Stavanger, December 20, 2024

University of Stavanger

Faculty of Science and Technology

ELE610 Applied Robot Technology, V-2025

Image Acquisition assignment 4

In this assignment you will continue to develop the simple image viewer and image processing framework using Python and Qt. A lot of ideas, hints and possible ways to make progress and to what could be implemented in your final solution is given in this assignment. Thus read the complete assignment before you decide what to do, this should depend on your programming skills.

Make a plan (list) with several steps. The more, and smaller, the steps are the better it is. Start the plan of what to do with some simple tasks. Then start doing the work: specify and document, program and test step by step. Check that solution (program) works after each new step is implemented.

Finally, you should have made your own improved image viewer program. If you are group X an appropriate name for your program is `appImageViewer4X.py` and this program should (probably) extend `appImageViewer1.py`.

For this assignment each group should write a brief report (pdf-file). Include your plan, and figures and images as appropriate. The intention here is that you should do as much as you are able to within the time limit, which is 15-20 hours for each student. A report containing a table showing time used, for each student of the group, will normally be accepted.

4 A rotating object in the scene

The Imaging Development Systems GmbH (IDS) μEye CP camera should be used in this assignment to capture an image of a scene containing a rotating object. The object here, on either of the two UiS Image Acquisition Test Rigs, is a disk faced directly towards the camera. The same camera driver as in assignment 2 and 3 should be used here.

The cameras on the two rigs are slightly different but should be similar to work with. On the new (the smaller one) camera rig the camera model is `UI-3140CP-C-HQ R2` ↗ and on the old rig the camera model is `UI-336xCP-C` ↗.

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

Each camera is mounted on one of the UiS Image Acquisition Test Rigs and can be connected to the computer using the USB3 interface, you may look at IDS paper on USB 3 ↗. For both rigs there is a trigger connected to the camera. Information for the camera can be found by scanning the QR-code attached to the rig, assuming your device is connected to Eduroam.

The two camera rigs are work in progress, continuously being repaired or developed mainly by Asbjørn, and the current features may be somehow different from what is written below. On both camera rigs the spinning disk is driven by a motor and the speed can perhaps be adjusted. On the new (the smaller one) rig it may be possible to read out the true speed. Using Python and image processing (OpenCV) the task is to find and show the angular speed of the disk (rotations per minute, rpm) based on one captured image, or a sequence of images (video).

Attach the 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. You can now try the different alternatives. Try some of these, in particular investigate the different options that can be adjusted for this camera. Note that more options can be used for the advanced CP camera than for the more simple XS 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.

It should also be possible to adjust some of the camera properties using the Python class made to make the use IDS Python package easier to use (for some basic functionality), i.e. the file `clsCamera.py` ↗.

4.1 Angle of disk in image

We could use at least three different approaches to measure the angular speed of the disk, as briefly indicated in the three points below.

i) If you know the exposure time the speed can be calculated based on how blurred the moving parts of the image is.

ii) If a video sequence of images exists and you know the frame rate the speed can be calculated based on how much the red sector (or the blue dot) has moved from one frame to the next

iii) If a trigger is available, which trigger the image capture when the object at a fixed position, and there is a small, fixed and known, delay from

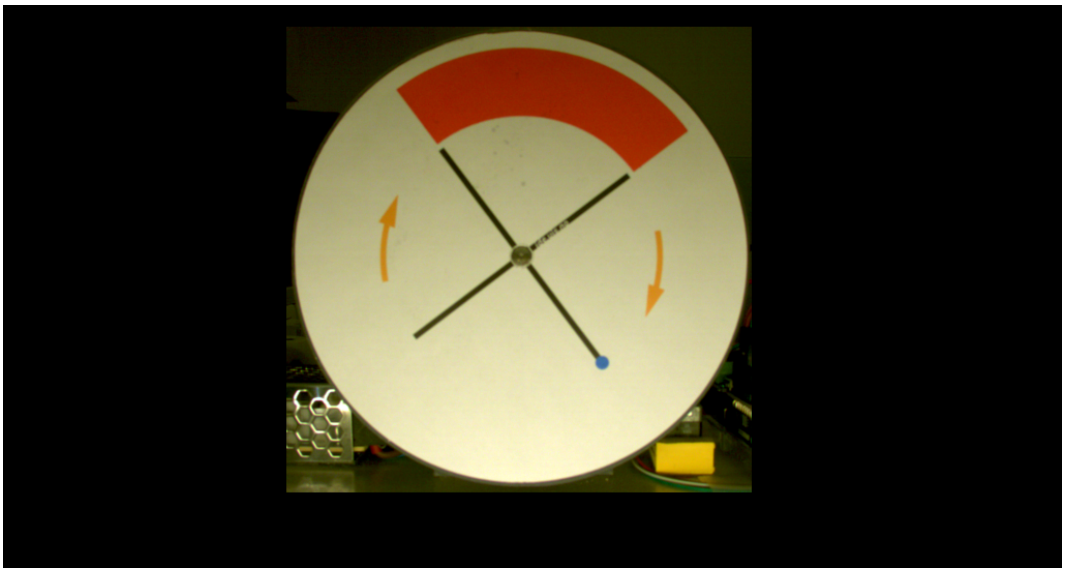

Figure 1: Image of disk on UiS camera rig. The disk is not moving and disk is in "trigger" position, i.e. the pin on the backside of the disk is positioned in the trigger optical switch, obstructing it. The image is size 2048x1088 (down sampled to 1024x544), but camera property size is set to 900x900 (down sampled to 450x450), which make the outside pixels black (but still in the image).

trigger time to exposure time, the speed can be calculated based on how much the object has moved from trigger position.

The last one, number iii), is the one you should do first here, and only if this does not work you may try one of the other methods. The list below gives some hints on what to do.

1. Check the Camera menu from `appImageViewer2X.py` for the CP camera, update the Camera On and Camera Off actions if needed, and make the program aware of which camera that is connected.
2. Update the Image Capture action on Camera menu if needed.
3. Add a feature (an action) that print information on the CP camera, or XS camera if that is what is connected. There should (probably) be some differences on what information is printed depending on camera type.
4. If you know the exposure time the speed can be calculated based on how blurred the moving parts of the image is. Don't try to do this before the last point is tried.

Figure 2: Image of disk on UiS camera rig. The disk is rotating and image is taken at random time. The image is size 2048x1088 (down sampled to 1024x544), but camera property size is set to 900x900 (down sampled to 450x450), which make the outside pixels black (but still in the image).

5. If a video sequence of images exists and you know the frame rate the speed can be calculated based on how much an object has moved from one frame to the next. Don't try to do this before the last point is tried.

6. If a trigger is available, which trigger the image capture when the object at a fixed position, and there is a small, fixed and known, delay from trigger time to exposure time, the speed can be calculated based on how much the object has moved from trigger position. This is the method you should try first.

4.2 Some ideas and hints

A basic feature of the program is to locate the disk in the image and to find the angle that the disk has rotated, relative to an initial position. Thus below the first steps towards getting speed based on angle of disk should be done. If you find it difficult to start you may look at my suggestion `appImageViewer3.py`. The list below gives more ideas on what to do.

a. Let the disk be still in "trigger" position, as in Fig. 1. Use the IDS μEye Cockpit program to adjust the camera options and capture an image.

b. Crop the image so that it contains only the disk. The crop function in `appImageViewer1.py` can be used as it is (even thought it is not good). Find out how it works and use it on the image from previous point. You may set all irrelevant pixels to black, perhaps this must wait to after next point, i.e. all pixels with a distance to the center larger than the disk radius are set to 0. You may also make the image gray scale and store it as a JPG image, which take much smaller space than the BMP image (perhaps) used when IDS μEye Cockpit program stored the image.

c. Find the center of the disk (pixel in image), these coordinates are needed to find the angle of the red sector later. First, simply look closely (in an image viewer program) at one image and find the coordinates of the disk center and use these coordinates as center. Then skip the rest of this point and continue with the next point, and only if you have more time in the end you may try to automate how to find disk center coordinates. You may then add a point to the menu that find the disk center coordinates, perhaps using a shortcut too. Since shortcut C is taken (Clear image) you may use shortcut M (for Mid point). Several ways to do this can be considered, explain your method well before you try to implement it. The method in `appImageViewer3.py` is rigid, it may be improved by adding a dialog, similar to the dialog classed used in `appImageViewer1.py`. It may also be that other methods can solve this problem just as well.

d. Find the angle of the disk. Do it manually first by measuring on the screen or on a paper, then make a Python function that does this. This could be the angle of the blue dot relative to clock background in Fig. 1 it is between four and five, or better, it can be the center of the red patch which here is just after twelve. Explain the definition you use for the reference angle and how you find it.

e. Let the disk be rotating slowly and capture a new image, as in Fig. 2. Find the angle for this image. Give the answer in degrees between 0 and 360.

To change unit for rotation speed notice that
6 [deg./ms] = $\frac{1}{60}$ [rot./ms] = $\frac{1000}{60}$ [rot./s] = 1000 [rpm],
where rot. is short for rotations, deg. is short for degrees, and rpm is rotations per minute.

Assume that the image of the spinning disk is correctly captured using the trigger. Then the angle θ for an image point (the center of the red sector or the blue dot) is related to the speed of disk ω [deg/ms], the delay time t_d [ms] and the exposure time t_e [ms] by

$$\theta = \theta_0 + (t_d + t_e/2)\ \omega \tag{1}$$

where θ_0 is the (limit for the) angle in degrees for the same point when $t_d = 0$ and t_e is approaching zero.

4.3 More ideas

Let the disk rotate with approximately half of maximum speed. Use the IDS μEye Cockpit program to set camera option to trigger mode.

a. Let the trigger delay time be 10 ms. Capture single images and observe that the angle of the disk will be the same for each image in this set. What is the angle of the disk in these images?

b. Adjust the trigger delay time to 50 ms and capture a new set of images. What is the angle of the disk in these images?

c. What is the difference for the angle between the second set of images and the first set of images? What is the delay time difference?

d. Explain how you can use the two numbers from previous point to calculate disk rotation speed in rpm (rotations per minute). Do the calculations, and find the answer.

e. What angle would you expect the disk to have if the delay time were 0 ms?

f. Increase the rotation speed of the disk, keep the trigger delay time to 50 ms and capture a new image. What is the disk angle now? And the disk rotation speed?

4.4 And more tasks

The points below are intended to be ideas and hints more than a TODO-list.

a. In assignment 3 you should have made three actions on the Camera menu of the program; open/initialize the camera, capture an image, and close the camera. Make sure that these three features (actions) works as intended and include (copy) them to your program. Alternatively, you may try the similar functions made in `appImageViewer2.py`.

b. Add a feature (an action) to the program that find the position of the disk center in the image, as done before. The results may also be printed to standard output.

c. Add a feature (an action) to the program that find the angle of the disk as done before. The results may be printed to standard output.

d. Add a feature (an action) to Camera menu that set the trigger on and that also set the trigger delay time to 50 ms. You don't need to use a dialog window here.

e. Modify the action in the program that find the angle of the disk to also find the rotation speed. Both results should now be printed to standard output.

f. Add an action to the program that check that trigger is on, capture an image using the trigger and calculate the rotation speed. The action could be on the program menu and perhaps have shortcut Ctrl+R.

g. Now, if you still have time left, you can clean up the program, make sure that all the features works as intended. It would be nice if the program detect by itself if a IDS camera is attached and which camera it is, and adjust the user interface accordingly.

h. If you still have time left, you can add an action to the program that start the camera in video capture mode, frame rate could be 1-3 fps. The action may also start a dialog window that display disk rotation speed, perhaps both by displaying a number and a slide bar. The dialog window should also have a Stop button that stop the video and close the window.