Project 5 Writeup

Project Overview

In this exercise, we have performed selective search algorithm for object detection. The parameters for the algorithm are taken from felzenszwalb paper. We tested the algorithm on 3 images each from 3 different fields of digital humanities namely Art history, Christian Archaeology and Classical Archaeology.

Implementation Detail

For selective search algorithm, we implemented the following functions:

- **generate-segments:** This function would divide the image into several small segments using Felzenswalb algorithm. It takes a 3D image as input, generates the initial image mask using Felzenszwalb algorithm and finally merge the image mask to the image as a 4th channel. The attached mask with the image is the output.
- **sim-colour:** It takes two regions as input and calculate colour similarity using the following formula:

$$C_i = \{c_i^1, ...c_i^n\}$$

$$s_{colour}(r_i, r_j) = \sum_{k=1}^{n} min(c_i^k, c_j^k)$$

  Here, Ci is the set of color histograms of n classes within i region.
- **sim-texture:** It takes two regions as input and calculate colour similarity using the following formulae:

$$T_i = \{t_i^1, ...t_i^n\}$$

$$s_{texture}(r_i, r_j) = \sum_{k=1}^{n} min(t_i^k, t_j^k)$$

  Here, Ti is the set of texture histograms of n classes within i region.
- **sim-size:** It takes two regions as input and calculate size similarity using the following formula:

$$s_{size}(r_i, r_j) = 1 - \frac{size(r_i) + size(r_j)}{size(im)}$$

- **sim-fill:** It takes two regions as input and calculate fill similarity using the following formula:

$$fill(r_i, r_j) = 1 - \frac{size(BB_{ij}) - size(r_i) - size(r_j)}{size(im)}$$

  Here, size(BB-ij) is the total bounding box size combining two regions.

- **calc-sim:** It takes two regions and calculate the similarity between them using the following formula:

$$s(r_i, r_j) = a_1 s_{colour}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + a_3 s_{size}(r_i, r_j) + a_4 s_{fill}(r_i, r_j)$$

- **calc-colour-hist:** It takes one region as input and calculate colour histogram of that region. For each colour channel, it gives 25 bins. So, the output histograms size will be BINS * COLOUR-CHANNELS (25*3=75). The output histogram is normalized.

- **calc-texture-gradient:** It takes the entire image and calculate texture gradient for the whole image. For this, we were using local binary pattern algorithm.

- **calc-texture-hist:** It takes previously calculated texture gradient for each region and calculate texture histogram. For each colour channel, it gives 10 bins. So, the output histograms size will be BINS * COLOUR-CHANNELS (10*3=30). The output histogram is normalized.

- **extract-regions:** It takes masked 4 channels image as input from generate-segments, extracts all the necessary values from each region and stores them in a dictionary R. At first, it counts pixel positions and calculates bounding box for each region. Then it calculates size, colour histogram and texture histogram for each region.

- **extract-neighbours:** It takes all the regions and analyzes whether the bounding boxes of these regions overlaps each other or not. If there is an overlap, then these two regions are considered as neighbours.

- **merge-regions:** It takes two regions as input and merge them. The output would be the modified dictionary R for the newly merged region. It calculates the color and texture histogram of the merged regions using the following formula:

$$C_t = \frac{size(r_i) \times C_i + size(r_j) \times C_j}{size(r_i) + size(r_j)}$$

  And the size and label of new region:

$$size(r_t) = size(r_i) + size(r_j)$$

$$label(r_t) = label(r_i) + label(r_j)$$

- **selective-search:** It takes original 3D image as input. It passes the image to generate-segments functions which divides the image into different segments using Felzenszwalb algorithm. Then the segmented image has been passed to extract-regions function which would extracts information from the segmentation and saves them in dictionary R. Then this data structure R has been passed to extract-neighbours which extracts the neighbouring information. Then, the similarity between neighbouring regions has been calculated using calc-sim function. After that, using Hierarchical search method, the regions which have highest similarity have been merged using merge-regions function. Then we delete the old similarities and calculate new similarities with the new region. Finally we generate the final region from R.

Result

We have experimented with different sigma values (width of Gaussian kernel for Felzenszwalb segmentation). Here shown some of the best results.

Figure 1: *Left:* annunciation(sigma=0.7). *Right:* baptism(sigma=0.6).


Figure 2: *Left:* ca-annun1(sigma=0.8). *Right:* ca-annun2(sigma=2.2).

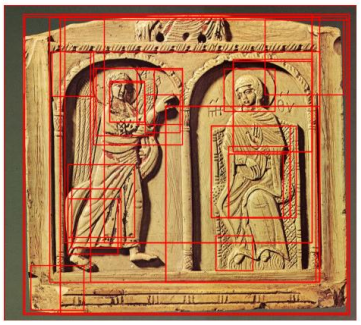


Figure 3: *Left:* ca-annun3(sigma=0.6). *Right:* ajax3(sigma=2.2).


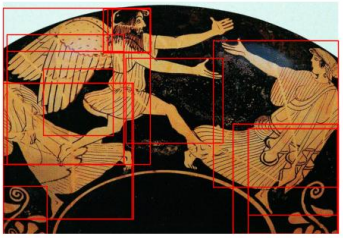

Figure 4: *Left:* leading1(sigma=1). *Right:* pursuit2(sigma=1.8).