Université Claude Bernard Lyon 1

MIF23 – TP Report

Master Informatique 1ère année – Université Claude Bernard Lyon 1

BENSOUSSAN-BOHM Daniel

DONNAY Robin

Content

Introduction

For this TP, we wanted to make a game where the user has to push back obstacles that are coming of both sides of the screen. The goal of this game is to touch hazards with your hands to push them back.

Implemented Algorithms

Knn

Our first objective was to extract the user's body, at least his head, arms and hands. That's why we decided to use an algorithm called KNN to filter these parts.

KNN (k-nearest neighbors) is an algorithm commonly used for supervised learning. This algorithm first computes the minimum distance between our test data and the training samples to get the K-nearest neighbors. Then, when we have all the nearest neighbors, we will consider most of these nearest neighbors to be the prediction of our test data.

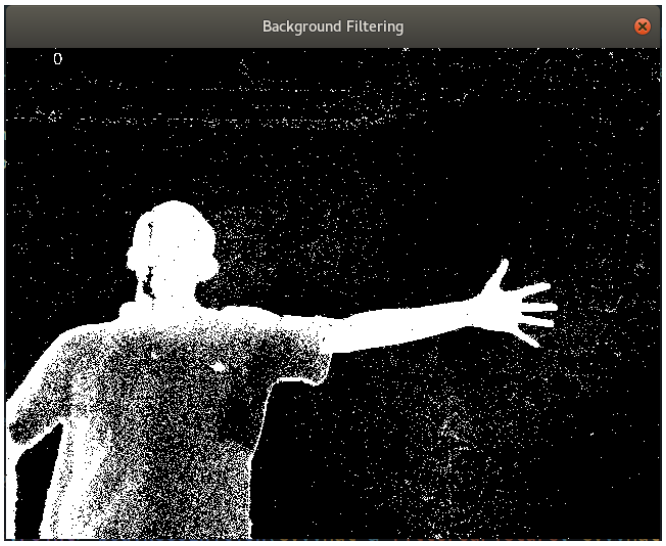

Figure 1 : Extracted body

Knn integration in the game

When we start the application, Knn is launched independently from the rest of the application. Since Knn is a learning algorithm, we store the last 2000 frames (which is 4 times more than in the initial algorithm offered in OpenCV) in order to refine our extraction as much as possible. Moreover, since every part of the application is based on Knn, this allows us to get a better precision for the detection.

Thanks to this algorithm, we were able to extract the user's body to use it for our next step. However, we also realized that there still was noise in the rendering. Hence, our next step was to remove that noise so we could get a better rendering and easier for the body parts identification.

findContours

Meanwhile, we also decided to draw the contours of the body. That's why we used the function findContours which also allow us to remove the noise from the rendering. Here's the way contours are calculated: every point with the same intensity or color are linked. The resulting curve creates the contours. In the picture below, the resulting contours of the body are drawn in green.

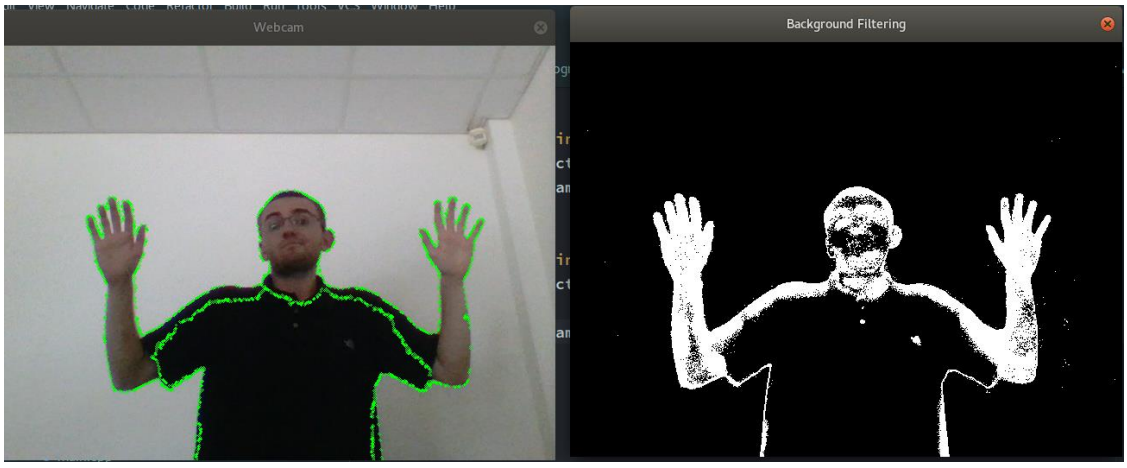

Figure 2 : Resulting contours of the body

GAMEPLAY IMPLEMENTATION

Menu

Now that we extracted the user's body, our goal is to implement the gameplay of our application.

First of all, we decided to create an interactive menu. Indeed, the user has to hold a button for 3 seconds to launch an activity. Four activities are available: see the contours of the body, see the result of the Knn extraction, see the user's arms position (Detector) and play a video game.

Figure 3 : Interactive menu

Game

The goal of the game is to touch balls that are appearing on the screen to push them back. Those balls can come from several directions of the screen. The player must touch the ball before it hits him.

Figure 4 : Screenshot of the game

Detection

This activity shows to the user the position of his arms on particular cases (arm raised, arm extended horizontally, ….).

Figure 5 : Arms position identification

Conclusion

To conclude, we have a single-player game with movement detection, but we can imagine having a multiplayer game. Furthermore, the extraction algorithm and the gameplay implementation are independent which allow us to create more games without worrying about the body identification part. Thanks to that, it is also possible to improve the extraction part without impacting the games.