IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received May 17, 2021, accepted May 26, 2021, date of publication June 14, 2021, date of current version June 23, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3086499

Closed-Loop Region of Interest Enabling High Spatial and Temporal Resolutions in Object Detection and Tracking via Wireless Camera

JACK CHEN[1,2], HEN-WEI HUANG[1,3,4], PHILIPP RUPP[1,5], ANJALI SINHA[4], CLAAS EHMKE[3], AND GIOVANNI TRAVERSO[1,4]

[1]Division of Gastroenterology, Brigham and Women's Hospital, Harvard Medical School, Boston, MA 02115, USA
[2]Department of Engineering Science, University of Toronto, Toronto, ON M5S2E4, Canada
[3]Koch Institute for Integrative Cancer Research, Massachusetts Institute of Technology, Cambridge, MA 02142, USA
[4]Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[5]Department of Information Technology and Electrical Engineering, ETH Zürich, 8092 Zürich, Switzerland

Corresponding author: Giovanni Traverso (cgt20@mit.edu)

This work was supported in part by the Karl van Tassel (1925) Career Development Professorship, in part by the Department of Mechanical Engineering, Massachusetts Institute of Technology, and in part by the Brigham and Women's Hospital (G.T.).

ABSTRACT The trade-off between spatial and temporal resolution remains a fundamental challenge in machine vision. A captured image often contains a significant amount of redundant information, and only a small region of interest (ROI) is necessary for object detection and tracking. In this paper, we first systematically characterize the effects of ROI on camera capturing, data transmission, and image processing. We then present the closed-loop ROI algorithm capable of high spatial and temporal resolution as well as wide scanning field of view (FOV) in single and multi-object detection and tracking via real-time wireless video streaming. With the feedback from real-time object tracking, the wireless camera is able to capture and transmit only the ROI which in turn enhances both the spatial and temporal resolution in object tracking. In addition, the proposed approach can still maintain a large FOV by processing regions outside of the ROI at lower spatial and temporal resolutions. When applied to a high spatial resolution wireless stream (5 MegaPixels), the closed-loop ROI algorithm improves the temporal resolution by up to 10× (from 2.4 FPS to 22.5 FPS). Specifically, camera processing is improved by up to 4.7×, data transmission is improved by up to 160×, and PC processing is improved by up to 2.5×. In a person tracking experiment, the closed-loop ROI algorithm enables a wide-angle camera to outperform both a normal wide-angle camera–which suffers from poor temporal resolution and motion blur–and a pan & tilt camera–which cannot automatically refresh tracking after the tracking is lost.

INDEX TERMS Machine vision, object detection, object tracking, real-time systems, region of interest, spatial resolution, temporal resolution.

I. INTRODUCTION

A fundamental trade-off between spatial and temporal resolution remains a key challenge in machine vision. Higher spatial resolution frames require more time to capture, process, and transmit, resulting in lower temporal resolution. However, both high spatial and high temporal resolutions may be desirable as they inform object detection and tracking accuracy. Spatial resolution can affect the accuracy of neural networks [1], [2], and even small changes can have a significant effect depending on the application [3]. Similarly, temporal resolution can affect the accuracy of object detection and object tracking [4]–[6]. As a result, there is often a trade-off between spatial and temporal resolutions [7], [8].

When detecting and tracking an object, its location forms a region of interest (ROI). However, a ROI may occupy only a small portion of the field of view (FOV). Thus, major computational resources may be spent on capturing, transmitting, and processing redundant information. To balance spatial and temporal resolution for real-time detection and tracking, previous researchers have proposed ROI coding (in which only the ROI is encoded in high quality) and ROI detection/prediction (in which only the ROI is transmitted or processed) [9]. The ROI concept has been

The associate editor coordinating the review of this manuscript and approving it for publication was Tomasz Trzcinski.

implemented in a wide variety of applications, including medical imaging [10], vehicle lane warning systems [11], and ultra-high-speed video streaming [12]. ROI coding and ROI detection/prediction methods are especially useful for wireless video streaming due to slower and less consistent image transmission [13]. However, previous solutions were designed for specific goals and not applicable for general object detection/tracking. Furthermore, ROI coding has generally been applied for post-processing and not fully applied at all levels of wireless video streaming: image capturing, image transmission, and image processing.

In this paper, we propose the closed-loop ROI algorithm, a general method for single and multi-object detection/tracking with a feedback loop involving a wireless camera and an external PC. In the proposed algorithm, the camera only captures the ROI and transmits the ROI to the PC via Wi-Fi; the PC then processes the ROI and centers the next ROI based on the object's current location. Thus, the location and information of the current ROI is processed to determine the location of the subsequent ROI inside a closed loop. Figure 1 shows how the proposed closed-loop ROI method enables a wireless camera to operate at both higher spatial and temporal resolutions than the normal method in which the camera captures and transmits the entire FOV for processing. Although we restrict ourselves to person detection in this paper, the method can be used with an arbitrary detection algorithm for a variety of applications where both high spatial and temporal resolutions are desirable, from surveillance to medical imaging.

Additionally, the proposed algorithm supports machine vision applications involving centralized control of swarm robotics. As opposed to a distributed system, a centralized system has higher quality scheduling and greater compatibility with current communication architectures, but it has limited scalability, which causes increased scheduling latency [14], [15]. The closed-loop ROI method can improve scalability by significantly reducing data transmission of every machine vision sensor in the swarm.

The main contributions of this paper are threefold. Firstly, a systematic study of the ROI shows the benefits of applying the ROI to all levels of wireless video streaming. Secondly, by only capturing, transmitting, and processing the ROI, we enable a wireless video stream with both high temporal and spatial resolutions. Thirdly, the high spatial resolution enables a video stream to operate at high FOV without compromising image quality, allowing a larger region of the environment to be monitored for object detection and tracking.

II. RELATED WORK

A. ROI CODING

ROI coding encodes subregion(s) of the image in high quality and the rest of the image in low quality through various encoding process [16]–[18]. Methods for determining the ROI range from manual selection [19] to hidden Markov chains and Gaussian mixture models [20]. Beyond image compression and storage, past works have applied ROI

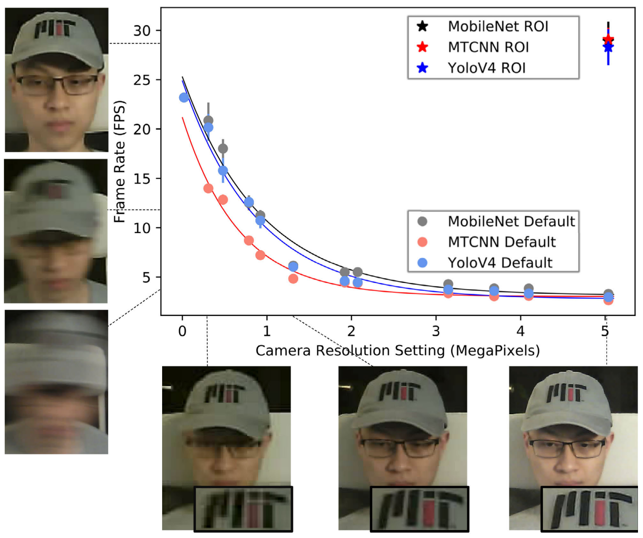

FIGURE 1. Comparison of wireless video streaming between the Default and closed-loop ROI methods. The closed-loop ROI method enables a 5 MegaPixels wireless stream at 30 FPS with a 0.03 MegaPixels ROI. Algorithms are implemented on a wireless OpenMV H7 Plus, a machine vision microcontroller. Object detection is done by one of three detection networks (MobileNet, MTCNN, YoloV4) on a NVIDIA Jetson AGX Xavier.

encoding to various applications such as medical imaging, in which only the diagnostically relevant region is required in high quality [21], and aerial surveillance, in which data transferred by aerial drones can be significantly reduced [9].

Since ROI coding is a post-processing step that occurs after the full image is already captured by the camera, it only affects the transmission and storage of images. Additionally, ROI coding can be computationally intensive and generally not possible on cameras with limited computational capabilities.

B. ROI DETECTION AND PREDICTION

Detecting and predicting the ROI has allowed for a variety of real-time detection and tracking applications. The methods in these works typically capture a video frame, detect object(s) of interest with a neural network, then predict the ROI location of the next frame using an estimation filter such as a Kalman filter. Past applications include magnetic resonance imaging [10], lane boundaries detection for vehicle warning systems [11], in-situ plankton tracking [22], micro-object detection under a microscope [18], ultra-high-speed fruit fly wing tracking [12], ball and athlete detection for adaptive compression during sports streams [13], remote photoplethymography [23], hand prints and hand vein detection [24], [25], and blink detection to monitor patients with amyotrophic lateral sclerosis [26].

Prior works have shown improvement in temporal resolution, but typically apply the ROI as a post-processing step and do not implement ROI-related operations at the camera level. Furthermore, prior works are designed for specific applications and require prior knowledge of the system and object dynamics. For instance, in the magnetic resonance imaging application, reconstruction of the image is computationally intensive; reconstructing on the ROI only leads to significantly faster processing [10]. Similarly, in the lane detection application, processing only a small ROI rather than the whole image allows for faster detection, which is critical for a real-time vehicle warning system [11]. However, in these works, the entire medical image or road image is acquired and transmitted, and time is only saved in the subsequent image processing steps. Additionally, these works are suitable only for the targeted application. The proposed closed-loop ROI method builds upon these works by leveraging the advantages of the ROI at every level of wireless video streaming for general object detection and tracking applications.

C. MACHINE VISION CAMERAS

There are a multitude of wireless machine vision cameras that capture images at varying spatial resolution, temporal resolution, format, etc. Popular camera providers include Sony, FLIR Machine Vision, and Cognex [27]. Alternatively, embedded machine vision products such as the ESP32, Raspberry Pi, and OpenMV provide low-cost options with small form factors [28]–[30]. The Raspberry Pi and OpenMV can run low-complexity neural networks in real-time for object detection and tracking. Similarly, the PixyCam can be taught to learn objects using a hue-based color filtering algorithm, but the objects must have distinctive colors. In general, the trade-off between spatial and temporal resolution remains a major challenge for machine vision cameras due to the limited RAM and processor performance for embedded computation. The closed-loop ROI algorithm overcomes the trade-off in spatial and temporal resolutions by only capturing, transmitting, and processing the ROI. Thus, implementing the closed-loop ROI algorithm on an intelligent camera can optimize the camera's capability. Additionally, the proposed algorithm overcomes limited processing power on embedded machine vision cameras by offloading heavy computation to an external PC.

D. OBJECT DETECTION AND TRACKING

Many neural networks have been proposed for object detection. Three examples of detectors that have been used in machine vision applications are YoloV4, MTCNN, and MobileNet. YoloV4 is a popular detector that makes detections with a single network evaluation, enabling it to operate at high speeds while achieving state-of-the-art accuracy [31]. MobileNet is an efficient model designed for mobile and embedded networks that optimizes the trade-off between speed and accuracy [32]. MTCNN utilizes a different approach involving a cascaded structure with a three-stage network that achieves state-of-art accuracy [33].

A common approach to single and multi-object tracking is with tracking-by-detection, in which detected objects in one frame are associated with detected objects in subsequent frames [34]. One method to associate detected objects is by calculating the intersection-over-union of detected objects between frames; this method is able to achieve state-of-the-art performance without using image information [35]. Alternative methods for object tracking involve using object motion, appearance, structure, and size [34], fusing sensor measurements and features from multiple sources with context aware descriptors [36], and integrating all available object observations to interpret scene dynamics [37].

III. DESIGN AND METHODS

A. CLOSED-LOOP ROI ALGORITHM

The closed-loop ROI algorithm involves a feedback loop between the camera and the external PC. Table 1 lists the camera parameters for the closed-loop ROI algorithm. F_w, F_h, ϕ_h, ϕ_w are camera specifications. θ and d are experimental setup values. Optimal dimensions for the ROI, R_h and R_w, are derived in Section III-B. Values for t_{loop}, t_{config}, and $t_{capture}$ are obtained after analyzing camera characteristics in Section IV; t_{loop} is a preset value, while t_{config} and $t_{capture}$ are fixed values dependent on ROI dimensions. Algorithm 1 details operations on the camera's controller to capture the ROI in the current frame. Algorithm 2 details operations on the external PC to predict the ROI location in the next frame.

TABLE 1. Camera parameters for the closed-loop ROI algorithm implemented on a wireless OpenMV H7 Plus.

Parameter	Description	Value
$F_w \times F_h$	Width × height of camera spatial resolution	2592 × 1944 pixels
$R_w \times R_h$	Width × height of ROI spatial resolution	400 × 150 pixels
$\phi_h \times \phi_h$	Horizontal × vertical field of view of camera	140 × 100°
θ	Angle of camera from the vertical axis	70°
d	Minimum distance of camera from object(s)	2m
t_{loop}	Preset time for one loop of the closed-loop ROI algorithm	0.1s
t_{config}	Time required to reconfigure ROI	0.026s
$t_{capture}$	Time required to capture one ROI	0.015s

The closed-loop ROI algorithm begins by establishing a wireless transmission control protocol (TCP) connection between the camera and external PC. Then, the camera captures an image of the entire FOV, encodes it as a byte array, and sends it to the PC over the TCP connection, which performs object detection and tracking. If the object is not detected, the PC sets the ROI for the next frame to be the entire FOV. If the object is detected, the PC predicts the location of the ROI in the next frame by centering the ROI around the object's current location (however, if the system and object dynamics are well known, the ROI prediction can be improved by using a predictive algorithm such as a Kalman filter or particle filter). The new ROI settings are sent by the PC to the camera. After the ROI is updated, the process repeats; the new ROI is captured and sent to the PC. For single object detection and tracking, only one closed-loop ROI needs to be instantiated. To enable multi-object detection

Algorithm 1 Camera: Capturing/Sending ROI

```
X, Y, W, H, n = 0, 0, $F_w$, $F_h$, 1;
while True do
    Configure_ROI(X, Y, W, H);
    Initialize_Timer();
    while Elapsed time < $t_{loop}/n$ do
        img = Capture_Configured_ROI();
        JPEG = Compress_JPEG(img);
        Send_to_PC(JPEG);
        Recv_from_PC(X, Y, W, H, n);
        if W == Fw then
            break;
        end
    end
end
```

Algorithm 2 PC: Processing/Predicting Multiple ROIs

```
IDcounter = 0;
while True do
    image = Recv_from_Cam();
    Detections = Detect(image);
    if Detections is None then
        | X, Y, W, H, n = 0, 0, $F_w$, $F_h$, 1;
    else if width(image) == Fw then
        n = num(Detections);
        for i = 0; i < n; i = i + 1 do
            ID = Track_Objects(Detections);
            location[ID] = Detections[ID];
        end
    else
        ID = Track_Objects(Detections);
        Object_Bounding_Box = Detections[ID];
        [x, y, w, h] = Object_Bounding_Box;
        X. = x - (Rw - w)/2;
        Y. = y - (Rh - h)/2;
        W = Rw;
        H. = Rh;
    end
    location[IDcounter % n] = (X, Y, W, H);
    (X, Y, W, H) = location[IDcounter + 1];
    Send_to_Camera(X, Y, W, H, n);
    IDcounter = (IDcounter + 1) % n;
end
```

tracking with n objects, a closed-loop ROI must be instantiated for each object.

The actual implementation of this algorithm is largely dependent on hardware. For the cameras we tested (OpenMV H7, OpenMV H7 Plus, FLIR Chameleon, FLIR Blackfly), updating the ROI requires the camera to settle and/or reset, requiring significant time. Thus, each closed-loop of the proposed algorithm runs for a preset period of time, t_{loop}. This and other camera characteristics are discussed in Section IV-A. The OpenMV H7 Plus cannot capture multiple ROIs in a single frame with the current firmware. Thus, for multi-object tracking with n objects, the OpenMV updates one ROI at a time and captures each ROI for t_{loop}/n seconds. Since the computation is offloaded to an external PC, this algorithm can always run state-of-the-art algorithms for object detection and object tracking assuming the PC has the necessary processing power.

B. OPTIMIZING ROI DIMENSIONS

In the closed-loop ROI algorithm, if the object moves out of the ROI before the ROI updates, the camera will capture the entire FOV in the next frame. Note that the object tracking is never lost as long as it remains in the FOV, but the frame rate is reduced temporarily until a new ROI is predicted. In this section, we determine the ROI dimensions required to track an object with a given speed using only an updating ROI. The model for the camera is shown in Figure 2.

To simplify the derivation, we assume the object's velocity has constant components v_x and v_z with the same maximum value. This confines the object movement during one frame within a square region. After the ROI's position is updated, the object moves for t_{loop} seconds before the ROI position is updated again. The next ROI will be centered on the object's current location, but the object will continue to move for time t_{config} while the ROI is being reconfigured. Thus, in time $t_{loop} + t_{config}$, the object must not move more than half of the ROI's width and height.

The ROI dimensions required to track the object with an updating ROI is expressed in Equations (1) and (2). The

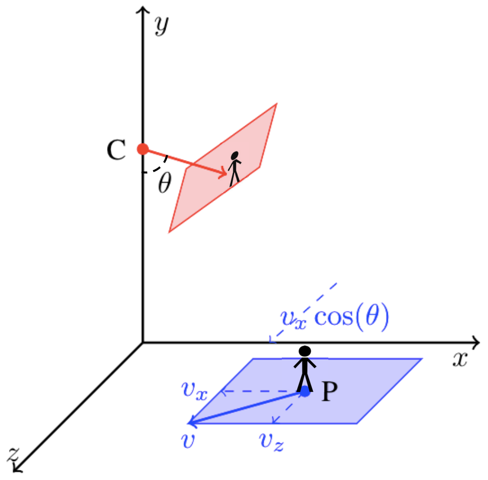

FIGURE 2. Image capturing model of a camera. The camera at point *C* with angle θ from the vertical is tracking an object at point *P* with constant velocity *v*. The shaded red region represents a captured video frame. The shaded blue region represents possible locations of P after one frame.

parameters ϕ_h, ϕ_w, F_h, F_w, and d are used to convert meters to pixels. Note that at the same ROI dimensions, a smaller t_{loop} results in faster v_x and v_z, meaning that the maximum object speed that can be tracked with an updating ROI is higher when t_{loop} decreases. However, we later show in Section IV-A

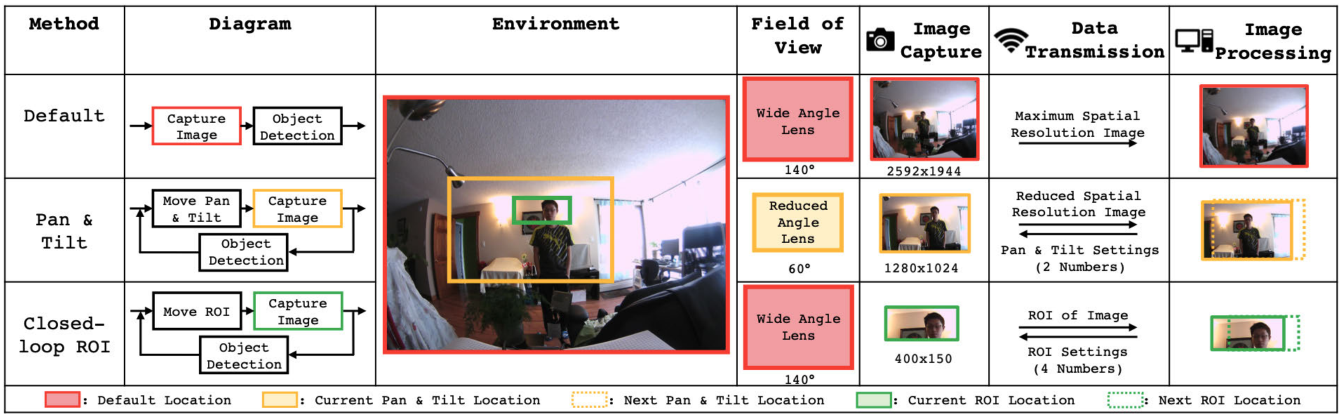

FIGURE 3. Visualization of camera streaming methods for object detection/tracking with a wireless camera.

that decreasing t_{loop} also decreases the temporal resolution. In Section IV-A, we set $t_{loop} = 0.1$ s to balance camera characteristics.

$$v_z * (t_{loop} + t_{config}) * \frac{F_w}{d * \phi_w} \leq R_w/2 \qquad (1)$$

$$v_x * \cos(\theta) * (t_{loop} + t_{config}) * \frac{F_h}{d * \phi_h} \leq R_h/2 \qquad (2)$$

Assuming a maximum speed of no more than 3 m/s, the required ROI dimensions is 401×144 pixels. In the experimental design, we use rounded values of 400×150 pixels (0.06 MegaPixels) for our implementation of the closed-loop ROI algorithm.

For multi-object tracking, only one closed-loop ROI is captured at a time. The camera cycles through the ROI of each object to capture the object's current location and predict its next location. Based on the previous derivations, the ROI must be updated within $t_{loop} + t_{config}$ seconds to track the object with a closed-loop ROI. To track n objects within $t_{loop} + t_{config}$, the camera must capture the ROI of an object at least once, requiring $t_{capture}$ seconds, then reconfigure the ROI for the next object, requiring t_{config} seconds. This is expressed in Equation (3).

$$n * (t_{capture} + t_{config}) \leq (t_{loop} + t_{config}) \qquad (3)$$

The number of tracked objects, n, has a maximum value dependent on the camera characteristics, experimental setup, and desired application. For our multi-person tracking demonstration, the closed-loop ROI for each person is set to the optimal value of 0.06 MegaPixels. Based on the camera characteristics discussed later in Section IV-A, $t_{capture}$ and t_{config} are fixed by the ROI dimensions, while t_{loop} is preset to 0.1 s. Thus, in our experimental setup, the maximum number of objects that can be tracked using the closed-loop ROI algorithm is $n = 3$. To track more than n objects, we recommend a multi-camera setup discussed later in Section IV-D.

C. EXPERIMENTAL SETUP

All algorithms are implemented on the OpenMV H7 Plus (OMV) with version 3.9 firmware. The OMV is a low-cost machine vision microcontroller with a built-in camera, Wi-Fi shield, and small form factor that is suitable for various robotics applications, especially applications with cost, mass, and size constraints [28]. The OMV has a maximum spatial resolution of 2592×1944 pixels (5 MegaPixels) and can use the ATWINC1500 Wi-Fi module for communication. For the external PC, we use a NVIDIA Jetson Xavier AGX (Jetson), which receives and processes images from the wireless OMV. To demonstrate the closed-loop ROI algorithm, we implement it for person detection/tracking. We test the closed-loop ROI algorithm with three different face detection neural networks: YoloV4, MobileNet, and MTCNN [31]–[33] that were optimized for the Jetson via TensorRT [38]. Object tracking is done by associating the locations of detected objects between frames using the intersection-over-union approach [35].

To quantify the performance of the closed-loop ROI algorithm, we use four different camera setups. For single person tracking, one person who walks three loops — the first slow, the remaining fast. For multi person tracking, two persons walked separate controlled paths. Figure 3 shows a visualization of each camera method, including ROI acquisition and prediction for the closed-loop ROI method. A video of the experimental setup and results is available.[1]

Setup 1) Default Camera: The Default Camera performs the normal camera operation, capturing the entire frame at a spatial resolution of 2592×1944 (5 MegaPixels). The PC receives the full image from the OMV and performs tracking-by-detection. A wide angle lens with 140° FOV is used to enable person tracking over the maximum possible region in the testing environment.

Setup 2) Pan & Tilt Camera: Scaling FOV and spatial resolution by the same amount results in constant image quality. A Pan & Tilt shield allows object tracking with a smaller FOV and thus, a smaller spatial resolution which improves temporal resolution. In this setup, the OMV is mounted on a Pan & Tilt shield. For object tracking, the difference between a detected object's location and the center of the FOV is used as the error for PID control of the Pan & Tilt shield. The OMV's

[1] Camera setups and operation: *https://youtu.be/OYW2WF3mKDA*

spatial resolution is set to 1280 × 1024 (1.3 MegaPixels) with the FOV scaled down from 140° to 60°, thus increasing the temporal resolution and allowing the OMV to capture a walking person without motion blur.

Setup 3) ROI Camera with Single Object: The ROI Camera is implemented with the closed-loop ROI algorithm. As in Setup 1, the OMV is set at maximum resolution with a 140° FOV lens. To track the person with an updating ROI, 0.06 MegaPixels ROI as determined in Section III-B.

Setup 4) ROI Camera with Multiple Objects: The ROI Camera is implemented with the closed-loop ROI algorithm for multi-object tracking. A closed-loop ROI is instantiated for each object. As in Setup 3, the OMV is set at maximum resolution with a 140° FOV lens and 0.06 MegaPixels ROI.

Test 1) Multiple, Multithreaded Cameras: In Section III-B, we derived that one OMV can track up to three objects with the closed-loop ROI method. To track additional objects, we test the feasibility of using multiple cameras with one external PC. Four TCP sockets are multithreaded on the external PC, each connected to a separate wireless microcontroller. A 0.06 MegaPixels ROI is sent continuously over each TCP socket.

Test 2) Trajectory Comparison: To ensure replicable moving trajectories for testing in different setups, we employed a 30 cm motorized arm setup that contains a pink paper attached to the end and was positioned 60 cm away from the camera.

IV. RESULTS AND DISCUSSION

The goal of the closed-loop ROI algorithm is to enable a wireless video stream at both high temporal and spatial resolutions for object detection/tracking. In general, object detection and tracking with a wireless camera consists of three levels: camera processing, data transmission, and PC processing. It is possible to apply the ROI to only one, two, or all levels. By exploring the characteristics of default camera streaming in Figure 4, the motivation for leveraging the ROI at each level becomes clear. At a high spatial resolution of 5.0 MegaPixels, there is a significant bottleneck in data transmission. By only transmitting a small ROI, this bottleneck is removed; in fact, transmitting a 0.5 MegaPixels ROI is actually faster than camera processing of a 5.0 MegaPixels image, making camera processing the new bottleneck. By only performing camera processing on a small ROI, this new bottleneck is removed. It remains unclear in Figure 4 if reducing spatial resolution for PC processing has any effect on temporal resolution; however, we later show in Section IV-C that the closed-loop ROI method does improve temporal resolution for some object detection algorithms. Each level of wireless video streaming is further discussed to characterize the effect of the closed-loop ROI algorithm.

A. CHARACTERIZATION LEVEL 1: CAMERA PROCESSING

As noted previously, the implementation of the closed-loop ROI algorithm may require adaptations depending on camera hardware. After updating the ROI on the OMV,

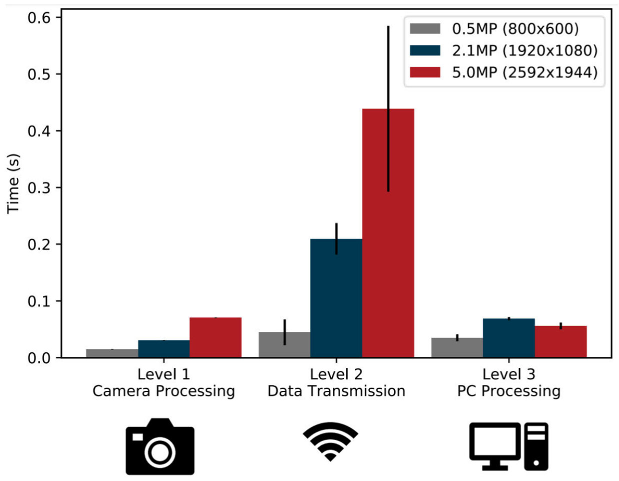

FIGURE 4. **Time consumption by level for object detection/tracking with a wireless video stream. Captured images are JPEG compressed. MTCNN is used for object detection in PC Processing.**

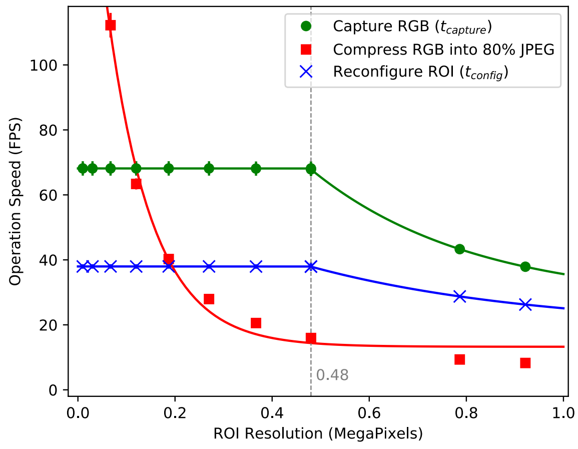

FIGURE 5. **Camera characteristics of the closed-loop ROI method for wireless video streaming and object detection/tracking.**

the subsequent frame may glitch. Thus, to reconfigure the ROI, the camera must change the ROI settings, then skip the subsequent frame. Additionally, the OMV limits the exposure time proportional to number of pixels captured. If less than 0.48 MegaPixels are captured, the low exposure time results in images with significantly lower luma (ITU-R 601-2). Thus, the minimum capturing window is set to 0.48 MegaPixels. To achieve a ROI with resolutions less than 0.48 MegaPixels, the OMV simply crops the ROI from the captured image before transmitting it. Finally, the OMV firmware captures and compresses images faster if the operations are combined; however, it does not permit the compressed images to be cropped. Since image cropping is required for ROI with resolutions less than 0.48 MegaPixels, the OMV must capture and compress images in separate steps.

The results of Figure 5 show how the implementation of the closed-loop ROI algorithm on the OMV affects characteristics on the camera processing level. At resolutions less than 0.48 MegaPixels, the capturing and reconfiguration speeds plateau because 0.48 MegaPixels is the minimum capturing window. However, image compression is only performed on the ROI–which is cropped from the capturing window. Thus, the compression speeds continues to increase at resolutions less than 0.48 MegaPixels.

According to Figure 5, at ROI resolutions less than 0.2 MegaPixels, reconfiguring the ROI becomes the most time-consuming step. In the experimental setup for person tracking, we used a 0.06 MegaPixels ROI, which is indeed less than 0.2 MegaPixels. Thus, instead of reconfiguring the ROI after every frame, we will reconfigure the ROI after every t_{loop} seconds.

Figure 6 shows the image capturing step combined with the ROI reconfiguration time. The figure is generated by capturing 100 images and reconfiguring the ROI after every t_{loop} seconds. $t_{loop} = inf$ seconds means that the ROI is never updated. Since the camera is only performing image capturing and nothing else, $t_{loop} = inf$ seconds is in fact repeating the Capture RGB curve from Figure 5. Conversely, $t_{loop} = 0.01$ s seconds means that the ROI is updated after every frame, since the time required to capture the smallest ROI is still longer than 0.01 s.

Figure 6 shows that a larger t_{loop} value dramatically increases the frame rate. A larger t_{loop} results in less time spent in reconfiguring ROIs and more time spent in capturing images, thus increasing the frame rate. However, as noted in Section III-B, a larger value of t_{loop} reduces that maximum object speed that can be tracked with an updating ROI. Thus, we select $t_{loop} = 0.1$ s to balance the image capturing frame rate and object tracking performance for single object tracking.

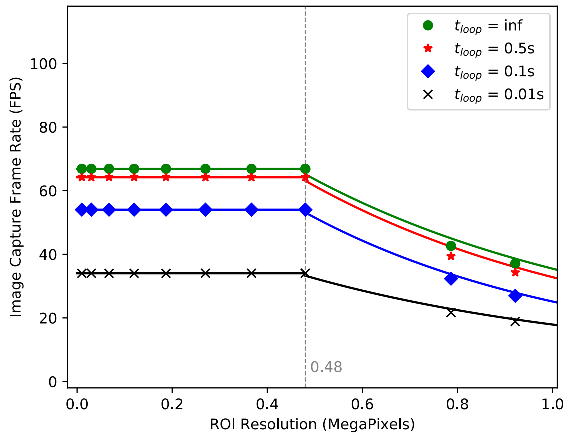

FIGURE 6. **Effect of time between reconfiguring ROI for detecting/tracking $n = 1$ objects on time required to capture ROI of image.**

In the experimental setup for person tracking, we use the minimum capturing window of 0.48 MegaPixels. According to Figure 4, camera processing for the 0.48 MegaPixels capturing window requires $t_{capture} = 0.015$ s. For a default camera streaming at maximum spatial resolution, camera processing requires 0.071 s. Thus, the closed-loop ROI method improves camera processing time by 4.7×.

B. CHARACTERIZATION LEVEL 2: DATA TRANSMISSION

The next step after camera processing is data transmission. Figure 7 shows that data transmission over USB is significantly faster than Wi-Fi data transmission over a TCP socket. The TCP protocol is a connection-oriented protocol, which reliably transmits data and includes error handling during transmission [39]. Thus, the closed-loop ROI method is especially useful for wireless video streaming and for cameras with high spatial resolution to reduce data transmission time.

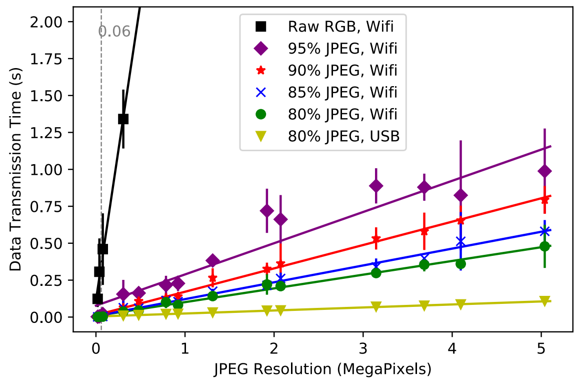

FIGURE 7. **Data transmission time of images at varying compression and spatial resolution.**

Due to the large sizes of raw images, it is not possible to transfer raw RGB images in real-time over Wi-Fi, even with the closed-loop ROI method. Thus, the image must be compressed prior to transfer. JPEG compression greatly decreases the image size, but the compression is not lossless. Fortunately, a high quality JPEG will generally not reduce the accuracy of convolutional neural networks (CNNs) for object detection [40]; the accuracy of CNNs decrease minimally above JPEG quality levels of 75 % [41]. Thus, we select 80 % quality level for our experimental setups which enables faster data transmission over Wi-Fi while maintaining detection accuracy.

In the experimental setup for person tracking, we use a 0.06 MegaPixels ROI. According to Figure 7, data transmission for the ROI requires 0.003 s when the image is JPEG compressed to 80 % quality. For a default camera streaming at maximum spatial resolution, data transmission for the entire FOV requires 0.48s. Thus, the closed-loop ROI method improves data transmission time by 160×.

C. CHARACTERIZATION LEVEL 3: PC PROCESSING

Once the ROI is transmitted to the external PC, the PC must perform object detection and determine the location of the next ROI. The closed-loop ROI algorithm can be used with a general neural network for detection, classification, segmentation, etc. given sufficient computational power on the PC. Figure 8 shows implementations of three different object detection neural networks with the closed-loop ROI algorithm, Though the algorithm reduces the size of the input image to the neural network, the inference speed does not necessarily become faster. The YoloV4 and MobileNet detec-

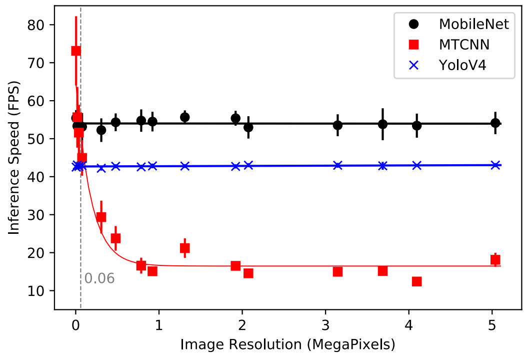

FIGURE 8. **Inference Speed of TensorRT accelerated object detection networks on NVIDIA Jetson Xavier AGX.**

tors have pre-defined input dimensions, requiring images to be resized to that input dimension. Thus, these detectors have constant inference times. In contrast, the inference time of the MTCNN detector drops dramatically as the image resolution decreases. During our experimentation, YoloV4 was more reliable and accurate at multi-person detection than MTCNN. Thus, we selected MTCNN for the experiment camera setups in Section III-C for single-object tracking to take advantage of the closed-loop ROI (0.06 MegaPixels) method at the PC Processing level; however, we selected YoloV4 for multiple-object tracking due to the higher multi-object detection accuracy.

In the experimental setup for person tracking, we used a 0.06 MegaPixels ROI. According to Figure 8, PC processing with MTCNN for the ROI occurs at 44.9 FPS. For a default camera streaming at maximum spatial resolution, PC processing with MTCNN occurs at 18.1 FPS. Thus, the closed-loop ROI method improves PC processing time by 2.5×.

D. RESULTS OF EXPERIMENTAL SETUPS AND TESTS

We implemented and compared the results of three camera streaming methods described in Section III-C: Default (in which the camera captures a wide-angle FOV at maximum spatial resolution), Pan & Tilt (in which the camera was mounted on a Pan & Tilt shield and captures a reduced FOV and spatial resolution), and closed-loop ROI (in which the camera captures a small ROI of a wide-angle FOV at maximum spatial resolution.

First, we compared differences in the tracked trajectories between the Default and closed-loop ROI methods to visualize differences in tracking. To do so, a color paper was rotated with a motorized arm and detected with a blob detection algorithm; this setup was adopted because the motion of the motorized arm can be replicated, allowing for comparisons of the tracked trajectories. The results in Figure 9 show that the closed-loop ROI method has a tracked trajectory closely matching the ground truth, which was captured at 30.2 FPS by connecting the OMV camera over USB. Both the ground truth and closed-loop ROI method show a smooth circular trajectory matching the rotation of the color blob on the motorized arm. Conversely, the lower temporal resolution of the Default method results in a discretized trajectory that deviates significantly from the ground truth.

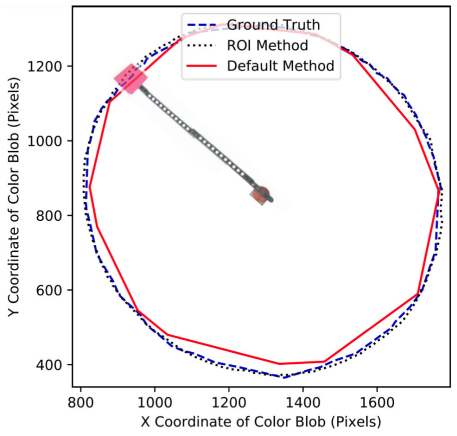

FIGURE 9. **Trajectory comparison between Default method and closed-loop ROI method of a tracked color blob. The tracked blob is a piece of pink paper attached to the end of a 30 cm motorized arm that is placed 60 cm away from the camera.**

The four camera streaming methods were implemented and recorded in an experiment for the person detection/tracking application in which the person(s) walked controlled paths that were mapped out in a testing environment. We quantitatively compared the different methods of wireless video streaming in Table 1 using metrics adapted from Graetzel *et al.* [12].

The closed-loop ROI method operates in real-time at 22.5 FPS, which is significantly faster than the Default and Pan & Tilt methods that operate at 2.4 FPS and 5.2 FPS, respectively. Implementing the closed-loop ROI method results in a 10× increase in temporal resolution over the Default method. Since the different camera streaming methods were evaluated on a person that walked the same three controlled loops, the number of frames that were recorded and analyzed varies due to the different frame rates.

For the Default method, the slow frame rate resulted in motion blur and caused false negatives in the person detection/tracking experiment. Only 71 % of frames had correct detections, resulting in 11 instances in which tracking was refreshed until the person was detected again in a subsequent frame. Due to the higher frame rates of the Pan & Tilt and closed-loop ROI methods, motion blur is not an issue; both methods correctly detected all people during the experiment.

For the Pan & Tilt method, the tracking is lost if the person moves out of the FOV, resulting in a tracking reset. In the experimental setup, the person walked the second and third loops fast, faster than the dynamic response of the Pan & Tilt camera. In both of these instances, tracking was lost until the person re-entered the FOV. Thus, the Pan & Tilt camera only spent 79 % of frames tracking the person. Due to the significantly higher FOV of the Default and closed-loop ROI methods, the person never walked outside the FOV; both methods captured the person in all frames during the experiment.

For the closed-loop ROI method, correct ROI predictions and optimal ROI dimensions enable the camera to only capture and transmit the ROI in 99 % of frames. The tracking is

TABLE 2. Person Detection/Tracking Metrics Comparison.

Camera Method	Default	Pan & Tilt	Closed-loop ROI Single Object	Closed-loop ROI Multiple Objects
Temporal Resolution (FPS)	2.4	5.2	22.5	17.5
# of Frames Analyzed	103	267	1048	991
Camera Field of View (°)	140	60	140	140
Camera Spatial Resolution (Pixels)	2592x1944	1280x1024	2592x1944	2592x1944
ROI Spatial Resolution (Pixels)	-	-	400x150	400x150
% Frames Without Motion Blur	71 %	100 %	100 %	100 %
% Frames Inside Field of View	100 %	79 %	100 %	100 %
# of Refreshed Tracking Instances	11	2	1	5
% Time Spent Tracking Object	71 %	79 %	99 %	98 %

For "Default", "Pan & Tilt", and "Closed-loop ROI Single Object" methods, one person walked the same controlled path in the testing environment. For "Closed-loop ROI Multiple Objects", multiple persons walked separate controlled paths. All camera methods were used to detect/track the person(s) as they walked the controlled path.

refreshed if the person moves out of the ROI before the ROI can be updated. However, the tracking is not lost. The ROI is simply configured to be the entire FOV until a new ROI is predicted. The results of Table 2 show that the closed-loop ROI method has the highest temporal resolution while set at maximum spatial resolution, and has the highest detection and tracking accuracy.

Setup 4) demonstrates multi-object tracking with the closed-loop ROI method. Each person is instantiated with a closed-loop ROI, and the OMV repeatedly switches between the ROIs and updates the ROIs separately. Every time the OMV switches between objects, the ROI position must be reconfigured, requiring t_{config} seconds. Thus, while single-object tracking operates at 22.5 FPS, multi-object tracking with two objects drops to 17.5 FPS. As in Setup 3), the tracking is refreshed if a person moves out of their ROI before the ROI can be updated. With two persons, it was observed that there were 5 instances of refreshed tracking during the experiment. However, the ROIs were immediately reconfigured, enabling the camera to only capture and transmit the ROIs in 98 % of frames.

In Section III-B, we showed that one OMV can track up to three objects with the closed-loop ROI method. To track more than three objects with the OMV using the closed-loop ROI algorithm, we recommend using multiple OMVs, each connected to the same external PC with multithreaded TCP sockets. Each wireless OMV requires one TCP socket to connect to the external PC in which a maximum of 65,535 TCP sockets can be opened [39]. Theoretically, with enough bandwidth in the network, multiple TCP sockets can be multithreaded on the external PC to communicate with multiple OMVs, each running its own instance of the closed-loop ROI algorithm. We tested this setup with up to four multithreaded TCP sockets, each connected wirelessly to a microcontroller continuously transmitting 0.06 MegaPixels ROIs. The additional TCP sockets did not cause any latency in the TCP connection; the closed-loop ROI algorithm operated at the same spatial and temporal resolutions with or without the additional TCP connections. Thus, multiple OMVs–each tracking up to three objects with the closed-loop ROI algorithm–can be used to track a large number of objects at high spatial and temporal resolutions.

In addition to enabling multi-object tracking, the multi-camera setup enables machine vision applications involving centralized control in swarm robotics. In this setup, all machine vision sensors connect and offload computation to one external PC. The external PC makes all scheduling decisions and control schemes for each individual agent; for the multi-camera setup of the closed-loop ROI algorithm, this involves predicting the location of the next ROI for each camera to be captured. As opposed to a distributed system, a centralized system has limited scalability and greater scheduling latency [14], [15]. The closed-loop ROI can improve scalability by significantly reducing data transmission of every machine vision sensor in the swarm.

V. CONCLUSION

In this paper, we present a general algorithm for wireless video streaming using a dynamically updated ROI to address the fundamental trade-off between spatial and temporal resolution in object detection and tracking. The comprehensive characterizations show that the closed-loop ROI algorithm can optimize the temporal resolution from three different levels: camera capturing, data transmission, and image processing while maximizing the spatial resolution without sacrificing the field of view. Future work may involve optimizing the closed-loop ROI algorithm, improving ROI prediction, and applying the algorithm to various robotics applications.

ACKNOWLEDGMENT

(Jack Chen and Hen-Wei Huang contributed equally to this work.)

REFERENCES

[1] M. Koziarski and B. Cyganek, "Impact of low resolution on image recognition with deep neural networks: An experimental study," *Int. J. Appl. Math. Comput. Sci.*, vol. 28, no. 4, pp. 735–744, Dec. 2018.

[2] J. Wang, C. Zhang, and H. yeung Shum, "Face image resolution versus face recognition performance based on two global methods," in *Proc. Asia Conf. Comput. Vis. (ACCV)*, Jan. 2004, pp. 48–49.

[3] R. Song, S. Zhang, J. Cheng, C. Li, and X. Chen, "New insights on super-high resolution for video-based heart rate estimation with a semi-blind source separation method," *Comput. Biol. Med.*, vol. 116, Jan. 2020, Art. no. 103535. [Online]. Available: https://app.dimensions.ai/details/publication/pub.1122642978

[4] P. Korshunov and W. T. Ooi, "Critical video quality for distributed automated video surveillance," in *Proc. 13th Annu. ACM Int. Conf. Multimedia (MULTIMEDIA)*, 2005, pp. 151–160.

[5] A. Fagg, "Why capture frame rate matters for embedded vision," Ph.D. dissertation, School Comput. Sci., Queensland Univ. Technol., Brisbane, QLD, Australia, 2018.

[6] A. Handa, R. Newcombe, A. Angeli, and A. Davison, "Real-time camera tracking: When is high frame-rate best," in *Proc. Eur. Conf. Comput. Vis.*, Oct. 2012, pp. 222–235.

[7] M. Jurik, V. Smidl, J. Kuthan, and F. Mach, "Trade-off between resolution and frame rate for visual tracking of mini-robots on planar surfaces," in *Proc. Int. Conf. Manipulation, Autom. Robot. Small Scales (MARSS)*, Jul. 2019, pp. 1–6.

[8] K. Debattista, K. Bugeja, S. Spina, T. Bashford-Rogers, and V. Hulusic, "Frame rate vs resolution: A subjective evaluation of spatiotemporal perceived quality under varying computational budgets," *Comput. Graph. Forum*, vol. 37, no. 1, pp. 363–374, Feb. 2018.

[9] H. Meuel, F. Kluger, and J. Ostermann, "Region of interest coding for aerial surveillance video using AVC & HEVC," Oct. 2015, *arXiv:1801.06442*. [Online]. Available: https://arxiv.org/abs/1801.06442

[10] J. Gao and Z. Jin, "A region of interest prediction method for real-time online dynamic magnetic resonance imaging," in *Proc. 10th Int. Congr. Image Signal Process., Biomed. Eng. Informat. (CISP-BMEI)*, Oct. 2017, pp. 1–5.

[11] J. Tian, Z. Wang, and Q. Zhu, "An improved lane boundaries detection based on dynamic ROI," in *Proc. IEEE 9th Int. Conf. Commun. Softw. Netw. (ICCSN)*, May 2017, pp. 1212–1217.

[12] C. F. Graetzel, B. J. Nelson, and S. N. Fry, "A dynamic Region-of-Interest vision tracking system applied to the real-time wing kinematic analysis of tethered drosophila," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 3, pp. 463–473, Jul. 2010.

[13] J.-Y. Kim, C.-H. Yi, and T. Y. Kim, "ROI-centered compression by adaptive quantization for sports video," *IEEE Trans. Consum. Electron.*, vol. 56, no. 2, pp. 951–956, May 2010.

[14] J. Hu, A. Bruno, D. Zagieboylo, M. Zhao, B. Ritchken, B. Jackson, J. Y. Chae, F. Mertil, M. Espinosa, and C. Delimitrou, "To centralize or not to centralize: A tale of swarm coordination," 2018, *arXiv:1805.01786*. [Online]. Available: http://arxiv.org/abs/1805.01786

[15] M. Schranz, M. Umlauft, M. Sende, and W. Elmenreich, "Swarm robotic behaviors and current applications," *Frontiers Robot. AI*, vol. 7, p. 36, Apr. 2020. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2020.00036

[16] Y. Xie and G.-Q. Han, "ROI coding with separated code block," in *Proc. Int. Conf. Mach. Learn. Cybern.*, Aug. 2005, pp. 5447–5451.

[17] D. Grois and O. Hadar, "Complexity-aware adaptive spatial pre-processing for ROI scalable video coding with dynamic transition region," in *Proc. 18th IEEE Int. Conf. Image Process.*, Sep. 2011, pp. 741–744.

[18] Z. Nan and Q. Xu, "Micro-object detection by dynamic ROI combined with pyramid template matching," in *Proc. 37th Chin. Control Conf. (CCC)*, Jul. 2018, pp. 5445–5450.

[19] Y. Niu, X. Wu, and G. Shi, "Edge-based dynamic ROI coding with standard compliance," in *Proc. IEEE Int. Workshop Multimedia Signal Process.*, Oct. 2009, pp. 1–6.

[20] Y. ZhiWei and X. Xiao, "Dynamic region of interest extract method for JPEG2000 coding," in *Proc. Int. Conf. Comput. Commun. Technol. Agricult. Eng.*, Jun. 2010, pp. 150–153.

[21] P. Rahmiati, A. Fajri, A. Handayani, T. L. R. Mengko, A. B. Suksmono, and J. T. Pramudito, "Distributed system for medical image transfer using wavelet-based dynamic RoI coding," in *Proc. 7th Int. Workshop Enterprise Netw. Comput. Healthcare Ind. (HEALTHCOM)*, Jun. 2005, pp. 191–196.

[22] X. Cheng, K. Cheng, and H. Bi, "Dynamic downscaling segmentation for noisy, low-contrast *in situ* underwater plankton images," *IEEE Access*, vol. 8, pp. 111012–111026, 2020.

[23] L. Feng, L.-M. Po, X. Xu, Y. Li, C.-H. Cheung, K.-W. Cheung, and F. Yuan, "Dynamic ROI based on K-means for remote photoplethysmography," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 1310–1314.

[24] H. K. Kalluri, M. V. N. K. Prasad, and A. Agarwal, "Dynamic ROI extraction algorithm for palmprints," in *Advances in Swarm Intelligence*, Y. Tan, Y. Shi, and Z. Ji, Eds. Berlin, Germany: Springer, 2012, pp. 217–227.

[25] W. Damak, R. Trabelsi, M. Alima, and D. Sellami, "A new dynamic ROI extraction method for hand vein images," *IET Comput. Vis.*, vol. 12, no. 5, pp. 586–595, Aug. 2018.

[26] K. Yano, K. Ishihara, M. Makikawa, and H. Kusuoka, "Detection of eye blinking from video camera with dynamic ROI fixation," in *Proc. IEEE SMC Conf. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 1999, pp. 335–339.

[27] *Top 10 Machine Vision Camera Providers Named in Study*. Accessed: Apr. 30, 2021. [Online]. Available: https://www.vision-systems.com/home/article/14168882/top-10-machine-vis%ion-solution-camera-providers-named-in-study

[28] I. Abdelkader, Y. El-Sonbaty, and M. El-Habrouk, "OpenMV: A Python powered, extensible machine vision camera," Nov. 2017, *arXiv:1711.10464*. [Online]. Available: https://arxiv.org/abs/1711.10464

[29] *Introducing Pixy2*. Accessed: Apr. 30, 2021. [Online]. Available: https://pixycam.com/pixy2/

[30] *Raspberrypi Camera Module V2*. Accessed: Apr. 30, 2021. [Online]. Available: https://www.raspberrypi.org/products/camera-module-v2/

[31] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," Apr. 2020, *arXiv:2004.10934*. [Online]. Available: https://arxiv.org/abs/2004.10934

[32] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: https://arxiv.org/abs/1704.04861

[33] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Process. Lett.*, vol. 23, no. 10, pp. 1499–1503, Oct. 2016.

[34] H. Karunasekera, H. Wang, and H. Zhang, "Multiple object tracking with attention to appearance, structure, motion and size," *IEEE Access*, vol. 7, pp. 104423–104434, 2019.

[35] E. Bochinski, V. Eiselein, and T. Sikora, "High-speed tracking-by-detection without using image information," in *Proc. 14th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, Aug. 2017, pp. 1–6.

[36] M. P. Muresan and S. Nedevschi, "Multi-object tracking of 3D cuboids using aggregated features," in *Proc. IEEE 15th Int. Conf. Intell. Comput. Commun. Process. (ICCP)*, Sep. 2019, pp. 11–18.

[37] A. K. A. Osep and L. Leal-Taixw, "EagerMOT: Real-time 3D multi-object tracking and segmentation via sensor fusion," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Apr. 2021, pp. 1–4.

[38] J. K. Jung. (2013). *Tensorrt Demos*. [Online]. Available: https://github.com/jkjung-avt/tensorrt_demos

[39] *IBM Documentation*. Accessed: Apr. 27, 2021. [Online]. Available: https://www.ibm.com/docs/en

[40] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *Proc. 8th Int. Conf. Qual. Multimedia Exper. (QoMEX)*, Jun. 2016, pp. 1–6.

[41] M. Poyser, A. Atapour-Abarghouei, and T. P. Breckon, "On the impact of lossy image and video compression on the performance of deep convolutional neural network architectures," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 2830–2837.

JACK CHEN is currently pursuing the B.A.Sc. degree in engineering science with the University of Toronto, Canada.

He was a Research Assistant with the Teaching Systems Laboratory, Massachusetts Institute of Technology, and the Parker Laboratory, California Institute of Technology. He is currently a Research Assistant with the Division of Gastroenterology, Brigham and Women's Hospital, Harvard Medical School, Boston, MA.

HEN-WEI HUANG received the B.S. and M.S. degrees in mechanical engineering from National Taiwan University, Taiwan, in 2011 and 2012, respectively, and the Ph.D. degree in robotics technology from ETH Zürich, in 2018.

He is currently a Research Fellow with Brigham and Women's Hospital, Harvard Medical School, Boston, MA, and the Koch Institute for Integrative Cancer Research, Massachusetts Institute of Technology. His research interests include *in vivo* wireless sensor networks, personalized medicine, MEMS, robotics, and ingestible/implantable electronics.

PHILIPP RUPP received the B.S. degree in information technology and electrical engineering from ETH Zürich, Switzerland, in 2019, where he is currently pursuing the M.S. degree in information technology and electrical engineering.

He is currently a Research Assistant with the Division of Gastroenterology, Brigham and Women's Hospital, Harvard Medical School, Boston, MA. His research interests include machine learning for biomedical applications, brain–computer interfaces, computer vision, and nanophotonic devices.

ANJALI SINHA is currently pursuing the B.S. degree in electrical engineering and computer science with the Massachusetts Institute of Technology, Cambridge, MA.

She is currently a Research Assistant with the Traverso Laboratory, Massachusetts Institute of Technology. Her research interests include robotics, autonomous systems, computer vision, and machine learning.

CLAAS EHMKE received the B.Sc. degree in information technology and electrical engineering from Technical University Munich, Germany, in 2017, and the M.Sc. degree in robotics, systems, and control from ETH Zürich, Switzerland, in 2021, where he is currently pursuing the Ph.D. degree in robotics with the Multi-Scale Robotics Laboratory.

He was a Research Assistant with the Koch Institute for Integrative Cancer Research, Massachusetts Institute of Technology, and worked on autonomous driving estimation algorithms with the Singapore-MIT Alliance for Research and Technology Centre and AMZ Racing.

GIOVANNI TRAVERSO received the B.A. degree from the Trinity College, University of Cambridge, U.K., in 1998, and the Ph.D. degree from Johns Hopkins University, in 2002. He subsequently completed medical school at the University of Cambridge, an Internal Medicine Residency with Brigham and Women's Hospital, and the Gastroenterology Fellowship Training with Massachusetts General Hospital, Harvard Medical School.

He is an Assistant Professor with the Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, and also a Gastroenterologist with the Division of Gastroenterology, Brigham and Women's Hospital, Harvard Medical School. His current research program is focused on developing the next generation of drug delivery systems to enable safe and efficient delivery of therapeutics as well developing novel ingestible electronic devices for sensing a broad array of physiologic and pathophysiologic parameters. Additionally, he continues his efforts towards the development of novel diagnostic tests that enable the early detection of cancer.