CS 6643 – HW2
Assigned March 5, 2018
Due March 26 (11.59pm)
Instructor: Guido Gerig
TA's: Yida Zhou, Zebin Xu, Andrew Dempsey, Monil D. Shah
TA office hours, see NYUClasses

Goals

The purpose of this assignment is to get familiar with implementation and application of spatial filters for image smoothing, edge detection and template matching.

Hint for implementations: Filtering and edge detection require to use float or double type for image arrays and filter. You can map those images back at the very end to integer for display purposes, and during this operation you can also optimally scale and shift the result to [0..255].

1 Image Smoothing

Implement a spatial filtering schemes for quadratic, symmetric filters. For simplification, do not filter the boundary as discussed in the course.

1.1 k*k smoothing kernel with equal weights

Implement a 2D scheme for 2D square filters of size k*k, and design a filter with equal weights (remember to normalize so that the weights sum up to 1.0). Use a k*k smoothing kernel (3*3, 5*5) to reduce noise and smooth input images. Apply a 3*3 and 5*5 smoothing to your favorite black and white images. Display images before and after filtering. You may also zoom a subregion to better show the filtering effect (as we have done in the slides).

2 Edge Detection

Objects in images can be segmented by detection of object contours (edges) followed by description of these contours and classification. Please remember that images need to be smoothed before applying derivatives. You can do this by applying the noise filtering as implemented in the previous section.

2.1 Local Edge Filtering

As discussed in class, edge detection can be implemented as a horizontal and vertical differentiation filtering. Common practice are centered 3x1 and 1x3 maskes for x and y derivatives. Application of these masks for vertical and horizontal filering results in estimates of image derivatives $I_x = \frac{\partial}{x} I(x,y)$ and $I_y = \frac{\partial}{y} I(x,y)$. These derivatives can be displayed as images (note that you get positive and negative values and that for display, it **is necessary to map those into a positive range - but this should be done only for display purposes**.

The two partial derivatives form a gradient $(I_x, I_y)^T$, which is a vector. The vector norm $\sqrt{I_x^2 + I_y^2}$ is calculated to result in an edge map. The gradient orientation is calculated as $\alpha = \tan^{-1}(I_y/I_x)$ and this local edge orientation forms another important information for subsequent processing of edge maps.

Implement the calculation of partial derivatives and calculation of the edge map. Apply edge detection to your image of choice, but remember to first run a smoothing filter. You can display the edge orientation by assigning an orientation angle to each pixel and displaying the image in the range of 0 to 360 degrees (**if**

your display is limited to 0 to 255 just scale the value range appropriately). Display the results as 6 images (original, smoothed, dx, dy, edge map, orientation map) and discuss.

3 Template Matching

Spatial filtering is also used for finding specific objects in images by template matching. Given a template of a sought object, the template acts as a filter, and the maximum correlation indicates the position of the object (see course slides on filtering).

Use the 2D spatial filtering technique implemented before to find image objects. Now, the filter weights are not a user-defined mask but a mask that is represented as a small image template which you select.

Please note that since the template is not normalized and may accumulate high values in bright regions and smaller ones in dark regions, we will need to apply a correlation between a zero-mean template and a normalized image region.

Steps:

- Select an image presenting a repeated pattern of same size/orientation objects where the task is to find the position of objects (e.g., similar cars in a parking lot, specific letter in an image of a text document). You can use one of the images shown here or an image which you select.
- Select a template, i.e. a very small subimage containing your object.
- Calculate the mean intensity of this template region and deduct the mean from each template pixel. The result is a zero-mean template $\tilde{w}(s,t) = w(s,t) - \bar{w} \quad \forall(s,t)$ within the template with positive and negative pixel values.
- Use the template as a mask m*n, and filter the image with this mask to obtain a correlation image. Important: For each move of the template to filter the image, we need to calculate the mean intensity within the respective region of the image and then subtract this mean $\bar{f}$ from each pixel value while calculating the correlation (same as done with the template). Representing the normalized template values as a 1-D string of length k and the normalized image intensities at the specific window location as a 1-D string with same length, this results in a correlation $g = \sum_{s=1}^{k}(w(s) - \bar{w}).(f(s) - \bar{f}) = \sum_{s=1}^{k} \tilde{w}(s).\tilde{f}(s)$.
- While the above should work well and is good enough to get the peaks for best matches, you can do the additional step towards a normalized cross-correlation by calculating $c = \sum_{s=1}^{k} \tilde{w}(s).\tilde{f}(s)\frac{1}{|\tilde{w}|.|\tilde{f}|}$, where $|\tilde{w}|$ and $|\tilde{f}|$ are the norms (magnitudes) of the two zero-mean vectors. This guarantees correlation results between -1 and 1.
- Apply thresholding to the correlation image to get the set of correlation peaks, which are indicating the locations of your selected template object. Please note that correlations can contain positive and negative values so that the result needs to be adjusted for display purposes (shifted, scaled).
- Would you not be completely happy with the thresholded correlation image, you can apply a peak detection to the correlation image to enhance peaks of best correlating regions. Remember that the Laplacian filter with 3x3 mask [[0,-1,0], [1,4,-1], [0,-1,0]] is an optimal peak detector.
- Show the original image, the correlation image and the thresholded and eventualy peak-detected correlation image side by side and discuss. For nicer results than just showing peaks, one can select the largest peaks and add the template subimage centered at these location as resulting image. This image would then clearly show where you found the best matching image regions that correlate with the template.

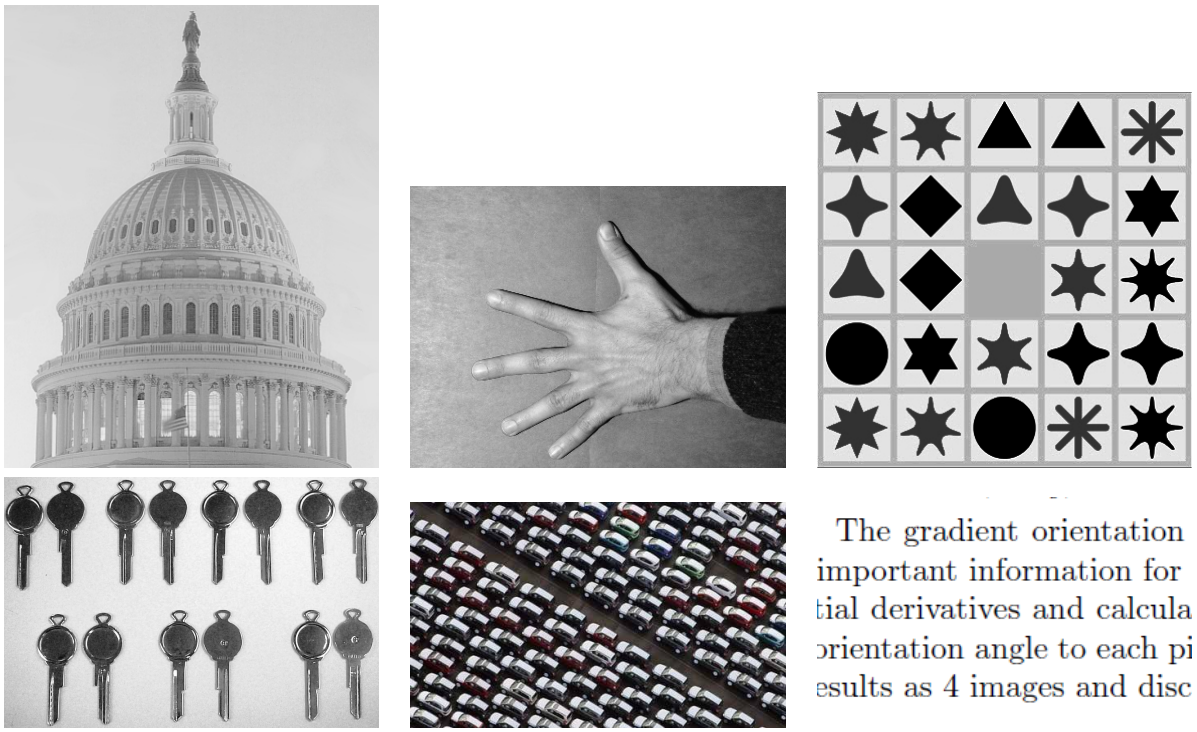

Figure 1: Test images that can be used for illustrating your solutions. Please feel free to use your own images and be creative. Filtering and edges: captitol.jpg , hand-bw.jpg. Template matching: shapes-bw.jpg, keys-bw.jpg, cars-bw.jpg, text.png.

4 Instructions, Requirements, and Restrictions

1. Please use your name **"NAME"** in all reports and submitted materials, to uniquely identify your projects.
2. We **do not allow to use toolbox functions** or other existing image processing libraries in order to give all students the same chances for code development.
3. You should have in your report a short description of each algorithm you used and documentation on how your code is organized. Failure to do this will result in a loss of points.
4. Your project report will be in the form of a pdf file.
5. You will submit a pdf for the report and a single tar file created from from your project.
6. Please remember or look-up the honor code and requirement to provide your own solution as discussed in the syllabus.
7. **Please look up the late policy as defined in the syllabus**