Final Report:
A Low-Cost System for Video Surveillance

Anbang Chen, anbang.chen@case.edu

Shibo Wang, shibo.wang@case.edu

Team 5

Index

1. Introduction

1.1 Motivation

Monitoring systems is becoming increasingly popular since it can help people to protect private places (like houses, garages, basements, etc.) from being trespassed. However, some major problems exist in current monitoring systems. Firstly, it requires manual operation and maintenance to keep observing the monitor screen. It's a very expensive solution and highly infeasible for individuals. Secondly, it is common that people will miss some quick invasions if they have not concentrated. Thirdly, it's difficult to find the evidence that who is invading from a long record. Therefore, we need some new method to replace/improve the current existing solution. In this project, we purpose a new implementation of real-time low-cost system for video surveillance. We will describe the methods we used for implementation, and then evaluate the performance and discuss the total cost of the system.

1.2 Platform

The system consists of:

Hardware: Raspberry Pi Model 3b, USB cameras, USB extension cable

Software: OpenCV library, Python

1.3 Problem Statement

In this project, the main goal is to design a system which uses camera captured video as input, analysis in real time, determines whether there exists a trespass human target, and raises proper warning as output. A Low-Cost System for Video Surveillance is designed to address this problem. The system mainly consists of multiple cameras and a central controlling embedded system (Raspberry Pi). The camera keeps capturing the images of the monitored site in real time at a certain rate and sending them back to the central controlling embedded system. The central controlling embedded system will analyze the images and detect whether there are moving objects. If there are moving objects and the objects are recognized to be human by the central controlling embedded system, the system will automatically save the images as evidence and alarm to the user. Then it comes to the problem that how to design such a large system.

Firstly, we need the system to be stable enough with as many cameras as possible. The cameras are controlled by the central system, so the central system should have a clear controlling logic to ensure that which camera is capturing and image from which camera is being processed. Moreover, it should know how to keep the other cameras working when images from one camera are detected containing suspicious moving objects.

Secondly, the method of alarm should be effective. For example, users can easily know there are intruders and when the intruders were detected. In this project, we simplify this problem to enable the central controlling embedded system to classify the evidence by time.

Thirdly, the cost of the whole system should be low, at least lower than employing people to watch the monitoring screens. To solve this problem, we can cut the cost both on cameras and central controlling system. If we cut the cost of each camera, it will lead to a reduction in the clearness of the feedback images. Thus, it depends on the high accuracy of the detection algorithm. On the other hand, if we cut the cost of the central controlling embedded system, it will have lower computing power and smaller storage space. Hence, the control algorithm and detection algorithm should have lower time complexity and space complexity.

2. State of the Art

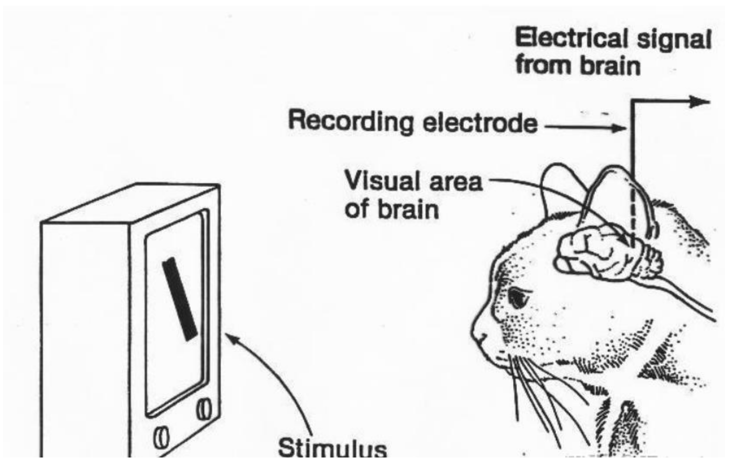
In this section we discuss the prior work, existing solution, state of the art technology and some relevant products.

2.1. Basic Methods

Motion Detection is a kind of technology to detect the changes of the relative position of the objects in the surrounding environment. There are several common methods to collect the data of the moving objects and the surrounding environment, including sensing infrared light, visible light, wave, magnetic field, energy and signal. In this project, we mainly used the method of sensing visible light, which means using camera to capture photos of the moving objects and the surrounding environment. To analyze the captured photos, we need the technology of Computer Vision. The group of captured photos are compared with each other and Computer Vision is used to analyze and classify the difference.

2.2. Computer Vision

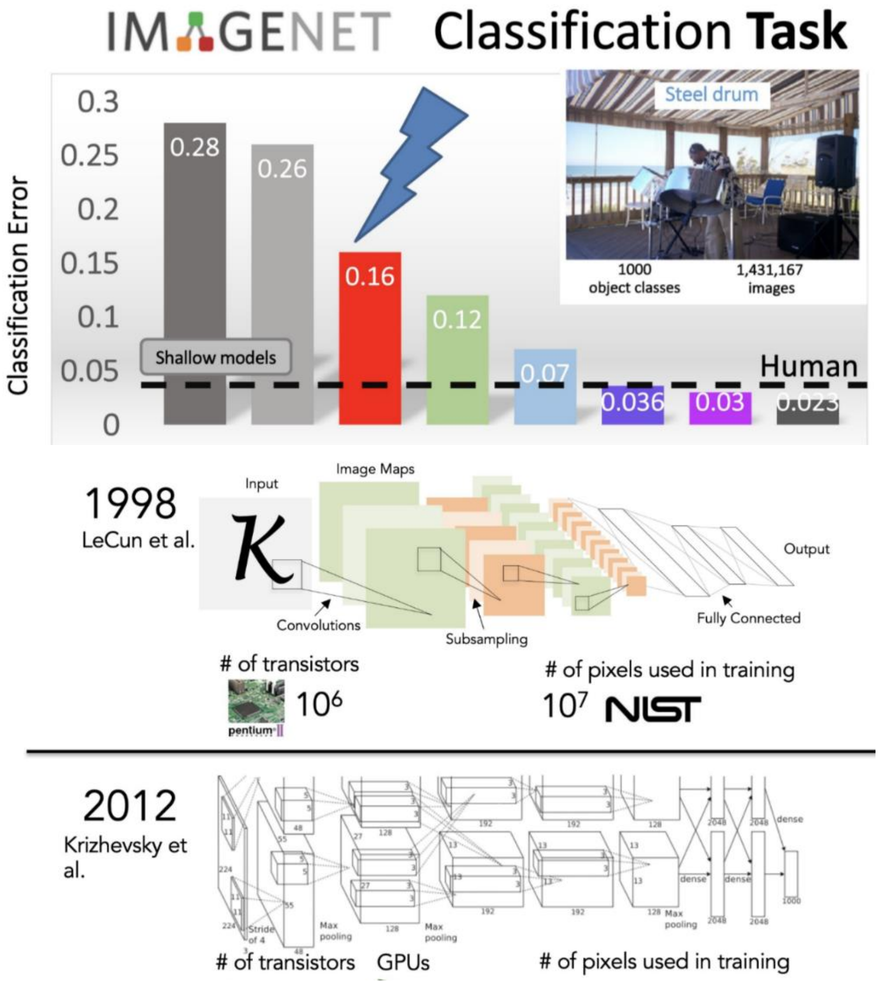
In 1959, neurophysiologists David Hubel and Torsten Wiesel discovered for the first time through visual experiments in cats that neurons in the visual primary cortex were sensitive to moving edge stimuli, and discovered the visual functional column structure, laying the foundation for visual nerve research - leading to the breakthrough development of computer vision technology 40 years later and laying the core guidelines for deep learning. [1]



In the same year, Russell and his classmates developed an instrument that could convert images into grayscale values understood by binary machines, which was the first digital image scanner, making it possible to process digital images. [1]

In 1980, Japanese computer scientist Kunihiko Fukushima, inspired by the research of Hubel and Wiesel, established a self-organizing artificial network of simple and complex cells Noncognition, including several convolutional layers (usually rectangular), and his Receptive field has a weight vector (called a filter). The function of these filters is to slide on a two-dimensional array of input values (such as image pixels) and generate activation events (two-dimensional arrays) after performing certain calculations, which will be used as inputs for subsequent layers of the network. Fukushima's Neocognitron can be said to be the first neural network, which was the initial example and inspiration for the convolutional layer and pooling layer in modern CNN networks. [1]

In 2012, Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton created a "large deep convolutional neural network", now known as AlexNet, which won the ILSVRC of the year. This is the first time in history that a model has performed so well on the ImageNet dataset. The paper "ImageNet Classification with Deep Convolutional Networks" has been cited approximately 7000 times and is widely regarded as one of the most important papers in the industry, truly showcasing the advantages of CNN. The error rate of machine recognition ranges from around 25%. It has decreased by about 16%, which is not much different from humans. It was only then that CNN became a household name. [1]



In 2014, the University of Montreal proposed Generative Adversarial Networks (GANs): having two competing neural networks can make machine learning faster. One network attempts to generate fake data by imitating real data, while the other network attempts to distinguish fake data. Over time, both networks will be trained, and generating adversarial networks (GANs) is considered a major breakthrough in the field of computer vision.

The development of deep learning frameworks from 2017 to 2018 has reached a mature stage. PyTorch and TensorFlow have become the preferred frameworks, both of which provide a large number of pre trained models for multiple tasks, including image classification.

In recent years, domestic and foreign giants have expanded into the field of computer vision and established computer vision research laboratories. Empowering existing businesses with new computer vision systems and technologies to expand the battlefield. Facebook's AI Research (FAIR) claimed in 2016 that its DeepFace facial recognition algorithm has a recognition accuracy of 97.35%, almost on par with humans, in terms of vision. In 2017, Lin, Tsung Yi and others proposed a feature pyramid network, which can capture stronger Semantic information from the deep feature map. At the same time, Mask R-CNN is proposed for instance segmentation tasks in images. It uses a simple and basic network design, and does not require much complex training optimization process and parameter settings to achieve the current best instance segmentation effect, with high operational efficiency. [2]

In 2016, Amazon acquired a top European computer vision team to add the ability to recognize obstacles and landing areas to the Prime Air drone. Develop drone delivery. In 2017, Amazon Web Services (AWS) announced a series of updates to its recognition service, providing cloud customers with machine learning based computer vision capabilities. Customers will be able to conduct real-time facial search on a collection of millions of faces. For example, Rekognition can be used to verify that a person's image matches another image in an existing database, with up to tens of millions of images and sub second level latency. [3]

At the end of 2018, NVIDIA released Video to Video synthesis, which synthesizes high-resolution, photo realistic, and time consistent videos through a carefully designed generator, discriminator network, and spatiotemporal adversarial objective lens, making AI more physically conscious, powerful, and able to be promoted to new and invisible scenes. [4]

In 2019, BigGAN, also known as a GAN, was only more powerful and equipped with smarter course learning techniques. The images it trained and generated could not even distinguish true from false, because unless viewed under a microscope, it would not be able to determine if there were any issues with the image. Therefore, it is even hailed as the strongest image generator in history. At the end of May 2020, Facebook released a new shopping AI, the universal computer vision system GrokNet, that made "everything available for purchase".

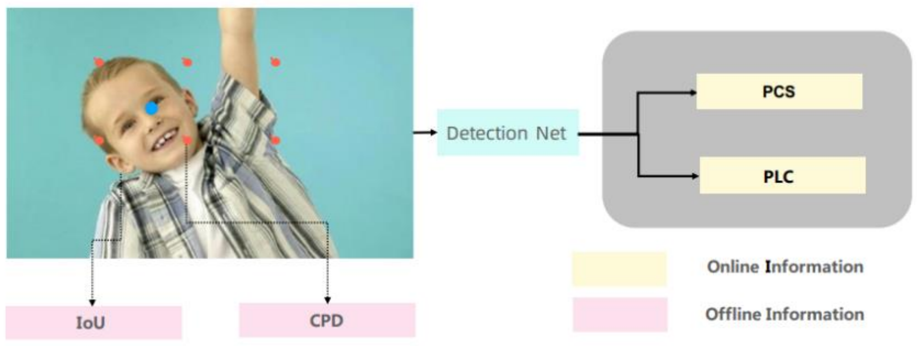
2.3. Face Recognition

After motion detected by using computer vision, we need to confirm that whether this motion should be considered as suspicious or not. One of the most important standards is to find out if the moving object is human being, which needs the technology of face recognition.

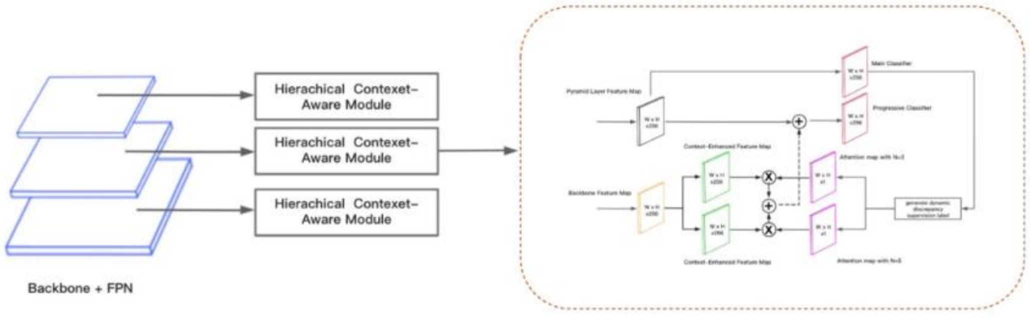
The latest method of face recognition contains three parts, including dynamic label assignment, false positive context analysis and pyramid layer level GT assignment.

1. Dynamic Label Assignment

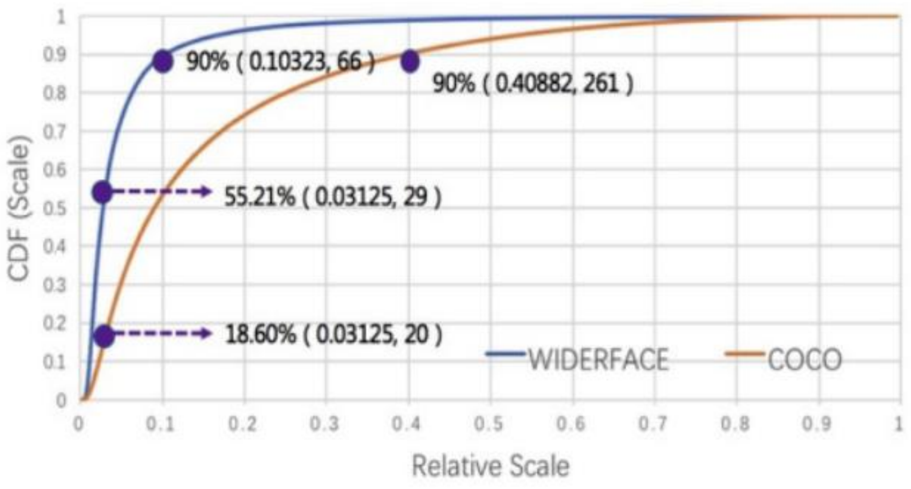
   In the latest research, an adaptive online incremental anchor mining strategy (Ali-AMS) is raised, which is based on the standard anchor matching strategy and further adapts to help the outer face match the anchor. [5]



2. It mentioned in the article that for the same FP, when its context changes, it may not be an FP for the same detector. [5] Therefore, Hierachical Context-Aware Module (HCAM), a two-step module is used to display encoding context information to help distinguish FP and TP, which can significantly reduce the number of FPs.


Hierachical Context-Aware Module
Hierachical Context-Aware Module
Hierachical Context-Aware Module
Backbone + FPN
Pyramid Layer Feature Map
Main Classifier
Progressive Classifier
Context-Enhanced Feature Map
Attention map with N=3
Backbone Feature Map
Context-Enhanced Feature Map
Attention map with N=5
generate dynamic discrepancy supervision label

3. Scale level data augmentation strategies are often used as the main means to address scale variance in general object detection and face detection. As shown in the following figure, compared with COCO, the scale distribution of faces in the Wider Face of the face detection dataset is more severe. To this end, we have proposed a new question, how to reasonably allocate the distribution of ground truth on different pyramid layers? What is the relationship between the performance of the detector and the number of ground truth matches for each pyramid layer? Is more better? Through rigorous comparative experiments, we found that 'for all pyramid layers, the more ground truth this pyramid layer matches, the better'. This indicates that in order to discover the best performance of each Pyramidlayer, it is necessary to control the proportion of ground truth allocation on this Pyramidlayer. [5]



3. Approach, Models, Methods, Techniques, Assumptions

This section discusses the methods and approach of the implementation of the model. The following image is the system diagram that describes each component of the system. It takes in the input of four cameras at most (depends on the requirement of the user, the system can handle a maximum of four cameras as input), then it will take advantage of the idea of Time Division Multiplexing to handle multiple input using one processor in real time. It will do a set of sequential computations, and the result of the computation is used to make the decision of whether there is a target passing by or approaching. And give the appropriate warning accordingly. More detailed description will be found in the following paragraphs.

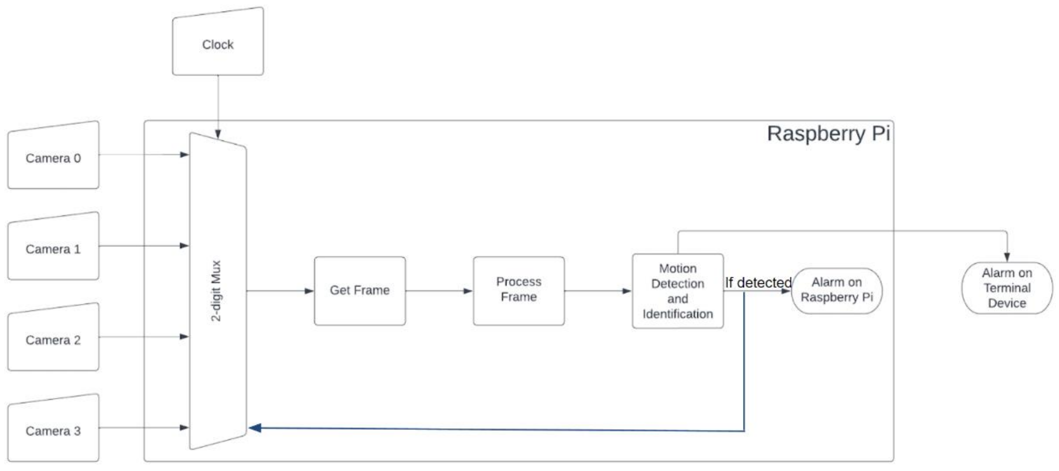


Figure1 System diagram

3.1 Implementation of the multi-camera system

As mentioned above, this system is able to handle up to four camera inputs using only one processor. This implementation has its own advantages and disadvantages, which is listed below:

Advantages:

- No potential processor communication issue, no need to worry about the synchronization among processors

- More stable for long time use
- Reduce power consumption

Disadvantages

- All inputs share one processor, so the frames per second drop rapidly when the number of the cameras connected to the system increase

After careful consideration and experiment, we decided to take advantage of the idea of Time Division Multiplexing to handle multiple input using one processor, basing on the observation that Raspberry Pi 3b model can run our model in real time with 65 frames per second, even when using the maximum four cameras, each camera can still be processed with more than 15 fps, sufficient enough to satisfy the usage scenario (since the target is to detect passing by/approaching human target, instead of generating continuous surveillance videos with high fps). The following image shows how the data flow of a four-camera system is processing with time division multiplexing. A, B, C and D represent the four cameras connected to the system. The parameters and result of each camera input will be store separately and will not interrupt each other. However, there will be flags and global variables to indicate the status of the system, so it will not raise repeat warning. (For example, if the target is passing by and detected by camera B, it will not warn again even if that person is also detected by camera C; from the other hand, once a camera raise the warning, it will not be cleared unless the suspicious target disappears from all cameras).

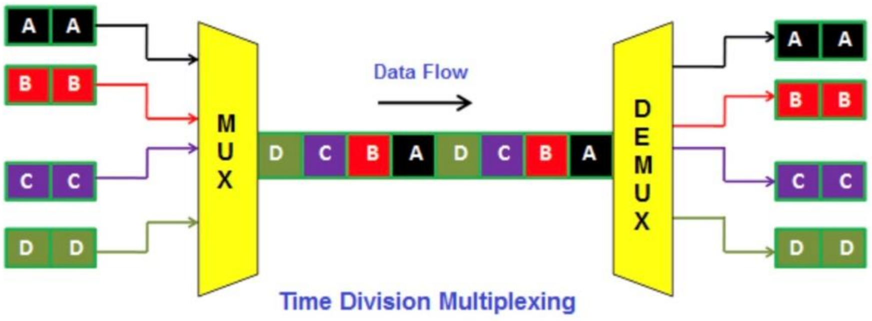


Figure2 Data flow of four-camera system with Time Division Multiplexing [1]

And the following screenshot is an example of how an three-camera system can be run and processed in real time in our system.

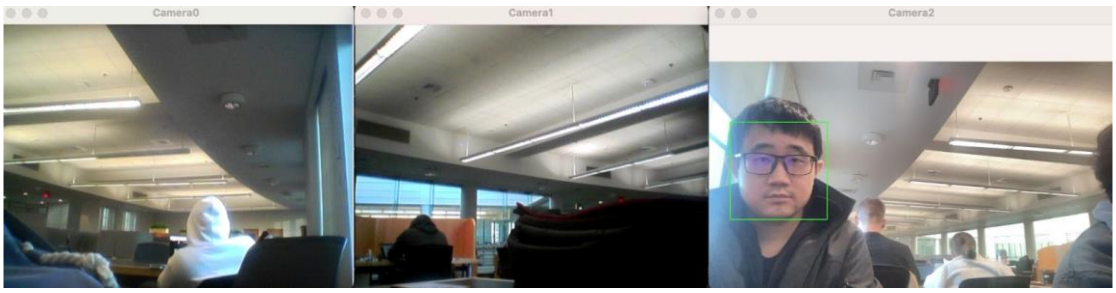


Figure3 Real time three-camera system

3.2 Video processing

This section includes how the system process the video input in real time. We describe the algorithm with one camera input for simplicity, but the idea can be easily extended to multiple cameras situation using the idea of Time Division Multiplexing. The step-by-step process and the corresponding result are listed below:

1. Input the current frame captured by the camera. For example, the image below can be the input of the camera, and the following analysis will be based on this image

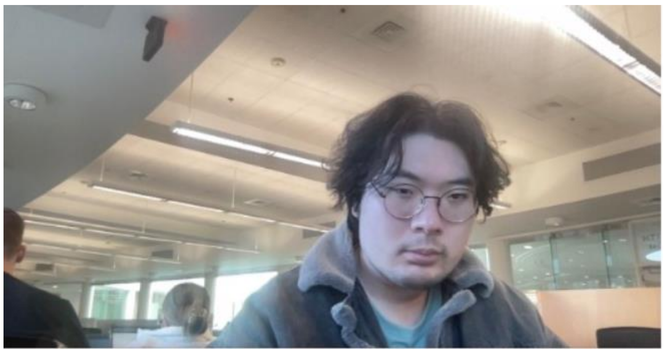


Figure4 Input frame

2. Detect frame difference using Gaussian Mixture Model (GMM)

   Gaussian Mixture Model (GMM) is a probabilistic model that assumes all the pixel points are generated from a mixture of Gaussian distributions. In practice, it is possible that the cameras we use have low quality and the performance of the system will be compromised because of the background noise. However, we can still make the assumption that the background noise exists in such a pattern that they distribute normally, which means they will cluster among mean μ and more than 99% of the pixel points gather within 3σ. Therefore, when the system starts, it uses the frames at the very beginning to build the Gaussian model, when new frame comes, it compares with the model to determine the probability that there are pixels that do not share the same layer with the model, and mark those pixels white. This is also how we achieve motion detection. The following image shows the result after running the GMM algorithm:

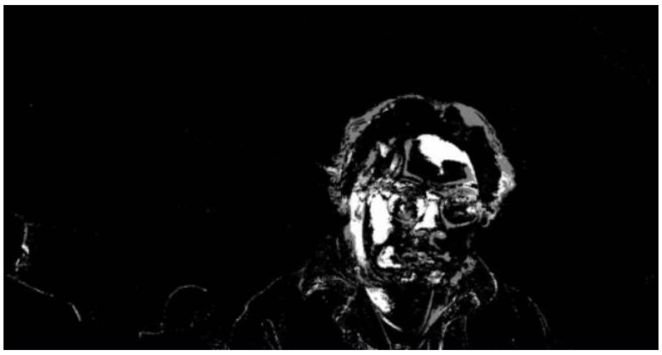


Figure5 Result after GMM algorithm

Comparing to traditional approach using frame difference, GMM algorithm has certain unique advantages. First, it is more stable to periodic background noise. And second, it can better separate background from foreground. Also, it is relatively cheap to compute, which makes this technique feasible.

3. Morphology transformation

After applying to GMM, the resolution of the result is too high, which is not a desired feature because it is more vulnerable to small noise, therefore reduce the accuracy of the following process, and will compromise the final performance of the system. To address this problem, we apply a 5x5 kernel in order to reduce the granularity of the image, smooth the edges and remove small objects. As shown below, the result after the morphology transformation seems like a reduction of pixel resolution. The purpose of this process is to better increase the accuracy of detection.

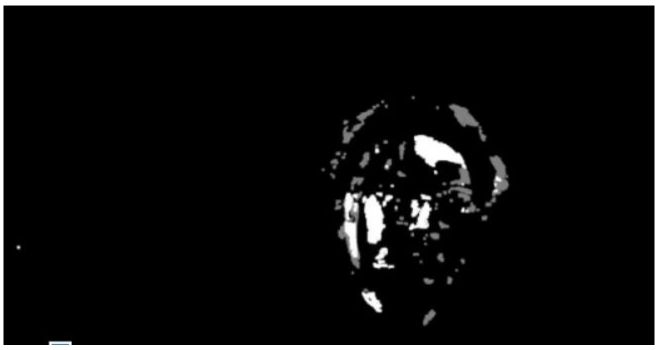


Figure6 Image after Morphology transformation

4. Thresholding

   After Morphology transformation, the image has black, white and gray area, depending on the actual input image. We use proper thresholding to make the result image either black or white. The purpose of this process is to reach a higher quality contour area calculation.

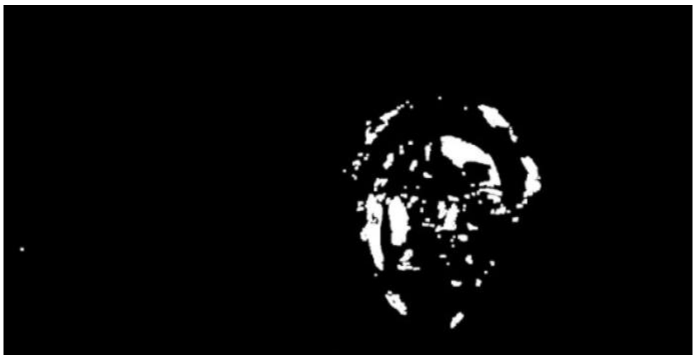


Figure7 Image after thresholding

5. Find contour ratio

   This is the final output of real time video processing, we begin by finding the contour area of the result after thresholding, then we obtain the ratio by dividing the contour area by the resolution of the input image. The result ratio represents the level of change in the image: as the ratio increase, the level of movement detected also increase. Therefore, we are able to use this ratio as the basis to determine the level of motioned detected basing on the real time video input.

3.3 Human detection

Even if the real time video processing (as described in the previous section) detect a very large ratio indicating the movement is violent, it does not necessary leads to a warning. Because such movement might be caused by sources other than human target, such as a passing by cat or dog, rotating fan, or other kind of background noise like slight shaking. Therefore, the system needs to perform human object detection before it raises a warning. This feature is implemented by using Cascade classifier, a built in pre-trained classifier of OpenCV and is able to detect human faces within the image. The Cascade classifier itself has its own limitation. First, it is relatively expensive to be run in real time with a Raspberry Pi system. It will significantly reduce the fps and make the system less stable. Therefore, in order to reduce the calculation cost when using this classifier, we only perform the human detection when movement is detected and the system is about to raise a warning. If the human detection is passed, the system will immediately raise the warning, but if the detection failed, the system will withhold the warning, keep performing face detection until it decides the object is indeed a human.

Second, the Cascade classifier works only when the target face directly opposite the camera, if the target covers his/hers face with a mask, or the angle exceeds certain limit, the classifier might fail for the detection. However, in the experiments we conducted, we find that this limitation is acceptable and rarely cause any real trouble, especially when the target approach from a distance. In general, Cascade classifier is much better than traditional solutions (like calculating the size of moving object), and more complicated/advanced deep learning algorithm can hardly run on a Raspberry Pi system in real time. We therefore conclude that this implementation of face detection reaches an acceptable trade off between feasibility and accuracy. The following screenshot shows that the classifier is capable of human object detection.

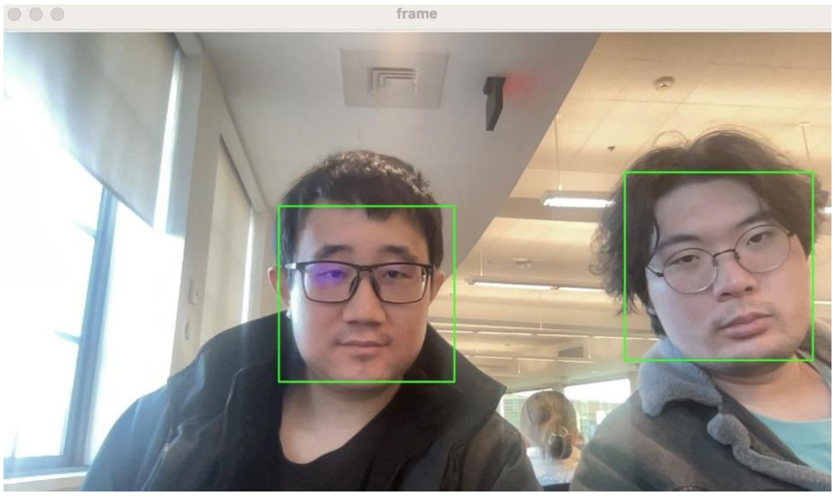


Figure8 Human detection with Cascade classifier

3.4 Stages of events

This section aims to describe the stages of events of the system using Finite State Machine. These different stages determine the behavior of the whole system (whether to raise an alarm or not). First, we define the following three events:

Event 1: No motion detected

Event 2: Target passing by

Event 3: Target approaching

And the behavior between stages can be described using the following Finite State Machine diagram:

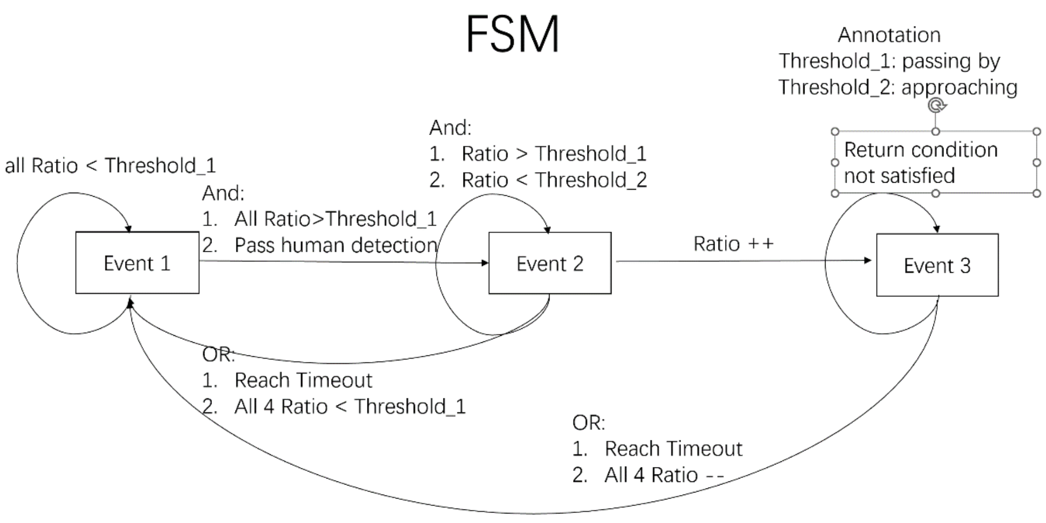


Figure9 Finite State Machine for stages of events of the system

As described above, the system run in real time permanently without interruption, and it will stay in Event 1 if no human motion is detected (no human motion happens either threshold condition for passing by is not satisfied for ALL cameras input, or it pass the threshold condition but fail the human object detection).

The system moves to Event 2 if any one of the ratios satisfies threshold condition for passing by AND the human object detection is passed. It will stay in Event 2 as long as the threshold condition for approaching is not satisfied. And it moves back to Event 1 if timeout is reached, or ALL ratios become lower than the threshold for passing by target.

The system moves to Event 3 if any one of the ratios satisfies threshold condition for approaching. Then it will stay in Event 3, until the timeout is reached, or ALL ratios become lower than the threshold for passing by target (which indicates the target now disappears from all cameras).

3.5 Alarm system

Once the system moves to Event 2 or Event 3 (events described in the previous subsection), the system will raise a warning in the terminal, showing explicitly whether the suspicious target is passing by or approaching. For the simplicity of the system, it will not give sound/LED flash light feedback. Instead, for each warning, the system will capture 10 consecutive screenshots and save them in the file system as evidence. The path of the screenshots can be freely specified by the user, and the folder name is the date when the screenshots are taken. The screenshots under the same date will be sorted depends on how many times it already triggered the alarm in that particular day. For example, if the system detects an approaching target, and the system has already detected three trespass cases (including both passing by and approaching target) in this day, then this particular approaching will be stored in "4-approaching" folder under the directory named after date.

3.6 Timeout system

Once the system is triggered and raises a warning, it will change the status and will not clear the warning until no more motion is detected (detailed description in section 3.5). The purpose of this mechanism is not only to avoid giving repeat warning (for example, if the system detects approaching by both camera 1 and 2, it will only raise one approaching alarm), but also to reduce the cost (the Raspberry Pi system cannot afford human detection using Cascade classifier in real-time). In most situations, this mechanism works accurate as designed, but in some rare situations interrupted by unexpected noise, this might cause the system very difficult to clear the warning status. This essentially block the system from detecting potential future trespass target.

To address this problem, we purpose a timeout mechanism which allows the system to count the time it stays in Event 2 (passing by) or Event 3 (approaching). Once it reaches the timeout limit, it clears the warning status regardless of the actual situation. As a result, the system becomes more stable and can resolve the self-blocking situation automatically.

4. **Results**

This section describes the detailed results of the implementation. Before running the program, the user needs to make several decisions with regard to the behavior and sensitivity of the system (however, if the user wants to run the program in default parameters, no decisions need to be made). These decisions are:

1. Specify the running mode, the system (based on Raspberry Pi model 3b) have two modes:
   - Display the real time surveillance video, with 50 fps.
   - Not to display the real time surveillance video, with 65 fps.

   (Note: the fps described above is the total fps of ALL cameras. For example, if two cameras are connected to the system, and the user decide not to display the real time surveillance video, then each camera have around 32 fps).
2. The sensitivity of detection (threshold for identifying passing by or approaching target). The default level of sensitivity for detection is valid in almost all usage scenarios, but the user has the option to determine the sensitivity basing on their actual requirement.
3. The number of cameras connected to the system.

After identifying the above-mentioned parameters, the system runs in real time permanently (without user's interruption). If no trespass target is identified, the system has no feedback. Once the trespass human target is identified, it raise proper warning (depending on whether the target is marked as passing by or approaching), the warning consist of two major parts: raise alarm in the terminal, and save screenshots as evidence. Basing on the hardware experiments and software stress test that we conduct, the system is accurate, stable and reliable, with the ability for long-term usage.

5. Cost analysis

In this section, we perform detailed cost analysis for implementing this system, and calculate the suggested retail price. The cost for this system consists of four major parts: hardware, amortized R&D cost, amortized marketing cost, and amortized cost for maintaining the business. All these costs are subjected to the number of products we are able to sell. (It is obvious how sales affect amortized cost, in addition, an increase in the purchase quantity often leads to a decrease in the unit price). To calculate the recommended sailing price with reasonable profit, we need to estimate the sales condition. To reach overall highest efficient, we plan to sale the product through online shopping platform including Amazon, eBay, BestBuy, etc. Therefore, depending on the overall sales status of our competitive products, we come up with three simulations, 1000 sales is the minimum, 5000 as expected, and 10000 for exceed expectation situation.

First, we begin by calculating the unit cost of hardware for 1000, 5000, 10000 sales

- Hardware cost, considering three different sales condition:

Item	Number	Unit cost (1000 sales)	Unit cost (5000 sales)	Unit cost (10000 sales)
Raspberry Pi 3b	1	89.99	79.99	69.99
USB camera	3	17.99	15.99	13.99
Expander cable	1	7.19	6.39	5.59
SD card 510G	1	26.99	23.99	20.99
Total cost	(Per system)	178.14	158.35	138.56

Then we analysis the total cost for R&D, marketing, and maintenance:

- R&D:

	labor hour	hourly wage	Total
Implementation	100	40	4000
Testing	10	30	300

Maintenance	10	30	300
R&D materials	---	---	500
Total			5100

- Marketing:

	Cost (total)
Website advertisement	5000
Mobile app advertisement	5000
Total	10000

- Maintenance

	Cost (per system)
Customer service	2
Package	10
Delivery	10
Warehouse	1
Miscellaneous	5
Total	28

Therefore, the total cost after considering amortized cost per system is:

	Unit cost (1000 sales)	Unit cost (5000 sales)	Unit cost (10000 sales)
Hardware	178.14	158.35	138.56
R&D	5.1	1.02	0.51
Marketing	10	2	1
Maintenance	28	28	28
Total	221.24	189.37	168.07

Therefore, it is suggested that each product sells at **$249.99**, and a $30 coupon can be applied if the sales exceed 5000.

6. Conclusion

In this section, we will summarize the conclusion and potential future work of this project. During our hardware experiments and software stress test, we validate that this implementation of real time video surveillance system is capable of long-term use and is capable of detecting trespass target as well as human object detection. In the cost analysis section, we take into account not only the cost of hardware implementation, but also the cost of other amortized cost, including research and development, marketing and maintenance. From the cost analysis, we come up with the total cost and the suggested retail price. Therefore, we reach the general conclusion that this implementation is feasible (in a sense that it is economical) and reliable (it behaves properly and the whole system is stable). However, basing on the way we implement the system and the tools that we use, there exist certain limitations that can compromise the performance of the system.

1. The program is based on Python, although it is relatively more convenient for implementation, it is not as fast as other programming languages like C or C++.
2. The hardware microcontroller we select is Raspberry Pi model 3b, we make this selection mainly because it is much cheaper and more affordable, when comparing with latest model 4. The limitation in hardware will not decrease the accuracy of detection, but will limit the maximum number of cameras that can be connected to the system.
3. We use Cascade classifier for human object identification, this method is cheap in terms of calculation resources. But this tool works by performing face detection, if the target is wearing a face mask or somehow cover their face, the whole system can fail the human detection.
4. To reduce the cost, we choose cheap USB camera with low resolution and poor image quality, this will not cause any problem when the target is close enough, but the low resolution makes it difficult for the system to identify human target more than 10 meters away.

5. To make the system compact, we did not use LED flash light or sound alarm, which might make the alarm methods oversimplified.

Basing on the limitation of current implementation, certain future works might be useful to further improve the performance of this system.

1. Rewrite the program in C++ to make it more efficient.
2. Select more advanced Raspberry Pi model, use state of the art deep learning algorithms instead of the Cascade classifier for the human detection.
3. Add LED flash light and sound warning, as well as other warning (like sending email to the user or app warning)
4. Choose better USB camera component

With these new features, we believe the performance of the system can be improved, basing on our preliminary evaluation, the cost will increase $150 per system for the improvement.

References

[1] Wang, Annie. Development History of Computer Vision. Last modified May 28, 2020. https://zhuanlan.zhihu.com/p/142927311.

[2] From FPN to Mask R-CNN. Last modified March 20, 2018. https://blog.csdn.net/dqcfkyqdxym3f8rb0/article/details/79630599.

[3] Amazon AWS's Computer Vision Enables Real-time Facial Recognition. http://wwwebrun.com/20171123.

[4] Review of Technological Breakthroughs in the Field of Computer Vision in 2018. Last modified January 4, 2019.
https://baijiahao.baidu.com/s?id=1621695581612030690&wfr=spider&for=pc.

[5] Liu, Yang, Fei Wang, Jiankang Deng, Zhipeng Zhou, and Baigui Sun. "MogFace: Towards a Deeper Appreciation on Face Detection."
https://openaccess.thecvf.com/content/CVPR2022/papers/Liu_MogFace_Towards_a_Deeper_Appreciation_on_Face_Detection_CVPR_2022_paper.pdf.