Lecturer: Prof. Dr. Henning Sprekeler

Assistants: Mathias Schmerling
(mathias.schmerling@bccn-berlin.de)

# Saliency

May 03, 2018

The solutions for these exercises (comprising source code, discussion and interpretation as an IPython Notebook) should be handed in before **May 11, 2018 at 10:15 am** through the Moodle interface (in emergency cases send them to `mathias.schmerling@bccn-berlin.de`).

## Exercise 1: Saliency Maps from Local Standard Deviation

In this exercise, we will compute saliency maps for natural images which provide an estimate of where subjects will (not) move their eyes. High values in a saliency map correspond to positions in an image which attract the viewer, and low values where they are unlikely to attend. We will program a purely bottom-up, i.e. physically driven and task-independent, saliency map, similar to the one proposed by Itti, Koch & Niebur (1998) (but excluding directional image information).

1. Pick one of the provided color images (*salience.zip*) and import it as a 3D-array (99x150x3 entries) using e.g. `matplotlib.pyplot.imread` or `scipy.misc.imread`. The third dimension represents the three colors red, green and blue. Plot the color image as well as the three channels separately as a gray level image into a single figure. You may notice that the information between the color channels is highly redundant. To quantify the redundancy, compute the three pairwise correlation coefficients between each of the three color channels across all pixels (hint: use the function `flatten` to convert each image channel into a vector, stack those vectors into a matrix and then apply `numpy.corrcoef` to obtain all pairwise correlation coefficients). Is there anything striking in the correlations? Why is this not surprising given the way natural images arise (hint: think about what determines the content of a photograph of a natural scene)?

2. To decrease the interdependence between channels, convert the image from RGB- to HSV-space (hint: use the function `matplotlib.colors.rgb_to_hsv`). Once more, plot the three channels separately as gray level images into a single figure together with the color image. To quantify whether the dependence has truly decreased, compute the pairwise correlation coefficients between the new HSV-channels. Explain why there is less correlation now! If you are not familiar with the HSV-representation, have a look at Wikipedia: HSL&HSV.

3. We will now compute three independent saliency maps for the image using the HSV-representation. Therefore, decompose the image into the separate HSV-channels. Then, perform the following computations for each channel separately: For each pixel in the image, extract the surrounding $15 \times 15$ pixel image patch

(hint: feel free to ignore the edges of the image for which no full patches can be obtained). For each patch compute the standard deviation of the pixel values. Note that the Hue-channel (but not the Saturation- or Value-channels) represents a circular variable ("color circle"), so that computing the ordinary standard deviation (the function `std`) may result in artifacts. Use circular standard deviation instead (hint: the function `scipy.stats.circstd` is of help here, but be sure that parameter `high`=1). Store the standard deviation of the patches around each pixel in a new "saliency image".

4. After finishing the previous task, you should have one saliency image for each image channel at a slightly smaller size than the original picture. Plot the separate saliency images into a single figure. How is each of the three images related to the visual properties of the image?

5. Compute a joint saliency map by summing the separate saliency images you have computed. This joint map represents the "attractiveness" of each position in the image when considering the information in all three image channels. Add this plot to the previous figure.

6. You will see that the combined saliency image still contains a lot of detail. To generate a more naturally behaving saliency map, smooth this image by low-pass filtering (hint: use `scipy.ndimage.gaussian_filter`. Experiment with different sigmas and padding modes). Plot the result into a new figure.

7. In a new figure, plot the original image and overlay the contours of the smoothed saliency map (hint: remember that the saliency map is smaller than the original image. Correct for this and then use the function `contour` to plot the map upon the image).

8. From the smoothed saliency map, extract the positions the subject will most likely and least likely look at (hint: use `argmax` and `argmin` in conjunction with `unravel_index` to locate the indices of those points within the image). Mark those points with a red and blue dot, respectively (hint: function `plot` with *marker='o'* works well). How do you interpret the completed image?

9. Repeat the above procedures to plot a saliency map onto the complete set of color images provided. Discuss possible shortcomings of the proposed saliency algorithm. To this end, pick images for which the most salient position according to the algorithm does not conform with your expectation of how human attention works. Name at least two different image features the algorithm does not take into account and illustrate each with a different image.

10. We know from experiments that subjects are much more likely to first attend the center of an image, instead of the borders. Please outline in words how this information could be included in the above saliency-map analysis to better predict eye-movements.

## Exercise 2: Mexican-hat convolution (optional/bonus task)

In the above exercise, we used a shortcut to compute saliency by simply determining the standard deviation on small image patches. A more advanced technique relies on a 2D-convolution of the separate image channels with a "mexican hat" filter as presented

during the lecture. If you are interested how this works in practice, you are invited to do this optional exercise and score bonus points. Essentially, this exercise corresponds to a different version of task 1.3, everything else should be performed in the same way as in exercise 1.

1. Decompose the HSV-image into separate channels. As a preparation for the following task, generate an array representing a "mexican hat" filter using the `helper`-function `mexican_hat` with a size of 20x20 pixels and a width of 3. The filter returned from the helper function is its spatial-domain representation and need not be transformed.

2. For each image channel repeat the following procedure: Convolve the image with the prepared "mexican hat" filter (hint: use `scipy.signal.convolve2d`). Try out what happens with different boundary conditions and explain which one produces the best result. Choose the `mode` parameter appropriately to get a result that is the same size as the original image.

3. The resulting convolution image will have negative as well as positive values. Apply the absolute value function in order to obtain a positive saliency map for each channel.

4. Plot the three resulting saliency maps into a single figure, and also add the combined saliency map that you obtain by summation. In which aspects do the saliency maps obtained by convolution differ from the saliency maps that you obtained by computing standard deviation on image patches. Can you give an intuition why?

5. Perform the low-pass filtering and the analysis of attracting and repelling image positions as described in exercise 1. Plot the obtained saliency contours and computed image positions onto the original images and compare the resulting graphs with those of the previous exercise. Do you think that computing the standard deviation across patches is a valid shortcut? What is the disadvantage of using the convolution technique on HSV-represented images?