Surveillance Camera Foreign Object Detection Through Image Processing and Machine Learning

Michael Hernandez – Master of Science in Analytics Program, University of Chicago

Summary

- In April of 2013, the U.S. was attacked when two individuals placed improvised explosive devices at the finish line of the Boston Marathon. Forensic video analysis was used, after the fact, to review the facts of the case.
- In the future, can things like this be prevented using video analysis systems?
- It is widely publicized that citizens are the first line of defense to these types of attacks.

Img 1 Img 2 Img 3

- Although this process can be effective at times (e.g. circumventing the attempted 2010 Times Square Bombing), it relies on human beings to be constantly observant and proactive.
- A popular alternative is to utilize image processing and machine learning.
- <u>Through a mix of image pre-processing, image segmentation, and deep learning, we can identify unattended baggage (potential security risk) using a highly scalable framework (that can server hundreds or thousands of camera feeds) relatively reliably.</u>

Methodology

- This implementation (which I will refer to as "the program") is written in Python and mainly uses OpenCV for image processing and segmentation. Currently, image classification is done externally through an API[5].
- The program moves between 5 different "states". These "states" represent the current level of alertness the program is in (see escalation).

Image Pre-Processing

- Each frame of video is fed through a pre-processing function. The function a) converts the frame to grayscale and b) performs a Gaussian Blur on the frame.

Pre-Processed Frame of me walking with a backpack.

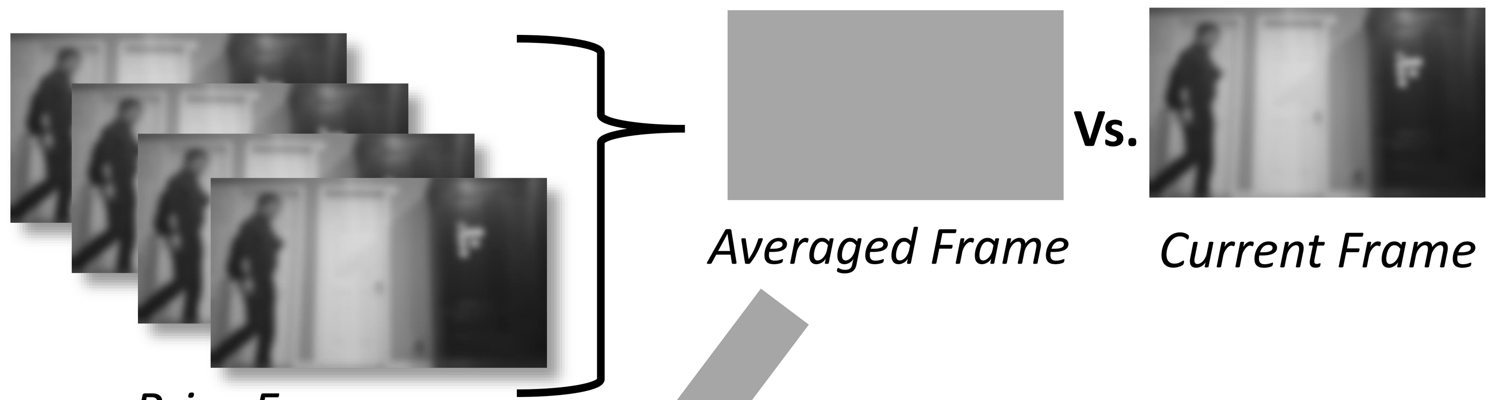
Grayscale: For each pixel apply: $$f(x) = \frac{r+g+b}{3}$$

Gaussian Blurring: For each pixel apply: $$g(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

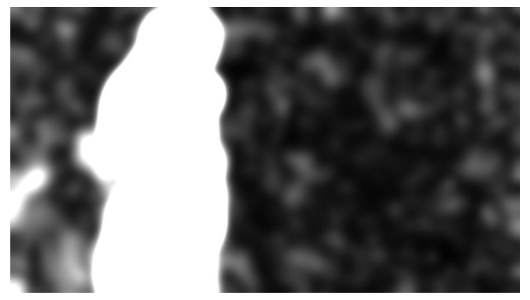
Image Differencing

- Each pre-processed frame is then differenced against a composite weighted average of past frames using a differencing function.

Image Differencing (Continued)



Prior Frames *Averaged Frame* *Current Frame*



Differenced Frame

Absolute Differencing: For each set of pixels apply: $h(x) = |p_1 - p_2|$ *Where p1 and p2 are our pixel intensities.*

Thresholding: For each pixel apply: $$p < t \rightarrow x = 0 \land p < t \rightarrow x = 255$$ *Where p is pixel intensity and t is our threshold (constant).*

Thresholded Differenced Frame

- A frame contains motion if the proportion of white pixels to total pixels ("pixel ratio"), in the frame, is above a certain (user-selected) threshold.

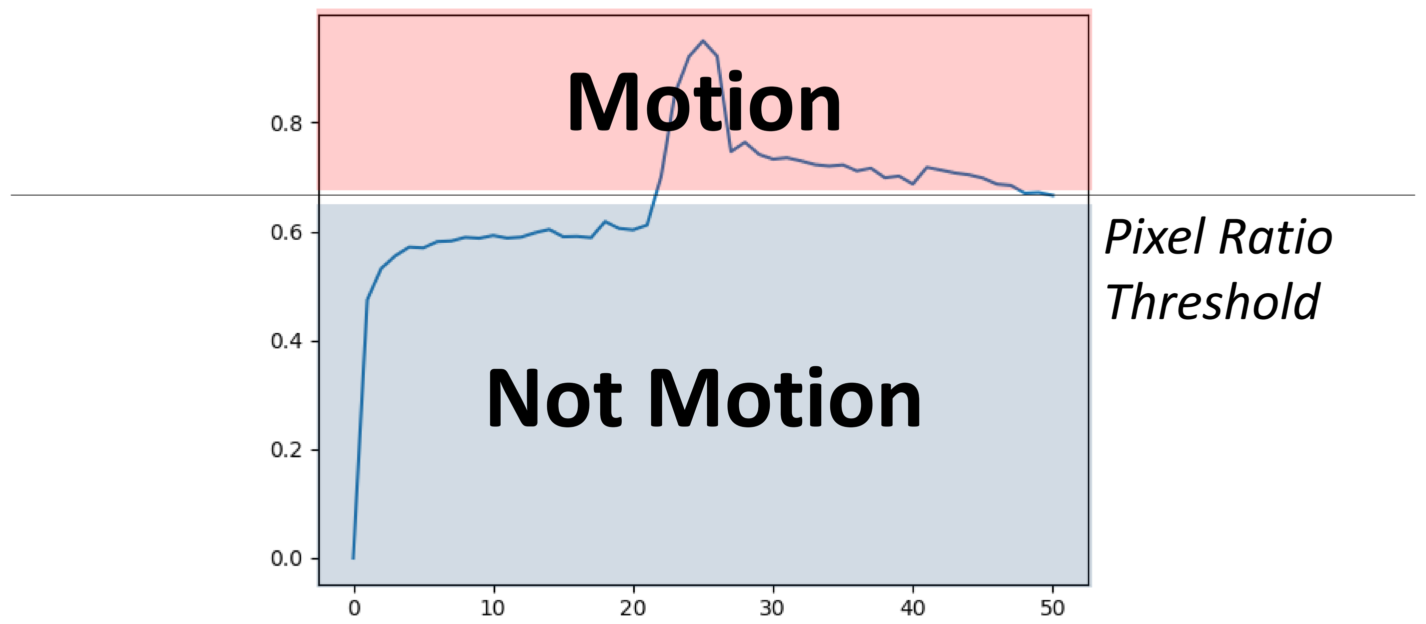
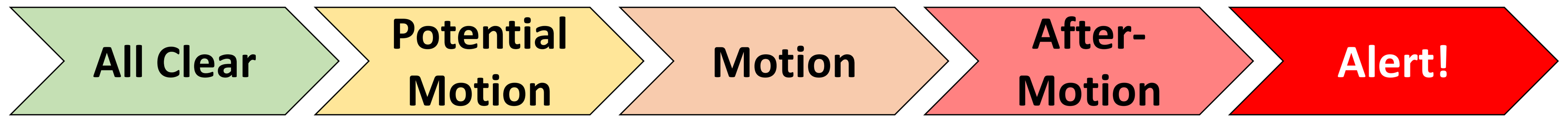


Illustration of Pixel Ratio Over Time

Escalation

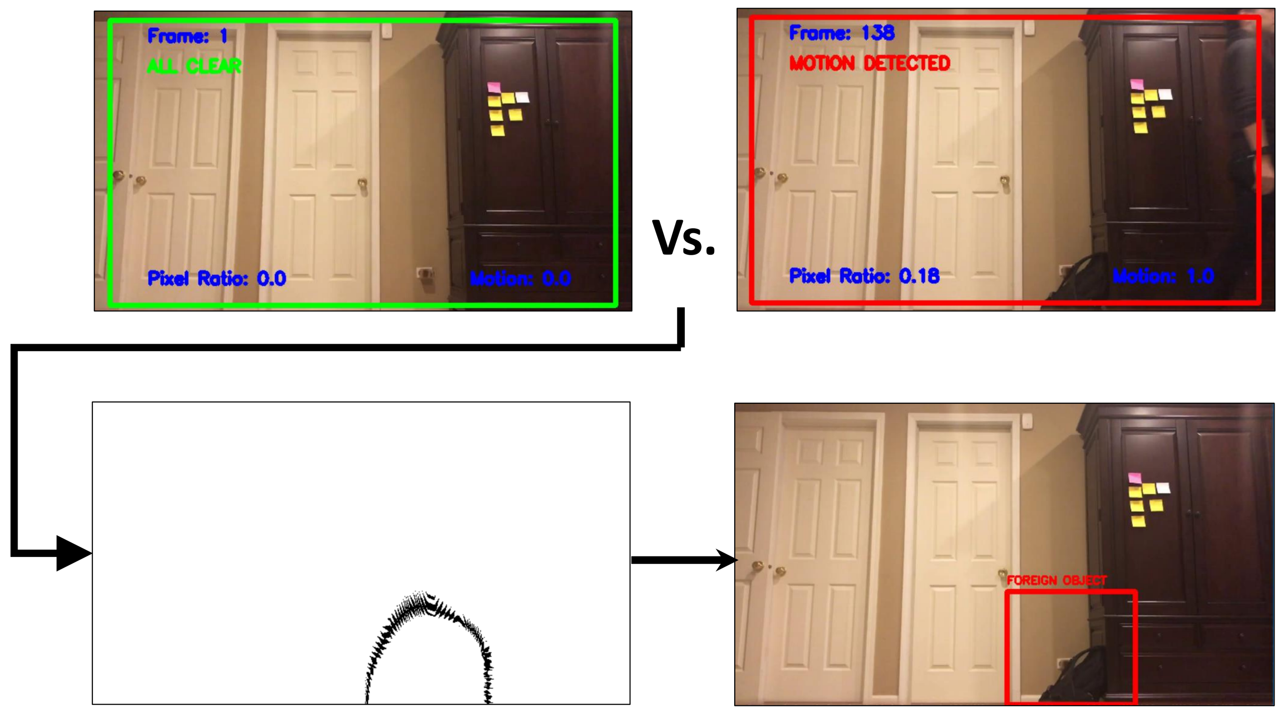
- On any given video frame, the system can either escalate to the next state, or de-escalate one (or multiple) state(s).



- **All Clear:** Is the default state when the program is launched. A frame is "all clear" if pixel ratio is below a user-defined threshold (T1). The state can de-escalate from "potential motion" to "all clear" if it fails to meet the criteria of potential motion. <u>In each "all clear" fame, a snapshot of the frame is saved for comparison later ("snapshot").</u>

Escalation (Continued)

- **Potential Motion:** The state can escalate to "potential motion" from "all clear" if the pixel ratio is above T1, but sustained motion (motion for several frames) has not been identified.
- **Motion:** The state can escalate to "motion" from "potential motion" if the proportion of frames that have potential motion, within a user-defined window of frames, is greater than a user-defined level (T2).
- **After-Motion:** The state escalates from "motion" to "after-motion" when the proportion of frames that have potential motion, within a user-defined window of frames, is lesser than T2.
- As motionless frames pass, the proportion of frames having potential motion will drop below a user-defined level (T3), which signals to the program that it should prepare for image segmentation and tagging.
- The saved frame ("snapshot") and the current frame are processed and compared. The result identifies any new object, which is sent, via byte-stream, to an API where tags are applied to the image.



- Alert!: The alert state is triggered if one of the "tags" returned by the API matches any item in a list of tags maintained by the user.

Subsetting Algorithm

- To find the top and left indexes boundary of the subset, apply to each pixel: $$p = 0 \cap p_{index} > p_{best} \rightarrow p_{best} = p_{index}$$
- To find the bottom and right indexes boundary of the subset, apply to each pixel: $$p = 0 \cap p_{index} < p_{best} \rightarrow p_{best} = p_{index}$$

Conclusion

- A hybrid image detector methodology (using pre-processing and machine learning) provides <u>advantages of:</u>
  - ✓ **Precision:** Only the relevant new image component is sent to the image recognition module.
  - ✓ **Scalability:** The highly scalable image processing component does most of the work, and deep learning only steps in when it needs to.
- <u>Areas for improvement include:</u>
  - ✓ **Reliance on motion:** Although the escalation system does prevent several "false alarms" from occurring, it also relies on several seconds (roughly 4 seconds) of motion to "activate".
  - ✓ **Optimized for single object:** The method of segmentation is currently optimized for only a single object. When multiple new objects enter the frame, the program can still succeed, but does so with lower success.
- **Overall, the image detection marketplace is crowded with several high quality implementations. This project was not an attempt to re-create the image detection "wheel", but rather create a highly scalable implementation of this technology based around a specific use case.**

Acknowledgements / Citations

1. "Proof of Concept: Detection of a Train's Passing and Direction. (n.d.). Retrieved March 1, 2017, from http://cmawer.github.io/trainspotting/trainspotting-blog.html"
2. Img 1: https://lamourecountynd.com/image3/images/b105196a71c8837933d569fcd46cee5bdc33b6d6.jpg
3. Img 2: https://www.dhs.gov/sites/default/files/images/seesay_0.jpg
4. Img 3: http://www.mta.info/sites/default/files/archive/imgs/see_something_lg.png
5. Clarifai API. https://www.clarifai.com/

Volume Rating ●●●●○
- Consumes and processes raw video data.
- When linked to multiple streams and depending on the video quality, the incoming data stream can become gigabytes.

Velocity Rating ●●●●●
- Able to process stored video data frame by frame.
- Also able to analyze streaming data in real-time.

Variety Rating ●●●○○
- Able to be connected to any video stream and can start analyzing immediately.
- Can have difficulty when videos constantly have motion.

Voracity Rating ●○○○○
- Video feeds, at any time, can be corrupted. One potential improvement would be to include image recovery built-in to the program.

Github
- Full project available for download from Github.
- https://github.com/mhernan88/fo_detect