Augmenting A Mountain Biker's Reality

John P Cavalieri
Umass Lowell
John_Cavalieri@student.uml.edu

ABSTRACT
This article presents an augmented reality platform for mountain bikers. The system is embedded in a full-face helmet with three types of augmented reality overlays projected onto the visor. Arrow markers positioned onto the trail for navigation, rider tracking to identify other riders in the trail, and simulation of hitting approaching jumps with user's riding characteristics and projecting a green or red overlay on the jump incline to indicate whether simulation results in successful landing or not. Overlay projection require user's gaze to be fixated on the trail so that the overlay is not obscuring peripheral vision—this is achieved via an eye tracking system incorporated into the AR helmet.

Author Keywords
Augmented Reality; Eye Tracking; Mountain Biking; Terrain Mapping; GPS

INTRODUCTION
There exist many use cases for augmented reality. In [3], a prototype AR museum guide provides users with AR meta-information for each museum display. The system tracks user's eye movements to detect fixations/gazes on a museum display. Upon gaze detection, the image from scene camera is piped into the object recognition pipeline. The pipeline recognizes the display and provides appropriate AR content to the user. Eye tracking is achieved via a dark-pupil system where the eye is illuminated by an IR LED. An IR camera watches the eye from a fixed position relative to the head. The face reflects the IR light emitted by the LED, but the pupil mostly absorbs IR resulting in a high-contrast dark ellipse that is easily identified by image analysis software [3]. The center of the pupil is calculated and mapped to a gaze position via eye-tracking algorithm. The object recognition pipeline requires a database of SIFT (scale invariant feature transform) extracted features from each museum display [3]. This database is populated by scanning each display from multiple views, extracting SIFT features and storing these labelled features in the database. As the user fixates on a display, SIFT features are extracted from the scene camera and those features are matched to features in the database using approximate nearest neighbor search [3]—a search algorithm that is chosen to spare computational resources. This sort of trade-off is not necessary with the advent of Qualcomm's neural net processor—a processing unit for mobile platforms designed to handle computationally intensive machine learning tasks like object recognition; the mountain biking AR system described in this article requires the neural net processing unit to handle the demands of detection and tracking in a highly dynamic environment.

A fascinating application of AR is attempting to empower human decision-making process by offloading certain analysis tasks to the CPU and utilizing an AR overlay to return those results to the user just-in-time to make a decision. The offloaded tasks are those better handled by a CPU than the human brain e.g. physics calcuations. This sort of application is described in [1] where a user's potential actions are analysed and consequences for each action are predicted and visualized with AR overlay. For example, consider the simple task of stacking blocks up to a certain height and then manipulating blocks within the tower without toppling it [1]. Using object tracking and a physics engine, the system understands the configuration of the blocks and its underlying kinematics. The user iterface allows the user to indicate a potential action e.g. perturbing the k'th block, and the system computes the consequences of this action—topple or not topple—and returns an AR overlay of the consequences to the user [1].

AR PLATFORM
This article intends to layout the framework for integrating AR into the mountain biking world. Many AR applications require a head-mounted display e.g. HoloLens and Google Glass. These technologies are far too cumbersome for a mountain biker to wear on the trails—not to mention it would prevent the rider from wearing a helmet. To address this issue, I have proposed embedding the AR system into a full-face helmet with accompanying processing unit contained in a backpack.

Fig. 1

The helmet shown in figure 1 is a full-face helmet that could contain the AR system. The overlay would be projected onto the visor and a wired connection would come out the back of the helmet and connect to a backpack where the processing unit is located. The AR system within the helmet consists of an eye-tracking system for gaze detection in addition to a scene camera for visual natural-feature tracking and an inertial monitoring unit (IMU). The combination of scene camera and IMU provides hybrid visual-sensor tracking and

achieves better performance than visual or sensor tracking individually [5]. This type of tracking is necessary for mountain bikers since visual tracking alone is subject to unrecoverable loss of tracking due to fast camera movements [2].

The scene camera and sensors within the helmet will track the trail in the user's field of view so that navigation markers and warning markers can be overlaid directly on top of the trail. The overlay will only be projected when the user's eyes are fixated on the trail to prevent obscuring peripheral view. An example overlay is shown in figure 2.

Fig. 2

Before the trail can be tracked, it must first be detected. For detection, I propose an off-line deep learning system like [4]. The off-line step consists of training a deep neural network on a subset of mountain bike videos on YouTube; during training, the algorithm learns which features to extract from videos of trail rides. At runtime, the same features will be extracted from the scene camera within the helmet to detect when the user is riding down a trail. Once a trail is detected, the tracking algorithm takes over and maintains correct registration between real-world coordinates and virtual-world coordinates as the user travels down the trail. The alignment of world coordinate systems ensures proper registration between virtual overlay and real-world [5]. Improper registration will result in an overlay like that in figure 3.

Fig. 3

Faulty tracking results projecting the overlay off the trail—this type of improper alignment is called registration error [7].

The detection algorithm will also learn to detect other riders on the trail. Assuming internet and GPS connectivity, the system will also identify the name of the rider and the AR will provide an overlay like that in figure 4.

Fig. 4

The most complex feature of the mountain biking AR system described in this article is the jump prediction and visualization system. Inspired by [1], the system relies on GPS to determine the location of jump within the trail—this could be achieved by geo-tagging each jump—and terrain mapping to provide characteristics of the jump such as incline, gap, and landing. The terrain and GPS maps would exist in an online database and could be loaded onto the AR processing unit before riding trail. The physics simulation also requires the mass of the bike + user system which would be manually entered by the user and instantaneous velocity which would be provided by GPS. Given these inputs, the physics engine simulates the jump and determines whether current riding characteristics—speed, angle of approach, and so on--will lead to successful jump. As the jump is approached, the simulation would update in real-time to provide user with current best information—allowing the user to adjust riding characteristics if needed. So long as the simulation indicates success, the AR overlay will outline the jump in green. If simulation results in failure then the jump is outlined in red as shown in figure 5.

Fig. 5

DETECTION & TRACKING

Registration is the most important problem in AR and can be described as aligning the world-coordinate system of the scene and virtual cameras [6]. The scene camera is the camera that records the real-word in front of the user—this camera is used for visual tracking only. The virtual camera is a component of the computer graphics system that renders virtual objects on the user's helmet visor. To understand the importance of accurate registration, compare figure 3 and figure 2 showing misaligned navigation markers or consider a chessboard where the virtual chess pieces are misaligned with the chessboard making the game unplayable.

AR tracking involves two major steps. First, the target is detected in the incoming video stream recorded by the scene camera using a detection algorithm [4]. This detection step yields the initial pose of the virtual camera with respect to the target and this data is used to initialize the tracking algorithm [4][2]. The tracking algorithm continues to update virtual camera pose until the target is no longer present in the incoming video stream. At this point, the detection algorithm resumes analyzing the incoming video stream until another target is detected [4].

As described in the introduction, the detection algorithm requires an off-line step. This is where the algorithm learns what features best describe trails, jumps, and other riders on the trail—all of which must be detected before they can be augmented by the AR system [4]. By training on YouTube videos of mountain bikers riding trails, the algorithm sees a variety of different trail types and thus learns to generalize so that the user can ride a trail not seen during training and still enjoy AR augmentation.

The AR platform described here relies on a fusion of visual and sensor based tracking [5][6]. Visual tracking accurately tracks the motion of the camera in a 3D environment and dynamically estimates the location of the camera [5][2]. The need for tracking can be understood by considering the AR chess game with a real chessboard and virtual chess pieces. The virtual pieces should remain aligned to the chessboard as I walk around the chessboard—without tracking the pieces would float all over the place as I move within the environment. A naïve approach is to always project the overlay on the center of the head-mounted display. In the chess example, this naïve approach fails if the user looks away from the chessboard and still sees virtual chess pieces.

Visual tracking breaks down into two categories: marker and marker-less based visual tracking [5][7]. Marker based visual tracking relies on fiducial markers placed within the environment that are easily identified in the incoming video stream and used to determine camera position within the environment [7]. This type of tracking is accurate and robust to jitter but suffers major drawbacks [5]. It is sensitive to occlusion—if the marker is occluded then the AR system has no idea where the camera is within the environment and any AR overlays will incur massive registration errors [7][5]. If I chose to employ marker-based visual tracking to track jumps on the trail then each jump would need a marker plastered on the incline as shown in figure 6.

Fig. 6

Marker-less based visual tracking encompasses several different methods, but recently the focus of much research is the hybrid visual tracking approach that incorporate sensor tracking [5]. Marker-less visual tracking amounts to maintaining statistical state of target object within an extended Kalman filter with assumed Gaussian errors [5]. For each frame of incoming video stream, the probabilistic estimate of target position based on Kalman filter is generated and compared with actual position of target within video stream [5]. The difference between actual and predicted is used to infer the motion of the camera and hence the location of the camera relative to the target [7].

JUMP SIMULATION AND VISUALIZATION

Inspired by the work done in [1], the AR system described here attempts to simulate hitting jumps on the trail and return a visualization to the user to inform his/her riding style on the approach to the jump. The goal of this type of system is to empower the user's decision-making process. To achieve this, the system must understand the set of potential actions a user may take in a situation [1]. For this application, the set of potential actions is 1) whether to hit or not hit jump and 2) how to adjust riding style on approach to jump. Given this set of action potentials, the system must simulate hitting the jump with under current riding characteristics and generate a visualization of the result for the user—in this case, a red or green outline of the jump indicating likely outcome. I have assumed the existence of GPS mapped trails and terrain mapped trails but I will not go into detail because they are beyond the scope of this article. The GPS position of the jump is known and this allows the AR detection algorithm for the jump to kick in at the appropriate time so the simulation results can be visualized as soon as the jump enters the video stream.

CONCLUSION

This article describes a platform for AR mountain biking. By integrating the AR system within a full-face helmet and placing the processing unit in a backpack, I have avoided the

cumbersome head-mounted displays that typify current AR devices while ensuring that the user is not forced to choose between a helmet or AR. Additionally, I have outlined the technologies necessary to achieve AR overlays for navigation, rider tracking, and reporting back jump simulation results to the user.

REFERENCES

[1] S.-w. Leigh and P. Maes, "AfterMath: Visualizing Consequences of Actions Through Augmented Reality," in *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, New York, NY, USA, 2015.

[2] J. R. Rambach, A. Tewari, A. Pagani and D. Stricker, "Learning to Fuse: A Deep Learning Approach to Visual-Inertial Camera Pose Estimation," in *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2016.

[3] T. Toyama, T. Kieninger, F. Shafait and A. Dengel, "Gaze Guided Object Recognition Using a Head-mounted Eye Tracker," in *Proceedings of the Symposium on Eye Tracking Research and Applications*, New York, NY, USA, 2012.

[4] O. Akgul, H. I. Penekli and Y. Genc, "Applying Deep Learning in Augmented Reality Tracking," in *2016 12th International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*, 2016.

[5] G. S. W. Klein and T. W. Drummond, "Tightly integrated sensor fusion for robust visual tracking.," *Image and Vision Computing*, vol. 22, pp. 769-776, 2004.

[6] D. P. Kaur and A. Mantri, "Computer vision and sensor fusion for efficient hybrid tracking in augmented reality systems," in *2015 IEEE 3rd International Conference on MOOCs, Innovation and Technology in Education (MITE)*, 2015.

[7] G. Klein, Visual Tracking for Augmented Reality, 2007.