Floorball Goalie Training in Mixed Reality

Johannes Gaber
Department of Mechanical and Process Engineering
ETH Zurich
jgaber@student.ethz.ch

Matthias Heyrman
Department of Mechanical and Process Engineering
ETH Zurich
mheyrman@student.ethz.ch

Annika Öhri
Department of Computer Science
ETH Zurich
oehria@ethz.ch

Abstract

This project concerns the development of an application for floorball goalkeeper training for the HoloLens 2. The application can automatically detect the goal and tracks the floorball ball and will provide additional visual cues to the goalie based on their location. This should help build their intuition and help them decide how to place themselves in order to optimally protect the goal against the attack.

1. Introduction

Floorball is a type of floor hockey where two teams, consisting of five players and one goalkeeper each, play against each other. The objective is to score goals with a perforated synthetic ball that is being moved by floorball sticks. A goalie's job is to defend the goal against these attacks. A crucial part in their training is that they learn where to position themselves and how much area they have to cover in order to protect the goal, since there is a delicate trade-off between the time it takes to move to that position, and the effectiveness of the protection. Objects such as ropes, tied between the ball and the goal corners, have been used to visualize the area at risk better and to help build the goalkeeper's intuition. Mixed Reality offers new, more flexible solutions for this.

1.1. Motivation

Floorball is the second most popular team sport in Switzerland [1], yet unlike other popular sports such as football, there is little development of technologies for floorball training, especially at a competitive level. This could be attributed to the fact that there is less economic incentive to develop specialized technology to train for floorball specifically. However, progress in mixed reality hardware development has made it possible to develop specialized floorball training software on already available and commercialized hardware, like the Microsoft HoloLens 2.

As a floorball goalkeeper is on their knees, they are not able to move as quickly and freely as the other players, and it is especially hard to match the speed of the ball. Floorball is an incredibly dynamic and quick game, and it is hence of great benefit to have a goalkeeper with good decision making. This includes them quickly and accurately recognizing how the ball could hit the goal, and consequentially adjusting their position to optimally defend the goal against these possible attacks. It could thus be beneficial to automatically provide them with explicit cues that convey this information. Mixed Reality offers a perfect environment for that, as it allows for the normal, real-time view on the game, while at the same time being able to place additional virtual objects within this world. Unlike Virtual Reality, where the game would have to be simulated to be as realistically as possible, and where only specified attacks would be possible, the fact that the HoloLens is see-through means the application can be as flexible as the game itself.

1.2. Previous Work

There are various different Virtual- and Alternate-Reality sports applications, ranging from fitness games to applications enhancing the experience of the audience of sports events [3]. There are many different advantages of these applications, such as their immersiveness, full control over the training conditions, or reducing the risk of injury or physical strain by targeted practice of e.g. just reactions. Extended Reality allows for very dedicated practice sessions and to devise more sophisticated training techniques and research, with the hope of increasing an athlete's advantage against others.

Bideau et al. have developed a Virtual-Reality application for handball goalkeepers [2]. Since this is fully virtual, they had to simulate the throws realistically enough to allow for good goalkeeper anticipation, no crucial information that is present in real life should be missing. While this is a disadvantage of a fully virtual environment when compared to the real-world view of the throw, it is also an advantage as it allowed them to examine what features are

relevant for the goalkeeper's estimation of the trajectory of the throw, and they were able to settle on a model with good correlation to the real world goalkeeper anticipation while at the same time allowing them to practice without opponents and being able to simulate specific throws.

Shimi et al. examined whether or not training goalkeepers in a more difficult task, specifically letting them catch balls at higher speeds, will improve their performance [5]. To do so, they used a Virtual-Reality application that allows to set the speed of the throws, and designed a study with five stages and two groups. Both groups had a pretest session, where their performance before having done the training, was assessed. Then, three training stages followed. One group received training of increasing intensity, each stage having a bigger speed than the last one, as well as a bigger speed than the pretest and post-test stage has. The speed of training of the other group matches the speed of the pretest and post-test stage in all three training stages. The study concluded that while both groups benefited from the training, the training with increased intensity did not create a significant advantage to the training with normal ball speeds.

1.3. Contribution

This is to the best of our knowledge the first Mixed Reality application that provides additional visual cues for floorball goalkeepers by doing goal detection and tracking the ball.

2. Approach

To create a training tool that could show a floorball goalie the possible zone-at-risk, the task was broken down into several components. Initially, a Unity environment was set up to provide all of the necessary calculations once the goal and ball positions were defined, as detailed in Sec. 2.1. After this, the computer vision tasks needed to be completed. First, the RGB camera images needed to be received from the HoloLens 2, then, the necessary image processing and maths were performed to determine the floorball's position in space, described in Sec. 2.2 and Sec. 2.3 respectively. ArUco marker detection was performed to automatically detect the goal position, detailed in Sec. 2.4. With the goal and ball positions relative to the user known, they could be translated to the HoloLens' defined world frame and input to the already developed Unity environment to then display the zone-at-risk of the ball entering the goal for the floorball goalie to use. The general flow of our application can be seen in Fig. 1.

2.1. Unity Object Setup

For the Unity setup, 2 key inputs are needed: the goal's position and orientation and the ball's position. Using these, rays are traced from the ball's position to the goal's four front corners and to its center. A projected quad object is used to represent the zone-at-risk of the ball entering the goal. The center ray is used to define a plane, which is always orthogonal to the center ray and used to calculate the positions of the corners of the quad for the zone-at-risk projection by calculating the intersections between the four goal-corner-to-ball rays and this orthogonal plane.

When running on the Microsoft HoloLens 2, the objects are all defined with respect to a world frame, meaning that if there are no position updates from any scripts, then the objects appear stationary despite the user's head movements. This simple but robust setup allowed us to move on to the computer vision tasks of determining the goal and ball positions in the world frame. These calculated positions could then finally be used to update the Unity environment objects' positions and orientations, yielding the desired visualizations.

2.2. HoloLens Camera Images

In floorball, the ball is quite small, has holes, and is often at a large distance from the HoloLens 2 moving at relatively high speeds, so the provided meshes calculated by the HoloLens 2 would not reliably detect the ball. Therefore, for the computer vision tasks of goal detection and ball tracking, we needed access to the HoloLens 2 camera array. The RGB camera was determined to be the best suited medium, as a brightly colored ball distinct from the environment is typically used in floorball, and would enable more robust methods of detection than those providing data that assists the HoloLens 2 in its mesh generation.

Other projects can often perform computer vision with just the depth and grey-scale camera on the HoloLens, which are accessible using methods like the built in Research Mode [6]. However, these methods would likely not be able to take advantage of the bright colors of the ball. Getting access to the RGB camera stream reliably with a high enough frame rate was more challenging than anticipated, and we learned only later that other projects such as one by Schott et al. [4] sometimes opt for using an external RGB camera. As the HoloLens 2 provides limited options for getting camera access, multiple options had to be explored to access the RGB camera stream.

To get access to the RGB camera stream, we tried PhotoCaptureFrame from Unity [7] first, as it is the most popular script to get access to an image from the camera, and was recommended by our supervisor. While this offered us a single image, changing the script to give continuous updating images from the dynamic scene created issues. As the images are automatically saved to memory, there is significant delay. It also consumed a majority of the HoloLens 2's CPU power, and generated at most about 3 images per second, causing frequent crashes. This also left no computational power for the necessary computer vision.

Using Unity WebCamTexture [8] instead, we were able

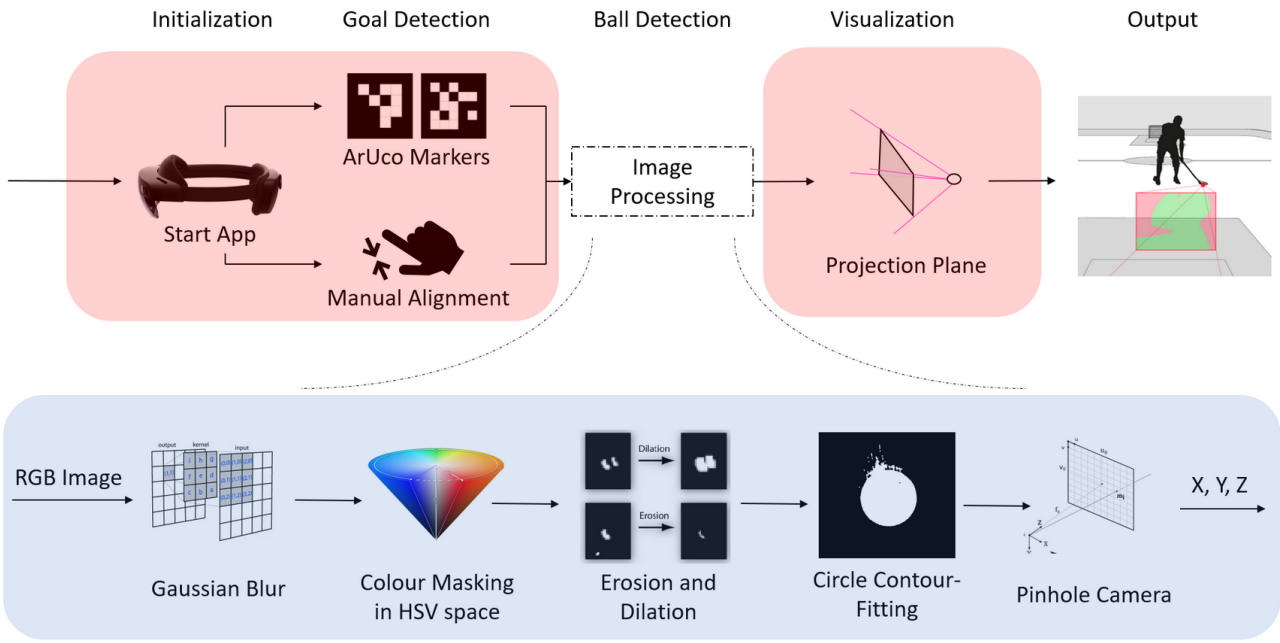

Figure 1. Application flow chart

to achieve about 5 frames per second and the script was more stable, causing no crashes from the application. However, we were not able to get the correct Camera Transformation, which is needed to properly locate objects in the world (as the HoloLens moves). This is because the camera transformation and WebCamTexture scripts both require camera access, but the HoloLens 2's software only allows a camera to be accessed from one source at a time, so accessing the transformation would stop the camera stream. The frame rate was still rather low with the image script alone.

We then used the VideoPanelApp from the HoloLensCameraStream plugin developed by VulcanTechnologies [9]. It makes the HoloLens video camera frames available to a Unity app as raw bytes in real time. As it was developed for an older Unity version incompatible with other parts of the project, we first had to search for appropriate conversions of certain commands. From this, we were able to get access to the raw Image bytes. To perform computer vision tasks on these more easily, we had to convert them to the format Texture2D. This crashed the app in release mode, however it worked well in debug mode. As debug mode is computationally limited and lowers the frame rate, a solution had to be found to run this plugin in release mode. The issue was found to be caused by the asynchronous nature of the application. Doing the image stream processing and computer vision in series in the same script was not a valid alternative, as it then resulted in the app crashing due to the higher single-thread workload. The solution to the problem was to put the conversion of the bytes into the texture into a separate thread from the original VideoPanelApp and assigning the new texture to another public texture from within that thread. This public texture can then be used by the computer vision scripts after checking that the image does indeed exist, and this prevented the scripts from crashing and finally gave reliable access to the RGB camera stream with a consistent frame rate of 30 FPS.

2.3. Ball Tracking

Once stable RGB camera stream was accessible, the ball tracking could be performed. A series of filters were placed over the RGB stream to best determine the ball's position. To do this, first a 3x3 Gaussian kernel with a smoothing amount of $\sigma = 2$ was applied to the image to provide blurring, removing some of the effect of frame tearing, occlusion (including holes in the ball), motion blur, and various other camera-based noises that could affect the final result. The image was then converted from RGB to HSV, which is better for the following masking as it defines colors through their hue, saturation, and value, which means the range we defined for our ball color was consecutive in this color space. We then produced a mask over a large part of the color space for the hue of the green ball used for testing, converting the image into a binary matrix of whether a portion of the image was within the ball's expected HSV space or not. We experimentally determined this range to work in different lighting conditions but at the same time to recognize other colors as little as possible. The binary image was then dilated with a 2x2 kernel, adding pixels around

the mask to remove the effects of slight occlusions, aggressive masking, and the holes in floorball balls, helping create full areas of masked pixels. The output is then eroded with a 2x2 kernel to remove some of the excess added boundary pixels and making the radius estimation more accurate again.

This image was then passed through a contour detector, which returns the shapes of the continuous contours within the image. As a bright, high-contrast ball with colors that should not be found in its environment was used, it is assumed that the largest contour will belong to the ball, and during testing it was typically the only detected contour. The contour with the greatest area then has a circular kernel applied at various sizes to determine the minimum enclosing circle of the contour and returns its radius and center. This is treated as the ball position and size within the image.

The focal length f of the used Microsoft HoloLens 2 RGB camera was found in pixels and could be used to calculate the distance of the ball from the HoloLens 2's camera frame using the equation

$$Z_C[m] = \frac{R_{ball}[m] * f[px]}{r_{ball,image}[px]} \tag{1}$$

where R_{ball} is the actual radius of the floorball ball and $r_{ball,image}$ is the radius we determined the ball to be in the captured image.

The ball distance can then be used to calculate the X and Y positions of the ball relative to the camera using the equations:

$$Y_C[m] = \frac{Z_C[m] * (v_i - \frac{v}{2})[px]}{f[px]} \tag{2}$$

$$X_C[m] = \frac{Z_C[m] * (u_i - \frac{u}{2})[px]}{f[px]} \tag{3}$$

where u_i and v_i are the x and y coordinates positions of the detected ball center within the image and they have half of the image width u and height v respectively subtracted as the ball's position needs to be calculated from the camera frame behind the image center, but the image's coordinate frame starts from the top left. This provides the floorball's X, Y, and Z coordinates in meters with respect to the HoloLens 2 camera frame. This must then be transformed from the camera frame in Unity to the HoloLens 2 world frame to take into account translational and rotational motion of the user's head. The transformation from the world frame to the HoloLens 2 camera needs to be accessed, and the position of the ball relative to the user can be transformed to the world frame using the equation:

$$P_W = T_{WC} * P_C \tag{4}$$

where T_{WC} is the transformation matrix from the world to the camera, and P_C denotes the position in camera coordinates while P_W denotes the position in world coordinates. As Unity uses a left-handed coordinate system and computer vision is standardized in right-handed coordinates, the ball's 3D world coordinates relative to the camera had to be shifted to left-handed coordinates first.

While this version gave us reasonable tracking, the estimation of the ball's position was still unstable. We attributed this mostly to the distance estimation based on the radius, due to multiple factors. For one, the ball is not very big, and if it is far away from the camera, few pixels in the camera make a big difference in the actual distance, it is thus very susceptible to small artefacts. Also, at the border of the ball, the masking is more likely to produce wrong results, as the background might influence the ball's color more there, thus the center is more reliable than the radius. Finally, occlusion plays a big role. Even though we employed multiple steps such as blurring and dilation to reduce the impact of occlusion, it is still a present problem, especially in the real-life setting of a game, where a player and their stick are likely to cover parts of the ball, making it seem smaller than it actually is. To avoid this causing the ball's distance estimation to be noisy, impacting the user experience, additional constraints were added to allow the ball position to be more independent of the radius estimation. Specifically, the ball was constrained to be the floor. With this additional constraint, an additional equation is gained, and we can estimate the Y position based on this instead of the radius, creating a more smooth result.

This produced stable as well as accurate tracking, given the ball is indeed on the floor. Usually, this is the case at the beginning of an attack on the goal, which is the moment the displayed projection is most crucial for the goalkeeper. Still, we have also added the option of disabling the constraint, should that be desired, see Sec. 2.5.

Note that more advanced algorithms for ball detection would be available, such as using the Hough transform for circles, or neural networks. We instead opted for this simple approach as the HoloLens computing capabilities are limited. While this simple script easily runs in real-time on our laptops, it already creates significant delay on the device itself, where the other mentioned algorithms would take even more computational effort. With the additional constraints, the results were already quite accurate, so we decided to favor performance instead.

To increase the frame rate, we down-sampled the RGB camera texture, which is originally 896x504. With the original resolution, we were only able to achieve a frame rate of about 3 FPS. With the new constraints, we were able to down-sample the texture without impacting the tracking, achieving an average of 15 frames per second on average. This allowed for fluent, dynamic tracking, even if not fully in real-time.

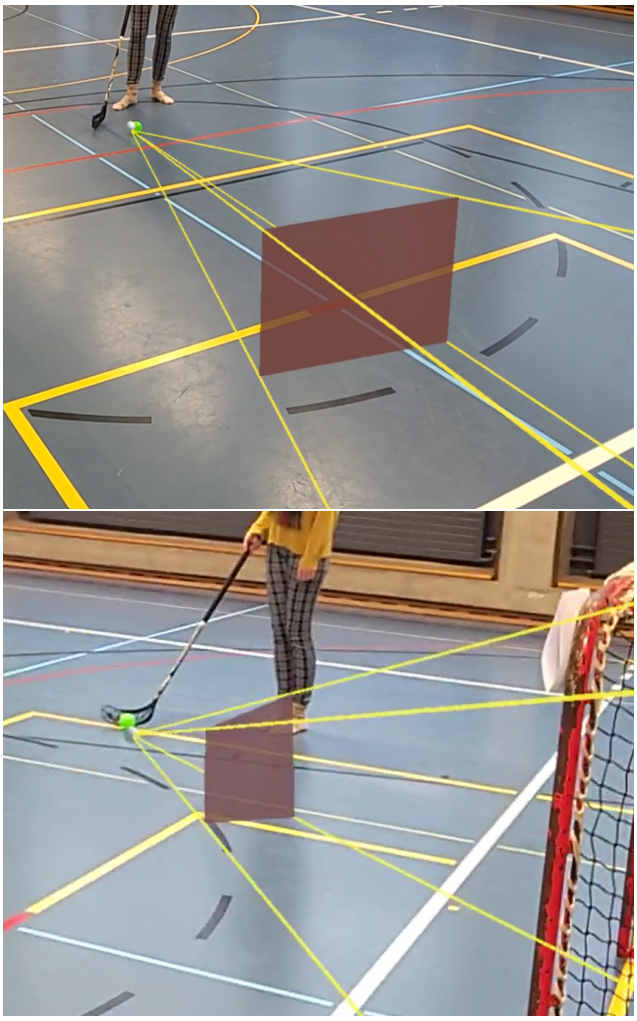

Figure 2. Ball Tracking in use, virtual objects adapt their position at about 15 frames per second.

2.4. Goal Detection

To calculate the rays between the ball and the goal corners to then project the zone-at-risk, the exact goal position needs to be known. Since the goal is stationary during play, it is enough to determine the position during the initialisation of the application. The position is saved in the world frame of the Hololens, so the spatial locator automatically keeps track during head movements.

To determine the position, different possibilities were discussed in advance. A very user friendly option would be to track the penalty box of the goal, so the scanning could take place while already looking at the ball. The main disadvantage of this option is that it is highly prone to changes of the environment. Fields can look different, with different colored lines on the ground, and also might get intersected by lines from other sports, so the detection would must take all these possibilities into account. The user might also want to train in an environment without floorball lines drawn, which would be goal detection impossible using this option.

To avoid this sensitivity to the environment, tracking ArUco markers is a good alternative to still support an automatic detection of the goal. Two ArUco markers are used in the upper goal corners, to stably track the position and angle alignment. This method works for all goal types and the user benefits especially in repetitive training sessions, where the ArUco markers stay on the goal, so just a short look at the goal is necessary for detection and localization, which can be seen in Fig. 3. To not be dependent on having markers and further account for detection errors, a manual goal alignment feature was also added. A virtual object with the MRTK manipulation handler is meant to be aligned with the goal by the user, to extract the goal position.

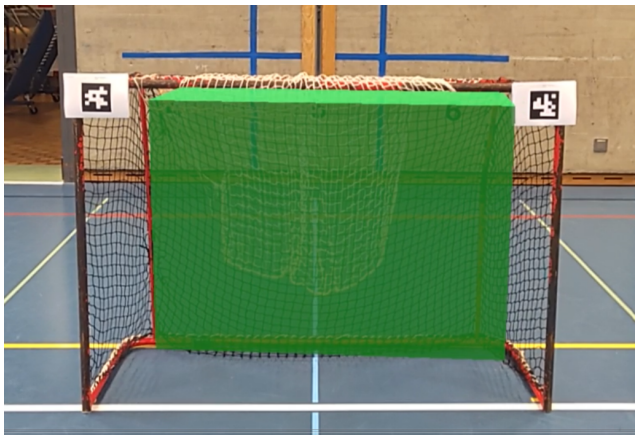

Figure 3. Fully automatic alignment of virtual goal, whose position is visualized by a green cuboid, position to real goal position using ArUco markers.

2.5. User Interface

To control the application and switch between different modes, a menu was necessary. The MRTK hand menu provides all important features and due to its hand following approach, is always available to the user. Since floorball goalkeepers usually do not have their open palm of hand visible to the camera, no conflicts during training will arise. In the final menu, the user can chose between the two goal detection modes, can switch off the visibility of the virtual goal, the ball and the connecting lines between ball and goal, switch from ball tracking to manual ball positioning to visualize certain game situations and also can switch off the constraint of the ball to the floor.

2.6. User Study

We have consulted five recreational floorball players to test out our application and give feedback. Most of them complained about jittery ball tracking results, which was

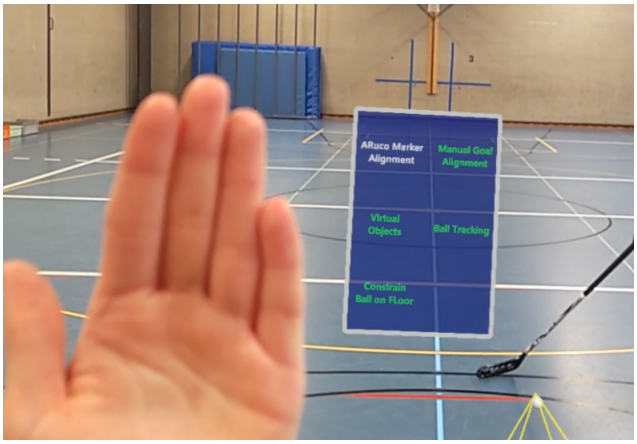

Figure 4. Hand menu, allowing the user to choose between multiple modes and visualization options

distracting them as the virtual objects jumped around to it. This is why in a second iteration, we added the option of constraining the ball to the floor, creating a more stable environment and reducing this impact. We also added the option of not displaying the lines anymore, as for some, this was also more distracting. Another general complaint was the delay in the floorball tracking results. To compensate for this, the processed image was further down-sampled to reduce the number of required calculations, but as this reduces tracking accuracy, a balance had to be chosen between delay and accuracy. Two asked users suggested the initial UI placement, which was set to be next to the goal as a known accessible location, was not ideal if they wanted to try out different options during the training, as that involves having to leave the goal again. From this suggestion, the final hand menu was implemented instead that appears when the user lifts their hand up with their palm facing their face, seen in Fig. 4.

3. Conclusion and Outlook

3.1. Summary

We have created an application that provides additional visual cues for floorball goalkeepers in the hopes of improving their training and better building their intuition by providing explicit visualizations of the area in danger. This was done by computing a projection between the position of the ball and the goal's corners. The goal's corners can be automatically set by using ArUco markers that can simply be attached to the goal, or alternatively the goal can also be moved manually. The ball is automatically tracked, giving stable and accurate results at a good frame rate of about 15fps, which is not fully real-time, but does not have noticeable delay when wearing the glasses in most cases. The application has an easy-to-use user interface that allows the user to switch between different modes such as automatic or manual goal alignment, and also offers other options such as disabling the ball tracking, allowing for a fully virtual environment to test out specific attacking positions by manually moving the ball.

3.2. Limitations and Future Work

The application is limited to a certain ball color, but it could in the future be extended to allow the user to change the ball color. The tracking relies on the color of the ball being unique in the environment, which is another limitation, however in floorball, the ball with the best contrast to the floor is chosen, as that helps also the players' recognition of the ball, so this should under normal circumstances be given. There are also some technical limitations of our application due to the device and framework that we used. One issue is that when opening the Windows menu from within the app, the HoloLens blocks the subsequent accesses to the RGB camera, so our computer vision tasks can no longer function. Another issue is that the HoloLensCameraStream we used needs some time to initialize and determine the Unity application's world frame, which can offset it from the HoloLens 2 world frame. Therefore, the user should keep their head still when starting up the app, as any unregistered movement at the beginning may result in an offset of the virtual objects.

There are also some exciting opportunities for future work, such as increasing the performance of our tracking beyond our sub-sampling approach, to allow for real-time tracking. As mentioned in Sec. 2.3, the HoloLens was not able to produce effective real-time tracking, however, real-time tracking works on more powerful devices such as laptops and computers. Therefore, doing remote computing and sharing the results between the device and the laptop would be an alternative to improve performance assuming that this sharing is faster than the computation.

It would also be interesting to further increase the scope of the application. The project was meant for floorball goalkeeper training only, however there would also be an opportunity to further extend the application of Mixed Reality to other players, and think of helpful cues to give to them during the game. As floorball is very dynamic, fast-paced and complex, this is beyond the scope of what was possible in the available time for the project, however we hope that this can be picked up by others in the future.

References

[1] Floorball, one of the most popular sports in switzerland, Apr 2019. 1

[2] Benoît Bideau, Franck Multon, Richard Kulpa, Laetitia Fradet, and Bruno Arnaldi. Virtual reality applied to sports: Do handball goalkeepers react realistically to simulated synthetic opponents? In *Proceedings of the 2004 ACM SIG-*

GRAPH International Conference on Virtual Reality Continuum and Its Applications in Industry, VRCAI '04, page 210–216, New York, NY, USA, 2004. Association for Computing Machinery. 1

[3] Andy Miah, Alex Fenton, and Simon Chadwick. *Virtual Reality and Sports: The Rise of Mixed, Augmented, Immersive, and Esports Experiences*, pages 249–262. Springer International Publishing, Cham, 2020. 1

[4] Danny Schott, Florian Heinrich, Lara Stallmeister, and Christian Hansen. Exploring object and multi-target instrument tracking for ar-guided interventions. *Current Directions in Biomedical Engineering*, 8(1):74–77, 2022. 2

[5] Andria Shimi, Kleanthis Neokleous, and Marios Avraamides. Athletic performance in immersive virtual reality: The effect of training intensity. *European Journal of Psychology Open*, 81:24–33, 05 2022. 2

[6] Dorin Ungureanu, Federica Bogo, Silvano Galliani, Pooja Sama, Xin Duan, Casey Meekhof, Jan Stühmer, Thomas J. Cashman, Bugra Tekin, Johannes L. Schönberger, Bugra Tekin, Pawel Olszta, and Marc Pollefeys. Hololens 2 research mode as a tool for computer vision research. *arXiv:2008.11239*, 2020. 2

[7] Unity. Photocaptureframe. documentation: https://docs.unity3d.com/550/Documentation/ScriptReference/VR.WSA.WebCam.PhotoCaptureFrame.html. 2

[8] Unity. Webcamtexture. documentation: https://docs.unity3d.com/550/Documentation/ScriptReference/WebCamTexture.html. 2

[9] Vulcan. Hololenscamerastream for unity. github: https://github.com/VulcanTechnologies/HoloLensCameraStream. 3