

List of projects for the course  
3D Augmented Reality A.A. 2020/2021

Simone Milani

A.A. 2020/2021

## Description

Here's the list of the projects for the course 3D Augmented Reality, Academic Year (A.A.) 2020/2021.

The output of each project's work must be a short report (at least 4-5 pages without considering the source code or MATLAB script), which will be presented.

The report must be sent to the teacher at least 1 day before the presentation. Projects can be performed individually or in groups of two persons. The programming languages are MATLAB/C#/Python/C or C++.

**Projects A.** To be developed in Unity, they concern an Augmented reality application.

**Projects B.** To be developed in MATLAB/Python/C++.

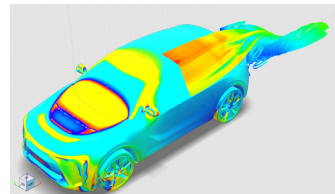
## Projects A

### Project A.1


#### **2020 Visualization Challenge: physical properties for mechanical models**

The goal of this challenge is to visualize in a Unity application some physical properties (pressure, aerodynamics) for a 3D mechanical model (a car). Datasets is made of a 3D model and a list of possible data localized in the 3D space around (and on the model).

Data includes pressure values (localized on the surface of the 3D model) and flow lines (sequences of 3D points - localized in space/time associated to air). See an example in the figure aside.



Students may design different interfaces and visualization (static or dynamic). Visualization strategies can include color mapping, 3D solid animations, particle systems. Projects will be presented at the final exam and ranked.

Dataset is courtesy of Altair [www.altair.com](http://www.altair.com).  **ALTAIR**

## Projects B

### Project B.1

#### Feature matching for Structure-from-Motion

Using the SfM software COLMAP <https://colmap.github.io/>, tests the effectiveness of different local descriptors in the 3D reconstruction (measuring the number of points, the average track length and the reprojection error). You need to provide to the software the list of keypoint positions and possible matchings.

In the experiments, you can use the descriptors

- SURF (implemented in OpenCV/MATLAB)
- ORB (implemented in OpenCV/MATLAB)
- BRIEF (implemented in OpenCV/MATLAB)
- binGAN (implemented in Python <https://github.com/maciejzieba/binGAN>)
- TFeat (in Python <https://github.com/vbalnt/tfeat>)
- LIFT (code in Python/C <https://github.com/cvlab-epfl/LIFT>).

Performances must be evaluated on the datasets Fountain and Castle ([https://github.com/openMVG/SfM\\_quality\\_evaluation/tree/master/Benchmarking\\_Camera\\_Calibration\\_2008](https://github.com/openMVG/SfM_quality_evaluation/tree/master/Benchmarking_Camera_Calibration_2008)) and Tiso and Portello (<http://www.dei.unipd.it/~sim1mil/materiale/3Drecon/>).

Programming languages: MATLAB/Python/C++.

### Project B.2

#### Local feature compression using autoencoders - SfM tests

Design a compression strategy for local SURF descriptors using autoencoders. Training data can be generated using the images of dataset Portello and Castle. Testing must be done on dataset FountainP-11 and Tiso (available at [https://github.com/openMVG/SfM\\_quality\\_evaluation/tree/master/Benchmarking\\_Camera\\_Calibration\\_2008](https://github.com/openMVG/SfM_quality_evaluation/tree/master/Benchmarking_Camera_Calibration_2008) and <http://www.dei.unipd.it/~sim1mil/materiale/3Drecon/>). Software must be implemented in MATLAB, Keras or Pytorch.

**Testing on 3D reconstruction using SfM** The reconstructed descriptors (only for the test set) are used to perform a SfM reconstruction using COLMAP (using the two test dataset).

Programming languages: MATLAB/Python/C++.

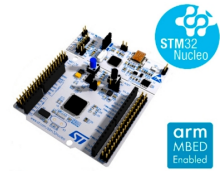
## Project B.3

### Local feature compression using autoencoders - embedded machine learning

Design a compression strategy for local SURF descriptors using autoencoders. Training data can be generated using the images of dataset Portello and Castle. Testing must be done on dataset FountainP-11 and Tiso (available at [https://github.com/openMVG/SfM\\_quality\\_evaluation/tree/master/Benchmarking\\_Camera\\_Calibration\\_2008](https://github.com/openMVG/SfM_quality_evaluation/tree/master/Benchmarking_Camera_Calibration_2008) and <http://www.dei.unipd.it/~sim1mil/materiale/3Drecon/>). Software must be implemented in Python TF (Keras).

#### Testing on ML boards

Final tests will be carried on ST Microelectronics chipset STM32L476. Using the STM suite it is possible to deploy the json and h5 models (trained in Keras) on the board, as well as controlling input and output. The designed autoencoder must be deployed on this ML board and evaluated in terms of power consumption and speed (STM suite provides these diagnostics).



Programming languages: Python.

The ML boards are courtesy of ST Microelectronics.

