University of Stavanger

Faculty of Science and Technology

Stavanger, September 23, 2021

ELE610 Robot Technology, autumn 2021

Image Acquisition assignment 2

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. 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.

2 Image acquisition using μEye XS camera

The Imaging Development Systems GmbH (IDS) μEye XS camera should be used in this assignment to capture an image, and to do some simple image processing on the image in Python. Generally, the IDS web pages are good, but **access is restricted**, only registered users may access many of the technical pages. The registration is free, and not too complicated. The most useful, for us, parts of the IDS documentation are listed below.

- The uEye Camera Manual page display the main documentation parts in the left part and has links to useful pages in the main part.
- IDS Camera Manager, select the camera to use, if several are available, and display camera properties.
- uEye Cockpit, adjust camera parameters and capture image or video.
- Part C Programming describes the function that also can be used from Python through the `pyueye` interface.
- Part D Specifications gives a detailed specification of the camera.

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

2.1 Use IDS programs

You should install the latest camera driver from IDS, per August 2018 it is version 4.91. Drivers are available at IDS downloads. There are drivers available for Windows, 32 bit and 64 bit, and Linux. The same driver can be used for all IDS cameras. Only registered IDS users can download the drivers, the registration is free, and not too complicated.[1]

Install drivers and programs from IDS on your laptop, or use on of the laboratory PCs where this software is installed. Attach camera to USB-gate on the PC and start "IDS Camera Manager". The attached camera should be visible in "Camera list" on the top of the program window. The buttons in the middle of the window will display general information and specific camera information. You may double-click on the line showing the camera to start the μEye Cockpit program. Especially note the help button which will start the useful "User Guide". Try this and learn to know which sections are available, and use this guide whenever needed.

Back in μEye Cockpit program you can now try the different alternatives, behind each of the larger buttons. Try some of these, in particular investigate the different options that may be adjusted for this camera. Eventually, use the "Optimal Colors" button and capture and display image and video. The scene should include some of the colored dices that should be on the camera rig table. Save one image and include it in the report, on example is in figure 1.

2.2 Use μEye camera and Python

This section continues the Python section in assignment 1. You may need to do parts of the tasks in previous assignment to continue in this assignment. In particular you need to install Python and all the needed packages. To do this, **carefully read** the instructions in the Fragments of Python stuff document. After installation you may continue directly with the list of tasks below. Always remember that it may be helpful to look back at the work you did in the last assignment, and perhaps even do some of the tasks you skipped earlier.

The example solution, `appImageViewer2.py`, builds on `appImageViewer1.py` by adding a Camera-menu to it, and the example solution, `appImageViewer3.py` also add a Color and a Dice menu, each with some few options, i.e. small examples of how a special task can be solved. It is when you try to do some programming yourself or modify existing code that you really see how well you understand Qt and OpenCV. These examples are not complete, there is still

[1] If you dislike registration, students may copy the driver from another student or the teachers laptop

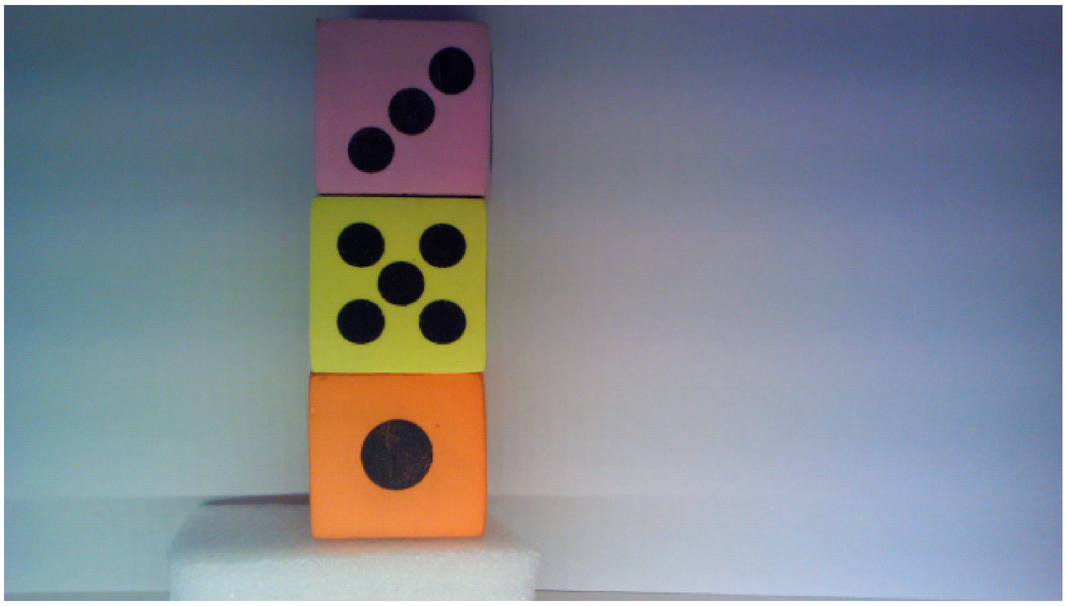

Figure 1: Image of dices captured using the uEye XS camera on UiS camera rig. The image is down sampled to size 640x360.

some (much) work left to make your program `appImageViewer2X.py` better (perfect?).

a. This task, and the next ones, do not use the Qt interface, although it may be helpful to see the example `pyueye_example_*.py` to better understand image acquisition. Install `pyueye` and check that it works as intended. You may use the `cam_info()` function in `bf_tools.py`[2]. Read the code in `cam_info()` carefully and try to understand it. In particular, look up the documentation for the used functions from the `pyueye` package; C++ versions of these functions are documented in uEye Camera Manual.

b. Make a function in Python that captures an image from the IDS XS camera (and stores it in a file). It may be helpful to look in `appImageViewer2.py`, but the point here is to make a simple not-Qt Python image capture program.

c. This task, and the following ones, all use the Qt interface. Install and run the example `pyueye_example_*.py` from the Fragments of Python stuff document. This code gives an example on how to use the IDS Python API, but don't worry if you don't understand all of it. Then, you can look at the code in `appImageViewer2.py`, which gives another, perhaps simpler, example on how to use the IDS Python API.

[2]I am not yet satisfied with `bf_tools.py`, it will probably be improved, or deleted, later.

d. Read in `appImageViewer2.py`, the program that is a preliminary example of the solution for this assignment, and try to understand the camera menu and what is done in these parts. You may also make your own program `appImageViewer2X.py` based on this, and perhaps make some improvements. Perhaps print more camera information, perhaps capture two images just after each other and find the difference between the two images, subtract the first from the second image. More difficult tasks are to change the camera options (not all are possible to change) and to capture a video sequence.

e. Add another menu `Dice` (in `appImageViewer2X.py`) and under it add an Action for `Black dots`. You should try to find the black pixels using "toBinary"-example in `appImageViewer1.py`. Morphological functions can remove noise and only keep the eyes as the black dots. Do not make the task more difficult than necessary, use only one dice and have a smooth background without black areas.

   The `Find circles` action uses `cv2.HoughCircles()` function and is an alternative to `Black dots`, this action/function is used in `appImageViewer3.py` and you may also look there.

f. Finally, add a feature, an action, `Count eyes`, that finds the number of eyes in the captured image after black dots are found. Place it under the new `Dice` menu in `appImageViewer2X.py`.

g. Make your solution `appImageViewer2X.py` even better. Correct any errors, clean up any messy part, make comments sufficient, but not too verbose.

For the last three points above I don't expect you to finish now, but don't worry, the next assignment continue using dice images and make your next (improved) solution in `appImageViewer3X.py`.