(t) = \sum_{i=t}^{L-1} \mathrm{p(i)}$$

$$\mu_0(t) = \sum_{i=0}^{t-1} \mathrm{ip(i)} / \omega_0(t) \qquad \mu_1(t) = \sum_{i=t}^{L-1} \mathrm{ip(i)} / \omega_1(t)$$

$$\mu_T(t) = \sum_{i=0}^{L-1} \mathrm{ip(i)}$$

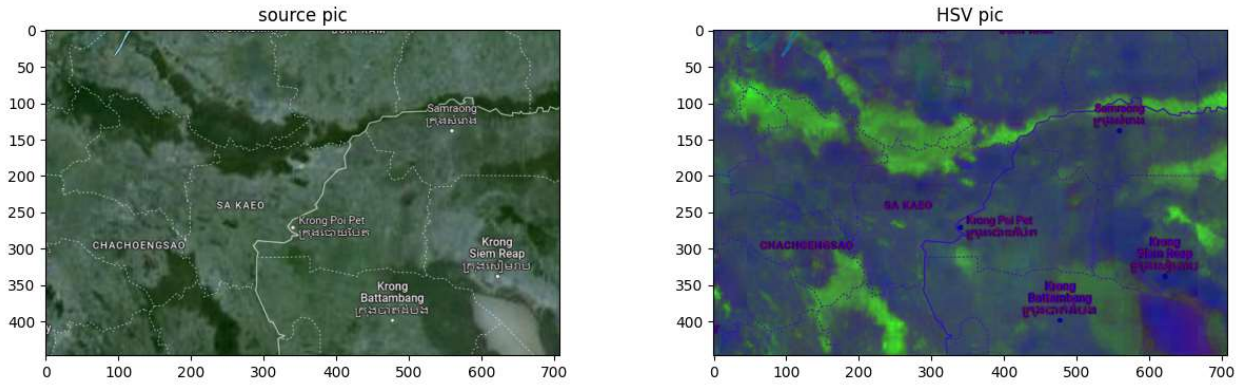
$$\sigma_0^2(t) = \frac{1}{N} \sum_{i=0}^{t-1} (\mathrm{p(i)} - \mu_0)^2 \quad \sigma_1^2(t) = \frac{1}{N} \sum_{i=t}^{L-1} (\mathrm{p(i)} - \mu_1)^2$$

เมื่อได้ค่า Threshold จากการคำนวณแล้ว นำมาใช้แบ่งภาพแปลง pixel เพื่อให้แบ่งพื้นที่ที่ต้องการได้

SourceCode

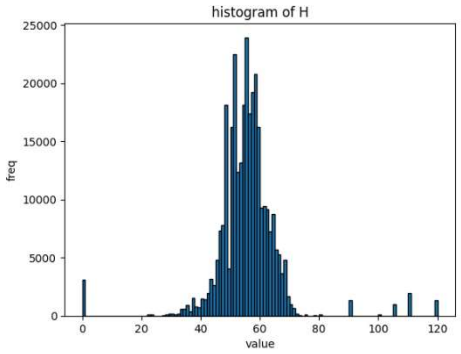
```
import cv2
import matplotlib.pyplot as plt
from collections import Counter
import statistics

import numpy as np
import pandas as pd

img = cv2.imread('C:/Users/ACER/Downloads/Teeth Segmentation.v2i.tensorflow/3.png',
1)
h, w, channel = img.shape

max_x = w - 1
max_y = h - 1
```

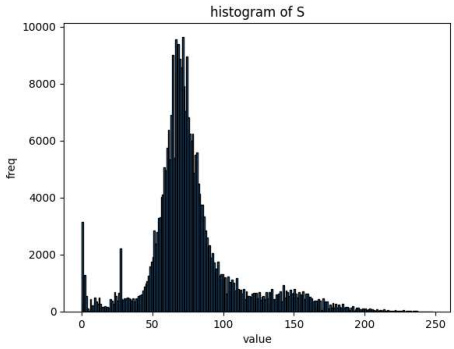
```
imgHSV =cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

fig, (ax_1, ax_2) = plt.subplots(1, 2, figsize=(15, 4))

ax_1.imshow(img)
ax_1.set_title('source pic')
ax_2.imshow(imgHSV)
ax_2.set_title('HSV pic')
```

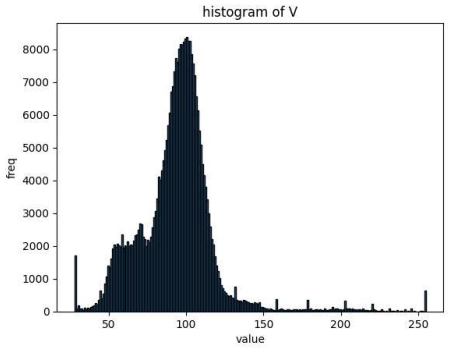



```
H = []
S = []
V = []

for i in range(max_y):
    for j in range(max_x):
        pix_val = imgHSV[i,j]
        H.append(pix_val[0])
        S.append(pix_val[1])
        V.append(pix_val[2])
```

```
plt.hist(H, bins = range(min(H), max(H) + 1),
            edgecolor='black')
plt.xlabel('value')
plt.ylabel('freq')
plt.title('histogram of H')
plt.show()

plt.hist(S, bins = range(min(S), max(S) + 1),
            edgecolor='black')
plt.xlabel('value')
plt.ylabel('freq')
plt.title('histogram of S')
plt.show()

plt.hist(V, bins = range(min(V), max(V) + 1),
            edgecolor='black')
plt.xlabel('value')
plt.ylabel('freq')
plt.title('histogram of V')
plt.show()
```








```
# Otsu to select V treshold
list_infra = []
list_inter = []

for t in range(0, max(V)+1):
    elements_less_than_t = [element for element in V if element < t]
    elements_more_than_t = [element for element in V if element >= t]

    counter_elements_less_than_t = Counter(elements_less_than_t)
    counter_elements_more_than_t = Counter(elements_more_than_t)

    w0_t = sum(counter_elements_less_than_t[i] for i in range(t))
    w1_t = sum(counter_elements_more_than_t[i] for i in range(t,max(V)+1))
```

```
    mu0_t = 0 if t == 0 else np.mean(elements_less_than_t)
    mu1_t = np.mean(elements_more_than_t)

    sigma0_squared_t = 0 if t == 0 else np.var(elements_less_than_t)
    sigma1_squared_t = np.var(elements_more_than_t)

    infra_class_variance = w0_t * sigma0_squared_t + w1_t * sigma1_squared_t
    inter_class_variance = w0_t * w1_t * (mu0_t - mu1_t)**2

    list_infra.append(infra_class_variance)
    list_inter.append(inter_class_variance)
```

```
max_inter_value = max(list_inter)
max_inter_indices = [i for i, value in enumerate(list_inter) if value ==
max_inter_value]

min_infra_value = min(list_infra)
min_infra_indices = [i for i, value in enumerate(list_infra) if value ==
min_infra_value]
print("\n=====================================================\n")
print("Maximum Inter-class Variance:", max_inter_value)
print("Corresponding t values at indices:", max_inter_indices)
print()
print("Minimum Infra-class Variance:", min_infra_value)
print("Corresponding t values at indices:", min_infra_indices)
```

```
=====================================================

Maximum Inter-class Variance: 29561446550632.504
Corresponding t values at indices: [88]

Minimum Infra-class Variance: 106354006.53482944
Corresponding t values at indices: [88]
```

```
for i in range(max_y):
    for j in range(max_x):
        pix_val = imgHSV[i,j]
        if(pix_val[2] < 88):
            imgHSV[i, j] = [0, 0, 0]
        else:
            imgHSV[i, j] = [255, 255, 255]



fig, (ax_3, ax_4) = plt.subplots(1, 2, figsize=(15, 4))
```

```
ax_3.imshow(img)
ax_3.set_title('source pic')

ax_4.imshow(imgHSV)
ax_4.set_title('Result')
```

ผลการทดลอง