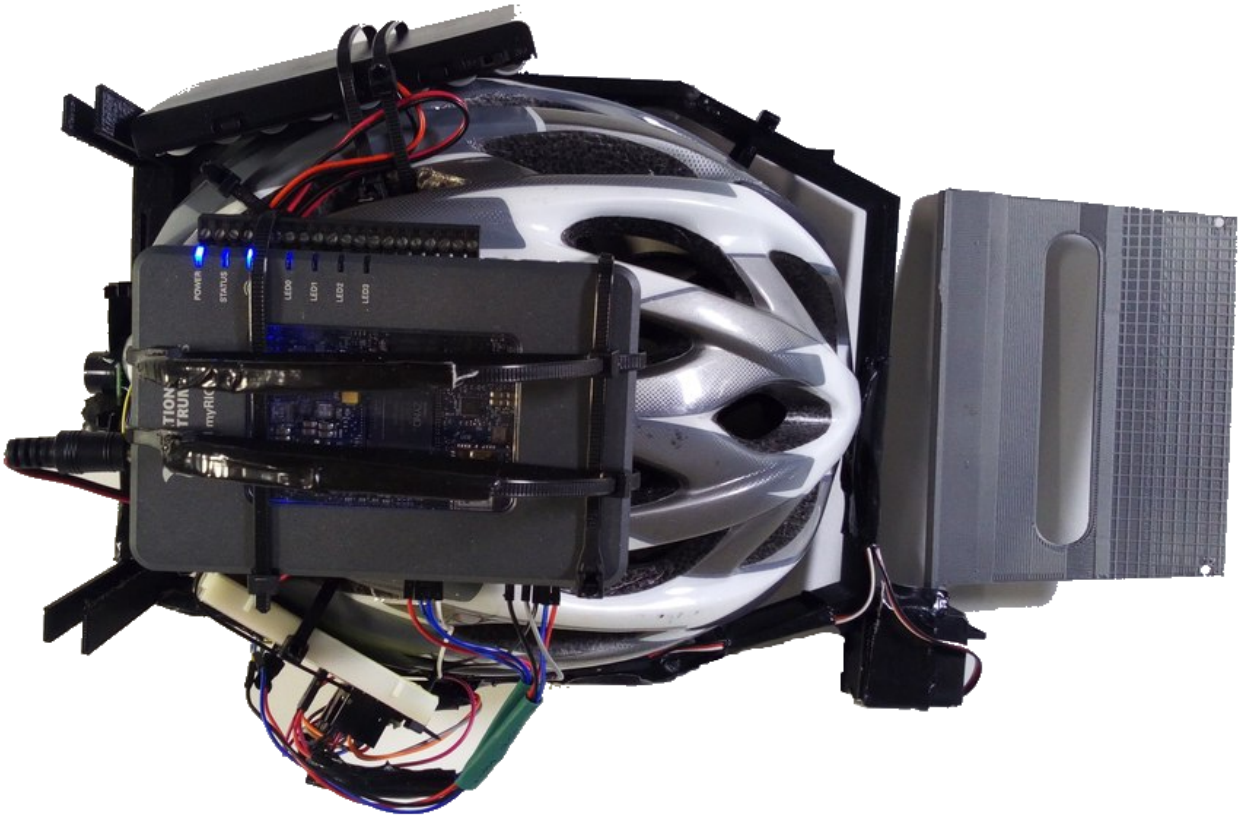
AUTONOMOUS VISOR CYCLING HELMET

sites.google.com/berkeley.edu/me135helmet



University of California, Berkeley
ME 135, Design of Microprocessor-Based Mechanical Systems
Spring 2018, Prof. George Anwar

Xander White
Mechanical Engineering, xanderwhite44@berkeley.edu
Andrew Shacker
Mechanical Engineering, ashacker@berkeley.edu, CLAD
Ning Xuan Tan
Mechanical Engineering, tanningxuan@berkeley.edu, CLAD

BACKGROUND & MOTIVATION

Nearly 10% of ER patients who suffer facial fractures are injured in a bicycle accident.[1] Maxillofacial fractures were found to be frequent even among riders who wore a national-standard helmet at the time of injury. Standard road cycle helmets provide significant protection against brain injury, but less protection against facial fractures. Midface fractures were frequent in bicycle injuries, with zygomatic fractures the most common site as the cheek is a likely first point of collision. Riding a road bicycle with a full-face helmet is problematic for cyclists since the forward weight of the helmet and visor induces strain on the back and neck of the rider. Full-face helmets also cause difficulty with breathing and limit field of view.

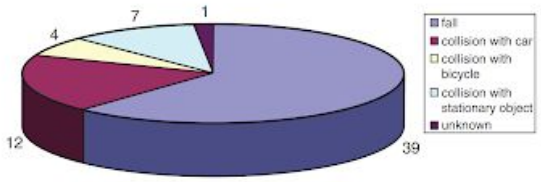


Fig 2. Causes of road bicycle-related injuries. (Fall = 61.9%, collision with car = 19.0%, collision with bicycle = 6.3%, and collision with a stationary object = 11.1%, cause unknown = 1.6%.)

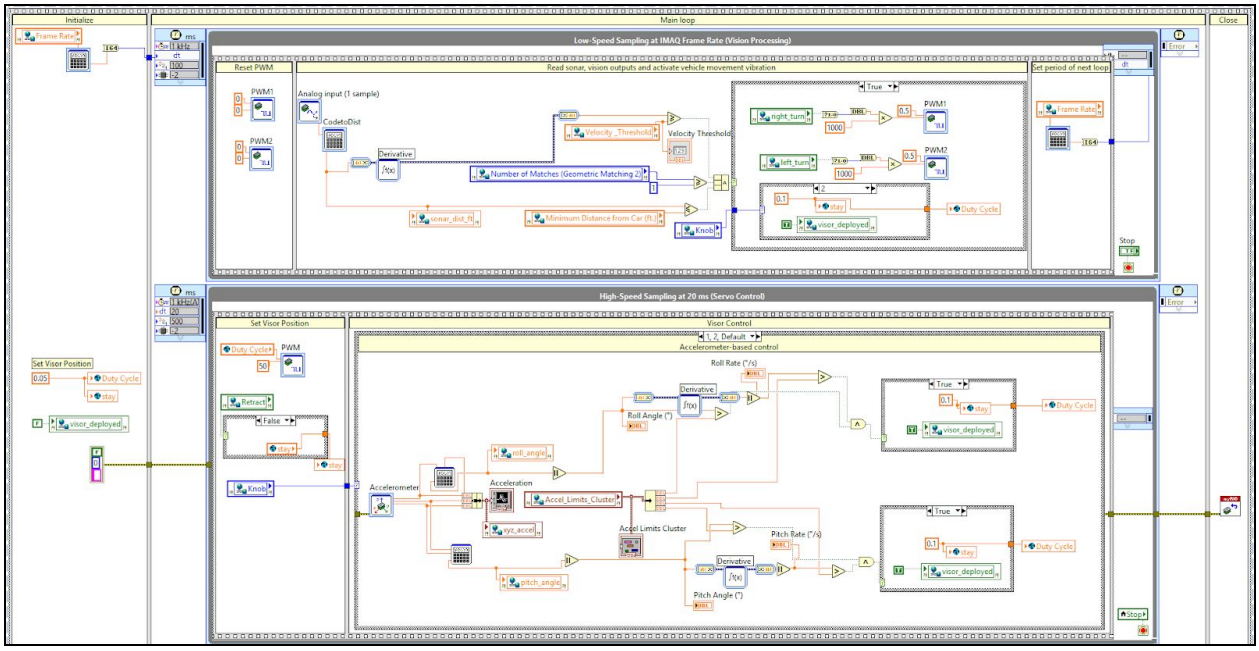
Table 1. Distribution of fracture sites

Site of fracture	Total (%)
High level	0
–	
Mid-face	48 (60)
– nasal ethmoidal	1 (1.1)
– nasal	6 (7.5)
– orbital	6 (7.5)
– zygoma	25 (31.3)
– maxilla	10 (12.5)
Mandible	32 (40)

Our traditional cyclist helmet mated with a retractable visor that deploys only when a rapid roll rate and excessive angle are detected preserves the rider's field of view, comfort, and appearance. The helmet features directional vehicle vibration warnings using image processing and motion estimation, activated when a high closing velocity and proximity are measured by the rear sonar sensor. Ideally, the visor would be made of polycarbonate material similar to face shields used for machining.

[1] https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1834-7819.2008.00056.x

FULFILLMENT OF PROJECT REQUIREMENTS

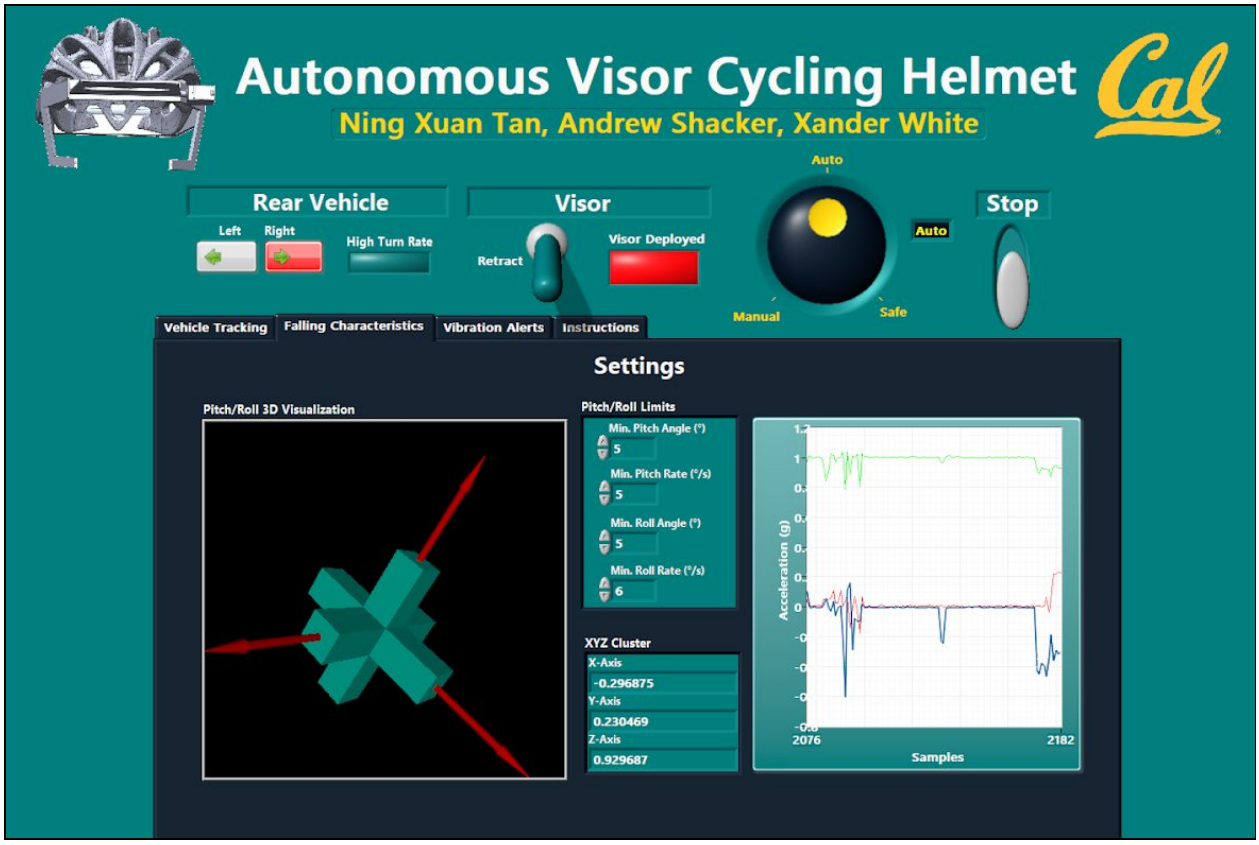


Real-Time Component

The VI running on the myRIO is configured around two timed loops: one low-speed sampling loop for receiving vision processing variables and one high-speed sampling loop at higher priority that actuates the visor servo based on accelerometer data if autonomous or safe mode is enabled. The low-speed sampling loop period changes dynamically based on the vision acquisition frame rate. The high-speed sampling loop updates at the servo pulse period of 20 ms. Manual mode disables accelerometer input to reduce latency in image processing on the PC. Typical vision frame rates from our webcam have been around 25-30 fps, while the IP phone camera refreshes at a much lower rate depending on strength of the WiFi network. We are currently using “IP Webcam” by Pavel Khlebovich for Android video broadcasting. In the future, we hope to use Ivideon to acquire video broadcasting over cellular networks in addition to WiFi.

Multi-Tasking Component

The high-speed sampling servo control loop (lower loop) is configured for a higher priority than the low-speed vision and sonar sampling loop. This is to ensure that the second loop executes first as it is the most critical to the rider’s safety. If the vision acquisition frame rate is slow and the visor control loop does not run until after the vision loop has finished (late), this could cause the visor to deploy long after the rider has been injured.



GUI

Visible above is the AVC helmet GUI with primary controls immediately visible outside of the tab block and secondary controls (settings) embedded into each tab. The VI is organized using a flat sequence within a while loop to execute the following actions: (1) read and write shared variables, (2) IP Camera image acquisition, and (3) conduct vision processing using the NI Vision Module.

PROJECTED TECHNICAL CHALLENGES

Some challenges we anticipated in our project were:

1. Implementation of vision processing, and how to carry out vehicle detection
2. Wireless connection between myRio, phone, and PC
3. Vision acquisition and image processing latency
4. Battery life and system portability
5. Simplifying our user interface

ACTUAL TECHNICAL CHALLENGES

Challenges we actually encountered during our project were:

1. Finding an accurate template for our vehicle matching function
2. Distance ranging in dark or obscure conditions
3. Optimizing motion estimation accuracy

4. Determining rider orientation and 3D GUI visualization
5. Wireless connection and configuring a camera IP adapter for Android

SOFTWARE

Vision Acquisition

Configuring IP Camera

We used the following tutorial to interface our Android camera stream with LabVIEW. Settings chosen were local broadcasting of the front-facing camera (for ease of testing) with 1280 x 720 resolution and photo storage off. Once the server on IP Webcam was started and camera feed entered into IP Camera Adapter, the video source could be selected using the Vision Acquisition Express VI or MAX. In case that the streaming frame rate lowered before a collision warning scenario, we selected continuous acquisition with inline processing to acquire the most recent image. In addition, we set the camera frame rate to change the low-speed myRio sampling loop period on each iteration for preventing missed cases.

Wireless Image Transfer

One issue we had was determining how to transfer images wirelessly while also communicating wirelessly between the PC and myRio. As we had not yet configured our camera on the cloud, a mobile hotspot allowed us to connect the phone to the PC and configure the PC and myRio to the same port.

Image Processing

Vehicle Detection

Another issue we had was determining the best means of detecting a vehicle. We chose the geometric matching function to detect one prevalent body outline shared by most vehicles: the trace between each side mirror and the top of the windshield. In order to use the matching function, we needed to extract the intensity color plane to convert the Android's RGB images to 8 bits. The outline was tested successfully on a small sample of cars. The most important adjustments were minimum score, edge threshold, and scale limits for accommodating a wide range of vehicle distances. In the future, we hope to implement a convolutional neural network using the Analytics and Machine Learning Toolkit to detect a wider variety of road vehicles.

Match Overlay

An array of bounding coordinates around each identified vehicle outline was unbundled and packed into an array to be overlaid on the input image using IMAQ Multiple Lines. In addition to the vehicle bounding box, two other bounding boxes were overlaid on the final image displayed in the GUI. The first was a bounding box for the points chosen to be tracked by the LKP motion estimation, and the second was the resulting location of those points at the end of the loop. We were able to reduce processing latency by removing wait functions from the loop and instead acquiring a successive frame later in the same loop.

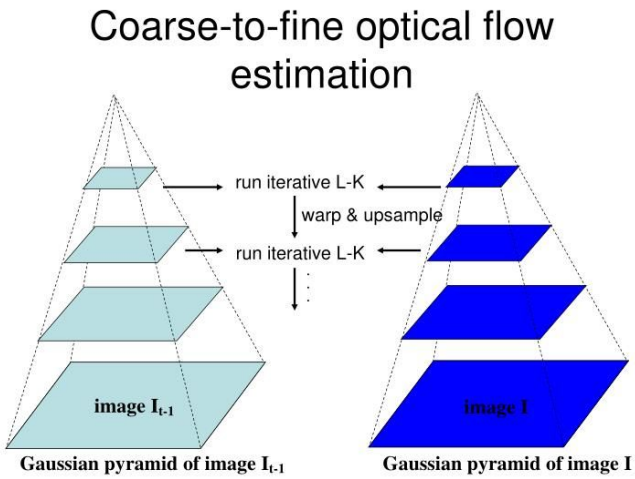
Motion Estimation

The motion estimation palette comes with NI VDM. It features optical (velocity) flow information using different algorithms. Motion estimation was the process of determining motion vectors In this project, we used the differential Lucas-Kanade method with Gaussian pyramidal representation to compute the change in location of an array of points obtained from geometric matching. This was helpful for determining the vectors of a detected vehicle in close range and providing a directional advisory to the cyclist in real-time.

The pyramidal Lucas-Kanade (LKP) algorithm makes the following feature point assumptions:

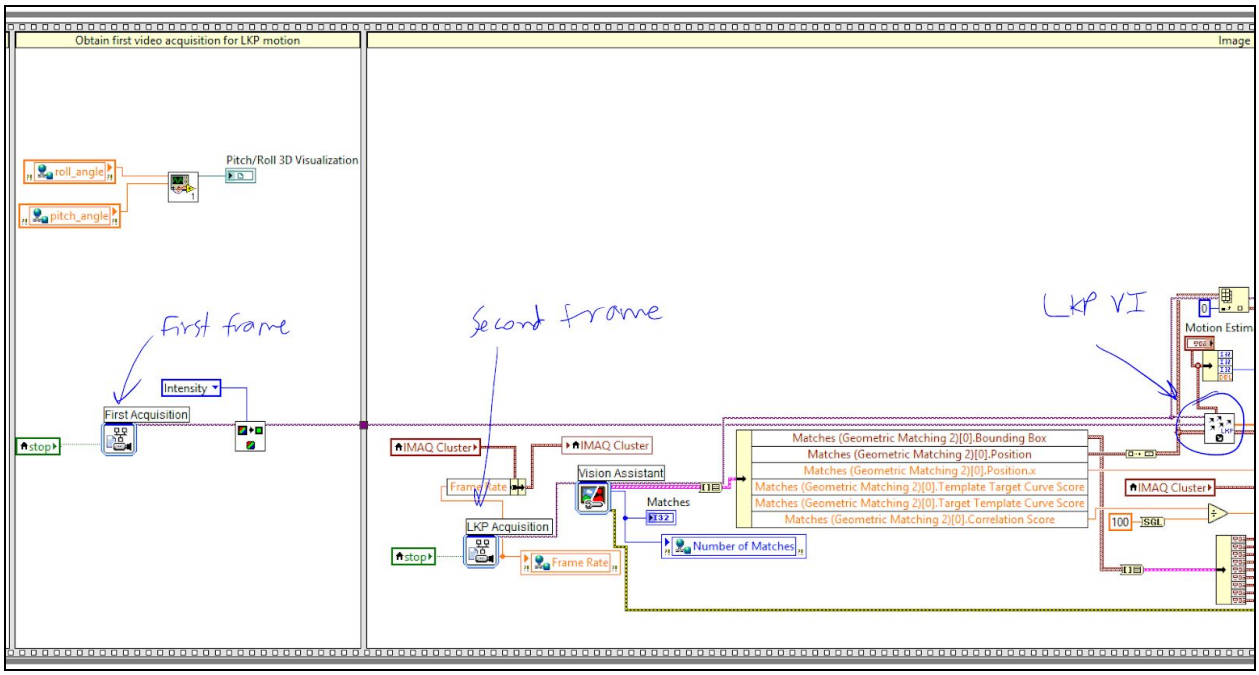
1. Intensity constant: projection of the same point and neighbors looks the same in every frame
2. Spatial coherence: points move like their neighbors

Normally a third assumption of "small motion" is asserted for LK. The pyramidal approach eliminates this constraint by downsampling to a low image resolution, computing optical flow, and upsampling to the next pyramid levels while refining estimations. This is known as "coarse-to-fine" optical flow estimation which works similarly to pooling in a convolutional neural network.



LKP vs. Successive Geometric Matching

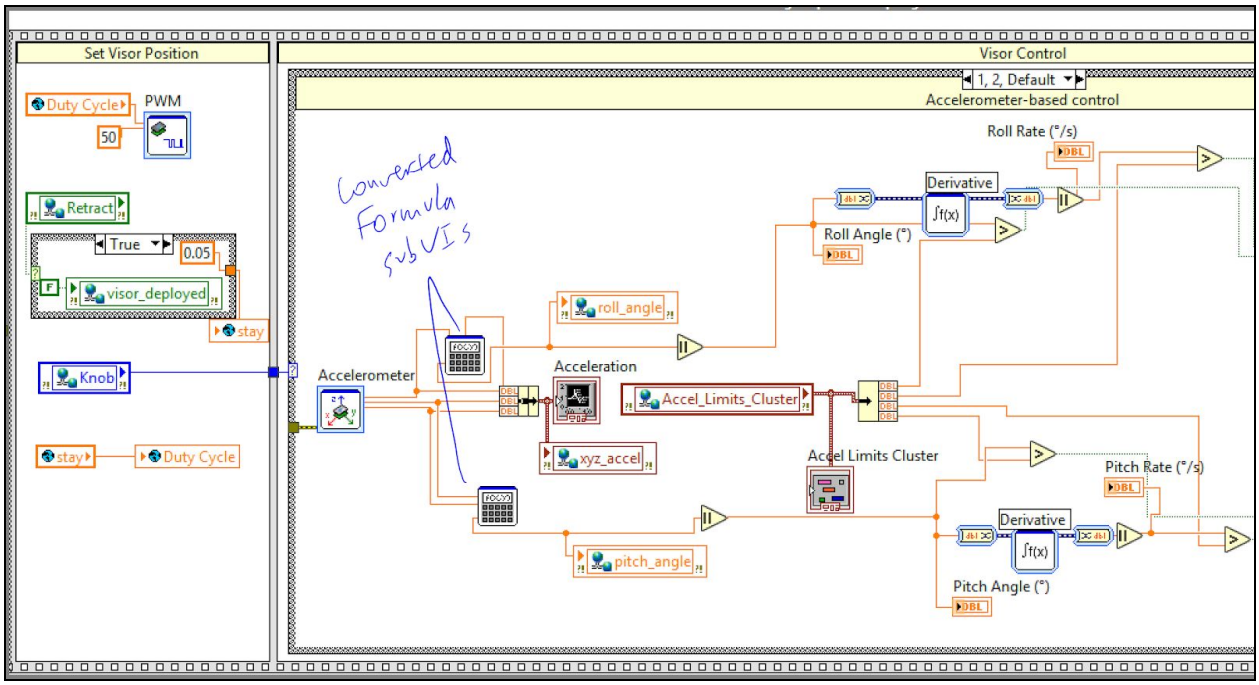
Two frames are input in LKP: one from the initial IMAQ acquisition at the beginning of the timed loop, and one near the end of the loop. Initially, we decided to compute the x-distance between the midpoint of each frame to determine the direction and rate of the detected vehicle. However, LKP provided greater accuracy and consistency by searching for intensities and spatial coherence in a window of variable size around each feature point (as opposed to just a geometric template). Once geometric matching has located the vehicle in the frame, the center coordinate of the bounding box can be passed to LKP with a smaller bounding box to isolate vehicle points from background points. The LKP algorithm can also be used to compensate for linear transformations where geometric detection might fail.



Accelerometer

3D Visualization

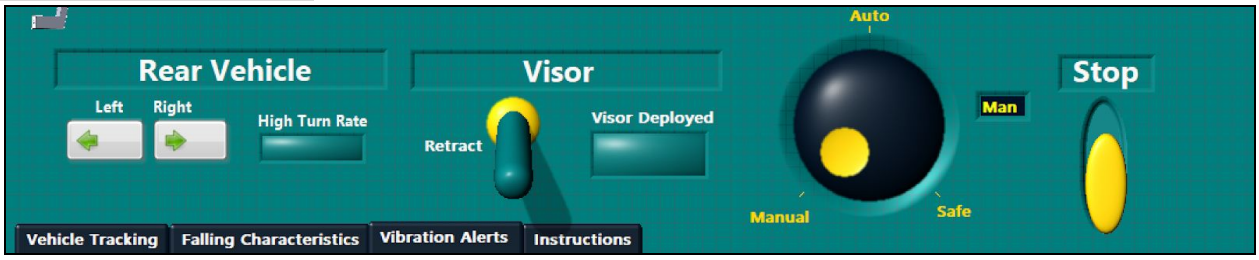
We used information from the following forum string to aid us in developing a 3D visualization of the rider's roll and pitch on the GUI. We also reduced computational latency by converting formula express VIs to SubVIs.



Limitations

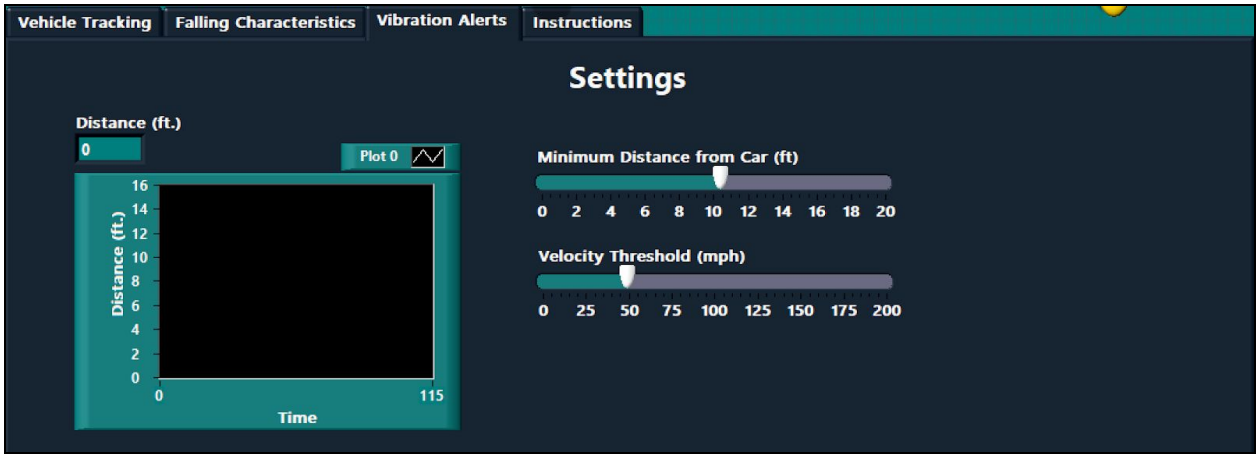
For this project, we assumed that a bicycle accident necessitates a tilt and that the attitude of the cyclist is very similar to that of the bike. It is possible for cyclists to fracture their face in other ways - e.g. stopping quickly and sailing over the handlebars, but a high acceleration does not reliably predict an accident while excessive tilt combined with high tilt rate does. Secondly, since the accelerometer data by itself is not filtered for horizontal acceleration, we would consider using an MPU-9250 with a complementary filter if a second iteration of the project were undertaken.

Graphical User Interface



Primary Row

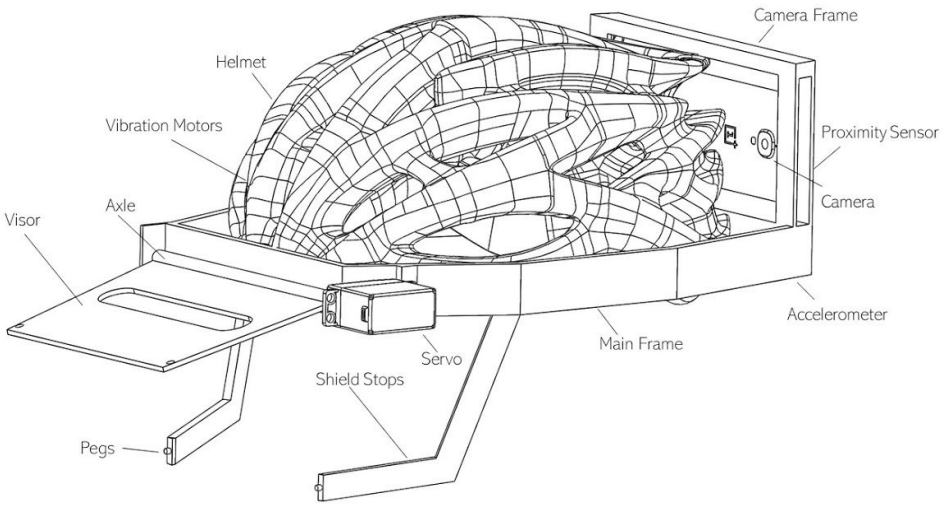
Rear Vehicle: Left and right boolean indicators show which direction the car being tracked by the rear camera is shifting using motion estimation. The retract control allows one to retract the visor (mainly for testing). The knob allows the user to cycle to their helmet mode.



Secondary Indicators (Settings)

The first setting tab, “Vehicle Tracking”, allows the user to observe if a vehicle match has been made, its score, camera frame rate, and motion estimation parameters. Three rectangles are visible in the image out display if a match has been made. A purple rectangle delineates the vehicle’s location, blue surrounding motion estimation points in, and green surrounding motion estimation points out. Under “Falling Characteristics”, the user is able to set their minimum roll/pitch angle and rate thresholds, as well as observe a visualization of the myRio’s orientation. Under “Vibration Alerts”, the user can observe the sonar distance readings as well as set thresholds for minimum warning distance and velocity. Finally, the “Instructions” tab offers a quickstart guide.

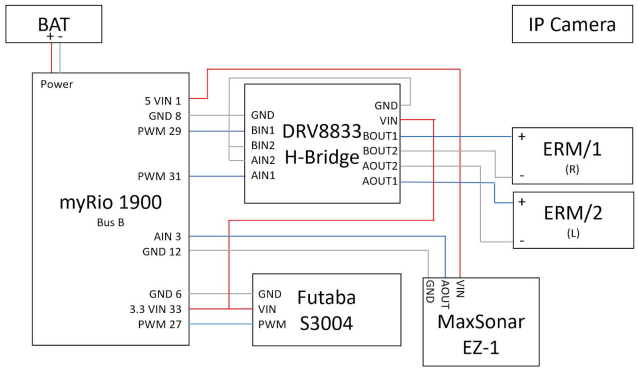
HARDWARE



The wiring diagram of our sensors and actuators is shown below. The AVC helmet uses the myRIO 1900 from National Instruments, running LabVIEW 2017. A 12-Volt battery powers the myRIO for portability,

and the myRIO powers an H-Bridge for dual-ERM motor control, a Futaba S3004 servo with 44 oz-in torque for visor actuation, and a 21 ft.-range MaxSonar EZ-1 ranging sensor. Two PWM pins connect to the H-Bridge to actuate each ERM motor. One motor corresponds to right-bound shifting of the vehicle detected by the rear camera, and the other left-bound shifting. Originally, we used a long-range SHARP infrared sensor, but decided that an ultrasonic sensor was more fitting for our application since proximity can be measured at any time of day and is not as sensitive to mist and dust particles or similar micro-forms of interference.

Three enumerated knob-controlled settings in the GUI affect how the sensors actuate the servo and ERM motors: (1) manual, (2) auto, and (3) safe mode. Vibration feedback for direction of vehicle shift at close proximity and high rate is enabled in all three modes. In autonomous mode, the visor deploys whenever a fall is detected (the roll/pitch angle and rate user-set limits have been passed). In safe mode, the visor deploys when a fall is detected or when a vehicle is detected and that vehicle is approaching at or within minimum set rate (and range). In manual mode, the visor is only deployed if a button on the myRIO is pressed.



Accelerometer input is disabled to aid the high-speed sampling loop in finishing on time. Positioning of the sensors and actuators are as follows: sonar sensor at the rear of the helmet, servo linking visor to axle fixed in front of the helmet, ERM motors embedded inside the helmet inner band, phone (camera) behind and on top of the sonar sensor, and myRIO and rechargeable batteries located on top of and center of the helmet. Note that in the above diagram, the front-facing camera is used for vehicle detection.

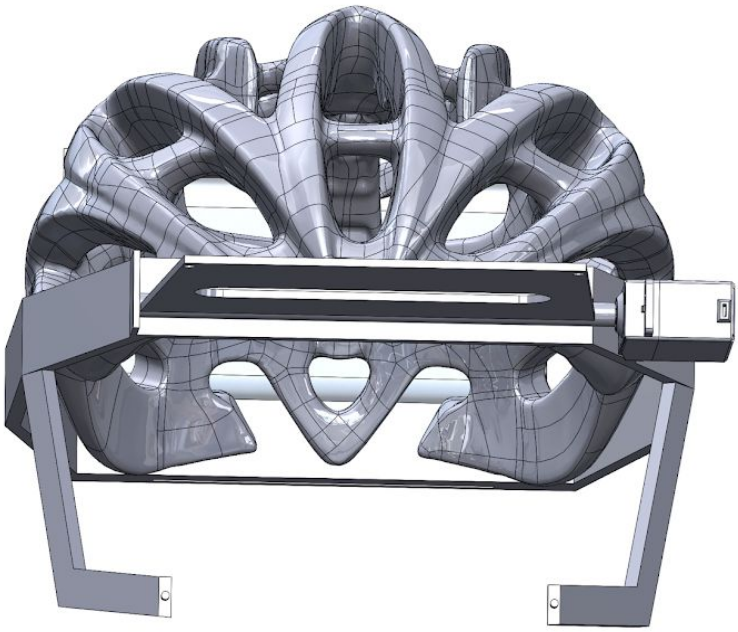
JOURNAL

1. Mid-Project Update: https://www.youtube.com/watch?v=flEV7e8rKg4
2. Testing: https://www.youtube.com/watch?v=x7Bmz4bzCZs
3. More videos on website: https://sites.google.com/berkeley.edu/me135helmet/programming

REFLECTION

While our AVC helmet demonstrated a successful proof-of-concept for lightweight safety visor helmets, there are many areas we can improve in. A production version of this helmet might require a gyroscope and rotating swivel to maintain the camera's alignment with rearward traffic, especially when the rider's head is tilted downwards. The vibration motors in this prototype notify the rider only of rear vehicles, but a final design might use the rotating swivel base in tandem with a 360-degree camera stitching app to activate vibration only on the side of the helmet corresponding to the location of the safety conflict relative to the bike. Another option would be to place the sensors on the bike frame, although this would

be less portable than a helmet configuration and prevents the rider from scanning different angles of their rear environment. Additionally, the visor used in the prototype design is optimized for 3D printing and less for safety. Polycarbonate material should be used for greater transparency and strength. The redesigned visor should feature a curved geometry to protect riders from cheek-area fractures further and be optimized for minimal drag. We also believe that vehicle detection can be improved using a neural network on the software side, and fish-eye lens and tethered camera on the hardware side for higher rates of data transfer.



ACKNOWLEDGMENTS

We would like to thank Professor **George Anwar** and GSI **Drew Sabelhaus** for their advice and thorough tutorials, as well as the Mechanical Engineering and EECS department for supplying much of the equipment and materials necessary to complete this project. We would also like to thank triathlete **Bola Malek** for his informative design feedback.