Assignment 2: Structured Light
(Informatica + EA-ICT)
Visual Computing

The goal of this project is to get 3D information, in the form of a point cloud, from a scene. This will be done using two cameras and a projector that projects specific patterns on the scene. These projections will give each point in our scene a unique index that can be used to match points between the left and right camera. These correspondences can then be used to retrieve 3D information.

1 Gray Codes

Gray codes are an ordering of the binary numeral system such that two successive values differ in only one bit (binary digit). For example, the representation of the decimal value "1" in binary would normally be "001" and "2" would be "010". In Gray code, these values are represented as "001" and "011"[1].
We will give each point in our scene a horizontal and vertical graycode. This combination of graycodes is unique within the image and the same point in the scene in the other image has the same graycode. These codes can thus be used to find correspondences.
In the `dataset` folder image sequences for both the left and right view are provided. These image sequences can be used to construct a horizontal and vertical binary graycode for each pixel. This graycode value should be tranformed into a regular decimal value which will give the unique identifier, see the Wikipedia page in the footnote for details. To construct the binary gray code value, loop through the images of the horizontal or vertical pattern. If a pixel is lit by the pattern, set the bit to `1` otherwise the bit is `0`. Do the same for all the other images in the pattern and append the bits to the right of the existing number. The first five images of a pattern are shown in Figure 1. To determine if a certain pixel is *lit*, the inverse of each pattern is also included. We consider a pixel lit if its intensity value is larger than the value of the same pixel in the inverse of the pattern.
Due to occlusions and shadows the projected patterns cannot illuminate each pixel. For these pixels we cannot use the graycodes to compute indices. You

[1] `https://en.wikipedia.org/wiki/Gray_code`

Figure 1: Graycode pattern projected onto a scene.

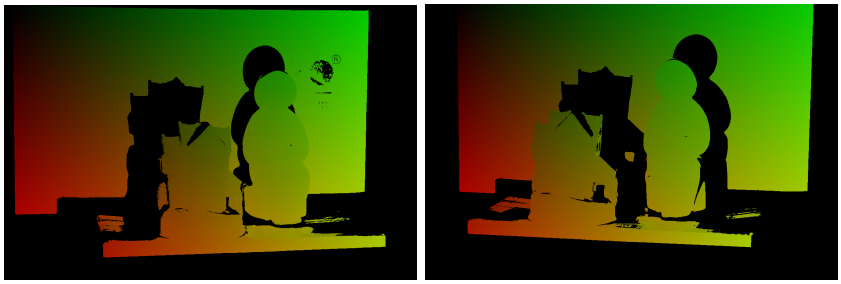

Figure 2: False color visualisation of left and right viewpoints.

should therefore compute a mask that indicates which pixels can be ignored when looking at graycodes. To do so we have included an image which is fully illuminated and one which is fully black. If the difference between falls below a certain threshold they cannot be used as the projector does not reach them.
Use computed vertical and horizontal codes to give each pixel a global identifier. Use this identifier to compute a false color visualisation. An example of such a visualisation is shown in Figure 2.

2 Calibration

Use the checkerboard patterns from the `chess` map to find the calibration matrix K of the camera. The sequences from both points of view were taken with the same camera, so only one matrix K has to be computed. Now remove the distortion from the input images using K. Read about camera calibration and undistorting images in OpenCV here.

3 Point Cloud Reconstruction

Use the global identifiers to find correspondences between the two cameras. Make a debug visualisation to check if the matches you found are correct. You could, for example, make a loop that selects a random match and draw a colored circle on the given locations in each image.
Use these found matches to compute the essential matrix. For this you can use `cv2.findEssentialMat`. Assuming the first camera is in the origin, compute the translation and rotation of the second camera. This can be done using `cv2.recoverPose` in OpenCV.

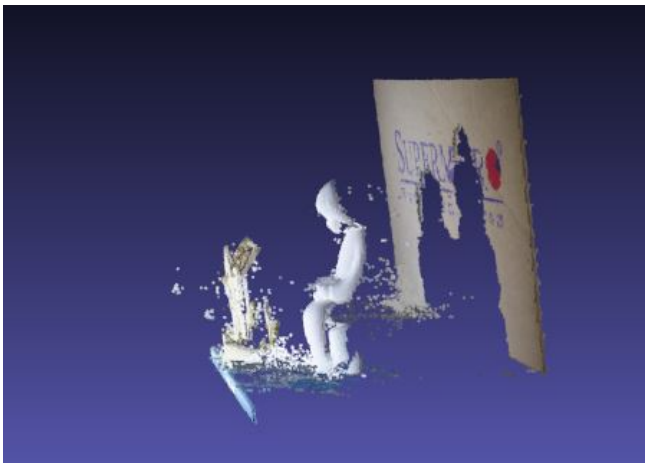

Figure 3: Point cloud of the given scene.

The 2D point correspondences can be used together with the calibration information to find 3D point coordinates. For this you can use OpenCV's `cv2.triangulatePoints` function. Finally, visualise the computed point cloud. For this you can use a point cloud viewing python package of choice. One example is PPTK. Give the point cloud some color by taking the pixel color values from the fully lit image.

Reporting

- Write a report explaining how you approached each step. Include some images showing the results of each step (False Color visualisation, point cloud, ...). Also provide your Python code to us.
- Your program code, the outputs and the brief report should be **committed and pushed to the GitHub repository of the assignment before Tuesday 22th of April at 18h00**.