CS/BIOEN 6640 – Project 2
Assigned Sept. 24, 2014
Due Oct. 8 (11.59pm)
Instructor: Guido Gerig
TAs: Avantika Vardhan, Padmashree Shrinivasa Shetty Teeka,
TA office hours Mo 3-5pm,Wed 11am - 1pm, office WEB 2626

Goals

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

1 Image Smoothing

Implement a spatial filtering schemes for quadratic, symmetric filters. For simplification, do not filter the boundary as discussed in the course. Two schemes have to be implemented, one for 2D k*k-size kernels and one for separable filtering with 1D kernels of size k in horizontal and vertical directions.

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 to 1). Use a k*k smoothing kernel (3*3, 5*5) to reduce noise and smooth input image. Apply a 3*3 and 5*5 smoothing to your favorite black and white images.

1.2 Separable Filtering Scheme

Implement a filtering scheme for separable filters (see course slides) where 1D filtering in horizontal followed by vertical is applied. Use this separable scheme to achieve the same effect of smoothing filtering (3*3, 5*5) by applying 1D smoothing filters with 3 and 5 pixel width. Again, normalization of the filter masks is required.

1.3 Separable Gaussian Filter

Use the separable filtering scheme for 2D Gaussian filters. We will use the standard equation of a 1D Gaussian filter centered at 0 $G(x,\sigma) = \frac{1}{\sqrt{(2\pi*\sigma^2)}} \exp\left(-\frac{1}{2}(\frac{x}{\sigma})^2\right)$ to construct 1D filters with weights that fall off with distance from the center.

By choosing a specific filter of width σ pixels, you can estimate mask weights for a symmetric filter in the range of $\pm 3\sigma$. For example, a filter of width $\sigma = 2$ extends 6 pixels to both sides from the center and thus forms a symmetric mask of size (2*6+1) pixels. Please note that the 1D filter weights need to be normalized to sum up to 1. This 1-D filter can then be applied in horizontal followed by vertical direction, which results in an efficient 2D filtering scheme.

Apply the Gaussian filtering to the same image(s) as before and compare results.

2 Edge Detection

Objects in images can be segmented by detection of object contours (edges) followed by description of these contours and classification.

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, or 3x3 masks (Prewitt, Sobel, book figure 3.41 and slides). 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**. **Please also see comment in the introduction w.r.t. scaling of derivative filter masks.**

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 (**see also book section 3.6.4 and slides**). 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. You can also 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 **four** results as 4 images (dx, dy, edge map, orientation map) and discuss.

2.2 Edge filtering at specific scales via smoothing

Whereas the previous section applies edge detection directly on the noisy images, a common approach is edge detection at a specific scale, i.e. estimation of edges after Gaussian filtering.

You can do this by first filtering images with a Gaussian filter (implementation above) and then applying local edge detection with x and y derivative filters. Test this procedure by filtering with a specific filter width (e.g. $\sigma = 2$) followed by gradient calculation. Please note that due to the 2D Gaussian filtering, you can reduce the gradient calculation to simple 1D filters with length 3 ($\frac{1}{2}[-1, 0, 1]$) applied in horizontal and also vertical direction rather than 3x3 masks (**since the filtering already provides regularization in the 2D domain**).

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/segment image objects. Now, the filter weights are not a user-defined mask but a mask that is represented as a small image template.

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 zero-mean template and image region.

Steps:

- Select an image presented 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).
- 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, whih are indicating the locations of the template structure. Please note that correlations can contain positive and negative values so that a display needs to be adjusted.

- Show the original image, the correlation image and the thresholded correlation image side by side and discuss.

4 Bonus

- For more advanced experts: Try to develop a procedure that finds these peaks and creates a list of their posion. Then, put the template at each of those positions to generate a resulting image that shows the segmentation result.

- Hint: the procedure above can also be applied to binarized/thresholded images, i.e. using thresholded images as input and a binary template. This way, the correlation peaks signify the number of pixels in the image and the template that overlap. The advantage would be that the correlation peak is independent of the absolute gray values. Would you have time, you can try this version in addition to gray-scale template matching.

5 Instructions, Requirements, and Restrictions

1. Please use your name **"NAME"** in all reports and submitted materials, to uniquely identify your projects.

2. For Matlab each individual function (including functions you define) should be a ".m" file, and your functions should call one another as necessary.

3. We **do not allow to use Matlab toolbox functions** (e.g. Imaging Toolbox) or other existing image processing libraries in order to give all students the same conditions for code development [1].

4. 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. Please remmember to **add your name** to the report title.

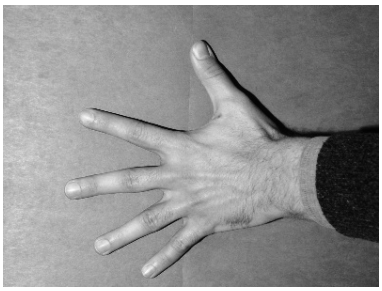
5. Your project report will be in the form of a pdf file.

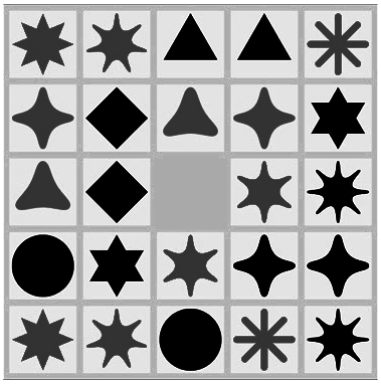
6. You will submit a pdf for the report and a single tar file created from from your project.

7. Please remember or look-up the honor code and requirement to provide your own solution as discussed in the syllabus.

8. **Please look up the late policy as defined in the syllabus**

[1] The core MATLAB package comes with several rudimentary functions that can be used to load, save, and perform custom functions on images. Taken from wikibooks





The gradient orientation
important information for
tial derivatives and calcula
orientation angle to each pi
esults as 4 images and disc

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