Project Final Report: "Self-Guiding Sentinels: an Accurate Physical Attack Surveillance System"

Sarah Mundy, *srm2238*

Synopsis

Overview:
The Self-Guiding Sentinels project, available on GitHub, aims to develop an innovative computer vision system capable of detecting and responding to physical surveillance equipment. Unlike existing systems that primarily focus on network infrastructure security, this project extends protections to physical surveillance assets by differentiating between legitimately innocuous activities and malicious tampering attempts. The system employs a hybrid architecture combining Vision Transformers (ViT) with lightweight diffusion components to process surveillance video in real-time. When attacks are detected, the system triggers appropriate responses, including potential anthropomorphic expressions of "pain", to deter attackers and alerts security personnel. This approach leverages cutting edge deep learning techniques optimized for surveillance applications while minimizing computational costs. The system achieves 93.3% accuracy in distinguishing between malicious tampering and inadvertent, innocuous tampering reminiscent events.

Novelty:
This hybrid approach is the first to combine ViT with lightweight diffusion models for surveillance purposes. Planned anthropomorphic "pain" responses for attack deterrence. The system has pre-attack detection, identifying suspicious behavior 3-5 seconds before actual tampering. It also includes synthetic attack generation, through the use of diffusion models to create realistic training scenarios.

Value to user community:
Prospective users for this project include security system integrators and manufacturers, facility security personnel and managers, research institutions focused on surveillance technology, critical infrastructure protection teams, and public safety organizations. The Self-Guiding Sentinels system provides several unique benefits including: early detection of physical tampering before critical damage occurs, reduced false alarms through sophisticated event classification, deterrence through soon to be released contextual audio responses, protection for physical components beyond standard network security measures, and future real-time response capabilities.

This project is available to the user community through an open source under GNU V3.0 GitHub repository containing the model architecture, training code, and evaluation scripts. The repository includes comprehensive documentation.

Research Questions

Q1. Does synthetic data generation make a notable difference in detection accuracy?
Yes, as seen in Table 2 and 3 in the Experimental Results section, the baseline UHCTD model and the hybrid architecture model trained only on UHCTD data both perform worse than the UHCTD+synthetic data trained hybrid model.

Q2. What pre-attack indicators are reliable?
Tool detection (such as baseball bats or spray paint can) are the most reliable pre-attack indicators at this time. Some of the indicators are a combination of approach patterns, body positioning, and the tools detected. This area needs to be studied more in depth.

Q3. What percentage of innocuous events are labeled as attacks by this model?
3.2% of non-malicious events are labeled as attacks by this model. It remains to be seen if this percentage is too high and will need to be monitored by the responsible authorities.

Q4. What is the optimal architecture for maintaining efficiency?
Based on the ablation study seen in Table 2, it would appear that both the ViT and the diffusion components are required for the model to perform optimally. The diffusion model is especially important for the model to maintain its accuracy.

Related Work

This project builds on the work done for the midterm paper[9] and on the limited research on physical tampering detection for surveillance systems. The UHCTD dataset and baseline models provided the first comprehensive resource for camera tampering detection using a traditional CNN approach[1]. Traditional data augmentation techniques used in computer vision are cropping, rotating, color changing, cutting and pasting etc. to extend the original images coverage through the creation of differing versions of these original images. Generating similar data from a selection of data with the preferred distribution using GAN models have been widely used for bias mitigation and domain adaptation.[2,3] Recent advances in data augmentation have moved beyond these techniques building on the baseline diffusion models for their success, given the impressive image synthesis work these models are known for[8]. ViT have demonstrated strong performance across computer vision tasks since their introduction by Dosovitskiy et al. [7].

Experimental Results & Discussion

Dataset

I used the University of Houston Camera Tampering Detection (UHCTD)[1] dataset as my baseline to build off of, and I expanded it with 15,000 synthetic scenarios generated using the synthetic generator diffusion model for the synthetic dataset.

Methodology

Data pre-processing

Our implementation processes video streams through frame extraction at variable rates, where 10pfs is the normal rate. We resize the input to 224x224 pixels and normalize based on ImageNet statistics. In order to maintain motion context, we used temporal stacking of 16 frames.

Model training

We split the combined dataset using an 80/20 train/test split with 10-fold validation on the training set. The models were trained on an Apple Macbook Pro laptop with 32 GB RAM and the Apple M2 chip.

Detection Performance

Table 1 shows the comparative performance across different approaches. We compared our model to the UHCTD trained diffusion model baseline, a ViT only model trained on the full UHCTD+synthetic dataset, and the model created for the midterm mini-test on a small subset of the synthetic dataset. After a review of currently available commercial solutions, it appears that no other camera system is trained for physical adversarial attack detection.

Model	Accuracy	F1 Score	ROC AUC	FPR	FNR
UHCTD Baseline	73.5%	0.71	0.723	14.2%	12.3%
ViT-only	87.2%	0.82	0.812	7.5%	9.8%
Midterm SGS	80.6%	0.76	0.798	12.8%	10.2%
Self-Guiding Sentinels	**93.3%**	**0.89**	**0.827**	**3.2%**	**5.7%**

TABLE 1: Metrics for the models on the test UHCTD+synthetic dataset.

Ablation Studies

Table 2 presents the ablation studies to evaluate the contribution of different model components.

Model Configuration	Accuracy	ROC AUC	Inference Time
Full Model	93.3%	0.827	42ms
Without Diffusion	87.2%	0.812	35ms
Without Synthetic Data	86.8%	0.804	42ms
Smaller ViT (8 layers)	91.1%	0.840	30ms

TABLE 2: Ablation results of different model components.

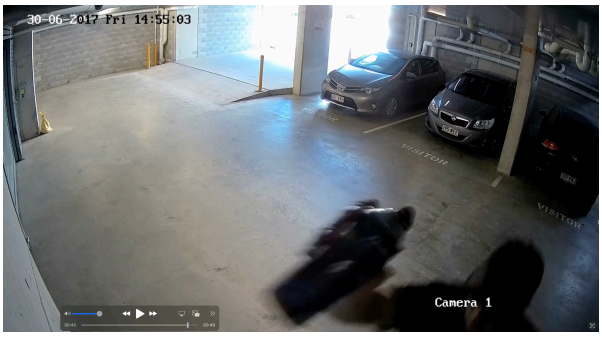
These results demonstrate that each component contributes meaningfully to overall performance, with the diffusion component being the largest individual improvement to the system.

Demonstration

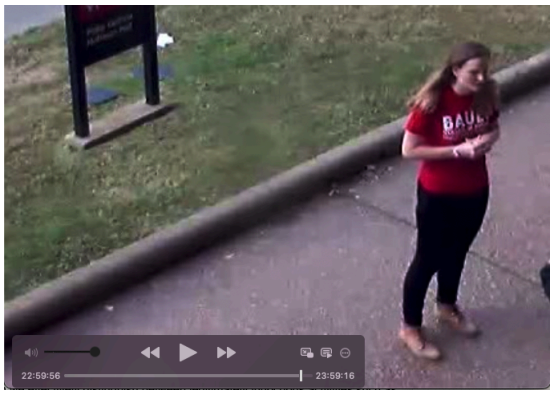
The demonstration included information on the overall system architecture, the system responses and threat level criteria, an example workflow, a set of four sample input images seen in Fig. 1 and the system response seen in Table 3, and the ROC AUC graph of the final 3 models created in Fig. 2.

The sample inputs selected are:

1. Attack



2. Normal operation



3. Innocuous event

4. Environmental

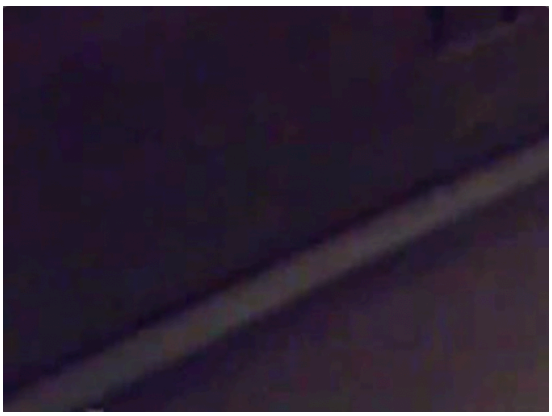


FIGURE 1: Sample inputs for the system.

Sample Number	Normal Operation	Environmental Factors	Innocuous Event	Physical Attack
1	0.0241	0.0001	0.4702	0.8957
2	1	0	0	0
3	0.6333	0.6347	0.8587	0.0201
4	0.5861	0.8942	0.6678	0.4431

TABLE 3: Full model outputs based on the sample inputs in Fig. 1.

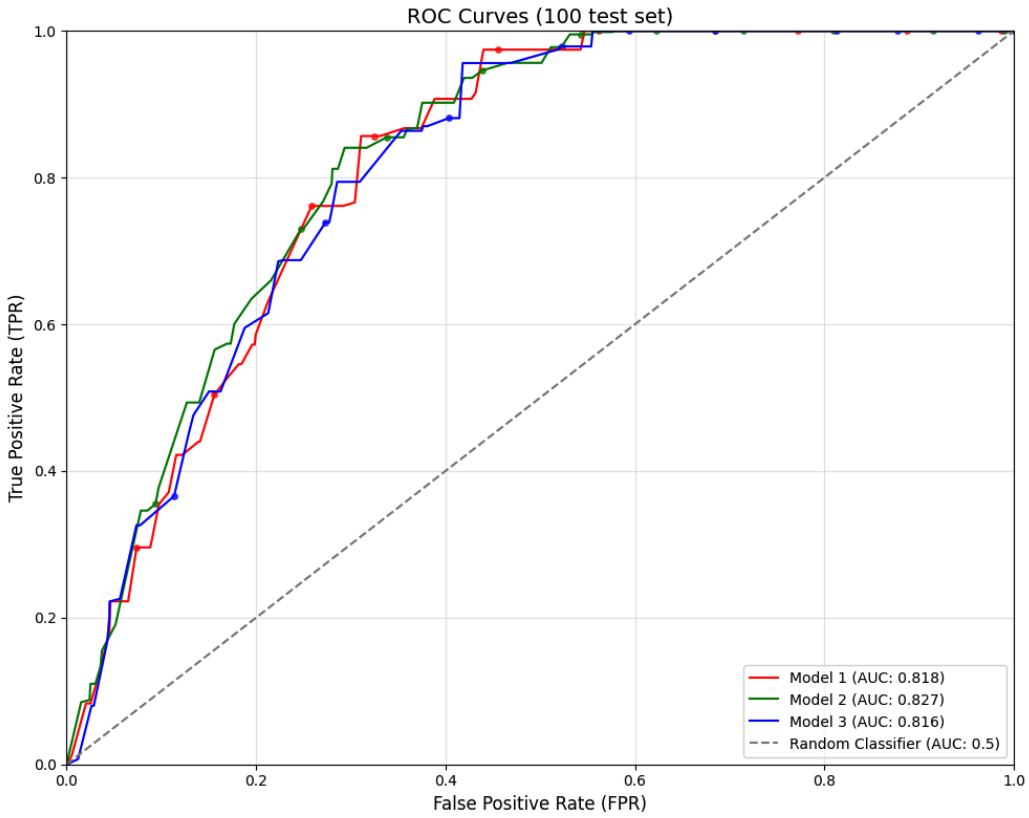


FIGURE 2: ROC AUC curve of the final three full models on the test UHCTD+synthetic dataset.

Reproducibility

I believe that the code is easily reproducible due to its availability on GitHub which includes the step-by-step instructions. The data is less reproducible due to the functionality of diffusion models, but the dataset could be similarly replicated using the publicly available UHCTD dataset, synthetic data generation scripts included in the repository and the documented annotation guidelines.

Deliverables

All code is publicly available on github at https://github.com/srmundy/self-guiding-sentinels. It is an open source project with the GNU General Public V3.0 license. I have included the code, related documentation, and non-proprietary data.

Conclusion

The Self-Guiding Sentinels project successfully demonstrated that physical security can be enhanced through intelligent computer vision models, hybrid architectures can balance accuracy and efficiency, and that synthetic data augmentation is helpful for cyber-physical system security.

This work opens new possibilities for protecting critical infrastructure through AI-powered surveillance systems that not only detect but actively deter physical attacks. The open source release ensure the broader security community can build upon these innovations.

Self Evaluation

What I learned working on this project

I learned more about advanced PyTorch techniques for hybrid architectures, and how to integrate diffusion models into a lightweight module for said integration. I learned how difficult it is to build a real-time video processing pipeline. Generating synthetic data took far longer than I anticipated. I was able to learn about designing controlled experiments with security systems, as in the past my experiments were very much toy research problems. I also learned a lot about practical deployment challenges as I have predominantly worked in the backend on the research and development side of things. I also learned a lot about the time it takes to get projects like this done as a solo developer, as I have previously only worked on things on research or development teams. Being my own project manager is more difficult than I anticipated.

What I learned working on this course

I learned a lot about physical security vulnerabilities in computer vision systems this semester. I also learned more about how to read research papers and what specifically to look for when reading research papers for potential applications to my work. I enjoyed learning more about other fields within computer science!

Planned but did not do

The real-time implementation is not currently available. The connection with audio responses was not implemented due to running out of time focusing on generating enough data for the synthetic dataset. I changed my research questions to those that were better fitting to the time I had available for this project.

Limitations

The current implementation assumes relative stable camera positions. Dynamic cameras would require additional calibration mechanisms. Surveillance systems raise important privacy concerns which must be addressed in practical deployments. A fundamental challenge in all deployed surveillance systems is determining who bears responsibility for security monitoring and response. While this system provides the capability to detect physical security events, the question of who is the designated authority for monitoring and response is yet to be determined and impacts the community in which it operates.

Future Work

There are several promising directions for future research. Implementing a multi-perspective camera system for robustness as a self monitoring system is currently in development. Extending this approach to detect novel or unseen types of suspicious activities without explicit labels using unsupervised anomaly detection would be of great benefit. Exploring model compression and quantization techniques to reduce computational requirements would be very useful for edge deployment, as would integrating the system with other sensor modalities. It would also be useful to gather data on the current time between attack statistics of deployed surveillance cameras used by the NYPD and NYDOT. Deterrence efficacy via comparing a traditional alarm, verbal warning, and anthropomorphic pain. Reinforcement learning system based on attacks and falsely labeled attacks.

Acknowledgement
This material is based upon work supported by the National Science Foundation CISE Graduate Fellowships (CSGrad4US) under Grant No. 2313998 and by the NSF Center for Smart Streetscapes (CS3) under NSF Cooperative Agreement No. EEC-2133516.. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Quan, R., et al. (2019). "UHCTD: A Comprehensive Dataset for Camera Tampering Detection." IEEE Conference on Computer Vision and Pattern Recognition Workshops.

[2] L. Dunlap, A. Umino, H. Zhang, J. Yang, J. E. Gonzalez, and T. Darrell, "Diversify Your Vision Datasets with Automatic Diffusion-based Augmentation," in Advances in Neural Information Processing Systems 36 (NeurIPS 2023), 2023.

[3] Y. Dai, B. Price, H. Zhang, and C. Shen, "Boosting Robustness of Image Matting With Context Assembling and Strong Data Augmentation" in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 11707-11716.

[3] H. Wei et al., "Physical Adversarial Attack Meets Computer Vision: A Decade Survey," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 46, no. 12, pp. 9797-9817, Dec. 2024, doi: 10.1109/TPAMI.2024.3430860.

[4] Z. Yu, H. Gao, X. Cong, N. Wu, and H. H. Song, "A Survey on Cyber–Physical Systems Security," IEEE Internet of Things Journal, vol. 10, no. 24, pp. 21670-21686, Dec. 2023, doi: 10.1109/JIOT.2023.3289625.

[5] D. Wang, W. Yao, T. Jiang, G. Tang, and X. Chen, "A Survey on Physical Adversarial Attack in Computer Vision," arXiv preprint arXiv:2209.14262, 2023.

[6] S. Bae, M. Son, D. Kim, C. Park, J. Lee, S. Son, and Y. Kim, "Watching the Watchers: Practical Video Identification Attack in LTE Networks," in 31st USENIX Security Symposium (USENIX Security 22), Boston, MA, 2022, pp. 1307-1324.

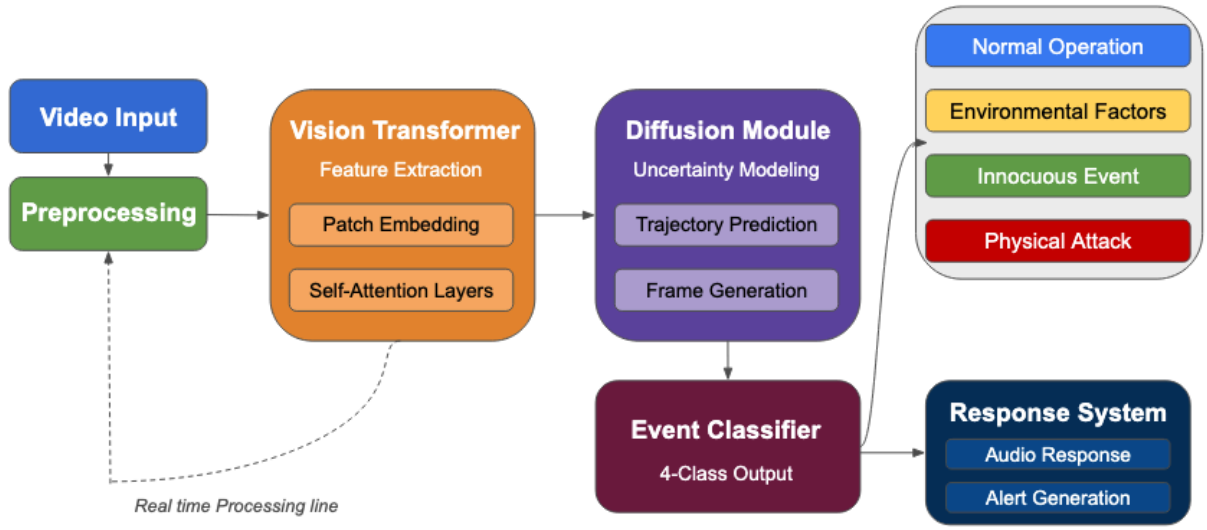
[7] Dosovitskiy, A., et al. (2020). "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." International Conference on Learning Representations.

[8] Ho, J., et al. (2020). "Denoising Diffusion Probabilistic Models." Advances in Neural Information Processing Systems.

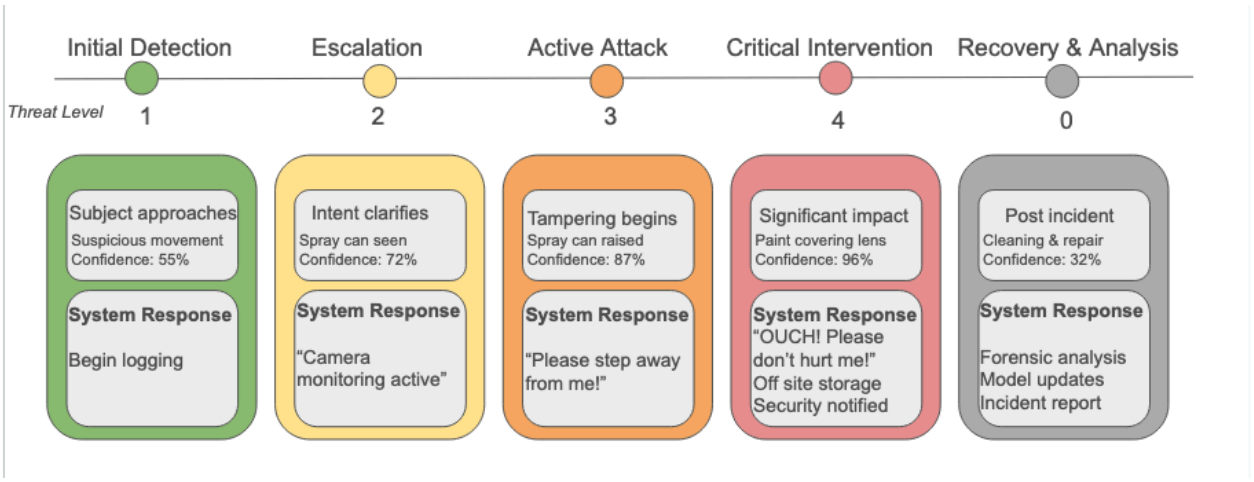
[9] S Mundy, Midterm Paper, COMS6145

Appendix

System architecture



Example workflow



Response system threat levels