Higher Level Computer Vision

Summer Semester 2018

Prof. Margret Keuper
keuper@uni-mannheim.de

Material provided by Mario Fritz and Mario Fritz @ MPI for Informatics, Saarbrcken

Exercise 1: Image Filtering and Object Identification

In this exercise you will first familiarise yourself with basic image filtering routines. In the second part, you will develop a simple image querying system which accepts a query image as input and then find a set of similar images in the database. In order to compare images you will implement several distance functions introduced in the lecture and evaluate their performance in combination with different image representations.

Download the file `exercise1.zip` from the course Ilias page and unpack it in a separate directory. The file consists of two directories - filtering and identification. Each directory contains a placeholder script (`filter.m` and `identification.m`, resp.) for functions you will have to implement in this exercise; your task is to fill the missing code corresponding to each subproblem and produce brief reports on the results whenever necessary.

The filtering part contains two images: `graf.png` and `gantrycrane.png`, which we will use for testing purposes. The identification part contains *query* and *model* images, which will be used to evaluate your implementation. The model and query images correspond to the same set of objects photographed from different viewpoints. The files `model.txt` and `query.txt` contain lists of image files arranged so that i-th model image depicts the same object as i-th query image. The placeholder scripts will also be used to test your solution. Ideally, after you have implemented all the missing code you should be able to execute it without errors.

Question 1: Image Filtering

a) Implement a method which computes the values of a 1-D Gaussian for a given variance σ^2. The method should also give a vector of values on which the Gaussian filter is defined: integer values on the interval $[-3\sigma, 3\sigma]$.

$$G = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{x^2}{2\sigma^2}). \tag{1}$$

Open the file `gauss.m` with your preferred editor and begin the script:

```
function [G,x]=gauss(sigma)
...
...
end
```

b) Implement a 2D Gaussian filter in `gaussianfilter.m`. The function should take an image as an input and return the result of convolution of this image with 2D Gaussian kernel. See Fig. 1 for illustration of Gaussian filtering. You can take advantage of the Matlab's `conv2` function if you don't want to implement convolution yourself.

Open the file called `gaussianfilter.m` with your preferred editor and begin the script:

```
function outimg=gaussianfilter(img,sigma)
...
...
end
```

Hint: use the fact that 2D Gaussian filter is separable to speed up computations.

c) Implement a function `gaussdx.m` for creating a Gaussian derivative filter in 1D according to the following equation

$$\frac{d}{dx}G = \frac{d}{dx}\frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{x^2}{2\sigma^2}) \tag{2}$$

$$= -\frac{1}{\sqrt{2\pi}\sigma^3} x \exp(-\frac{x^2}{2\sigma^2}) \tag{3}$$

The effect of applying a filter can be studied by observing its so-called *impulse response*. For this, create a test image in which only the central pixel has a non-zero value:

```
imgImp = zeros(25,25);
imgImp(13,13) = 1.0;
```

Now, create the following 1D filter kernels `G` and `D`.

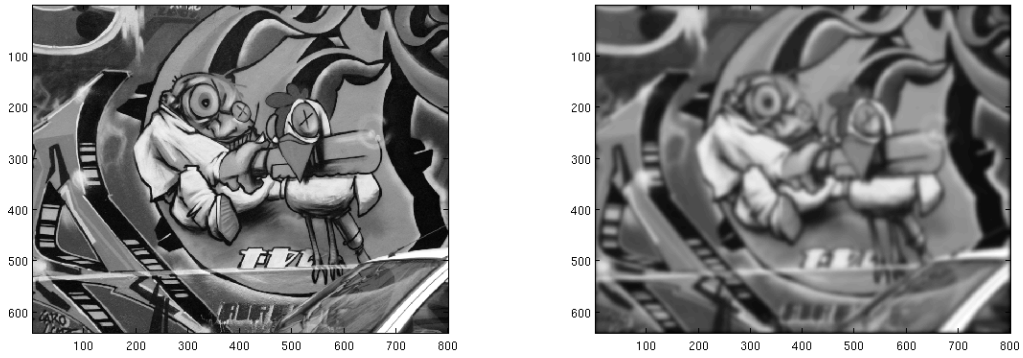

Figure 1: (a) Original image (b) Image after applying a Gaussian filter with $\sigma = 4.0$.

```
sigma = 6.0;
G = gauss(sigma);
D = gaussdx(sigma);
```

What happens when you apply the following filter combinations? (For best visualization, display the result image with the `imagesc` command).

1. first G, then G'.
2. first G, then D'.
3. first D, then G'.
4. first G', then D.
5. first D', then G.

d) Use the functions `gauss` and `gaussdx` in order to implement a function `gaussderiv` that returns the 2D Gaussian derivatives of an input image in x and y direction. Try the function on the given example images and comment on the output.

Question 2: Image Representations, Histogram Distances

a) Implement a function `normalized_hist.m`, which takes gray-value image as input and returns normalized histogram of pixel intensities. Compare you implementation with built-in Matlab function `hist.m`. Your histograms and histograms computed with Matlab should be approximately the same, except for the overall scale, which will be different since `hist.m` does not normalize its output.

b) Implement other histogram types we partly discussed during the lecture. Your implementation should extend the code provided in the files `rgb_hist.m`, `rg_hist.m` and `dxdy_hist.m`, where `rgb_hist.m` computes a 3D rgb histogram, `rg_hist.m` a 2D histogram of two color channels red and greed after normilization and `dxdy_hist.m` computed the 2D histogram of partial derivatives in x and y directions. Make sure that you are using the correct range of pixel values. For "RGB" the pixel intensities are in $[0, 255]$, for "rg" the values are normalized to be in $[0, 1]$. For derivatives histograms the values depend of the σ^2 of the Gaussian filter, with $\sigma = 6.0$ you can assume that the values are in the range $[-34, 34]$.

c) Implement the histogram distance functions discussed during the lecture, by filling the missing code in files `dist_l2.m`, `dist_intersect.m`, and `dist_chi2.m`.

Question 3: Object Identification

a) Having implemented different distance functions and image histograms, we can now test how suitable they are for retrieving images in query-by-example scenario. Implement a function `find_best_match.m`, which takes a list of model images and a list of query images and for each query image returns the index of closest model image. The function should take string parameters, which identify distance function, histogram function and number of histogram bins. See comments in the beginning of `find_best_match.m` for more details. Additionally to the indices of the best matching images your implementation should also return a matrix which contains distances between all pairs of model and query images.

b) Implement a function `show_neighbors.m` which takes a list of model images and a list of query images and for each query image visualizes several model images which are closest to the query image according to the specified distance metric. Use the function `find_best_match.m` in your implementation. See Fig 2 for an example output.

(a) (b)

Figure 2: Query image (a) and model images with color histograms similar to the query image (b).

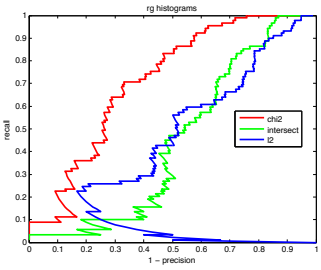

Figure 3: Recall/precision curve evaluated on the provided set of model and query images.

c) Use the function `find_best_match.m` to compute recognition rate for different combinations of distance and histogram functions. The recognition rate is given by a ratio between number of correct matches and total number of query images. Experiment with different functions and numbers of histogram bins, try to find combination that works best. Submit the summary of your experiments as part of your solution.

Question 4: Performance Evaluation (10 points)

a) Sometimes instead of returning the best match for a query image we would like to return all the model images with distance to the query image below a certain threshold. It is, for example, the case when there are multiple images of the query object among the model images. In order to assess the system performance in such scenario we will use two quality measures: *precision* and *recall*. Denoting threshold on distance between images by τ and using the following notation:

TP (True Positive) = number of correct matches among images with distance *smaller* then τ,
FP (False Positive) = number of incorrect matches among images with distance *smaller* then τ,
TN (True Negative) = number of incorrect matches among images with distance *larger* then τ,
FN (False Negative) = number of correct matches among images with distance *larger* then τ,

precision is given by

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{4}$$

and recall is given by

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{5}$$

For an ideal system there should exist a value of τ such that both precision and recall are equal to 1, which corresponds to obtaining all the correct images without any false matches. However in reality both quantities will be somewhere in the range between 0 and 1 and the goal is to make both of them as high as possible.

Implement a function `plot_rpc.m` in which you have to compute precision/recall pairs for a range of threshold values and then output the precision/recall curve (RPC), which gives a good summary of system performance at different levels of confidence. See Fig 3 for an example of RPC curve.

b) Plot RPC curves for different histogram types, distances and number of bins. Submit a summary of your observations as part of your solution.