Training Archery Form Using Augmented Reality

CSE 493V Final Report

MATTHEW HE, University of Washington

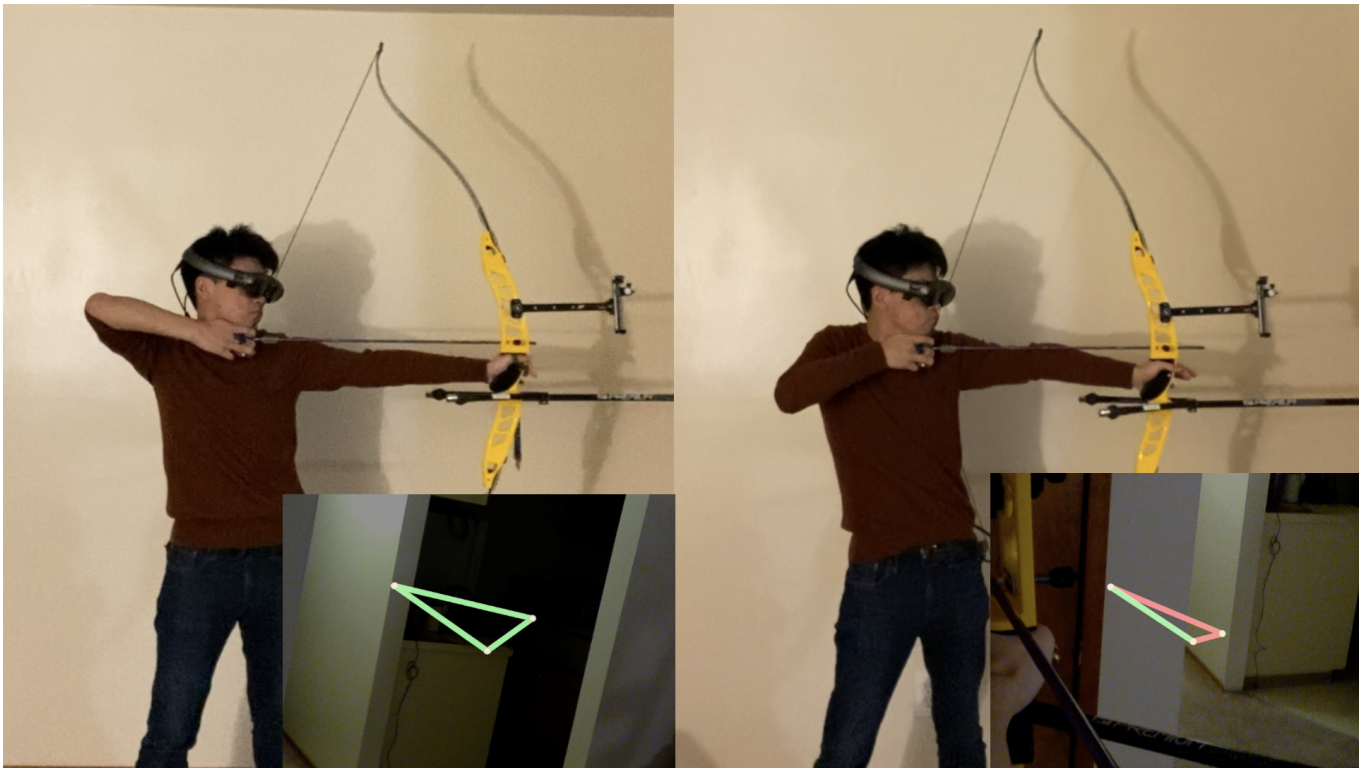

Fig. 1. While shooting, Microsoft Kinect feeds full-body data to

Mastering archery form requires countless hours of practice. Archers often develop bad habits in their form, such as misaligned forearms or high shoulder placement. Without additional tools or help, archers cannot improve their form while shooting. A common approach used by most archers is to film themselves shooting and watch the replay, using that to inform their next round. Other archers rely on coaching to hone their form. Using augmented reality, it's possible to provide real-time analysis of an archer's form as they shoot, telling them if their form is off and letting them correct before they commit shots to muscle memory.

1 INTRODUCTION

In archery, an archer cannot watch their form while drawing and aiming the bow. While an archer can feel physical stimulus from the drawing motion, it's difficult to discern proper form without visual aids. Thus, a common approach for archers is to record and replay videos of their shooting, so that they can improve their future shooting. This, however, does not eliminate the lack of visual indication of form while shooting. Without a coach, many archers will struggle to improve their form even in this manner.

Author's address: Matthew He, mhe9467@cs.washington.edu, University of Washington.

Augmented reality (AR) provides a solution to this problem. AR systems provide additional information and context to users that might not normally be available to them. Additionally, AR headsets provide a hands-free method of delivering such information. Thus, AR headsets provide a platform to which visual analysis of an archer's form from a third-person perspective can be fed to them as they shoot.

As far as my research has gone, no AR application currently exists to train the archer's form as they shoot. In a 2015 journal, an AR trainer providing three views of the archer from three sides was proposed, but no system was ever developed. Many AR archery games exist, but none provide realistic form training in the same scope. I believe this is mainly due to a lack of demand or exploration in the space. Archers at higher levels have coaches who provide analysis backed by experience and knowledge. Furthermore, advanced video analysis rigs exist that allow for coaches to provide multiple view analysis to archers after the shot.

In this project, I developed an AR application which displays key points of an archer's form as they draw so that they can adjust their form as they draw the bow. As shown in **Figure 1** above, three red

lines are drawn between three key points in the archer's form: bow hand, draw elbow and chest. As the archer draws through their form, the red lines turn green to indicate that that portion matches the proper form. When the entire triangle turns green, the archer has drawn fully into the correct form.

Ultimately, this tool is a solid proof of concept for more advanced AR archery training systems. However, as discussed later in this paper, various limitations of both AR and archery reduce the overall usefulness of this for most archers. This tool is most useful for beginner archers looking to improve consistency in form.

1.1 Contributions

- I developed an algorithm that scores the user's form against a preset form skeleton.
- I also developed an algorithm to correct the noise from inferred positions and extrapolate the hand position using other tracking points.
- Finally, I built on the head-locking feature in Unity to display the body tracking points into a bounded space and scale them to fit an abnormal viewing angle.

2 RELATED WORK

In 2015, a small section about AR archery found its way into a sports science journal. As shown in Figure 2, the proposed AR archery system would utilize three cameras to provide real-time footage to the archer about their form. While this system was never actually completed, the core idea is similar to the idea behind this project. In this project, we utilized 3D body tracking to analyze the archer's form. This takes the idea a step beyond the three camera solution, where the archer needs to monitor the footage as they shoot. Instead, my application analyzes the user's form and provides important information directly to the user.

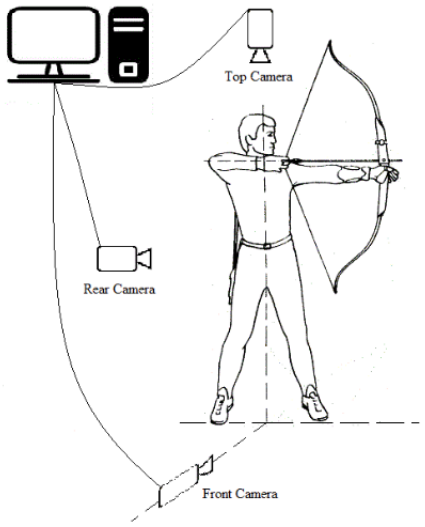

Fig. 2. A schematic showing a proposed design for an AR trainer for archers [2015].

3 METHOD

In the process of developing this application, there were three main problems to solve in order to put together the product. First, I needed to get full body tracking data in order to do initial form calculations. Second, I needed to send the tracking data to the application, which would then do any necessary calculations or processing. Finally, the application would need to render any visuals in the augmented reality space.

In terms of body trackers for archery training, we need a few key trackers from full body data. Generally, the six most important points to track are the two elbows, two shoulders and two hand. For the rest of the paper, we'll assume right eye dominance, so the bow hand will be the left hand and the draw hand will be the right hand. In proper form (Figure 3), the entire left arm and body will draw a straight line. That is, the right shoulder, left elbow and left hand should be in a line. Furthermore, the right elbow, right hand and left hand should form another line.

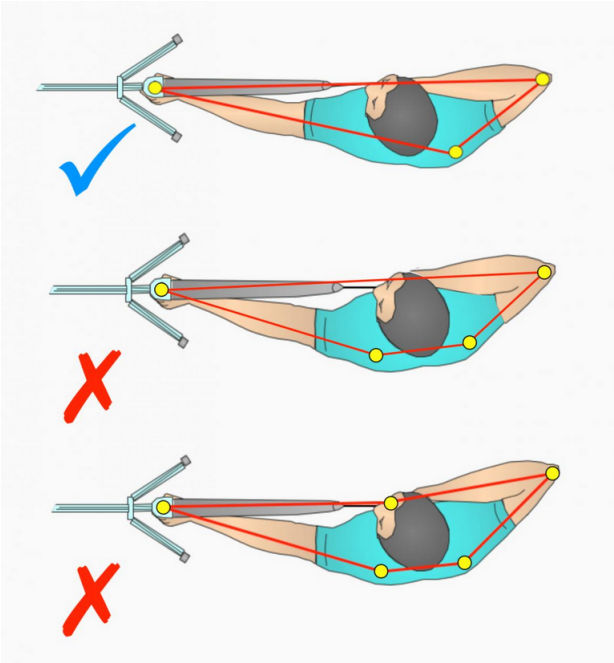

Fig. 3. A diagram showing proper alignment in the archer's body.

So with this information, we need a set of six trackers to assume the proper form tracking. Additionally, we can add hip, knee, and foot trackers to check that the archer has consistent anchoring. In our implementation, these tracker positions will be given in world coordinates from the Microsoft Kinect sensor (see Implementation Details).

After retrieving our tracker positions, we need to determine what we consider proper form. We can simply record the tracker positions in what we believe to be proper form. From the proper tracker positions in world space, we compute the vectors to form the triangle seen in Fig 3. We treat this set of vectors as the proper "form" in the

application. Since the user may be closer or further from the sensor, we'll also normalize these vectors.

When the application measures position as the archer draws, it collects a new set of position data. We again compute the vectors and normalize, but we pin the center. To determine if the archer's form is good, we calculate the percentage difference between the proper "form" and our newly measured set of form vectors. If the deviation falls below some threshold, say 5%, the archer's form is considered valid.

Finally, we need to render the archer's form in the augmented reality space. More details can be found in the upcoming section. Since we're given the tracker positions in world space, we can use these positions to render the locations of the trackers in AR and connect them with lines. However, we want to render these trackers such that they follow the camera space, instead of staying stationary in world space. We can implement this with a series of rotation matrices if we're given the camera/headset orientation.

4 IMPLEMENTATION DETAILS

For the AR headset, I used the Magic Leap 1 headset, which uses it's own Lumin OS. The Magic Leap 1 has Unity support, which I used to write the entire application. The entire Unity application was built using the Magic Leap package and default assets. Since the application requires full body tracking, which is a feature that Magic Leap 1 can't provide, I needed an external body tracking solution.

To get the body trackers, I utilized Microsoft's Xbox 360 Kinect. The Kinect can be utilized to get full body position data and is supported by the Windows Kinect SDK. Furthermore, there already exists Unity packages that support the Kinect SDK. The Kinect is connected to a Windows computer over USB. I overlooked this requirement when writing my project proposal. It was difficult to find Mac support for Kinect, so I needed to use an old Windows laptop for Kinect development. Furthermore, this direct USB connection meant that it would be impossible to connect the Kinect straight to the Magic Leap.

To get the tracking data from the Kinect, I used Amethyst, an application built to get Kinect full body tracking to VR head sets for use in VRChat or similar applications [2022]. With the Kinect connected, Amethyst computes the body tracking skeleton (Figure 4). Amethyst also provides support for the OSC protocol, a network protocol that can be used to send tracking data to a target device over the local network.

Using the extOSC package in Unity, we can collect the OSC packets being sent to the application by Amethyst. These OSC packets contain both position and rotational data of knee, hip, chest, elbow, foot and head trackers in the skeleton. This shows a major limitation of using Amethyst. Since Amethyst is built to support VRChat, it uses the VRChat OSC definitions. As a result, four critical trackers (hands and shoulders) aren't provided by Amethyst. However, since no similar applications could be found that were compatible with 360 Kinect, I had to address this limitation in software.

Using OSC, the Unity application now has access to nine different trackers. However, the hip and foot trackers were too inconsistent to use in the application. Furthermore, the proper form for the lower body can't easily be approximated with these simple trackers. Thus, we're mainly concerned with the chest, elbow and head trackers. Since we don't have access to hand or shoulder trackers, we have to approximate their location from the other trackers. Using the center of a 90° horizontal sweep of the right elbow, I was able to approximate the location of my right shoulder. This approximation only works since I assume that my chest is parallel to the Kinect sensor. I also roughly approximate the location of the left hand by extending the position of the left elbow. Again, this requires the assumption that the fully extended left arm will be in a straight line (as it should in proper form). Finally, we use the head tracker to approximate the location of the right hand, since we know that the right hand must be touching the face when correctly anchored. Thus, despite limited tracking data, we were able to extrapolate the trackers required for form tracking (with some simplifications).

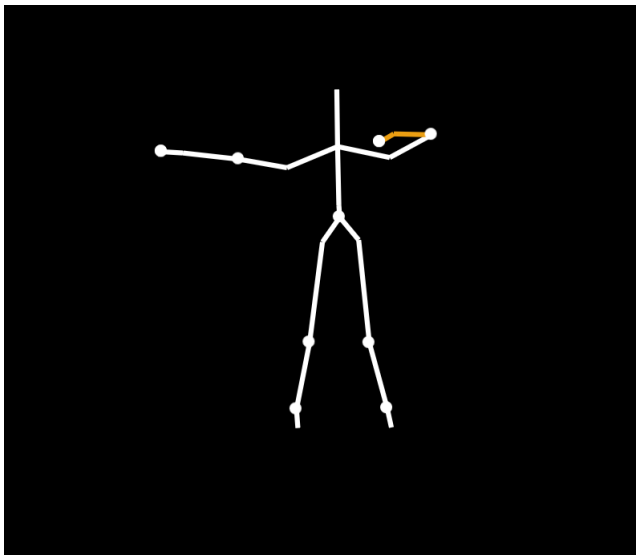

Fig. 4. An example body tracking skeleton generated by Amethyst.

To render the form visualization in the Magic Leap, we simply render the points in Unity and connect them with lines. There are two key implementation details here. First, to get the correct color, we record the positional data of the ideal "form" and implement the algorithm described previously to check percentage deviation. If the alignment is too far off, we set the color of the line to red. We also set an OSC Event Receiver to update the rendered trackers when the Kinect computer sends updated positional data. This leaves one final issue to solve. In the Magic Leap Unity application, the trackers are rendered in world space. However, as the user moves around and turns, the camera is not necessarily near or looking at the world space center. Thus, it becomes necessary to lock the rendered images to the user's view. Thankfully, Magic Leap provides an example implementation, which we can borrow to lock the rendered images to the camera. This involves a set of rotations and translations, all of which are already defined for us. The only change we need to do is update the deprecated libraries to match the current version of the Lumin SDK.

5 EVALUATION OF RESULTS

Overall, the project was successful in providing an example of how AR can be utilized in archery. My application is successful in identifying correct form versus bad form and displaying it to the user. However, the idea of "correct form" is manually defined and may vary from user to user. Users unfamiliar to archery may not be able to set the "correct form" themselves. In the same vein, the missing trackers from Amethyst necessitate manual calibration to guess the locations of the remaining trackers. This simplification reduces accuracy and relies on assumptions that may reduce the usefulness of the project.

In fact, the majority of limitations of this project stem from the limitations of the full body tracking method. The limited trackers stem from Amethyst, of which the intended applications have built in solutions or do not require the missing trackers. I found that it was out of the scope of this project to develop a complete system to get body tracking data from the Kinect and send it over OSC to the Magic Leap. As a result, I had to work with these limited trackers.

During testing, I found various limitations that have to do with the application of AR to archery in general. As shown in Figure 1, when correctly drawn, neither the bow or sight is in frame of the Magic Leap viewer. In fact, the right eye cannot see through the sight because it is blocked by the headset. This hardware limitation limits the usability of the system because it makes training difficult, which is the opposite of the intended use. Software workarounds could be implemented, but I don't think they would be useful. In general, at full draw, the headset blocks line of sight from the dominant eye to the target.

The other major limitation of this type of form analysis is that it is limited to form during the drawing process. Much of the archery form lies in the release and follow through, both of which are not easily provided nor captured by an application like this. AR is not necessary for post-release analysis, which may be better provided by video analysis with a coach. For intermediate and more advanced archers, issues in form are likely to be in the release and follow through, not the initial draw, thus limiting the usability of this type of application for non-beginner archers (although it looks cool).

6 FUTURE WORK

Ultimately, I don't think there's much that can be done about the limitations of the problem itself. Even in improving the application for beginning archers, it is reliant on headset improvements that provide higher fields of view. The archer's head angle makes any AR headset applications because the rendering frame lies mostly in peripheral vision. Aside from these limitations, if this project were to be improved, it would start from improving the body tracking solution. Instead of the Amethyst limited trackers, a solution could be developed to send all necessary trackers over OSC instead. This would improve the solution accuracy and remove reliance on assumed tracker positions.

7 CONCLUSION

This project is a proof of concept for archery form analysis in augmented reality. As an application, it provides a platform for helping beginning archers understand basic shooting form, especially when they cannot visualize the correct form while shooting. However, various hardware limitations limit the actual usability of the application, as well as its accuracy. Due to these limitations, it is difficult to imagine further developments that can improve the usefulness of such an application for more advanced archers.

ACKNOWLEDGMENTS

Thanks to the CSE 493V staff for providing the necessary hardware resources and for pointing me in the right direction when choosing the proper development hardware and software.

REFERENCES

Zafer Bozyer. 2015. Augmented Reality in Sports: Today and Tomorrow. In *International Journal of Science Culture and Sport (Special Issue 4)*. International Journal of Science Culture and Sport, 322–323.

K2VR Team. 2022. Amethyst Docs. https://docs.k2vr.tech/en/