INFOMCV Assignment 2

Marios Iacovou (1168533), Christos Papageorgiou (9114343)

Summary

For background subtraction we trained 3 different models and experimented with thresholding and contour detection, as well as with morphological operations, to improve the accuracy of the subtraction.

For our voxel model we construct a voxel space of 128x128x128 and get the resulting lookup table to check the voxels that are in view of the 4 cameras and visualize them. This is done using masks from each frame.

Extrinsic parameters

To calibrate the camera intrinsics, we extracted 1 frame per second of every intrinsics (50fps video). We kept only the frames that had their chessboard points automatically detected to improve calibration. After the calibration was done, we ran our discarding algorithm from assignment 1 to remove any images that caused the calibration to worsen. The results can be found in our plots folder for which images were discarded and how the error improved. Camera 1 had a reprojection error of 0.53 after discarding 3/38 frames, camera 2 had a reprojection error of 0.21 after discarding 1/51 frames, camera 3 had a reprojection error of 0.20 and didn't need any discarding, and camera 4 had a reprojection error of 0.39 after discarding 1/64 frames.

After intrinsics calibration was done, we extracted test frames from the videos with the chessboard to calculate the extrinsic. As automatic corner detection fails on these, we had to run manual selection. As a choice task we automated this task and details can be found later in the report.

Camera 1

$$M = \begin{bmatrix} 491.98 & 0 & 334.1 \\ 0 & 494 & 228.9 \\ 0 & 0 & 1 \end{bmatrix} \quad R = [-1.33 \quad 0.55 \quad 0.63] \quad T = [239.85 \quad 731.16 \quad 4745.83]$$

Camera 2

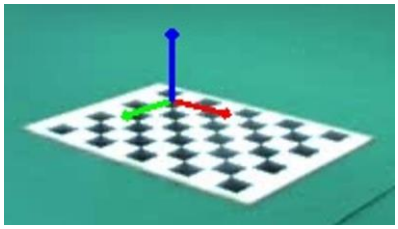
$$M = \begin{bmatrix} 488.25 & 0 & 329.86 \\ 0 & 490.51 & 230.8 \\ 0 & 0 & 1 \end{bmatrix} \quad R = [-1.59 \quad 0.03 \quad 0.03] \quad T = [-247.24 \quad 1475.86 \quad 3672.13]$$

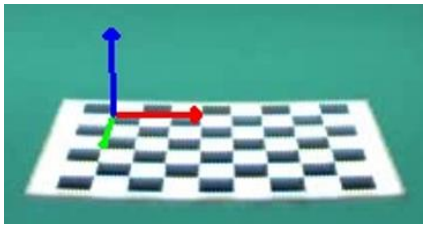
Camera 3

$$M = \begin{bmatrix} 489.94 & 0 & 322.32 \\ 0 & 488.55 & 232.9 \\ 0 & 0 & 1 \end{bmatrix} \quad R = [-1.07 \quad -1.34 \quad -1.42] \quad T = [-725.01 \quad 1266.09 \quad 2502.55]$$

Camera 4

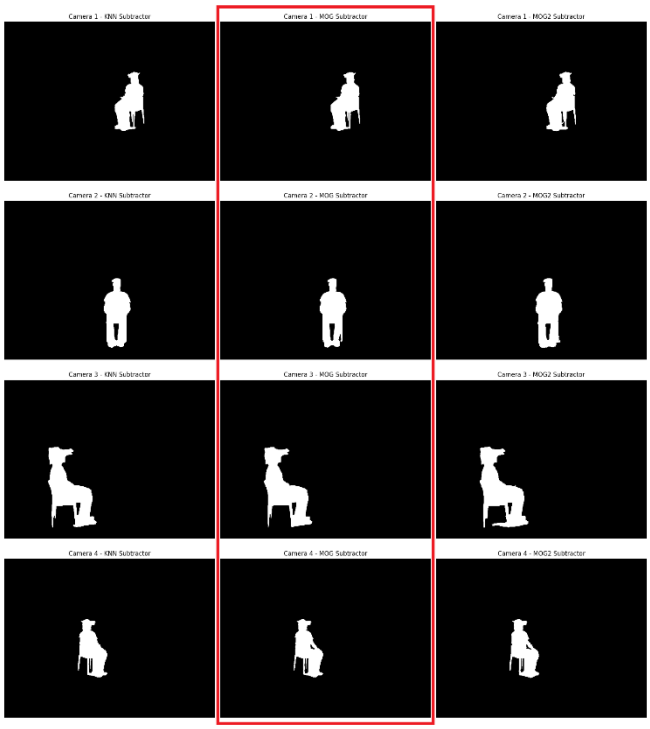
$$M = \begin{bmatrix} 490.21 & 0 & 345.01 \\ 0 & 492.41 & 242.45 \\ 0 & 0 & 1 \end{bmatrix} \quad R = [-1.35 \quad -0.65 \quad -0.71] \quad T = [-983.81 \quad 967.51 \quad 4273.79]$$





Background subtraction

For the background subtraction process we employed three distinct methods: K-Nearest Neighbors (KNN), Mixture of Gaussians (MOG), and MOG2 (MOG Version 2), in order to test their efficiency. The KNN approach classifies pixels based on their distance from the nearest samples. The MOG method models each pixel as a blend of Gaussian distributions, which identifies the background under gradual lighting changes. MOG2, an advancement of MOG and has improved shadow detection.



Each model was trained through multiple iterations to identify the optimal parameters for its implementation on our training video. After the model gave us an initial foreground mask, we used some additional techniques to improve the thresholding. We wanted to keep the legs of the chair and contours where the arm gap is on the figure visible. As such we used an external threshold to find the figure contour (by finding figures with threshold value) and then an internal threshold to black out the inner contours (by finding inner contours with the threshold value). These thresholds were identified through multiple iterations for each camera. We also have filters to run morphological operations before and after the extraction of contours to improve it. Opening morphological operations perform an erosion and then a dilation to remove noise and closing morphological operations perform a dilation and then an erosion to close black holes. These were especially useful for image 3 that to close gaps on the arm of the figure. Additionally, for image 4 it was useful to reduce the operations to keep one of the chair legs in the foreground and not remove it.

Our parameters are as follows:
cam1_bg_model_params = [5000, 115, False, False, True, True]
cam2_bg_model_params = [5000, 115, False, False, True, True]
cam3_bg_model_params = [5000, 175, False, True, True, True]
cam4_bg_model_params = [5000, 115, False, False, False, True]

In order: outer threshold, inner threshold, perform opening before contour detection, perform closing before contour detection, perform opening after contour detection, perform closing after contour detection.

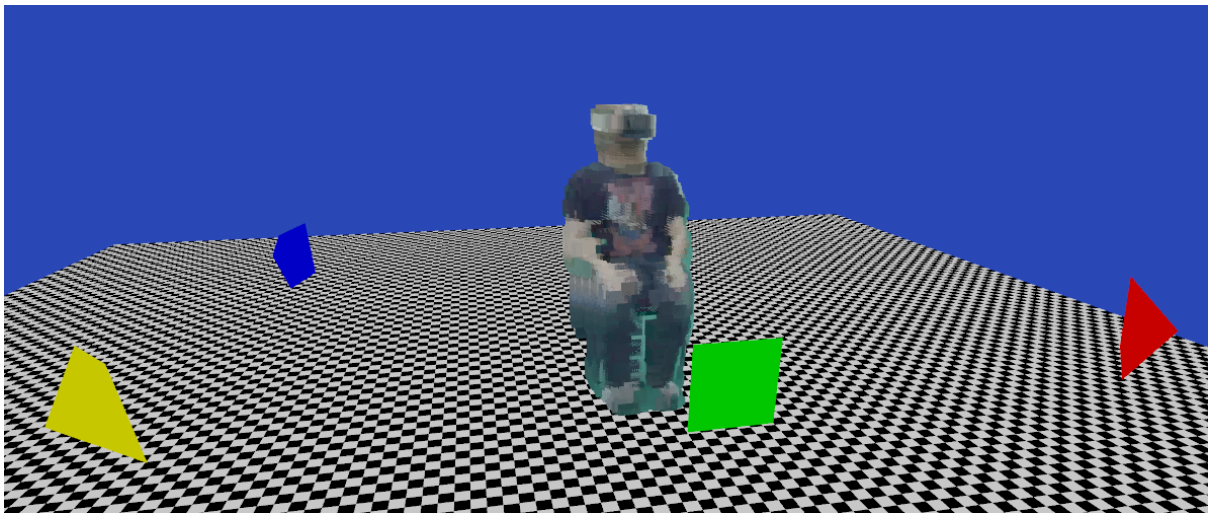
After evaluating the outcomes of the three methods across different camera angles, we concluded that the MOG model outperformed the others in key aspects we paid attention to. The details on the chair legs are preserved and shadows are minimized. We also see a clear outline of the figure all around but also inside with the black contour areas. The comparison image is shown above but is also in our plots folder.

3D Voxel reconstruction

For voxel reconstruction we first created a lookup table. We used a voxel volume of 128x128x128 to give us enough detail on the figure, while also keeping memory and simulation time in check. The ranges of the points for the X, Y, Z ranges of the voxels were tested iteratively to find spaces that correspond to good projections with many image points. We save the image points corresponding to the projections of the 3D voxel points to the 2D image plane for every camera and every voxel.

After the lookup table is created, we perform a search to find voxels that are viewed by 1 or more cameras and save them. We also extract colors to use them for a choice task explained later. To do this we use the foreground masks extracted by a MOG background subtractor from the previous task. If an image pixel is on for a view and it's not out of bounds, then we mark the voxel view.

After the voxels are extracted, we threshold the voxels such that a turned-on voxel is a voxel in view of all 4 cameras. We perform a final scaling for visualization purposes and change of axes systems to support OpenGL's methods. To understand the visualization, we plotted the cameras in the appropriate positions and angles using extracted camera parameters. Every time the function to visualize the voxels is called, the video moves to the next frame.

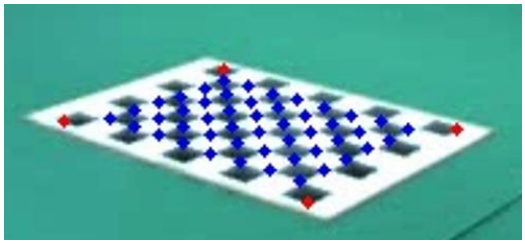


Choice tasks

CHOICE 1: Automating the detection of the four corner points of the chessboard:

To do this task we created a KNN background subtraction model to remove the background from the image with the chessboard. With the remaining paper with chessboard pieces on it, we perform a

series of operations. We perform morphological operations to remove noise from the model's mask output and extract the white pieces region after improving contrast. This white region is used to find black contours inside white regions (black squares). After the contours are extracted, we construct a hull around the region extract the black squares when comparing with the inverse of the initial mask. Finally, for the extracted squares we find the 4 corners on the outer edges of the chessboard. Our algorithm for interpolating points was improve to account for the cases where we have only the outer 4 corners instead of the inner ones. Every time a manual detection of corners is required, the user is shown the results of the automatic detection with our method and asked to sort them. Then they are shown the results of the interpolation and can discard or keep them. If they discard them then manual selection is performed as in assignment 1. The accuracy of the approximation of the corners with this method is very high and we used it for our extrinsic calibration.

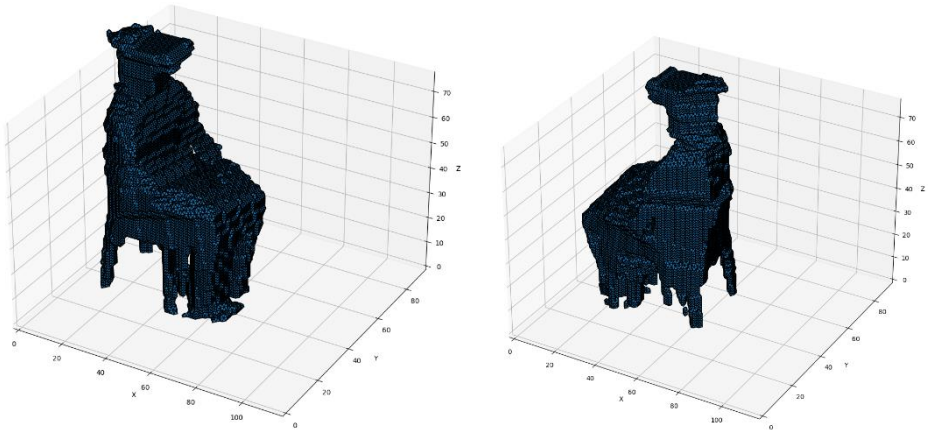


CHOICE 3: Coloring the voxel model:

For every voxel that is turned on (viewed by 4 cameras), we used the coloring from the appropriate frame to give it color. The frame used for coloring is the view of camera 2 which faces the model from the front and gives the most accurate coloring. The results are shown above in a previous section.

CHOICE 4: Implementing the surface mesh:

We used an implementation of the marching cubes algorithm and Poly3Dcollection to plot the surface mesh. We also rotate the model around to view on both sides.



Link to video

https://www.youtube.com/watch?si=lNUU8YzbzM8unze8&v=epp5ns2wpv0&feature=youtu.be