



Prof. B. Schiele  
<schiele@mis.tu-darmstadt.de>

Micha Andriluka  
<andriluka@mis.informatik.tu-darmstadt.de>

## Exercise 3: Global Recognition Methods

(due May 7th 2008, 23:59)

### Question 1: Object Recognition – Global Approaches

Download the file `exercise3.tar.gz` from the class web page and uncompress it in your working directory. This file contains several example images, which you will manipulate in the following.

- a) Read an image and convert it to grayvalues. Reshape the 2D image array of size  $(N,M)$  to 1D vector of size  $(N*M,1)$ ; Display a histogram of gray values using histogram function. Compare histograms for different number of bins. Explain the observations.

```
img=double(rgb2gray(imread('model/obj100__0.png')));
img1D=reshape(img,size(img,1)*size(img,2),1);
bins=10;
hist(img1D,bins);
bins=20;
hist(img1D,bins);
bins=40;
hist(img1D,bins);
```

- b) Implement a function called `myhist` in file `'myhist.m'` which takes a filename and a number of bins as an input and returns a 1D histogram vector of grayvalues. Quantize image such that it has only bins number of values i.e multiply image by factor and round the values.

```
function h=myhist(filename,bins)
%read the image and convert it to grayvalues.
img1=double(rgb2gray(imread(filename)));
%quantize the image to "bins" number of values
img1=floor(img1*bins/255)+1;
%define a histogram vector with "bins" number of entries
h=zeros(bins,1);
%execute the loop for each bin number
for i=1:bins
    %return a binary image with 1 for pixel values equal to bin number and 0 for other va
    imgtmp=(img1==i);
    %compute the number of values which are equal to the bin number
    val=sum(sum(imgtmp));
    %assign the value to the corresponding bin.
    h(i)=val;
end
%normalize the histogram such that its integral (sum) is equal 1
h=h/sum(sum(h));
%display the histogram
bar(h);
```

Compare the result with the histogram obtained with function `hist`. Explain the observations.

- c) Implement a new function 'myhist2.m' which takes the same input as the previous one but returns a histogram of RGB values.

```
function h=myhist2(filename,bins)
    %read the image and convert it to double.
    img1=double(imread(filename));
    %quantize the image to "bins" number of values
    img1=floor(img1*(bins)/255)+1;
    %define a 3D histogram with "bins^3" number of entries
    h=zeros(bins,bins,bins);
    %execute the loop for each pixel in the image,
    for i=1:size(img1,1)
        for j=1:size(img1,2)
            %increment a histogram bin which corresponds to the value of pixel i,j; h(R,G,B)
            R=img1(i,j,1);
            G=img1(i,j,2);
            B=img1(i,j,3);
            h(R,G,B)=h(R,G,B)+1;
        end
    end
    %normalize the histogram such that its integral (sum) is equal 1
    h=h/sum(sum(sum(h)));
    %reshape the histogram to obtain a 1D vector
    h=reshape(h,bins*bins*bins,1);
```

Optionally. Remove the bins corresponding to the black color (background) before normalization.

- d) (5 points) Implement a function which takes as an input two histograms  $h_1$  and  $h_2$  and returns the distance between them:

$$dist(h_1, h_2) = \sqrt{\sum_{i=1}^D (h_1(i) - h_2(i))^2} \quad (1)$$

compute the distance between images 'model/obj1\_0.png' and 'model/obj91\_0.png', 'model/obj1\_0.png' and 'model/obj94\_0.png', which distance is smaller?

- e) (5 points) Implement other distance functions you know from the lecture and repeat the previous exercise.
- f) (5 points) Implement a new function 'myhist3.m' which takes the same input as the previous one but returns a histogram of  $r$ ,  $g$  values, where  $r = \frac{R}{R+G+B}$  and  $g = \frac{G}{R+G+B}$
- g) (5 points) Implement a new function 'myhist4.m' which takes the same input as the previous one but returns a histogram of  $dx$ ,  $dy$  values, which are Gaussian derivatives of the image in  $x$  and  $y$  directions. Note that derivatives can have negative values. To correct that add a constant value to the derivatives.

- h) Implement a function 'allhist.m' which takes as an input files 'model.txt' with a list of image names and returns a 2D array of RGB histograms of the images. The histograms are the column of the array. Begin the function with

```
function histograms=allhist(files,hist_type,bins)
    files=textread(files,%s);
    histograms=0;
    %nb=1;%matlab
    nb=2;%octave
    %define a size of the array depending on the histogram dimensions
    switch hist_type
        case 'RGB'
            histograms=zeros(size(files,nb),bins^3);
        case 'rg'
            histograms=zeros(size(files,nb),bins^2);
        case 'dxdy'
            histograms=zeros(size(files,nb),bins^2);
        otherwise
            end
    %compute the histogram for each image
    for i=1:size(files,nb)
        %filename=char(files(i));%matlab
        filename=nth(files,i);%octave
        switch hist_type
            case 'RGB'
                h=myhist2(filename,bins);
            case 'rg'
                ...
            case 'dxdy'
                ...
            otherwise
                end
        histograms(i,:)=h;
    end
end
```

- i) (5 points) Implement a function which takes as an input a filename containing model names and an input image and returns the filename of model histogram for which the distance to the input image is minimum. The function should compute (a) an array of histograms for model image and a histogram for input image, (b) a vector of distances between the single histogram and each of the histograms from the array. (c) find the minimum distance in the vector of distances, (d) use the index of the minimum distance to display the corresponding filename.
- j) (5 points) Implement a function which takes as an input two filenames 'model.txt' and 'query.txt' containing the list of files in the model directory and in the query directory. The function should compute two arrays of histograms; one for model and one for test images. Each array will be of dimension  $N \times M$  where  $N$  is the number of images in each directory and  $M$  is the size of histograms. Implement a loop which computes a distance between each pair of histograms (model, query) and store the distances in an array, where the column number corresponds to the number of a model images and the row number correspond to the number of a query images. Define two counting variables representing number of positive (correct) and false (incorrect) detections. Find the minimum distance for each column of the array. Search the minima in the array and increment the counter for the positive detections if the number of row in which you find the minimum is equal to the number of column, otherwise increment the counter for the false detections. Divide the variables by the number of columns to normalize the score.

- k) (5 points) Instead of finding minima in distance array, find distances which are smaller than a chosen threshold i.e., 0.3. Count the number of positive and false detections. Divide the number of positive detections by the number of columns. Divide the number of false detections by the total number of distances below threshold. Compute the results for different thresholds [0:0.01:0.4]. Display a plot: positive detections on Y axis and threshold on X axis. Display a plot: positive detections on Y axis and negative detections on X axis. Save the plots in PNG format and submit them along with your solution.
- l) (5 points) Use different methods for computing histograms and compare the resulting positive detections and false detections. Repeat the exercise for different distance functions. Try to find a combination of histogram method and distance function which gives the best results. Write a short summary of your observations and your explanation for them and submit it along with your solution.

---

*Please turn in your solution by sending an email to Micha Andriluka ([andriluka@mis.informatik.tu-darmstadt.de](mailto:andriluka@mis.informatik.tu-darmstadt.de)) including all relevant m-files before Wednesday, May 7th, 23:59*