Numerical Methods and Machine Learning for Image Processing

Week 2, Class 2: Basic Color Segmentation

September 28, 2021

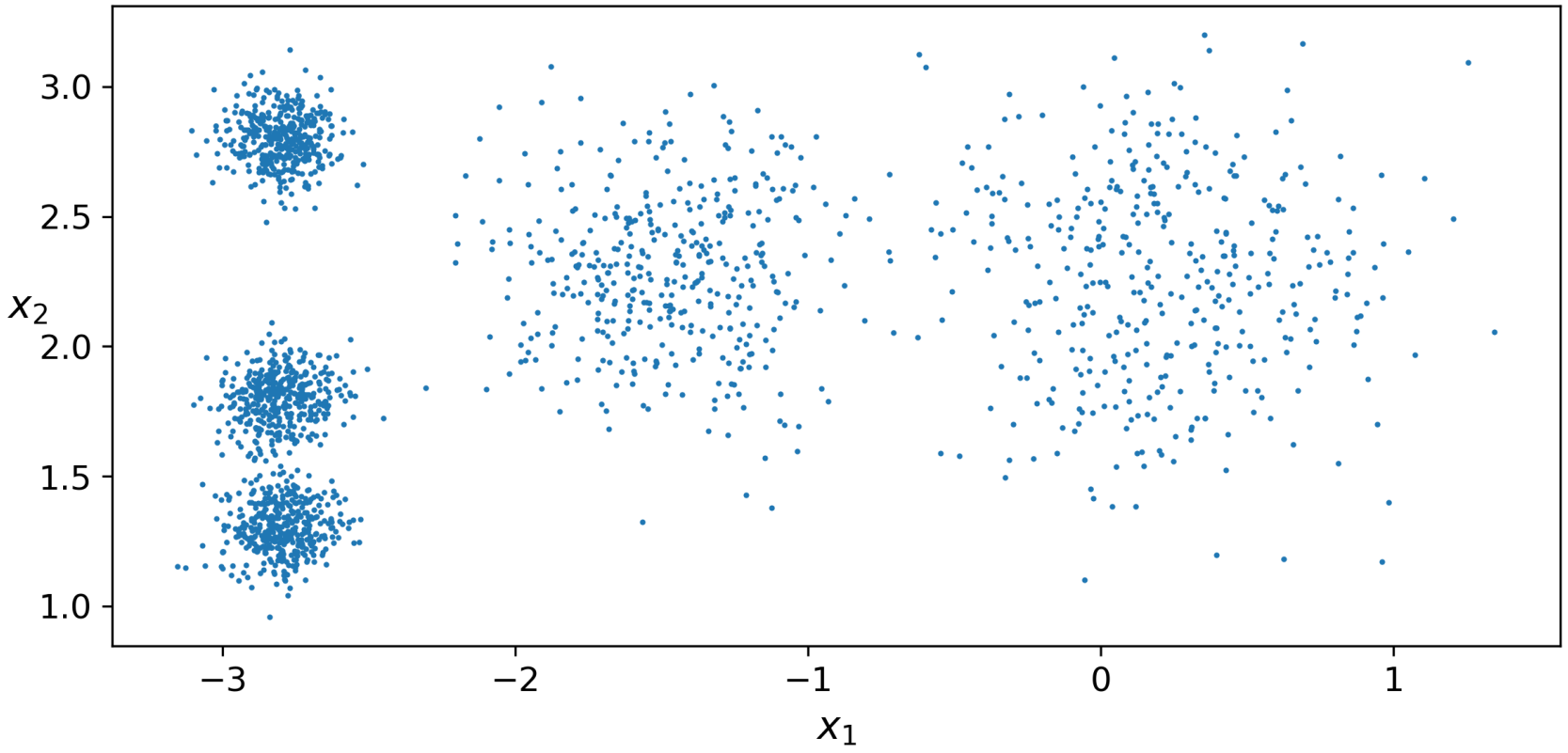
Damon M. Chandler and Yi Zhang

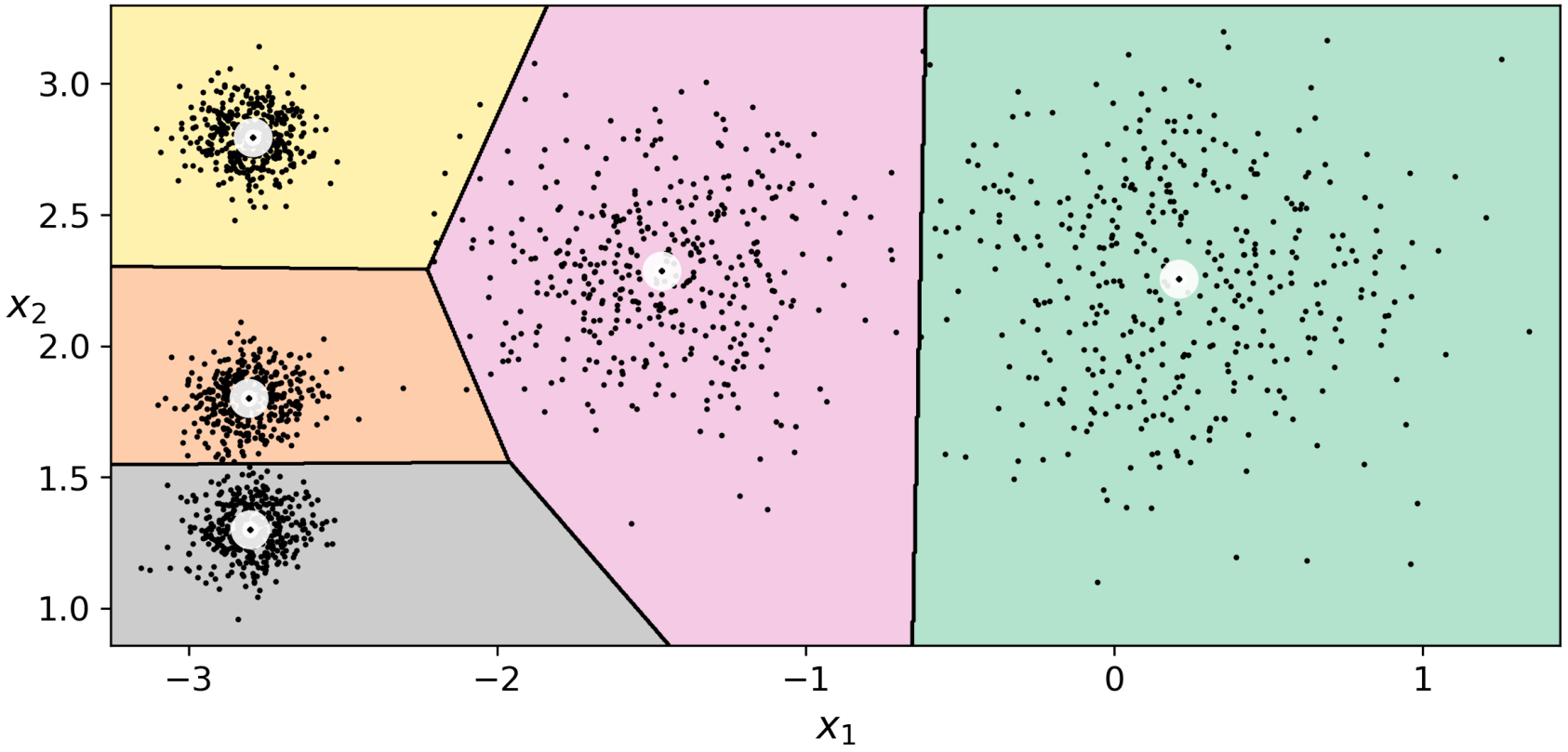
Last Time: Basic Color Segmentation, Part 1

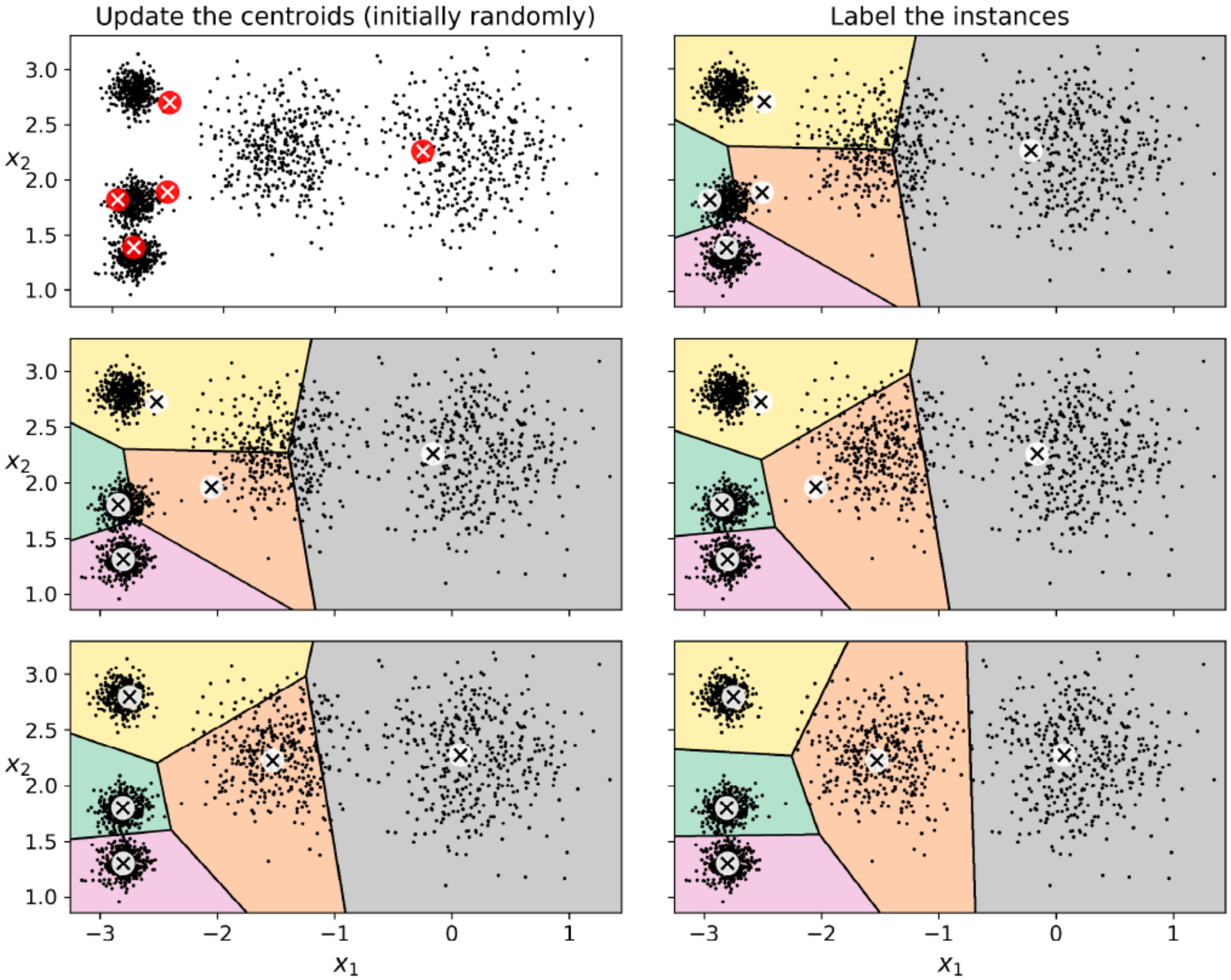
1. What is color?
2. How to load and work with RGB images
3. How to convert between RGB and the…
   - Hue, Value, Saturation (HSV) color space
   - Opponent color spaces:
     - YCrCb color space
     - CIE L*a*b* color space
4. How to do simple color segmentation using HSV distances

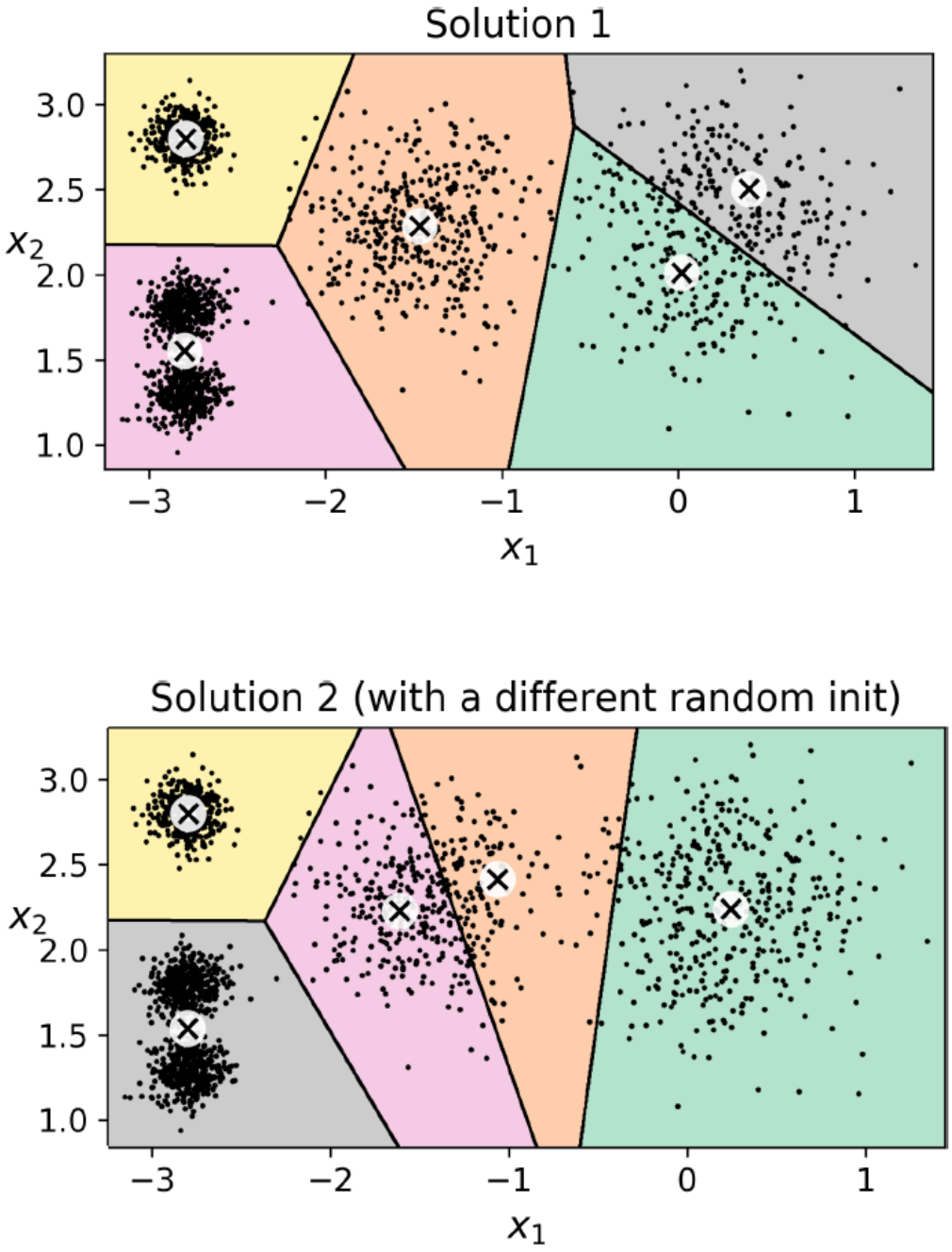
Today: Basic Color Segmentation, Part 1

1. *k*-means clustering idea
2. Using *k*-means clustering of colors for image segmentation in the…
   - RGB color space
   - Hue, Value, Saturation (HSV) color space
   - CIE L*a*b* color space
3. Using color (and lightness) to find salient regions
   - How to compute lightness and color distance feature maps


3.0
2.5
2.0
1.5
1.0
x_2
−3
−2
−1
0
1
x_1


3.0
2.5
2.0
1.5
1.0
x_2
−3
−2
−1
0
1
x_1


Update the centroids (initially randomly)
Label the instances
x_2
x_1
3.0
2.5
2.0
1.5
1.0
−3
−2
−1
0
1


Solution 1
3.0
2.5
2.0
1.5
1.0
x_2
−3
−2
−1
0
1
x_1
Solution 2 (with a different random init)
3.0
2.5
2.0
1.5
1.0
x_2
−3
−2
−1
0
1
x_1

k-means clustering (RGB color space)

```python
import numpy as np
import cv2 as cv
import ipcv_utils.utils as ipcv_plt
import matplotlib.pyplot as plt

img_bgr = cv.imread("imgs/1600.png")
img_bgr = img_bgr.astype(np.float32) / 255

img_rgb = cv.cvtColor(img_bgr, cv.COLOR_BGR2RGB)
ipcv_plt.imshow(img_rgb)

rgb_vals = img_rgb.reshape(-1, 3)

stop_criteria = (cv.TERM_CRITERIA_EPS +
  cv.TERM_CRITERIA_MAX_ITER, 100, 0.2)
_, labels, centers = cv.kmeans(rgb_vals, 3, None,
  stop_criteria, 10, cv.KMEANS_RANDOM_CENTERS)

print("~ Labels ~~~~~~~~~~~~~~~~~~~~~~")
labels = labels.flatten()
print(labels.shape)
print(labels)

print("~ Centers ~~~~~~~~~~~~~~~~~~~~~~")
print(centers.shape)
print(centers)

fig = plt.figure()
ax = fig.add_subplot(111, projection="3d")
ax.scatter(rgb_vals[::100, 0], rgb_vals[::100, 1],
  rgb_vals[::100, 2], c=rgb_vals[::100], s=10)
ax.set_xlabel("R")
ax.set_ylabel("G")
ax.set_zlabel("B")
plt.show()
```

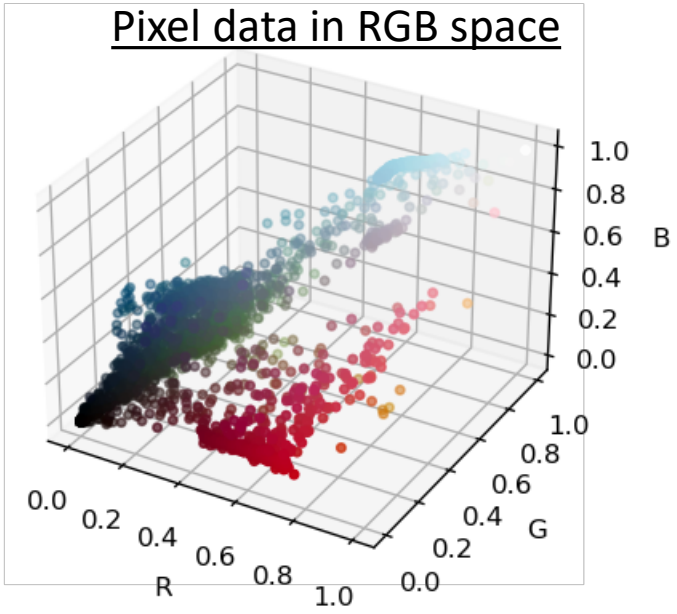
k=3 clusters

```python
seg_rgb_vals = centers[labels]

fig = plt.figure()
ax = fig.add_subplot(111, projection="3d")
ax.scatter(rgb_vals[::100, 0], rgb_vals[::100, 1],
  rgb_vals[::100, 2], c=rgb_vals[::100], s=1,
  alpha=0.1)
ax.scatter(centers[:, 0], centers[:, 1],
  centers[:, 2], c=centers, s=75, marker="^",
  edgecolors=[(0, 0, 0)], alpha=1)
ax.set_xlabel("R")
ax.set_ylabel("G")
ax.set_zlabel("B")
plt.show()

fig = plt.figure()
ax = fig.add_subplot(111, projection="3d")
ax.scatter(rgb_vals[::100, 0], rgb_vals[::100, 1],
  rgb_vals[::100, 2], c=seg_rgb_vals[::100], s=10)
ax.set_xlabel("R")
ax.set_ylabel("G")
ax.set_zlabel("B")
plt.show()

seg_img_rgb = seg_rgb_vals.reshape(img_rgb.shape)
ipcv_plt.imshow(seg_img_rgb)
```

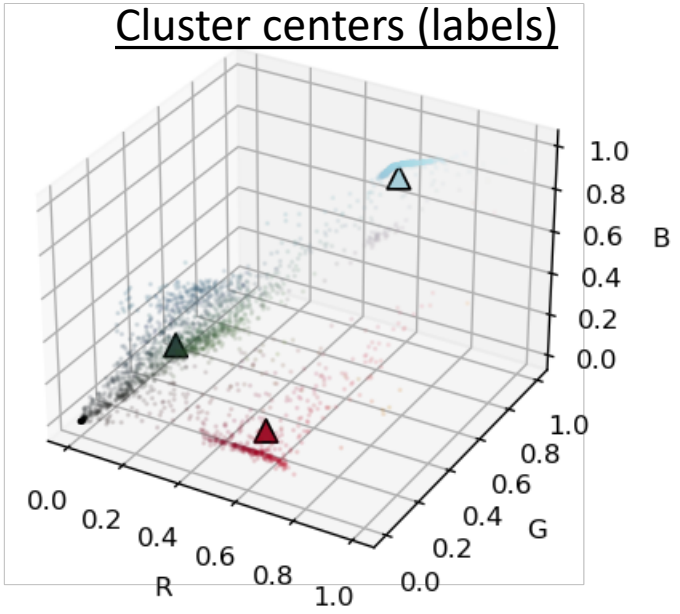
k-means clustering (RGB color space)

```
~ Labels ~~~~~~~~~~~~~~~~~~~~~~
(262144,)
[2 2 2 ... 0 1 1]

~ Centers ~~~~~~~~~~~~~~~~~~~~~
(3, 3)
[[0.6214775 0.07328323 0.16530055]
 [0.1819372 0.27022806 0.24974917]
 [0.6595138 0.81818056 0.86416453]]
```

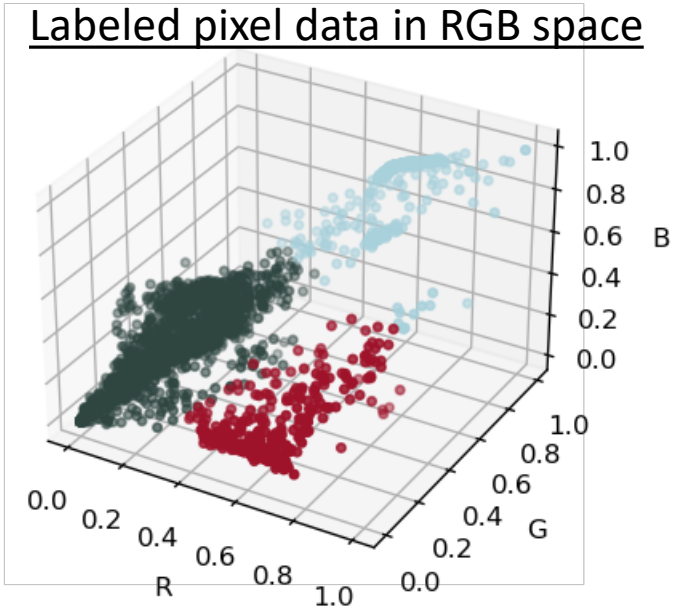
Pixel data in RGB space



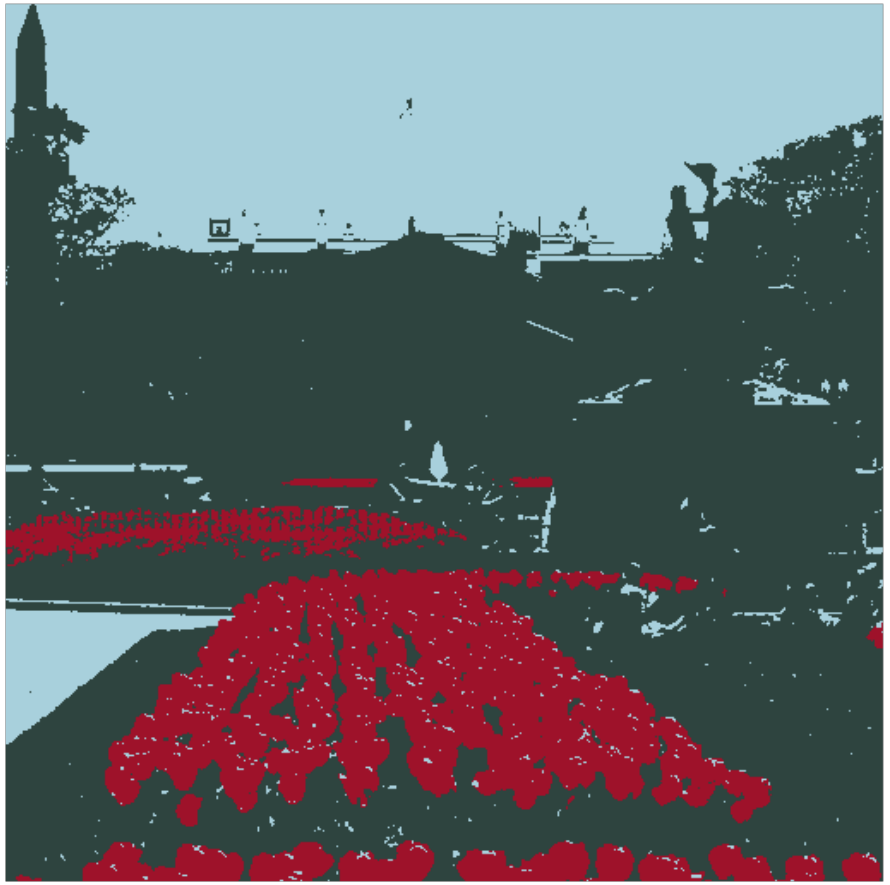
Cluster centers (labels)



Labeled pixel data in RGB space



k-means clustering (RGB color space)



Segmented image using 3 clusters in RGB

k-means clustering (HSV color space)

```python
import numpy as np
import cv2 as cv
import ipcv_utils.utils as ipcv_plt
import matplotlib.pyplot as plt

img_bgr = cv.imread("imgs/1600.png")
img_bgr = img_bgr.astype(np.float32) / 255

img_rgb = cv.cvtColor(img_bgr, cv.COLOR_BGR2RGB)
ipcv_plt.imshow(img_rgb)

img_hsv = cv.cvtColor(img_rgb, cv.COLOR_RGB2HSV)

rgb_vals = img_rgb.reshape(-1, 3)
hsv_vals = img_hsv.reshape(-1, 3)

stop_criteria = (cv.TERM_CRITERIA_EPS +
  cv.TERM_CRITERIA_MAX_ITER, 100, 0.2)
_, labels, centers = cv.kmeans(hsv_vals, 3, None,
  stop_criteria, 10, cv.KMEANS_RANDOM_CENTERS)

print("~ Labels ~~~~~~~~~~~~~~~~~~~~~~~~")
labels = labels.flatten()
print(labels.shape)
print(labels)

print("~ Centers ~~~~~~~~~~~~~~~~~~~~~~~")
print(centers.shape)
print(centers)

fig = plt.figure()
ax = fig.add_subplot(111, projection="3d")
ax.scatter(hsv_vals[::100, 0], hsv_vals[::100, 1],
  hsv_vals[::100, 2], c=rgb_vals[::100], s=10)
ax.set_xlabel("H")
ax.set_ylabel("S")
ax.set_zlabel("V")
plt.show()
```

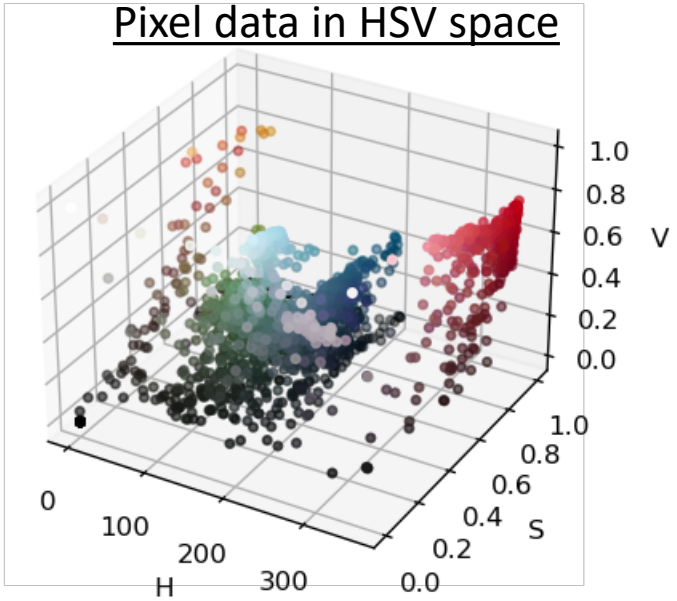
k=3 clusters

```python
seg_hsv_vals = centers[labels]

# convert the HSV centers and HSV segmented list to RGB
# so that we can color each point in the 3D graphs (for
# plotting purposes only)
centers_as_rgb = cv.cvtColor(
  centers.reshape((-1, 1, 3)), cv.COLOR_HSV2RGB)
seg_hsv_vals_as_rgb = cv.cvtColor(
  seg_hsv_vals.reshape((-1, 1, 3)), cv.COLOR_HSV2RGB)

fig = plt.figure()
ax = fig.add_subplot(111, projection="3d")
ax.scatter(hsv_vals[::100, 0], hsv_vals[::100, 1],
  hsv_vals[::100, 2], c=rgb_vals[::100], s=1, alpha=0.1)
ax.scatter(centers[:, 0], centers[:, 1], centers[:, 2],
  c=centers_as_rgb, s=75, marker="^",
  edgecolors=[(0, 0, 0)], alpha=1)
ax.set_xlabel("H")
ax.set_ylabel("S")
ax.set_zlabel("V")
plt.show()

fig = plt.figure()
ax = fig.add_subplot(111, projection="3d")
ax.scatter(hsv_vals[::100, 0], hsv_vals[::100, 1],
  hsv_vals[::100, 2], c=seg_hsv_vals_as_rgb[::100],
  s=10)
ax.set_xlabel("H")
ax.set_ylabel("S")
ax.set_zlabel("V")
plt.show()

seg_img_hsv = seg_hsv_vals.reshape(img_hsv.shape)
seg_img_rgb = cv.cvtColor(seg_img_hsv, cv.COLOR_HSV2RGB)
ipcv_plt.imshow(seg_img_rgb)
```

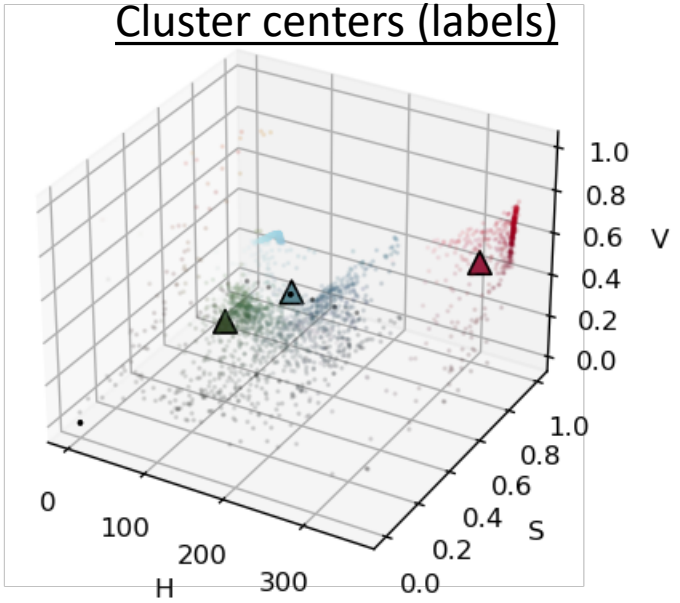
k-means clustering (HSV color space)

```
~ Labels ~~~~~~~~~~~~~~~~~~~~~~
(262144,)
[1 1 1 ... 0 2 2]

~ Centers ~~~~~~~~~~~~~~~~~~~~~
(3, 3)
[[3.4369376e+02 7.9921579e-01 5.9261608e-01]
 [1.9129132e+02 3.8068202e-01 5.7006133e-01]
 [9.5794380e+01 4.2606351e-01 2.9646501e-01]]
```

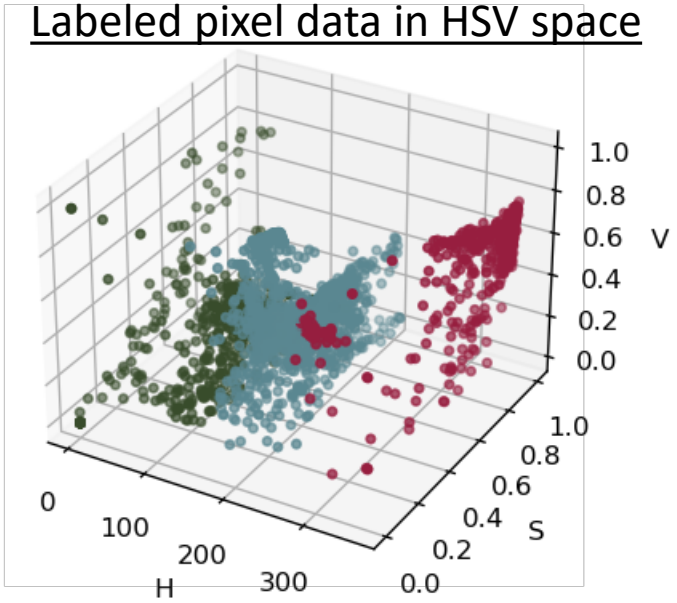
Pixel data in HSV space



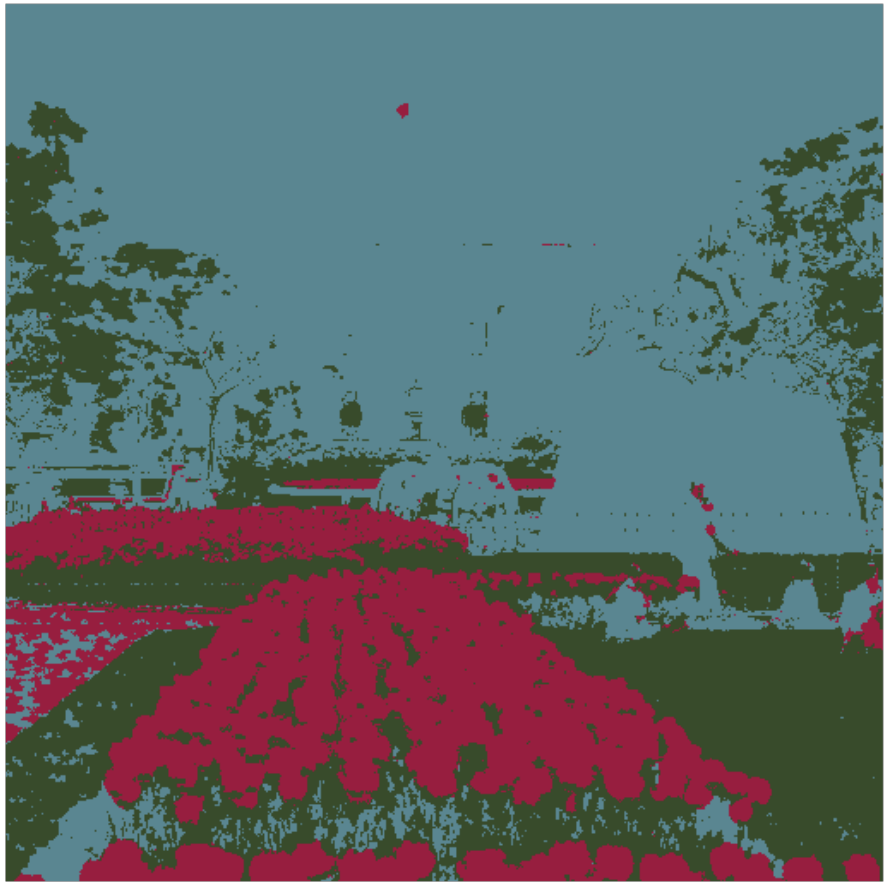
Cluster centers (labels)



Labeled pixel data in HSV space



k-means clustering (HSV color space)



Segmented image using 3 clusters in HSV

k-means clustering (Lab color space)

```python
import numpy as np
import cv2 as cv
import ipcv_utils.utils as ipcv_plt
import matplotlib.pyplot as plt

img_bgr = cv.imread("imgs/1600.png")
img_bgr = img_bgr.astype(np.float32) / 255

img_rgb = cv.cvtColor(img_bgr, cv.COLOR_BGR2RGB)
ipcv_plt.imshow(img_rgb)

img_lab = cv.cvtColor(img_rgb, cv.COLOR_RGB2Lab)

rgb_vals = img_rgb.reshape(-1, 3)
lab_vals = img_lab.reshape(-1, 3)

stop_criteria = (cv.TERM_CRITERIA_EPS +
  cv.TERM_CRITERIA_MAX_ITER, 100, 0.2)
_, labels, centers = cv.kmeans(lab_vals, 3, None,
  stop_criteria, 10, cv.KMEANS_RANDOM_CENTERS)

print("~ Labels ~~~~~~~~~~~~~~~~~~~~~~~~")
labels = labels.flatten()
print(labels.shape)
print(labels)

print("~ Centers ~~~~~~~~~~~~~~~~~~~~~~~")
print(centers.shape)
print(centers)

fig = plt.figure()
ax = fig.add_subplot(111, projection="3d")
ax.scatter(lab_vals[::100, 0], lab_vals[::100, 1],
  lab_vals[::100, 2], c=rgb_vals[::100], s=10)
ax.set_xlabel("L")
ax.set_ylabel("a")
ax.set_zlabel("b")
plt.show()
```

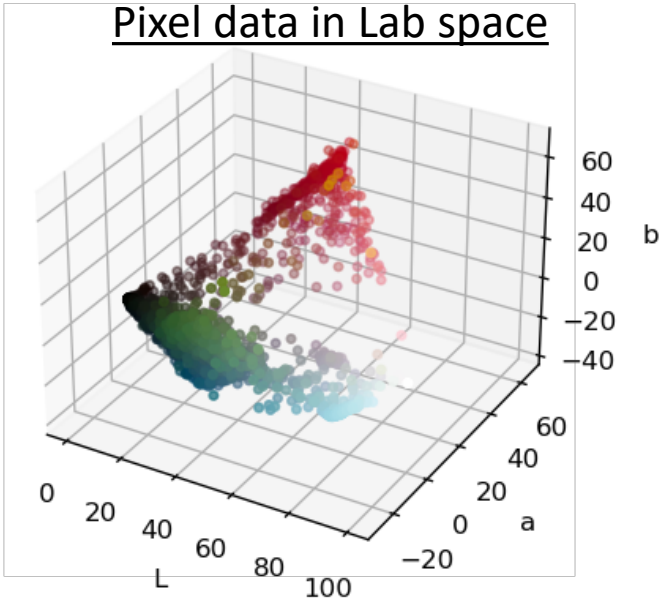
k=3 clusters

```python
seg_lab_vals = centers[labels]

# convert the Lab centers and Lab segmented list to RGB
# so that we can color each point in the 3D graphs (for
# plotting purposes only)
centers_as_rgb = cv.cvtColor(
  centers.reshape((-1, 1, 3)), cv.COLOR_Lab2RGB)
seg_lab_vals_as_rgb = cv.cvtColor(
  seg_lab_vals.reshape((-1, 1, 3)), cv.COLOR_Lab2RGB)

fig = plt.figure()
ax = fig.add_subplot(111, projection="3d")
ax.scatter(lab_vals[::100, 0], lab_vals[::100, 1],
  lab_vals[::100, 2], c=rgb_vals[::100], s=1, alpha=0.1)
ax.scatter(centers[:, 0], centers[:, 1],
  centers[:, 2], s=75, marker="^",
  edgecolors=[(0, 0, 0)], alpha=1)
ax.set_xlabel("L")
ax.set_ylabel("a")
ax.set_zlabel("b")
plt.show()

fig = plt.figure()
ax = fig.add_subplot(111, projection="3d")
ax.scatter(hsv_vals[::100, 0], hsv_vals[::100, 1],
  hsv_vals[::100, 2], c=seg_lab_vals_as_rgb[::100],
  s=10)
ax.set_xlabel("L")
ax.set_ylabel("a")
ax.set_zlabel("b")
plt.show()

seg_img_lab = seg_lab_vals.reshape(img_lab.shape)
seg_img_rgb = cv.cvtColor(seg_img_lab, cv.COLOR_Lab2RGB)
ipcv_plt.imshow(seg_img_rgb)
```

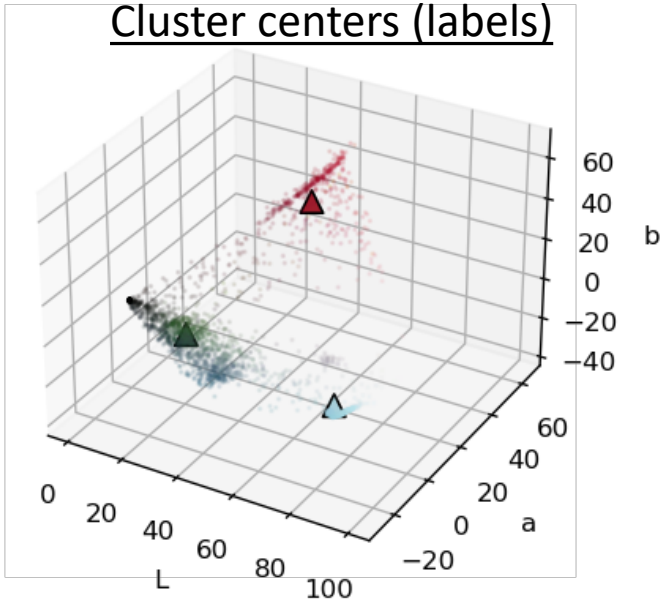
k-means clustering (Lab color space)

```
~ Labels ~~~~~~~~~~~~~~~~~~~~~~
(262144,)
[0 0 0 ... 1 2 2]

~ Centers ~~~~~~~~~~~~~~~~~~~~~
(3, 3)
[[ 80.91052    -10.976559   -10.021635  ]
 [ 34.762596    51.322037    25.893234  ]
 [ 26.973904    -8.763228     0.48884922]]
```

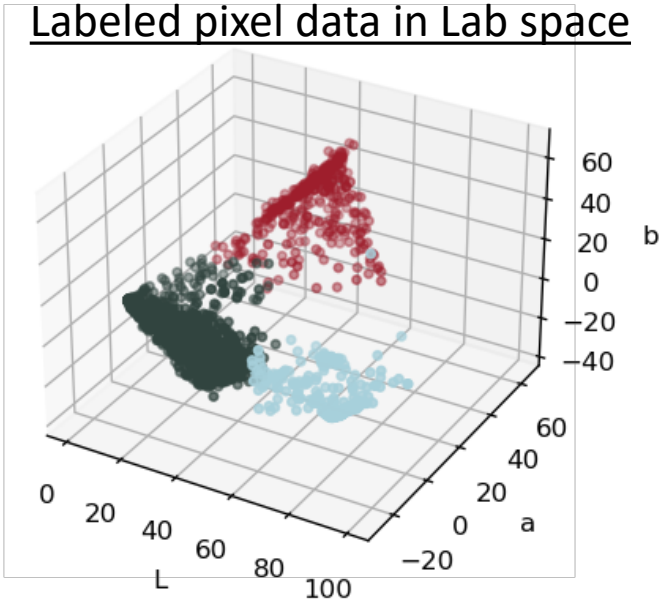
Pixel data in Lab space



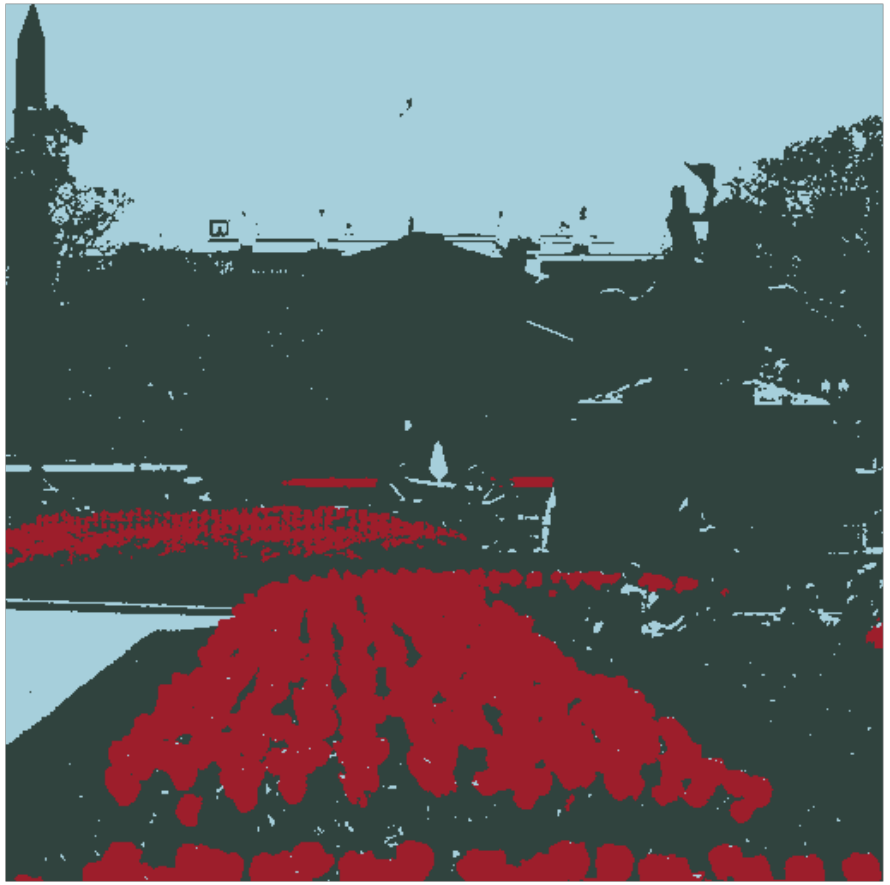
Cluster centers (labels)



Labeled pixel data in Lab space



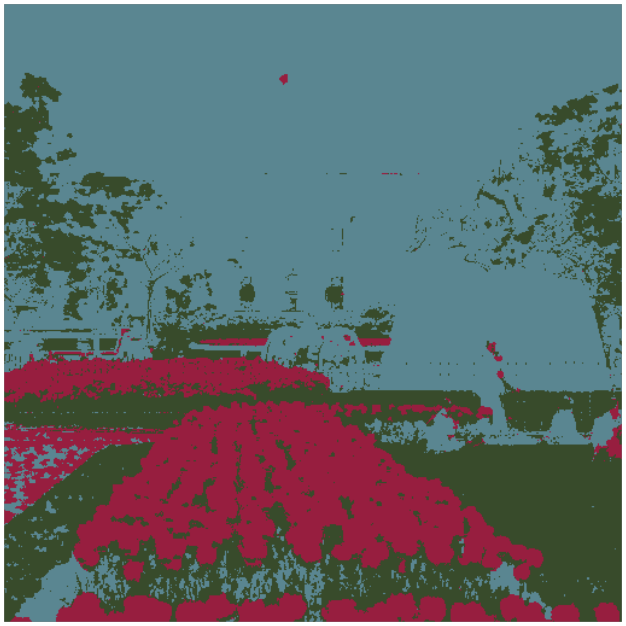
k-means clustering (Lab color space)



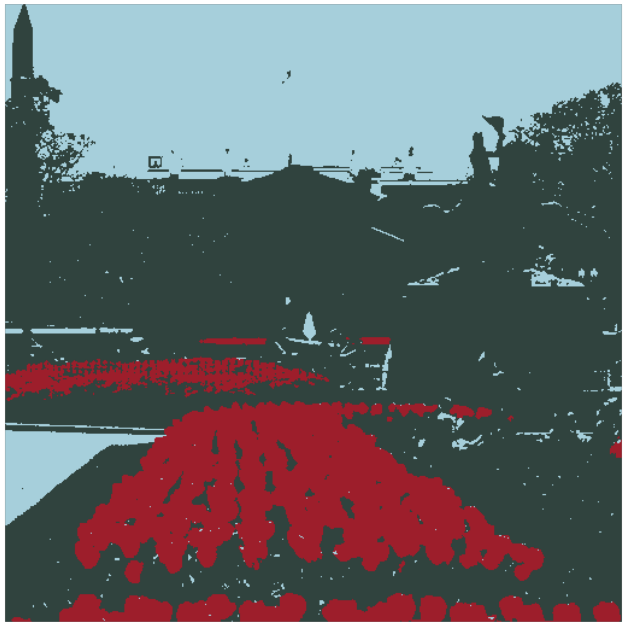
Segmented image using 3 clusters in Lab

K-means color segmentation with **3 clusters**

RGB color space



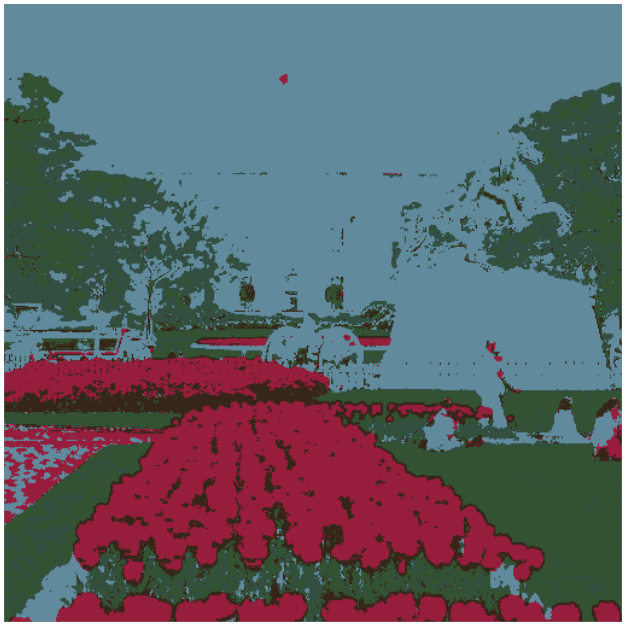
HSV color space



Lab color space

K-means color segmentation with **5 clusters**

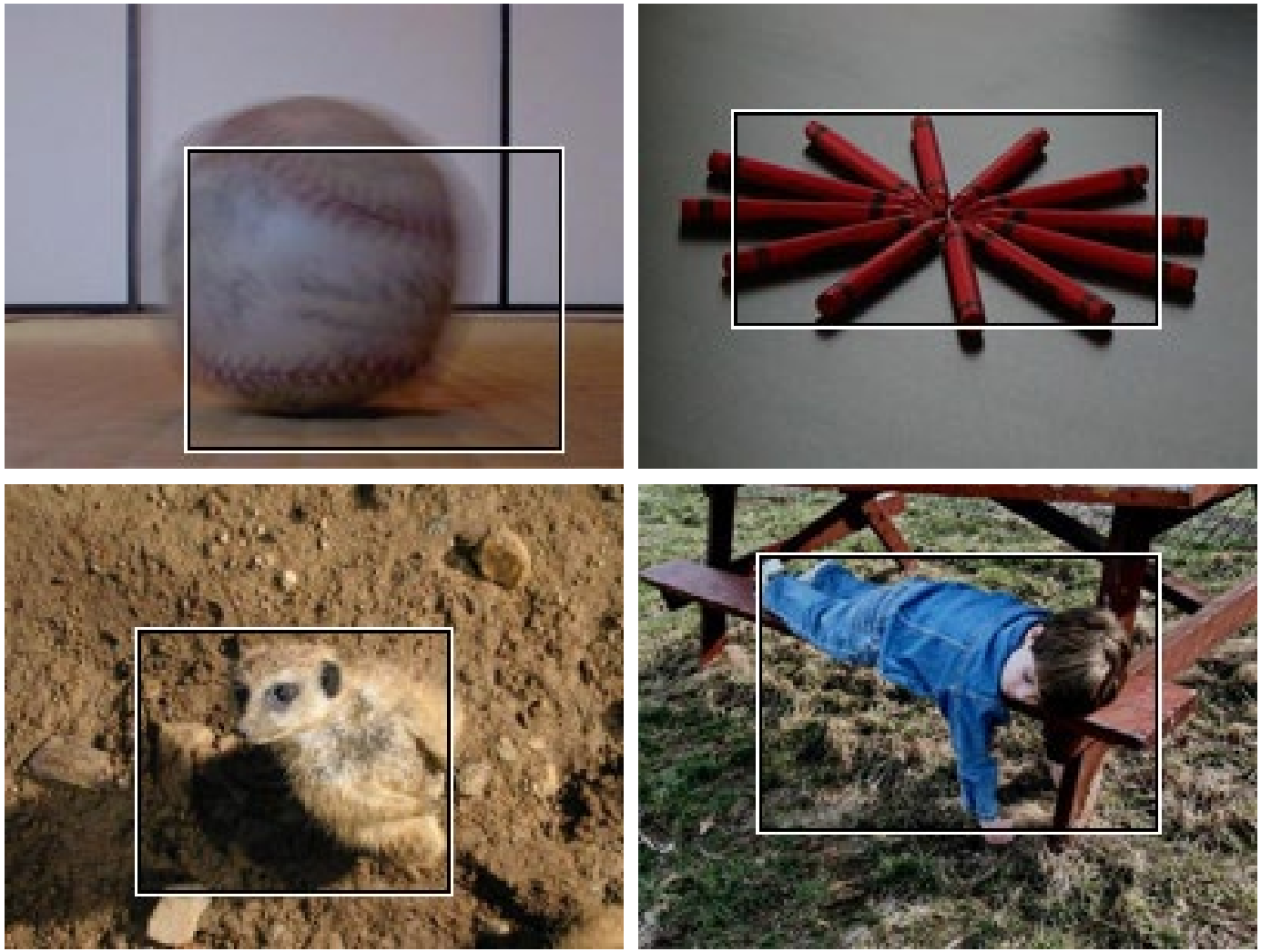
RGB color space



HSV color space

Lab color space

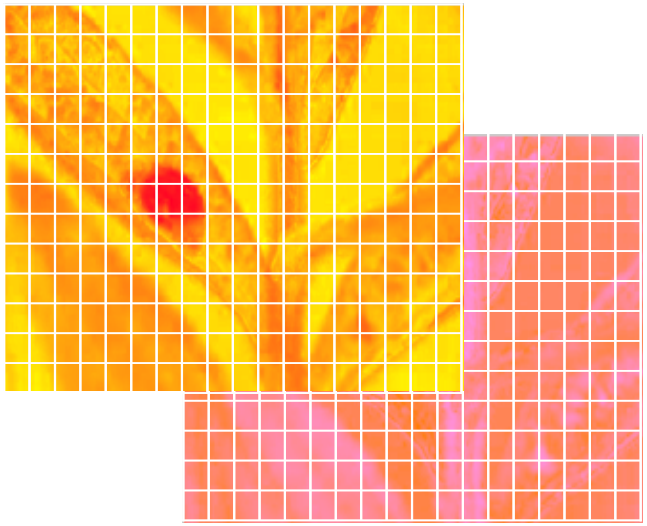
Main Subject Detection in Photographs



Color Distance

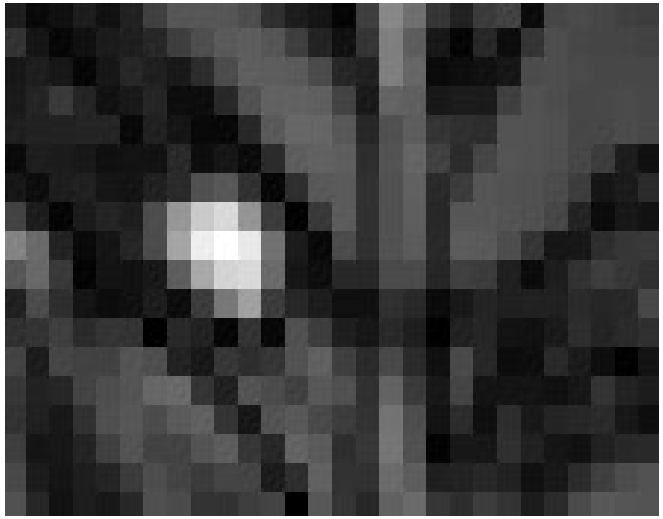
(1) Original image

(2) Convert to *a**, *b**



(3) 8x8 blocks w/ 50% overlap

Measure dist. from avg. block *a**, *b** to avg. bkgnd *a**, *b**

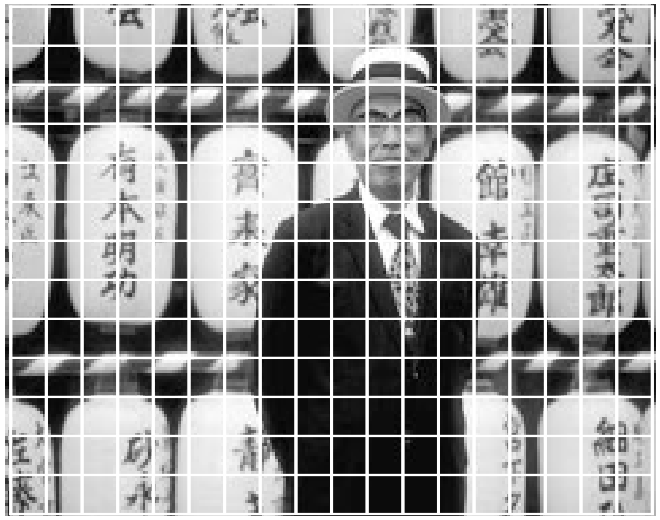


Local color dist. map

Lightness Distance

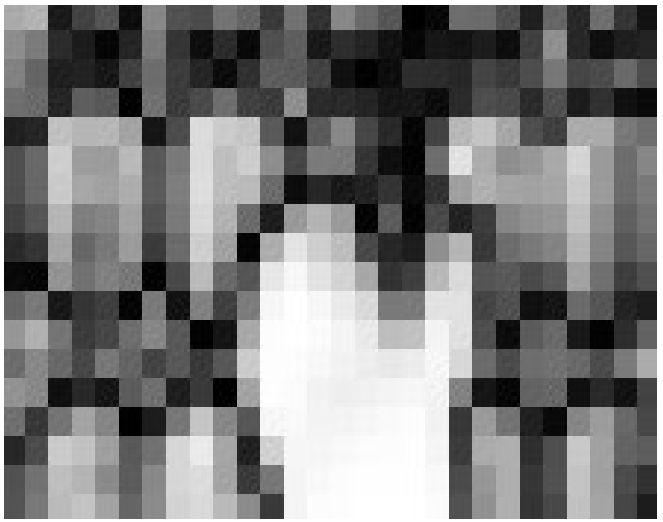
(1) Original image

(2) Convert to *L**



(3) 8x8 blocks w/ 50% overlap

Measure dist. from avg. block *L**, to avg. bkgnd *L**



Local lightness dist. map

Color and Lightness Distance Feature Maps

```
import numpy as np
import cv2 as cv
import ipcv_utils.utils as ipcv_plt

img_bgr = cv.imread("imgs/ladybug.png")
img_bgr = img_bgr.astype(np.float32) / 255

img_rgb = cv.cvtColor(img_bgr, cv.COLOR_BGR2RGB)
ipcv_plt.imshow(img_rgb)

num_rows = img_rgb.shape[0]
num_cols = img_rgb.shape[1]

img_lab = cv.cvtColor(img_rgb, cv.COLOR_RGB2Lab)

avg_img_l = img_lab[:, :, 0].mean()
avg_img_a = img_lab[:, :, 1].mean()
avg_img_b = img_lab[:, :, 2].mean()

l_dist_map = np.zeros((num_rows, num_cols),
  np.float32)
c_dist_map = np.zeros((num_rows, num_cols),
  np.float32)

blk_size = 9
half_blk_size = int(blk_size/2)
dblk = 4

for r in range(0, num_rows, dblk):
  r_bgn = max(0, r - half_blk_size)
  r_end = min(r + half_blk_size, num_rows)

  for c in range(0, num_cols, dblk):
    c_bgn = max(0, c - half_blk_size)
    c_end = min(c + half_blk_size, num_cols)

    blk = img_lab[r_bgn:r_end, c_bgn:c_end]
    avg_blk_l = blk[:, :, 0].mean()
    avg_blk_a = blk[:, :, 1].mean()
    avg_blk_b = blk[:, :, 2].mean()

    # compute the lightness distance
    d = np.abs(avg_blk_l - avg_img_l)
    l_dist_map[r_bgn:r_end, c_bgn:c_end] = d

    # compute the a,b distance
    d = np.sqrt(
      (avg_blk_a - avg_img_a)**2 + \
      (avg_blk_b - avg_img_b)**2
      )
    c_dist_map[r_bgn:r_end, c_bgn:c_end] = d

ipcv_plt.imshow(l_dist_map, vmax=l_dist_map.max())
ipcv_plt.imshow(c_dist_map, vmax=c_dist_map.max())
```

Color and Lightness Distance Feature Maps

Lightness distance map

Color distance map