Assignment 4: Markers and Computer Vision (15P)

Goals

You are familiar with basic functions of the OpenCV library. You can write programs to detect fiducial markers, for example AruCo. You know about perspective transformation and can apply it to extract and warp regions of an image.

1 Perspective Transformation 5P

Recommended Packages: *OpenCV*

Write a program called *image-extractor.py* that loads and displays an image with OpenCV. By clicking into the displayed image, users should be able to select four points. The selected region is then extracted and perspectively warped to a rectangle. After all four points have been selected, the warped result should be displayed. By pressing `ESC`, users should be able to discard the changes and start over. By pressing `S` in the result view, the image should be saved.

Paths to the input file and output destination, as well as the result's resolution should be specified via command line parameters.

Score

(**1P**) The image is successfully loaded and displayed.
(**1P**) Selecting the corner points works and there is visual feedback for the user.
(**2P**) Perspective transformation to the target resolution works.
(**1P**) Command line parameters and shortcuts work.

Hand in until 22.05.2024 23:59!

2 AR Game 10P

Recommended Packages: *OpenCV, numpy, pyglet*

Create a program called *AR-game.py*. The program should read out your webcam image. Use a board with an AruCo marker in each corner, extract the region between the markers, and transform it to a rectangle with the resolution of your webcam. Keep in mind that not all webcams have the same resolution! Display the extracted and warped rectangle in a pyglet application. Now, add some game mechanics based on the image in the extracted rectangle. For example, players could be able to use their finger to destroy targets or to move things around.

Score

(**2P**) The region of interest is detected, extracted, transformed, and displayed.
(**4P**) Objects (such as fingers) in the region of interest are tracked reliably and interaction with game objects works.
(**2P**) Game mechanics work and (kind of) make sense.
(**1P**) Performance is ok.
(**1P**) The program does not crash.