German University in Cairo
Faculty of Media Engineering and Technology
Winter 2022

DMET 901 – Computer Vision

# Assignment #2 and #3 (15%)

(Due on December 29th, 2022 at midnight)

This is an intensive assignment (assignment 2 and 3 together as a mini-project) where you will use clustering algorithms to segment images. You will then use these segmentations to identify foreground and background objects.

Your assignment will involve the following subtasks:

**Clustering algorithms**: Implement K-Means clustering (and Hierarchical Agglomerative Clustering (optional)).

**Pixel-level features**: Implement a feature vector that combines color and position information and implement feature normalization.

**Quantitative Evaluation**: Evaluate segmentation algorithms with a variety of parameter settings by comparing your computed segmentations against a dataset of ground-truth segmentations.

## 1. Clustering Algorithms

### 1.1 K-Means Clustering

As discussed in class, K-Means is one of the most popular clustering algorithms. We have provided skeleton code for K-Means clustering in the file **segmentation.py**. Your first task is to finish implementing **kmeans** in segmentation.py. This version uses nested for loops to assign points to the closest centroid and compute a new mean for each cluster.

We can use numpy functions and broadcasting to make K-Means faster.
Implement **kmeans_fast** in segmentation.py. This should run at least 10 times faster than the previous implementation.

### 1.2 Hierarchical Agglomerative Clustering (Optional (no marks))

Another simple clustering algorithm is Hieararchical Agglomerative Clustering, which is sometimes abbreviated as HAC. In this algorithm, each point is initially assigned to its own cluster. Then cluster pairs are merged until we are left with the desired number of predetermined clusters (see Algorithm 1).
Implement **hiererachical_clustering** in segmentation.py.

**Algorithm 1** Hierarchical Agglomerative Clustering

```
Require: Points $x_1, \dots, x_m \in \mathbb{R}^n$, desired number of clusters $k \in \mathbb{Z}$
  Assign each point to its own cluster
  while There are more than $k$ clusters do
      Compute the distance between all pairs of clusters
      Merge the two closest clusters
  end while
```

German University in Cairo
Faculty of Media Engineering and Technology
Winter 2022

DMET 901 – Computer Vision

# Assignment #2 and #3 (15%)

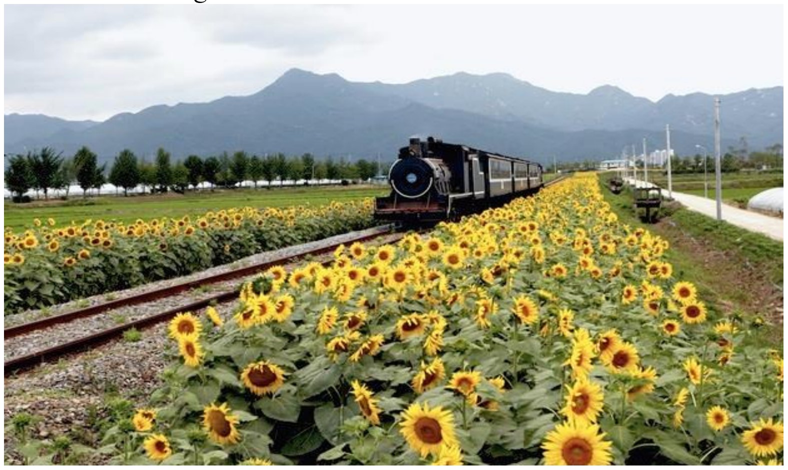
(Due on December 29th, 2022 at midnight)

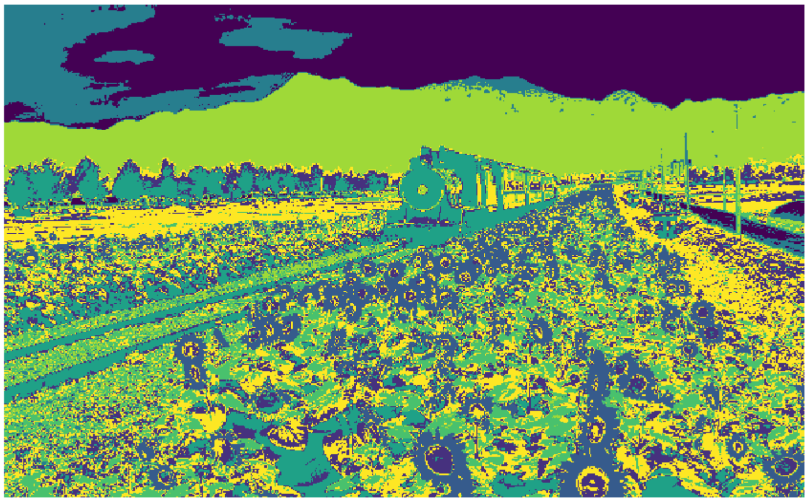
## 2. Pixel-Level Features:

Before we can use a clustering algorithm to segment an image, we must compute some feature vector for each pixel. The feature vector for each pixel should encode the qualities that we care about in a good segmentation. More concretely, for a pair of pixels pi and pj with corresponding feature vectors fi and fj, the distance between fi and fj should be small if we believe that pi and pj should be placed in the same segment and large otherwise.

### 2.1 Color Features

One of the simplest possible feature vectors for a pixel is simply the vector of colors for that pixel.
Implement **color_features** in segmentation.py.
Output should look like the following:





German University in Cairo
Faculty of Media Engineering and Technology
Winter 2022

DMET 901 – Computer Vision

# Assignment #2 and #3 (15%)

(Due on December 29th, 2022 at midnight)

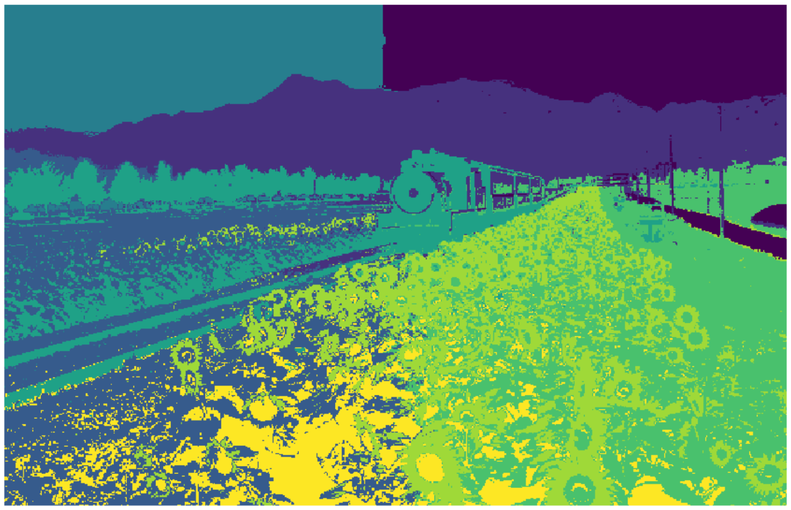
## 2.2 Color and Position Features

Another simple feature vector for a pixel is to concatenate its color and position within the image. In other words, for a pixel of color (r,g,b) located at position (x,y) in the image, its feature vector would be (r,g,b,x,y). However, the color and position features may have drastically different ranges; for example each color channel of an image may be in the range [0,1), while the position of each pixel may have a much wider range. Uneven scaling between different features in the feature vector may cause clustering algorithms to behave poorly.

One way to correct for uneven scaling between different features is to apply some sort of normalization to the feature vector. One of the simplest types of normalization is to force each feature to have zero mean and unit variance.

Implement **color_position_features** in segmentation.py.

Output segmentation should look like the following:



## 3. Quantitative Evaluation

Looking at images is a good way to get an idea for how well an algorithm is working, but the best way to evaluate an algorithm is to have some quantitative measure of its performance.

For this project we have supplied a small dataset of cat images and ground truth segmentations of these images into foreground (cats) and background (everything else). We will quantitatively evaluate different segmentation methods (features and clustering methods) on this dataset.

We can cast the segmentation task into a binary classification problem, where we need to classify each pixel in an image into either foreground (positive) or background (negative). Given the ground-truth labels, the accuracy of a segmentation is (TP+TN)/(P+N).

Implement **compute_accuracy** in segmentation.py.

German University in Cairo
Faculty of Media Engineering and Technology
Winter 2022

DMET 901 – Computer Vision

# Assignment #2 and #3 (15%)

(Due on December 29th, 2022 at midnight)

## Extra Credit:

After completing the required tasks, observe your results carefully and try to answer the following questions:

1. Based on your quantitative experiments, how do each of the segmentation parameters affect the quality of the final foreground-background segmentation?
2. Are some images simply more difficult to segment correctly than others? If so, what are the qualities of these images that cause the segmentation algorithms to perform poorly?
3. Also feel free to point out or discuss any other interesting observations that you made.

## Submission Guidelines:

- Submit your solution on the form: **https://forms.gle/kTkEQpV91Fmb9f9G8** and name the zip file by the format [*Txx_46_xxxx_Txx_43_xxxx.zip*].
- This assignment can be done in groups of maximum 2 students.

Good Luck!