Computer Version Assignment 3

Yunming Hui(5334977)
Graduate School of Natural Sciences
Utrecht University

Yuqi Liu(5383986)
Graduate School of Natural Sciences
Utrecht University

The assignment is based on the Windows operating system, OpenCV 4.2.0 and Visual Studio 2022, with the sample code framework provided as the most basic.

I. Basic tasks

A. Intrinsics and extrinsics calibration

The way that we calibrate intrinsics and extrinsics is exactly the same as we did in Assignment2.

B. Voxel clustering

Voxel clustering is implemented using the K-Means algorithm, specifically using the "kmeans" function of "OpenCV". Since the person is standing, the difference between voxels belonging to the same person after voxel reconstruction is not significant on the x and y axes, only on the z axis.
So the basic idea of clustering is to project all voxels onto the ground plane, ignoring the height information of the voxels, to form a two-dimensional point set. The point set is then clustered, which has been achieved by grouping voxels belonging to the same person together to form a cluster, while voxels belonging to different people are separated in different clusters. The code for this is in the "cluster" method of the "Reconstructor.cpp" file.
In the concrete implementation, we have considered the problem if the person disappear (the actual number of clusters is not 4 but smaller). The exact implementation is specified in section 2.
In addition, to avoid clustering into local optima, we can use the "attempts" parameter of the "kmeans" method. This parameter indicates the number of iterations. Initially, the clusters' center are generated using a random number method and the label indicating the cluster to which each point belongs is also randomly generated. Each iteration is therefore independent of each other, so increasing the number of iterations avoids getting stuck in a local optimum.

C. Color model building

We use the color histogram to construct color models of each person in four different cameras. Next, the process of obtaining a color model of a person in a camera will be briefly explained as an example of the process.

Before generating a colour histogram, we identify the main colours used for identification. To solve the problem of occlusion, we have to try to choose frames where the characters are separated for four cameras. We cluster the foreground parts of frames by HSV respectively, each picture has 10 clusters. In this way we find the ten main colours of each of the four camera images under HSV.

After identifying the main colours, the image of a camera at that moment and the set of pixels visible after the voxel projection of a person to that camera at that moment are used as input. Since the color of the character's pants is very similar, the color of the pants part is not meaningful for the subsequent identification, so we restrict the z-coordinate values of the voxels when projecting them. We only project voxels with z-coordinate values greater than 300. For each visible pixel, we determine which cluster center it is closer to and turn its value into the value of the center of the nearest cluster. We then put these visible pixels into a new Mat (e.g. with n visible pixels, we get a Mat of size 1*n). We have a histogram of the colours of this new Mat.The normalized color histogram is the color histogram of the person in that camera.

In this assignment, four cameras are used and there are four people in the video, so we get a total of 16 people color models (color histograms). All 16 color models are stored as .xml files for later use by identification.

The code for generate offline color histogram this is in the "getInitHis" method of the "Reconstructor.cpp" file. And the code for generate online color histogram this is in the "getHistogram" method of the same file.

D. Identification

We have implemented two methods of identification, based on the position of the person and based on the colour features of the person. In addition to this, we analyse the differences between the two implementations, as detailed in section 3. The position-based approach is described in detail in Section 2, and below we describe the specific implementation of the colour feature-based approach.

In the previous subsection, we described how to get the color model of the character. We first just used four character models from one camera for identification. These four models were used as baseline models. During the online process, we will get the color histogram of the character in each frame. We use the compareHist function provided by OpenCV to compare the color model obtained in each frame with each of the four reference color models. Calling the compareHist function gives a value indicating how similar the two color models are, the higher the value the more similar the color models are. The character will be considered as the person corresponding to the model with the highest index.

But using only one camera creates a problem: when a person is obscured by another person in the frame, the obscured person will be marked as the person who obscured him. To solve this problem, we use four cameras for identification, which will be described in the next section.

E. Implementation of visualisation of colouring results and path tracing

After identification, we represent the same person in different frames with the same colour, so that when the voxel reconstruction results are displayed, one colour corresponds to a specific person. The results are shown in the third section with pictures and there is a link to the video presentation in the "Supplement material".
As a result of the clustering we can obtain the center coordinates of each cluster, which represent the 2-dimensional (ground plane) location information of each person in each frame. After identification, we save the same person's position information in different frames to form the tracking path of this person. and plot it on the ground plane. The exact code is in the "drawWalkingTracking" method in the "Glut.cpp" file.
As each person's location information is stored separately, his track stops when he disappears from the screen and updates again when he is redisplayed. Specifically, when a person disappears, first the number of clusters of the clustering algorithm is reduced to three, then the three people are matched by identification to three of the original four people, and then the location information of these three people is updated. However, the problem of disappearing people can be solved after implementing a comprehensive identification using multiple cameras (occlusion problem solved) and an appropriate increase in the size of the display area (adjusting the m_height value).

II. Choice tasks

A. Identification using multiple cameras

In order to use multiple cameras for identification, we first need to correspond the baseline character models in the four cameras. For example, we want to know which baseline baseline model in camera 1 corresponds to the baseline color model of person 1 in camera 0. We still use the compareHist function for correspondence, and the two most similar models in the two cameras will be considered as if they correspond to the same person.

Then, we will get the annotation of the character model in each frame from each of the four cameras. When four cameras have different annotations for a character model. We will use majority voting to identify the person corresponding to that model. For example, when more than two of the four cameras believe that the color model corresponds to Person 1, then we will assume that the color model corresponds to Person 1.

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

As the K-Means clustering algorithm requires a prior input of the value of "K" (the number of clusters). Therefore, when a person disappears from the display area, we only need to find the correct value of "K". In practice, if only three people are actually in the display area, the sum of the distances (compactness values) of all points from the centre for "K=4" is significantly greater than for "K=3", and the corresponding " K=2" is also greater than "K=3". This value is the return value of the "kmeans" method of "OpenCV". So we start with "K=4" and if a smaller K value results in a smaller "compactness" value, we reduce the value of "K" until we find the correct K value.
The code for this is in the "cluster" method of the "Reconstructor.cpp" file.

C. Implementing tracking based on previous position and movement

First we analyse the feasibility of location-based identification. Since people are made up of collision volumes (two people cannot overlap in 3 dimensions) and the interval time between each frame is very short. So of all the distances between a person's point in a given frame and all the points in the previous frame, the distance to this person's point in the previous frame should be the smallest. So the idea works.
In the implementation, the "kmeans" method is used to obtain information about the position of everyone in a frame. When we read a frame, we first get all the positions of the previous frame, and then we get all the positions of the current frame by using the "kmeans" method. Identification is achieved by finding the minimum distance. The code for this is in the "labeling_bylocation" method of the "Reconstructor.cpp" file.
Location-based identification is very dependent on the accuracy of voxel reconstruction. If the voxel reconstruction is inaccurate and there are many noisy voxels (especially when two people are close together, there are significantly more noisy voxels around them), then the clustered location information does not accurately represent the actual location of the person, and the identification error occurs. And since the method's identification result in one frame is based on the result of the previous frame, once the identification error occurs, the error will persist.

D. Updating color model over time

During the online process, we will update the color model in real time. The update strategy is that after a majority vote, we will sequentially check if the camera has classified the character correctly. For each camera, for the baseline color model corresponding to the correctly classified character, we will again check the similarity of this baseline color model to the color model of that character in the current frame (using the compareHist function), and if the similarity is greater than 0.95, we will update this baseline model.

III. Result

A. Intrinsics and extrinsics calibration

Figure 1 shows the calibration results of four cameras.

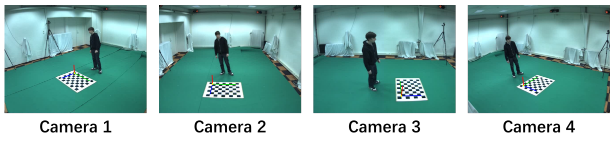

Fig. 1: Calibration result images of 4 cameras

B. Color model building

Figure 2 shows the visualization of the color histogram of the four characters in Camera 1. It is obvious that the color models of the four individuals have large differences and can be used for character matching.

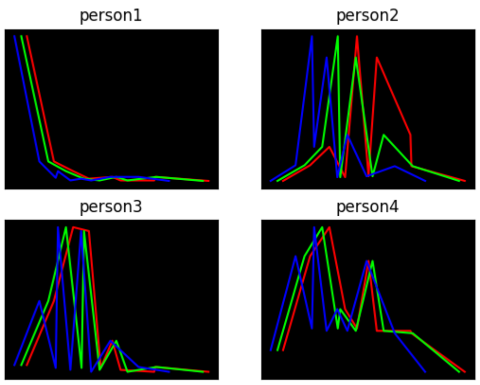

Fig. 2: Histogram of four people in Camera 1

C. Colored 2D trajectories of each person

Figure 3 shows the path image generated by two methods. The paths of the four people shown are represented by four coloured lines, if a person identification is wrong, the colour of their path will change accordingly. As can be seen from the left, the colour-based feature identification has the ability to self-test, self-correcting back to green when the colour green is incorrectly thought to be light blue. A light blue line segment is visible in the middle of the green line on the path. The corresponding red circle in the diagram on the right shows that when the location-based identification is wrong, it stays wrong. The light blue line when turns blue, stays blue.
We can see the intermittent breaks in the chart, when a person disappears their path also stops updating and when they are re-identified the path continues, so causing the intermittency of the path.
For full colour results and a demonstration of the paths please see the video, the link to which is in "Supplement material".

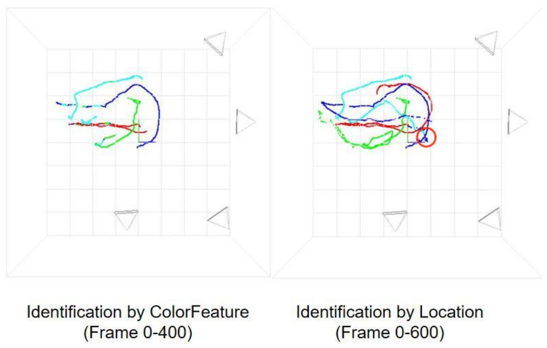

Fig. 3: Path image by two identification methods

IV. Supplement material

Result Video: https://youtu.be/4YcOPWZ3NXE