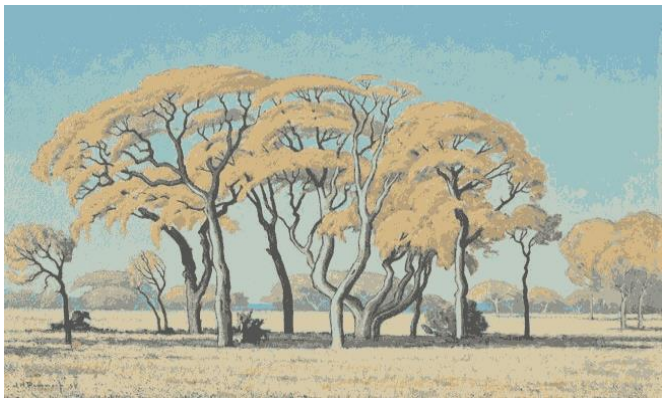
CS 498 AML HW8 REPORT

Pengyu Cheng pcheng11
Haotian Qiu hqiu9
Caiting Wu cwu72

Clarification:

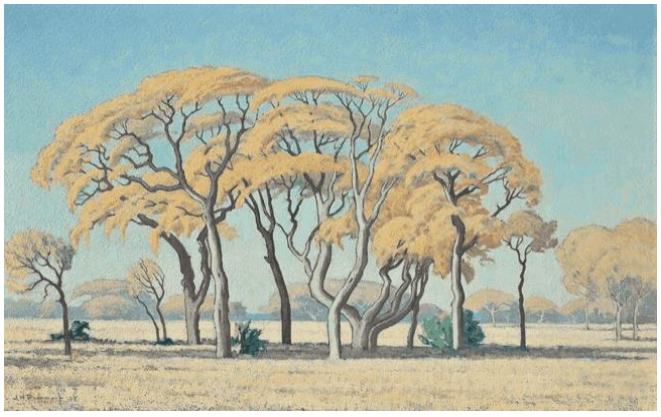
For this assignment, we used K-means as our initialization.

- trees ie2cf92c40c3f3306321d789f7e9c12893.jpg - segmented into 10, 20 and 50 segments
  - 10 segments:



  - 20 segments:

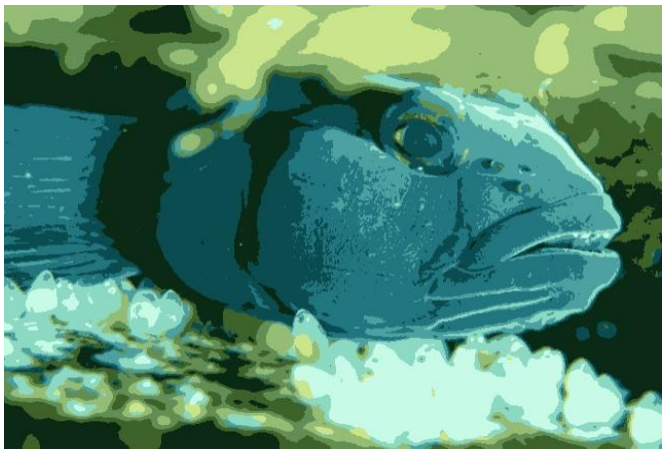
  - 50 segments:



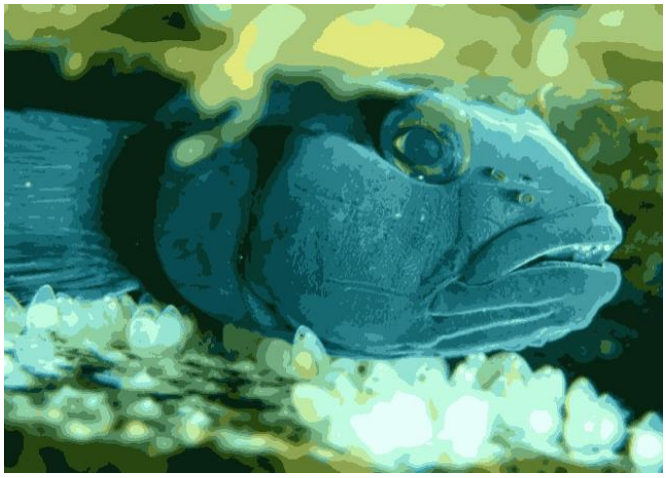
CS 498 AML HW8 REPORT

Pengyu Cheng pcheng11
Haotian Qiu hqiu9
Caiting Wu cwu72

- RobertMixed03.jpg segmented into 10,20,50 segments
  - 10 segments:



  - 20 segments:



  - 50 segments:

CS 498 AML HW8 REPORT
Pengyu Cheng pcheng11
Haotian Qiu hqiu9
Caiting Wu cwu72

- smallstrelitzia.jpg segmented into 10, 20 and 50 segments
  - 10 segments:

  - 20 segments:

  - 50 segments:

- smallsunset.jpg segmented into 10,20 and 50 segment

CS 498 AML HW8 REPORT

Pengyu Cheng pcheng11
Haotian Qiu hqiu9
Caiting Wu cwu72

- 10 segments:

- 20 segments:

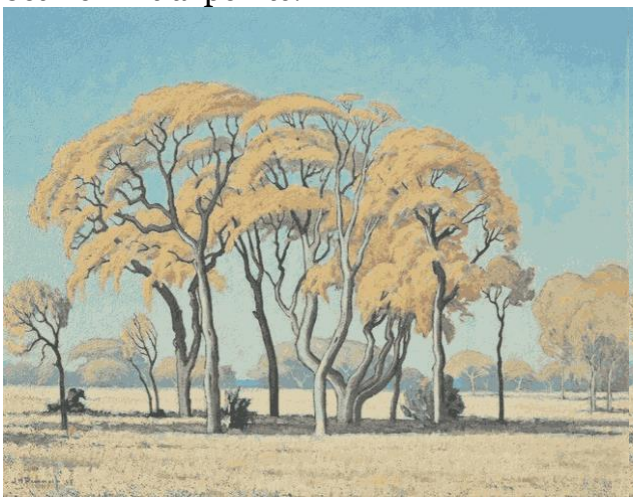
- 50 segments:

CS 498 AML HW8 REPORT

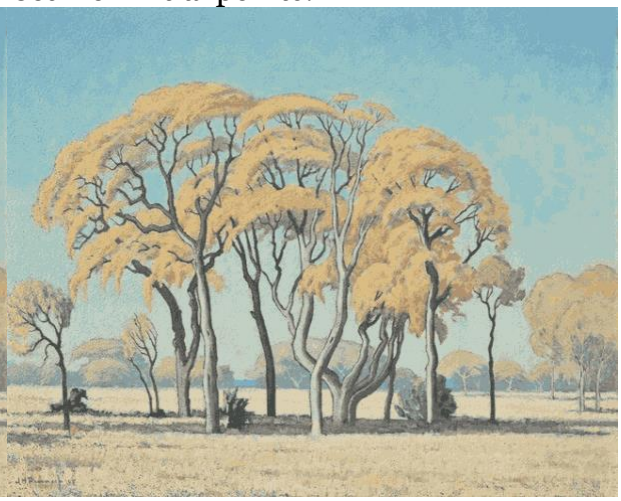
Pengyu Cheng pcheng11
Haotian Qiu hqiu9
Caiting Wu cwu72

Display of tree image with 20 segments with 5 different initial points

Set 1 of initial points:



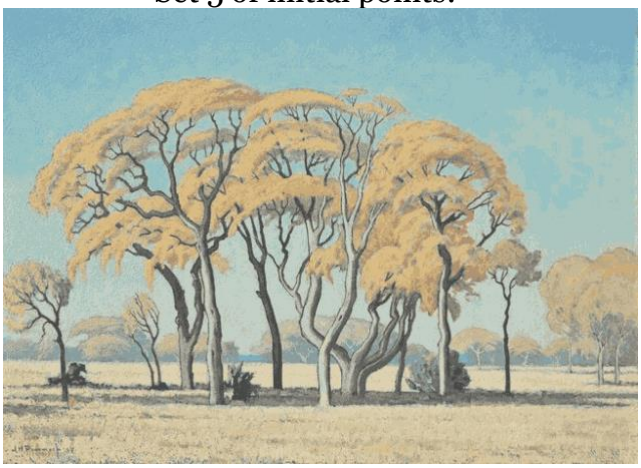
Set 2 of initial points:



Set 3 of initial points:

Set 4 of initial points:

Set 5 of initial points:



CS 498 AML HW8 REPORT
Pengyu Cheng pcheng11
Haotian Qiu hqiu9
Caiting Wu cwu72

- Code snippets for EM initialization, updates

```python
def random_init(k, data, n):
    n = np.random.randint(0, n+1, size = k)
    pi = np.array([(1/k)]*k)
    return data[n, :], pi

def boot_step(k, data, state):
    #initialize cluster center
    kmeans = KMeans(n_clusters=k, random_state=state).fit(data)
    #initialize weight(scala) for each blob of data
    pi = np.array([(1/k)]*k)
    return kmeans.cluster_centers_, pi

def e_step(x, mu, pi, k, n):
    #vectorize all the computations
    w = np.zeros((n, k))
    for i in range(n):
        #for numerical stability add dmin
        dmin = min(np.sum((x[i,:] - mu)**2, axis=1))
        for j in range(k):
            d1 = (x[i,:] - mu[j,:])
            d2 = (x[i,:] - mu[j,:])
            w[i,j] = np.exp(-0.5*(np.dot(d1.T, d2) - dmin)) * pi[j]
    w = w / np.sum(w, axis=1)[:, None]
    return w

def m_step(w, x, mu, pi, k, n, img_dim):
    mu = np.dot(x.T, w).T
    mu = mu / np.sum(w, axis=0)[:,None]
    pi = np.sum(w, axis=0) / (img_dim[0] * img_dim[1])
    # make sure pi sums up to 1:
    # set one probability to (1-all other)
    pi[k-1] = 1 - sum(pi[:k-1])
    return mu, pi
```

CS 498 AML HW8 REPORT

Pengyu Cheng pcheng11
Haotian Qiu hqiu9
Caiting Wu cwu72

- Other relevant code (optional)

```
def assign_pixel(w, image_data, k, n, img_dim):
    pred_cluster = np.argmax(w, axis=1) #nx1
    #get avg pixel value for ith cluster
    avg_pixel_for_cluster = [np.mean(image_data[pred_cluster==i], axis = 0) for i in range(k)]
    #new graph:
    new_graph = np.zeros((n, 3))
    for i in range(n):
        new_graph[i,:] = avg_pixel_for_cluster[pred_cluster[i]]
    new_graph = np.asarray(np.reshape(new_graph, (img_dim[0], img_dim[1], 3)), dtype="uint8")
    return new_graph

def show_graph(new_graph, name, k, state):
    plt.imshow(new_graph)
    plt.imsave(name[:-3] + '_' + str(k) + 'state_' + str(state) + 'segmentations.jpg', new_graph)
```

```
def main():
    random = False
    epsilon = 0.1
    image_path = '/Users/pengyucheng/Desktop/Applied-ML/cs498_aml_hw8/dataset/'


    for image_name in ['RobertMixed03.jpg', 'smallsunset.jpg', 'tree.jpg', 'smallstrelitzia.jpg']:
        print('image:', image_name)
        if image_name == 'tree.jpg':
            k = 20
            for i in [0, 19, 22, 42, 69]:
                image_data, img_dim = load_image(image_path, image_name)
                n = image_data.shape[0]
                if (random):
                    mu, pi = random_init(k, image_data, n)
                else:
                    mu, pi = boot_step(k, image_data, i)
                old_mu = np.Inf
                iter = 1
                while(True):
                    print('iteration:', iter)
                    iter += 1
                    w = e_step(image_data, mu, pi, k, n)
                    mu, pi = m_step(w, image_data, mu, pi, k, n, img_dim)
                    diff = check_convergence(mu, old_mu)
                    old_mu = mu
                    if diff < epsilon:
                        break

                graph = assign_pixel(w, image_data, k, n, img_dim)
                show_graph(graph, image_name, k, i)

        for k in [10, 20, 50]:
            print('segmentation length:', k)
            image_data, img_dim = load_image(image_path, image_name)
            n = image_data.shape[0]
            if (random):
                mu, pi = random_init(k, image_data, n)
            else:
                mu, pi = boot_step(k, image_data, 0)
            old_mu = np.Inf
            iter = 1
            while(True):
                print('iteration:', iter)
                iter += 1
                w = e_step(image_data, mu, pi, k, n)
                mu, pi = m_step(w, image_data, mu, pi, k, n, img_dim)
                diff = check_convergence(mu, old_mu)
                old_mu = mu
                if diff < epsilon:
                    break

            graph = assign_pixel(w, image_data, k, n, img_dim)
            show_graph(graph, image_name, k,  0)
```