AI Smart Security Camera for Person Detection, Face Recognition, Tracking and Logging

Diego Rodriguez Garcia, Alvaro Aparicio Serna, David Sancho Crespo, Xinrui Yu 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*—Although the surveillance industry is growing due to increased concerns for public safety, traditional surveillance systems can be ineffective due to fixed positions, limited coverage, and the need for human operation. This paper proposes an edge computing surveillance system that can detect, track, and identify human presence using AI software. The system includes a video camera integrated into a structure that allows pan and tilt movements and a processing unit that performs all computations locally. The system can operate in two different modes, manual and automatic, and it can be controlled from a user-friendly web application. The manual mode allows the user to control the camera remotely, adjusting the position with buttons and handling additional functionalities like zoom. On the other hand, in automatic mode, the system detects people and identifies if they are an unknown subject or not. Identifications are recorded in an online log and the system takes pictures of unknown people to recognize recurrent intruders.**

Keywords—Edge Computing, IoT, Smart Security Camera, Tracking, Detection, Face Recognition, Computer Vision

I. Introduction

Security cameras are vital for public safety, however, traditional models have limitations like fixed positions, blind spots, and the requirement for human intervention. This paper introduces an advanced edge computing surveillance system equipped with pan and tilt capabilities, that addresses those limitations using AI software for smart human detection, tracking, and identification, all processed locally for real-time efficiency. This approach aligns with recent advancements in person tracking and recognition at a distance, emphasizing the importance of edge computing in real-time analytics [1][2]. This paper specifically focuses on home security applications, a market currently valued at USD 7.4 billion and projected to reach USD 30 billion by 2030 [3]. Our system targets this expanding niche surpassing traditional closed-circuit television (CCTV) and exploiting the shift towards IoT and smart camera technology.

With Edge Computing, our system allows for real-time tracking and face recognition without the computational and latency challenges that affect similar technologies. By integrating AI with IoT connectivity, our smart security camera provides not only remote motion control and live video streaming but also maintains a cloud-based log of known and unknown detected subjects and their frequency of detection. A detailed explanation of the different functionalities is provided in the Implementation section.

II. Technical Description

This paper proposes a smart camera that uses AI resources to detect people in the frame and zoom in on their faces to perform facial recognition. The computations are performed locally on an edge computing platform. The camera is part of a larger structure that can move with two degrees of freedom, allowing the camera to track the movement of the person in the frame. The system also includes a web application that can be used as the HMI of the system. From the app, the user will be able to watch the surveillance video and input commands. Users can also view a log of the detected subjects over time. The system has two modes of operation: manual and automatic modes. Their functionalities will be explained later.

A. System Architecture

We can differentiate four main nodes in the system: the HMI, the main processing module, the log of detections, and the smart camera including the set of sensors and actuators. An overview is given in Fig.1. As HMI we designed a web application where the user can input motion, zoom, and focus commands while viewing the live video footage. The main processing module is the single-board computer in charge of processing video. This is one main difference concerning other products in the market, as most of them process the video footage on a server. Our choice of a localized processing unit over cloud-based processing models is supported by recent research advocating for the efficiency of edge computing in surveillance applications, particularly in face recognition and control of PTZ (Pan-Tilt-Zoom) cameras [4] [5]. The processing module also translates the user commands received from the web application into PWM signals for controlling the pan, tilt, and zoom micro-stepper motors that are part of the smart camera. Meanwhile, the video is received from the camera and streamed into the web application in real time. Additionally, the computing platform stores the log data in the cloud database.

Overall, user commands flow from the web application to the camera actuators. On the other hand, the information captured with the camera lens flows backward to the processing module until finally reaching the web application.

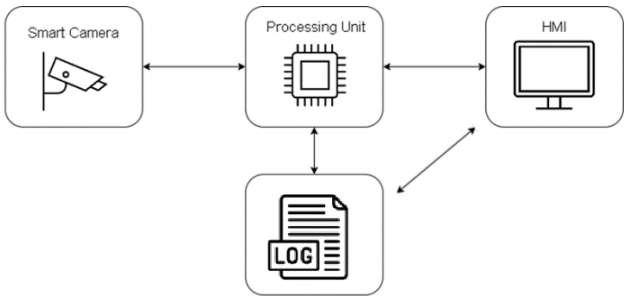

Fig. 1. Overview of the System Architecture.

B. Manual Mode

In this mode, accessible through a user-friendly web application interface with intuitive buttons, users can watch the video feed and control the operation of the system remotely. This mode allows users to control pan and tilt movements via two servo motors, adjusting the camera's angle for expanded surveillance coverage. Additionally, users can use analog zoom controlled by another servo motor for closer examination of any area under surveillance. The focus feature enables manual adjustment of the lens focus to enhance image quality if it appears blurry. An IR-cut filter can also be leveraged to facilitate infrared light passage through the lens, improving vision in low-light conditions; this requires an infrared light source. Moreover, users have the option to activate the automatic mode, enabling the system to operate independently, thus freeing them to focus on other tasks.

C. Automatic Mode

In automatic mode, the system runs the detection, tracking, and face recognition algorithms while logging detected subjects. The camera no longer responds to manual input commands from the user. Instead, the motion and zoom commands are computed automatically to track the detected people. The tracking algorithm runs in parallel for both detection and face recognition, computing the necessary motion commands so that the detected subject always remains centered on the frame (by calculating the distance between the center of the subject's face and the frame's center). Two detection algorithms run concurrently, one detects bodies to generate the annotations for tracking and the other detects frontal faces to trigger the face recognition mode.

By default, the system starts running detection and tracking and when it finds a stable face during several consecutive video frames, the camera uses optical and digital zoom to get a closer image of the face and switches to face recognition and tracking. The subject is classified as either authorized or unauthorized. During this mode, the results of the identification are shown in the video in real time. The footage displayed on the web application is the processed video including bounding boxes and labels around objects and faces.

Additionally, the system logs detected subjects, including their labels and detection timestamps, in a cloud database accessible through the web application. For recurrent unknown subjects, the camera captures their faces and stores them in a local database to identify repeated intruders. Lastly, users also have the flexibility to switch back to manual mode at any time.

III. IMPLEMENTATION

A. Hardware Implementation

The Arducam camera is the main sensor used in this paper, as it contains the image sensor, the servo motors (for pan and tilt, zoom, and focus), the optical zoom-in capabilities, and the structure in a single component. Specifically, we are using the Arducam IMX477 12MP PTZ Camera, which has a 12.3MP sensor with a pan and tilt range of 0° to 180°.

During the writing of our paper, we evaluated two processing units, the Jetson Nano and the Jetson Orin Nano, to assess their performance. According to the manufacturer, the Jetson Orin Nano demonstrates considerable improvements over the Jetson Nano, especially in AI performance, computation capabilities, and CPU efficiency, facilitating the execution of advanced AI models with greater performance and energy efficiency. These enhancements include a 6.6-fold increase in CPU performance and better performance per watt, indicating a leap in computational power and energy efficiency. Fig. 2 summarizes the performance comparison metrics.

In terms of communication protocols, between the user device and the Jetson board, which hosts the web application, the communication is based on the HTTP protocol. Meanwhile, between the Jetson board and the Arducam PTZ controller, to send the necessary commands to move the camera, the I2C protocol is used. The PTZ controller is connected to the Jetson Nano through 4 wires. There is a 5V and ground connection to power the camera, and SDA and SCL cables for serial communication using the I2C protocol. The power supply used provides 5V with up to 4A, and the motors are connected through a breakout cable to two jumper wires. The connections of the Arducam with the Jetson Nano are described in Fig. 3.

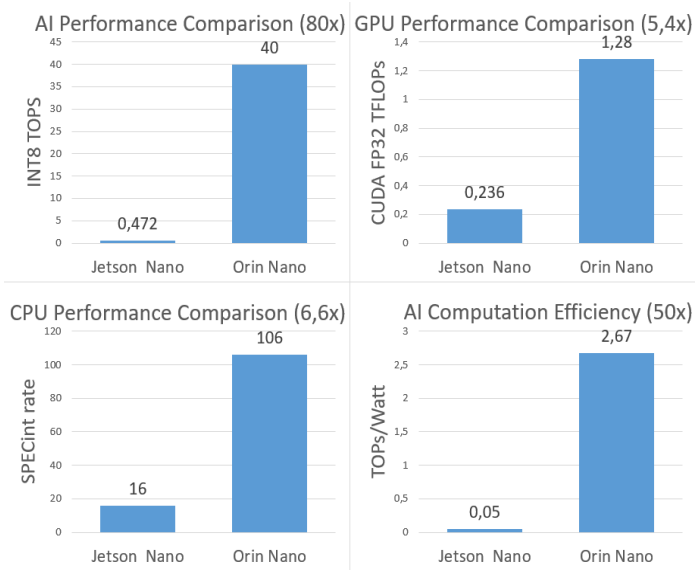

Fig. 2. Jetson Nano vs. Jetson Orin Nano performance metrics [6].

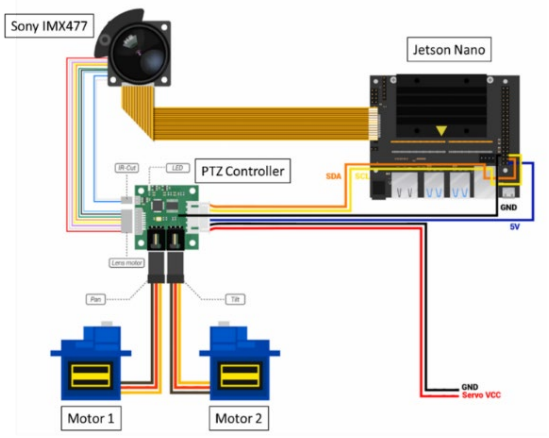

Fig. 3. Hardware Schematic of the System [7].

B. *Camera Software*

The Arducam is programmed using functions provided by the developers. The codes for the IMX477 PTZ model are found in [8]. This code is written in Python 3 and is based on the OpenCV library. The GitHub repository [8] includes several Python functions and scripts that can be used to control the Arducam in different ways. The most relevant are "Focuser.py" and "JetsonCamera.py". The JetsonCamera class handles the video connection between the Arducam and the Jetson Nano using GStreamer.

C. *Cloud Log Software*

For cloud logging, we have developed a database based on SQL hosted in AWS RDS. Amazon Web Services Relational Database Service simplifies the setup and operation of online databases and it also provides cloud storage. In addition, we have used another service from AWS called Quicksight that allows users to create interactive dashboards to visualize data from an existing AWS database. Finally, we have used the mysql.connector Python package to link our code to the online database and send the log data for storage. This log includes a timestamp and an identification label.

D. *AI Software Implementation*

The most important part of the system is the automatic mode. For this mode of operation, we have developed a Python program capable of detecting people, tracking their movement, and performing facial recognition. The methodologies employed here draw upon principles from [9] [10], which have shown the effectiveness of deep learning algorithms in enhancing surveillance systems' accuracy in people detection and tracking.

For detection, we have used two sets of algorithms, one for person detection and one for frontal face detection. The first one provides the annotations for tracking and the second one is used to trigger the face recognition mode. For person detection, we have studied two different models YOLOv5 and MediaPipe's pose landmarks. YOLOv5 [11], a single-stage object detector, consists of a backbone, neck, and head. The backbone utilizes a Cross Stage Partial (CSP) Network for feature extraction, while the neck employs a Path Aggregation Network (PANet) for creating feature pyramids to manage objects of various sizes. The head produces final outputs, including class probabilities, objectness scores, and bounding box coordinates, with the model pre-trained on the COCO dataset [12]. For real-time performance, we opted for YOLOv5n (nano), filtering for people detections despite its ability to recognize over 80 classes. MediaPipe's Pose Landmark model [13] integrates a detector-tracker system based on the BlazePose model with a MobileNetv2-like backbone and GHUM for pose landmark prediction. Initially, a lightweight detector identifies human figures in an image, and then a tracker refines and tracks body landmarks. This setup leverages MobileNetV2, optimized for mobile devices with depthwise separable convolutions for efficiency and accuracy. Additionally, GHUM enhances 3D body pose estimation by providing depth and spatial details in images or videos. For face detection, the program uses the OpenCV library. More precisely, it employs a pre-trained Haar Cascade classifier that is included in this library [14]. This classifier uses a special type of kernel to extract Haar-like features for training. The result is tens of thousands of features that are reduced to the most useful by using the optimizer AdaBoost. Then, the algorithm applies a cascade classifier that divides the remaining features into stages. For a face to be detected in an image, this image has to fulfill all the stages because, if one of the stages fails the image is discarded.

For face recognition, the system uses the face recognition package [15] from the Dlib library of Python. This algorithm creates a histogram of gradients (HOG) in an image and applies a Support Vector Machine (SVM) to detect faces. These detected faces are encoded using a version of ResNet-10 and compared to the images in a dataset computing the Euclidean distance. The dataset stores images of the authorized users, so if there is a match the code shows the confidence percentage of the identification and the name of the subject. The implementation of real-time face detection and recognition resonates with findings from [16] [17].

For tracking, the program computes the pixel difference between the center of the reference point and the center of the frame. The reference point varies depending on the person detection algorithm used, but it is roughly located on the face of the detection to ensure a seamless transition from detection to face recognition mode. When using MediaPipe's pose landmarks the reference point corresponds to the subject's nose landmark. When using YOLOv5 the reference point is located on the upper-middle region of the bounding box. The difference between the reference point and the center of the frame is the input of a simple proportional controller that outputs the new angle of the camera after every frame. Then, the motors are set to the new position using the functions from the Focuser.py class of Arducam.

The code operates in an infinite loop, capturing and processing frames with two modes dictated by a "mode" flag (0 or 1). In mode 0, person and face detection run in parallel using YOLO or MediaPipe for person tracking and a pre-trained Haar Cascade classifier for face detection. If a face is consistently detected, the system zooms in, activates optical zoom, and switches to mode 1 for face recognition using Dlib, marking unknown faces as intruders and storing them in a database. Mode 1 still carries out tracking, enhances face recognition with maximum optical zoom for better accuracy, up to a distance of

20 feet, and reverts to mode 0 for broader surveillance when no face is detected. This dynamic tracking and recognition process, detailed in Fig. 4, ensures efficient monitoring and identification.

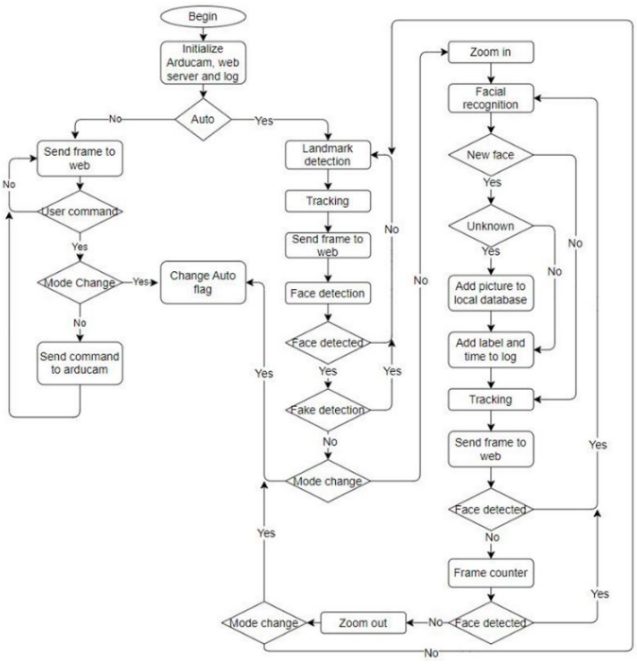

Fig. 4. System flowchart.

E. *Night Vision Implementation*

The Arducam features an IR-cut filter for accurate color capture during daylight and includes functionality to remove the filter via software for potential night vision use, aligning with our security camera paper's goals. We have tested the AI software's performance with the IR-cut disabled and an infrared light source, which is necessary to fully utilize this night vision capability.

F. *Web Application Implementation*

The Human-Machine Interface (HMI) of the system is a web able to stream live video locally extracted from the camera connected to the Jetson Nano. This code uses the HTTP modules Python library, a package that assembles multiple modules to ease the development of software that uses the HTTP protocol. This library is based on Python. For the web app structure of the human-machine interface, the program uses HTML. In addition, the code includes the Bootstrap package to shape the design of the buttons. Bootstrap is a front-end toolkit that contains prebuilt grid systems and components to use in HTML pages. For the logging database feature, the web application includes a button in automatic mode that redirects users to a Quicksight page, displaying logged subjects and timestamps. Fig. 4 shows the detailed system flowchart.

IV. RESULTS

This report presents an edge computing Smart Security Camera that can be interfaced remotely through a web application and used in manual and automatic modes. Our system allows users to view the real-time video footage remotely, control the camera's point of view and angles, and let the system perform automatic people detection and face recognition up to 20 feet while tracking the subjects to be in the frame. Fig. 5 shows the hardware setup for the system. The Jetson Nano is the main processing module, which is connected to the Arducam.

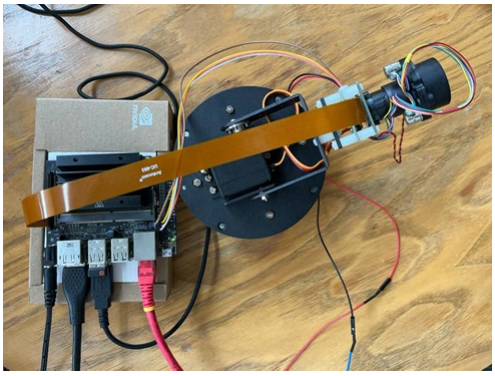

Fig. 5. Smart security camera prototype hardware.

A. *Web Application Results*

Figures 6 and 7 show a view of the web application user interface in both manual and automatic modes. As you can see in manual mode the user sees the raw video footage and can make use of the buttons to: move the camera angle, use the optical zoom, change the focus of the lens, take off the IR-cut filter, and activate the automatic mode. On the other hand, in automatic mode, the user sees the processed video footage with bounding boxes and subject labels around detected faces. In Fig. 7 you can see the body landmarks drawn over the subject, which is the result of MediaPipe's detector. In Fig. 8 there is a labeled bounding box that identifies the subject according to the authorized people dataset. The AI software manages to recognize people with over 90% confidence at a distance of up to 20 feet without false positives. The automatic mode website also includes a button to open the security log in the Quicksight website. The appearance of this log is shown in Fig. 9.

Fig. 6.Web application in manual mode.

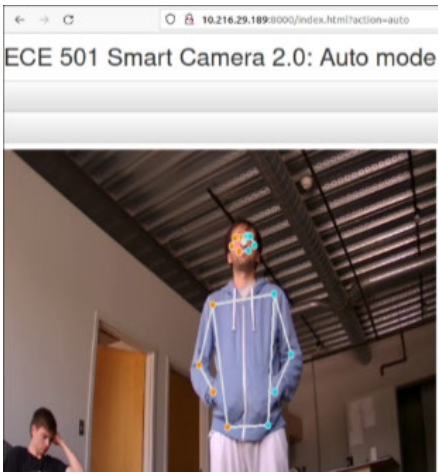

Fig. 7. Web application in automatic mode

Fig. 8. Result of face recognition.

Field wells Rows log_time (SECOND) label

Sheet 1

Count of Records by Log_time and Label

log_time	label	Co
Jun 18, 2023 12:25pm	David.jpg	1
Jun 18, 2023 12:25pm	Intruder0.jpg	1
	Unknown	1

Fig. 9. Security log.

B. *Night Vision Results*

The user can command the IR-cut filter, which prompted us to conduct tests under low-lighting conditions to evaluate its functionality. An infrared light was flashed while disabling the filter, leading to a remarkable enhancement in the captured image quality compared to when the filter was enabled. This effectively produced a night vision image. It's essential to note that the IR-cut filter alone does not yield any significant improvements without the complementary use of an infrared light source. The images presented to the right depict the outcomes obtained from applying the detection and face recognition algorithms to night vision images. For reference, at the top of Fig. 10, we show the normal image taken by the Arducam in a dark environment. The rest of the pictures show the images that the Arducam can capture without the IR-cut filter when the infrared light illuminates the surroundings.

Fig. 10. Night vision image results.

As the images show, MediaPipe's landmark detector and YOLOv5 object detection are quite robust and work very well in this situation. However, face recognition does not work as well, because it needs more details than the previous algorithms. The night vision images are blurrier than daylight images, so for face recognition to work it needs to see a very clear face like in the first example.

C. *Performance Comparison*

In the course of writing this paper, we have worked with the Jetson Nano and the Orin Nano as alternative processing units. In addition, we have used 2 alternatives for person detection, YOLOv5 and MediaPipe. So, we have taken note of several metrics to compare the performance of the different algorithms in the two single-board computers. The results using the Jetson Nano are shown in Table 1. Dlib works very smoothly while Mediapipe and YOLO CPU are very slow, because these algorithms are more demanding at the time of inference. However, it is important to note that the face recognition AI

solution only processes one out of every two frames. While the other solutions process every frame, this also affects the results. We can also clearly see an improvement when YOLO uses the GPU instead of the CPU because the prior is more suitable to run AI software. The results using the Orin Nano are shown in Table 2.

TABLE I. JETSON NANO PERFORMANCE RESULTS.

Jetson Nano	FPS	CPU/GPU	Inference time	Total Power
Dlib	20	70%/10%	N/A	5600 mW
Mediapipe	2	80%/20%	N/A	5500 mW
YOLO CPU	3	90%/30%	280 ms	6900 mW
YOLO GPU	5.5	25%/90%	138 ms	6650 mW

TABLE II. JETSON ORIN NANO PERFORMANCE RESULTS

Jetson Orin Nano	FPS	CPU/GPU	Inference time	Total Power
Mediapipe + Dlib	13	25%/25%	N/A	4600 mW
YOLO CPU	1	99%/0%	750 ms	7075 mW
YOLO GPU	22	25%/75%	28 ms	8300 mW

The MediaPipe library is still in development, and they still do not include GPU support for their AI solutions. Therefore, the performance of the pose landmark detector is not as good as it could be, but it is still way better than in the Jetson Nano due to the improved CPU capabilities in the Orin Nano. The YOLO CPU in the Jetson and the Orin are both slower than their GPU counterparts. Especially, in the Orin Nano the improvement in using the GPU to run YOLO is great, due to the increased capabilities of the Orin. The YOLO GPU in the Jetson Nano and the Orin Nano use the same code so it is very clear how the Orin Nano is immensely superior to the Jetson in FPS and inference time, while using a smaller percentage of their respective GPUs. However, this improvement also brings a higher power consumption.

V. CONCLUSIONS

This work presents an edge computing Smart Camera for surveillance capable of detecting, tracking, and identifying human presence through face recognition in real time. The system can be controlled remotely through a web application and used in manual and automatic modes. The web application allows users to view the real-time video footage remotely, and control the camera's point of view and angles. It also makes the system perform automatic people detection and face recognition up to 20 feet while tracking the subjects to be in the frame. The system is based on an ArduCam IMX477 12MP PTZ camera and uses a Jetson Nano single-board computer as the main processing module to control the servomotors for the camera, run the computer vision algorithms, stream video to the web application and respond to remote user commands. We conducted tests with night vision images, yielding satisfactory results and demonstrating the system's adaptability to various lighting conditions.

Finally, this paper examines the feasibility of deploying the system on different boards, comparing the performance of the Jetson Nano against the Jetson Orin Nano. The findings highlight the importance of utilizing software instructions to effectively leverage the hardware's computational power and GPU for optimal model inference.

REFERENCES

[1] U. Park, H.-C. Choi, A. K. Jain, and S.-W. Lee, "Face Tracking and Recognition at a Distance: A Coaxial and Concentric PTZ Camera System," IEEE Transactions on Information Forensics and Security, vol. 8, no. 10, pp. 1665–1677, Oct. 2013.

[2] Vassilis Tsakanikas and Tasos Dagiuklas, "Enabling Real-Time AI Edge Video Analytics," Research Open (London South Bank University), Jun. 2021.

[3] Grand View Research "Smart Home Security Camera Market [2023 global report]," GVR. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/smart-home-security-camera-market. [Accessed: 30-Apr-2023].

[4] V.-A. Dao, D.-H. Nguyen, V.-B. Nguyen, Thom Tran Thi, and Hoang-Anh Nguyen The, "Face Recognition System for Unconstrained Condition," Oct. 2023.

[5] M. S. Al-Hadrusi and N. J. Sarhan, "Efficient Control of PTZ Cameras in Automated Video Surveillance Systems," Dec. 2012.

[6] "Develop AI-Powered Robots, Smart Vision Systems, and More with NVIDIA Jetson Orin Nano Developer Kit", NVIDIA Technical Blog, https://developer.nvidia.com/blog/develop-ai-powered-robots-smart-vision-systems-and-more-with-nvidia-jetson-orin-nano-developer-kit/ [accessed Jun. 22, 2023]

[7] "Arducam Quick start," Quick Start - Arducam Wiki, https://docs.arducam.com/Nvidia-Jetson-Camera/Pan-Tilt-Zoom-Camera/quick-start/ [accessed Jun. 22, 2023].

[8] ArduCAM, "ArduCAM/MIPI_CAMERA," GitHub, https://github.com/ArduCAM/MIPI_Camera [accessed Jun. 22, 2023].

[9] Y. Zhou, "Deep Learning Based People Detection, Tracking and Re-identification in Intelligent Video Surveillance System," Aug. 2020.

[10] J. Chen, K. Li, Q. Deng, K. Li, and P. S. Yu, "Distributed Deep Learning Model for Intelligent Video Surveillance Systems with Edge Computing," IEEE Transactions on Industrial Informatics, pp. 1–1, 2019.

[11] Ultralytics, "Ultralytics/yolov5: Yolov5," GitHub, https://github.com/ultralytics/yolov5 [accessed Jun. 22, 2023].

[12] T.-Y. Lin et al., "Microsoft COCO: Common Objects in Context," arXiv (Cornell University), May 2014.

[13] Google, "Mediapipe/docs/solutions/pose.md at master · google/mediapipe," GitHub, https://github.com/google/mediapipe/blob/master/docs/solutions/pose.md [accessed Jun. 22, 2023].

[14] "Face detection using Haar Cascades," OpenCV, https://docs.opencv.org/3.4/d2/d99/tutorial_js_face_detection.html [accessed Jun. 22, 2023].

[15] A. Geitgey, "Face-recognition," PyPI, https://pypi.org/project/face-recognition/ [accessed Jun. 22, 2023].

[16] X. Zhang, W.-J. Yi, and Jafar Saniie, "Home Surveillance System using Computer Vision and Convolutional Neural Network," May 2019.

[17] X. Zhang, T. Gonnot, and J. Saniie, "Real-Time Face Detection and Recognition in Complex Background," Journal of Signal and Information Processing, vol. 8, no. 2, pp. 99–112, May 2017.