INFOMCV Assignment 3

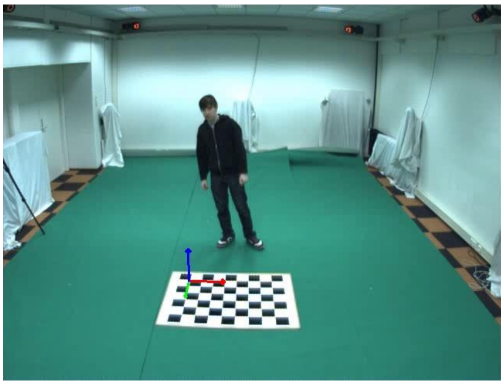
Marios Iacovou (1168533), Christos Papageorgiou (9114343) (Group 74)

Summary

We provide a summary of every part of the assignment.

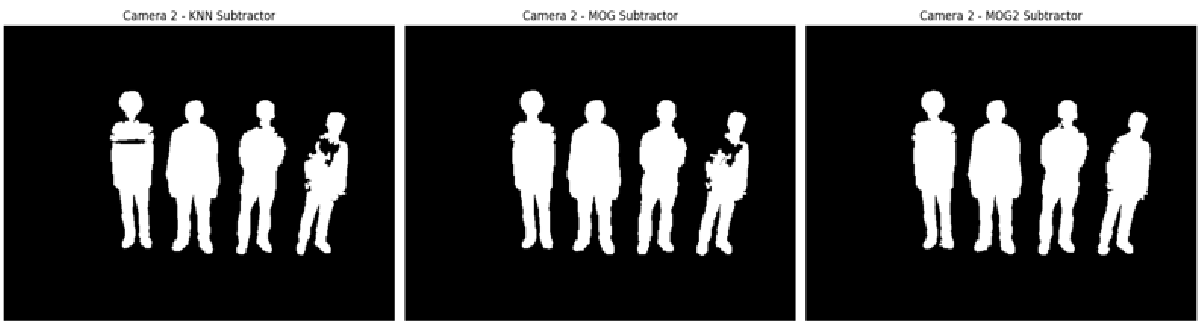
Camera calibration

We kept the intrinsic parameters from the previous assignment but calibrated the extrinsic parameters with the new chessboard frames. For this phase we updated our automatic corner detection from our choice task on the previous assignment to remove the silhouette of the person standing next to the chessboard. Thus, we were able to extract the extrinsic parameters automatically with high accuracy.



Background-subtraction method

In our previous assignment we had the most success in foreground extraction using a Mixture of Gaussians (MOG) approach. For this assignment, we switched to MOG2 as it provides better masks in varying conditions and illumination changes. The thresholds for the model have also changed to be able to capture more foreground figures inside the frame.



Voxel reconstruction

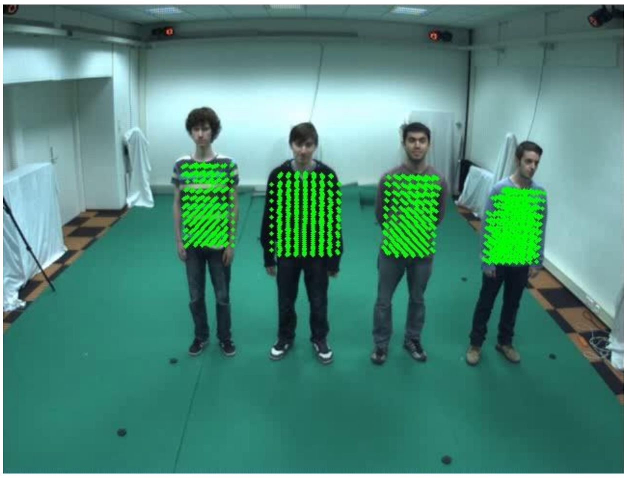
The voxel space used is kept at the provided 128x64x128 width, height, and depth sizes but the voxel sizing has been changed from 30 to 60 to fit more figures into the provided grid. This voxel space captures the figures adequately when projected to image points. Voxels that are in the foreground masks of all 4 cameras are considered visible.

Voxel clustering

Expanding the functionalities of the previous assignment, we now cluster voxels by taking into account only their width and depth and ignoring the height (blob on the ground) using K-means clustering. We cluster the voxels into 4 clusters which is the number of the figures in the provided video. For this step we also consider potential outliers which are tackled using distances from centroids. The process for that is explained in our choice tasks.

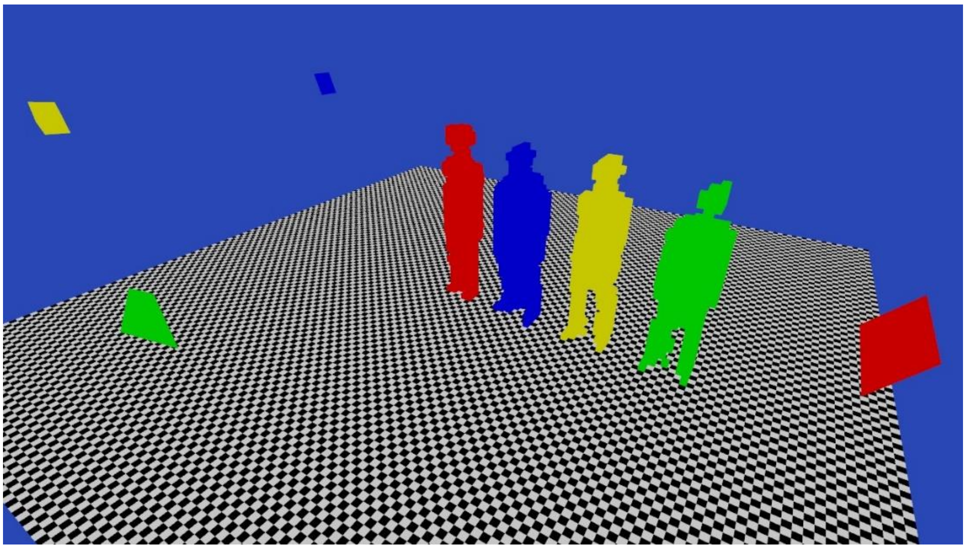
Color models

During the offline phase (first frame of the video), we create color models by projecting the 3D cluster points of each figure to the 2D image plane of a camera. From the image points corresponding to each cluster, we crop the legs, head, and arms to focus our color models on colors from shirts and try not to let similar colors from the jeans, skin, or hair influence the values. This is done using a 20% crop from the top to remove the neck and head, a 45% crop to remove the legs while keeping the shirts hanging over the hips, and a 20% crop from the left and right to remove the arms. The crop for the arms was subsequently removed as it was unnecessary, with our outlier removal process during the clustering phase already taking away some of the arm area to improve the clusters. For each color model we compute a 180x256x256 HSV histogram, where its each dimension conveniently fits inside a byte (180 bins for Hue as it’s half of the color wheel). Using HSV histograms means we can focus on perceptual color properties instead of individual color channels like RGB histograms. Each histogram is normalized between 0 and 1. We create color models for multiple cameras, which is explained further in our choice tasks. Our offline color models are built using the first frame of each video, where occlusion is none or minimal and colors are extracted well.



Online matching of color models

In order to match online color models to their corresponding offline ones, we compute a distance matrix of every online model’s similarity to each offline model. In order to calculate the distance of each position, we compare the relative histograms for Hue, Saturation, and Value and aggregate their distances. Our method for comparison is Bhattacharyya distance in order for similarity to be close to 0 while difference takes higher values. Multiple cameras are taken into account and aggregated further in this step which is explained later in the choice tasks. After filling each position of the distance matrix, we can proceed to use the Hungarian algorithm to compute a matching between every online color model and an offline color model. The similarity from the distance method we used being close to 0 is appropriate for this algorithm as its goal is minimizing costs, which in our case are distances. After we have the matching, we can label each cluster with the colors of the original offline clusters it matched to. Using this matching we can build trajectories using cluster centers and track each figure as the video progresses.

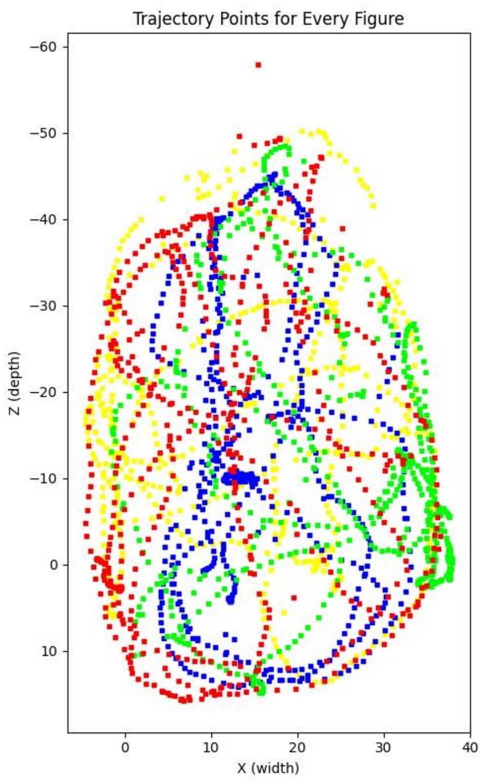


Link to video

A demonstration of our cluster labeling throughout the whole video with an interval of 10 frames to progress through scenes where minimal changes are observed can be found here: https://youtu.be/6g3rXIq_Rzk

Trajectory image

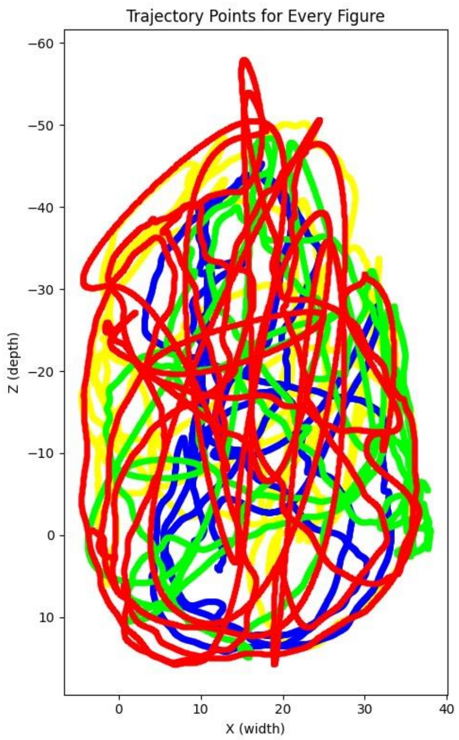
The following trajectory image shows the trajectories of every figure during the video. The figures standing in a straight line in the first frame are labeled from left to right as red, blue, yellow, and green. These trajectories are smoothed for a choice task, which is shown later.



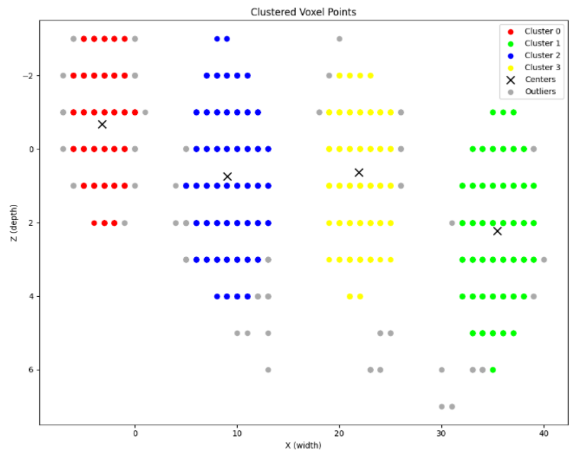
Choice tasks

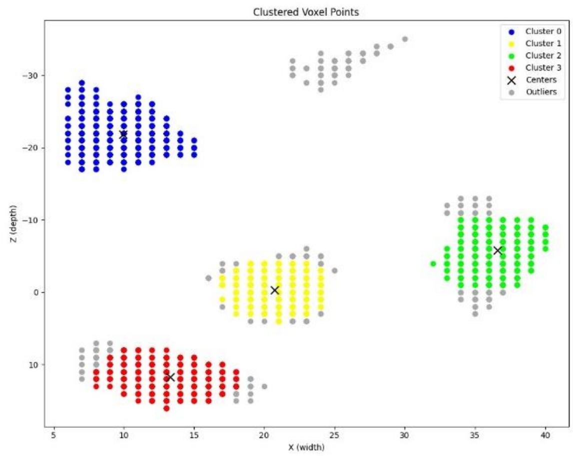
CHOICE 1: Use multiple cameras to increase the robustness:
When creating the color models, we use multiple cameras in order to take into account information from different views. Our offline models are built at the start of the video and use views from cameras 1, 2, and 4 for the histograms, as camera 3 shows occlusion on the first frame and overall gave bad performance during matchings when used, even with a different frame for its offline color models. During the matching step, instead of aggregating distances of histogram channels of a single camera view when comparing an online model to an offline one, we aggregate the distances of those histogram channels across the 3 views. This helped us overcome cases where the similarities of an online model were close to multiple offline models, which gave unreliable matchings. During these cases the additional cameras decrease or increase the similarity successfully and the overall correct matchings increase throughout the video.

CHOICE 4: Smooth trajectories:
In order to fix the gaps in our original trajectory image, we interpolate points using a set number of new samples. The new samples are set to 5 times the number of frames divided by the frame interval. This successfully makes the trajectories continuous and clearer when viewed.



CHOICE 5: Get rid of outliers and ghost voxels:
When clustering the voxels, there are times where more than 4 clusters are visible in the data. These outlier points and ghost voxels get mixed with points that we want to keep in our clusters and reduce the quality of the clustering. In order to deal with this problem, after running an initial clustering with K-means, we calculate the distance of each point belonging to a cluster. These distances are used to calculate the mean and the standard deviation. We then set a threshold of 1.5 standard deviations away from the mean for points to be considered outliers. These points are then removed, and the rest of the points are then re-clustered to obtain a better clustering than the initial one. Below are some examples of plotted cluster points where we can see outlier points and ghost voxels having been removed.





Bonus: Improve functionality of interface during execution:
In order to visualize our results more easily and provide additional visualizations in the interface that renders the voxels, we decided to implement the functionality to switch between different renderings with buttons 1-5:
1: Show voxels with outlier voxels, use generic coloring
2: Show voxels with outlier voxels, use coloring from camera 2
3: Show voxels without outlier voxels, use generic coloring
4: Show voxels without outlier voxels, use coloring from camera 2
5: Show voxels without outlier voxels, use coloring depending on color model labels
Additionally, pressing G renders the next frame as before, but we included the choice to press L to loop through renderings of frames for visualization. We also provided the functionality to take screenshots during the execution and compile them into videos, which is how we created the video that showcased our results in a previous section.