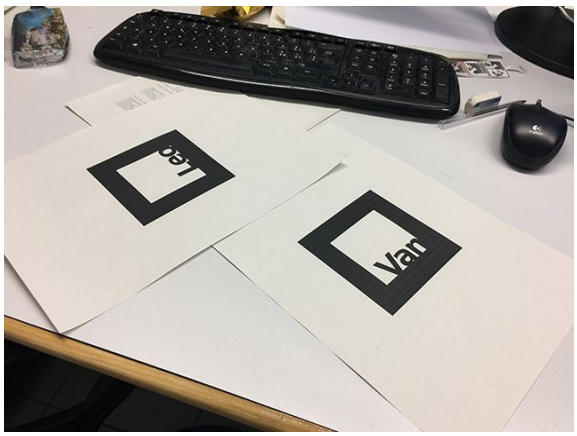
Computer Vision 2016/2017

Final Project

The final project consists in a development of a simple augmented reality system. You'll have to create a C++ program (using the OpenCV library) to process frames from either a webcam, a video file or single png images and replace all the occurrences of some pre-defined markers in a scene with their corresponding pictures.

The replacement has to be performed considering the projective transformation that affects each marker when captured by the camera. That is, the corresponding pictures should appear as if they were actually found at the scene at the same position, orientation and spatial extent of the marker. For example:

Acquired Input frame:	Expected output:
	

Attached to this file you can find all the pairs of marker models and the corresponding pictures to be substituted, together with some test image and video data. In particular, the marker model to picture mapping is shown in the following table:

Name	Model	Picture
Marker 0	0M.png 	0P.png 
Marker 1	1M.png 	1P.png 

Image warping

The transformation mapping each marker from its ideal reference frame (ie. as seen by a camera perfectly orthogonal to it) to the scene is a projective transformation described by a 3x3 homography matrix **H** transforming points from $P^2 \rightarrow P^2$.
When some marker points are identified from each acquired frame, the OpenCV function cv::findHomography can be used to compute the homography matrix **H** given a set of point-to-point correspondences (see the OpenCV documentation for details about the actual function implementation). For the theory behind the 2D projective transformations estimation techniques, see the Chapter 4.1 of (Hartley and Zisserman 2004)

When the proper transformation has been computed, the function cv::warpPerspective can be used to perform the warping between the two images.

Minimum requirements

To be successful, your program must correctly detect the markers and substitute the corresponding pictures in all the test images supplied (`test1.jpg`, `test2.jpg`, `test3.jpg` and `test4.jpg`). It's suggested to also test the program with the video files `test_video1.mp4` and `test_video2.mp4` that are designed to stress critical parts of the algorithms you are using (small angles, occlusions, camera blurring, etc). It's not required that you detect the markers for all the video frames but your code is supposed to work properly on the vast majority of them.

Optional Features:

You are encouraged to also display (on secondary video window) any debug information that may be useful to better understand how your program works internally. For example, the debug window may include a rendering of the detected lines and corners, the marker regions, the effect of any applied filters, etc.
Finally, a reasonable real-time processing speed will be evaluated as a positive feature of your work (although not mandatory to pass the exam).

Additional notes:

There are no particular constraints on the number of function/classes/cpp files produced as long as no additional libraries are used except for OpenCV. You are encouraged to reuse the code produced for the previous assignments where possible.

Comment your code whenever possible. Since no additional report is required, source comments are a good way to clarify what your code is supposed to do (it will also help you during the exam!).

Exam Information

Final course exam consists in an oral discussion about the project.

When ready to take your final exam, package the source code in a zip file named `<name>_<surname>_exam.zip` and submit it via moodle. Then, notify me via mail (filippo.bergamasco@unive.it) so that we can arrange a date (usually within a couple of weeks from the submission).

During the exam, I'll ask you to explain all the interesting parts of the source code, focusing on the implementation details and the algorithms involved. You are supposed to discuss

strengths and limitations of any choice you made during the development. During the discussion, you'll also have to show a demo of your project either on my laptop or yours, based on your preference.

I'll consider the exam passed with full marks if you demonstrate proficiency on the computer vision techniques we have seen during the course in relation to the project you have developed. This means that the actual final performance of the produced code is not critical if the behaviour is properly justified and discussed.

For any question feel free to mail me at filippo.bergamasco@unive.it.

Bibliography

Hartley, Richard, and Andrew Zisserman. 2004. *Multiple View Geometry in Computer Vision*.