Lab 2

Aruco markers

Machine Vision

General Goal

The goal of this project is to introduce how to use aruco markers which is a part of the OpenCV library. Students are also required to work with cameras connected to a system via USB.

Prerequisite

Each exercise is connected to Lab 1 and a lot of code from this project will be reused.
In addition, you will need a web-camera or a video feed and a print-out of an aruco marker that is included in the files for this project.
Tip: It is not mandatory, but it is good if you have a white background when doing this lab.

From OpenCV you can find the following new commands useful to be able to finish the exercises:

- Functions:
  - cv2.VideoCapture
  - cv2.adaptiveThreshold
  - aruco.DetectorParameters_create
  - cv2.aruco.Dictionary_get
  - aruco.detectMarkers

Hand in

Students should hand in a zipped file containing a video showing that the code works as well as the actual code. You must comment your code well where you point out functionalities and how the image “travels” trough your program to the final output shown on the display.

Note: Submissions that fail to follow above mentioned pattern will not be assessed and will be reported as failed. No resubmission is allowed in this case.

Grading

Each exercise are pass/fail.

Step 1 – Preparation of code from first lab

Create a script called MyDetectionMethods.py and create a class called MyDetectionMethods. Inside this class create methods that accept image data as input and returns contours. This method should contain the image preprocessing steps done in exercise 6 in the first project. Make one method for canny filter and one for binarization.

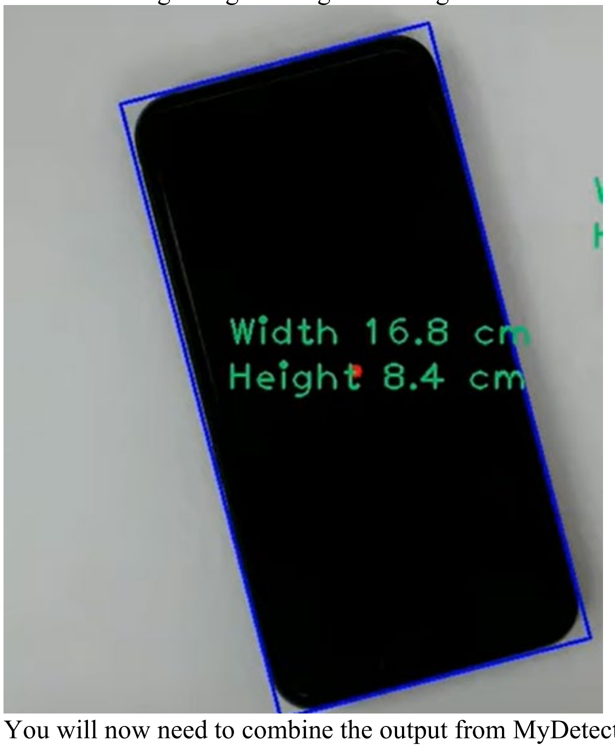
Step 2 – Detecting the Aruco marker

OpenCV includes several predefined dictionaries of ArUco markers, each with a different number of markers and sizes. These dictionaries help in generating and detecting markers efficiently. ArUco markers are detected by identifying their square shape and decoding the binary pattern inside. This allows for robust detection even in varying lighting conditions and angles. This makes them perfect for this lab where we are going to use them as know objects to measure unknow object using something called ***pixel to cm*** ratio.

Create a new "main" script that can call your class MyDetectionMethods.
Now create a live stream from your camera that can detect and draw a box around the aruco-marker (if you don't have movable camera you can record a movie with your phone and use that as a stream instead) you have been given on canvas, you might need to print one.
If correctly programed, you should get a green rectangle around the aruco-marker.

Step 3 – Object Measurement

Place an object beside the aruco-marker. Make the program draw a box around the detected object and give information regarding the length and height in the window, such as this:



You will now need to combine the output from MyDetectionMethods (which should be countour object) with the ***pixel to cm*** ratio calculated from the aruco marker to make a proper measurement.

Step 4 – Specific Object Measurement

Place multiple objects at the same time in front of the camera. The program should be able to detect objects approximately of credit card size (85.6 mm wide by 53.9 mm high) and of an AAA battery cell size (10.5 mm in diameter and 44.5 mm in length, including the positive terminal button) with +/- 5 mm precision. All other object needs to be ignored by the program.