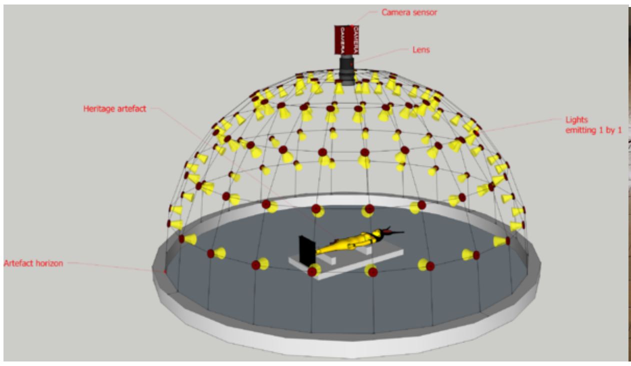
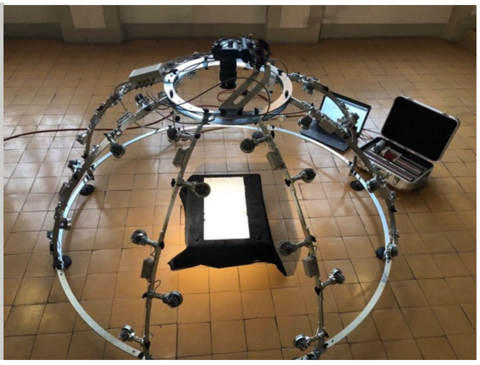
Geometric and 3D Computer Vision 2024/2025

Final Project – “Smartphone-based RTI”

Reflectance Transformation Imaging (RTI) is a technique to capture the reflectance of an object under different lighting conditions. It enables the interactive relighting of the subject from any direction and the mathematical enhancement of its surface shape and color attributes. Typically, RTI uses a static camera and an array of light sources at known (or knowable) positions.

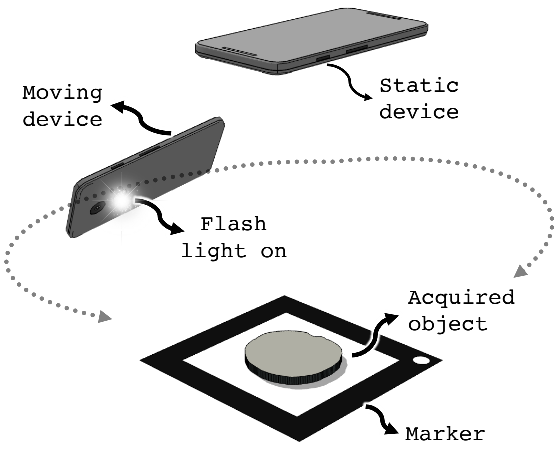




In this project, you’ll explore an RTI technique proposed in this paper:

Pistellato Mara, and Filippo Bergamasco. "On-the-go reflectance transformation imaging with ordinary smartphones." European Conference on Computer Vision. Cham: Springer Nature Switzerland, 2022.

It uses a cheap setup composed of two consumer smartphones equipped with a camera and LED flashlight. The idea is to use one of the two smartphones as a **movable light source** while the other captures the object from a frontal position. We exploit the camera of the movable light source to recover the light direction vector at each point and provide the same data obtainable with a light dome.



The acquisition process can be summarized as follows:

1. The target object is placed on a known planar square marker. We assume the object to be coplanar with the marker.
2. Smartphone 1 (**static device**) is placed above the object with the flashlight turned off. The camera starts to acquire a video sequence.
3. Smartphone 2 (**moving device**) is set with the camera and flashlight on. Then, it is moved around the object to illuminate it from different points of view. Care must be taken to ensure the marker is visible throughout the frames.
4. After the acquisition, videos acquired by the two smartphones are temporally synchronized.

In the post-processing, frames are analyzed to recover the light vector for each point of the object together with the acquired intensity/color. An interpolation function is then estimated to predict the point intensity for light directions that were not directly acquired. This will allow the object to be further relighted through an interactive user interface.

Mathematical details on how to perform the post-processing were given in a dedicated lecture of the course. If you have any questions, please get in touch with the teacher.

Calibration

The moving device must be calibrated before usage. Together with the RTI software, you are also asked to create a program for calibrating a camera using multiple exposures of a given chessboard.

The calibration software should work **without any user intervention**. In other words, it should load the provided calibration video file, detect the chessboard for at least 20 different frames (possibly evenly distributed among the sequence), and estimate the camera intrinsic parameters using the OpenCV function `calibrateCamera`.
After the calibration, the camera parameters should be written in a file and used for the subsequent operations. I suggest using a 5-parameters polynomial distortion model.

For additional information on how to use OpenCV to calibrate the cameras, refer to the following page:

https://docs.opencv.org/master/dc/dbb/tutorial_py_calibration.html

Important notes:

1. Image **frames must be undistorted** before doing any processing
2. **Do not use getOtimalNewCameraMatrix()** as shown in the tutorial. Just use the function cv.undistort() providing the intrinsic camera matrix K and the distortion coefficients k returned by cv.calibrateCamera()

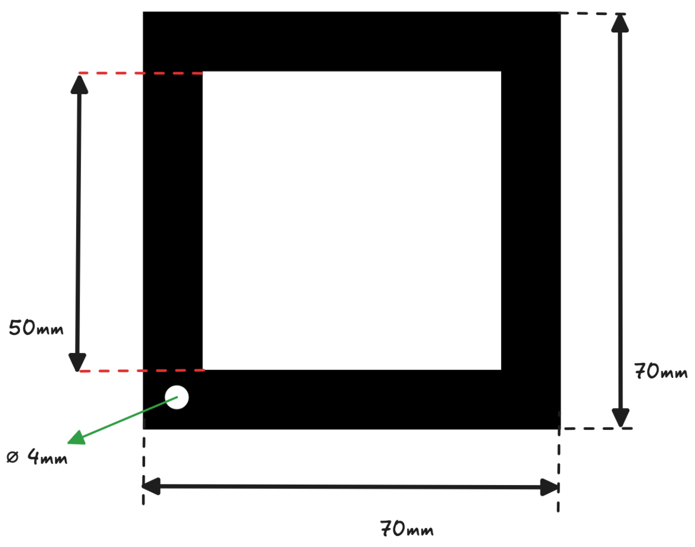
Video data

A package containing four different video sequences and the calibration video can be downloaded at the following URL:

https://www.dsi.unive.it/~bergamasco/teachingfiles/G3DCV2024_data.7z

Content:

- `cam1 - static` directory contains the 4 sequences: `coin1.mov`, `coin2.mov`, `coin3.mov` and `coin4.mov` acquired by the static camera. `calibration.mov` contains the video sequence of the calibration chessboard
- `cam2 - moving light` directory contains the same 4 sequences acquired by the moving light plus its calibration video
- `calibration_pattern.png` is the image of the calibration chessboard used (for reference)
- `marker.svg` is the square marker used. For reference, you can open this file with Inkscape to inspect its dimensions that are here summarized:



Note that the videos acquired by the two cameras are **not temporally synchronized**. That means that the first frame of cam1 does not correspond to the first frame of cam2. I suggest **extracting the audio track** of each video to compute the correct time skew. Moreover, the framerates are slightly different. Your programs should handle this difference when processing frames.

The developed project should fulfill all the requirements described below. The program should be kept general enough to work with any dataset, but parameters and algorithms can be tuned to "work well" with the given samples.

Requirements

The project must contain 3 different programs:

1. A *"camera calibrator"* that loads one of the two provided calibration videos and computes the intrinsic parameters of the camera (the intrinsic matrix K and the lens distortion vector assuming a 5-parameters model) **without any user intervention**.

2. The "analysis" program to process a video sequence and compute a per-pixel reflectance function f (ie a function mapping light direction and pixel position to pixel intensity). The whole analysis must be performed **without any user intervention**. Tunable parameters should not depend on the specific sequence processed.

   At least two interpolation functions should be implemented:

   1. **Polinomial Texture Maps**, proposed in the paper:
      Tom Malzbender, Dan Gelb, and Hans Wolters. 2001. Polynomial texture maps. In Proceedings of the 28th conf. on Computer graphics and interactive techniques (SIGGRAPH '01). Association for Computing Machinery, New York, NY, USA, 519–528. DOI: https://doi.org/10.1145/383259.383320

   2. **Linear RBF** implemented in the scipy library:
      https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.Rbf.html

   In both the cases, consider discretizing f into a fixed set of light directions and store the resulting interpolated values on a tensor.

   Optional: consider implementing the neural model proposed in the paper (Pistellato and Bergamasco) to avoid the discretization of f.

3. An "interactive relighting" program to render a previously processed object according to a user-definable light source direction

The three programs **must be written in Python3** and can use NumPy, OpenCV, and all the additional libraries you consider useful for the project.

Additional notes:

There are no particular constraints on the number of function/classes/Python files produced. You are highly encouraged to give visual feedback to better showcase each algorithm involved. For example, the analysis program could show the boundary of the square marker, the detected light direction, etc. Any debug information useful to explain the operations involved is welcome and will contribute to the final evaluation of the project during the oral examination.

A reasonable processing speed will be evaluated as a positive feature of your work (although not mandatory to pass the exam). The relighting program should work in real-time.

Comment your code whenever possible. Since no additional report is required, comments are a good way to clarify what your code is supposed to do.

Exam Information

The final course exam consists of 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). Please, **do not submit any data related to the project (video files, images, calibration data, etc.)**.

During the exam, I will 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 the strengths and limitations of any choice you made during the development. During the discussion, you will also have to show a demo of your project.

I'll consider the exam passed with full marks if you demonstrate proficiency in 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 behavior is properly justified and discussed.

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

09/12/2024
Filippo Bergamasco
Geometric and 3D Computer Vision a.a. 2024/2025