University of Stavanger

Faculty of Science and Technology

Stavanger, August 25, 2021

# ELE610 Robot Technology, autumn 2021

# Image Acquisition assignment 1

For this assignment each group should write a brief report (pdf-file). Answer the questions and include figures and images as appropriate. You may answer in Norwegian or English, or a mix. Note that to give exhaustive answers to all questions and tasks below within reasonable time is impossible for most students. Especially, part 1.5 below will take long time to solve perfectly. The intention here is that you should do as much as you are able to within the time limit for each assignment, which is 15-20 hours. A report containing a table showing time used, for each student of the group, will normally be accepted even if all tasks are not done. If all tasks are done, the time report does not need to be included.

## 1 Image acquisition using smart phone

The first task is to look up technical information on the smart phone camera, and to understand what the different properties means. Note that not all listed properties are available or relevant for the smart phone camera. Then you should capture an image using the camera on your smart phone and transfer it to your laptop. Finally some simple image processing should be done.

Karl Skretting, Department of Electrical Engineering and Computer Science (IDE), University of Stavanger (UiS), N-4036 Stavanger. E-mail: karl.skretting@uis.no.

## 1.1 Camera properties

Many years ago, when cameras were analog, there was some few common properties to adjust when an image was captured, mainly: exposure time as fractions of second (lukketid), the physical shutter mechanism (lukker), aperture size as f-number (blender). When digital cameras were introduced more properties (both for camera and image capture) was added, but the old properties related to the lens and objective are still relevant.

Below is a list, helter-skelter i.e. totally unsorted, of camera related terms. The **purpose** of this list is that it gives some key-words that may be useful to refresh or learn a little bit on cameras and photography. The **task** here is to briefly explain each of them and if possible relate them to the camera you mainly use. Technical information and camera properties are not always easy to find, especially for cell phone cameras, as most customers are more interested in practical information, subjective image quality and performance. Nevertheless, here are the list of some terms:

a. Resolution, several options may be available
b. Chroma and pixel depth, several options may be available
c. Flash
d. (external) Trigger
e. Frame rate
f. Sensor type, most common are CMOS and CCD.
g. Optical area, and pixel size
h. Shutter
i. Exposure time (minimum - maximum)
j. Lenses and lens parameters
k. Sensitivity, (for old/film type cameras: film speed, ISO number),
l. Quantum efficiency
m. Memory, storage
n. Image processing (compression)
o. Interface (connector)
p. Stabilizing

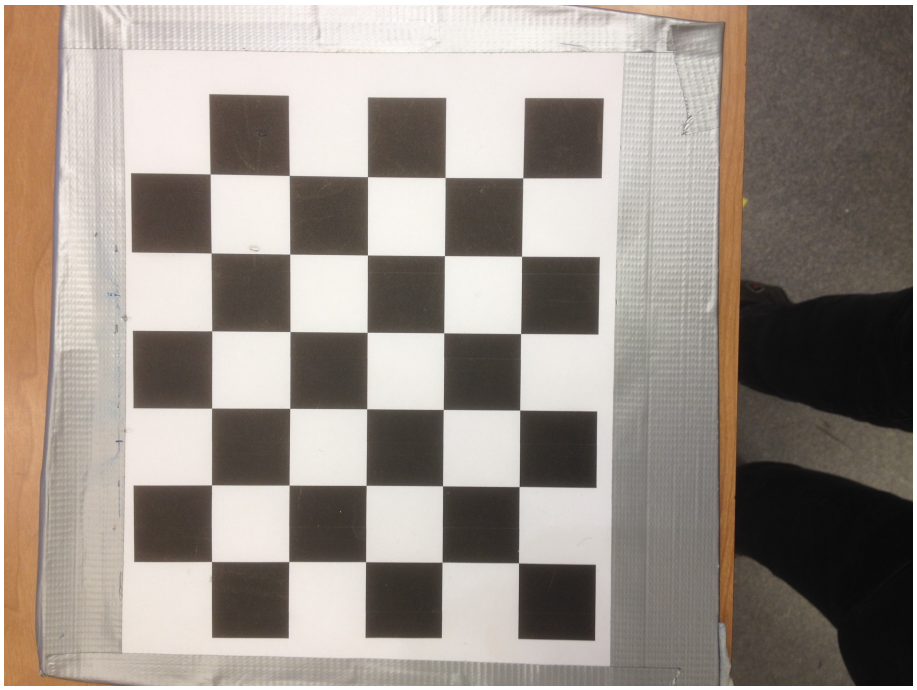

Figure 1: Image from Iphone 4, width 3264 and height 2448 pixels.

## 1.2 Capture a test image

The task here is to capture an image using the camera on your smart phone. The scene should be simple, a chessboard pattern or an image with black and white areas and straight edges or lines. Transfer the image to your laptop, or another computer, and view it in an image viewer program, or MATLAB, to explore the image. Answer the points below, a chessboard pattern is assumed.

a. Include the image in your report.

b. What meta information is available for the image.

c. Explore pixels of black areas. What is the common color for these (RGB values), what is the range (for RGB values) in black areas.

d. Explore pixels of white areas. What is the common color for these (RGB values), what is the range (for RGB values) in white areas.

e. How can sharpness be assessed? Are the edges sharp?

f. Do image properties, i.e. RGB values for black or white areas and sharpness, vary locally?

g. Are all straight lines in the scene also straight lines in the image? Should they be?

h. Each of the black squares of the chessboard has a true side length, measure this. Also, each of the black squares in the image has a side length in pixels, count this. Are all side lengths the same, should the be?

i. How can you use the camera information and the measures found above to calculate the distance from camera (lens center) to the chessboard (center)?

## 1.3 Install and check Python

The first thing to do here is to install Python, including all the needed packages. To do this, carefully read the instructions in the Fragments of Python stuff document. Make the decisions appropriate for you, do the necessary installations, and check that everything is as it should be. To confirm the installation answer the following questions.

a. What kind of computer do you use?

b. What kind of OS do you use?

c. Which version of Python (`sys version`) do you use?

d. Which editor, or IDE, do you use?

e. Which version of `numpy` do you use?

f. Which version of OpenCV do you use?

g. Which version of Qt do you use?

h. Have you installed and checked `pyueye`?

i. Have you installed and checked `qimage2ndarray`?

## 1.4 Image Processing using Python and Qt

Qt is a library for writing GUI programs. A very basic example of an image viewer is in `appSimpleImageViewer.py`. If you are new to Qt you should start with this program, first run it and see how it works. Then read the code and try to understand both the overall structure and each line of this code. On *canvas* in the Modules (Moduler) section some **video lectures** are published, these may be helpful to understand how the image viewer programs work. I recommend that you start with the first video (appSimpleImageViewer). If you have some experience with Python or Qt you may rather start with `appImageViewer.py` which is slightly larger and more complicated (?). There

are even more simple examples of Qt applications available in the Fragments of Python stuff document. Anyway, the huge Qt documentation will be useful, even if it is from a C++ perspective.

Your task here is to extend the `appImageViewer.py` program by adding new functionality, your solution should be named `appImageViewerX.py` where `X` is the letter for your group and laboratory desk. It is when you try to do some programming yourself that you really see how much you understand. If you struggle to understand Qt you may answer the first questions here and go on to the next section.

a. Download the example, the simple example in `appImageViewer.py` or the even simpler example in `appSimpleImageViewer.py`, and run it.

b. Read the code carefully and try to understand most parts of it.

c. Load the image of the chessboard pattern into Python as a `numpy` array.

d. Add a new function to `appImageViewer.py` for cropping the image. You may add a menu 'Edit' under the main menu and put 'cropImage' into that menu. I recommend that you make it easy here, just crop the central (fixed) part of the image or use the `QInputDialog`-class to get the area to crop.

e. Add a new function for converting the image into gray scale. You may put 'grayScale' into the menu 'Edit'.

f. Add a new function for saving the image into a new file. You may put 'saveFile' into the menu 'File'.

My suggestion for how to solve these points are in `appImageViewer1.py`.

## 1.5 More Image Processing using Python and OpenCV

To solve the problems below you should use the OpenCV documentation, and perhaps also the tutorials, whenever needed. Especially the `imread, cvtColor, Sobel, sepFilter2D, threshold, Canny, HoughLines, HoughLinesP, line, ...` documentation will be useful here.

Below I indicate that the solution to these tasks can be integrated in the Qt program from last section, `appImageViewerX.py`. However, it may be better to start from `appImageViewer1.py`, this program also include suggestions for how to solve some of the questions below. Your solution here should be named `appImageViewer1X.py` where `X` is the letter for your group and laboratory desk.

If you found Qt awkward and (very) demanding you don't need to use it here, the more important thing here is to get experience with OpenCV. For this purpose, you may alternatively try to make a simple (set of) command line python program(s). Then, `matplotlib` function(s), or `cv2.imshow(..)`, can be used to display the results. Note that the 2-3 first points below don't use Qt anyway.

a. **Optional if Qt is used later.**
   Display the image, the chessboard pattern, on screen using OpenCV function(s). `cv2.imshow(..)` and `cv2.waitKey(..)` should be used. I made this task voluntary as other methods to display the image may be more flexible, i.e. `pyplot` to make a figure (to plot or include in a report) or Qt to include the image in a GUI.

b. Display the image on screen using `matplotlib` function(s), i.e. `pyplot`. You should include a screen-dump (or window dump) in the report as well as the code lines used to display the image.

c. Observe what color space the image representation is, RGB or BGR?

d. Make and display a histogram of the pixel values in the gray scale image. Include the resulting histogram plot in the report. You may add this as an option in the `Edit`-menu in `appImageViewer1X.py`.

e. Emphasize image edges using the `Sobel` filter. You may look at `appImageViewer1.py` for an example of how it could be done.

f. Threshold the gray scale image into a binary image.
   OpenCV thresholding tutorial may be helpful. You may look at `appImageViewer1.py` for an example of how it could be done.

g. Locate lines using `HoughLines` and `HoughLinesP` functions. If you chose to include this in `appImageViewer1X.py` you may simply print (some selected) results from the functions.

h. Plot the lines into the image. The `draw_lines` function in `HoughLines` tutorial may be useful. Include the resulting image in the report. You may also add this as an option in the `Edit`-menu in `appImageViewer1X.py`.

i. You may further extend the program by adding some more functions. Perhaps, locate the corners where vertical and horizontal lines cross, count the number of black squares (and white squares). Even more challenging would it be to locate the squares when the image (motive) is rotated an arbitrary angle, perhaps in that case a new menu item `Align` would be helpful.

Remember to keep track of hours used for this exercise, when each student has used 20 hours you should stop.