Assignment 4: Markers, Computer Vision, and AR (20P)

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

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

3 AR Game – now 3D **5P**

Recommended Packages: *OpenCV, numpy, pyglet*

Now, we will move into a 3D space. Use multiple markers with different IDs to project different 3D models to your webcam image. You can use the example program *AR_sample_3d.py* which imports *AR_model.py* – this already makes a 3D model appear on a marker. You do not have to model your own 3D objects but can use existing ones – cite your sources![1] Again, implement simple game mechanics to make different 3D objects interact with each other or with the user.

To name a few examples to do so:

- You can again detect objects (such as fingers). The finger can hit projected button(s) which results in firing an attack. The attack is visualized by another 3D object.
- You can again detect objects (such as fingers). The finger points to a certain position. The 3D object moves towards that position or fires another 3D object in this direction.
- You can add another character which appears at another marker. As soon as the two characters recognize each other, they react to that (e.g. by firing an attack, sprakling, moving towards each other, ...)

Score

(1P) At least one additional 3D model is added correctly to the webcam image – only appearing at its assigned marker

(1P) Objects (such as fingers) in the region of interest are tracked reliably or the integration of further (moving) 3D game objects works

(2P) Interaction with or between game objects works and is nicely visualized, game mechanics work and (kind of) make sense

(1P) Performance is ok and the program does not crash

[1]Hint: If you have found a 3D model you want to use but it is the wrong file type or not displayed correctly in pyglet, open your model in Blender, select the object and export it as *.obj*. This will give you all the necessary files (*.obj*, *.mtl*, and *.png* if needed). The files have to be in the same directory as your 3D game python script.