Anna Dollbo 1843672
Flavius Marc 0862037

INFOMVC 2022
Group 25

Report Assignment 3: Color-based voxel labeling

1. Calibration and foreground processing

The camera intrinsic parameters stayed the same as in Assignment 2, since the same cameras were used to record the videos in that assignment. The extrinsic parameters were obtained by calling the `Camera::detExtrinsics()` method and thereafter manually selecting the corners of each checkerboard, always starting at the same upper left corner for each camera.

Compared to the 2nd Assignment, for the background subtraction per frame, we had to change the radius of the structural element to 2 for the opening operation, in order to eliminate some more outliers, as a preprocessing step.

2. Clustering

The K-means algorithm in the OpenCV framework was used for clustering, as it is easy to implement and guarantees convergence. For each frame, the ground coordinates for each visible voxel was given as input to the algorithm. The k-value was set to four, as we were looking for center points of the four people present in the video. The algorithm was set to terminate after 100 runs or when the epsilon value of 1.0 was reached. We also used the flag `KMEANS_PP_CENTERS` which uses the center initialization by Arthur and Vassilvitskii[1]. These measures were taken to avoid getting stuck in a local minima, as the algorithm always outputs the best clustering found according to the criteria. The algorithm output of labels for each voxel as well as the four center point coordinates were saved to be used later in the program.

3. Color Model

For the construction of the color model, the data was first observed in order to find a good frame to use for building the model. Frame 533 for camera three and four was chosen as all four people were well separated in the frame and the two cameras provided different lighting on the people as well as captured them at different angles. The ground coordinates for the people in the frame were then fed to the K-means algorithm, and for each cluster samples in the form of rgb-values for each pixel was taken. For the color model, we chose to build a Gaussian Mixture Model (GMM) for each cluster, as they use unsupervised learning to find the best mixtures for the data during training and can then be used as a classification algorithm.

The models were built using OpenCV's ml::`EM()` model, which is an expectation maximization algorithm that returns the maximum likelihood of a sample belonging to the model. The number of mixtures were set to three, as the number of color components of these data were estimated to three, representing the hair color, the skin color, and the color of the shirt the person was wearing. However, after some testing, the number of mixtures were changed to two, which appeared to give better predictions and less flickering in the final 3D model. A choice was made to only take the colors of the upper body as all people in the video were wearing roughly the same color of trousers, thus not providing any distinguishing information about that person.

---

[1] Arthur, D., & Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. *SODA '07*.

Each of the four models were trained on pixels corresponding to one of the four people in the video, meaning that each model modeled the color mixture of one person. After training, each model was saved in an xml-file which was then loaded in the online phase to provide the predictions.

4. Color matching process

During the online phase, for each frame, each visible voxel's ground coordinate is first fed to the K-means algorithm to obtain the cluster labels for each voxel. The pixels corresponding to each cluster are then taken from two cameras (camera three and four) and the pixels for each cluster are put in separate matrices. The four GMM models are then loaded and each cluster matrix is fed to the models using the `predict2()` function, which returns the log likelihood of the samples in the cluster matrix belonging to that model (person). The log likelihood of each cluster for each model is then converted to the absolute value (as the output is always negative) and is put in a cost-matrix where each row corresponds to one cluster and each column corresponds to the output of one model. This matrix is then fed to a Hungarian algorithm, which is an optimisation algorithm that seeks to assign one label to each sample based on the calculated cost of each sample for each job. This means that the algorithm looks at the absolute log likelihood of each cluster belonging to each model and decides based on these which cluster best fits the model based on the shortest path. The algorithm was implemented using third party code.[2] The labels of each cluster is then assigned globally which determines the color of the cluster in the 3D projection.

5. Occlusion handling

To deal with occlusions, each voxel in the voxel space was first processed in an offline phase to see which voxels projected to the same point and region of interest, which we set to be one pixel radius around the projection point of that voxel. Voxels' distance to the camera that are projected to the same region of interest was calculated and the voxels that had a shorter distance were stored in a vector in the voxel structure of the corresponding voxel being occluded by those voxels. During testing it was noticed that the offline phase took a significant amount of time, whereupon the step size was changed from 32 to 64, which meant that the voxels were more sparse in the voxel grid, but with faster processing time for the offline phase and with little loss in the distinction of the objects in the 3D scene.

During the online phase, for each voxel its list of possibly occluding voxels was processed to see whether any of them were within the visible voxels for the frame. If they were within the visible voxels the pixel for the voxel being processed was not considered in the classification, as it was being occluded by another voxel. This led to only the foremost voxels that projected to the pixel position being considered as a sample to the GMM models.

6. Choice tasks

6.1. Using multiple cameras to increase the robustness

Two cameras, camera three and four, were used in the creation of the color models and pixels from the view of these cameras were also drawn at each frame. This was done to ensure robustness of the model, as having more information could aid classification of which cluster belongs to each person. This is especially true in the cases where one person is almost completely occluded in one view, where information about their colors could then still be collected from the other view.

[2] Repository for Hungarian algorithm: https://github.com/mcximing/hungarian-algorithm-cpp

6.2. Finding a way to deal with people outside the voxel space

To deal with people disappearing outside of the voxel space, the voxel space was increased and the bounding box was moved to also be able to include the people in the space. Changes to xL,xR,yR,yL and m_height were made to accomplish this.

6.3. Color Trajectory

The position of the cluster centers were plotted on the 3D scene throughout the video. Each trajectory was colored based on the label of the person in the frame, this means that the trajectories also have different colors due to incorrect labeling, where correct labeling would lead to each person having one color throughout the whole video, and the trajectory also having the same color. This was done by storing the centers in a vector and creating a separate function in Glut.cpp called `drawTracking()` that draws points at the corresponding centers at each frame.

7. Results & Discussion

Despite what we believe to be a correct implementation conceptually the color matching still does not work correctly, which can be seen from the fact that the colors of the four people as they move in the 3D space are not stable. A possible obstructing factor could be that we did not have time to deal with ghost voxels at this time, leading to some ghost voxel-clusters sometimes being mistaken for a person-cluster by the K-means algorithm in the situations when two people are very close to each other in space. Another possible obstructing factor could be that the color models are not optimal. While looking at the performance live, it can be observed that the model correctly classifies the person with the black sweater most of the time, as can be seen in the image of the people's walking trajectories (blue line). This could indicate that the other people are too similar in colors to be easily distinguishable from each other. Further investigation of different parameters for the GMM models is therefore needed to find the best models. Other factors that lead to the disappointing performance could be logic-problems in the code or other code-related problems. In conclusion we are displeased that we did not manage to make the color matching work considering the effort we put in, but have learnt a lot in the process and would like to hear feedback on where we could possibly have gone wrong. [3]

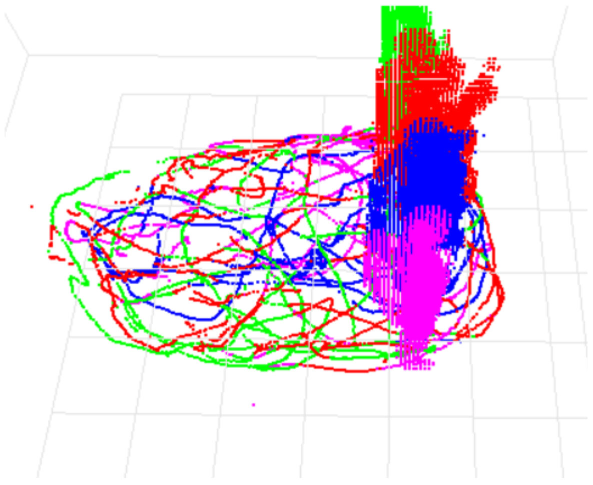
Figure 1. Trajectory paths shown in the last frame

Link to video:
https://www.youtube.com/watch?v=Xeew6q2O4c4

Note: the first 47 seconds in the video represents the model without dealing with occlusions, the other part is with the code that deals with occlusions.

[3] **Note:** information about the code structure can be found in Readme.txt