Computer Graphics
Prof. Dr.-Ing. Hendrik P.A. Lensch
Jieen Chen

Eberhard Karls
Universität
Tübingen

Displays - WS 2017/18

Exercise 3

Submission deadline: Thursday, 14.12.17

- Please, upload your solution to ILIAS before the next exercise meeting!
- C++ and OpenCV are mandatory

Motivation: Assuming there is enough information about a real world object, fascinating mappings can be projected on those objects. For an example, see NuFormer: 3D mapping projection on buildings.

Task: In this exercise, we want to combine one camera and one projector into one integrated setup. Your task is to implement a basic mapping between them and to implement an edge highlighting projection. We start by projecting structured light onto a planar surface (wall) (Exercise 3.1) with the projector. Then, pictures of the projected light are captured with the camera. Based on that projected and captured data, camera-pixel to projector-pixel correspondences can be calculated (Exercise 3.2). Finally, this knowledge can be used to capture a different picture with the same setup (Exercise 3.3). Multiple applications can be implemented, one possible application is edge highlighting projection, which highlights the edges of the object shined by the projector. To get there, go through the following steps:

3.1 Structured Light (50 Points)

The first step consists in designing the structured light patterns which shall be projected to the wall later. Therefore, implement a tool that generates a set of so called "gray code" images. In "gray code" binary patterns, two successive values only differ in one bit. Part of such a sequence is shown in the figure below.

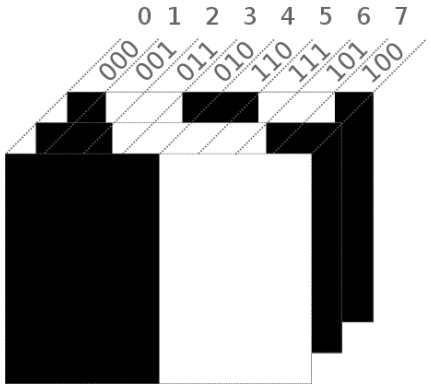

Abbildung 1: Vertical gray code pattern for column indices of projector with 8×8 resolution

To identify a precise mapping in x- and y-direction, you need to generate a set of patterns with horizontal as well as vertical alternating stripes and their inverted versions. The biggest stripes should be half the resolution white (1 == light on) and the other half black (0 == light off). For the smallest alternating stripes you can take a width of 2 pixel. The idea consists in creating a binary code for every pixel and use it to identify the camera pixel to projector pixel correspondences. Besides, you might also want to take a completely white and a completely black frame as an additional reference.
Your task is to write a function to generate the gray code patterns with 1024×768 resolution and another function to decode the patterns to row and column indices.

Hints:

- The Hamming distance of gray code is 1, that is to say that adjacent codes has only one bit difference.
- You may need to generate inverted patterns (switch 0 and 1 in all patterns) to compute the subtraction to decrease the noise for Exercise 3.2.
- Using such binary "gray code" patterns is a quite robust and very common approach. However, you can also start generating simple binary sequences first.
- Please submit your structured light encoding and decoding program.

Links:

- Gray Code
  https://en.wikipedia.org/wiki/Gray_code
- Structured Laser Scanner
  https://en.wikipedia.org/wiki/Structured-light_3D_scanner

3.2 Projector - Camera Setup (50 Points)

Now, set up a projector and a camera as shown in the following image. Project the previously generated gray code patterns image by image to a planar surface (e.g. wall) and capture each projected pattern with the camera.

Abbildung 2: Camera Projector Setup

For the setup, two CASIO XJ-A256 projectors and one Point Grey Flea3 cameras are located in room C410. The SDK of the camera is already installed in the machines. Type flycap in the console to start a GUI with which you can have a real-time view and change camera settings.
Use the knowledge about the projected binary pattern as well as the images captured with the camera to identify the pixel-to-pixel correspondences between the camera and the projector. In other words, calculate the mapping that tells which displayed point corresponds to which point in the camera.

Hints:

- Test the exposure settings in the camera GUI flycap first and use the same settings later in your own code.
- As soon as the setup is moved, the calibration from above has to be processed again.
- Link the `flycapture` library in `/usr/lib`, change the linker setting accordingly.
- A class for image capturing, and one test programm are provided in ILIAS.

3.3 Edge Highlighting Application (50 Points)

As soon as you have the projector-camera mapping there are a lot of nice things one can do.
For example, edge highlighting application. Some objects can be put within the projected area. One can first of all

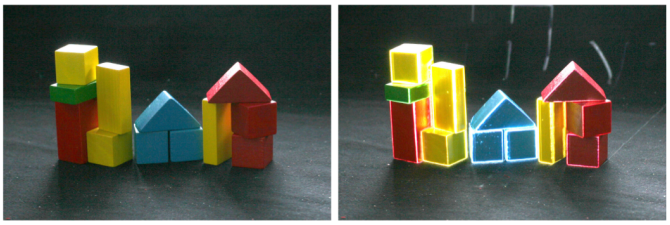

Abbildung 3: Example of edge highlighting application

compute the edge from the captured image from the camera. The pixel indices of the edges are stored. Since the pixel correspondence is measured following Exercise 3.2, one can compute the pixel indices of the projector from the stored camera pixel indices. At last one image will be generated with all the edges assigned new values by the user.
You can find more applications in the Section 4. of the following paper: "A Context-Aware Light Source" (Wang et. al) (Uploaded in the ILIAS)

Hints:

- One can use `Laplacian` to compute the edges of the image.