IoT-Enabled Smart Bike Helmet with an AI-Driven Collision Avoidance System

Jacob Solus, Maureen Rakotondraibe, Xinrui Yu, Won-Jae Yi, Mikhail Gromov and Jafar Saniie
Embedded Computing and Signal Processing (ECASP) Research Laboratory (http://ecasp.ece.iit.edu)
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago IL, U.S.A.

***Abstract*—This paper presents a system design for a smart bike helmet with multiple safety features that are intended to empower bicycle riders to proactively avoid potential sources of danger or injury. A Smart Sensor/Actuator Node (SSAN), driven by an Arduino Uno single-board microcontroller, contains input sensors and actuators to provide riders the ability to send and receive warnings promptly on their helmet. A Vision Node, driven by an NVIDIA Jetson Nano and a cable pin-connected camera, executes AI object detection algorithms for any dangerous objects that are out of sight of the rider and sends alerts to the SSAN as needed. By combining safety features of the SSAN and Vision Node while continuously sending data to an IoT-enabled backend web server, the safety operation of a typical bike ride can be substantially improved.**

Keywords—Computer Vision, Artificial Intelligence, Deep Learning, Edge Computing, Internet of Things, Bicycle Safety

I. Introduction

Many cities have made great progress over the past 5-10 years in building out infrastructure for bike transportation. Municipalities have employed separate bike lanes on major streets, biker-friendly crossing signals, harsher penalties for drivers who injure bikers, and even biker-only side roads. Yet despite societal advances in the safety of bike transportation, accidents remain a common and occasionally fatal occurrence in the daily life of many riders: approximately 130,000 cyclist injuries and 1,000 cyclist fatalities occur every year in the United States [1]. Enhanced protections from cities are not moving fast enough to prevent this seemingly avoidable source of injury and death. The bike helmet is the main source of protection for cyclists, and enhancing the capabilities of the helmet would be a seamless way to improve the safety of the bicycle riding experience.

Our smart bike helmet design incorporates multiple technical components that send and receive data through the Bluetooth protocol to help riders prevent accidents. Our helmet can detect collisions (using an accelerometer and an ultrasonic sensor) and send GPS-located messages to emergency contacts after a crash; an airbag-type system is then automatically deployed if a "false error" button is not pressed after the crash. This GPS sensor also enables the rider to warn other users of similar helmets within a user-defined distance if they see an unsafe driver, which results in network-empowered safety effects. The helmet uses a barometer to detect atmospheric shifts that are precursors to rain and warns the user when such a shift occurs. Finally, AI object detection algorithms are utilized on input images captured by a camera mounted on the back of the helmet. This system setup is used to detect whether an object outside of the rider's line of sight is both dangerous and within collision distance and alerts the user in these situations. All input and output data is continuously sent to a backend server for easy access and visualization by the user.

II. Related Works

Some previous research has explored basic technological improvements for bike helmets. Reference [2] developed a helmet with theft protection features and capabilities to detect whether the rider has consumed alcohol before the initiation of a ride. Reference [3] designed an RFID-based security system that ensures an electronic bike could only be started once the helmet is in the vicinity of the bike. Reference [4] created a system that ensures the helmet is being worn while riding a bike and provides a basic accident avoidance module that warns users when certain types of collisions may occur. Reference [5] incorporated similar alcohol detection features but additionally included a collision module with a GPS interface that allows the helmet to send a location signal of the biker to nearby hospitals when an accident occurs. However, even in the most sophisticated smart bike helmets that are commercially available (such as the Livall BH MT1, Lumos Ultra Helmet, and Hovding 3 Helmet), we see a lack of helmets that incorporate a cohesive set of safety features that harness the latest technological capabilities in the fields of artificial intelligence and edge computing.

Our AI image recognition research is inspired by previous work on plant image recognition [6] and license plate recognition [7]. However, we take our research farther by incorporating the output of an image recognition system with a supplementary system that uses positional and size data of the detected object to determine the proximity of the object to the camera. In our bike riding application, including this supplementary collision evaluation module is necessary in order to make AI image detection features actionable. Without such a module, the system would warn the rider of threatening objects that are too far from the rider to pose a realistic threat.

III. Smart Bike Helmet System Design

Our system contains three main components: a Smart Sensor/Actuator Node (SSAN), a Vision Node, and a backend web server. A diagram of the system, including the components and the types of communication, is shown in Fig. 1.

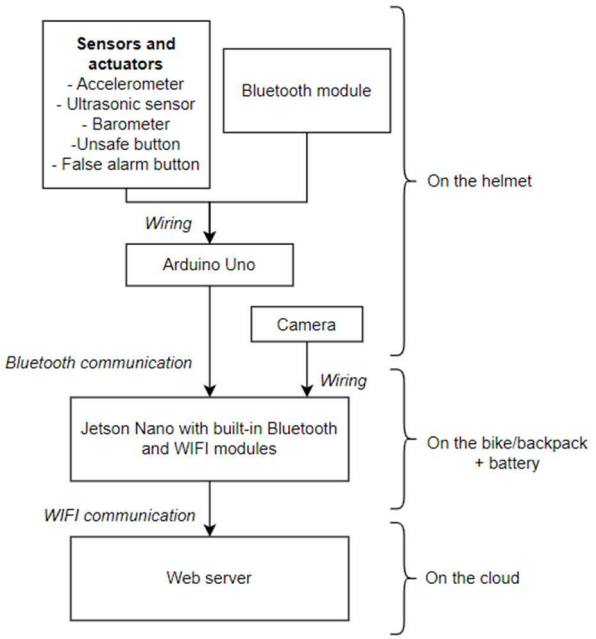

Fig. 1. System Flowchart of Smart Bike Helmet

A. Hardware Components of the System

The SSAN of the system is driven by an Arduino Uno single-board microcontroller operating at 5V voltage. An HC-06 Bluetooth module is connected to the Arduino Uno, which allows wireless communication with other parts of the system (both receiving and sending data). An ADXL345 accelerometer and HC-SR04 ultrasonic sensor work in tandem to detect when the rider has experienced a collision. To simulate the deployment of an airbag, an LED light is implemented to indicate when a signal is received for deploying an airbag in the case of a crash. Two push buttons are present on the helmet: one that allows the rider to inform the system they are safe when a crash is detected (and hence prevent emails being sent to emergency contacts), and one that allows the rider to alert others within their vicinity of unsafe riding conditions (such as a dangerous driver on the road). A BMP180 barometer detects changes in atmospheric pressure that could indicate the possibility of future unanticipated rain, and a Neo-6M GPS receiver collects GPS data while the system is powered on. All sensor node components are connected via pin wiring.

The core component of the Vision Node is an NVIDIA Jetson Nano with a built-in 128-core GPU with a 10W power consumption rate. The Jetson Nano in our design is equipped with an IMX 219 8MP camera and an Intel 8265 Wi-Fi/Bluetooth adapter. Finally, the back-end web server for the system is a ThingSpeak web server, which is accessed via a Python script running on the Jetson Nano that sends data via an HTTP web socket connection.

The entire system is powered by a portable and rechargeable power source with 24,000 mAh capacity. In practice, a power source with much less capacity could be employed, which would improve system cost and weight considerations.

B. Design Architecture of Programmatic Components

The SSAN, located on the helmet, is responsible for collecting the data from the different sensors and activating alerts and protection features if needed. The general flowchart of the SSAN is shown in Fig. 2.

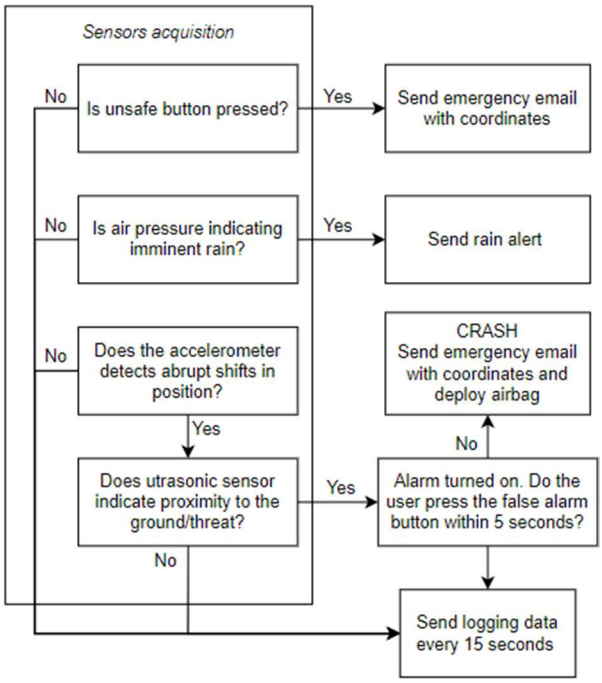

Fig. 2. Operation Flowchart for SSAN

If the acceleration magnitude is sufficiently high and the ultrasonic sensor detects an object within a certain distance (both inputs configurable by the user), the crash alert is raised. Also, if the unsafe button is pressed, the unsafe alert flag is set to 1 and sends an email to an emergency contact for assistance (alternatively, the system can send a warning to other users of the helmet within a certain distance of the rider). For rain detection, if the air pressure drops below a set threshold, the buzzer emits an alarm.

If there are no crashes or unsafe alerts, the system sends log data using the Bluetooth module every 15 seconds. If a crash is detected and the false positive button is not pressed within 5 seconds, an emergency email is sent to an emergency contact along with the last known GPS coordinates of the rider. With the false positive button acting as a safeguard, if the biker experiences a fall event but does not require assistance, then the biker can simply press the false positive button to avoid unnecessary contact triggers.

The design of data transmission across 15-second intervals allows us to reduce the power consumption of the Bluetooth module. Within 15 seconds, the air pressure and GPS data are unlikely to change significantly, so the chance of losing important data is low. This also ensures that the Bluetooth module is not overflowed and that the server has the time to process each received message. The message sent from the SSAN is a string containing all the information with a comma as

a separator. In addition, we include start and stop characters to ensure the message is not truncated.

C. AI Features and Algorithms

For designing the AI algorithms used to detect potential collisions, it is desired to split the task into two main components, since determining the potential danger posed by a captured image can be segmented into two questions: whether an object that is detected in the image is potentially dangerous, and whether that object is approaching a collision zone near the rider. We call these, respectively, the object recognition module and distance/collision evaluation module.

The object recognition module is tasked with detecting an object within an image captured by the rider, classifying that image, and then determining whether that image falls within a predefined set of potentially dangerous objects. For this module, we evaluated the performance of various Deep Neural Networks (DNNs) for recognizing live objects with the Jetson Nano camera in the street. We used this approach instead of testing on static image files to ensure that the DNN was appropriately vetted for the environment in which our helmet would be applied.

Each DNN detects the "class" of an object in an image and outputs the confidence level of the algorithm in its prediction (from a level of 0-100%). These confidence levels are used in our system for allowing users to heighten or lower the frequency of warnings being sent to the rider; if the DNN is not sufficiently confident that the detected object is a threat, no warning will be sent. For each network, we tested 50 live video samples, with half representing a car passing by on the street and the other half representing a non-car object such as a mailbox, garbage can, or tree. We stored several measures during each trial, which are presented in Table I. The average frames per second (FPS) of the DNN tells us how many image frames are processed per second and is an important measure of the computational speed of each network. We stored whether the DNN made a successful or erroneous prediction in each trial and the confidence level of the DNN for each prediction. The false positive rate tells us the percentage of positive threat warnings that were incorrect, and the false negative rate tells us the percentage of non-threat predictions that were incorrect (i.e. the network indicated an object was not a threat when it was a threat). Average confidence levels were bifurcated for successful predictions and unsuccessful predictions. Only three of the networks provided positional coordinates of detected objects in the image (i.e. the region within the image that a detected object lies within), which is an important input for the distance/collision evaluation module.

The detected objects are then analyzed by a distance/collision evaluation module. To derive a solution without additional hardware or complex software implementation, we designed an algorithm that directly analyzes the content of each frame. To avoid detecting cars parked on the side of the road or pedestrians on the sidewalk as threats, a "threat zone" is defined that excludes sidewalks and street parking spaces. An example is shown in Fig. 3; the threat zone is delimited by the white line in the bottom center of the frame.

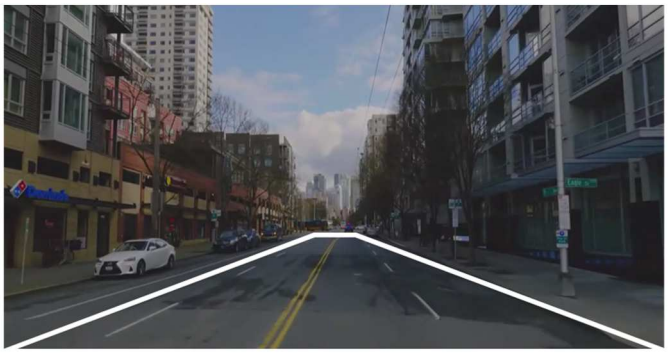

Fig. 3. Evaluated Threat Zone for Distance/Collision Evaluation

If a threat object is detected within the threat zone, the system then checks the size of the detected object. The closer it is to the biker, the bigger it will be on the frame due to perspective. We used OpenCV to visualize the bounding boxes and threat status. A box is drawn if the object detected is a threat. If all the criteria are met, the system flags a high collision risk. A message is then sent to the SSAN on the helmet to turn on the buzzer and alert the user. We evaluated research when considering the incorporation of additional OpenCV features in our evaluation module [8], but as mentioned above, we found that too much computational complexity for this module led to poor performance that compromised the ability of warnings to be sent to the rider within a sufficient time to react to the threat. An operation flowchart is shown in Fig. 4.

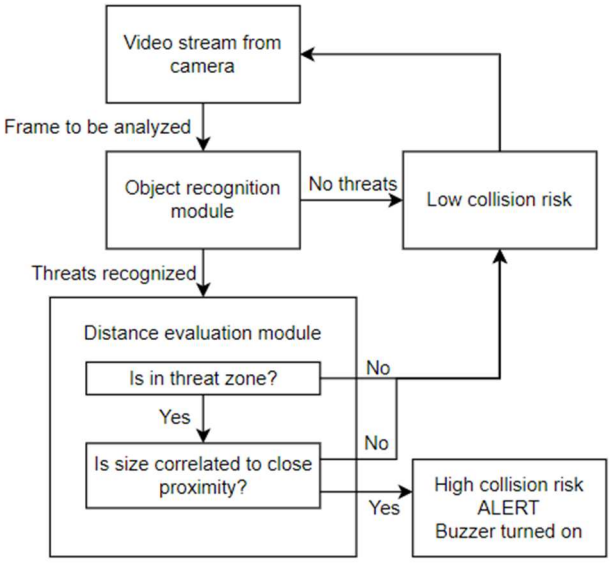

Fig. 4. Operation Flowchart for Vision Node

IV. Results and analysis

A picture of the implemented helmet is included in Fig. 5. The helmet was tested in street conditions, and multiple safety features were vetted.

Fig. 5 Picture of Implemented Smart Bike Helmet

A. Live Trial of Crash Event

We performed a live demonstration of a rider getting into a crash while wearing the helmet. First, the user fell to the ground and did not press the "false alarm" button, which caused the accelerometer and ultrasonic sensor to trigger a warning. The warning was processed by the SSAN and sent via Bluetooth to the Vision Node, which immediately sent a notification to an emergency contact with the GPS coordinates of the fallen rider. Subsequently, the rider sent an unsafe conditions alert to an emergency contact, which was received promptly after the input button was pressed.

B. Object Recognition DNN Comparison

When choosing an object recognition algorithm to use for the system, we evaluated performance measures collected for each network as described above and presented in Table I. We eliminated networks from consideration with average frames per second (FPS) under 20, since FPS values under this level reflect compromised ability to detect fast-moving vehicles. Notably, not all of the DNNs provided positional coordinates of the detected object, which is an important input to the distance/collision evaluation module; accordingly, networks without these capabilities were eliminated. With these filters in place, we then chose the network with the highest average confidence on successful predictions and lowest average confidence on erroneous predictions; SSD-mobilenet-v2 yielded superior performance when combining these considerations. We note that some research has indicated superior image recognition accuracy of the Yolo v3 architecture when compared to SSD-mobilenet-v2 [9], although these researchers have also found computational performance of Yolo to be lacking. Hardware differences and the relative importance of quick detection in our application were the main drivers of our difference in conclusion.

Table I. DNN Performance Comparison

DNN	Avg FPS	Fls. + rate	Fls. - rate	Avg conf., error	Avg conf., success	Incl. coord ?
AlexNet	60	2%	22%	38%	66%	No
GoogleNet	59	2%	16%	32%	80%	No
Inception-v4	10	2%	10%	12%	72%	No
ResNet-18	77	4%	26%	16%	68%	No
SSD-mobilenet-v2	22	0%	22%	12%	84%	Yes
SSD-Inception-v2	24	2%	16%	26%	76%	Yes
Yolo v3	2.5	4%	46%	32%	74%	Yes

C. Distance/Collision Evaluation Test

To test the crash prevention feature, we took our helmet to a busy street in order to feed input images to our camera that reflect the same images a biker would collect from the perspective of the back of their helmet. As displayed in Fig. 6, the algorithm detects the persons, cars, and bus in the frame. In the first picture, since the people and the bus are not in the threat zone, the collision risk is not high. The cars on the road coming towards the biker are in the threat zone, but they aren't big enough on the frame, meaning that they are not close enough to be considered as high collision risk. The borders of the boxes stay green if there is a low risk of collision.

In the second picture, the cars get closer to the biker and the bounding box around the dangerous car turns red. This means that this car has a high collision risk with the rider, and a Bluetooth message was sent to the SSAN that activated the helmet buzzer. We note that positioning the camera on the back of the helmet proved particularly useful for cars positioned in the biker's blind spots. In several cases, electric cars with silent engines (which are growing increasingly prevalent on city streets) were detected by the helmet, and these cars would have been very difficult for a bike rider to detect in their blind spot due to the lack of engine noise.

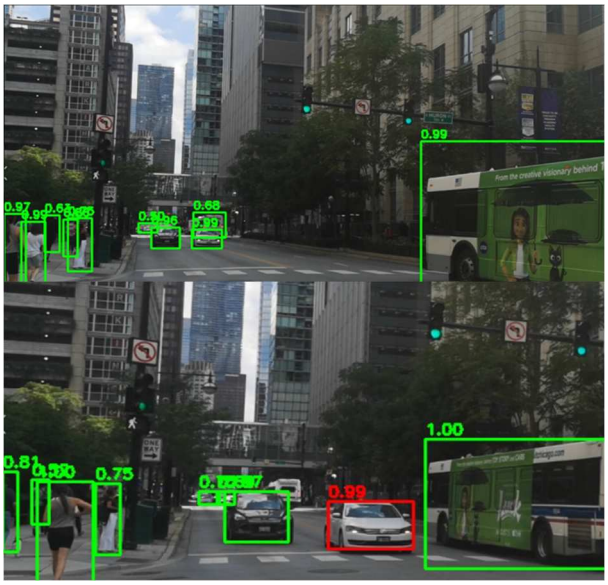

Fig. 6 Output from Distance/Collision Evaluation

D. *Data Visualization in the ThingSpeak Server*

The IoT capabilities of the helmet allow the biker to create visualizations and view statistics of their journey by accessing their account on the ThingSpeak web server. The biker can view a graphical history of accelerometer and barometer data, along with their longitude and latitude throughout their previous ride. This GPS data can also be accessed via a geographical map developed with a MATLAB script, which is presented in Fig. 7.

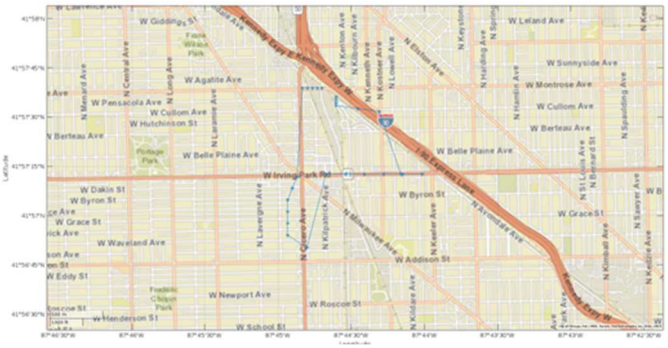

Fig. 7 Graphical Display of Historical Ride Coordinates

V. Conclusion

In this paper, we have presented a full-featured bike helmet safety system. This system contains multiple nodes that work together to take input from the biker and their surroundings and provide output to the rider that alerts them of unsafe conditions during their ride. In street testing, the helmet provided accurate detection of potentially dangerous objects that were out of the line of sight of the rider yet within collision distance and position.

Various improvements, left as future work, could enhance the overall design and usability of the system. Ideally, the way the rider would interact with the system would be via a smartphone app, which would reduce the system weight on the helmet and provide a more seamless user experience. Replacing the Jetson Nano in our system with a smartphone that is capable of running the collision evaluation algorithms, and integrating the SSAN with a smartphone app via Bluetooth, would provide a more seamless experience for the user.

For warning other riders of unsafe conditions, it would be ideal if an open-source API could be developed to allow multiple entities the ability to provide warnings. For example, if a city has red light cameras that detect unsafe drivers, then it is imaginable that our helmets could receive alerts of these situations; this would require coordinated development into an open-source API with government entities. As mentioned above, using a lower-cost power source and reducing the power requirements of the system could improve cost considerations. Of course, replacing our LED light emulation of an airbag with a deployable safety device is an additional area for future development. Finally, we note that safety standards for bike helmets exist in many countries, such as the U.S. CPSC Safety Standard for Bicycle Helmets for Persons Age 5 and Older; our helmet was not yet vetted for such standards.

References

[1] Centers for Disease Control and Prevention Web-based Injury Statistics Query and Reporting System (WISQARS). [online]

[2] P. C, R. C, P. N. M, R. P. S and S. M, "Smart Bike Helmet with Vehicle Tracking System using Arduino," 2022 International Conference on Edge Computing and Applications (ICECAA), Tamilnadu, India, 2022, pp. 579-582, doi: 10.1109/ICECAA55415.2022.9936590.

[3] D. K. P. Gudavalli, B. S. Rani and C. V. Sagar, "Helmet operated smart E-bike," 2017 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS), Srivilliputtur, India, 2017, pp. 1-5, doi: 10.1109/ITCOSP.2017.8303138.

[4] N. Nataraja, K. S. Mamatha, Keshavamurthy and Shivashankar, "SMART HELMET," 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 2018, pp. 2338-2341, doi: 10.1109/RTEICT42901.2018.9012338.

[5] A. Jesudoss, R. Vybhavi and B. Anusha, "Design of Smart Helmet for Accident Avoidance," 2019 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 2019, pp. 0774-0778, doi: 10.1109/ICCSP.2019.8698000.

[6] S. Chavan, J. Ford, X. Yu and J. Saniie, "Plant Species Image Recognition using Artificial Intelligence on Jetson Nano Computational Platform," 2021 IEEE International Conference on Electro Information Technology (EIT), Mt. Pleasant, MI, USA, 2021, pp. 350-354, doi: 10.1109/EIT51626.2021.9491893.

[7] N. Awalgaonkar, P. Bartakke and R. Chaugule, "Automatic License Plate Recognition System Using SSD," 2021 International Symposium of Asian Control Association on Intelligent Robotics and Industrial Automation (IRIA), Goa, India, 2021, pp. 394-399, doi: 10.1109/IRIA53009.2021.9588707.

[8] F. K. Noble, "Comparison of OpenCV's feature detectors and feature matchers," 2016 23rd International Conference on Mechatronics and Machine Vision in Practice (M2VIP), Nanjing, China, 2016, pp. 1-6, doi: 10.1109/M2VIP.2016.7827292.

[9] A. C. Rios, D. H. dos Reis, R. M. da Silva, M. A. de Souza Leite Cuadros and D. F. T. Gamarra, "Comparison of the YOLOv3 and SSD MobileNet v2 Algorithms for Identifying Objects in Images from an Indoor Robotics Dataset," 2021 14th IEEE International Conference on Industry Applications (INDUSCON), São Paulo, Brazil, 2021, pp. 96-101, doi: 10.1109/INDUSCON51756.2021.9529585.