Bachelor Project

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Graphics and Interaction

Human tracking using computer vision with a data output

Vilém Jonák

Supervisor: MgA. Vojtěch Leischner
2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Jonák**	Jméno: **Vilém**	Osobní číslo: **491942**	

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**

Studijní program: **Otevřená informatika**

Specializace: **Počítačové hry a grafika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Trekování lidí pomocí počítačového vidění s datovým výstupem

Název bakalářské práce anglicky:

Human tracking using computer vision with a data output

Pokyny pro vypracování:

The thesis aims to design and create a spatial audio interface in the real world. The student will do this using human tracking and will augment the spatial audio application Trick the Ear (Leischner, 2021). The users will wear headsets while they move in the designated area. There will be objects in this area representing the virtual audio sources. As the user moves in the area, his audio mix changes according to his position to these virtual audio sources.
The student will be solving these tasks:
 Research on computer vision approaches for human tracking and on suitable camera types.
 Select appropriate methods to determine the user's position in the area
 Propose also a solution to a problematical tracking of humans from the top view with a minimizing mismatch rate.
 Create a mobile application that will continuously
o display the scene with the user´s position and the position of the audio sources visualized as a 2D interface,
o send the user a proper audio mix to the headset,
o notify the user if the tracking or sending of the audio failed.
 Use Processing IDE with JAVA language for implementation.
 Test the application and the interface with human participants of different heights and looks and
o measure the tracking accuracy for 10 users.
o Determine the dependency of tracking accuracy on an increasing number of participants.
Define the limitations of the application as well as the optimal state (with how many participants will the system be able to give them a pleasant experience)

Seznam doporučené literatury:

[1] Akpan, I., Marshall, P., Bird, J., & Harrison, D. (2013). Exploring the effects of space and place on engagement with an interactive installation. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13). Association for Computing Machinery, New York, NY, USA, 2213–2222. doi:https://doi.org/10.1145/2470654.2481306
[2] Leischner, V. (2021). Trick The Ear. Retrieved from https://tricktheear.eu/
[3] Trifonova, A., Jaccheri, L., & Bergaust, K. (2008). Software engineering issues in interactive installation art. Trondheim, Norway. Retrieved from https://www.inderscienceonline.com/doi/abs/10.1504/IJART.2008.019882?fbclid=IwAR1QiZle5enqfvC0DZN5tIfJLTJgUh4njaBmWr45o7oRP_uvK5uvoxmqfos
[4] Veeramani, B., Raymond, J., & Chanda, P. (2018). DeepSort: deep convolutional networks for sorting haploid maize seeds. BMC Bioinformatics 19. doi:https://doi.org/10.1186/s12859-0182267-2

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Mgr. art. Vojtěch Leischner   katedra počítačové grafiky a interakce   FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **16.02.2022**   Termín odevzdání bakalářské práce: **20.05.2022**

Platnost zadání bakalářské práce: **19.02.2024**

Mgr. art. Vojtěch Leischner podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

.

Datum převzetí zadání	Podpis studenta

Acknowledgements

I would like to express my deepest thanks to MgA. Vojtěch Leischner for his guidance, patience, and helpful suggestions. A special thanks goes to Ing. Roman Berka, Ph.D., and his approval to test my work at the Institue of Intermedia at FEE CTU.

Declaration

Abstract

We have developed an interactive installation where our program tracks people with a camera mounted on the ceiling and creates an appropriate spatial audio mix for their headphones. The spatial mix is rendered with the Resonance Audio engine[1]. The main problem that we needed to solve was to remember detected listener positions, so if the detection fails and loses track of them and then again finds them, their identity would not change. Minimizing this mismatch rate is crucial for the correct function of the whole installation. Human detection and tracking is a shared computer vision task. Many solutions exist that are appropriate for particular use cases. We needed to solve how to track people from the top view, assign them an id, and remember them with the given id. We decided to use the convolutional neural network YOLOv4 (You Only Look Once) for people detection in the frame received from a camera and the DeepSort algorithm for tracking. YOLO is trying to predict bounding boxes alongside the class probabilities for these boxes. It is a single neural network [2]. This paper's objective is not to create a tracking algorithm but to decide which one will suit our use case, extend it and implement it properly. We have verified that our application is viable for multiple use cases with user testing. For example, it might be suitable in gallery settings or silent concerts in public spaces.

Keywords: spatial audio, human tracking, computer vision, YOLO, DeepSort, Resonance Audio

Supervisor: MgA. Vojtěch Leischner
Karlovo náměstí 13,
12000 Praha 2

Abstrakt

Vyvinuli jsme interaktivní instalaci pro trackování lidí kamerou zavěšenou u stropu a následné vytvoření odpovídajícího prostorového audia v jejich sluchátkách. Prostorový mix je renderován pomocí Resonance Audio enginu[1]. Hlavní problém, který jsme řešili, bylo zapamatovávání si detekovaného posluchače tak, aby pokud detekce selže, ztratí posluchače, a po chvíli jej znovu objeví, se jeho identita neměnila. Minimalizace této chyby je zásadní pro správné fungování celé instalace. Detekce lidí a jejich trackování je sdílená problematika počítačového vidění. Existuje proto mnoho řešení, hodící se vždy na konkrétní případ. Potřebovali jsme vyřešit jak trackovat lidi shora, přiřadit jim identifikátor a ten si společně s nimi zapamatovat. Rozhodli jsme se pro použití konvoluční neurální sítě YOLOv4 pro detekci lidí z obrazu přijatého z kamery a DeepSort algoritmus pro trackování. YOLO se snaží předpovídat ohraničujicí oblast společně s třídními pravděpodobnostmi pro tyto oblasti. Je to samostatná neurální síť[2]. Cíl této práce není vytvoření trackovacího algoritu, ale výběr toho nejvhodnějšího a jeho správná implementace a rozšíření. Testováním s uživateli jsme potvrdili, že naše aplikace najde uplatnění ve více prostředích. Například by se dala využít v galeriích nebo na silent disco koncertech.

Klíčová slova: prostorové audio, trackování lidí, počítačové vidění, YOLO, DeepSort, Resonance Audio

Překlad názvu: Trackování lidí pomocí počítačového vidění s datovým výstupem

Contents

Figures

Chapter 1

Introduction

The recent years introduced us to the trend called "silent disco," which is becoming more popular every year [3]. A silent disco is an approach where visitors of a concert hear the music only in their headphones. With broadcast directly from the performer's output mix, there is no need for PAs[1], and the concert can remain silent. The possibility of experiencing live music performance without any disturbing effects inspired us to push this interaction even further.

This trend is famous for implementing a solution that is not disturbing to the nearby inhabitants and the listeners. Everyone can adjust the volume and converse with other participants quickly after putting their headphones off.

However, with this approach, the user loses the feeling of music propagating through the space from a single point which is a significant disadvantage. The static audio the user hears in their headphones during these concerts is flat as it lacks the natural reverb and repercussions. This non-natural sound is one of the primary deficiencies compared to regular live performances.

In our previous work [4], we tried to control the position of audio sources around the listener using a haptic interface. It inspired us to try an inverse approach - control the audio of multiple listeners moving around a stationary virtual audio source, and create an interactive installation. The listener's audio mix in his headphones would be affected by his movements around static virtual audio sources. A similar effect can be achieved using a VR headset. The developer can implement virtual audio sources with specific properties, and the user will hear them in his headset accordingly. However, this approach needs special equipment that is usually very expensive. Users are limited to staying in a virtual reality environment, which means they would not see other people. Therefore, it is not an ideal solution to this problem.

The goal was to create this installation accessible for anyone without any uncommon devices. The listeners would not need anything more than a mobile phone and headphones. The installation would consist only of a web camera, optionally Raspberry Pi for better performance and control, and a reasonably strong computer.

[1]Public Address System - equipment for making the sound louder in a public place[36]

A setting like this is suitable also as an interactive installation in a museum. It will attract the user's attention to a specific place or work as a substitute for usual commented tours. For example, a painting could be an audio source representing information about the exhibit. Alternatively, it could augment the silent disco concert where the user will experience the performance with better immersion.

Chapter 2
Review

2.1 State of the Art Tracking Algorithms

Our first step in choosing the suitable methods for tracking people was to choose the tracking algorithm. We need to track the people accurately in real-time from a camera stream. The neural networks are rapidly developing, and many of them can solve this problem. The following chapter will discuss the advantages and disadvantages of a few neural networks between which we were deciding.

2.1.1 Region-Convolutional Neural Network (R-CNN)

Region-Convolutional Neural Network appeared in 2014 due to stagnating neural network research. It beat that time's best results on PASCAL VOC 2012 by 30%. The idea was to modify the CNN (Convolutional Neural Network). Three modules create together the R-CNN. Firstly, it generates region proposals that are class-independent. These create the set of candidates for the detector. Secondly, a large CNN gets a feature vector with a fixed length from each region. Last module consist of a set of linear SVMs[1] which are class-independent[24]. However, even with many improvements like Fast R-CNN and Faster R-CNN, this method was obsolete with different approaches called YOLO and (You Only Look Once) and SSD (Single Shot Multi-Box Detector)[25].

2.1.2 YOLO and SSD Comparsion

The main idea for creating YOLO was to detect objects in real-time. Firstly introduced in 2016 as YOLOv1, it made the detection of objects a regression problem for separating bounding boxes of the detected objects. The bounding boxes and their class probabilities are predicted in a single evaluation, making YOLO a single neural network. Since version 1, much newer upgraded versions were introduced. The real-time detection was optimized with the One-Staged Detection model implemented in YOLOv4[26].

[1]Support vector machines [35]

On the other hand, we were considering the use of SSD. This algorithm detects objects with the deep neural network. The SSD considers the shape of objects to make the detection more precise. With the values generated for each category object in a bounding box, the algorithm gives us another fast One-Staged detection neural network.

We decided to choose the YOLOv4 over SSD because it has better performance and is more accurate. A notable fact is that the choice of a dataset for training affects the performance[26]. The YOLO we used is trained on the COCO dataset[29]. COCO offers object segmentation and recognition of 80 classes[31]. It is essential for us that the dataset consists of RGB images. That was one of the main reasons we decided to use an RGB camera, which we will explain in section 2.3.1

2.1.3 Differences Between YOLO Versions

Nowadays, the first two versions of YOLO are considered obsolete, and the newest versions are v3 (2018), v4 (March 2020), and v5 (May 2020). All mentioned networks output set of bounding boxes and classes. With a proper dataset and fast GPU, their accuracy can reach 98% in real-time[27]. The new thing about v5 is that it relies on Python language. Its predecessors used C. In this work [28], the best results were achieved with a YOLOv4 with *tiny* model. However, this work [29] shows that the YOLOv5l is more accurate and slightly slower than YOLOv4 and YOLOv3. That means it depends highly on the use case and the dataset used.

All of the mentioned YOLO versions can be used with different trained models. The models differ in size and the number of layers, making them run faster for the cost of accuracy. For example, the YOLOv3 and v4 have normal and *tiny* models. The YOLOv5 comes in v5s/m/l/x[28] variants. It gives the user choice of non-programmable optimization for his application. Choosing correctly between a "fast and less accurate" or" slow and highly accurate" solution is crucial for a particular use case.

2.1.4 DeepSort

We decided to use the YOLOv4 with *tiny* model as it appeared to suit our use case the most. Our solution uses an approach where the DeepSort algorithm accompanies the YOLOv4. It tracks objects using their motion and appearance by comparing the last and current frame and the detected objects in them[30]. It helps us remember the ids of the detected persons correctly by tracking people's movements. We have an object detector (YOLOv4) and an object tracker (DeepSort) to detect and remember persons with their ids.

2.1.5 Blob Tracking

One of the well-known tracking algorithms is blob tracking. Usually, the objects are tracked by comparing color or light differences between frames with correlation or the color histogram. It typically uses background subtraction to make the tracking more robust. Pixels that are close enough are considered one blob accordingly to specified rules. Realize that the background subtraction approach is only possible if the camera is stationary[33].

Because this approach is straightforward and well documented caused us to think of it at first. However, for our use case, blob tracking has a significant disadvantage. When two blobs, in our case, people, happen to be touching each other, the blobs merge into one. They split after the people move far from each other. That must never happen, as we instantly lose the reference to that person and cannot send the audio to the proper user. We can avoid it by counting the probable trajectory from previous frames. However, if two persons walk toward each other and stand still for a while during a conversation, this approach would not have any effect. That is why blob tracking is not suitable for our purpose and why we had to choose a different tracking algorithm[33].

2.2 Resonance Audio

Resonance Audio is an SDK (Software Development Kit) for spatial audio with high fidelity at scale. AR, VR game developments, and video are all fields using it, as well as spatial audio players for augmenting music experience[32]. Supported platforms are Unity, Unreal, FMOD, Wwise, **Web**, Digital Audio Workstation (DAW), Android, and iOS. It is designed for simulating the spatial behavior of sound. That makes the software an ideal choice for our installation.

The physical sound environment is very complex, with many variables. Sound bounces off the surfaces, can be absorbed, and every person has a subjective perception of it. The fact that humans have two ears helps them determine the location of the sound source.

The interaural time difference (ITD) is the difference between the wave's arrival between the left and the right ear. That helps people localize the horizontal position of low frequencies. This perception is relative. That means the time difference grows with the distance from the source[5].

To determine the horizontal location of high frequencies, people cannot use ITD. They use the interaural level difference (ILD) instead. That works similarly to ITD, but perceived is the difference between the sound volume between the left and the right ear[5].

Finding the elevation level of the sound source works in a slightly different way. The changes in frequency or so-called spectral effects help people find the vertical position of the source. The wave is reflected in our ears differently depending on the direction from which the wave came. That, together with perceiving the changes in frequencies, is used to help us determine the vertical position[5].

The Resonance Audio simulates this reality using Head-Related Transfer Functions (HRTF) for credible spatial sound effects. HRTF is described as the ratio between two Fourier transforms. One of the signals on the listener's eardrum and one at the center of the listener's head with them absent[6].

2.3 Camera Systems

Another long talk during our preparations was about what camera would be ideal to use. We decided between a standard RGB camera, IR (Infrared) camera, and a stereo camera. Every system has advantages and disadvantages, and we needed to choose the one that would suit our problem. The neural network we have chosen, YOLOv4 with DeepSort, has a model trained with an RGB camera. The idea of having an installation that is not expensive and does not require much special equipment convinced us to use an RGB camera, which is ideal for the tracking mechanism we have chosen.

2.3.1 RGB Camera

RGB camera is a system known probably to everyone reading this paper. It appears in every smartphone, every web camera, and many other places. RGB Cameras are not hardware-specific, making them "plug and play" and do not need any additional complex setup or calibration.

They acquire an image using the CMOS chip [7] which is a part of a sensor that creates an RGB frame as a result. As mentioned above, the advantages are availability and easy usage. The camera's price can vary depending on the field of view and the camera's resolution.

There are also some disadvantages which create inconveniences in our installation. The main disadvantage is that RGB cameras need stable light conditions to show the images properly. Not only stable, but the light has to be intense to make the neural network track robustly. We will describe how problems with light conditions during testing confused the tracking mechanism in chapter 4.

2.3.2 IR Cameras

The IR camera can either track the thermal heat of users or can track infrared light, which is an invisible specter to people. It will solve the problem of RGB cameras with light conditions.

The possibility of tracking IR markers is common for determining the pose of VR controllers. That is usually done by a camera in the headset, which tracks IR LEDs on the controllers [8]. We could determine the position of a person, but the problem is that we would not have any possibility to remember the IDs of these persons. It could be done similarly to our solution, which would defy the advantage of the IR system. The IR markers also need to be implemented on the headsets, and possible occlusions could appear, causing the person to be lost.

A different approach is to emit an IR light to the scene using an IR illuminator. Surveillance cameras use that to capture video during nighttime. These cameras often have an internal illuminator created by many IR LEDs around the eye of the camera, which illuminates the space with, for humans, invisible light. Using night-vision cameras like these would create a solution invulnerable to sudden light changes or streams of light directed into the eye of the camera. However, the main disadvantage is that this system could not be used outdoors during the daytime as the sun emits a lot of IR light, and the camera vision would be washed out.

2.3.3 Depth Detection Cameras

Stereo RGB cameras are a solution for achieving depth perception. The system is inspired by how people use their eyes to perceive depth. It counts the depth from the frames of two cameras, knowing the distance between them and determining how far the object is.

The main disadvantages are the high computational demand of the algorithms counting the pixel differences between frames and the need for illumination (same as RGB Cameras). The lack of proper lighting conditions or the lack of textures will make the system less accurate[9]. The accuracy of the system decreases with increasing distance[10].

Knowing the depth of the objects, we could detect people by tracking the highest object, which would be the head of the user. Possible glitches could appear in the edge case if the person bends or gets very close to another. The stereo cameras have high computation demand and more complex calibration than RGB cameras. Another solution to determining the depth in the frame is depth cameras. However, these systems usually have a limited field of view and maximum distance for which they work properly. For example, the Kinect v2 and newer Azure Kinect DK have small maximum operating ranges. Kinect v2 4.5m[11] and Azure Kinect 5.46m for resolution 320x288p[12].

Chapter 3
Implementation

3.1 Camera Stream

We decided to use Raspberry Pi Camera module V2.1 for our experiments. Compared to standard web cameras, it has a more stable capture performance and has properties modifiable with Raspberry Pi libraries. Even though the Pi Camera gives us the possibility to capture video stream in 720p resolution on 60 FPS [13], we were not able to achieve a stable framerate when we were sending frames of this resolution.

Raspberry Pi 3B+, which we used in our setup, comes with 1GB RAM and with 1.4 GHz 64-bit quad-core ARM Cortex-A53 CPU [14]. That gives us enough strength to capture and send a video stream in 640x480p resolution. It is the highest possible resolution we were able to stream in real-time with a reasonable 60 FPS without any, or rare, image tearing or corruption. We confirmed that when we set up the environment for development and observed the quality of the stream affected by the resolution.

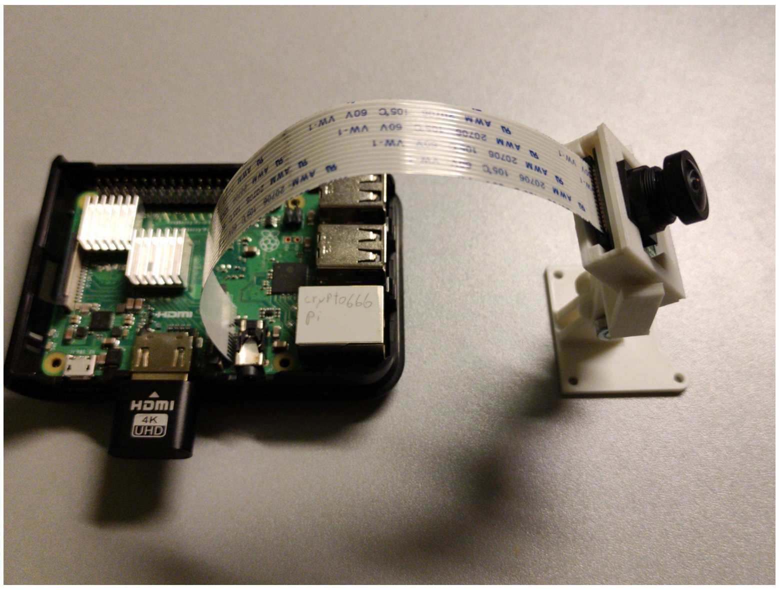


Figure 3.1: Raspberry Pi 3B+ and Camera module V2.1

3.1.1 Open CV

OpenCV is an open-source computer vision library suited for video capture and image processing on Raspberry Pi[15]. We use the latest version, 4.5.5. The library offers us a straightforward method of capturing an image from a camera. That is afterwards decoded as *.jpg* format image. Before sending the packet, we parse the image as a byte array. Chunks of size 2^{16}-64 bytes are then used as packets for UDP. The subtraction of 64 bytes prevents the UDP from frame overflow. Fragments of the code that performs these actions:

```
MAX = 2**16
MAX_IMG = MAX - 64
...
compressed = cv2.imencode('.jpg', img)[1].tobytes()
count = math.ceil(len(compressed)/(self.MAX_IMG))
array_start = 0
while count:
    array_end = min(size, array_start + self.MAX_IMG)
    self.s.sendto(struct.pack("B", count) +
        compressed[array_start:array_end], (self.addr, self.port))
    array_start = array_end
    count -= 1
```

3.1.2 UDP

User Datagram Protocol (UDP) is a mechanism based on packet-switching in a computer communication using Internet Protocol (IP)[16]. The idea is to offer the possibility to send messages to other programs with the lowest possible need for protocol management. The main difference between UDP and TCP (Transmission Control Protocol) is that the UDP is not as reliable but is a faster protocol. Redundancy or successful reaching of the destination is not guaranteed, which results in higher stability and speed of the protocol. If the application needs a reliable transmission, rely on the TCP[17]. In our solution, the speed of the stream is crucial, so we chose to use the UDP.

We connect the Raspberry Pi with a PC via Ethernet cable and local network for even more excellent stability in our solution. Raspberry Pi has integrated 2.4GHz, and 5GHz IEEE 802.11.b/g/n/ac wireless LAN [13], but during experiments, we encountered occasional connection drops when using the wireless connection. We send the packets to the IP address of our computer's ethernet network interface controller. The computer receives the data on the same IP address.

3.2 Tracking Mechanism

We face the problem of tracking people from the top. After the research described in chapter 2, we decided to use a tracking mechanism implemented with YOLOv4, DeepSort, and TensorFlow[18]. The YOLOv4 neural network dataset obtains more types of objects—for example, cars. What we want to track is passed to the detection method as a parameter. Precisely as an array of strings representing the classes. So we add to our list only the string "person." There are 80 classes that the neural network will recognize[31]. We need to keep in mind that we also need to perform perspective projection with the received frame. That is because we need to have linear distances between all points in the frame to avoid glitches when rendering spatial audio.

3.2.1 Neural Network Setup

We can run the neural network on a CPU or GPU. GPU has a complex setup and architecture limitations. On the other hand, the CPU has computational limitations. We need to set up the CUDA toolkit when using the GPU variant. That limits us to use only NVIDIA graphic cards. In our PC, we have GeForce GTX 750Ti, which gives us performance comparable with the CPU of our PC, which is Intel Xeon W-2125 4.00GHz.

Our algorithm finds objects using YOLOv4 and then tracks their position using Deepsort. We downloaded a pre-trained model for YOLOv4[18]. For faster performance, we can use the *tiny* model, which is less accurate and more storage efficient, which results in a better performance of the program. We convert this model into the corresponding TensorFlow model used in the detection program with python script.

3.2.2 Image Perspective Transformation

Our script runs in an infinite loop. In each iteration, it receives a frame from our camera stream described in section 3.1. After the image decoding, we need to perform a perspective transformation to ensure that the frame has uniform distances between points to track people's movement reliably. The OpenCV library offers us a very effective and straightforward way to do this.

First, we need to find 4 points in our source scene. These points will represent the borders of the output screen. As we are testing this installation at the Institute of Intermedia (IIM) at CTU [19], I have chosen these points as follows:

1. The bottom left is the left corner of the camera FOV.
2. The bottom right is the right corner of the camera FOV.
3. The upper left is the corner of the dance floor installed in IIM
4. The upper right is at the intersection of the dance floor and the camera field of view.

The upper points can be seen at figure 3.2 as pink and white dot respectively.

We need another four points representing borders into which we want to fit our trimmed source frame. In our case, these will be corners of the 640x480p frame. These two arrays of points are passed to the OpenCV method *getPerspectiveTransform()* [34] which will return a 3x3 transformation matrix.

Next, we call the method *warpPerspective()* [34] with our source image, the transformation matrix, and the resolution of the destination image as parameters.

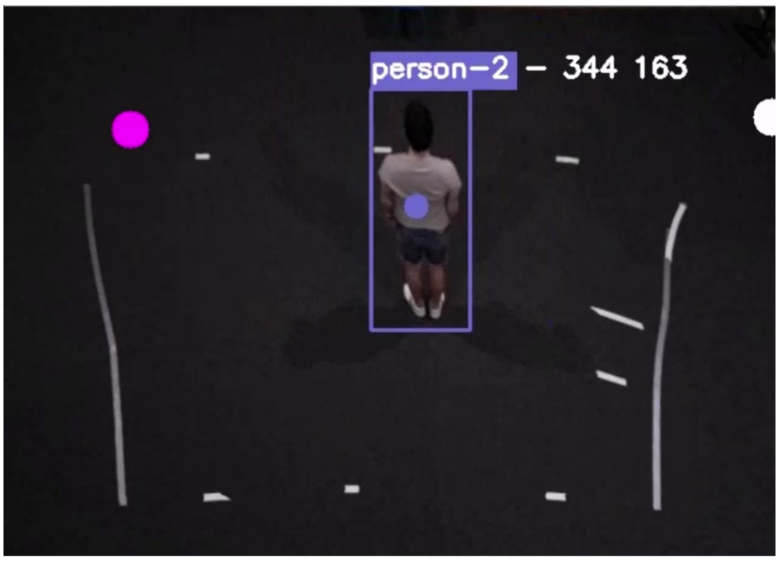
Finally, this function returns a transformed image [34]. Image as this is ready to be used in the tracking algorithm. The algorithm uses the TensorFlow library to find the objects described above in our array.

3.2.3 Tracking

If we want to detect, for example, people and cars, we need to modify the array as mentioned above. The recognized objects set up in the array are then put into the Deepsort tracker, which returns a list of *tracker* class instances with appropriate ids. It gives us the tracked object as a bounding box of it. So we get coordinates of the upper left and bottom right corner in the form of the python tuple together with a unique id. We added the computation of coordinates of the tracked person's centroid. That was achieved easily by using the coordinates of the corners as follows

$$\mathbf{m}_{(x,y)} = [(l_1 + r_1)/2, (l_2 + r_2)/2], \tag{3.1}$$

where $\mathbf{l}$ is the top left corner and $\mathbf{r}$ is the bottom right corner, which results in following



(a) : The rectangle is created by the output of the YOLOv4 and DeepSort. The dot at the centre of it is computed centroid, which represents the coordinates, that are sent to the server and used to render the spatial audio.

Figure 3.2: Tracked user and the computed coordinate that is sent to server

We store the $\mathbf{m}$ and a unique id into an instance of a class if an instance holding the same id does not already exist. In that case, we update the coordinates with an interpolation mechanism, which will be described in detail later on.

It is inevitable to encounter situations where the camera stream loses the frame, we lose sight of a person because of occlusion, for example, or the tracking mechanism is confused and cannot recognize a person it did recognize in the previous frame. After the first experiments, the neural network with *tiny* model proved its quality at re-recognizing lost persons. If we start tracking a person and mark it with id 1, lose the person for a few seconds, and then the person reappears, the YOLOv4 with *tiny* model and DeepSort will very probably give them the same id. So we try not to forget the person's last known location by storing it as a value of the class instance. According to initial setup tests, we found out that the offset of 3 seconds works best - in most cases, it gives the lost person the same id it had before it was lost.

Another encountered issue was that after finding a person that was lost recently, the tracked location "jumped" from the last known position to the new one. That would appear as an uncomfortable and unrealistic audio glitch in the final spatial mix. We used the same approach as in our previous work[4], where the same problem appeared. To solve this we are using smoothed interpolation between previous and new position. The interpolation update function looks like this:

```python
def update(self, newLocation):

    #acceleration is based on the distance between last known
        and current position
    acceleration = vp.vector(newLocation[0] - self.coord[0],
        newLocation[1] - self.coord[1], 0)

    #'norm' values are values scaled to the size of the frame
    normNewTarget = vp.vector(newLocation[0]/480,
        newLocation[1]/640, 0)
    normCoord = vp.vector(self.coord[0]/480, self.coord[1]/640,
        0)
    normDist = calc_distance(normNewTarget, normCoord)

    #Here I do some minimal distance offset
    if sqrt((self.coord[0] - newLocation[0])**2+(self.coord[1] -
        newLocation[1])**2) < 10:
        #self.coord = newLocation
        return
    else:
        #I change the magnitude of the acceleration according to
            the size of the distance
        acceleration.mag = 1 + 2*(normDist*20)
    velocity = acceleration
    #I returned 'moved' coordinates of last know position
    self.coord = (int(self.coord[0] + velocity.x),
        int(self.coord[1] + velocity.y))
```

We use the coordinates described in our *update()* function to create an appropriate spatial audio mix. This approach performs a smooth interpolation from the last known position to the currently tracked position. The neural network can return slightly different results in each frame if the person is standing still (we speak about a few pixels). We added the minimal offset to the update function. If the new location changes by less than 10 pixels, we do not perform any update. Otherwise, the tracked point would oscillate around the tracked person even if they were standing still.

3.3 Coordinate Stream

Our problem with receiving is that our strategy sends all detected coordinates to every device. So we must determine which are the correct coordinates on the JavaScript application side. We are adding heuristics to help minimize the mismatch rate of choosing the correct coordinates. We will describe that in this section.

3.3.1 Used Communication Protocols

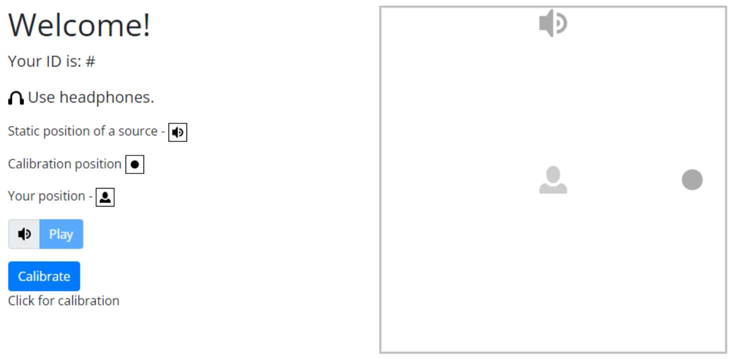
We stream coordinates locally using Open Sound Control (OSC) protocol to a Processing[20] script and from there to a proxy server using WebSocket protocol. We do so for two reasons. The main reason is to perform a simple conversion between OSC and WebSocket. Our server script is in JavaScript, and it does not natively support OSC. We also did not find any simple way to use WebSocket in python without making it a parallel application. We decided to create a proxy script in Processing. It accepts the OSC packet, parses it as a string, and sends as a WebSocket packet to our proxy server. The other reason is readability. This wasy the python tracking script can only contain logic maintaining the neural networks and camera stream.

OSC is mostly described as a protocol. However, it does not implement error handling, negotiation, or processing schematics. A better way to describe the OSC is to describe it as a content format. The other formats are, for example, XML or JSON. Initially, the OSC was implemented for communication between sound synthesizers but found much broader usability. It can be sent via UDP/IP, Ethernet, USB, or Bluetooth but has to be received with another OSC parser[21].

The WebSocket Protocol enables a form of two-way communication starting with a handshake. It is based on the TCP protocol, and HTTP servers interpret the handshake. The WebSocket protocol is a single modified TCP connection used in both directions. This approach avoids using HTTP as a bidirectional communication. As HTTP was not implemented to have such a function, using it so would create a variety of problems[22].

3.3.2 Initial Calibration

When a user wants to connect to the application for the first time, calibration needs to be done. The user repeats the exact calibration when the neural network loses them and cannot find them again in time. The calibration is controlled by the user via the web application, which informs them about success eventually.


Welcome!

Your ID is: #

Use headphones.

Static position of a source -

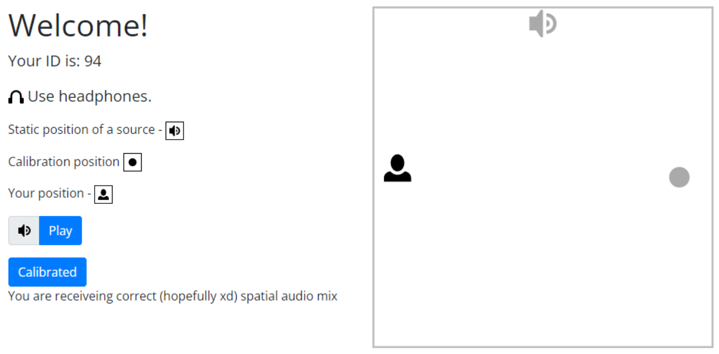
Calibration position

Your position -

Click for calibration

Figure 3.3: Basic look of the web interface before calibrating

First the user needs to stand in the designated calibration area. This area is a circle marked on the floor. The program knows this calibration area as an exact point but checks for any coordinates in a circular vicinity. When the user is in place and presses "Calibrate" button in the app, the label changes to "Calibrating" and the button becomes disabled. During a 5 seconds timeout, the script checks received coordinates and look for one whose distance from the calibration point is less than our offset. If it finds none, or if there are some other coordinates in the vicinity, the calibration will be unsuccessful, and the user shall try it again.


Welcome!

Your ID is: 94

Use headphones.

Static position of a source -

Calibration position

Your position -

You are receiveing correct (hopefully xd) spatial audio mix

Figure 3.4: Look of the web interface when the user is calibrated

3.3.3 Audio Rendering

If the calibration is successful, the program remembers the id associated with that coordinates, and the user will start receiving audio. The app contains a typical play/pause button if the user wants to stop the audio but not the stream. The audio stream will be rendered using the spatial audio engine

Resonance Audio. It is initialized with the size of the room, materials from which it consists, and the audio source's coordinates. This setup affects the rendering mechanism accordingly with the real-time received listener coordinates.

The web application visualizes the room, which appears as a rectangle with three icons. These represent the stationary audio source, calibration point, and the listener's corresponding position in the scene. This visualization is handy for checking if the listener receives coordinates corresponding to his position in the scene. The icon of the calibration point is there to help the user avoid it, if possible, to prevent other users from experiencing unsuccessful calibration.

3.3.4 Unsuccessful Tracking

If the web application receives negative coordinates, the tracking mechanism has lost the user from sight for a longer time. If that happens, the app's state returns to "Not Calibrated," which appears on the screen to inform the listener. During this time, the user hears audio with their last known position coordinates, and they should go to the calibration point and calibrate them accordingly to 3.3.2.

However, sometimes the tracking mechanism loses the person only for a short time and then assigns it a different id. When this happens, we have an opportunity to prevent the user from the need of re-calibration. We achieve this with a simple heuristic. By remembering the coordinates from the last frame, we can decide if some newly received coordinates with another id could be our coordinates only with a mismatched id. We use a similar approach we have done in the calibration process. If, for example, our last received coordinates were [43, 50], and then the negative coordinate is received, we look into the pack of all received coordinates for all ids. If there are some close to our last known position, e.g. [46, 48], we check if there is no one in the vicinity. If so, we decide that these coordinates are ours and update the id. If not, we cannot assure that we choose correctly, and the user needs to do the calibration again.

Chapter 4

Testing

As noted later in section 5.1.2, our testing space, which is a trapezoid with a surface taking up to around 16.165m^2, is suitable maximally for three people. That gives us approximately 5.3 square meters per person. How this could be improved is described in chapter 5 The testing took place at IIM, and three contestants participated. It aimed to measure the occurrence of the tracking algorithm errors that lead to the need for re-calibration. Also, it should help us determine which users' behavior creates those errors most often and receive the overall feedback of the users on the whole installation in the form of discrete interviews after the testing itself.

4.1 Scenarios and Research Questions

We decided to try three different scenarios of the users' behavior. Each should point to different yet expected cases that could happen when using our installation. We executed the first scenario twice. Once for measuring errors during the user testing, then without the video render to measure the FPS of the application. Users received brief instructions on how the web application works. We told them how to calibrate themselves and that any other relevant information or instruction is displayed in the web application.

The three scenarios were:

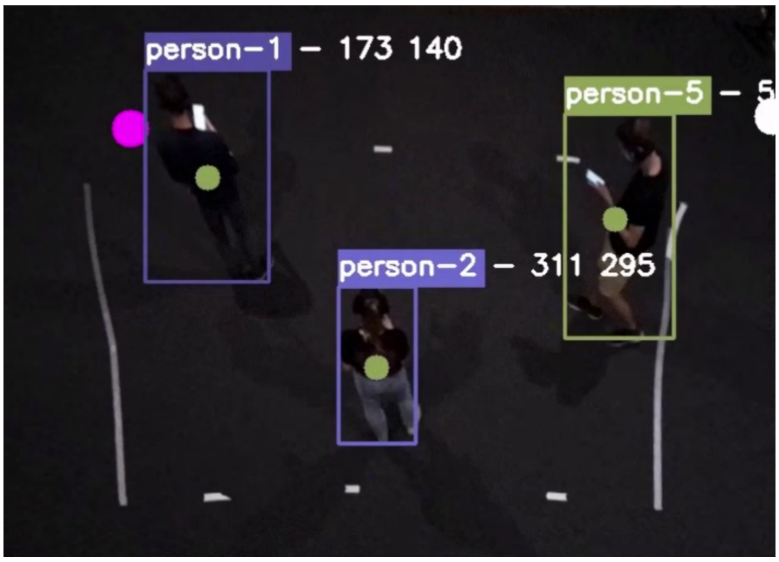
1. Users are slowly walking, trying to avoid direct contact with other users and observing the behavior of the application.
   - this scenario would represent usage in a museum exposition
2. Users are passing each other very closely and occasionally body hitting each other.
   - this scenario would represent a silent disco production
3. Users are passing each other closely and are exchanging object (like bags) between each other.

4.1.1 Testing Scenarios with Participants

Our three participants were of different appearances and heights. Let us call them user T, user M and user V. User V measures 193cm; User T measures 179cm; M has 169cm. All of the participants did not know the purpose of the application. Acquainted only with the instructions describing the application's user interface, and no expertise on computer vision or spatial audio, they should offer a non-biased review.

Testing Scenario 1

The aim of scenario one was to test if the application is usable in an environment with slow movement and without collisions with other users. Such conditions represent art expositions, which is one of the possible usages of our installations as described in chapter 1. The DeepSort lost sight of a user four times because of occlusions. However, it found them in under three seconds (which is our timeout) and assigned them the correct id, so there was no need for re-calibration.



(a) : We can see user V with ID 5 successfully finishing the calibration.

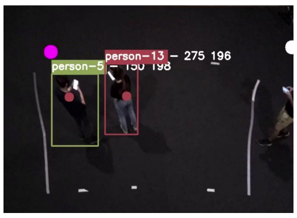
Figure 4.1: Participants right after calibration.

The measurement lasted for one minute and forty-five seconds. All the participants calibrated in 30 seconds without encountering any problems and without the need for assistance. During scenario one, no errors appeared, and no user had to re-calibrate.

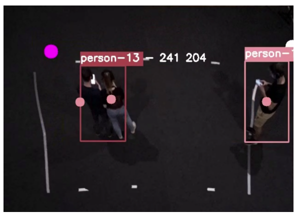
Testing Scenario 2

In scenario 2, we were trying to simulate an environment similar to the silent disco environment. Users had instructions to pass each other very closely and collide their bodies with other users on occasion.

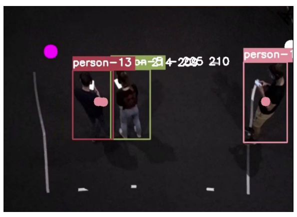
- At 0:30 of the test, participant T with id 13 and participant M with id 15 walked toward each other. Their bodies collided and then went back. The DeepSort switched their ids and created an error. Users were immediately aware of that thanks to our visualization displayed in the web application, left the scene, and at 0:40, headed for the re-calibration. T successfully calibrates with id 22.



(a) : T and M closing to each other



(b) : T and M hits each other



(c) : T's and M's ids switched

Figure 4.3: Users bounced off each other and their ids switched

- At 1:09, M re-calibrated and received id 25 for just one second after the successful calibration. After that, DeepSort incorrectly changed M's id to 27. However, our algorithm noticed this, and because no other user was around, and M had not gone away, it updated M's id to 27. The error was intercepted, and there was no need for re-calibration.
- At 1:50, M and V bounced off each other; ids switched; they needed to undergo the re-calibration.
- 2:01 M tries to re-calibrate. The same situation as at 1:09 happened, and the DeepSort switched the id right after the successful calibration. However, M walked away fast this time, and the update was unsuccessful. The error was not intercepted and M needed to undergo the re-calibration.
- At 2:30 the testing ended.

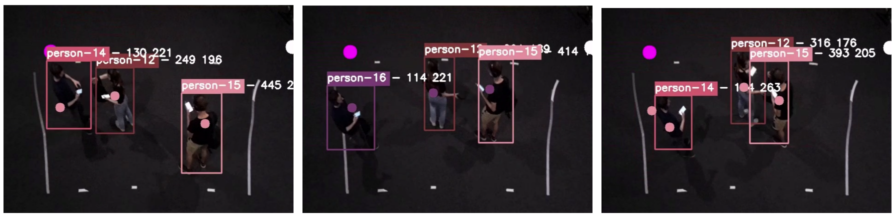
The measurement lasted for two and a half minutes. During this scenario, no one was lost for a time longer than our offset (three seconds), the id was changed by the DeepSort twice—our algorithm intercepted it once, and the ids of two users switched twice. There were five needs for the re-calibration.

Testing Scenario 3

The final scenario tested unordinary interactions between users. We wanted the users to exchange some visible objects with each other. As users V and M had bags with them, we decided to use their bags. T used a headphones case as the object. During the test, users were walking with bags on their backs, and at some time, they took their bag in hand and exchanged it for a nearby user's bag or the headphones case.

- At 0:29 V, T, and M have all been successfully calibrated.

- At 0:58, M and T approached each other and started the process of exchanging their objects. During that, M occluded T, and the detection algorithm lost T for a while. However, this lasted only 2 seconds. Closely after the T was spotted again, its id changed from 14 to 16. Our algorithm successfully intercepted that situation.



(a) : T and M are close to each other and exchange their objects

(b) : T changes id to 16

(c) : T's id changes again. Beginning of the change between M and V

Figure 4.4: Users are exchanging objects

- At 1:09, T's the DeepSort changed T's id again. Now back from 16 to 14. Again, our algorithm intercepted this error successfully.
- 1:33 and T exchanged bag with V without any problems or occlusions.
- 1:45 Testing ended.

The final scenario lasted one minute and forty-five seconds and proved that most errors occur when users' bodies collide. During scenario 3, our algorithm successfully intercepted two occurrences of incorrect change of user ids, and there was no need for re-calibration.

Measuring Performance in Scenario 1

We repeated the first scenario with three users and measured the FPS. We did not render any video, disabled control output prints, and concentrated the computer's performance only on the tracking algorithm. With a setting like this, the resulted FPS were 17.3 at peak and 14.1 at minimum. The total average during the 2-minute long run was 16.37 FPS. The performance was unaffected by the number of people in the scene. However, we could do the measurement only with three people, and the difference between detecting 1 or 3 people is inconsiderable. We repeated the measurement twice, and the values above are the minimum, the maximum, and the average of both runs. Measuring the regular model was not accounted for in the test, as the implementation was developed for using the *tiny* model. However, the regular model was running at 6FPS with video render during the experiments.

4.1.2 Research Interviews with Participants

We interviewed the participants discretely and separately so their answers would not affect other users' thoughts and bias them. The research survey consists of 4 subjects with a few subquestions as follows:

- User Interface of the Web Application

  RQ1.1: Was the interface comprehensible?

  RQ1.2: Did the visual representation of yourself helped you with orientation?

  RQ1.3: How long did it take to realize your position was switched with other user?

  RQ1.4: Do you think you will remember the controls of the application?

- Spatial Audio

  RQ2.1: How did you hear the audio during movements?

  RQ2.2: Did you notice any glitches in the audio?

- Calibration

  RQ3.1: Was the process of calibration intuitive?

  RQ3.2: Was the text feedback together with the visual representation enough to inform you about the need of re-calibration?

  RQ3.3: Did the text feedback and the visualisation help you during your calibration?

- Overall Experience

  RQ4.1: Was the experience enjoyable?

  RQ4.2: Can you imagine any practical or art usages of the installation?

  RQ4.3: What do you think would make the application more enjoyable?

Result of the Interviews

- User Interface of the Web Application

  All of the users found the interface easy-to-use, comprehensible, and agreed that the visualization of their position helped them with orientation in the space. Every user is sure that they remember the controls and instructions very well, and they will be able to use the application in the future without any further instructions.

- Spatial Audio

  T and V noticed that the audio changes accordingly to their position to the sound source icon in the web application. V described it followingly:

  > *"I heard changes in volume and pan between left and right speaker. When standing at the upper right corner of the scene, I have heard most of the audio coming out of my left speaker."*

  On the other hand, M did not glimpse the feeling of spatial effect but heard changes accordingly to their position. M and T noticed no glitches. V noticed one:

  > *"When I stood exactly in the middle, in front of the audio source, and made a step to any side, there was a sudden jump in the audio change."*

  The Resonance Audio engine probably creates this effect.

- Calibration

  All users agreed that after the entry instructions described where the calibration point is and how to calibrate properly, and with the text feedback, the calibration process was intuitive and easy to remember. They also agreed that the text and visual feedback were enough to inform them about the error and that they needed to re-calibrate. V added:

  > *"The visual feedback could be even magnified by changing the color of the button from blue to red, for example, when we need to re-calibrate"*

- Overall Experience

  Everyone agreed, that they enjoyed the application, and that it needs little improvements in preventing the need of re-calibration, when bodies of users collide. M and T mentioned the possible usage in a silent disco production, assuming the IR camera will be used. M and V agreed, that the installations would find best usage as interactive art exposition, making the exhibits "talking", for example. V added, that them can imagine, that the application could somehow complement AR glasses. Their ideas for making the application more enjoyable differed. T stated, that the application is nice, and needs to make the tracking more robust. M would prefer the possibility to choose their own song and their own icon in the web interface. M also stated, that the quality of the sound could be improved. V would like to see other users in the interface, or to choose which of the other users hey want to see—possibility to form parties of friends.

Chapter 5

Future Improvements

After the testing sessions with participants, we observed where our application could be improved. As the application acted robustly during the experiments, the major problem - the mismatch rate of assigning the ids correctly - could be even lowered with more complex heuristics. A better choice of hardware would also be a great benefit to the application's usability.

5.1 Hardware

5.1.1 Raspberry Pi

Our first deficiency starts right with the camera stream. We could not maintain better stream performance than sending 640x480p frames without corruption. The exchange of the Raspberry Pi 3B+ for the newer Raspberry Pi 4 would be beneficial. The newer model is better in every aspect of the computer. The Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz CPU, which is installed in the Raspberry Pi 4, offers a slightly bigger clock rate than the older model (which has 1.4GHz)[23]. The difference between the RAMs of these two models is significant. However, RAM should not be the bottleneck of the problem of sending a live video stream.

5.1.2 Camera

After installing the Raspberry Pi with a camera in the IIM, we encountered a problem with the camera's field of view. The Camera Module V2.1 we use has a focal length of 3.04 mm, a horizontal field of view of 62.2 degrees, and a vertical field of view of 48.8 degrees. It hangs at the ceiling of IIM, which is about 6.5 meters high and points under an angle of 30 degrees. This setup and the perspective transformation of the frame discussed in the section 3.2.2 creates such a frame that maximally, three people can use the installation comfortably. The view problem could be improved by choosing a more extensive field of view camera and placing it higher.

We want to try the IR camera in the future. IR camera would make this installment invulnerable to direct light-rays, and the installation would not need consistent light conditions. These factors would move our application to fit the requirements for silent disco production even more.

5.1.3 PC Hardware

The biggest bottleneck of our application is the main PC. The Nvidia GeForce GTX 750ti is an older graphics card that can run the tracking script at 16FPS when using a pre-trained *tiny* model. The regular model is running at 6FPS. Vojtěch Leischner stated that he tried the algorithm on his Nvidia RTX 3080 and reached 60FPS with the regular model. A better GPU would improve overall performance and make the application more robust because we could use the regular model with a reasonable frame rate.

5.2 Implementation

The testing showed that the calibration, the way we designed it, is intuitive, and people can undergo it mostly correctly. Although, a very welcomed improvement would be the usage of QR codes. The idea is that instead of pressing a button in the web application, the user would scan a QR code (on the ground, for example). The benefits of this approach are that they do not need to write a wi-fi password and the IP address of the server PC to their browser manually.

The heuristic we designed proved can solve the problem for which it was developed correctly in most cases. Nevertheless, another heuristic that would solve the switching of two users' ids would make the application even more robust and lower the mismatch rate and the number of needs for re-calibration to the minimum.

5.3 Environment

During the tests in IIM, we encountered a problem with lighting. IIM is a space designed for theater-like performances and has multiple strong lights. They create many shadows, which appear to confuse neural network detection. The regular model sometimes detected the shadow as a person. When we set up the lights, so the image would not be overexposed, the detecting gave the best results. A tip from Mr. Berka, the head of the IIM, also warned us that the carpet on the floor behaves like a static and can affect the accuracy of the detection algorithm.

Another experienced problem, which is more problematic, is the color difference between the background (which means the floor of the scene) and the user's clothes. If they match, the algorithm works much worse, and the rate of losing the user's sight, for a time longer than the offset, rises drastically.

Chapter 6
Conclusion

We created an installation giving a user possibility to experience a real-time interactive augmentation of the audio. A museum exhibition is a most fitting scenario for problem-free usage. The application has to be improved to augment the silent disco music production and give people a new way to enjoy live music.

We used YOLOv4 neural network for detection and Deepsort for tracking the users. The neural network infers people's position from a Raspberry Pi camera video stream sent by UDP. The program then sends the tracked coordinates to a JavaScript application. Users control web application to self-calibrate and play or pause rendered spatial audio and get visual feedback on their position in the scene.

The tests proved that our applications' user interface is easy to learn, and no test users had any problem understanding it. The visualization, which represented the user's position in real-time, helped them significantly.

Neural network faults create the only inconsistencies. These need to be improved in the future by creating more robust heuristics solving all the cases in which the detection algorithm can make mistakes, or switching to different means of tracking.

Bibliography

[1] Resonance Audio, Resonance Audio (2018), https://resonance-audio.github.io/resonance-audio/

[2] M. I. H. Azhar, F. H. K. Zaman, N. M. Tahir and H. Hashim, "People Tracking System Using DeepSORT," 2020 10th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), 2020, pp. 137-141, doi: 10.1109/ICCSCE50387.2020.9204956.

[3] Silent Disco: A Popular Trend that Has Been Out For Years, (2022), The Silent Disco Company, https://thesilentdiscocompany.co.uk/blog/silent-disco-history/

[4] GitHub repository, vilijonak/Bachelor-thesis/Semester Work, (2021) https://github.com/vilijonak/Bachelor-thesis/tree/main/Semester%20Work

[5] Resonance Audio, Resonance Audio - Fundamental Concepts, https://resonance-audio.github.io/resonance-audio/discover/concepts.html

[6] D. Y. N. Zotkin, J. Hwang, R. Duraiswaini and L. S. Davis, "HRTF personalization using anthropometric measurements," 2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No.03TH8684), 2003, pp. 157-160, doi: 10.1109/ASPAA.2003.1285855.

[7] CMOS (Complemenary Metal Oxide Semiconductor) Definiton, TechTerms.com (2017) https://techterms.com/definition/cmos

[8] Tracking Technology Explained: LED Matching, Oculus For Developers (2019), https://developer.oculus.com/blog/tracking-technology-explained-led-matching/

[9] Stereo Vision for 3D Machine Vision Applications, ClearView Imaging (2021), https://www.clearview-imaging.com/en/blog/stereo-vision-for-3d-machine-vision-applications

[10] E. DANDIL and K. K. ÇEVİK, "Computer Vision Based Distance Measurement System using Stereo Camera View," 2019 3rd International

Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), 2019, pp. 1-4, doi: 10.1109/ISMSIT.2019.8932817.

[11] Microsoft Kinect v2 3D-Camera, FAPS - Institute for Factory Automation and Production Systems, Julian Seßner, M. Sc., https://www.faps.fau.eu/ausbio/microsoft-kinect-v2-3d-camera/

[12] Azure Kinect DK hardware specification, Microsoft Docs (2021), https://docs.microsoft.com/en-us/azure/kinect-dk/hardware-specification

[13] Raspberry Pi Documentation: Raspberry Hardware, Raspberry Pi (2022), https://www.raspberrypi.com/documentation/computers/raspberry-pi.html

[14] Raspberry Pi Documentation: Camera, Raspberry Pi (2022), https://www.raspberrypi.com/documentation/accessories/camera.html

[15] OpenCV: OpenCV modules, Doxygen, 2021, https://docs.opencv.org/4.5.5/

[16] Postel, Jon. "User datagram protocol." (1980).

[17] Xylomenos, George, and George C. Polyzos. "TCP and UDP performance over a wireless LAN." IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320). Vol. 2. IEEE, 1999.

[18] GitHub Repository, theAIGuysCode/yolov4-Deepsort: Object tracking implemented with YOLOv4, DeepSort, and Tensorflow (2020), https://github.com/theAIGuysCode/yolov4-deepsort

[19] Bittner, Jiri, and Jiri Zara. "DCGI Laboratories at CTU Prague."

[20] Reference / Processing.org, Fry, Ben and Reas, Casey (2004), https://processing.org/reference

[21] Freed, Adrian, and Andrew Schmeder. "Features and Future of Open Sound Control version 1.1 for NIME." NIME. Vol. 4. No. 06. 2009.

[22] Fette, Ian, and Alexey Melnikov. "The websocket protocol." (2011).

[23] The MagPi Magazine, Raspberry Pi 4 vs Raspberry Pi 3B+ - The MagPi magazine, (2020), https://magpi.raspberrypi.com/articles/raspberry-pi-4-vs-raspberry-pi-3b-plus

[24] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

[25] M. Noman, V. Stankovic and A. Tawfik, "Object Detection Techniques: Overview and Performance Comparison," 2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), 2019, pp. 1-5, doi: 10.1109/ISSPIT47144.2019.9001879.

[26] M. R. Fairuzi and F. Y. Zulkifli, "Performance Analysis of YOLOv4 and SSD Mobilenet V2 for Foreign Object Debris (FOD) Detection at Airport Runway Using Custom Dataset," 2021 17th International Conference on Quality in Research (QIR): International Symposium on Electrical and Computer Engineering, 2021, pp. 11-16, doi: 10.1109/QIR54354.2021.9716186.

[27] C. Kumar B., R. Punitha and Mohana, "YOLOv3 and YOLOv4: Multiple Object Detection for Surveillance Applications," 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), 2020, pp. 1316-1321, doi: 10.1109/ICSSIT48917.2020.9214094.

[28] Sozzi, M.; Cantalamessa, S.; Cogato, A.; Kayad, A.; Marinello, F. Automatic Bunch Detection in White Grape Varieties Using YOLOv3, YOLOv4, and YOLOv5 Deep Learning Algorithms. Agronomy 2022, 12, 319. https://doi.org/10.3390/agronomy12020319

[29] Nepal, U.; Eslamiat, H. Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs. Sensors 2022, 22, 464. https://doi.org/10.3390/s22020464

[30] Guo, Xiaotong, et al. "Behavior monitoring model of kitchen staff based on YOLOv5l and DeepSort techniques." MATEC Web of Conferences. Vol. 355. EDP Sciences, 2022.

[31] COCO - Common Objects in Context, COCO (2021), https://cocodataset.org/#home

[32] Leischner, Vojtěch, and Zdeněk Míkovec. "Spatial audio music player for web."

[33] M. Isard and J. MacCormick, "BraMBLe: a Bayesian multiple-blob tracker," Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001, 2001, pp. 34-41 vol.2, doi: 10.1109/ICCV.2001.937594.

[34] OpenCV: Geometric Image Transformations, OpenCV: Open Source Computer Vision (2022), https://docs.opencv.org/4.x/da/d54/group___imgproc___transform.html

[35] Matas, Jiri and Ondrej Drbohlav. Support Vector Machines, Czech Technical University, Prague (2018), https://cw.fel.cvut.cz/b211/_media/courses/b4b33rpz/pr_07_svm_2018.pdf

[36] Cambridge Dictionary, Cambridge University Press (2022), https://dictionary.cambridge.org/dictionary/english/public-address-system