Computer Vision 1: Assignment 2 (Due 17.12.2020)

Submission instructions:

- For each programming task, submit a `.py` source code file. Do not need to include any images or data files you used.
- For each pen & paper task, submit a `.pdf` file. Type your solution in LaTeX, Word, or another text editor of your choice and convert it to PDF – **do not submit photographs or scans of handwritten solutions!**
- In all files, include at the top names of all students in the group
- Choose exactly one person in your group that submits your solution via Moodle, it will count for the entire group.

Task 1: Template matching (programming)

Correlation filters respond most strongly to regions of images that look like the filter itself. This property is used for template matching. In template matching, we cross correlate a template with an image. The template is another image that is a prototype for what we want to localize in an image. The cross-correlation is the strongest where the image looks most like the template.

- Download a clock template image and an image containing a clock from Moodle under the names `coco264316clock.jpg` and `coco264316.jpg`, respectively.
- Read the template and the image. Convert them to grayscale.
- Use the function `skimage.feature.match_template` to compute the normalized correlation between two images. Visualize the matching result. Verify that the brightest pixel in the result corresponds to the clock.
- Flip the template horizontally. Verify that the clock can no longer be found.

Task 2: Edge detection (programming)

Noise negatively affects edge detection. To examine the effect of this in practice, we apply edge detection on a noisy image and a smoothed image.

- Load the image `woman.png` provided. Note that it is already grayscale.
- Using `skimage.util.random_noise`, add Gaussian distributed noise to the image with a variance of 0.01. This gives you a noisy image N.
- Apply `skimage.filters.gaussian` to filter N. Use $\sigma = 1.0$. This gives you a smoothed image S.

Next, apply the Sobel filter as follows.

- Use the function `skimage.filters.sobel`. This function applies the two Sobel kernels (horizontal and vertical) on the image, and then computes the root of the squared sum of the results.
- Visualize the result of applying the function on the noisy image N, and on the smoothed image S.

Suppose the output of the `sobel` function on the noisy image is F_n and the output on the smoothed image as F_s.

- Select two threshold values t_n and t_s for edge detection. Draw histograms of intensity values in F_s and F_n to come up with suitable threshold values.
- Create a binary mask by applying the logical operations $F_n > t_n$ and $F_s > t_s$ in NumPy. The parts of the image where the threshold is exceeded (there is an edge) are indicated by true values. Visualize the binary masks.
- Tune your threshold values until the detected edges show the outline of the woman's face.
- Did you notice it is almost impossible to tune the threshold for the noisy image to show just the outline of the face? Why is edge detection improved by applying smoothing?

Task 3: Image pyramids (programming)

Recall the Gaussian image pyramid as shown in the image below.

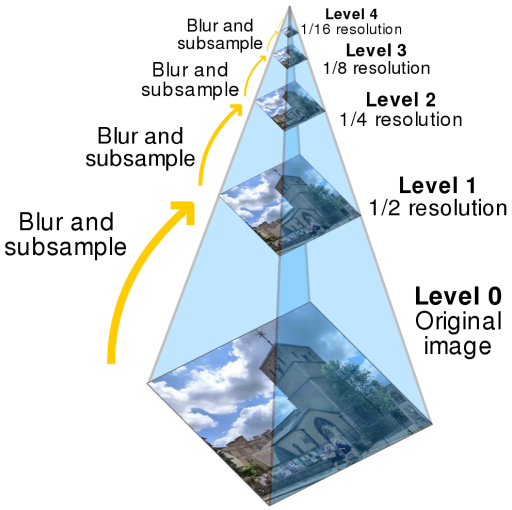

Figure 1: A Gaussian image pyramid with four layers. Image by user "Cmglee", Wikipedia.

In this task, we use Gaussian pyramids to calculate a saliency map, following the procedure illustrated in Figure 2. The approach generates two Gaussian pyramids, a center pyramid and a surround pyramid. Subtracting two layers from these pyramids corresponds to applying a Difference of Gaussian filter. This is useful to detect contrasts in images, which is the basis for saliency computation.

First read the documentation of the function `skimage.transform.pyramids.pyramid_gaussian` online. Load the image `visual_attention.png` from Moodle, and convert it to grayscale. Then, calculate the center and surround pyramids. Use the following parameters:

- Use `max_layers=4` to obtain five layers as shown in Figure 1.
- Center pyramid: $\sigma_1 = 9$,
- Surround pyramid: $\sigma_2 = 16$.

We shall denote layer j of the center pyramid by C_j and layer j of the surround pyramid by S_j. Visualize each of the layers in both pyramids.

Next, calculate *on-off contrast pyramid* and the *off-on contrast pyramid* as follows:

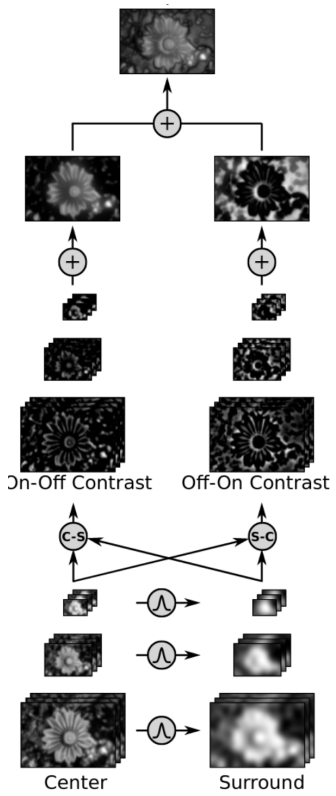

Figure 2: Saliency computation using two difference-of-Gaussian pyramids

- For the on-off contrast pyramid, the jth layer is given by $C_j - S_j$
- For the off-on contrast pyramid, the jth layer is given by $S_j - C_j$
- Important: use `np.clip` to clip values from both subtractions to the range $[0, 1]$.
- Visualize each of the layers in both pyramids.

For both the on-off pyramid and the off-on pyramid, calculate the feature maps for both pyramids as follows:

- Upsample each layer $j \geq 2$ to the same size as layer $j = 1$. Use `skimage.transform.resize` which will perform interpolation to upsample.
- Average all layers pixelwise to obtain the feature map.
- Visualize the average image.

Finally, calculate the conspicuity map or saliency map by averaging the two feature maps (on-off and off-on). Visualize the final conspicuity map.

What are the differences of this approach of computing a saliency map compared to the integral image method from assignment sheet 1? What are the advantages of the approach based on image pyramids?