Khulna University of Engineering & Technology

A PROGRAMMING ASSIGNMENT ON

Advanced Digital Image Processing

(CSE 6243)

Submitted By	Submitted To
Md. Mahfuzur Rahman ID: 2107501	Dr. Sk. Mohammad Masudul Ahsan Professor, Department of CSE, Khulna University of Engineering & Technology

SUBMITTED DATE: 09.07.2021

Assignment # 02
Topic: Image Segmentation

1. Introduction

Image segmentation is used to separate an image into several “meaningful” parts. It is an old research topic, which started around 1970, but there is still no robust solution toward it. There are two main reasons, the first is that the content variety of images is too large, and the second one is that there is no benchmark standard to judge the performance.

For this assignment, I will use feature-space based image segmentation technique. The feature-space based method is composed of two steps, feature extraction and clustering. Feature extraction is the process to find some characteristics of each pixel or of the region around each pixel, for example, pixel value, pixel color component, windowed average pixel value, windowed variance, Law’s filter feature, Tamura feature, and Gabor wavelet feature, etc. After we get some symbolic properties around each pixel, clustering process is executed to separate the image into several “meaningful” parts based on these properties.

2. Ground Truth Image Design

For generating ground truth images for given image dataset, I used a tool named ImageJ. This is a free and open-source tool.

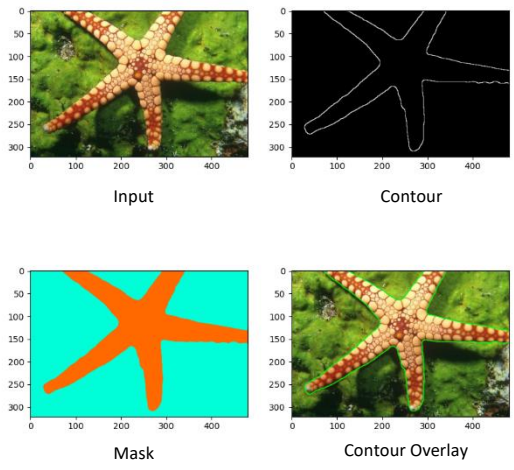


Fig: Img01

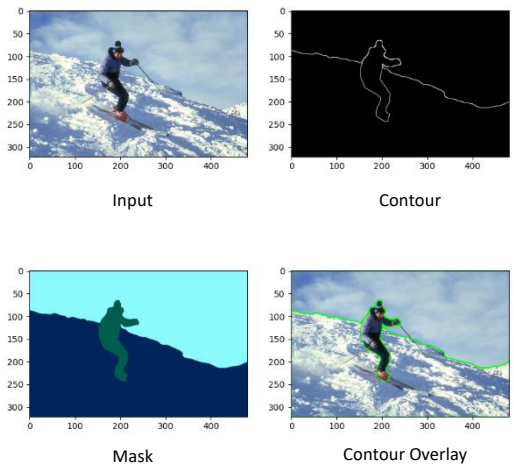


Fig: Img02

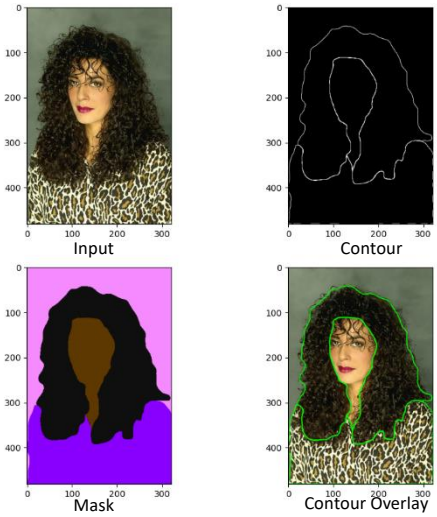


Fig: Img03

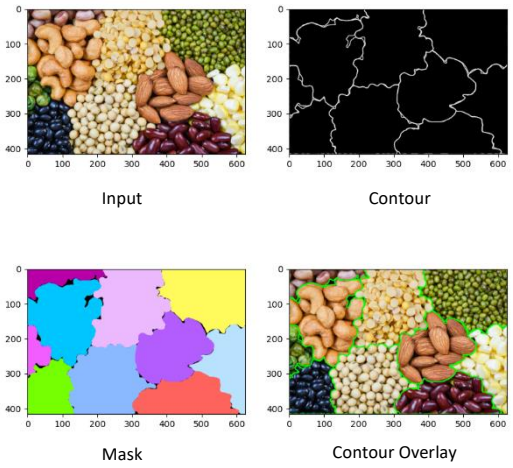


Fig: Img04

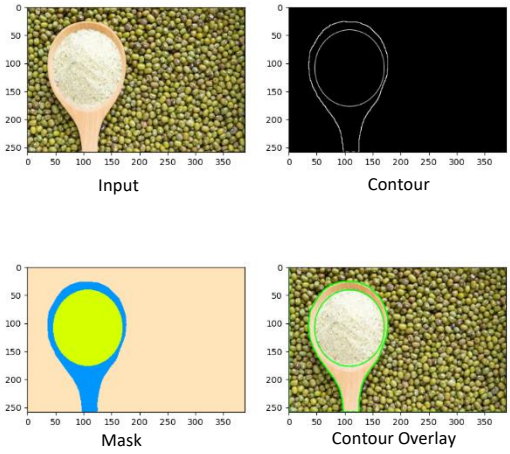


Fig: Img05

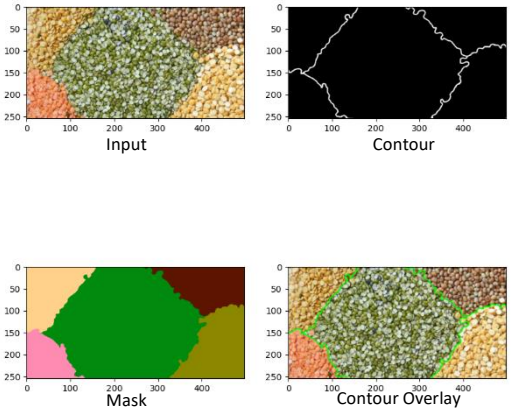


Fig: Img06

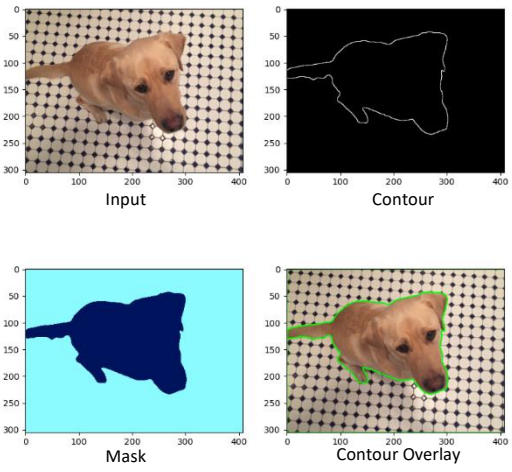


Fig: Img07

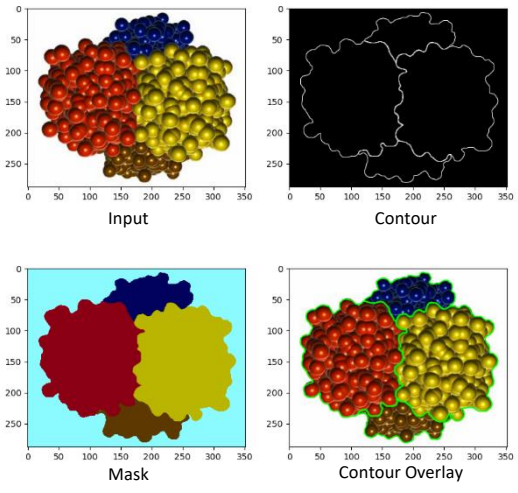


Fig: Img08

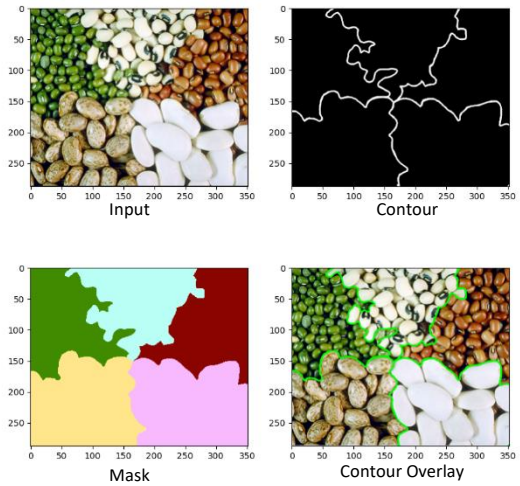


Fig: Img09

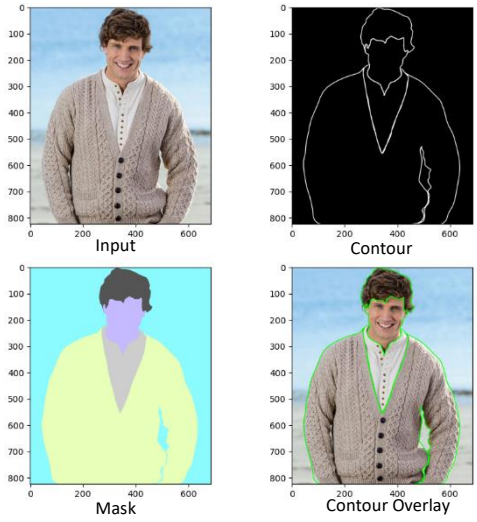


Fig: Img10

3. Convert Images into Feature Space

3.1 Convert into Color Space

Color space usually refers to the combination of hues, tones and shades that are used to recreate a particular color on a certain device.

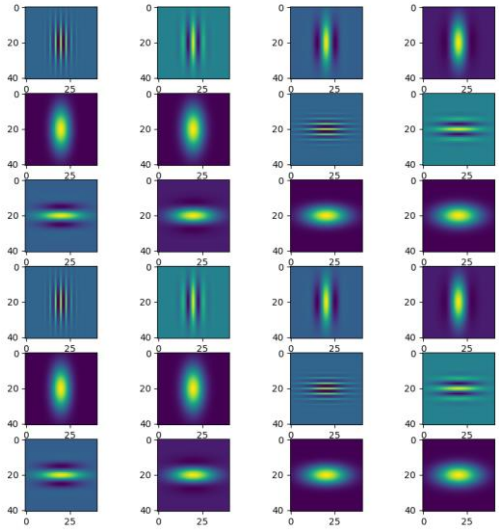
RGB: The RGB color space represents images as an m-by-n-by-3 numeric array whose elements specify the intensity values of the red, green, and blue color channels. The range of numeric values depends on the data type of the image.

HSV: The HSV (Hue, Saturation, Value) color space corresponds better to how people experience color than the RGB color space does. For example, this color space is often used by people who are selecting colors, such as paint or ink color, from a color wheel or palette.

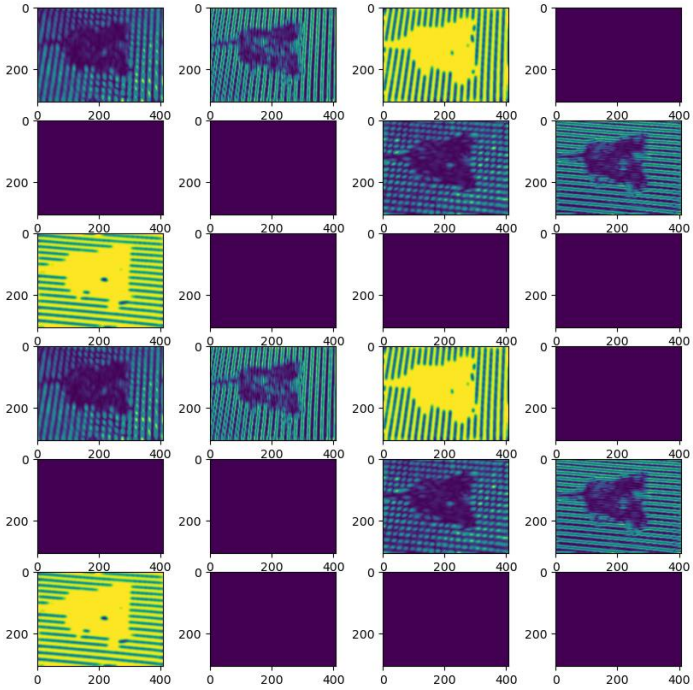
3.2 Texture Segmentation using Gabor Filters

From experimentation, it is known that Gabor filters are a reasonable model of simple cells in the mammalian vision system. Because of this, Gabor filters are thought to be a good model of how humans distinguish texture, and are, therefore, a good model to use when designing algorithms to recognize texture.

For experiments, I used 24 Gabor Kernel.



For experiment, I used an image of a dog with the filename Img07.png & applied gabor filter then I got the result:



Implementing Gabor Filter:

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

image = cv.imread('./assets/img07/Img07.png')
rows, cols, channels = image.shape
gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
```

```
def build_gabor_kernels():
    filters = []
    ksize = 40
    for rotation in orientations:
        for wavelength in wavelengths:
            kernel = cv.getGaborKernel((ksize, ksize),
            4.25, rotation, wavelength, 0.5,
             0, ktype=cv.CV_32F)
            filters.append(kernel)
    return filters

gaborKernels = build_gabor_kernels()
gaborFilters = []
for (i, kernel) in enumerate(gaborKernels):
    filteredImage = cv.filter2D(gray, cv.CV_8UC1,kernel)
    # Blurring the image
    sigma = int(3*0.5*wavelengths[i % len(wavelengths)])
    # Sigma needs to be odd
    if sigma % 2 == 0:
        sigma = sigma + 1
    blurredImage = cv.GaussianBlur(filteredImage,(int(sigma)
    ,int(sigma)),0)
    gaborFilters.append(blurredImage)

numberOfFeatures = 1  + len(gaborKernels) + 2
featureVectors = []
for i in range(0, rows, 1):
    for j in range(0, cols, 1):
        vector = [gray[i][j]]
        for k in range(0, len(gaborKernels)):
            vector.append(gaborFilters[k][i][j])
        vector.extend([i+1, j+1])
        featureVectors.append(vector)
# Normalizing the feature vectors
scaler = preprocessing.StandardScaler()
scaler.fit(featureVectors)
featureVectors = scaler.transform(featureVectors)
```

4. Implementation Algorithm

4.1 Kmeans: K-means clustering is a simple and elegant approach for partitioning a data set into K distinct, non-overlapping clusters. To perform K-means clustering, we must first specify the desired number of clusters K; then the K-means algorithm will assign each observation to exactly one of the K clusters.

```
def kMeans(image, n_clusters):
    from sklearn.cluster import KMeans

    rows, cols, channels = image.shape
    all_pixels = image.reshape(-1, channels)

    km = KMeans(n_clusters=n_clusters)
    km.fit(all_pixels)

    centers = np.array(km.cluster_centers_, dtype='uint8')
    new_img = np.zeros((rows * cols, chan-
nels), dtype='uint8')

    labels = km.labels_
    new_img = centers[labels]

    new_img = new_img.reshape((image.shape))

    return new_img, labels
```

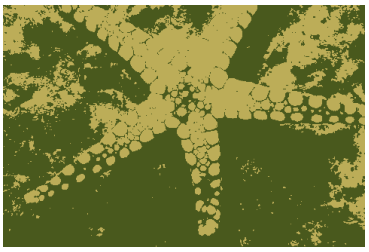
Mean Shift: Mean-shift algorithm basically assigns the datapoints to the clusters iteratively by shifting points towards the highest density of datapoints cluster centroid. The difference between K-Means algorithm and Mean-Shift is that later one does not need to specify the number of clusters in advance because the number of clusters will be determined by the algorithm w.r.t data.

```
def meanShift(image):
    from sklearn.cluster import MeanShift

    print("started Mean Shift Algorithm")
    originShape = image.shape
    flatImg=np.reshape(image, [-1, 3])

    ms = MeanShift(bandwidth = bandwidth, bin_seeding=True)
    ms.fit(flatImg)
    labels=ms.labels_
    cluster_centers = ms.cluster_centers_
    # Finding and diplaying the number of clusters
    labels_unique = np.unique(labels)
    n_clusters_ = len(labels_unique)
    print("number of estimated clusters : %d" % n_clus-
ters_)
    segmentedImg = cluster_centers[np.reshape(labels, ori-
ginShape[:2])]
    segmentedImg = segmentedImg.astype(np.uint8)
    return segmentedImg, labels
```

5. Experimental Results

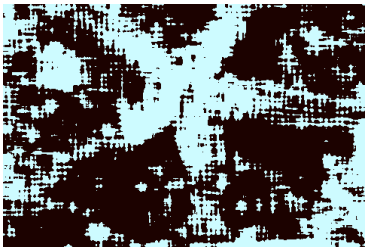
Result on Img01

Input

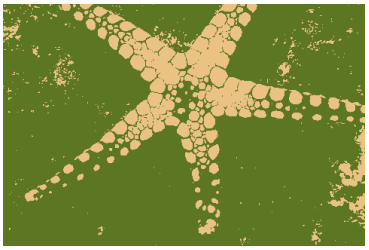


Segmented With K-Means using RGB Color Space

Segmented With K-Means using LAB Color Space



Segment With K-Means using Gabor Filter

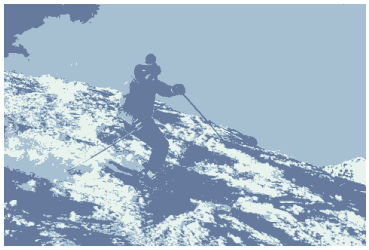


Segmented With Mean Shift using RGB Color Space

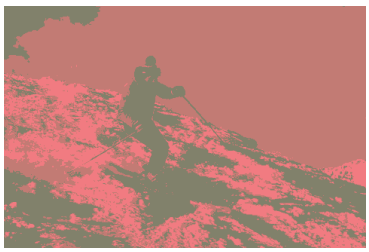
Segmented With Mean Shift using LAB Color Space

Result on Img02

Input



Segmented With K-Means using RGB Color Space



Segmented With K-Means using LAB Color Space

Segment With K-Means using Gabor Filter

Segmented With Mean Shift using RGB Color Space

Segmented With Mean Shift using LAB Color Space

Result on Img03

Input

Segmented With K-Means using RGB Color Space

Segmented With K-Means using LAB Color Space

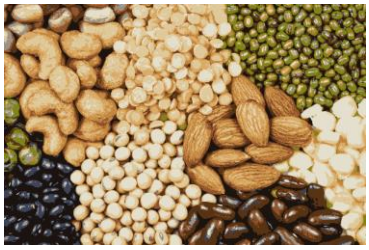
Segment With K-Means using Gabor Filter

Segmented With Mean Shift using RGB Color Space

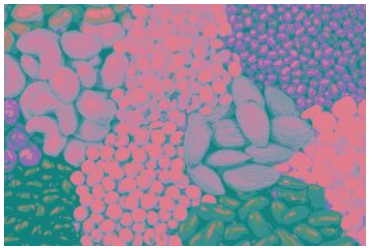
Segmented With Mean Shift using LAB Color Space

Result on Img04

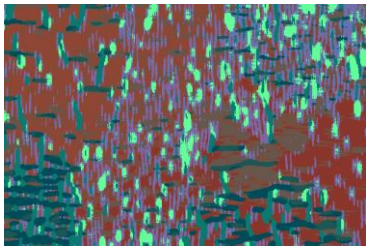
Input



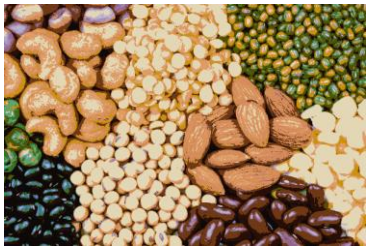
Segmented With K-Means using RGB Color Space



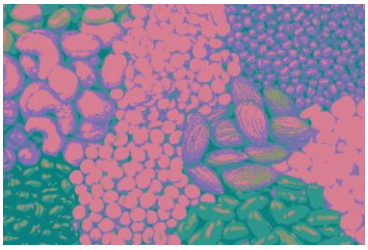
Segmented With K-Means using LAB Color Space



Segment With K-Means using Gabor Filter



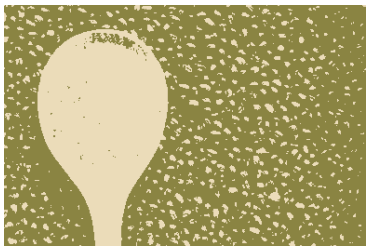
Segmented With Mean Shift using RGB Color Space



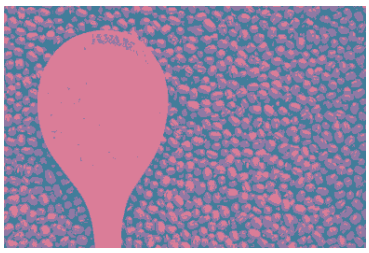
Segmented With Mean Shift using LAB Color Space

Result on Img05

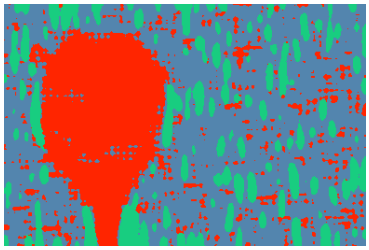
Input



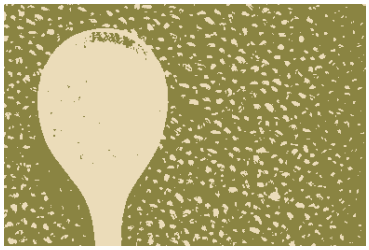
Segmented With K-Means using RGB Color Space



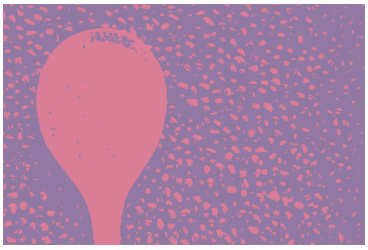
Segmented With K-Means using LAB Color Space



Segment With K-Means using Gabor Filter

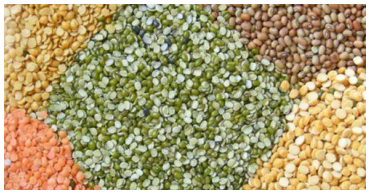


Segmented With Mean Shift using RGB Color Space

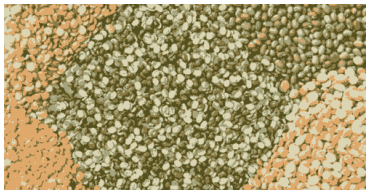


Segmented With Mean Shift using LAB Color Space

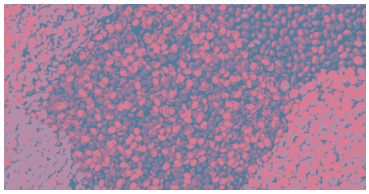
Result on Img06



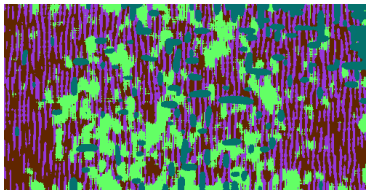
Input



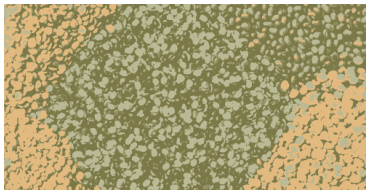
Segmented With K-Means using RGB Color Space



Segmented With K-Means using LAB Color Space



Segment With K-Means using Gabor Filter

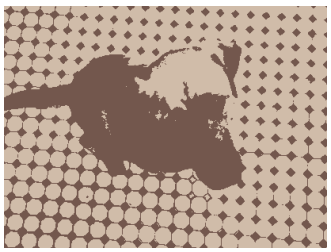


Segmented With Mean Shift using RGB Color Space

Segmented With Mean Shift using LAB Color Space

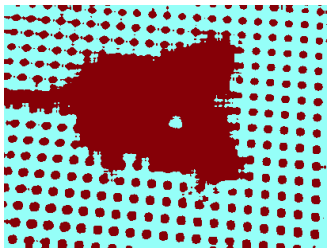
Result on Img07

Input

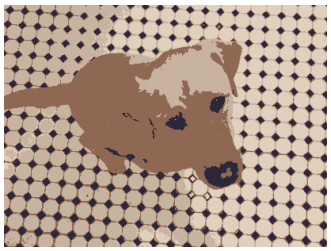


Segmented With K-Means using RGB Color Space

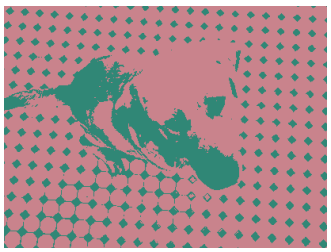
Segmented With K-Means using LAB Color Space



Segment With K-Means using Gabor Filter

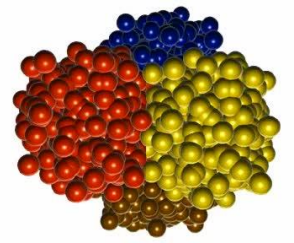


Segmented With Mean Shift using RGB Color Space

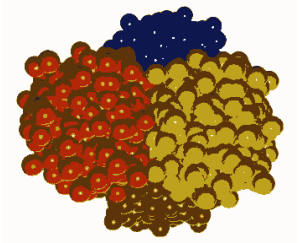


Segmented With Mean Shift using LAB Color Space

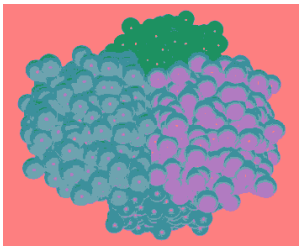
Result on Img08



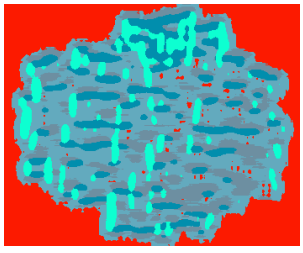
Input



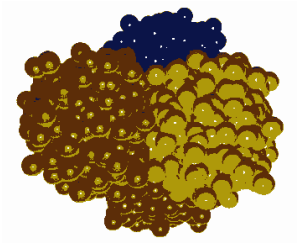
Segmented With K-Means using RGB Color Space



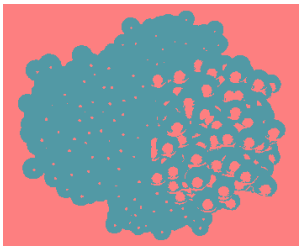
Segmented With K-Means using LAB Color Space



Segment With K-Means using Gabor Filter



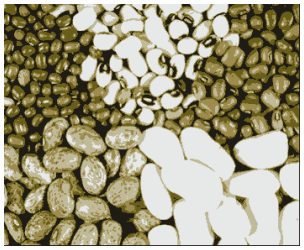
Segmented With Mean Shift using RGB Color Space



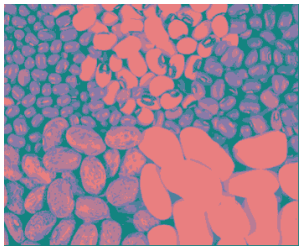
Segmented With Mean Shift using LAB Color Space

Result on Img09

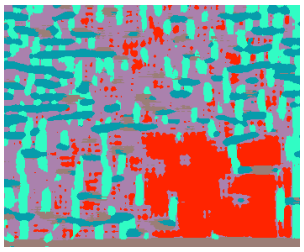
Input



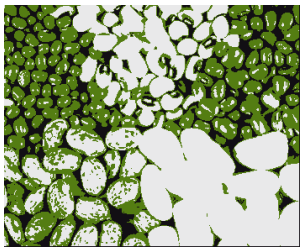
Segmented With K-Means using RGB Color Space



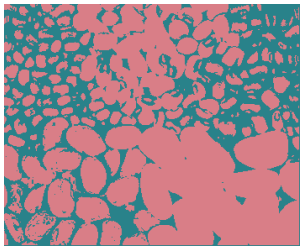
Segmented With K-Means using LAB Color Space



Segment With K-Means using Gabor Filter



Segmented With Mean Shift using RGB Color Space



Segmented With Mean Shift using LAB Color Space

Result on Img10

Input

Segmented With K-Means using RGB Color Space

Segmented With K-Means using LAB Color Space

Segment With K-Means using Gabor Filter

Segmented With Mean Shift using RGB Color Space

Segmented With Mean Shift using LAB Color Space

6. Comparing the Results

Comparison of machine generated segmentation result with GT images:

Image name	Algorithm	Filters	Bandwidth/ Cluster Value	Precision	Recall	fscore	IoU	Peer GT Images			
								Precision	Recall	fscore	IoU
img01.jpg	kmeans	RGB	2	72.62%	76.33%	0.74	59.27%	72.72%	76.55%	0.74	59.48%
img01.jpg	kmeans	LAB	2	72.62%	76.33%	0.74	59.27%	72.72%	76.55%	0.74	59.48%
img01.jpg	kmeans	GABOR	2	33.51%	30.17%	0.28	18.87%	33.46%	30.04%	0.28	18.81%
img01.jpg	Mean Shift	RGB	80	85.83%	77.84%	0.81	68.98%	86.00%	78.07%	0.81	69.27%
img01.jpg	Mean Shift	LAB	50	39.20%	37.54%	0.38	23.73%	39.03%	37.30%	0.38	23.57%
img02.jpg	Mean Shift	RGB	60	72.83%	53.98%	0.39	44.93%	26.79%	45.95%	0.34	20.37%
img02.jpg	Mean Shift	LAB	30	18.34%	20.01%	0.19	10.58%	18.30%	19.97%	0.19	10.56%
img02.jpg	kmeans	RGB	3	36.53%	16.27%	0.19	12.68%	49.69%	45.42%	0.45	31.11%
img02.jpg	kmeans	LAB	3	11.39%	11.28%	0.11	6.01%	11.41%	11.60%	0.11	6.10%
img02.jpg	kmeans	GABOR	3	36.10%	24.29%	0.29	16.98%	37.62%	21.09%	0.26	15.63%
img03.jpg	kmeans	RGB	4	37.06%	43.39%	0.40	24.98%	37.22%	45.53%	0.41	25.76%
img03.jpg	kmeans	LAB	4	37.06%	43.39%	0.40	24.98%	37.49%	53.40%	0.44	28.24%
img03.jpg	kmeans	GABOR	4	30.92%	35.47%	0.31	19.79%	30.41%	36.25%	0.30	19.82%
img03.jpg	Mean Shift	RGB	40	51.40%	45.94%	0.45	32.03%	14.88%	20.14%	0.17	9.36%
img03.jpg	Mean Shift	LAB	40	34.23%	35.13%	0.31	20.97%	20.71%	10.87%	0.14	7.68%
img04.jpg	kmeans	RGB	10	17.03%	18.96%	0.16	9.86%	10.78%	8.28%	0.09	4.91%
img04.jpg	kmeans	LAB	10	16.72%	16.62%	0.14	9.09%	14.13%	13.70%	0.13	7.47%
img04.jpg	kmeans	GABOR	10	8.28%	8.66%	0.08	4.42%	16.08%	17.23%	0.16	9.07%
img04.jpg	Mean Shift	RGB	20	3.58%	2.26%	0.02	1.40%	3.94%	5.24%	0.04	2.30%
img04.jpg	Mean Shift	LAB	20	2.45%	3.38%	0.03	1.44%	4.99%	4.37%	0.04	2.38%
img05.jpg	kmeans	RGB	3	41.90%	44.13%	0.33	27.38%	49.82%	50.62%	0.42	33.53%
img05.jpg	kmeans	LAB	3	41.90%	44.13%	0.33	27.38%	49.82%	50.62%	0.42	33.53%
img05.jpg	kmeans	GABOR	3	49.19%	57.10%	0.49	35.92%	49.53%	43.57%	0.34	30.17%
img05.jpg	Mean Shift	RGB	40	47.20%	62.19%	0.51	36.67%	47.55%	62.25%	0.51	36.91%
img05.jpg	Mean Shift	LAB	40	46.05%	61.02%	0.49	35.58%	46.35%	61.08%	0.49	35.78%
img06.jpg	kmeans	RGB	5	19.09%	12.20%	0.13	8.04%	21.55%	20.60%	0.19	11.77%
img06.jpg	kmeans	LAB	5	17.76%	9.90%	0.11	6.79%	19.03%	12.23%	0.13	8.04%
img06.jpg	kmeans	GABOR	5	17.71%	18.26%	0.11	9.88%	25.58%	27.93%	0.18	15.41%
img06.jpg	Mean Shift	RGB	40	13.10%	7.92%	0.08	5.19%	16.91%	15.51%	0.15	8.80%
img07.png	kmeans	RGB	2	72.51%	77.33%	0.74	59.80%	72.59%	77.62%	0.74	60.03%
img07.png	kmeans	LAB	2	72.51%	77.33%	0.74	59.80%	72.59%	77.62%	0.74	60.03%
img07.png	kmeans	GABOR	2	76.90%	84.52%	0.77	67.41%	76.90%	84.77%	0.77	67.56%
img07.png	Mean Shift	RGB	40	21.52%	10.13%	0.14	7.40%	28.45%	13.76%	0.17	10.22%
img07.png	Mean Shift	LAB	40	64.39%	63.70%	0.64	47.11%	64.47%	63.88%	0.64	47.25%
img08.jpg	kmeans	RGB	5	40.88%	48.59%	0.40	28.53%	36.36%	31.89%	0.34	20.47%
img08.jpg	kmeans	LAB	5	36.63%	31.85%	0.34	20.54%	36.36%	31.89%	0.34	20.47%
img08.jpg	kmeans	GABOR	5	19.39%	25.49%	0.21	12.38%	37.97%	34.05%	0.34	21.88%
img08.jpg	Mean Shift	RGB	30	66.13%	68.59%	0.67	50.75%	15.93%	13.33%	0.14	7.83%
img08.jpg	Mean Shift	LAB	30	2.98%	5.74%	0.04	2.00%	11.97%	25.18%	0.16	8.83%

Image name	Algorithm	Filters	Bandwidth/ Cluster Value	Precision	Recall	fscore	IoU	Peer GT Images			
								Precision	Recall	fscore	IoU
img09.png	kmeans	RGB	5	27.04%	28.31%	0.27	16.05%	33.33%	36.04%	0.34	20.95%
img09.png	kmeans	LAB	5	27.04%	28.31%	0.27	16.05%	15.40%	14.67%	0.15	8.12%
img09.png	kmeans	GABOR	5	36.95%	34.16%	0.34	21.58%	20.70%	18.76%	0.18	10.92%
img09.png	Mean Shift	RGB	40	21.67%	36.63%	0.27	15.76%	16.79%	26.82%	0.20	11.51%
img09.png	Mean Shift	LAB	60	13.63%	31.28%	0.19	10.49%	10.69%	25.66%	0.15	8.16%
img10.jpg	kmeans	RGB	4	37.97%	32.70%	0.31	21.31%	46.57%	46.07%	0.29	30.14%
img10.jpg	kmeans	LAB	4	24.77%	19.29%	0.20	12.16%	25.75%	9.03%	0.06	7.16%
img10.jpg	kmeans	GABOR	4	44.51%	34.81%	0.32	24.28%	25.90%	16.55%	0.17	11.23%
img10.jpg	Mean Shift	RGB	40	29.28%	24.62%	0.26	15.44%	49.12%	50.73%	0.44	33.25%
img10.jpg	Mean Shift	LAB	20	41.05%	46.83%	0.44	28.00%	45.27%	45.91%	0.45	29.53%

7. References

- https://www.mathworks.com/help/images/ref/imsegkmeans.html
- https://www.mathworks.com/help/images/texture-segmentation-using-gabor-filters.html
- Making Ground Truth for cell segmentation with ImageJ on http://youtube.com
- Factorization-Based Texture Segmentation on https://sites.google.com/site/factorizationsegmentation/
- Image Segmentation using K-means on http://youtube.com
- K-Means Clustering for Image Segmentation on https://www.thepythoncode.com/article/kmeans-for-image-segmentation-opencv-python