**Assignment 1**

**Statement of Work (SOW) for Real-time Anomaly Detection in Video Surveillance**

1. **Introduction**
   1. **Background and Significance**

Nowadays, the number of surveillance cameras in the United States has been steadily increasing over the years because of technological advancements and security concerns both in public spaces and private environments. Monitoring feeds and detecting anomalies in video surveillance systems frequently involves human operators. However, humans can get tired, distracted, and neglectful. Therefore, developing a real-time automatic anomaly detection in video surveillance can address this problem and offer a proactive response to shootings and other types of crime such as shoplifting, assault, and robbery because it can keep watching non-stop throughout the day and night, providing a prompt response to any odd activity. In addition, such a system could serve as an early threat detection, which makes it possible to spot unusual or unexpected events as they take place. This can involve people loitering in sensitive areas, vehicles moving in peculiar patterns, or intruders breaking into secured facilities. By quickly identifying anomalies, security guards and police can respond quickly to possible threats reducing the chance of danger or damage.

* 1. **Objectives**
     + Early Threat Detection: to design and develop a real-time automatic anomaly detection system in video surveillance that can identify unusual or unexpected events as they take place so that the police and security guard can quickly respond to possible threats and reduce the chance of danger or damage.
     + Scalability: to build a real-time automatic anomaly detection system that can process multiple video surveillance streams from thousands of cameras at once, which provides a scalable solution that retains accuracy over big installations.
     + Reduced False Positives: to develop a real-time automatic anomaly detection system that can learn from past data and normal behavior patterns to decrease the number of false or unnecessary alarms so that it can be focused on genuinely suspicious activities.
     + Proactive Response: Nowadays, monitoring feeds and detecting anomalies in traditional video surveillance systems frequently involves human operators. However, humans can get tired, distracted, and neglectful. Automated real-time anomaly detection can keep watch non-stop throughout the day and night, providing a prompt response to any odd activity.
     + Emergency Notification: to build a real-time automatic anomaly detection system that can send a notification to users and local police when an anomaly action is detected. This way we can immediately stop the crime and prevent any further damage to the property or people.

Through these objectives, we aim to create a real-time automatic anomaly detection in video surveillance that can offer accurate identification of unusual events and enhance proactive response to any potential threats.

1. **Scope**
   1. **Definition**

The primary objective of this project is to develop a robust and efficient real-time automatic anomaly detection system that can accurately identify anomalous events in video surveillance streams. The system will use machine learning and computer vision techniques to analyze video input and detect deviations from normal behavior.

* 1. **Limitations**
     + Model Generalization: Anomaly detection models may struggle with generalizing to unseen anomalies that differ from the training dataset.
     + False Positives and Negatives: All anomaly detection systems have a trade-off between false positives and false negatives. Balancing this trade-off is challenging and may require ongoing adjustments.
     + Continuous Improvement: Anomaly detection models are not static and may require continuous retraining and refinement to adapt to evolving anomalies and new patterns of behavior.

1. **Data Sources**

For this project, we will utilize the UCF-Crime Dataset available at [<https://www.dropbox.com/sh/75v5ehq4cdg5g5g/AABvnJSwZI7zXb8_myBA0CLHa?dl=0>]. The UCF-Crime Dataset contains a collection of video clips capturing various crime and normal activities. Each video clip is labeled with information about the type of anomalous event or normal, providing a suitable foundation for training and evaluating our real-time anomaly detection system.

1. **Methodologies**
   1. Convolutional Neural Networks (CNN):
      * Feature Extraction: Use convolutional layers to automatically extract hierarchical
      * features from images, enabling the network to learn representations of various objects and textures within the images.
      * Learned Filters: Use sets of filters that slide across the input image. These filters are adjusted during training to recognize different shapes, textures, and patterns.
   2. You Only Look Once (YOLO) Object Detection:
      * Single Pass Detection: Divides the input image into a grid and performs object detection in a single pass. Each grid cell predicts bounding boxes, class probabilities, and confidence scores for objects present within the cell.
      * Anchor Boxes: Create anchor boxes of varying sizes and aspect ratios to predict objects. This allows the model to handle objects of different scales and orientations.
      * Real-time Processing: Architecture is designed for real-time processing, making it efficient for video analysis and applications where speed is crucial.
   3. OpenPose:
      * Pose Estimation: Identify joints on human bodies, including wrists, elbows, shoulders, hips, knees, and ankles. These joint positions define the posture and pose of individuals.
      * Multi-person Detection: Identify multiple individuals in a single image or frame, allowing for the analysis of group behavior and interactions.
      * Movement Analysis: Tracking the changes in joint positions over time, enabling the identification of abnormal or unusual behavior patterns.
   4. Video Explainity Methods:
      * Use LIME, sharp, or CLAD to explain why was the detection decision was made
2. **Timeline**
3. **Week 1-2**: Requirement gathering and literature review.
4. **Week 3-4**: Data collection and preprocessing.
5. **Week 5-6**: Prototype development.
6. **Week 7-8**: Model optimization.
7. **Week 9-10**: Implement user interface.
8. **Week 11-12**: Performance evaluation and fine-tuning.
9. **Week 13-14**: Documentation and final testing.