Person-Tracking Camera

Authors: Nathan Levin, Jerry Ding, and Karthik Natarajan
Electrical and Computer Engineering, Carnegie Mellon University

Abstract — **Our system is a person-tracking security camera made to help stores and homeowners protect their properties. To do this, our security camera will zoom and track such that a suspicious person's actions can be clearly seen on the camera footage. Our system will improve upon existing tracking security cameras by using advanced machine learning algorithms and by being a self-contained edge device.**

Index Terms — **AdaFruit, Camera, DeepPhi, Deep Learning, Inference, Power, Security, Tracking, Servo Motors, Yolov3, Arduino C**

I. Introduction

It is not uncommon to see video surveillance systems being installed on street lights, houses and small storefronts. Though they can be effective at detecting trespassers or following the motion of a known target, it can be difficult to obtain a clear view of a person, even with a high video resolution and a modest distance between the camera and the target. With a fixed mount camera one often has no choice but to use a large field of view to avoid having blind spots in their camera system, resulting in objects in view being too small to reliably identify. However, a zooming camera can obtain high resolution images of individual targets of interest without permanently restricting the field of view of the camera. With recent research in neural computer vision, advanced algorithms like Yolov3 can be used to reliably identify and locate people within view, making an automatic person tracking camera feasible.

Currently, automatic person tracking security cameras are rare in the marketplace, and the few that exist are usually high-end PTZ cameras that require a complex setup process and a central server for computation. These PTZ cameras typically do not use deep neural networks, and as a result the tracking feature is not very reliable.[1] We plan to create a better solution: a self-contained security camera that uses highly accurate machine learning algorithms to successfully zoom into a person.

[1] https://ipvm.com/reports/should-you-use-autotracking-ptzs

II. Design Requirements / Metrics

The target audience we decided to focus on are typical homeowners and stores. These users are more likely to find an automated, no-hassles security solution appealing. To guide our design specifications, we offer a formal description of our product as follows:

A compact and self-contained security camera that automatically tracks and zooms into any potentially suspicious person, and that an average store or homeowner can easily install and use.

A. Convenience Requirements

We want a compact and self-contained system to ensure that the installation process is not too difficult. We also require the camera to automatically zoom, so that targets within view can be identified. To be able to hone into any person within view, the camera would need to pan and track while it is zoomed in.

To quantify the compactness and self-contained properties, we decided to establish these design requirements:

The system must not require a central server. It should support both plugged-in and battery mode operation. Ideally, on one charge it will be able to continuously track people and capture footage for up to 500 minutes, or remain operational for up to 30 days while waiting for people to enter the field of view.

The battery mode operation enhances the ease of installation, as an average homeowner or store may find it inconvenient to run power cables to their front door. The running time requirements are based on similar commercial security camera products.[2] These strict runtime requirements on a limited power source make power consumption a preeminent concern.

B. Tracking Requirements

The tracking specifications are as follows:

[2] https://www.amazon.com/Battery-Powered-Security-Wireless-Wire-Free/dp/B07HH6Z357/

With up to three people approaching between 5 and 20 feet of the front door, with an equal probability of approaching and leaving in any direction at any time of the day, and assuming ample lighting, the person-tracking security camera should fail to zoom and track each person motion less than one time in 50 trials on average. When tracking properly, the person's height should take around 80% of the camera frame height.

The figure is motivated by the frequency of theft faced by homeowners and stores. Specifically, we found that the average household receives around 27 packages per year[3], and that shoplifting incidents typically fall around 18 incidents per outlet per year[4]. Most of these incidents don't involve more than two criminals, so we chose to test with three people in view.

The 20 foot benchmark is based on the typical length of a driveway, since it is less likely for a person beyond that distance to be a meaningful target. Since the zoomed in view can only accommodate one person, we would have to alternate between tracking different people in view and schedule the camera's viewing time to minimize the probability of missing people less than 20 feet away. We choose to support down to a minimum distance of 5 feet as a reasonable range where the person will occupy most of the field of view without being too close to see.

Other requirements include a 720p 30fps video footage recording capability. This resolution and frame rate is standard for many modern security cameras and webcams. Tracking should be done at least 10 fps time resolution, which allows the tracker to see at least one frame of a person no matter how fast they run. We would also need our servos to have a high enough rotation rate to track a moving person from 5 feet away, but given that our servos are capable of 60 degrees of rotation in less than half a second, this is should not be challenging to meet.

The zoom metric was based on the idea that we want to see the actions the suspicious person is taking. To do this, we cannot be fully zoomed into the person's face because if we do this we will be unable to see what the person is doing with the rest of their body. So, by fixing the person's height to be around 80% of the frame height, we will have full view of the suspicious person's body and will therefore be able to see their actions.

C. Testing Plan

To test these requirements, we will place the system on a table near the back door of Fairfax Apartments, because the view there is similar to that of small shop or townhouse, complete with a sidewalk, a street, and a corridor that can treated as a driveway. We will test the system with these scenarios:

- Individual person walking at a brisk pace from a random direction, reach a closest approach of between 5 feet and 20 feet, waiting for two seconds, then leaving in a random direction, at night or at the daytime. (50 trials)
- Three simultaneous people each independently choosing a path as above. (50 trials)
- Take random frames after the camera has finished zooming and check to see that the person takes up around 80% of the frame.

[3] https://www.inc.com/john-white/tired-of-getting-your-packages-stolen-heres-what-to-do.html
[4] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.182.6338&rep=rep1&type=pdf

III. Architecture

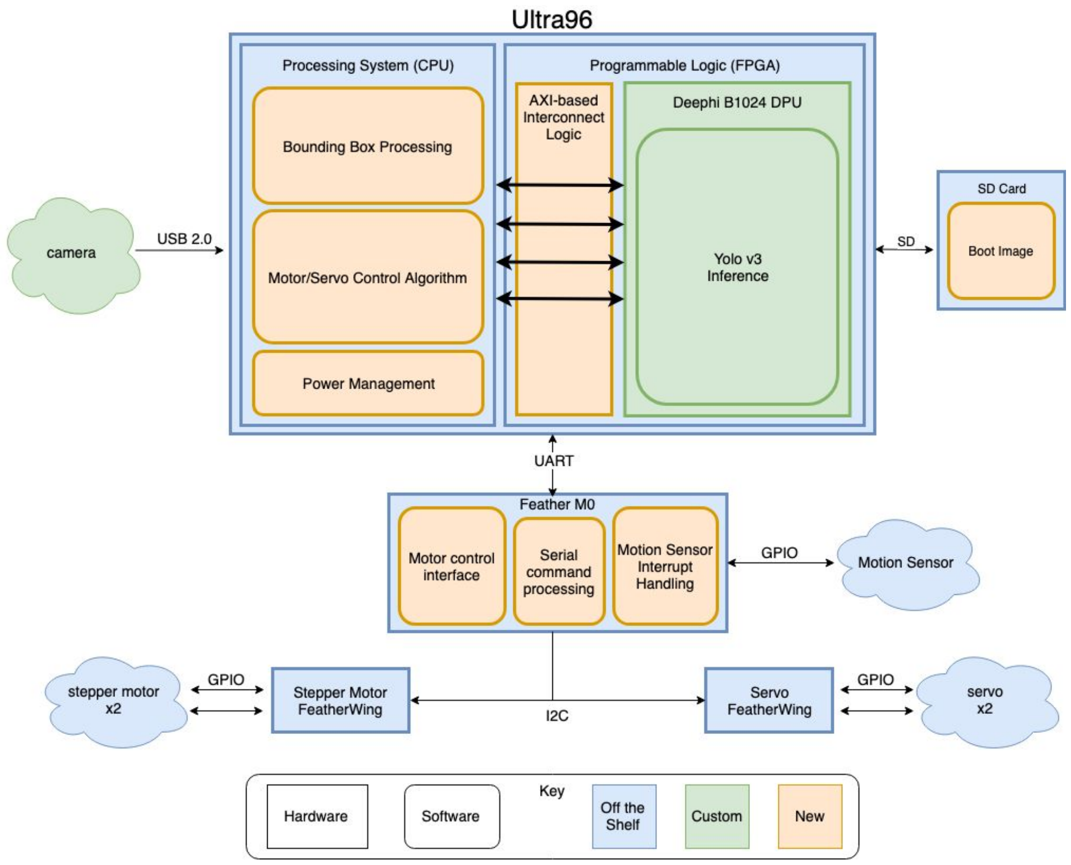


Fig 1. Block diagram of hardware architecture and arrangement of software subsystems.

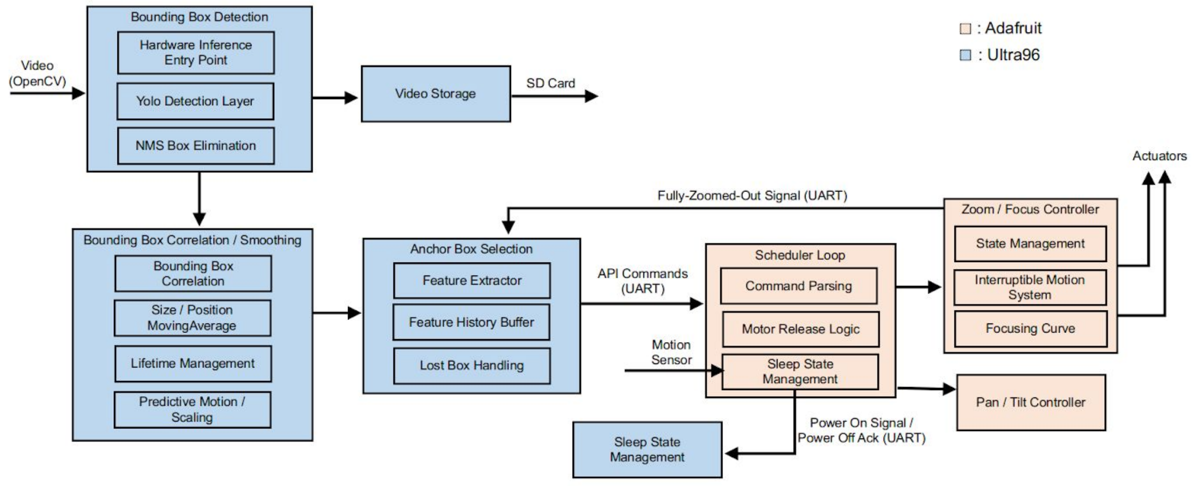


Fig 2. Block diagram of software architecture and general flow control

The foundation for our approach to human identification and tracking is deep learning inference accelerated by the programmable logic/FPGA portion of the Xilinx MPSoC on the Ultra96 board. Xilinx's solution to this problem, referred to as "edge inference" in the industry, is the use of their newly acquired subsidiary Deephi, a company which created a deep learning accelerator architecture implemented on Xilinx FPGAs to enable fast, low power deep learning inference.

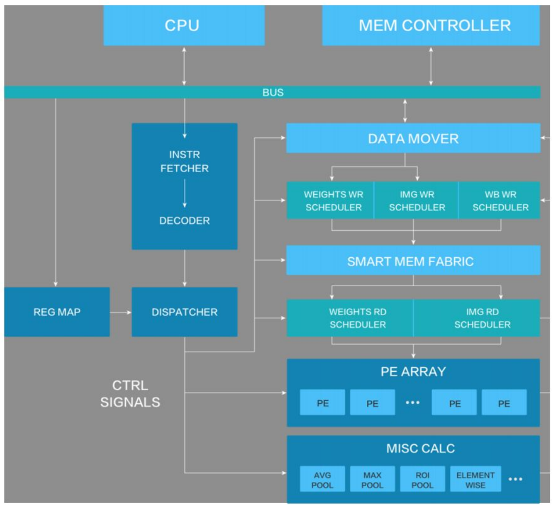


Fig 3. Architecture of the Deephi Aristotle inference accelerator

We chose this architecture because it is new (Deephi was acquired in July 2018), allowing room for experimentation beyond established use cases, and it benefits from Xilinx's robust documentation, making it possible for amateurs to work with. It is additionally configurable allowing for customized approaches to minimize power while meeting our performance targets.

Deephi/Xilinx not only provide the hardware, but they also provide a software ecosystem in the form of the Deep Neural Network Development Kit (DNNDK). This includes tools to compress and optimize a neural net (via DECENT), and then compile it to work with their accelerator architecture (via DNNC). This includes native support for Yolov3, one of the leading object (including human) identification models.

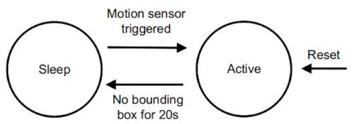


Fig 4. State diagram for the system

A high level state diagram of the full system is shown in Fig. 4. In summary, we will need to support a sleep mode where the system is not recording footage to the SD card and is not using the FPGA fabric for inference.

In the sleep state, the Ultra96 board will be suspended, and the Adafruit Feather M0 will monitor the motion sensor to determine when to wake the Ultra96. Though the Adafruit board will occasionally perform some processing, it will usually be in an idle state that only consumes less than 30mW[5]. The Ultra96 chip consumes about 35 mW[6], and the RAM self-refresh cycle consumes up to 40mW. Together, the power consumption will be around 100mW, which is on track to give us a 30 day idle battery life.

There are likely to be a significant number of false positives from the motion sensor alone, however. To prevent fully powering on the board on each false positive, we include an intermediate low power mode which only uses a subset of the Ultra96 board's capabilities to run simple image processing algorithms at a low frame rate. The goal is for this stage is to conservatively filter out likely sources of false positives such as cars and wind, ensuring that the board usually powers fully on only when a person enters the view. We believe that by tuning the sensor and Ultra96 board to minimize the energy spent in this mode, we can obtain a 30 day idle battery life even considering false positive detections.

Finally, in the active state, the board will begin recording footage to the on-board SD card and use neural algorithms to generate a bounding box for people in view. We estimate that this mode will consume between 5 and 15 watts, which allows the camera to operate for 500 minutes.

The software block diagram is shown in Fig 2. The software components can be divided roughly into two groups: an image processing group and a sensorimotor group. Image processing code is implemented on the Ultra96 and sensorimotor code is implemented on the Adafruit Feather M0. Communication between the two will be done via UART.

Low power person detection is implemented by comparing the current video frames with a

[5] https://electronics.stackexchange.com/questions/52991/is-the-cortex-m0-really-low-power/52995
[6] https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842181/Zynq+UltraScale+MPSoC+Power+Off+Suspend

background image. The background image is obtained with a temporal low-pass filter, which averages the on-screen pixels over a long period of time (fully resetting the background image if the camera has suddenly moved, which suddenly changes the pixels in the view). Using these frames and the background image, it is relatively easy to compute crude features such as the speed and apparent size of the moving object, which can be filtered to exclude obvious false positives.

When the low power person detector finds a probable person, it will bring the board to active mode. A priority scoring system will use features such as the size and motion of each detected bounding box to choose which bounding box to focus on at any given point in time. The bounding box information is used by the Adafruit Feather M0 in the motor controller, which uses code written with the Arduino C stepper and servo libraries to move the camera. Similarly, an autofocus algorithm will be implemented in the Feather board to adjust the stepper motors in the focuser.

IV. Design Trade Studies

A. DPU Power Consumption

Our most important design tradeoff for this project was the balance between the capabilities of the DPU and its power consumption. We had a target inference rate of 10fps, and initially expected to optimize to just hit this frame rate, so as to minimize the power of the system (thus extending its runtime) within that tracking requirement. What we discovered, however, is that the capabilities of the DPU with our optimized neural net far exceeded our expectations. The default Ultra96 implementation ran in excess of 30fps, which is the maximum frame rate supported by the 1080p video stream we were reading in from the camera. Therefore, we predicted that our performance would almost certain surpass our use case's requirement, and our effort would be best spent minimizing the DPU configuration, and thus its power consumption.

Xilinx provided a pre-build Ultra96 image with an instantiation of the Deephi B1152, which is what we initially expected to have to work with. However, during the course of the project they posted the IP repository for the Deephi accelerator core. Despite being nominally compatible only with a different Xilinx product, we were able to modify the IP core in Vivado (Xilinx's hardware design tool) to reduce its configuration to a lower power, yet still extremely capable end result.

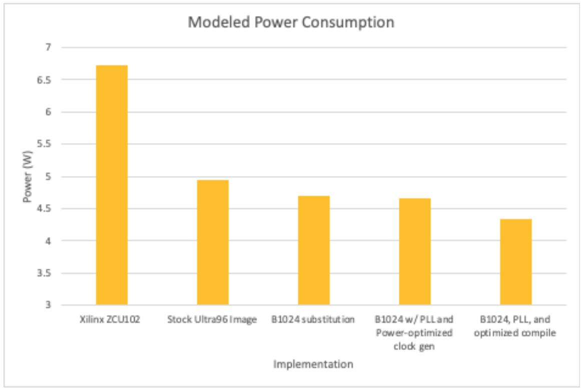


Fig 5. Vivado-derived modeled power consumption of Deephi DPU implementations. Power totals include SoC processing system (hard IP cores).

To derive an optimal configuration for our system, we first started by compiling results for Xilinx's ZCU102 image (where the IP was initially pulled from), and then with the stock configuration present on the Ultra96.

The ZCU102, as a substantially larger implementation of the DPU, unsurprisingly has power consumption far in excess of the Ultra96, but the Ultra96 configuration itself still consumed substantial power. With that in mind, we strove to cut down the DPU configuration to one better suited to a power-constrained environment. The first step was to drop the DPU configuration from the default B1152 to the B1024 design, which is the lowest end configuration supported by our neural network compiler tools (DNNDK). Give then 3x margin by which our measured performance exceeded our requirement, we predicted this lesser DPU configuration to have an inconsequential impact on performance.

This change reduced power consumption by ~0.24W, which is not insignificant, but we thought we could push the gains further. Optimizing the clock generating logic in Vivado, including replacing the extraneous MMCM clock generation with more efficient PLL-based clock generation, yielded an additional ~40mW improvement. However, it was by combining these improvements with a power optimized compilation step that more significant gains were seen, further dropping the power by approximately 0.32W, for a total of >12% power reduction compared to the default configuration.

However, with these changes in place, we needed to verify that the performance of our system was still within specification. Upon testing with the completed system, our inference frame rate with this reduced configuration still exceeded 25fps, a limitation imposed by the CPU-bound storage of the video stream.

We additionally used Deephi profiling tools to verify the DPU utilization was not limiting the system performance.

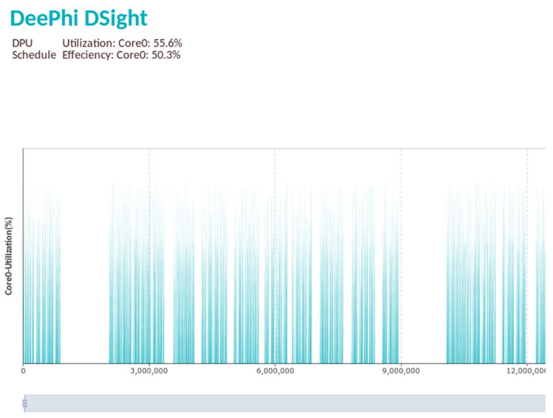


Fig 5. Profiled utilization numbers for minimized DPU configuration.

With utilization of the smallest configuration failing to reach even 60%, it's clear that the DPU is not the limiting component of the system, despite all of the effort put into minimizing its capabilities. Furthermore, the Xilinx tools suggested that with this configuration, the system's power consumption is dominated by the processing system (hard IP cores), particularly the CPU, and further optimization of the DPU core could only provide marginal gains at best.

Unable to provide substantial further power reduction, we instead chose to use this excess DPU performance to improve tracking accuracy by averaging over multiple frames, while still remaining well above the 10fps requirement for our ultimate tracking frame rate.

V. System Description

Our solution is built hierarchically and contains two main subsystems. Below is a detailed description of the implementation choices in each component.

A. Ultra96 Subsystem

1. Bounding Box Detection

a. Hardware Inference Entry Point

This component uses DeePhi's DNNDK framework to instantiate a DPU kernel for the Yolov3-tiny neural network, generate DPU tasks from OpenCV images collected from the camera, and submit the tasks to the DPU for inference.

One undocumented detail of DeePhi's DNNC compiler used to create the Yolov3-tiny kernel is how it handles input data. It assumes each pixel in the image is encoded with a floating point number between 0 and 255, while DarkNet trains the neural network assuming a range of 0 to 1. Therefore the Caffe model was modified to match the changed data format.

b. Yolo Detection Layer

The logic for the output layer of the Yolov3-tiny architecture cannot be implemented in the DPU, so it had to be re-implemented in software. We did not use a library for this - instead, we read the Yolov3 research paper to reproduce the definition of this layer.

Within this layer, there are a number of tunable parameters that affect the quality of bounding box detection, most notably a confidence threshold between 0 and 100 for marking a bounding box as a detected person. We tuned the confidence value to find the lowest threshold before the network drew bounding boxes around non-human objects in the testing lab (25), and settled on a value of 33 to leave a safety buffer.

2. Bounding Box Correlation / Smoothing

The neural network treats each frame of the video independently, and therefore explicit processing is needed to establish a temporal link between bounding boxes across frames (e.g. to track one person without being distracted by the presence of another). In addition, the size and position of the boxes tend to fluctuate across frames, which can strain the motor system if the fluctuations translate into rapid back and forth motions. This is why a bounding box smoothing system is included.

a. Bounding Box Correlation

To identify which boxes in a new frame correspond to which boxes in previous frames, we compute the intersection over union (IoU) between all bounding boxes for a new frame and the set of smoothed boxes for the previous frame. We establish that a new box is temporally linked to a past box if the IoU exceeds 0.33, and if a single box is temporally linked to multiple past boxes then the oldest (which is arguably the more robust) past box is chosen for the correlation.

b. Size / Position Moving Average

Smoothing is implemented with an exponential moving average of the position and the size of the boxes across frames. For each frame, this smoothing is only applied to bounding boxes for which a correlation was found in the bounding box correlation step. The weights used for the averaging is roughly equivalent to averaging three frames for the box position, and averaging 12 frames for the box size.

c. Lifetime Management

There can be times when one of the smoothed bounding boxes for the previous frame is not correlated with any boxes for the current frame. This occurs either when the person in the box leaves the field of view, or when the person in the box is briefly occluded. For these boxes where no correlation is found, a staleness counter is incremented. Otherwise, if a correlation is found the staleness counter is decremented, stopping at 0. Boxes with a staleness greater than 15 (equivalent to 0.625 seconds with no correlation) are removed. An age variable is kept track for all live boxes, and only boxes with an age greater than 10 frames are used for the anchor box selection system.

With this system, boxes will be retained when a person is briefly occluded, and will be removed when a person leaves the field of view.

d. Predictive Motion / Scaling

In addition, boxes with no correlation are speculatively moved and scaled according to the changes observed in the previous frame for up to 5 frames. This speculative motion improves the IoU score once a bounding box is re-established, making it more likely that the bounding box will be retained after a brief occlusion.

3. Anchor Box Selection

The system generally chooses one box in view to be the anchor, and periodically changes the anchor to focus on multiple targets. This system is responsible for this functionality.

a. Feature Extractor

When the system needs to choose a new anchor box, a number of features are extracted from the pixels within each bounding box in view. These features are the average values of the R, G, and B channels for 8 equal-sized vertical strips and 4 equal-sized vertical strips from the box.

b. Feature History Buffer

A buffer of 5 feature vectors from past anchor boxes are used to decide on a new anchor box. The new anchor box chosen will be the one whose features are the most dissimilar to the closest match among the features in the history buffer, where similarity is measured with the L2 distance metric.

c. Lost Box Handling

When the anchor box is lost, a grace period of one second is given for the system to find a new box. If found, the new box is re-established as the anchor. Otherwise, the Ultra96 sends a signal to fully zoom out the view before finding a new anchor box (where a fully-zoomed-out signal from the Feather M0 is used to determine when zoom is restored to default). This helps handle up intermittent periods where the anchored person could not be detected.

4. Video Storage

A persistent file is used to store a video index, which is incremented every time the system boots up. This index helps create a unique video file on each boot. A multithreaded MJPG video encoder from OpenCV is used to generate the video file.

5. Sleep State Management

When the Ultra96 subsystem detects no bounding boxes for 20 seconds, it sends a signal to the Feather M0 to enter the shut down state. When an ACK message is received, the Ultra96 powers itself down. When the Feather M0 decides to wake the Ultra96, a signal is sent directly to the low speed expansion header to power on the Ultra96 board.

B. Feather M0 Subsystem

1. Scheduler Loop

a. Command Parsing

Commands are single capital letters followed by an integer parameter. Characters are buffered until a newline character is found.

b. Motor Release Logic

When no zoom commands are received for 500ms, the Feather M0 releases the stepper motors to save power.

c. Sleep State Management

When the Ultra96 subsystem sends a shutdown command to the Feather M0, the latter replies with an ACK, releases the servos and the stepper motor, and enters the sleep state where it only monitors the motion sensors for waking up..

2. Zoom / Focus Controller

a. State Management

No sensors are available to measure the position of the zoom and focus stepper motor, so this component keeps track of the current step counts for the zoom and focus steppers explicitly. When the system fully zooms out, a signal is sent to the Ultra96 via UART.

b. Interruptible Motion System

Moving the zoom and focus steppers takes time, and performing a large zoom change all at once will leave the Feather unresponsive due to a lack of multithreading. Instead, zoom changes are implemented as a sequence of small steps toward a target. Each small step moves both the zoom and focus steppers to maintain a focused view, and the derivative of the focusing curve is used to keep the amount of time spent in each step roughly constant.

If the scheduler receives a command for a zoom in the opposite direction of the current target, the target is immediately moved to be in the opposite direction.

c. Focusing Curve

The focus stepper was manually calibrated at 13 zoom levels to find the proper focus vs. zoom curve for a target around 20 feet away, and a 5th degree polynomial was fitted to these points to interpolate between the 13 zoom levels.

We observe that the clarity of a target did not depend very much on the distance of the target, and so a fixed focus vs. zoom curve can be used without fine-tuned adjustment.

3. Pan-Tilt Controller

This component controls the pan-tilt servos for the camera mount. Since a target position is directly encoded as a PWM signal of the Feather, the system remains fully responsive while the servos are moving without extra effort.

Neural Network Details

The Yolov3-tiny neural network is trained from scratch with the COCO and Caltech Pedestrian datasets, adding to about 70,000 images. Some filtering is done to remove very small (< 30 pixel height) bounding boxes in the ground truth data. In addition, data augmentation is used to make the system more robust to people partially in view. In particular, 10 percent of the images were duplicated and modified in one of these three ways:

- Cropped from above to hide part of a person's upper body
- Cropped from below to hide part of a person's lower body
- Blurred to simulated imperfect focus

Finally, around 1000 images from the NYU Depth v2 dataset are used as negative examples in the training set. These images have a variety of objects but no people in view.

Hardware component choice criteria

Ultra96 - Chosen for its exceptional performance relative to its price and power consumption. The leading edge TSMC 16nm FinFET process was particularly appealing as a power-focused project, and the Ultra96 is one of the few boards compatible with Xilinx's new Deephi inference accelerator ecosystem at this time. Additionally, the integrated GPU makes for easier development and demonstration compared to alternative boards, and the Xilinx ZU3 SoC used is strong enough for real time video inference with a variety of models.

SEN0192 Motion Sensor - Chosen for its range, reliability, affordable price point, and flexible form factor. These criteria make it useful for our custom system, which like any such security device, demands reliability.

Adafruit Feather M0 Basic Proto - Chosen for it's low idle power (<10mA) and compatibility with Adafruit FeatherWing expansion boards and the Arduino software libraries, enabling easier motor control than porting the code to the Ultra96 or writing our own.

Logitech C920 Pro - Chosen for its relatively low price, decent video quality, and it's compatibility with the lens adapter used to connect our zoom lens with our Camera's PCB.

ServoCity SPT200 Pan & Tilt Kit + HS-485HB servos - Chosen for high durability and the strength to support weighty attachments to the camera, such as the lens and modified webcam with the lens adapter.

SL-27135MFZ Motorized Zoom Lens- This zoom lens has a magnification factor of 5 while also being relatively cheap, in this case only $11. Also, because the average driveway is around 20 ft in length, the zoom lens can zoom into a person so they appear only 4-5 ft away thereby giving us a good view of the people near your house or storefront.

Software component choice criteria

Yolov3 - We chose this machine learning model because of its native support through the Deephi tool, very wide use, and ease of customization geared towards improved human detection. Neural Network is used to help us track people while in active mode.

OpenCV - We can use this for miscellaneous image processing, such as the code used for the low power person detection algorithm and priority scoring.

DNNDK - Part of the Xilinx/Deephi ecosystem, and designed to optimize neural nets for low precision (8 bit int), low power inference. It is therefore ideal for our use case.

Arduino C- We chose Arduino C for 2 specific reasons. Firstly, Arduino C has very well documented libraries for both servo motors and stepper motors which will make it easy to use with our motors. On top of that, Arduino takes very little space which is good for us because our board only has 32 KB of RAM.

VI. Project Management

A. Schedule

As you can see, Fig. 6 has the schedule of tasks throughout the semester. As of right now, our schedule has had some minor changes, but nothing that puts us way behind schedule.

The main significant change was the time for ordering and shipping the parts for our project. This had to be pushed forward into early March because we had many changes in the overall design of our system as we went through different use cases and tried to solidify our metrics. And, because we have not gotten all of our parts, we have only been able to verify that the parts have arrived.

Outside of that, everything else has been going to schedule. Specifically, on the machine learning side, we have been able to extract and customize the Yolov3 network to track people. And, on the hardware side, we have started to interface with the with the board through GPIO while waiting for the remaining parts.

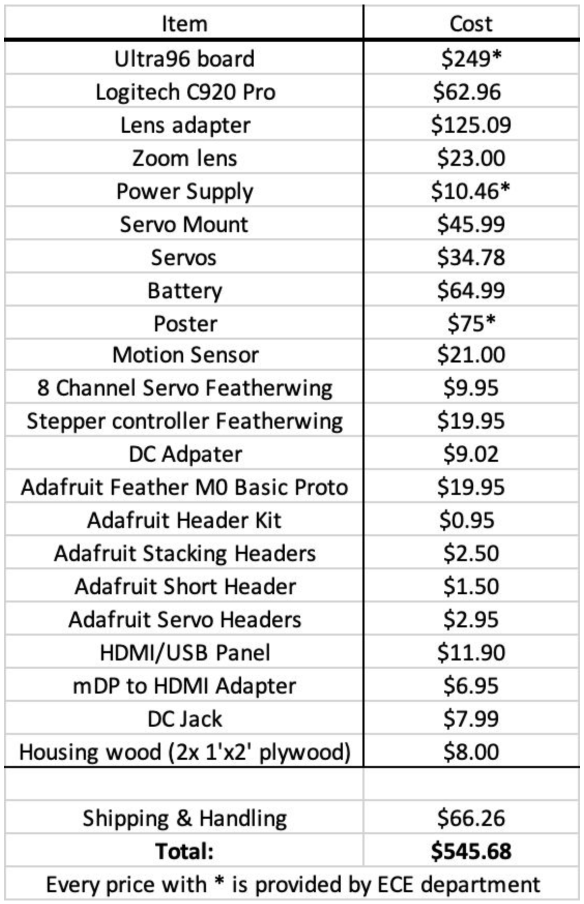
B. Team Member Responsibilities

For this project, we tried to divide the responsibilities so that we can parallelize the most amount of work. So, each of us are working on a different area of the project. Specifically, Jerry is doing most of the machine learning tasks like setting up the ML model and customizing the Yolov3 network because he has access to a GPU which gets rid of the need for AWS credits. Then, Nathan’s tasks mostly involve working with the Ultra96 board and trying to reduce the amount of power that is used by the board. This was decided based on his level of knowledge with the Ultra96 and its components. Finally, Karthik is working on the controllers and the logic for allowing devices to interface with the board. This was decided based on previous class experience with making controllers.

While these general descriptions help indicate how we assigned the bigger tasks for the project, we all have the secondary responsibility to integrate each of our individual parts to make sure they can work together as one system. Therefore, when we want to unit test the entire system, we all work together to make sure the individual parts are functioning properly.

C. Budget

Fig. 7 shows us the overall budget usage for the project.



Item	Cost
Ultra96 board	$249*
Logitech C920 Pro	$62.96
Lens adapter	$125.09
Zoom lens	$23.00
Power Supply	$10.46*
Servo Mount	$45.99
Servos	$34.78
Battery	$64.99
Poster	$75*
Motion Sensor	$21.00
8 Channel Servo Featherwing	$9.95
Stepper controller Featherwing	$19.95
DC Adpater	$9.02
Adafruit Feather M0 Basic Proto	$19.95
Adafruit Header Kit	$0.95
Adafruit Stacking Headers	$2.50
Adafruit Short Header	$1.50
Adafruit Servo Headers	$2.95
HDMI/USB Panel	$11.90
mDP to HDMI Adapter	$6.95
DC Jack	$7.99
Housing wood (2x 1'x2' plywood)	$8.00
Shipping & Handling	$66.26
Total:	**$545.68**
Every price with * is provided by ECE department	

Fig 7. Budget detailing all the materials we are using

D. Risk Management

Our primary risk management technique, and our most successful one by far, was carefully planning the entire project from beginning to end. By laying out the basic structure and function of each subsystem, we were able to ensure that there were minimal hiccups in the process, and we encountered no obstacles that fundamentally threatened the functionality of our system.

Part of this planning process was doing our homework on the capabilities of all the components we intended to use. We requested the Ultra96 instead of one of the already available dev boards because from our research, the extra CPU power would be necessary to handle some software-defined parts of our neural net-based tracking algorithm and the video storage. And indeed we discovered that we almost maxed out the 4 A53 cores with our program. Likewise, we made sure that our board was compatible with the Deephi tools so that we wouldn't have to rely on untested functionality for the DPU functionality, even if we did eventually end up building the implementation ourselves.

Because of our meticulous planning, we did not have extraneous budget expenses, nor did we have to pay for rushed shipping for any of our supplies. Additionally, we made sure that the hardware-dependent functionality was cemented before the project progressed further, while the software was allowed to be slightly less rigid. Hardware is inflexible by nature, while software is more malleable. An awareness of each's strengths and weaknesses was vital to our execution, and helped ensure that the project we delivered was exactly what we set out to create.

VII. Related Work

While many other security systems exist, upon researching the competition, we found a number of flaws in existing implementations that we were displeased by, and encouraged the idea of this project as a viable competitive option. Most security systems require a central server for video collection/processing, often using proprietary and sometimes costly software. Additionally, few cameras had any form of tracking, instead relying on expensive multi-camera systems. How convenient for the camera vendors.

The tracking cameras we did find were of the most primitive variety, often only offering tracking in one direction of movement (to say nothing of zoom), and operating based solely upon movement instead of object recognition. They also generally were low resolution, and had a very coarse and limited tracking range.

The last key feature we found missing was optical zoom. We found not a single camera with both tracking an optical zoom, and the optical zoom cameras we did find started at $800, and absurd entry price for such a valuable feature.

By incorporating all of these features into a single, relatively inexpensive device, we truly created something unavailable on the greater market.

VI. Summary

We did what we set out to do. Our system meets our runtime requirements, with active power consumption under 10W (producing on-time of over 600 minutes, exceeding or 500 minute requirement), and idle power of approximately 60mW, producing an idle runtime of over 2 months, again exceeding our 30 day requirement. The tracking and zoom are both smooth and functional, and the frame rate numbers for tracking exceed our specifications. We were able to record video at 24 fps due to a limitation of the video encoder, but this is relatively close to our goal of 30fps.

If we had to make some changes, the only major considerations would be to further optimize the logic and software for lower power consumption, but we're overall pleased with the results we were able to achieve. As for advice for other students, all we'd say is plan well, and keep your head on your shoulders. Don't try to do the impossible.

VII. References

https://ipvm.com/reports/should-you-use-autotracking-ptzs

https://www.amazon.com/Battery-Powered-Security-Wireless-Wire-Free/dp/B07HH6Z357/

https://www.inc.com/john-white/tired-of-getting-your-packages-stolen-heres-what-to-do.html

http://citeseerx.ist.psu.edu/viewdoc/download?
doi=10.1.1.182.6338&rep=rep1&type=pdf

https://electronics.stackexchange.com/questions/52991/is-the-cortex-m0-really-low-power/52995

https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842181/Zynq+UltraScale+MPSoC+Power+Off+Suspend

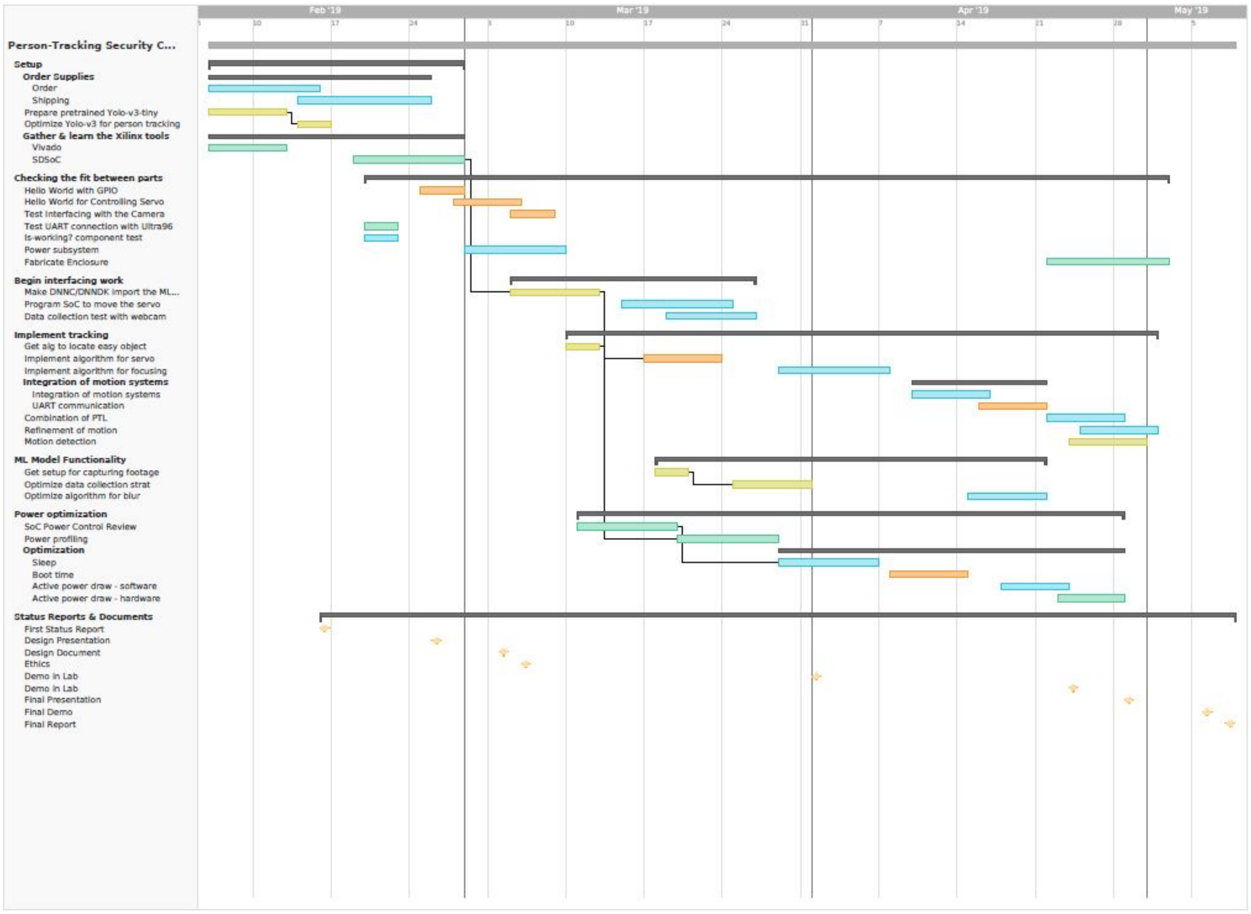


Fig 8. Gantt Chart detailing the schedule for the project. Blue is everyone, yellow is Jerry, orange is Karthik, and green is Nathan.