Upload Assignment: Assignment 2: Voxel-based 3D reconstruction

ASSIGNMENT INFORMATION

Due Date	Points Possible
Sunday, March 5, 2023 11:59 PM	**100**

This document contains a detailed description of Assignment 2, the data and the provided Python code used to visualize a voxel reconstruction. Use the code as a basis of your own program. The main goals of this assignment are (1) to calibrate four cameras (intrinsics and extrinsics), (2) to do background subtraction for each view, and (3) to implement the silhouette-based voxel reconstruction algorithm.

Download data + example code

- Download data (51MB)

*When finished, make sure to click **Submit**.*
*Optionally, click **Save as Draft** to save changes and continue working later, or click **Cancel** to quit without saving changes.*

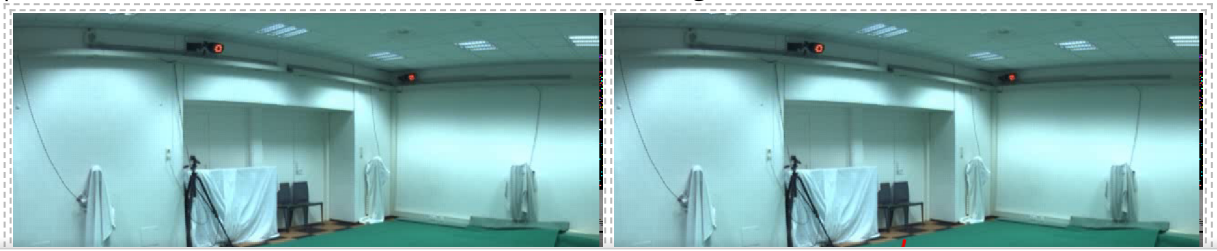
Cancel Save Draft Submit

1. **intrinsics.avi**
    - The video file contains recordings of a chessboard moving around each camera.
2. **checkerboard.avi**
    - This video file will provide your camera extrinsics.
3. **background.avi**
    - Use this video to pick a frame (or do more complex processing, see below) to create a **background** image.
4. **video.avi**
    - This is the video from which the data has to be reconstructed in 3D.

Task 1. Calibration

1. Obtain the intrinsic camera parameters with your code from Assignment 1, for every camera. Use the **intrinsics.avi** files of each view. You can (hand or automatically) pick a number of frames from each video. The number of rows/columns and the size of a tile can be found in **checkerboard.xml**.

2. We will now use the **checkerboard.avi** files to calculate the camera extrinsics. Check for a function to calculate the extrinsics provided that the intrinsics are already known.
   As **cv2.FindChessBoardCorners(...)** will not work, you should use the interface you built in Assignment 1 for getting chessboard corners. Do this precisely. Ways to automate or speed-up this process are eligible for **choice points**. To select the chessboard corners, start at the **same** chessboard corner (e.g. left-upper corner; look at the color of the corner cell) for each camera. Keep in mind that the chessboard is not square. Do the extrinsics calibration using the selected checkerboard corners for each camera. This stackoverflow answer can give you some hints about how to do it: https://stackoverflow.com/a/55284535/3602047 After completing the intrinsics and extrinsics calibration process, visualize the 3d world coordinates on the starting corner of the checkerboard for each camera:



*When finished, make sure to click **Submit**.*
*Optionally, click **Save as Draft** to save changes and continue working later, or click **Cancel** to quit without saving changes.*

3. After finishing with all cameras, write one final **config** file containing all camera properties (including extrinsics and intrinsics). The designated directory is **data/camX/** (with X the camera number). This allows you to use a previously obtained calibration. Use the **intrinsics.xml** for an example how to store the intrinsics parameters (including distortion coefficients).

Task 2. Background subtraction

1. Use the **background.avi** files to create a background model. A single frame may not give the best results. Averaging frames may help to improve the background subtraction process somewhat. How to create the best possible background image is up to you. You can also replace this with a more sophisticated model, e.g. by modeling each pixel as a Gaussian (mixture) distribution. That will give more points.
2. Once a background model is made, you will be doing background subtraction on each frame of **video.avi**. This means you have to find all foreground pixels. Convert the foreground frame image of a given camera from BGR to HSV color space, and calculate the difference with your background model. Threshold the differences for the Hue, Saturation and Value channels. See how you combine the different channels to determine if a pixel is foreground (value 255) or background (value 0).
3. Setting the thresholds automatically is a **choice task**. Two suggestions, but there are many more viable solutions:
    1. Make a manual segmentation of a frame into foreground and background (e.g., in Paint). Then implement a function that finds the optimal thresholds by comparing the algorithm's output to the manual segmentation. The XOR function might be of use here. You can look over different combinations of thresholds, and simply select the combination that gave the best results.
    2. Assume that a good segmentation has little noise (few isolated white pixels, few isolated black pixels). Implement a function that tries out values and optimizes the amount of noise. Be creative in how you approach this. Perhaps the functions *erode* and *dilate* can help.
4. Also think about post-processing, e.g. using *erosion* and *dilation*. Other techniques such as *Blob detection* or *Graph cuts* (seam finding) could work. You don't need to implement these algorithms yourself, use the OpenCV API to find the relevant functions/methods. Motivate in your report why and how you use these.

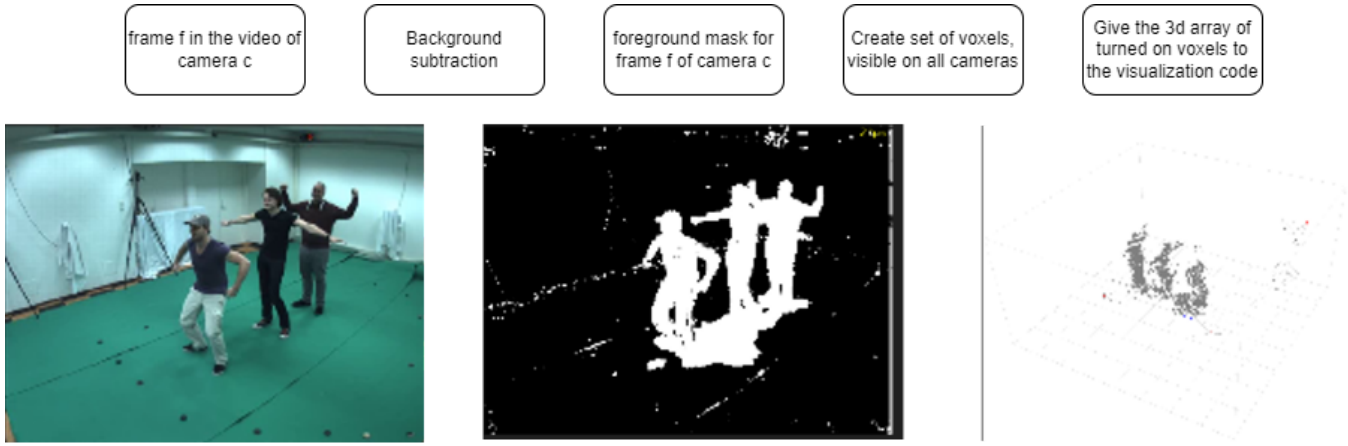
Task 3: Voxel reconstruction

1. You will need to create the lookup-table for the valid voxel back-projections onto each of the camera's FoVs.

*When finished, make sure to click **Submit**.*
*Optionally, click **Save as Draft** to save changes and continue working later, or click **Cancel** to quit without saving changes.*

2. Now develop the voxel reconstruction algorithm such that the voxels that correspond to the person in the scene are "on", based on the four foreground images. The result of your work should be a 3D cubic half-space containing the voxel reconstruction of the provided input video (**video.avi**). Use the provided functions. Noise and voxels corresponding to shadow reduce the quality of your reconstruction. Think about ways to remove such voxels. Similarly, if there are (small) holes in your reconstruction, think of a way to fill those, without making unrealistic assumptions. A **choice task** is to "color" the voxels according to what you see in the video. Another **choice task** is to convert the voxel model into a surface mesh.

The basic drawing pipeline



Codebase: Install the relevant libraries following the installation instructions on github: https://github.com/dmetehan/Computer-Vision-3D-Reconstruction.git. When you run the executable.py you should see an empty scene with floor grid and 4 dummy cameras. You can press 'G' to generate random voxels and use 'W', 'A', 'S', 'D', and your mouse to traverse the 3d space. Overall you don't need to worry about the details of the visualization code. All the functions you need to edit are in assignment.py. You're free to create as many files/classes as necessary. The position and orientation of the cameras in world coordinates should be estimated using the cameras' extrinsics matrices. The 3D voxel space is a half-cube, with side lengths 128 and height 64. y is the up direction. You can set the voxel locations using list of lists. [[x1, y2, z1], [x2, y2, z2], ..., [xn, yn, zn]].

Report: Your report should be around 2 pages (you can use this Word template) and contain:

1. An explanation of your methods

*When finished, make sure to click **Submit**.*
*Optionally, click **Save as Draft** to save changes and continue working later, or click **Cancel** to quit without saving changes.*

5. A **link to a video** (Youtube, Vimeo, Wetransfer, etc.) clearly showing the 3D reconstruction of the input videos. Make sure the link is accessible.

Submission: Through Blackboard, hand in your **report** (2 pages). Include your **source code** (do not include the resources, .jpg, .png, etc.) and the calibration files of each camera (**no videos!**).

Grading: The maximum number of points for this assignment is 100. The assignment counts for 10% of the total grade. You can get up to 70 points for these tasks:

- Quality of calibration (and calculated/motivated properly): 20
- Ingenuity of background subtraction method: 15. More points are given for more robust techniques.
- Quality of background subtraction: 10
- Quality of voxel model (and sophistication of tricks): 15
- Report (motivate your choices, **indicate choice tasks**): 10

In addition, you can earn a maximum of 30 choice points by:

- CHOICE 1: Automating the detection of the four corner points of the chessboard: 10
- CHOICE 2: Automating the way the thresholds are determined: 10
- CHOICE 3: Coloring the voxel model: 10. Importantly, take into account whether a voxel is visible and not occluded by a camera.
- CHOICE 4: Implementing the surface mesh: 10. Convert the voxels to a surface mesh with (this is the exception to the rule of no alien code) an implementation of Marching Cubes (or another implementation).
- CHOICE 5: Optimizing the construction of the voxel model: 10
- CHOICE 6: Speeding-up the creation of the look-up table, background subtraction and/or voxel reconstruction: 5. Parallelization is eligible.
- Other tricks to either speed up the process or improve the results: check with Ronald.

Doing 'bad stuff' in any choice part will not lead to point reduction of the 70 points, just in fewer than 30 bonus points. In short, you will not be punished for trying out cool stuff!

Example result videos from previous years

- https://www.youtube.com/watch?v=BHoOkjGBHPI

*When finished, make sure to click **Submit**.*
*Optionally, click **Save as Draft** to save changes and continue working later, or click **Cancel** to quit without saving changes.*

Q: How can we get the same origin of the world coordinates from all views?

A: To have the same origin of the world coordinates from multiple cameras, one of the solutions is to do the calibration semi-manually by clicking the checkerboard's corners. However, you can also draw the grid lines using an OpenCV function, and ensure that the starting position of the lines is the same for all views. If you choose to do this, you have to generate your own config.xml-file for each of the cameras, which must contain both the camera intrinsics and extrinsics.

Q: We have tried to remove shadows from the silhouettes, but we cannot remove them entirely. What should we do?

A: In general, shadow detection and removal are still an open problem in computer vision. Thus, some noise due to shadows will be tolerated.

Q: Why are the origins of our cameras not correctly located in the 3D space?

A: The provided 3D reconstruction code requires the information of the size of the squares (the black or white squares) of the checkerboard in millimeters. You set the value in checkerboard.xml.

Q: Are we allowed to use an existing code available in the internet to do the background subtraction?

A: No, use OpenCV (except for the surface mesh choice task)

ASSIGNMENT SUBMISSION

Text Submission

Write Submission

*When finished, make sure to click **Submit**.*
*Optionally, click **Save as Draft** to save changes and continue working later, or click **Cancel** to quit without saving changes.*

Group Name
Assignment pair 43

Last Activity Recorded
Paula Castro Ramirez (Mar 5, 2023 12:42:23 PM)

ADD COMMENTS

Comments
For the toolbar, press ALT+F10 (PC) or ALT+FN+F10 (Mac).

P 0 WORDS POWERED BY TINY

*When finished, make sure to click **Submit**.*
*Optionally, click **Save as Draft** to save changes and continue working later, or click **Cancel** to quit without saving changes.*

*When finished, make sure to click **Submit**.*
*Optionally, click **Save as Draft** to save changes and continue working later, or click **Cancel** to quit without saving changes.*