Instruction

This lab has two parts. For each part, implement the outlined functions in `lab4.py` and run them in the notebook `lab4.ipynb`. For open questions, write your answers directly into the notebook. Expected outputs are supplied as reference, but your outputs do not need to match exactly.

Upload to Canvas your completed `lab4.py` and `lab4.ipynb` by zipping them into a file named AXXX1_AXXX2.zip, where AXXX is the student number of the group members. Submit one file per group. Missing files, incorrectly formatted code that does not run, etc. will be penalized.

He Qiyuan is the TA in charge of this lab. Please post any questions onto the Canvas forum or attend one of the FAQ sessions during the Lab session on Apr. 15 or 17.

Objective

In Lecture 6, we learned how to build textons to describe textures. In Lecture 10, we learned how to describe the motion (displacement) of each image pixel. In this lab, we will explore how to combine these two visual features for segmentation on video.

In the first part, you will implement the Horn-Schunck optical flow and use the motion as the feature for segmentation. In the second part, you will learn textons from a given training dataset and describe the textures using a texton histogram. In part three, you will combine these two features for segmentation on video.

Part 1: Optical Flow (50%)

An initial idea is to use k-means clustering on RGB values directly. However, due to the complicated background, such segmentation is highly undesired. We have provided the code for this part for your reference.

Notice that the motion from the foreground object is largely distinct from the background. Therefore, a natural idea is to compute Horn-Schunck Optical Flow and use it as the feature for k-means clustering.

Horn-Schunck Optical Flow (40%)

- `calcOpticalFlowHS()` function computes Horn-Schunck Optical Flow with parameters `lambda` for weighting hyperparameter and `delta` for convergence threshold.
  1. Precompute image gradients I_y, I_x.
  2. Precompute temporal gradients I_t.
  3. Initialize flow field $u = 0; v = 0$.
  4. Until convergence, compute flow field updates for each pixel:

$$\hat{u}_{kl} = \bar{u}_{kl} - \frac{I_x\bar{u}_{kl} + I_y\bar{v}_{kl} + I_t}{\lambda^{-1} + I_x^2 + I_y^2} I_x \quad \hat{v}_{kl} = \bar{v}_{kl} - \frac{I_x\bar{u}_{kl} + I_y\bar{v}_{kl} + I_t}{\lambda^{-1} + I_x^2 + I_y^2} I_y$$

- With only optical flow as the feature for segmentation, you can correctly segment moving objects. However, without appearance information, the static background cannot be semantically segmented. In the next step, you will combine motion and appearance information together for segmentation.

Feature Combination (5%)

- In `lab4.py`, we provide the helper function to combine two features together as `combine_and_normalize_features()` with the weighting parameter `gamma`. Giving the first feature F_1 and the second feature F_2, the function returns the concatenated feature $[N(F_1), \gamma N(F_2)]$, where $N(\cdot)$ stands for normalization operation. In this section, you will combine RGB values and flow features.

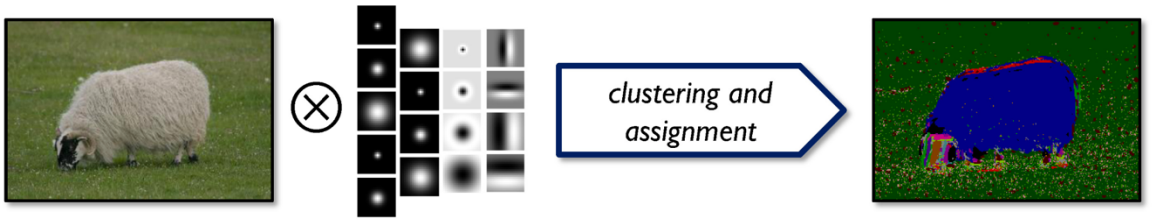
- Based on the helper function, adjust `gamma` for better results and add appearance information such that the white wall is correctly segmented from the green leaves.
- You will notice that the leaves and their reflection on the water are still hard to segment due to the complex pattern. In the next part, you will learn how to describe these textures.

Exploratory questions (10%)

- (6%) How do the parameters `lambda` and `delta` affect the optical flow results?
- (4%) How does the weighting parameter `gamma` affect the final segmentation results?

Part 2: Textons (45%)

In this section, a visual dictionary is introduced to facilitate the segmentation of textured backgrounds. This process is referred to as textonization and is depicted in the following figure.



Building Filter Bank

To finish the following section, you need to firstly understand the helper function `features_from_filter_bank()` in The image is convolved with a 17-dimensional filter-bank at scale κ. The filter-bank consists of Gaussians at scales κ, 2κ and 4κ, x and y derivatives of Gaussians at scales 2κ and 4κ, and Laplacians of Gaussians at scales κ, 2κ, 4κ and 8κ. The Gaussians are applied to all three color channels, while the other filters are applied only to the luminance. The perceptually uniform CIELab color space is used.

- `features_from_filter_bank()` returns 17-dimensional feature vectors for the input image.
    1. Convert image to CIELab color space.
    2. Apply the three Gaussian kernels (with σ = 1, 2, 4) to L,a,b channel, producing 9 filter responses.
    3. Apply four LoGs (with σ = 1, 2, 4, 8) to the L channel, producing 4 filter responses.
    4. Apply four derivatives of Gaussians to the L channel, producing 4 final filter responses.
    5. In total, each pixel has associated a 17−dimensional feature vector.
    6. Whiten all features.

Textonization (30%)

Next, you will learn the textons by clustering the training data (The Microsoft Research Cambridge (MSRC) database2) using the K-mean algorithm. Each texton is represented by a cluster center. We have loaded the data to a NumPy array called `training_imgs` for you.

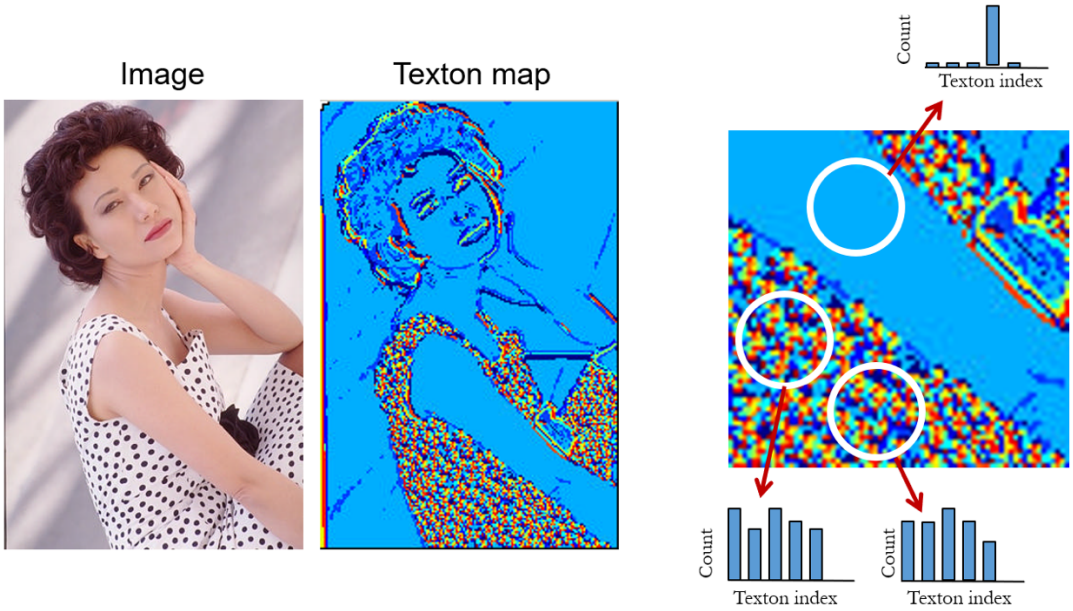
Implement the `Textonization` class in `lab4.py`.

- (15%) `Textonization.training()` function takes all training images as input, finds texton clusters, and stores the clustering centers for testing. Hint: You can improve the training speed by using the mini-batch k-mean method from `sklearn.cluster`.

- (15%) `Textonization.testing()` will predict the texture label for each pixel of the input testing image. For each pixel in the test image, an ID from a learned texton dictionary can represent it. Based on feature vectors at each pixel, it gets assigned to the nearest cluster center. The cluster ID is the texton ID. Hint You can use the KD tree `sklearn.neighbors` to improve testing speed.

Describe Texture by Texton Histogram

- We have provided the texton histogram by computing the distribution of texton indices within the window, as illustrated in the following diagram. And you are required to understand it to answer the exploration question.



- The `histogram_per_pixel()` function gathers all texton indices within a window around the pixel and computes an index distribution by counting the texton index. The size of the window is controlled by the `window_size` parameter. Note that you can use a rectangular window instead of the circle shown in the demo.

Exploratory questions (15%)

- (8%)How does the window size of the texton histogram impact the final results?
- (7%)Compared to segmentation based only on texton ID, what are the advantages and disadvantages? Please explain your reasoning. Hint: Analyze the boundary

<u>Part 3: Object Segmentation (5%)</u>

In the final step, you need to combine the optical flow feature from Part 1 with the texture features obtained from Part 2.

- Adjust `gamma` for better results.

Congratulations! You have reached the end of the Lab 4.