Measuring features

Arnold Smeulders, Ran Tao and Zhenyang Li*

March 24, 2014

1 Introduction

In this practical assignment, we differentiate between:

- **Do** and **Analyze:** Just do it, no report required
- **Assignment:** Experiment and report about it. We expect a report in PDF format and seek for condensed answers.

Download the images, and Matlab files from `http://staff.science.uva.nl/~rantao/a25/measure_features_handout.zip`.

1. Read the exercises carefully (it really helps).
2. Form teams of two students. It is preferred when for all pairs one student has some experience with Matlab.

2 Useful Matlab Tips and Commands

The following is a list of useful Matlab commands. We encourage you to look into the help of these commands.

Read and convert images. Note that Matlab likes images in double format the best, while loading in images usually results in an integer array. You may need to convert. Some useful functions:

```
imread
im2double
rgb2gray
imshow
```

Code to create a circle. Look at the output of the `meshgrid` command to understand the function. Note that it's also a good idea to create code for a rectangle.

```
function im = circleIm(sizeIm, r)
% im = circleIm(sizeIm, r)
%
% Creates an image of a circle with radius r in an image canvas of size
% sizeIm x sizeIm

center = round(sizeIm /2);
[x y] = meshgrid((1:sizeIm) - center, (1:sizeIm) - center);
cDist = x.^2 + y.^2;
im = cDist < r^2;
```

* Text originally provided by Efstratios Gavves and Jasper Uijlings

Mathematical Morphology operators. Notice that these operators want a structural element.

```
strel % create structural element
imerode
imdilate
imopen
imclose
```

Getting the boundary contours of an object.

```
bwboundaries
```

Miscellaneous:

```
a > b   % Thresholding (works for matrix a and value b)
sum     % Sum a matrix over a dimension
size    % Get dimensions of a matrix
find    % Find non-zero elements
a(i,j)  % Access matrix element(s)
a{i,j}  % Access cell element(s)
```

3 Part 1: Measuring Physical Quantities

The goal of this assignment is to create a method for measuring the area and circumference of a photographed object as accurately and precisely as possible. You can use the methods which were discussed in the lecture and try to improve them.

In the IMAGES directory, you can find some coin pictures named "coin_XX.jpg", recorded under different conditions. Please use these pictures for this assignment.

Assignment: For a round object, measure the diameter and the area. Using these, report an estimation of π. To do this you should threshold your image such that it becomes a **binary** image. Artifacts may be removed by using appropriate matlab functions and techniques described in previous section. What influences the estimation of π?

Discuss for *all* tried methods (not only your successful methods): Does the method work or not? Under which conditions does the method work, and under which conditions does it fail? Can you explain the reasons for failure or success?

Hand in a document of 1 or maximum 2 pages in which you describe how you address the problem and your findings.

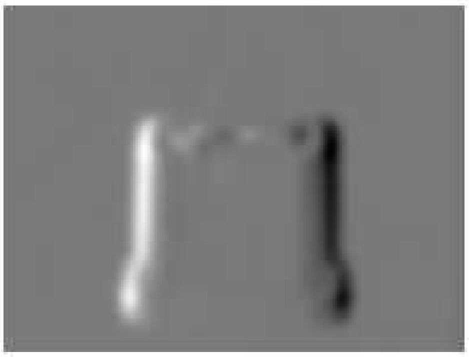
4 Part 2: Image Features and Invariance

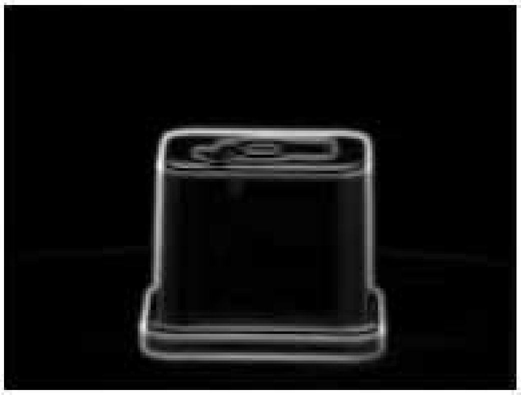
4.1 Gaussian-based Image Features

Note: The images can be found at the download location specified at the first page of this document.

The following commands display the first order x-derivative, $im_x = \frac{\partial}{\partial x}$ with $\sigma = 3.0$.

```
>> im = imread('images/8_l8c1.png');
>> imGray = rgb2gray(im2double(im));
>> imx = gDer(imGray,3,1,0);
>> imshow(imx,[]); % The empty list [] is for automatic scaling of the image
```





4.2 Simple Invariance

Do: Program the image gradient magnitude of image I, called I_M, calculated by $I_M = \sqrt{I_x^2 + I_y^2}$

```
>> imshow(imageGradient(imGray, 1),[]) % implement imageGradient(image, sigma)
```

Analyze what kind of invariance the gradient feature I_M does and does not exhibit.

4.3 Color Invariance

With invariant features, object properties can be measured, independent of illumination direction, viewing direction, etc. For example, an invariant property W, is impervious to global illumination changes, by dividing all color channels by the intensity E. In the example code we have supplied an implementation of the color x-edge response of invariant W, W_x, computed from $W_x = \sqrt{\left((\frac{E_x}{E})^2 + (\frac{E_{\lambda x}}{E})^2 + (\frac{E_{\lambda\lambda x}}{E})^2\right)}$. This implementation can be found in the file `invariantWx.m`

Notice the λ-notation. If you look at the slides of the colorspace E, you can see that E_λ really is the derivative with respect to the wavelength. More information can be found in [1].

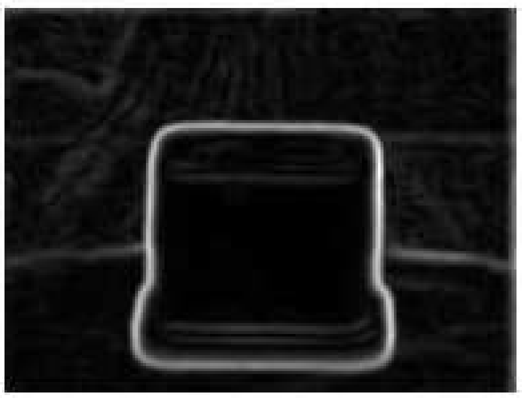
Do: Write a function to calculate the gradient magnitude for colorspace W.

Now, we consider the gradient of the shadow and shading invariant colorspace C from [2]. This colorspace has two color channels only:

$$C_{\lambda x} = \frac{E_{\lambda x}E - E_\lambda E_x}{E^2} \qquad C_{\lambda y} = \frac{E_{\lambda y}E - E_\lambda E_y}{E^2} \qquad \rightarrow C_{\lambda m} = \sqrt{C_{\lambda x}^2 + C_{\lambda y}^2}$$
$$C_{\lambda\lambda x} = \frac{E_{\lambda\lambda x}E - E_{\lambda\lambda} E_x}{E^2} \qquad C_{\lambda\lambda y} = \frac{E_{\lambda\lambda y}E - E_{\lambda\lambda} E_y}{E^2} \qquad \rightarrow C_{\lambda\lambda m} = \sqrt{C_{\lambda\lambda x}^2 + C_{\lambda\lambda y}^2}$$

Do: write a function to calculate the gradient magniture of invariant C.

```
>> imshow(invariantCm(im, 2),[]) % implement
```



In the resulting image of invariant C, some of the edges within the object are still detected.

Analyze: Why is the invariant sensitive to these edges? What happens to the background in the image? Try to implement a way to suppress these background artifact edges.

5 Part 3: Feature Computation and Object Recognition

In the IMAGES directory, different versions of the same object are given. These different versions are denoted with `l8c1`, `r5`, `l1c1`.

5.1 Simple Feature Computation

We give some example code to read in some images, and write the image features from Harris points to disk. The function `computeFeatures(objectNrs)`, takes a list of objects as its argument, and writes the features to the directory FEATURES (make sure this directory exists).

```
>> computeFeatures(1:5)
file: 1_l1c1.png found: 15 points
file: 1_l8c1.png found: 15 points
file: 1_r5.png found: 15 points
file: 2_l1c1.png found: 13 points
file: 2_l8c1.png found: 15 points
file: 2_r5.png found: 15 points
file: 3_l1c1.png found: 15 points
file: 3_l8c1.png found: 15 points
file: 3_r5.png found: 15 points
file: 4_l1c1.png found: 15 points
file: 4_l8c1.png found: 14 points
file: 4_r5.png found: 15 points
file: 5_l1c1.png found: 8 points
file: 5_l8c1.png found: 7 points
file: 5_r5.png found: 8 points
```

Do: Get acquainted with the code of this function.

5.2 Simple Feature Matching for Object Recognition

The given example code takes the computed feature values from one image per object, and takes for each image (recorded under the second condition) its features and compares them to the original object features. Two images are compared by taking the absolute differences between all points of the two images, and then finds for each point the closest point from the other image. This is done for two images; these distances are then added and divided by the total number of points in the two images.

The function `matchFeatures(objectNrs, testSuffix)`, takes a list of objects as argument 1, and the suffix of the test condition as its second argument. The function reads the features as computed in the previous section, from the directory FEATURES. A ranked list is returned, indication the position of the correct object. Furthermore, some scoring statistics are computed:

```
>> matchFeatures(1:5, 'r5')
median of objects rankings: 1
total nr. of objects recognized: 3 out of 5

ans =

     1
     1
     1
     3
     2
```

Do: Run the object recognition experiment for the varying illumination condition. Explain the hardness of the problem for the two conditions, and give examples of problematic images.

References

[1] J. M. Geusebroek, R. van den Boomgaard, A. W. M. Smeulders, and A. Dev. The spatial structure of color images. In *ECCV*, 2000.

[2] J. M. Geusebroek, R. van den Boomgaard, A. W. M. Smeulders, and H. Geerts. Color invariance. In *TPAMI*, 2001.