EG UK Computer Graphics & Visual Computing (2023)
D. Hunter and P. Vangorp (Editors)

Augmenting Anomaly Detection Datasets with Reactive Synthetic Elements

I. Nikolov

Computer Graphics Group, Department of Architecture, Design and Media Technology, Aalborg University, Denmark

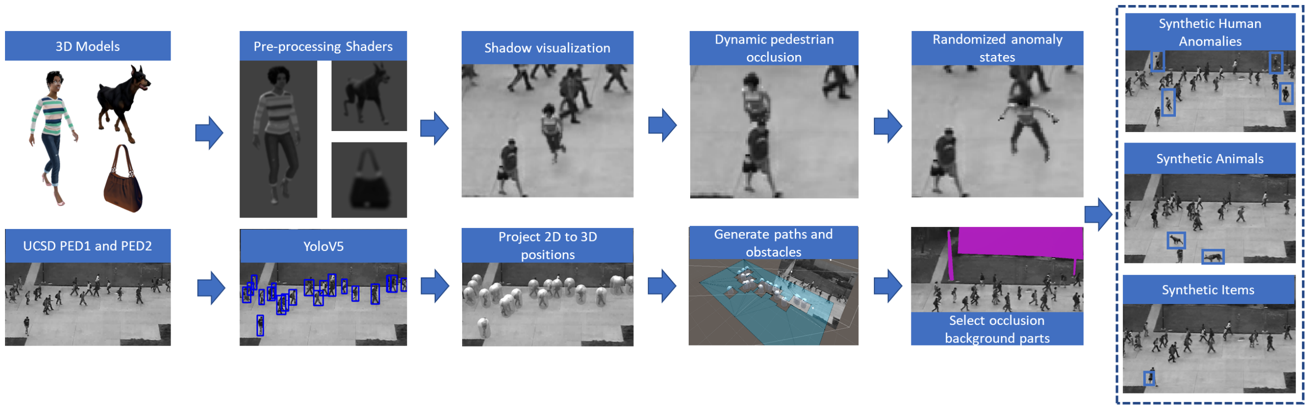

3D Models
Pre-processing Shaders
Shadow visualization
Dynamic pedestrian occlusion
Randomized anomaly states
Synthetic Human Anomalies
UCSD PED1 and PED2
YoloV5
Project 2D to 3D positions
Generate paths and obstacles
Select occlusion background parts
Synthetic Animals
Synthetic Items

Figure 1: *Overview of the synthetic data rendering pipeline, together with the three types of augmentation*

Abstract

Automatic anomaly detection for surveillance purposes has become an integral part of accident prevention and early warning systems. The lack of sufficient real datasets for training and testing such detectors has pushed a lot of research into synthetic data generation. A hybrid approach by combining real images with synthetic elements has been proven to produce the best training results. We aim to extend this hybrid approach by combining the backgrounds and real people captured in datasets with synthetic elements which dynamically react to real pedestrians and create more coherent video sequences. Our pipeline is the first to directly augment synthetic objects like handbags and suitcases to real pedestrians and provides dynamic occlusion between real and synthetic elements in the images. The pipeline can be easily used to produce a continuous stream of randomized augmented normal and abnormal data for training and testing. As a basis for our augmented images, we use one of the most widely used classical datasets for anomaly detection - the UCSD dataset. We show that the synthetic data produced by our proposed pipeline can be used to make the dataset harder for state-of-the-art models, by introducing more varied and challenging anomalies. We also demonstrate that the additional synthetic normal data can boost the performance of some models. Our solution can be easily extended with additional 3D models, animations, and anomaly scenarios.

CCS Concepts

• ***Computing methodologies*** → ***Image processing;*** *Neural networks;* ***Anomaly detection;***

1. Instructions

The research field of automatic anomaly detection in outdoor pedestrian videos has seen significant growth, focusing on improving accuracy and robustness, and expanding the range of detectable anomalous events. However, despite the extensive efforts, reliable solutions remain elusive. The primary challenge stems from the scarcity of consistent abnormal data, as it is rarely captured independently. Additionally, the context, camera positions, time of day, and other factors greatly influence anomaly detection, further complicating the task. Consequently, existing algorithms often address only a limited subset of anomalies, hindering the generality and reproducibility of the proposed methods.

Anomaly detection models mostly employ the idea of treating anomalies as an outlier detection task, where parts of the footage

DOI: 10.2312/cgvc.20231204

that deviate from the learned normality are flagged as anomalous [ZLK*19, TPC*21]. As shown by Acsintoae et al. [AFG*22] there is also a second class of anomaly detectors that function as action-recognition models, which are weakly supervised on both normal and abnormal data [LCC*21, PNH20, ZLL*22, PNH20, RJ20]. The common factor between the two types of anomaly detectors is that they require a large number of normal and abnormal images for training and testing. Another commonality is that all these methods require annotated data, either coarsely annotated on frame level or more fine-grained annotations of the anomalies themselves either through bounding boxes or per-pixel masks [LSJ13, RJ20]. Several open-source outdoor datasets exist that are widely used and focus on specific anomalies - connected to either pedestrian behavior, movement of vehicles, or interactions between the two [MLBV10, COCH21, LLLG18]. Another such dataset that is the focus of this paper is the widely used UCSD [MLBV10]. The dataset provides many challenges for anomaly detection like grayscale images, limited resolution, and crowded scenes. On the other hand, the dataset has been considered "solved" in recent years, because of its limited types of anomalies - bicycles, cars, and skateboards on a pedestrian street have made it easy for modern anomaly detectors to achieve accuracy above 90% both per frame and per anomaly detected [AFG*22, AZLL21, LCC*21, ZLL*22, PNH20].

Synthetic augmentation is used to try to combat the lack of quantity and diversity of data. Such datasets exist for scene parsing [WU18, RHK17], pedestrian tracking [FBM*21], and semantic segmentation [XWY*19] among others. These datasets can be separated roughly into ones that have fully generated backgrounds and foregrounds using programs like Blender, Cinema4D, Autodesk Maya, etc., and ones that combine backgrounds from real images with synthetic objects in the foreground. For anomaly detection of pedestrians most synthetic data is done by combining real images with synthetic augmentations [AFG*22, MSG*23, GBŠ21]. This is seen as better for training purposes, as real-world backgrounds shorten the distribution gap between synthetic and real images. But even they need to take additional steps to ensure that models trained on them can be successfully used on real data, like employing Generative Adversarial Networks (GANs) to translate the synthetic objects to a more realistic look.

With this paper, we would like to propose a novel rendering pipeline for combining the UCSD dataset with synthetic data which can dynamically react to the real parts of the dataset. we would like to bring new life to this dataset and make it even more useful for researchers. To combine real pedestrians with synthetic ones, we propose a pre-processing step using the YoloV5 model to detect the bounding boxes of people in the dataset. These bounding boxes serve as obstacles for synthetic objects, which interact with them while navigating the generated scene. To enhance the integration of 3D models with the low-resolution real parts of the scene, we employ various post-processing shaders. Moreover, we dynamically generate occlusions for foreground pedestrians, ensuring realistic occlusion of synthetic objects by real people. Additionally, we provide a manual background occlusion tool, enabling users to specify which portions of the real background should occlude synthetic objects. We use three types of synthetic objects - humans, animals, and carry-on objects like purses and backpacks, with a number of anomalies connected with each. The proposed rendering pipeline is built through Unity and is easily extendable through the addition of more models, animations, and behaviors. The pipeline can be expanded to other datasets as well.

For testing, we select five state-of-the-art anomaly detectors that achieve above 90% accuracy on the real anomalies on the UCSD datasets. We test them on anomalies created through the proposed rendering pipeline. All five models experience significantly lower performance and struggle to detect all synthetic anomalous frames, showing that anomalies created through the pipeline can make the UCSD dataset more challenging and thus more useful for developing such algorithms. In addition, we generate synthetic normal pedestrian data and use it together with the real training images to see if it can boost the performance of the anomaly detectors on the real anomalies. We achieve mixed results with some detectors getting better results, while others getting worse accuracy.

In summary, the paper's main contributions are:

1. An extendable synthetic data rendering pipeline that combines real dataset foregrounds and backgrounds with synthetic objects that dynamically react to the real ones for generating randomized training and testing data for anomaly detection;
2. To our knowledge the first solution that can add synthetic items like handbags and briefcases to real pedestrians and creating a new type of anomaly scenario by dropping them;
3. Method for occlusion of synthetic objects by dynamic real foreground pedestrians;
4. Demonstrating that the synthetic anomalies can make an old dataset more challenging and the synthetic normal data can boost model accuracy.

2. Related Work

2.1. Anomaly Detection Datasets

Anomaly detection datasets can be separated into single-scene and multiple-scene. Widely used single-scene datasets like UCSD [MLBV10], Avenue [LSJ13], StreetScene [RJ20], and ADOC [PZ*20] train and test models that rely on temporal and movement information and capture a normality model specific to a certain dataset. On the other hand, multiple-scene models like UMN [MOS09] and ShanghaiTech [LLLG18] are used to train more generic models that learn a normality model that can be used between datasets but are more constrained to representations that do not follow specific temporal changes from pedestrians, vehicles, etc. Anomalies, by their very nature, are scene-specific and challenging to capture and reproduce in a natural and unstaged manner. Consequently, anomaly detection datasets predominantly consist of normal data, far outnumbering abnormal instances. This leads to the prevalence of open-set benchmarks in these datasets, where training subsets only contain normal data or, if anomalies are present, they differ from those in the testing subsets. As a result, most datasets offer a limited number of anomalies primarily related to pedestrian activities or vehicles. Unfortunately, this lack of anomaly variety makes the datasets appear staged and artificial, deviating from the complete representation of reality. Consequently, they fail to provide the necessary challenges to accurately evaluate the performance of anomaly detection models. Additionally, not all datasets provide pixel-level annotations for anomalies, instead offering only frame-level annotations. This limitation reduces their usefulness in scenarios where precise identification of anomaly type and location is crucial.

2.2. Synthetic Datasets

The reliance on deep learning models on large quantities of data has pushed the development of more and more synthetic datasets for different use cases. Self-driving cars [WU18] have greatly benefited from synthetic datasets, together with indoor [CWB*20] and outdoor [EDV*22] scene understanding and semantic segmentation. Synthetic datasets can be roughly divided into two main groups - fully synthetic and combined real and synthetic. Fully synthetic datasets rely on data gathered either through games [LYA*22], game engines [Uni17], rendering engines like Blender or Maya [WMH18], or through deep diffusion models and GANs [HSY*22]. This way of generating synthetic data has the benefit of a potentially unlimited number of unique scenarios and visuals. It also has the significant drawback of a large distribution gap between the fully synthetic data and the real-life data, which can hurt the accuracy and robustness of models using it for training and testing. A lot of times to limit the distribution gap, additional post-processing of the synthetic data is required using GANs, which limits the ease of use and versatility of such datasets.

On the other hand, datasets created by combining real-world data with synthetic objects aim to naturally shorten the distribution gap between the dataset and real data. These datasets can be additionally subdivided into two types - ones that the synthetic parts are images composited onto other images [CLU*21, TCA*19], and ones that use 3D models and game or rendering engines to augment real images with synthetic parts [EJZ*21].

Two of the state-of-the-art datasets for anomaly detection, utilizing this method of augmenting real image backgrounds with 3D models are the works by Madan et al. [MSG*23] and Acsintoae et al. [AFG*22]. The work by Madan et al. focuses on thermal imaging data and only synthesizes falling human anomalies, making it more limited. The work by Acsintoae et al. focuses on creating a large variety of anomalies and normal scenarios but only uses static backgrounds and does not give an easy way to generate more data as the authors estimate it took them 41.1 days to render the dataset of 236,902 images.

To the best of our knowledge, we present the first rendering pipeline for augmenting an existing widely known dataset for urban anomaly detection. We aim to make the pipeline flexible enough that additional synthetic augmentations can be added easily like more models, animations, scenarios, and even different types of anomalies. Our solution is also the first one to contain moving synthetic objects and real pedestrians in the same video sequences, with the synthetic objects reacting to the real ones both movement-wise and visually. Our rendering pipeline can generate training, validation, and testing data, as well as frame-level and pixel-level annotations.

3. Proposed Rendering Pipeline

In this paper, we created a rendering pipeline that can be used to generate synthetic training and testing data to augment a widely used, but ultimately "solved" dataset - the UCSD [MLBV10]. This pipeline can then be used together with the dataset to generate new training and testing data, based on a user's requirements. We will give a step-by-step overview of the different parts of the pipeline in this section. An overview of the whole process can be seen in Figure 1.

3.1. Dataset Pre-processing

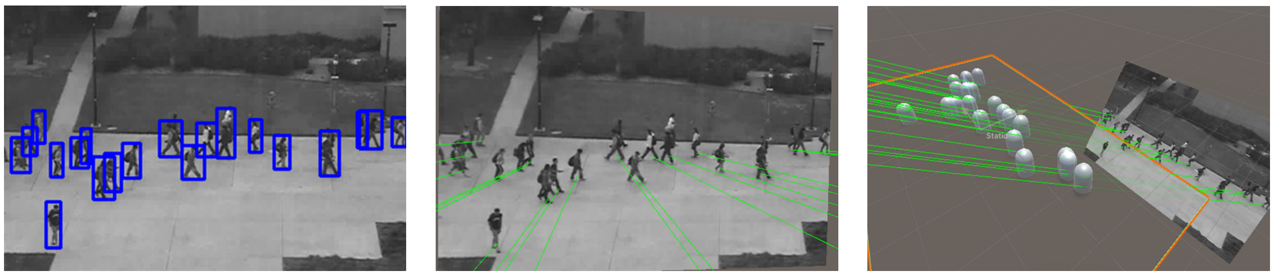
The UCSD dataset needs to be pre-processed before it can be used as part of the dataset rendering pipeline in Unity. To correctly position the Unity cameras corresponding roughly to the one used to capture the dataset, the extrinsic and intrinsic parameters of the camera need to be found. As these are not known, we have chosen to approximate them using fSpy [Gan18]. This is done by manually selecting two sets of perpendicular vanishing points on an input image, from which the software approximates the focal length, orientation, and position of the camera that captured these images. Because the UCSD dataset is comprised of two subsets each with a static camera, we need to do this two times. The PEDS1, with a camera oriented towards the movement of pedestrians, and the PEDS2, with a camera oriented perpendicular to the movement. Once these parameters are calculated they can be exported to Blender where a 3D camera, a plane representing the ground seen in the images, and a projection plane on which the images are shown are created and later exported to Unity.

We also need to know the positions of the real pedestrians in each image and their relative sizes, we run the subsets through the YoloV5 [JSB*20] object detector. For each image, the bounding boxes of the detected pedestrians are extracted for use in Unity (Figure 2a). We have chosen the YoloV5 algorithm as it provides very good detection performance on images with limited resolution and color information, which the grayscale 360x240 and 238x158 UCSD images are. YoloV5 provides feature interchangability between the semantic and context layers, giving it higher robustness to different camera angles, orientations, and scales. Using this model makes it easier to extend our solution to other datasets later on.

The bottom part of the calculated bounding box for each detected pedestrian is used to shoot a ray toward the camera's 3D position (Figure 2b). Where these rays hit the designated ground plane a 3D capsule volume is created. This volume is scaled based on the distance from the bounding box to the 3D camera (Figure 2c). These capsule volumes will be later used for both dynamic occlusion of the synthetic objects, as well as obstacles for their pathfinding movement.

3.2. Synthetic Object Visualization and Blending

Cameras used for surveillance are quite often lower-resolution and susceptible to environmental noise degradation and compression artifacts [TNR*12]. We introduce a number of post-processing shaders that try to mimic this behavior for the added 3D models, together with transforming them from RGB to grayscale to blend better with the real backgrounds. In addition, we also introduce a transparent shadow catcher shader used on the ground surface plane, so the 3D models can have correct shadowing. Finally, as we want to



(a) *YoloV5 with bounding boxes* (b) *Raycasts from each pedestrian* (c) *3D positions on the walking plane*

Figure 2: *Pre-processing steps for extracting the real pedestrians from the dataset and finding their 3D positions. The YoloV5 model is used to detect the pedestrians in each frame (Figure 2a). A ray is shot from each bounding box on the projection plane towards the 3D camera found using fSpy (Figure 2b). Where these rays intersect the walking plane a capsule volume is created representing the real pedestrian's 3D position (Figure 2c).*

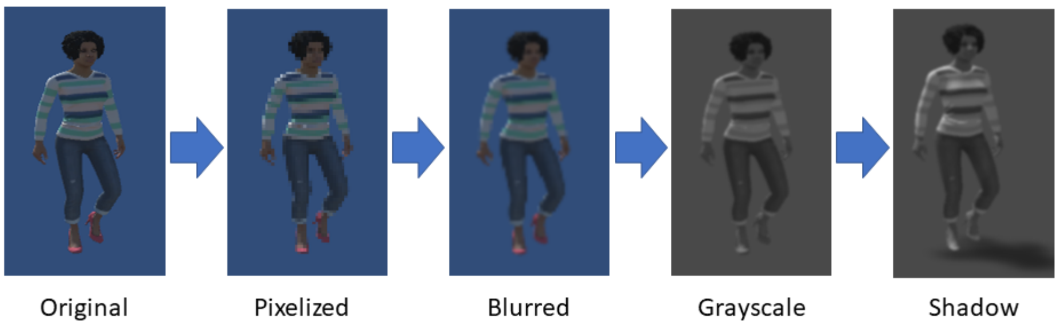


Figure 3: *Shader processing of the 3D objects. The original image of the 3D model is first pixelized, then blurred in the horizontal and vertical direction using a Gaussian filter. The resultant image is then transformed to grayscale and finally, the shadow of the model is visualized on the transparent walking surface*

capture ground truth data for the synthetic anomalies we also use the Unity ML-ImageSynthesis package [Uni17] for extracting instance and semantic segmentation. Below we explain each of the post-processing steps shown in Figure 3.

1. **Pixelize:** To achieve the image degradation that happens from jpeg and low bitrate compression in CCTV cameras, like the one used for the UCSD dataset, we use a pixelize shader to approximate the block artifacts in the real moving pedestrians. For each synthetic model, a bounding cube is created capturing the whole object, and the pixelize shader is applied to this cube. We capture the contents of the framebuffer into a texture and pixelize it using Equation 1, where T is the texture coordinates, R is the camera target resolution, and S_p is the pixelization size.

$$\mathrm{T_{pixelate}} = \mathrm{round}\left(T \div \frac{S_p}{R}\right) \cdot \frac{S_p}{R}, \tag{1}$$

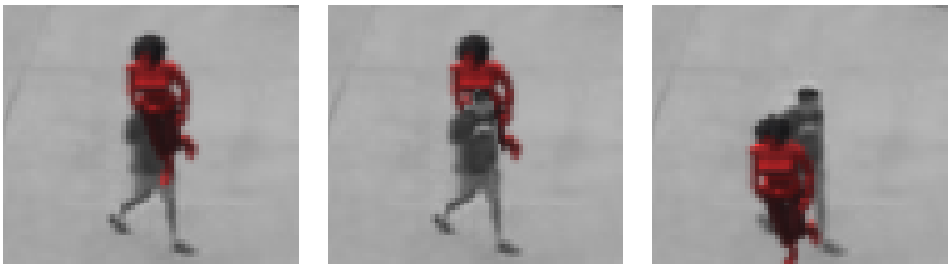
2. **Gaussian blur:** The 3D models have sharp edges even after the pixelization, while the real pedestrians have a smoother look from the lower quality of the captured images and the motion blur when moving. To achieve a similar effect we use Gaussian blur on the framebuffer texture, applied with a 3x3 kernel to each pixel. The implementation is based on the one presented by Daniel Illet [Ile22].
3. **Grayscale:** To achieve the grayscale visualization we take the fragment colors of the synthetic models and use the weighted Equation 2 to transform them into grayscale, preserving their overall luminosity, where R, G, and B are the three color channels

$$C_{gray} = 0.30 \cdot R + 0.59 \cdot G + 0.11 \cdot B, \tag{2}$$

4. **Transparent Shadow Collector:** As the ground plane is transparent, we create a simple shadow catcher shader, which only visualizes shadows. This is done by combining alpha blending together with getting the light attenuation from directional light that can be manually oriented to better represent the light direction present in the images. A default orientation of the lighting sources is set orthogonal to the walking plane.

3.3. Occlusions between Real and Synthetic

One of the main problems with augmenting synthetic elements in real footage is the incorrect occlusions that can happen if parts of the background should hide a synthetic element, but it is just rendered on top, creating incorrect depth cues. This becomes even



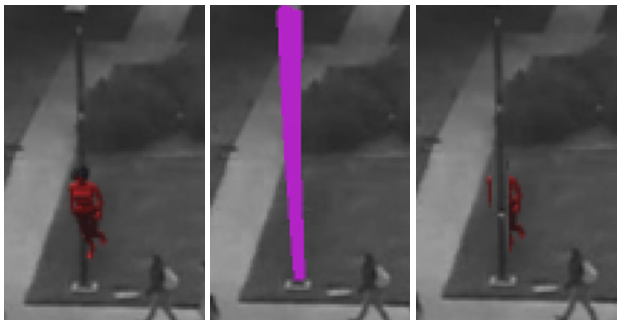
(a) *Synthetic behind - naive incorrect occlusion* **(b)** *Synthetic behind - shader correct occlusion* **(c)** *Synthetic front - shader correct occlusion*

Figure 4: *Occlusion problem between real pedestrians and synthetic objects (Figure 4a). To address the problem, we use the calculated 3D capsule volumes for real pedestrians and a custom transparency queue shader on both the capsules and the synthetic objects. The result gives correct occlusions both when the synthetic object is behind (Figure 4b) and in front of the real pedestrians (Figure 4c). The synthetic human is shown with a red tint for easier visualization.*

more complicated when we factor in that real moving pedestrians will be present in the images and the synthetic objects need to be dynamically occluded as necessary. We address both problems in the proposed rendering pipeline.

For the dynamic occlusion between real and synthetic objects, we use the capsule volumes created for marking where the real pedestrians bounding boxes are on the 3D walk path. As the transparency queue is sorted normally back to front with objects in the back rendered first, we create shaders with different queue numbers for the ground plane, the synthetic objects, and the capsule volumes. We set the queue of the volumes lower than the synthetic objects, but higher than the ground plane. This way whenever the fully transparent capsule volume is in front of the synthetic 3D object, it will be rendered after it and create an occlusion volume. If the 3D object is in front of the capsule volume then it will be rendered first and remain visible. The naive approach together with our proposed solution are given in Figure 4.

For selecting parts of the background that should occlude the synthetic objects, we introduce an initial manual masking step. The user selects points on the image, which encompass parts of the background that should occlude the synthetic 3D objects behind them. Examples of this can be trees, lamp posts, bushes, signs, etc. Once the user has selected everything that they deem an occlusion part of the scene, these points are used as vertices to create a polygonal plane, oriented toward the camera. As a basis for this, we use the Unity library BMesh [Mic20]. The plane is given the same transparent material with a render queue higher than the 3D objects. Because the created polygonal plane is created closer to the camera it will always occlude everything behind it (Figure 5).



(a) *Background occlusion problem* **(b)** *Occlusion region selection mesh* **(c)** *Correct background occlusion*

Figure 5: *Occlusion problem between the background and synthetic objects (Figure 5a). To address this problem we introduce an initial manual selection step, where the user can draw polygons with the mouse over parts of the scene that should occlude the synthetic objects (Figure 5b). The generated polygons have the same shader material used for the dynamic occlusions with pedestrians. The result can partially or fully occlude synthetic objects behind parts of the scene (Figure 5c) The synthetic human is shown with a red tint for easier visualization.*

3.4. Implemented Synthetic Anomalies

Examples of the three different types of anomalies implemented for the paper are given in Figure 6. Most datasets connected to outdoor urban anomaly detection focus on pedestrians and their actions [AFG*22, RJ20, LSJ13]. The UCSD dataset also contains mainly pedestrian and vehicle anomalies. We have selected to focus on generating synthetic pedestrian anomalies as for the first type of anomalies incorporated in the presented solution. For this 6 models are selected from Adobe Mixamo [Bla14], representing men and women with various skin colors and clothes. Each person is given one normal walking and idle animation and 6 anomalous walking animations - *injured walk*, *injured jog*, *drunken walk*, *drunken jog*, *limping walk* and *limping jog*. In addition, each model can also have 6 anomalous behaviors - *trip and fall*, *stumble back*, *fall forward*, *jump forward*, *go limp and fall* and *backflip*. The models can be set to walk on the walking path together with the real pedestrians or create further anomalous scenarios by walking on the grass paths.

The second type of anomaly we have selected is a class that is very rarely represented in any real or synthetic urban area datasets - animals. Most examples of such anomalies are made for non-static datasets connected to self-driving cars and the created animal anomalies are very rudimentary [GBŠ21, CLU*21]. By introducing

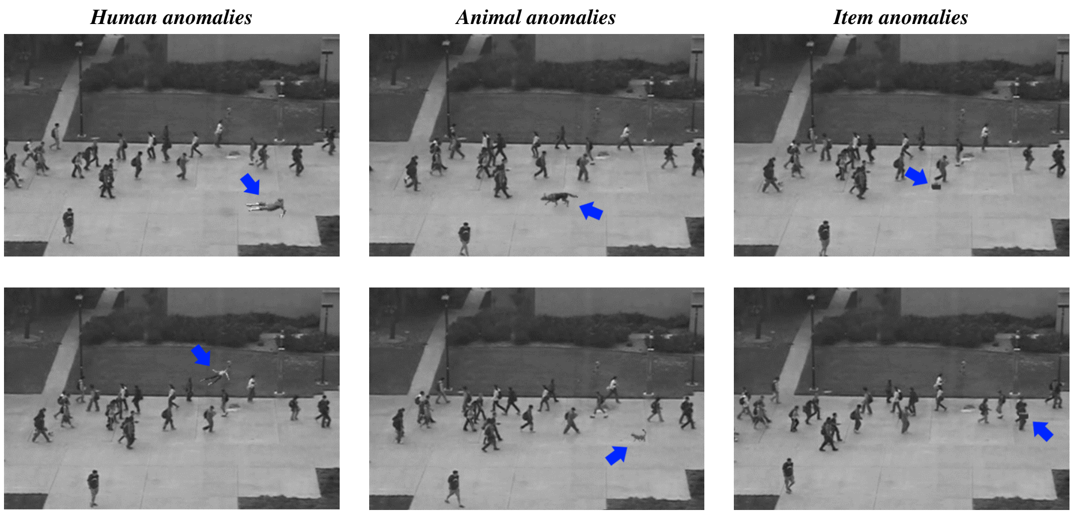
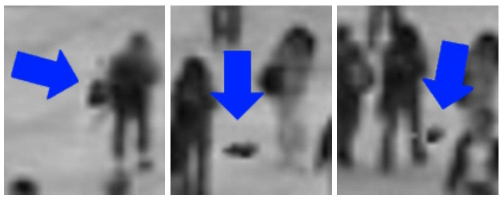


Figure 6: *Example synthetic anomalies*

animal 3D objects to the UCSD we aim to diversify its anomaly detection potential. For our proposed solution we have selected 5 animal models - two cats, two dogs, and a horse from the dataset provided by Truebone [Tru20]. All 3D models come with walking and idle animations. We have selected these animals as they showcase various sizes and colors and will complicate anomaly detection. The presence of the animals themselves is designated as an anomaly.

The third type of anomaly we have implemented is considered the most challenging - a person carrying an item and switching it or dropping it on the ground. To our knowledge, there are no other synthetic datasets that contain this type of anomaly, while the detection of such anomalies can be vital to the security of many places like airports, bus stops, parks, etc. Two types of items are implemented as part of this paper - a handbag and a briefcase. To be able to create such anomalies without the need for synthetic pedestrians, we have chosen to use the already known positions of the real people in the dataset. When this type of anomaly is selected, real pedestrians are chosen and the synthetic items are connected to their position, either in the left or right hand. The item is moved with the tracked pedestrian until the tracking is lost at which point the item is switched to the closest pedestrian or until the item is dropped on the ground. The items move with the real pedestrians (Figure 7a), behave as physical entities when dropped (Figure 7b), and have the same correct occlusion properties as the other synthetic objects (Figure 7c).

For making the movements and interactions between the synthetic objects and real pedestrians more believable we utilize the NavMesh library in Unity, with the detected real pedestrians set as dynamic obstacles. Each time a new image is loaded together with the detected pedestrians, the movement paths of all synthetic humans and animals are recalculated, and once they reach their goal they transition to an idle state.



(a) *Holding* **(b)** *Dropped* **(c)** *Occluded*

Figure 7: *Examples of synthetic bags augmented onto real pedestrians. The bag is attached to a random pedestrian and moves with them(Figure 7a) until an anomaly event, is created where the bag is dropped and behaves like a physical object, which can interact with the real and synthetic pedestrians (Figure 7b). The bag has the same shaders on it providing it with correct dynamic occlusion from real people (Figure 7c)*

3.5. Randomization for Data Gathering

As the main idea of the proposed solution is to be able to generate both training and testing data, based on the requirements of the users, we have added a large number of randomization parameters that can be tweaked to create different image augmentations.

The main distinction is for generating training and testing data. When training data is selected only synthetic pedestrians walking and stopping on the main path are created. Their starting and goal positions are selected randomly and the users can choose how many pedestrians there can be or if their number should be randomized. If testing data is selected then the user can choose which and how many of the three anomaly types they would like to add to the data. In addition, users can tweak the movement speed of the synthetic objects, the type of movement and when an anomaly should happen. For synthetic items, the possible randomized factors are, to which person they will be attached, their position on the left or right side of the pedestrian, as well as when the item will be dropped.

The number of augmented full video variations can be selected so a large number of training and testing sequences can be created one after the other. Finally, if anomalies are selected for synthetic pedestrians and items the frame in which the anomaly occurs is saved in a separate file so it can be used to create frame-level annotations. The augmented grayscale image and ground truths are saved for each processed frame from a sequence.

4. Experiments

To show the usefulness of our proposed synthetic dataset rendering pipeline we test five state-of-the-art anomaly detection models in two main experiments - MNAD$_{\text{pred}}$ / MNAD$_{\text{recon}}$ [PNH20], LGN-Net [ZLL*22], MPN [LCC*21], and LNTRA [AZLL21].

4.1. Anomaly Detection Models Setup

Each of the five selected models is trained using the parameters given by their respective authors. For the MNAD model, we train both the predictive (MNAD$_{\text{pred}}$) and reconstruction (MNAD$_{\text{recon}}$) variations. For the MNAD$_{\text{pred}}$ model we set both the compactness and separation losses to 0.1, while for the MNAD$_{\text{recon}}$, they are set to 0.01. Both variations are trained using a batch size of 4 and a learning rate of 0.0002 for 60 epochs and an Adam optimizer. For the LGN-Net model, we set the compactness and separation losses to 10 and 5 respectively. The Adam optimizer is used, together with a batch size of 6 and a learning rate of 0.0002 for 60 epochs. For the MPN model, we set the frame and feature reconstruction loss weights to 1 and the feature distinction loss weight to 0.1. We again use the Adam optimizer, with a batch size of 4 and a learning rate of 0.0001, and train for 100 epochs. Finally, for the LNTRA model we use the skip frame-based implementation, with a pseudo anomaly jump inpainting of 0.2 and jumps to $2,3,4,5$ pseudo anomalies. The batch size is set to 4 and the learning rate is set to 0.0001, with the model trained for 60 epochs. For an evaluation metric for both experiments, we use the widely-adopted area under the curve (AUC) based on frame-level annotations.

4.2. Synthetic Testing Data Experiment

First, we want to see how the performance of the models fairs against the more complex and diverse anomalies presented by our pipeline. For this, we use both PEDS1 and PEDS2 and generate three synthetic testing sets for each of the three main anomaly types. To not mix real and synthetic anomalies we augment one of the training video sequences for each subset. We generate 14 synthetic sequences for each anomaly type, giving us $8,672$ frames for PEDS1 and $6,496$ frames for PEDS2. To make the testing scenario harder we only use one anomalous object in each sequence. We compare the performance of the models on the real anomalies versus their performance on the synthetic ones.

4.3. Synthetic Training Data Experiment

The second experiment is aimed at seeing if introducing augmented normal data to the training process can be used to boost the accuracy of the models. For this, we only use sequences from PEDS2 as it is the more widely used one and select a number of the training sequences of the UCSD dataset. We generate augmented variations of them, using between 2 and 6 synthetic pedestrians that do not exhibit anomalous behavior. We generate 68 training sequences, containing $10,630$ frames. We train the five models on a combination of real and synthetic training data and compare the results to the performance of the models only trained on real data.

5. Results and Discussion

The results from the synthetic testing data experiments are given in Table 1. We first test all the models on the real testing data and can see that all of them achieve above 90% accuracy on frame-level anomaly detection on the more widely used PEDS2 dataset and above 75% accuracy on the PEDS1 dataset. Especially the LGN-Net and LNTRA model achieving accuracies that would be considered enough to saturate the PEDS2 subset. Compared to that, the synthetic anomalies achieve much lower accuracy, with the item anomalies achieving the worst with an average of 43.76 for PEDS2 and 45.21 for PEDS1 and the animal anomalies having the best average one between all models of 60.50 for PEDS2 and 63.80 for PEDS1. The synthetic human anomalies also prove problematic with an average accuracy of 43.90 for PEDS2 and 47.12 for PEDS1. These results can be explained by the animals being easier to detect as just being in the frame is considered an anomaly, even though they have the largest difference between the separate anomalies - horses being easy to detect, while cats proving much harder. The small size of the item anomaly proves hard to detect. Finally, humans prove hard as synthetic pedestrians stop moving once they experience an anomaly. Another more problematic explanation for the human anomalies is that the synthetic objects do not completely blend with the real ones and the detectors are flagging them as anomalies before an anomalous action is seen. To see if this is a problem we do a second study on the PEDS2 subset by running the synthetic human anomaly dataset through the detectors trained on the combination of real and synthetic data. Here we calculate the precision and recall of each model, as they will give us a better idea of the false positive detection of normal synthetic images as anomalies (Table 2).

We can see that the recall for (MNAD$_{\text{pred}}$), and especially for (MNAD$_{\text{recon}}$) and LNTRA gets significantly higher using the model trained on real and synthetic images, pointing to the fact that more synthetic data helps to detect the true anomalies. The precision for these models on the other hand also gets higher but much less so, specifying that showing synthetic humans to the models helps in misidentifying them as anomalies. The other two models LGN-Net and MPN, on the other hand, get worse results, showing that even showing synthetic humans to the models does not guarantee that they will perform better when the anomalies are as complicated as the ones generated by our solution.

The results from the synthetic training data test are given in Table 3. For this test, we can show that some of the models like MPN and MNAD$_{\text{pred}}$ achieve higher frame-level accuracy when trained on a combination of real and synthetic data, while other models get varying degrees of lower accuracy, with the LGN-NET model suffering the strongest degradation. The degradation in some of the models can be explained by not enough variation with the gener-

Table 1: *Frame-level results from the five state-of-the-art models on the real vehicle anomalies in PEDS1 and PEDS2, together with the human, animal, and item synthetic anomalies generated by our proposed pipeline. We can see that even though the real anomalies saturate PEDS2 and to some extent PEDS1, the synthetic anomalies provide a much greater overall challenge. Thus making the dataset again useful to researchers and valuable for testing.*

	PEDS1 Real	PEDS1 Synthetic			PEDS2 Real	PEDS2 Synthetic		
Models	Vehicles	Humans	Animals	Items	Vehicles	Humans	Animals	Items
MNAD$_{\text{recon}}$	68.22	24.07	52.87	60.73	90.64	46.32	37.93	35.81
MNAD$_{\text{pred}}$	77.35	50.14	63.27	49.95	92.12	45.55	82.45	50.64
LGN-Net	78.94	48.79	65.83	56.68	97.07	62.49	45.51	36.58
MPN	80.23	46.56	64.91	49.70	96.13	23.22	49.31	37.70
LNTRA	88.77	56.49	72.13	18.55	96.50	41.93	87.32	58.08

Table 2: *Second study using the synthetic human anomaly testing data. The precision and recall of the five models trained only on real data and trained on real and synthetic data are compared to see if exposing the models to synthetic humans throughout training will have large changes in their performance. This is done as a measure to verify if the generated synthetic data has a large distribution gap with the real images.*

	Real		Real + Synthetic	
Model	Precision	Recall	Precision	Recall
MNAD$_{\text{recon}}$	0.239	0.420	**0.519**	**0.645**
MNAD$_{\text{pred}}$	0.271	0.932	**0.289**	**0.934**
LGN-Net	**0.299**	**0.967**	0.276	0.853
MPN	**0.199**	**0.544**	0.173	0.432
LNTRA	0.189	0.454	**0.338**	**0.835**

Table 3: *Frame-level results from the five state-of-the-art anomaly detector models trained only on the real data compared to being trained on a combination of real and synthetic data. Only the PEDS2 subset of the UCSD dataset is used. We can see that two of the models achieve higher accuracy, while the other three achieve slightly worse results, which can be attributed to not enough diversity of the synthesized data.*

Model	AUC$_{\text{real}}$	AUC$_{\text{real+synthetic}}$
MNAD$_{\text{recon}}$	**90.64**	88.17
MNAD$_{\text{pred}}$	92.12	**95.33**
LGN-Net	**97.07**	91.24
MPN	96.13	**96.94**
LNTRA	**96.50**	94.92

ated augmented training data resulting in overfitting. This can be remedied by generating training data from more of the real data and using a larger number of synthetic objects to give a better variation.

6. Conclusion

We propose a rendering pipeline for generating training and testing data in surveillance anomaly detection, focusing on augmenting the well-known UCSD dataset. To bridge the gap between synthetic and real data, we introduce a novel approach that combines real backgrounds and moving pedestrians with synthetic objects. This augmentation ensures accurate occlusions between real and synthetic objects, along with preprocessing techniques to enhance their blending. The pipeline is versatile, allowing the generation of diverse normal and anomalous data, and can be expanded to include different scenarios, emergencies, and synthetic objects. Furthermore, we provide both frame-based and pixel-based annotations.

To evaluate the effectiveness of our pipeline, we generate three types of synthetic anomalies: synthetic pedestrians in anomalous scenarios, animals within the camera's field of view, and real pedestrians carrying and dropping synthetic items. We select five state-of-the-art anomaly detection models that exhibit over 90% accuracy on simple real anomalies already present in the dataset. However, when tested on our generated synthetic anomalies, the accuracy of these models significantly decreases. This highlights the increased difficulty and variability introduced by our synthetic anomalies, making the dataset a valuable testbed for researchers to evaluate new anomaly detection methods. Additionally, we generate augmented normal data for training, demonstrating that it can improve the accuracy of certain models.

For future work, we intend to extend the pipeline to incorporate other widely used anomaly datasets such as Avenue and ShanghaiTech, while also introducing new anomalies and scenarios. We also plan to enhance the rendering pipeline to simulate weather condition changes like rain, snow, and fog. Previous research [NPL*21] has shown that such environmental factors can negatively impact model performance in real-world deployments, making it essential to include them for comprehensive evaluation.

References

[AFG*22] Acsintoae A., Florescu A., Georgescu M.-I., Mare T., Sumedrea P., Ionescu R. T., Khan F. S., Shah M.: Ubnormal: New benchmark for supervised open-set video anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 20143–20153. 2, 3, 5

[AZLL21] Astrid M., Zaheer M. Z., Lee J.-Y., Lee S.-I.: Learning not to reconstruct anomalies. *arXiv preprint arXiv:2110.09742* (2021). 2, 7

[Bla14] Blackman S.: *Rigging with Mixamo*. Apress, Berkeley, CA, 2014, pp. 565–573. 5

[CLU*21] Chan R., Lis K., Uhlemeyer S., Blum H., Honari S., Siegwart R., Fua P., Salzmann M., Rottmann M.: Segmentmeifyoucan: A benchmark for anomaly segmentation. *arXiv preprint arXiv:2104.14812* (2021). 3, 5

[COCH21] Corona K., Osterdahl K., Collins R., Hoogs A.: Meva: A large-scale multiview, multimodal video dataset for activity detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (2021), pp. 1060–1068. 2

[CWB*20] Cruz S. D. D., Wasenmuller O., Beise H.-P., Stifter T., Stricker D.: Sviro: Synthetic vehicle interior rear seat occupancy dataset and benchmark. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (2020), pp. 973–982. 3

[EDV*22] Ebadi S. E., Dhakad S., Vishwakarma S., Wang C., Jhang Y.-C., Chociej M., Crespi A., Thaman A., Ganguly S.: Psp-hdri +: A synthetic dataset generator for pre-training of human-centric computer vision models. *arXiv preprint arXiv:2207.05025* (2022). 3

[EJZ*21] Ebadi S. E., Jhang Y.-C., Zook A., Dhakad S., Crespi A., Parisi P., Borkman S., Hogins J., Ganguly S.: Peoplesanspeople: a synthetic data generator for human-centric computer vision. *arXiv preprint arXiv:2112.09290* (2021). 3

[FBM*21] Fabbri M., Brasó G., Maugeri G., Cetintas O., Gasparini R., Ošep A., Calderara S., Leal-Taixé L., Cucchiara R.: Motsynth: How can synthetic data help pedestrian detection and tracking? In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 10849–10859. 2

[Gan18] Gantelius, Per: fspy. https://fspy.io/, 2018. 3

[GBŠ21] Grcić M., Bevandić P., Šegvić S.: Dense anomaly detection by robust learning on synthetic negative data. *arXiv preprint arXiv:2112.12833* (2021). 2, 5

[HSY*22] He R., Sun S., Yu X., Xue C., Zhang W., Torr P., Bai S., Qi X.: Is synthetic data from generative models ready for image recognition? *arXiv preprint arXiv:2210.07574* (2022). 3

[Ile22] Ilett D.: More shader fundamentals. In *Building Quality Shaders for Unity®: Using Shader Graphs and HLSL Shaders*. Springer, 2022, pp. 321–356. 4

[JSB*20] Jocher G., Stoken A., Borovec J., NanoCode012, ChristopherSTAN, Changyu L., Laughing, tkianai, Hogan A., lorenzomammana, yxNONG, AlexWang1900, Diaconu L., Marc, wanghaoyang0106, ml5ah, Doug, Ingham F., Frederik, Guilhen, Hatovix, Poznanski J., Fang J., Yu L., changyu98, Wang M., Gupta N., Akhtar O., PetrDvoracek, Rai P.: ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements, Oct. 2020. URL: https://doi.org/10.5281/zenodo.4154370, doi:10.5281/zenodo.4154370. 3

[LCC*21] Lv H., Chen C., Cui Z., Xu C., Li Y., Yang J.: Learning normal dynamics in videos with meta prototype network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 15425–15434. 2, 7

[LLLG18] Liu W., Luo W., Lian D., Gao S.: Future frame prediction for anomaly detection–a new baseline. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 6536–6545. 2

[LSJ13] Lu C., Shi J., Jia J.: Abnormal event detection at 150 fps in matlab. In *Proceedings of the IEEE international conference on computer vision* (2013), pp. 2720–2727. 2, 5

[LYA*22] Li C., Yi R., Ali S. G., Ma L., Wu E., Wang J., Mao L., Sheng B.: Radepthnet: Reflectance-aware monocular depth estimation. *Virtual Reality & Intelligent Hardware 4*, 5 (2022), 418–431. 3

[Mic20] Michel, Elie: Bmesh. https://github.com/eliemichel/BMeshUnity, 2020. 5

[MLBV10] Mahadevan V., Li W., Bhalodia V., Vasconcelos N.: Anomaly detection in crowded scenes. In *2010 IEEE computer society conference on computer vision and pattern recognition* (2010), IEEE, pp. 1975–1981. 2, 3

[MOS09] Mehran R., Oyama A., Shah M.: Abnormal crowd behavior detection using social force model. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 935–942. doi:10.1109/CVPR.2009.5206641. 2

[MSG*23] Madan N., Siemon M. S. N., Gjerde M. K., Petersson B. S., Grotuzas A., Esbensen M. A., Nikolov I. A., Philipsen M. P., Nasrollahi K., Moeslund T. B.: Thermalsynth: A novel approach for generating synthetic thermal human scenarios. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (2023), pp. 130–139. 2, 3

[NPL*21] Nikolov I. A., Philipsen M. P., Liu J., Dueholm J. V., Johansen A. S., Nasrollahi K., Moeslund T. B.: Seasons in drift: A long-term thermal imaging dataset for studying concept drift. In *Thirty-fifth Conference on Neural Information Processing Systems* (2021), Neural Information Processing Systems Foundation. 8

[PNH20] Park H., Noh J., Ham B.: Learning memory-guided normality for anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), pp. 14372–14381. 2, 7

[PZ*20] Pranav M., Zhenggang L., et al.: A day on campus-an anomaly detection dataset for events in a single camera. In *Proceedings of the Asian Conference on Computer Vision* (2020). 2

[RHK17] Richter S. R., Hayder Z., Koltun V.: Playing for benchmarks. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 2213–2222. 2

[RJ20] Ramachandra B., Jones M.: Street scene: A new dataset and evaluation protocol for video anomaly detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (2020), pp. 2569–2578. 2, 5

[TCA*19] Tripathi S., Chandra S., Agrawal A., Tyagi A., Rehg J. M., Chari V.: Learning to generate synthetic data via compositing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 461–470. 3

[TNR*12] Tsifouti A., Nasralla M. M., Razaak M., Cope J., Orwell J. M., Martini M. G., Sage K.: A methodology to evaluate the effect of video compression on the performance of analytics systems. In *Optics and Photonics for Counterterrorism, Crime Fighting, and Defence VIII* (2012), vol. 8546, SPIE, pp. 235–249. 3

[TPC*21] Tian Y., Pang G., Chen Y., Singh R., Verjans J. W., Carneiro G.: Weakly-supervised video anomaly detection with robust temporal feature magnitude learning. In *Proceedings of the IEEE/CVF international conference on computer vision* (2021), pp. 4975–4986. 2

[Tru20] Truebones Motion: Truebones zoo. https://truebones.com/, 2020. 6

[Uni17] Unity Technologies: Unity Image synthesis. https://bitbucket.org/Unity-Technologies/ml-imagesynthesis/src/master/, 2017. 3, 4

[WMH18] Ward D., Moghadam P., Hudson N.: Deep leaf segmentation using synthetic data. *arXiv preprint arXiv:1807.10931* (2018). 3

[WU18] Wrenninge M., Unger J.: Synscapes: A photorealistic synthetic dataset for street scene parsing. *arXiv preprint arXiv:1810.08705* (2018). 2, 3

[XWY*19] Xu Y., Wang K., Yang K., Sun D., Fu J.: Semantic segmentation of panoramic images using a synthetic dataset. In *Artificial Intelligence and Machine Learning in Defense Applications* (2019), vol. 11169, SPIE, pp. 90–104. 2

[ZLK*19] Zhong J.-X., Li N., Kong W., Liu S., Li T. H., Li G.: Graph convolutional label noise cleaner: Train a plug-and-play action classifier for anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 1237–1246. 2

[ZLL*22] Zhao M., Liu Y., Liu J., Li D., Zeng X.: Lgn-net: Local-global normality network for video anomaly detection. *arXiv preprint arXiv:2211.07454* (2022). 2, 7