Assignment 1: Feature Extraction
Vision and Image Processing

Kim Steenstrup Pedersen, François Lauze, and Jan Kremer

November 14, 2013

This is the first mandatory assignment on the course Vision and Image Processing. The goal is for you to get familiar with programming for computer vision with a focus on feature extraction. Feature extraction forms the basis of many of the techniques we are going to study on this course. This assignment 1 must be solved **individually**.

You have to pass this and the following mandatory assignments in order to pass this course. There are in total 4 mandatory pass/fail assignments.

The deadline for this assignment is Monday 2/12, 2013. You must submit your solution electronically via the Absalon home page. Go to the assignments list and choose this assignment and upload your solution prior to the deadline. Remember to include your name and KU user name (login for KUnet) in the solution. If you do not pass the assignment, having made a SERIOUS attempt, you will get a second chance of submitting a correct solution.

A solution consists of:

- Your solution source code in original plain text format (Matlab / Python scripts / C / C++ code) with comments about the major steps involved in each question (see below).
- Your code should be structured such that there is one main file that we can run to reproduce all the results presented in your report. This main file can, if you like, call other files with functions / classes.
- Your code should also include a README text file describing how to compile (if relevant) and run your program, as well as a list of all relevant libraries needed for compiling or using your code. If we cannot make your code run we will consider your submission incomplete and you may be asked to resubmit.
- The code, auxiliary files and README file should be put into a compressed archive file in either the ZIP or TAR format (RAR is not allowed - we simply cannot read your archive).
- A PDF file with notes detailing your answers to the questions, which may include images, graphs and tables if needed (**Max 5 pages** text including figures and tables). Do NOT include your source code in this PDF file.

A note on relevant software

We recommend that you select either Matlab / Python / C / C++ as the programming language you use for your solutions for the assignments on this course. We also recommend that you select the language you are most confident in. The focus should be on learning the methods and not learning a new programming language.

If you wish to use Matlab, the University of Copenhagen has a license agreement with MathWorks that allows students to download and install a copy of Matlab on personal computers. On the KUnet web site you find a menu item called Software Library (Softwarebiblioteket) `https://intranet.ku.dk/Sider/Software-bibliotek.aspx`. Under this menu item you can find a link to The Mathworks - Matlab & Simulink + toolboxes. Click this link and follow the instructions for how to install on your own computer.

If you use C / C++ we recommend that you use the OpenCV library `http://opencv.willowgarage.com/wiki/` and/or the VLFeat library `http://www.vlfeat.org/`. Both libraries provide an extensive collection of implementations of central computer vision algorithms. OpenCV also provides an interface for Python. Follow the installation instructions on these websites to install on your own computer. OpenCV is also available via MacPorts on MacOSX.

If you wish to program your solutions in C++, we also recommend the Shark machine learning library `http://image.diku.dk/shark/sphinx_pages/build/html/index.html` which should work on Windows, Linux and MacOSX. You need to have CMake and the boost library installed before you install Shark.

1 Detecting interest points (Features)

We start by detecting interest points in images (also sometimes referred to as features). As we will see throughout this course, the ability to solve this problem well is necessary for the success of many computer vision applications.

Find a couple of images to apply your solution to and illustrate that it works. Apply either a blob detector, such as either Difference of Gaussians (DoG) or the Laplacian, or the Harris corner detector to your images. In this part you are allowed to use library functions in your solution, but we recommend that you attempt to implement the detection algorithm from scratch. Draw the detected points on top of the image at the detected locations and include these images as figures and comment on your results in your report. In your report you also have to explain how you made your solution.

Consider extending your solution to multi-scale detectors such as the multi-scale Laplacian, multi-scale Harris corner or Harris-Laplacian detectors.

2 Simple matching of features

Next let us try to perform a simple feature matching based on your detected points. As part of the assignment in Absalon you find a pair of images (`img001_diffuse` and `img002_diffuse`) showing the same object from slightly different views (different view angle). These images are taken from the DTU robot data set `http://roboimagedata.imm.dtu.dk/`. We can compare interest points from image A (`img001_diffuse`) with those in image B (`img001_diffuse`) and find

points that match. We recommend you use the small gray scale versions of these images.

Apply your interest point detectors to these images and collect all your detected points. For each point extract a small square patch of pixels centered on the detected point. Let us use this patch as our feature describing the local image structure around the interest point. As a dissimilarity measure we will use the normalized cross correlation between the two patches that we are comparing. To select a match between a point in image A and points from image B, you can either pick the point pairs with the smallest dissimilarity measure, or look at the ratio between the dissimilarity measure for the closest match divided by the dissimilarity measure of the second closest match and keep all points with a ratio below some threshold value (you need to find a good threshold). You may risk having several points from image B matching with the same point in image A and need to be able handle this situation.

A popular approach to illustrating results on matching interest points between two images, is to put the two images beside each other and then draw lines between the matching interest points. Illustrate the performance of your implementation using this approach and comment on your results. What patch size should you use to get good results? Can you think of some normalization steps you could apply to the patch in order to obtain a more robust feature?

Try to repeat the experiement by matching `img001_diffuse` with `img009_diffuse`. The difference in view angle with respect to `img001_diffuse` is larger for `img009_diffuse` than for `img001_diffuse`. Explain why your performance on this pair of images changes.