1. K-means and image sharpening

1.1 Introduction

In this exercise, you are asked to approximate the *depth of field* (DOF) effect on an image. The *depth of field* is the distance between the nearest and farthest objects in a scene which appear acceptably sharp in an image. When the DOF is small, the object of interest (i.e., focused object) appears sharp and the rest appears blurred. This effect is usually employed for artistic reasons and is achieved by having a large aperture while capturing an image.
In our case, the image has already been captured, and we are trying to simulate a similar effect. The task is to first segment the image into 3 areas (front/middle/background areas), and then to sharpen the object of interest while blurring the rest.

1.2 Instructions

In our case in addition to the RGB image, a depth image has also been captured. The intensity of each pixel in the depth image correspond to the inverse distance between the camera and the objects of the scene (up to a scale). Instead of finding the correct segmentation of the image, which is a difficult task, we are going to perform the segmentation on the depth image. The depth image, however, is usually a bit noisy, so a simple thresholding does not produce good results in most cases, thus you will need to implement a k-means segmentation algorithm.

1.3 Goals

In this exercise, you are asked to:

- implement the k-means algorithm to segment the depth image into three disjoint regions. This will produce a label image where the three objects (doll, hand, background) have a different label.
- generate three binary images each focusing on a different object.
- sharpen the focused object and smooth the rest, using these binary images and appropriate filters.

(a) (b)

Figure 1.1: Pair of images, captured by Microsoft Kinect. (a) depicts the RGB image and (b) shows the corresponding depth image.

1.4 Exercise

K-Means is a least-squares partitioning method that divides a collection of N-dimensional points into K groups. You are asked to segment the depth image from Fig. 1.1. In this simple example, we are looking for 3 clusters.

Assume we are given a pair of an original gray-scale image $I[i, j]$ and a corresponding depth image $D[i, j]$, of size $H \times W$, such that $1 \leq i \leq H$, and $1 \leq j \leq W$. We are also given the initial values for the clusters: $k^l = (k_d^l, k_i^l, k_j^l), l = 1, 2, 3$, where k_d^l corresponds to the mean depth of the l^{th} cluster and k_i^l and k_j^l to its mean spatial location along i and j coordinate respectively.

Exercise 1.1 Depth-based segmentation.
You are asked to implement the following steps:

1. $\forall$ pixels $(i, j) : 1 \leq i \leq H, 1 \leq j \leq W$ of the depth image, compute the Euclidean distance between its depth and each cluster mean: $dist_l = \sqrt{(D[i, j] - k_d^l)^2}, l = 1, 2, 3$. For each pixel find the smallest distance and assign this pixel to the corresponding cluster.
2. Compute the mean of each cluster.
3. Iterate over the above two steps till the cluster centroids (mean of the clusters) do not move any more or a fixed number of iterations is reached.
4. What is the disadvantage of the result? Is the segmentation correct?

■

We also want to take into account the spatial proximity of the pixels. This can be accomplished by adding 2 dimensions to the original depth image representing the i and j coordinates of each pixel.

Exercise 1.2 Using location information.
The algorithm for this case looks very similar:

1. $\forall$ pixels $(i, j) : 1 \leq i \leq H, 1 \leq j \leq W$ of the depth image, compute the Euclidean distance between its depth and each cluster mean: $dist_l = \sqrt{(D[i, j] - k_d^l)^2 + (i - k_i^l)^2 + (j - k_j^l)^2}, l = 1, 2, 3$. For each pixel find the smallest distance and assign this pixel to the corresponding cluster.
2. Compute the mean of each cluster.
3. Iterate over the above two steps till the cluster centroids (mean of the clusters) do not move any more or a fixed number of iterations is reached.
4. How does addition of spatial information improve the segmentation results?

■

In this toy example we do not use normalization of the values of the depth image and the distances, because there exist many ways of doing it and every single method may lead to different segmentation results. In general for real application normalization needs to be performed, before the k-means algorithm is applied.

We further need to implement a region-based filtering function to sharpen the object of interest and blur the rest of the image. In the `main` function, three binary images are generated from the label image computed in the first part, each image focusing on a different object. If you didn't complete the first part, you can use the pregenerated label image provided in the package by uncommenting the corresponding line in the `main` file.

Exercise 1.3 Your task is to sharpen the focused regions by applying a Laplacian filter and smooth the other regions using a Gaussian filter.

1. The Laplacian filter can be modeled by a single parameter α, as follows:

$$L = \nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} = \frac{4}{(\alpha+1)} \begin{bmatrix} \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \\ \frac{1-\alpha}{4} & -1 & \frac{1-\alpha}{4} \\ \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \end{bmatrix} \tag{1.1}$$

   The sharpening filter is then defined as $\delta - L$, where is a 3x3 Kronecker delta kernel with a value of 1 at the center position and zero elsewhere.
2. the Gaussian filter is parametrized over the standard deviation, σ, of the Gaussian function.
3. For this exercise set α to 0.5 and σ to 2.0. As for the kernel filters, we use a 3×3 pixel size kernel for the sharpening filter and a 7×7 kernel for the smoothing one.
   Hint: in order to generate the kernels, you can either implement/use your own functions or use built-in MATLAB functions.

   You need to implement the above filtering step by filling in the body of the function `FilterRegions()`, the parameters of which are described within the file `FilterRegions.m`.

■

Fig. 1.2 depicts sample filtering output.

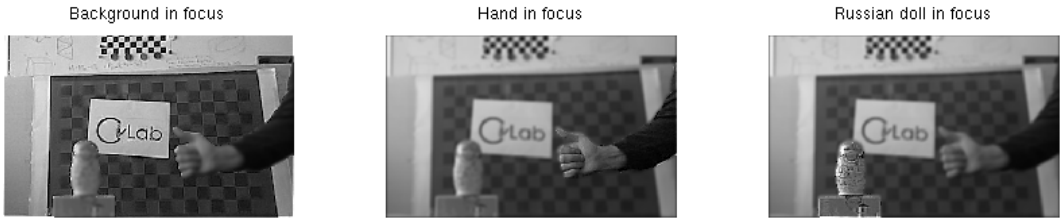

Figure 1.2: Results of the *depth of field* effect.