algorithms

MDPI

Article

Fusion of CCTV Video and Spatial Information for Automated Crowd Congestion Monitoring in Public Urban Spaces

Vivian W. H. Wong and Kincho H. Law *

Department of Civil and Environmental Engineering, Stanford University, Stanford, CA 94305, USA; vwwong3@stanford.edu
* Correspondence: law@stanford.edu

Abstract: Crowd congestion is one of the main causes of modern public safety issues such as stampedes. Conventional crowd congestion monitoring using closed-circuit television (CCTV) video surveillance relies on manual observation, which is tedious and often error-prone in public urban spaces where crowds are dense, and occlusions are prominent. With the aim of managing crowded spaces safely, this study proposes a framework that combines spatial and temporal information to automatically map the trajectories of individual occupants, as well as to assist in real-time congestion monitoring and prediction. Through exploiting both features from CCTV footage and spatial information of the public space, the framework fuses raw CCTV video and floor plan information to create visual aids for crowd monitoring, as well as a sequence of crowd mobility graphs (CMGraphs) to store spatiotemporal features. This framework uses deep learning-based computer vision models, geometric transformations, and Kalman filter-based tracking algorithms to automate the retrieval of crowd congestion data, specifically the spatiotemporal distribution of individuals and the overall crowd flow. The resulting collective crowd movement data is then stored in the CMGraphs, which are designed to facilitate congestion forecasting at key exit/entry regions. We demonstrate our framework on two video data, one public from a train station dataset and the other recorded at a stadium following a crowded football game. Using both qualitative and quantitative insights from the experiments, we demonstrate that the suggested framework can be useful to help assist urban planners and infrastructure operators with the management of congestion hazards.

Keywords: deep learning; computer vision; graph representation learning; crowd congestion

Citation: Wong, V.W.H.; Law, K.H. Fusion of CCTV Video and Spatial Information for Automated Crowd Congestion Monitoring in Public Urban Spaces. *Algorithms* **2023**, *16*, 154. https://doi.org/10.3390/a16030154

Academic Editors: Guanqiu Qi and Frank Werner

Received: 16 January 2023
Revised: 6 March 2023
Accepted: 8 March 2023
Published: 10 March 2023

1. Introduction

With the rapid growth of urban populations, crowding in public venues, such as sports stadiums and train stations, has become increasingly problematic. Understanding the distribution of human crowds in a built environment is essential for timely and appropriate infrastructure management and service delivery. For example, uncontrolled crowding at exits causes sluggish evacuation during disasters; overly-dense crowds lead to stampedes [1]. For this reason, real-time monitoring and early warning of dense crowd flow are crucial for a variety of public safety-related applications, including public event management, safety monitoring, and disaster evacuation management. For instance, effective congestion detection and prediction can trigger timely support services such as the dispatching of human guides to direct crowd flow, the establishment of queue lines, and the planning of additional spaces and services for occupants.

Closed-circuit television (CCTV) surveillance cameras are frequently put in public areas to understand crowd behaviors, detect anomalous events, and identify congestions for security and safety purposes [2]. By evaluating camera footage and using their own judgments, stakeholders such as security professionals, infrastructure operators, and crowd managers are able to monitor crowds and identify potential hazards. However, manual observation of videos is labor-intensive and error-prone, particularly when crowds are

dense [3]. Therefore, it is essential to increase the role of technology in crowd management. Through interviews with crowd managers and assessment of the current state of practice, research has shown that crowd monitoring and event planning are sophisticated, but operate with minimal technology support at present [4]. In addition, crowd managers prefer to increase their use of technology and seek improved tools to assist them in their work.

In recent years, leveraging computer vision algorithms and greater computational capacity, research has continued to expand towards automating the process of vision-based crowd congestion analysis. Earlier research with pixel and color-based methods were often used for people recognition, tracking, and crowd counting until the more recent rise of convolutional neural network (CNN) based approaches, which have been shown to be more robust under scenarios of varying lighting conditions, image resolution and sizes [5]. CNNs are particularly effective at image recognition and processing applications due to their convolutional kernels, which extract visual features while preserving the spatial relationships between pixels. Numerous CNN-based approaches, including YOLO [6], Faster R-CNN [7], and Mask R-CNN [8], have demonstrated remarkable detection abilities for a variety of objects, including humans. However, two research gaps remain between human detection models and their direct applicability to congestion monitoring. Firstly, dynamic crowd mobility information should be learned by fusing both spatial and temporal features. Existing studies rarely consider the spatial connectivity of the surrounding space, such as the location of the egresses, which is essential for congestion monitoring and management. Secondly, while there is much research applying computer vision to the detection, tracking and prediction of individual pedestrian movements, macroscopic crowd congestion statistics such as crowd count, density, and flow estimation have received less attention.

The main objective of this paper is to illustrate an application of computer vision and machine learning methods to enhance crowd safety in a public space. Using deep learning and computer vision techniques, this paper proposes to fuse features from CCTV footage and spatial information of the public space to establish a framework for monitoring crowd congestion in public urban spaces. The key contribution of this effort is threefold: (1) A modular framework is presented to extract crowd congestion information from raw surveillance footage using deep learning-enabled detection and tracking algorithms. (2) The crowd mobility graph (CMGraph) data structure is proposed to store dynamic, macroscopic crowd flow data. The CMGraph formulation is spatiotemporal: the graph topology contains information on the spatial connectivity of the egresses, and the node feature stores temporal crowd flow data. (3) Leveraging the generated spatiotemporal crowd data, we develop real-time congestion alerts and future-time prediction visualizations to assist with manual crowd congestion monitoring. Furthermore, to quantify the benefits of our proposed framework, we validate our implementation with a fully annotated publicly available dataset from New York's Grand Central Station, a busy public urban location [9]. In order to demonstrate the generalizability and capacity in providing qualitative analysis, we also collect an unannotated video from a stadium during a crowded football game. While these datasets have varying crowd density, locations, and public space use cases, the framework can potentially be extended to both videos and likely to other crowd congestion applications.

This paper is organized as follows: In Section 2, we review existing crowd-monitoring methods that leverage deep learning and computer vision technologies. In Section 3, we present the overall workflow framework for crowd congestion analysis and detail the three main components of the proposed framework: (1) Dataset Preparation, (2) Trajectory Generation, and (3) CMGraph for Efficient Congestion Monitoring. Section 4 reports qualitative and quantitative results obtained from experiments on real CCTV footage. Section 5 discusses conclusive remarks.

2. Related Work

Despite rapid growth in computer vision technologies, accurate monitoring of crowd flow in congested scenes remains a challenging task of deep learning. Different automated deep-learning solutions are constantly being developed based on computer vision techniques

applied to surveillance videos. In this section, we review crowd-monitoring methods with deep learning and computer vision technologies.

While there have been different approaches that monitor crowd movements and behaviors, vision-based monitoring with images and videos using CNN is the most widely used method. Among them, density estimation with CNN is a popular approach. Density estimation CNN models take an input image containing crowds and generate a density map that represents the distribution of people in the image. The first known CNN model for density estimation and counting is CrowdCNN [10], a model consisting of three convolution layers followed by three fully connected layers. Further down the road, multi-column CNNs that have filters of different sizes in the columns are developed to handle the variation of pedestrian head and body sizes in images [11–13]. Subsequently, single-column counting networks are proposed for learning size-relevant without relying on multi-column networks using modules containing multiple filters of different receptive field sizes [14,15]. The density map outputted is useful for identifying where crowds gather and can be integrated to obtain an estimate of the crowd count [2]. The density estimation approach is particularly effective when crowds are extremely dense, and occluded and individuals are small.

A limitation of density estimation, however, is that the approach can only compute the spatial distribution of the macroscopic crowd count, rather than the monitoring of individuals. An alternative approach is detection-based, where a CNN model first performs pedestrian detection, generating a set of bounding boxes around any individuals in an image. For instance, variants of a one-stage detector, You Only Look Once (YOLO) have been used for video-based crowd counting in specific regions to enforce social distancing [16,17]. Detection-based methods tend to perform well when the crowd is sparse. However, when the crowd becomes dense, occlusion and small person-size tend to reduce performance significantly [18]. On the other hand, since detection-based methods yield bounding boxes around individual pedestrians, microscopic monitoring of each individual is possible. In this work, we show that when detection methods are integrated with tracking and information on the surrounding space, spatiotemporal mapping of individual trajectories can be obtained. The spatiotemporal mapping can then be leveraged to provide useful visualization and even spatiotemporal forecasting to assist the process of crowd congestion monitoring.

3. Methodologies

The overall workflow of the framework is shown in Figure 1. The workflow is composed of three modules. In the Dataset Preparation module, raw CCTV footage is preprocessed in preparation for the integration of spatial information and subsequent extraction of crowd congestion data. The Trajectory Generation module then sequentially employs a pedestrian detector, tracker, and spatial mapper to generate a spatiotemporal mapping of each occupant. Lastly, the CMGraph for Efficient Congestion Monitoring module enables the representation of spatiotemporal characteristics as graph data, facilitating real-time monitoring and prediction of future crowd congestions.

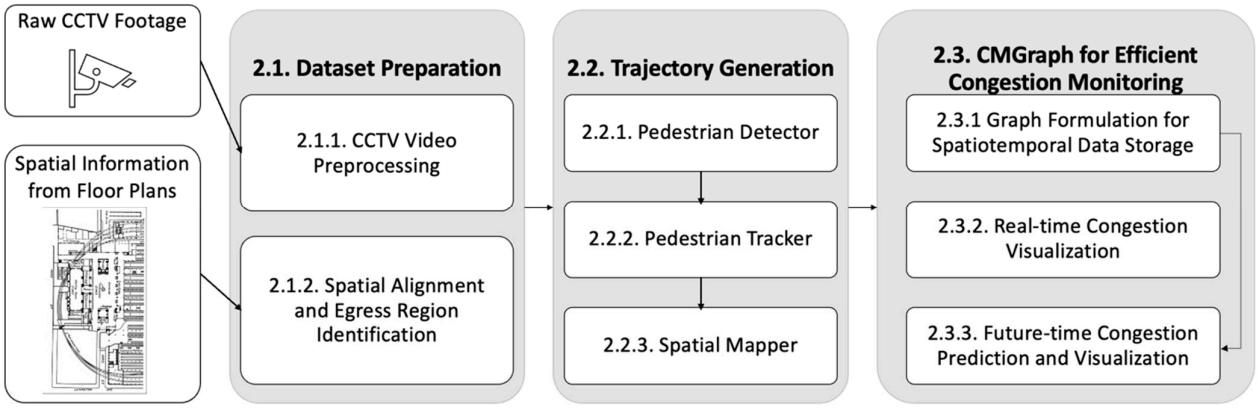

Figure 1. Workflow of the congestion monitoring framework.

3.1. Dataset Preparation

Prior to using raw CCTV footage for training and validating data for the extraction of crowd congestion statistics, several preparatory processes are required. This section describes the process of curating a dataset from unprocessed CCTV videos for the proposed framework.

(1) **Padding** CNN networks for human detection frequently extract a visual feature map from an input image. The feature map can be viewed as a downsampled tensor of the original input. For instance, models belonging to the YOLO family are designed such that input images are downsampled by a factor of 32 [6]. To ensure compatibility with most open-source detector networks, input videos are required to be padded with zero-value pixels such that their width and height are multiples of 32. In our case study of New York's Grand Central Station, the original video resolution is 1920×1080. Padding transforms the video to 1920×1088. In the other case study of a football stadium, the original video's resolution is 720×480 and padded to 736×480.

(2) **Optional frame rate reduction** We note that there is a trade-off between data efficiency and tracking precision, as videos with a high frame rate capture motion in shorter intervals, which can improve tracking performance. However, a high frame rate increases data storage and processing costs. Therefore, we advise optionally lowering the frame rate of the input video to adjust to storage and tracking precision needs, particularly when there is a constraint on video storage and computational capacity. In our case study of Grand Central Station, we demonstrate that even with 1.25 frames per second (FPS) and low resolution, our approach provides relatively good detection and tracking performance to sufficiently capture trends in crowd mobility.

(3) **Spatial alignment** Fusion of video and spatial information requires the alignment of the video's image-plane coordinate system with the physical world's coordinate system. This can be viewed as obtaining a projection from a plane in the image to a plane on the floor plan. Nevertheless, the coordinate systems of the two planes are often initially unknown. An important step of preparing a dataset is therefore to define the two planes. Therefore, we manually identify four key points in the surveillance video and on the floor plan, consistent in order and winding. An illustration of this step is shown in Figure 2, where in the Grand Central Station, the plane in the floor plan coordinate system is drawn on the left (Figure 2a), and the plane in the video frame's coordinate system is drawn on the right (Figure 2b).

(4) **Egress region identification** The floor plan provides locations of entrances and exits, which are key locations that are vital for crowd congestion analysis. In a crowded, urban space, people naturally flow from one entrance to another exit. To better model crowd flow, all floor plans are processed such that they are divided into egress regions based on the locations of the entrances and exits of the floor plan, thereby requiring manual attention to read the floor plans. For example, Figure 2a shows the nine egress region divisions based on the floor plan information of the Grand Central Station.

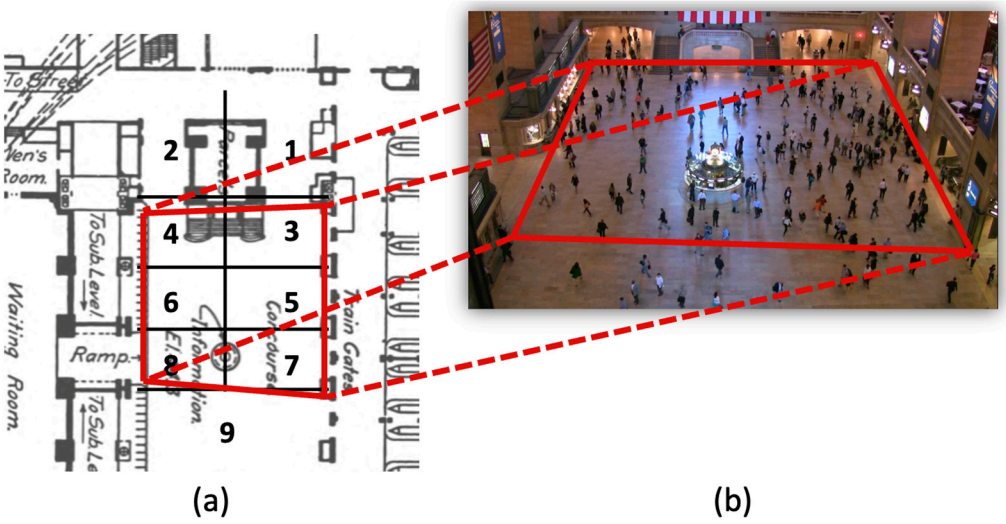

Figure 2. Illustration of the spatial alignment and egress region identification process. (**a**) The location of the egress regions overlaid on the Grand Central Station floorplan, adopted from [19]. (**b**) The video frame needs to be aligned spatially.

3.2. Trajectory Generation

This section outlines the method used to generate pedestrian trajectories from the fusion of video and spatial data. The trajectory generation component can be separated into three smaller modules: pedestrian detector, tracker, and spatial mapper.

3.2.1. Pedestrian Detector

Pedestrian detection entails generating a set of bounding boxes around any individuals in images and videos. In recent years, numerous applications in domains such as surveillance [20], autonomous vehicles [21], and behavioral analysis [22,23] have demanded advancements in pedestrian detectors, resulting in active research into the application of deep learning technologies in the field. The predominant approaches can be split into two types: one-stage and two-stage detectors.

One-stage detectors, such as You Only Look Once (YOLO) [6], RetinaNet [24], and Single Shot Detector (SSD) [25] use a single convolutional neural network (CNN) to directly predict bounding boxes and the associated confidence score for each pedestrian in an input image. One-stage detectors train faster and more efficiently than two-stage detectors, as the entire detection process is performed in a single pass of the CNN.

Other CNN architectures, such as the Regional Convolutional Neural Network (R-CNN) [26] Faster R-CNN [7], and Mask R-CNN [8], are two-stage detectors, which consist of two distinct steps. The first is the region proposal network (RPN), which generates a set of rectangular candidate regions likely to contain objects in the input image. These proposed regions aid the second deep neural network in concentrating on the most promising areas of the input image. The second network then conducts bounding box regression using the set of proposed regions. Despite the fact that two-stage detectors are slower and more computationally intensive than single-stage detectors, the use of region proposals typically enables two-stage detectors to achieve superior detection performance for images with fine details.

In this paper, both a one-stage and a two-stage detector are implemented to demonstrate that the proposed framework allows the user to interchange detector models flexibly. Although there are numerous candidate architectures for the detector module, the following criteria are used to select the most suitable for real-time pedestrian tracking: First, the work focuses on the accurate retrieval of each pedestrian's coordinates rather than pixel-level accuracy. Therefore, models with segmentation heads introduce unnecessary computational costs for the purpose of this framework; second, real-time inference speed along with acceptable accuracy are needed, prompting the use of lightweight models only. Both the YOLOv7

network proposed by Wang et al. [27] and the Faster-RCNN network proposed by Ren et al. [7] can serve as the basis for analysis in the pedestrian detector module.

Detailed discussions of the YOLOv7 and the Faster R-CNN algorithm can be found in [27] and [7], respectively. The following provides a description of the modules of the YOLOv7 and the Faster R-CNN model, as well as their use to generate detection bounding boxes on individual pedestrians during the inference stage.

You Only Look Once (YOLO) v7 The recently published YOLOv7 algorithm is faster and more accurate than all known object detectors (including two-stage detectors such as variants of the R-CNN) [27]. The YOLOv7 model is a single-stage detector with three components. First, the backbone of ELAN is used to extract feature maps of multiple resolutions. Second, the neck enhances the feature maps by performing multi-resolution aggregation of the features. Lastly, the head generates final predictions of detection bounding boxes, which are represented as a tuple of (x, y, w, h), where x and y represent the coordinates of the center of the bounding box that surrounds the detected human; w and h represent the width and height of the bounding box, respectively. Together, the tuple (x, y, w, h) describes the location and size of the detected object in the input image.

Faster R-CNN The Faster R-CNN proposed by Ren et al. [7] is a two-stage object detection algorithm. The overall architecture of Faster R-CNN can be summarized as follows: CNN layers are initially employed to extract image features. An RPN is then applied to the set of feature maps, which generates a set of rectangular region proposals. A Region of Interest (ROI) pooling layer then takes as input both the set of region proposals and the previously extracted set of feature maps and generates a feature map of fixed size for each region proposal. The proposal feature maps are then fed into a fully connected network that classifies the objects within each proposal and refines the bounding boxes. The use of a shared CNN extracted feature map for both stages is one of the key innovations of Faster R-CNN, which makes the algorithm significantly faster than previous two-stage object detection methods. Similar to YOLOv7, the output of the final fully-connected network contains detection bounding boxes, however in a different format, a tuple of (x_1, y_1, x_2, y_2), where (x_1, y_1) is the top left corner of the bounding box, and (x_2, y_2).

Existing research has shown that Faster R-CNN makes fewer localization errors but more background detection errors than YOLO [6]. In Section 4, we present a detailed experimental analysis with both detector algorithms, YOLOv7 and Faster R-CNN to assess their performance as well as to illustrate the flexibility of the modular framework for accommodating different models. Note that the models are only used for inference, as pre-trained models are publicly available for use [7,27]. As a result, no training hyperparameter tuning is needed. The pre-trained models are used as is with no modification to the originally provided configurations and weights.

3.2.2. Pedestrian Tracker

In the previous section, we described how a set of bounding boxes can be obtained using a single-stage or two-stage detector for every image frame. CCTV video can be viewed as a sequence of image frames over time. In this section, we describe how a tracker can be used to associate pedestrians with the additional temporal dimension. We note that both SORT and DeepSORT trackers are publicly available for use [28,29]. While further discussions of SORT and DeepSORT are available in these references, in this subsection, we describe the algorithms and the modifications that we made to deal with datasets from real-world scenarios that may have a low frame rate and lack annotations.

To obtain the precise trajectory of a single object across multiple video frames, it is necessary to have a tracker algorithm that associates each bounding box to an identity. The tracking stage can be formulated as an assignment problem: At any time t, the tracker must associate each bounding box in the set of all bounding boxes $\mathcal{B}$ generated by the pedestrian detector to a pedestrian with identity i, whose trajectory up to frame t is $\mathcal{T}_i^t$. The trajectory $\mathcal{T}_i^t$ is composed of a sequence of m_i bounding boxes $\{\mathcal{B}_i^1, \ldots, \mathcal{B}_i^{m_i}\}$. Thus, m_i denotes the total length of the trajectory $\mathcal{T}_i^t$. In this work, we modify and implement two

popular Kalman-filter-based tracking algorithms, SORT [28] and DeepSORT [29] to better accommodate our use cases, which presents new challenges for pedestrian tracking that are not present in modern high-quality public pedestrian tracking benchmarks.

SORT The SORT algorithm first employs a Kalman filter predictor to estimate the state of a pedestrian based on observations from previous frames using a linear constant velocity model to represent each pedestrian's motion, observed as a sequence of bounding boxes. Given a set of $|\mathcal{B}|$ bounding boxes from the detector as the input, and a set of $|\mathcal{B}_{KF}|$ predicted bounding boxes from the Kalman filter algorithm, SORT uses the Hungarian algorithm [30] to find the best assignment of bounding boxes in $\mathcal{B}$ (from the detector) to boxes in $\mathcal{B}_{KF}$ (predicted by the Kalman filter) that minimizes the total cost. In SORT, the cost is computed using the intersection over union (IoU) of assigning a bounding box $\mathcal{B}_i \in \mathcal{B}$ to a box $\mathcal{B}_{KF,\,j} \in \mathcal{B}_{KF}$, given by:

$$cost(i,j) = 1 - \text{IoU}\left(\mathcal{B}_i, \mathcal{B}_{KF,\,j}\right) = 1 - \frac{\mathcal{B}_i \cap \mathcal{B}_{KF,\,j}}{\mathcal{B}_i \cup \mathcal{B}_{KF,\,j}} \tag{1}$$

The resulting assignments are then used to update the existing trajectories and create new tracks for unmatched detections.

DeepSORT The DeepSORT algorithm, similar to SORT, uses the Kalman filter for state estimation and solves the assignment problem with the Hungarian algorithm. As an extension to SORT, DeepSORT incorporates a reidentification (Re-ID) model, a CNN feature extractor, to improve tracking by using a matching cascade for the detection association process. The feature extractor is pre-trained on a large-scale person Re-ID dataset and is available publicly [31]. The matching cascade computes both the Mahalanobis distance and similarity in deep appearance features between predicted boxes and detections for the cost of association. The deep appearance similarity between a bounding box $\mathcal{B}_i \in \mathcal{B}$ (from the detector) to a box $\mathcal{B}_{KF,\,j} \in \mathcal{B}_{KF}$ (predicted by the Kalman filter) is computed as the cosine distance between the deep features of the pedestrian enclosed by $\mathcal{B}_i$, and those of the pedestrian enclosed by $B_{KF,j}$. Trajectories and detections that are unable to be matched by the matching cascade process are then associated using their IoU metric as in SORT.

Existing benchmarks for pedestrian tracking datasets are typically created with the objective of accurate algorithm development in nearly ideal conditions using high-quality videos. The most notable dataset benchmarks are the MOTChallenge benchmarks, including MOT17 [32] and MOT20 [33], which create leaderboards that spark much academia and industry efforts to develop new tracking algorithms that score higher in a range of accuracy metrics. However, surveillance videos in the real world differ from these selected quality benchmarks. Empirically, we observe at least two significant difficulties, namely frame rates, and annotations, from real-world datasets, including our experimental ones, that are rarely seen in high-quality benchmarks such as the MOTChallenge. The following describes how the tracking algorithms are adjusted to deal with the real-world datasets that are used in this study.

(1) **Low frame rate**

High-quality data often comes with a higher frame rate. For example, MOT20 [33] contains videos of 25 FPS. On the other hand, the Grand Central Station videos are only annotated every 20 frames, rendering the effective frame rate of 1.25 FPS. This is especially challenging for the tracker, as bounding box association works better when there is more overlap between bounding boxes of the same pedestrian in consecutive frames.

Reduced performance at lower frame rates can be explained by the matching process used by SORT and DeepSORT. To cross-check detections and tracks, SORT first use IoUmatching followed by track decay. Before IoU matching, DeepSORT incorporates an additional stage known as the matching cascade. During the IoU matching stage, unassigned track and unassigned detection boxes with IoU higher than a certain threshold are matched using the Hungarian algorithm.

However, because pedestrians have traveled a large distance between successive frames when the frame rate is low, a smaller IoU between the detections and tracks is more

appropriate. We, therefore, lower the IoU threshold to 0.1 (instead of the conventional implementation of 0.7) in order to retain unmatched candidate tracks and detections that would be otherwise discarded due to the low frame rate. With a low IoU threshold, pedestrians can be tracked more effectively in videos with a low frame rate.

(2) **Lack of associated bounding box annotation**

Data annotation is expensive and time- and labor-intensive. The average time required to manually annotate a bounding box, according to research, is at least 88 s per bounding box [34]. For the tracking problem, additional time is necessary to assign each bounding box a unique person ID. Consequently, it is challenging to construct high-quality tracking annotations for surveillance videos featuring large crowds. Typically, high-quality benchmark datasets include these annotations so that detectors and trackers can be trained on the provided video data. The Grand Central Station dataset utilized in this research comprises just point annotations, whereas the stadium video is unannotated. We, therefore, rely on using detectors and trackers pre-trained on other datasets of completely different scenarios.

In addition, since there is no bounding box ground truth to compare the Grand Central Station dataset to during the evaluation phase, we employ the Euclidean similarity measure instead of the IoU similarity measure to compare each track to the point-wise ground truth. The Euclidean similarity thresholds the Euclidean distance between two points. For numerical experiments in this paper, the threshold is set to be a 1 m-distance in the physical space.

3.2.3. Spatial Mapper

Although the pedestrian detector and tracker modules permit the extraction of each pedestrian's position in each CCTV video frame, these positions are relative to the picture plane coordinates of the video. The pedestrian's position needs to map onto the physical space for performing scenario analysis of crowds. To extract the spatial and temporal position of each pedestrian relative to the physical space, it is necessary to find a projection from the image plane coordinates of the CCTV to the physical space's plane coordinates.

The process of mapping a pedestrian's position from the image plane to the physical space plane can be viewed as mapping a point from one plane to another, as shown in Figure 3. The original plane containing the point is in the $X - Y$ coordinate system, whereas we want to find its mapped position on a plane in the $X' - Y'$ coordinate system. We, therefore, introduce the spatial mapper module, which finds the homography matrix, or the transformation matrix between the two coordinate planes [35]. Formally, a homography transformation is defined as:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2}$$

where the homography matrix $H \in \mathbb{R}^{3\times3}$ has 8 degrees of freedom, generally normalized such that $h_{33} = 1$; The vector $\{x,\ y,\ 1\}$ represents a point in the image coordinate plane, whereas the vector $\{x',\ y',\ 1\}$ represents a point in the physical space coordinate plane.

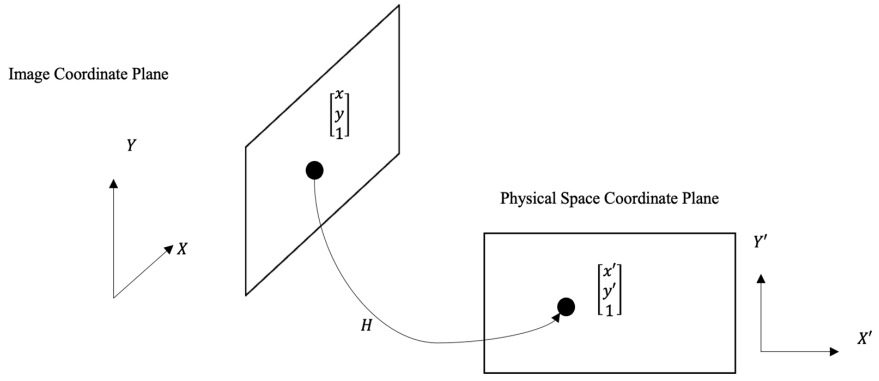

Figure 3. Spatial mapping from the image coordinate plane to the physical space coordinate plane.

To obtain the homography matrix H, the four points identified in the spatial alignment step (Section 3.1 step (3)) are used. The least square method proposed by Bazargani et al. [36] and implemented in OpenCV is adopted, where given n (x, y) points and corresponding (x', y') points, the following back-projection error term is minimized:

$$error = \sum_{i}^{n} \left(x'_i - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 + \left(y'_i - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 \quad (3)$$

The back-projection error term can be interpreted as the sum of the errors between each point $(x'_i,\ y'_i)$ on the physical coordinate plane and the corresponding point $(x_i,\ y_i)$ mapped from the image coordinate plane via homography transformation. With the homography matrix H derived, the video frame position of any pedestrian generated from the pedestrian tracker can be mapped to the physical space with respect to the floor plan using simple matrix multiplication.

3.3. CMGraph for Efficient Congestion Monitoring

The information on crowd mobility acquired by fusing CCTV and floor plan information encompasses both spatial data of the physical space and temporal data describing the evolution of crowd movements over time. In order to analyze crowd flows in a physical space, however, there is a need for tools and techniques to visualize and interpret the spatiotemporal data. The spatiotemporal data must be adequately represented, as a simple representation in Euclidean space is insufficient for representing spatial connectivity. In this section, we propose a new spatiotemporal graph formulation, Crowd Mobility Graph (CMGraph), which can be used to represent the macroscopic crowd flow between egress regions. We also illustrate how this new representation can be leveraged to achieve efficient real-time and future time congestion monitoring.

3.3.1. Graph Representation for Spatiotemporal Data

In this section, we describe the modeling of crowd mobility data as a sequence of crowd mobility graphs (CMGraphs). Crowd volume data recorded over a discrete time span $t = 1, \ldots, T_{obs}$ are formulated as a set of undirected and unweighted dynamic graphs $\{G(t) = (V(t),\ E)\}$. Here, $V(t) = \{v_1, \ldots, v_N\}$ denotes the set of N nodes where each node corresponds to an egress region. The node feature matrix, $X^{N \times D}$ stores the collective crowd flow data, where D is the number of node features. E denotes the set of time-invariant edges, where an edge $e_{jk} \in \{0, 1\}$ connecting node v_j and node v_k is 1 if two egress regions are adjacent to each other and 0 otherwise. Two egress regions are adjacent if a pedestrian is able to walk from one region to another without entering a third egress region. Figure 4 shows an example of how the egress regions of the Grand Central Station in New York are modeled as the CMGraph.

CMGraph allows our framework to integrate space-time information into a graphical data structure. Since edges connect egress regions in the vicinity, the topological structure of the CMGraph naturally models the flow of pedestrians from one region to another. For example, a graph neural network (GNN) model can pass node information along the edges of a CMGraph such that the crowd flow information in an egress region relates to the flow of nearby regions. This allows us to account for the effect of the physical space on crowd dynamics. On the other hand, the temporal data stored in the nodes $V(t)$ allow us to use sequence-generating neural networks, such as recurrent neural networks (RNN) to perform time series forecasting.

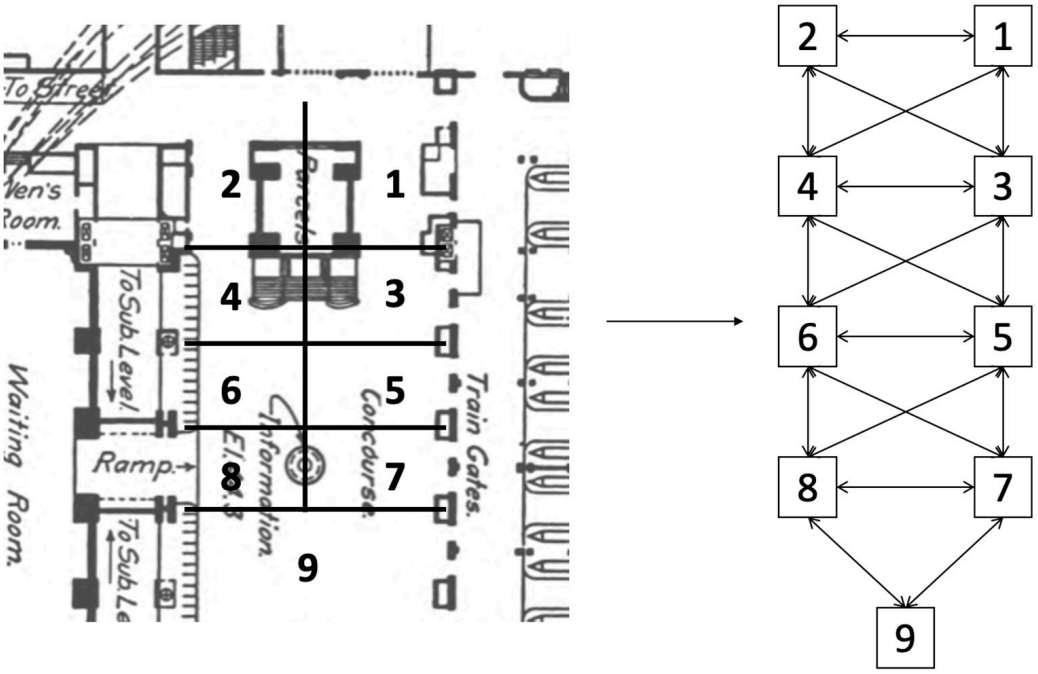

Figure 4. Egress region division and CMGraph formulation of the Grand Central Station. Nine egress regions were manually identified. Grand Central Station floorplan adopted from [19].

3.3.2. Real-Time Congestion Visualization

While the preceding sections describe the methodology for acquiring data on crowd mobility, we provide two data visualization interfaces so that various stakeholders, including security professionals, infrastructure operators, and crowd managers, can easily interpret the data generated by our framework for congestion monitoring.

(1) **Crowd count and individual monitoring** The objective of the first data visualization tool is to reduce the amount of human effort required for surveillance video observation. We add a crowd count and assign a unique identifier to each pedestrian so that surveillance operators may simply determine the general crowd size and identify individuals who may require particular attention. This is especially beneficial in crowded areas when it is difficult to quantify the number of people and to locate an individual. Figure 5a depicts the visualization that was implemented.

(2) **Real-time congestion alert** The second tool is designed to notify the operator of probable congestion hotspots. To do this, a density heatmap is constructed to indicate busy areas from a bird's-eye view of the physical space. The color automatically alters dependent on a region's congestion level. Figure 5b illustrates a congested egress in the Stanford Stadium. In order to be alerted about overcrowding and to prevent hazards such as stampedes from occurring, operators can continuously use the heatmap to monitor crowded areas.

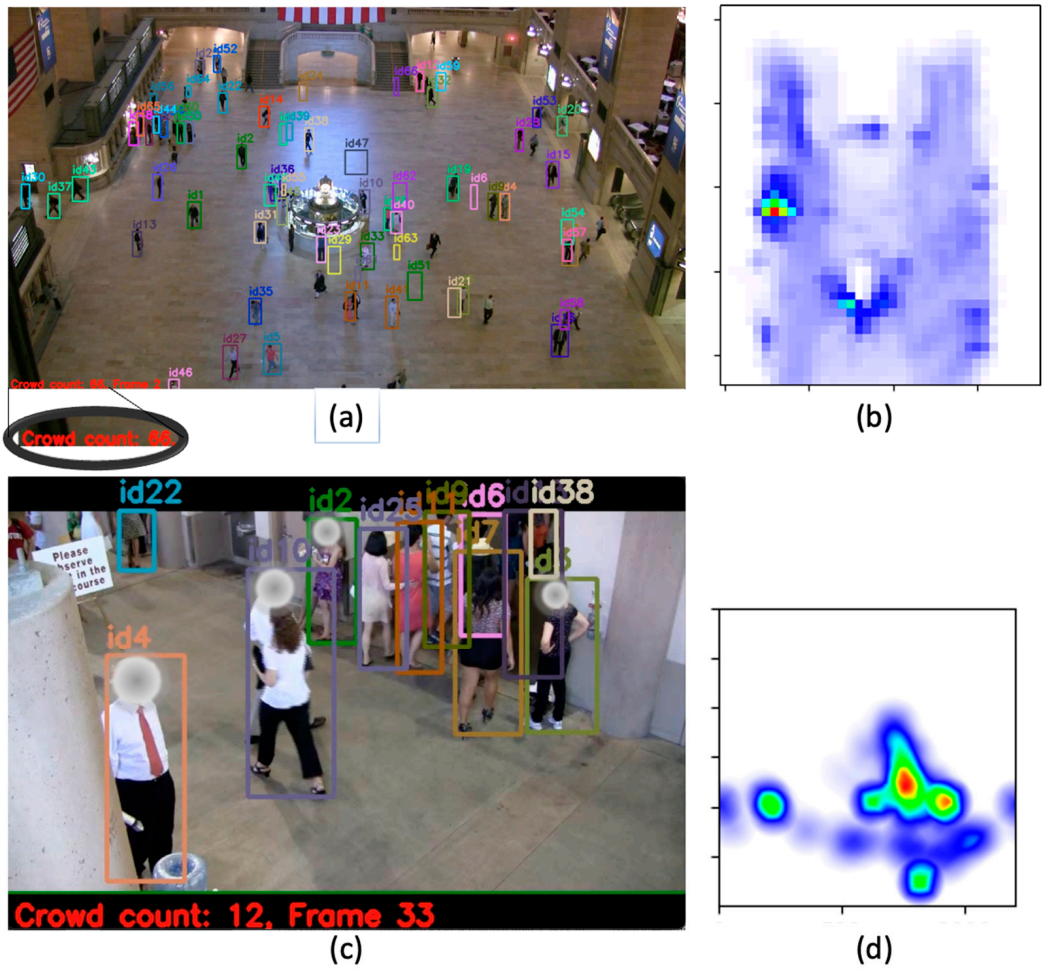

Figure 5. (**a**) Visualization including crowd count and individual monitoring at the Grand Central Station. (**b**) Real-time congestion visualization with highlighted congestion hotspots at the Grand Central Station. The density map colors range from blue to red with red being the highest density. (**c**) Visualization including crowd count and individual monitoring at the Stanford Stadium. (**d**) Real-time congestion visualization with highlighted congestion hotspots at the Stanford Stadium.

3.3.3. Future-Time Congestion Prediction

The ability to get notifications in advance about potential congestion problems is a crucial aspect of crowd monitoring. To accomplish this, it is essential to predict crowd movements to identify probable congested areas. Crowd movements feature complex spatial, temporal, and social dependencies that can affect collective crowd dynamics. Essentially, the movement of several crowd groups can be viewed as several spatiotemporal signals that are non-IID, making the forecasting of such signals challenging. In this section, we present an approach that uses graph convolutional network (GCN) and recurrent neural network (RNN), each leveraging spatial and temporal signals respectively, to obtain crowd flow predictions from CMGraphs to be used for advanced congestion alerts.

In many domains, numerous existing studies also deal with spatiotemporal data. Many such spatiotemporal problems are frequently solved with a combination of graph neural network (GNN) to learn the spatial features and recurrent neural network (RNN) to learn the temporal dependencies. Various combinations of GNN and RNN models have been used for spatiotemporal tasks such as pandemic forecasting [37], and more closely related to this problem, highway traffic forecasting [38–40]. However, cars commuting on highways tend to behave in a more routine manner (for instance, rush hours usually lead to heavier car flow), whereas pedestrian crowds in the public urban space have more stochastic and complex patterns, making the forecasting task even more difficult.

Motivated by the challenges that lie within crowd flow forecasting and the lack of deep learning-based spatiotemporal crowd flow forecasting methods, we propose a novel deep learning model, GCN-GRU that uses a combination graph convolutional network (GCN) [41] and gated recurrent unit (GRU) [42] to learn from CMGraphs and perform the crowd forecasting task. The GCN operations can be thought of as learning an embedding from spatial features resulting from the topology of the CMGraphs. The embedded graphs are used as the input to the GRU cells, which learn temporal representations from the time series data. We combine these two operations to process the spatiotemporal features resulting from complex crowd data. The architecture of the model is schematically shown in Figure 6. In the following, we describe the mathematical formulation of the problem and present the spatial and temporal operations that forecast the crowd flows with the graph neural networks.

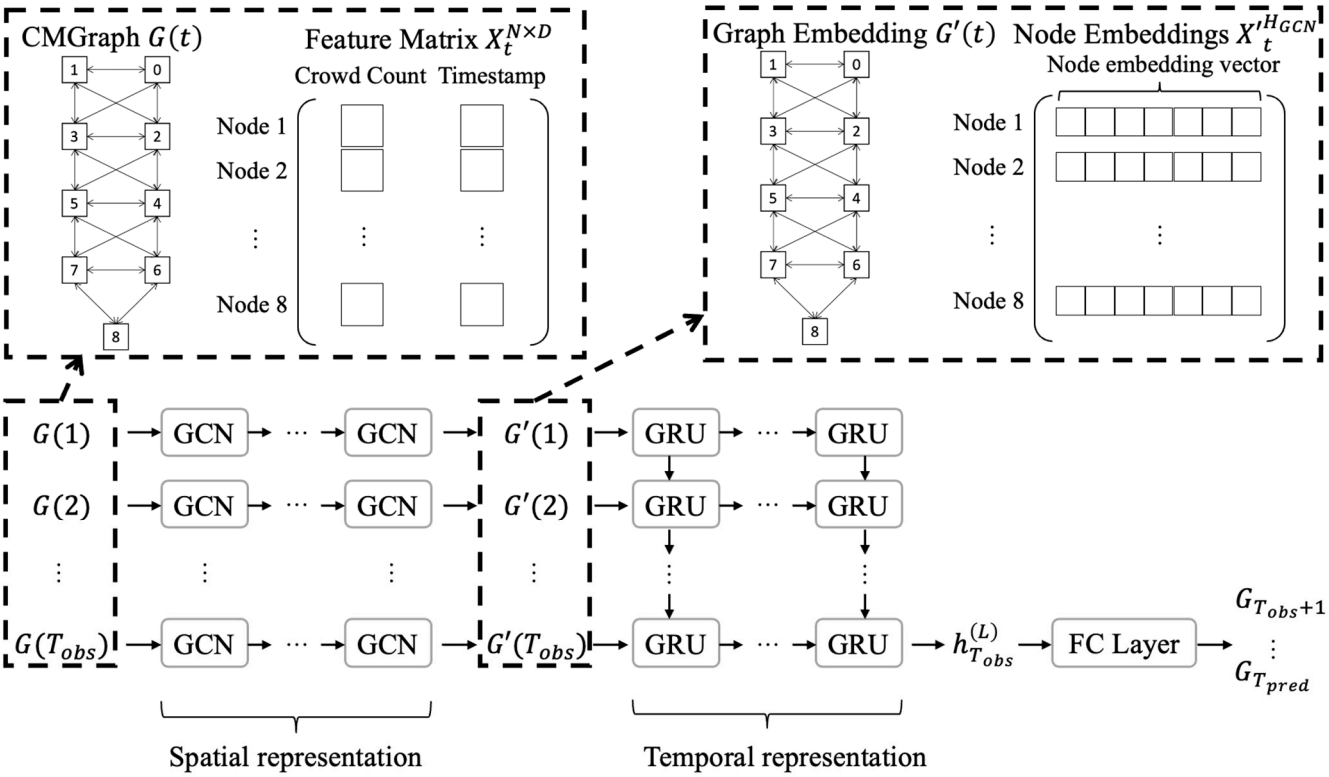

Figure 6. The architecture of the proposed GCN-GRU model.

Problem Definition

The crowd flow prediction problem can be defined as follows: Given the crowd flow information during an observed discrete time horizon 1 to T_{obs}, the goal of crowd flow forecasting is to predict the crowd flow information during a future time horizon T_{obs+1} to T_{pred}. The problem can therefore be written as the sequence generation task of learning a function that maps historical crowd flow data in a sequence of CMGraphs $G_1, G_2, \ldots, G_{T_{obs}}$ to another sequence $G_{T_{obs+1}}, G_{T_{obs+2}}, \ldots, G_{T_{pred}}$. Leveraging the CMGraphs, the spatial and temporal information of crowd flow are learned using GCN and GRU, respectively.

Spatial Representation Learning with GCN

For a set of N nodes in a CMGraph $G(t) = (V(t),\ E)$, a GCN layer updates the nodal information using a node's neighboring nodal information for all nodes. More formally, given a node v_i, whose node embedding vector is x_i (the i^{th} row of the feature matrix $X(t)$), and its set of neighboring nodes J, a GCN layer updates the node embedding as follows:

$$x_i^{(k)} = \frac{1}{|\,J\,|}\sum_{j\in J} x_j^{(k-1)} \tag{4}$$

where $W^{(k)}$ and $x_i^{(k)}$ are a learnable parameter and the i^{th} node's updated embedding of the k^{th} layer, respectively. In the first layer (i.e., $k = 1$), $x_i^{(0)}$ is the initial feature vector of the node $v_i \; \forall \; v_i \in V(t)$.

Stacking K GCN layers allow us to update node embeddings using information aggregated from nodes in the K-hop neighborhood. After K GCN layers, we have learned the embedded graph, $G'(t)$, whose node embedding matrix is $X'(t)$, each row being the updated embedding vectors $x_i' \; \forall i \in V(t)$. Each node embedding vector is of an embedding dimension H_{GCN}, a tunable hyperparameter. The dimension of $X'(t)$ is therefore $N \times H_{GCN}$.

Temporal Representation Learning with GRU

While GCN uses spatial information and aggregates information on the relative locations of pedestrians at the same time instant, it does not deal with information in the temporal dimension. On the other hand, GRU can be used to compute hidden state representations of time series data. Mathematically, each GRU operation in a layer l can be expressed as follows:

$$r_t^{(l)} = \sigma\left(W_{ar} a_t^{(l)} + b_{ar} + W_{hr} h_{t-1}^{(l)} + b_{hr}\right) \tag{5}$$

$$z_t^{(l)} = \sigma\left(W_{az}^{(l)} a_t^{(l)} + b_{az} + W_{hz}^{(l)} h_{t-1}^{(l)} + b_{hz}^{(l)}\right) \tag{6}$$

$$n_t^{(l)} = \tanh\left(W_{an}^{(l)} a_t^{(l)} + b_{an}^{(l)} + r_t * \left(W_{hn}^{(l)} h_{t-1}^{(l)} + b_{hn}^{(l)}\right)\right) \tag{7}$$

$$h_t^{(l)} = \left(1 - z_t^{(l)}\right) * n_t^{(l)} + z_t^{(l)} * h_{t-1}^{(l)} \tag{8}$$

where $W_{ar}^{(l)}$, $W_{hr}^{(l)}$, $W_{az}^{(l)}$, $W_{hz}^{(l)}$, $W_{an}^{(l)}$, $W_{hn}^{(l)}$, $b_{ar}^{(l)}$, $b_{hr}^{(l)}$, $b_{az}^{(l)}$, $b_{hz}^{(l)}$, $b_{an}^{(l)}$, $b_{hn}^{(l)}$ are learnable parameters of the l^{th} layer. $a_t^{(l)}$ and $h_t^{(l)}$ are the input and output hidden states of the l^{th} layer at time t, respectively. At $l = 1$, $a_t^{(l)}$ is the output from the GCN part, $X'(t)$. At $l > 1$, $a_t^{(l)}$ is the hidden state from the previous layer $h_t^{(l-1)}$. Moreover, $h_{t-1}^{(l)}$ is the hidden state of the l^{th} layer at time $t - 1$, except initialized to 0 when $t = 1$. σ is the sigmoid function. $r_t^{(l)}$, $z_t^{(l)}$, $n_t^{(l)}$ are the reset, update, and new gates of the l^{th} layer, respectively. $*$ denotes element-wise multiplication. The final output at time $t = T_{obs}$ after L GRU layers are then $h_{T_{obs}}^{(L)}$, a vector with a length H_{GRU}, a tunable hyperparameter. To compute a sequence of predictions from $T_{obs+1}, \ldots, T_{pred}$, the output is passed through a fully connected (FC) layer to obtain a vector of the desired sequence length.

4. Experiments and Results

This section discusses the experiments conducted to evaluate the performance of the modules, namely (1) Trajectory Generation, (2) Congestion Prediction, and (3) Congestion Visualization, implemented in the crowd monitoring framework. In this section, we describe the datasets used in this study the evaluation metrics employed to measure the performance of the modules, the implementation details, and the results of the experiments.

4.1. Quantitative Performance Evaluation of the Framework

To quantify the performance of the modular framework, the three components, namely the trajectory generation, congestion prediction, and congestion visualization were experimented on with the New York Grand Central Station (GCS) dataset, collected by Zhou et al. [9]. Point-wise individual trajectories were manually annotated by Yi et al. [43]. A detailed description of the dataset can be found in the cited references. The dataset consists of 17,682 trajectories, with 6000 video frames annotated at 1.25 FPS.

4.1.1. Trajectory Generation

Evaluation Metrics

To assess the performance of the detectors and trackers, we employ the CLEAR multiple object tracking (MOT) metrics, a commonly used collection of metrics for evaluating tracking algorithms [44]. Among the list of CLEAR metrics, we report several key metrics that can be measured for a dataset with only point-wise trajectory annotation (rather than bounding box annotations). Specifically, multiple object detection accuracy (MODA) is used as the primary metric to evaluate the detector, and multiple object tracking accuracy (MOTA) as the primary metric to evaluate the tracker. Additionally, to provide more detailed information about where errors occur (whether errors are due to missed tracks or inaccurate tracks), recall and precision on the detections are also reported following common MOT challenge practices in crowded scenes [33]. The four metrics are computed as follows:

$$\text{MODA} = 1 - \frac{\sum_t (\text{FN}_t + \text{FP}_t)}{\sum_t \text{GT}_t} \tag{9}$$

$$\text{MOTA} = 1 - \frac{\sum_t (\text{FN}_t + \text{FP}_t + \text{IDSW}_t)}{\sum_t \text{GT}_t} \tag{10}$$

$$\text{Recall} = \frac{\text{TP}_t}{\text{TP}_t + \text{FN}_t} \tag{11}$$

$$\text{Precision} = \frac{\text{TP}_t}{\text{TP}_t + \text{FP}_t} \tag{12}$$

where FN_t, FP_t, TP_t, GT_t and IDSW_t are the number of false negatives, false positives, true positives, ground truths, and of identity mismatches at time t, respectively.

Implementation Details

The YOLOv7 network and the Faster-RCNN network are compared as the pedestrian detector module, followed by the SORT algorithm as the tracker module. For faster inference speed, we employ the YOLOv7-tiny configuration, which slightly affects performance but can conduct inference in real time due to smaller parameter sizes. Both detectors are pre-trained using the ImageNet dataset [45].

The experiments were conducted on a computer equipped with an Intel Core i7-7820X processor and an NVIDIA GeForce GTX 1080 Ti graphics processing unit (GPU). The code repository is publicly available at [46].

Experimental Results

The results are detailed in Table 1. The YOLOv7-tiny network outperforms the Faster-RCNN in MOTA, as seen in Table 1. The recall and precision are also increased. Nonetheless, both YOLO and Faster-RCNN detectors have a high precision and a low recall, indicating a high number of false negatives (missed targets), FN. A high FN indicates the presence of a large number of untracked pedestrians. This is possibly owing to the fact that the GCS video has a low frame rate, and as a result, pedestrians move at a very quick speed, frequently without overlap between successive bounding boxes.

Table 1. CLEAR MOT metrics of YOLOv7-tiny and Faster-RCNN detectors with the SORT tracker.

Detector	Tracker	MOTA	MODA	Recall	Precision
YOLOv7-tiny	SORT	**62.8**	**70.8**	**72.2**	**98.2**
Faster R-CNN	SORT	58.8	66.5	71.0	94.0

The DeepSORT algorithm with the Re-ID model is employed as the pedestrian tracker to improve the outcomes. As shown in Table 2, we notice a significant improvement in MOTA, MODA, and recall when the detector is either YOLOv7-tiny or Faster-RCNN, albeit at the cost of a slight reduction in precision. This is perhaps due to the fact that incorporating the Re-ID model allows bounding boxes to be associated with the same

person even when there is no overlap, as the pedestrian would have similar visual features between frames. The Re-ID model, extracting deep appearance features, allows tracks to be more accurately associated by considering the cosine distance between detection and predicted bounding boxes during the matching cascade process. This could potentially be beneficial for scenarios such as the GCS, where occlusions and low frame rate both could be challenging for solely IoU-based matching. In summary, employing YOLOv7-tiny and DeepSORT as the detector and tracker, respectively, produces the greatest quality trajectories among all tested combinations empirically on the GCS dataset. Additionally, this experimental study demonstrates the modularity of the framework that allows rapid testing and evaluation of the two models.

Table 2. CLEAR MOT metrics of YOLOv7-tiny and Faster-RCNN detectors with the DeepSORT tracker.

Detector	Tracker	MOTA	MODA	Recall	Precision
YOLOv7-tiny	DeepSORT	**73.7**	**81.2**	**86.2**	**94.6**
Faster R-CNN	DeepSORT	64.2	71.3	82.2	88.2

4.1.2. Congestion Prediction

Evaluation Metrics

To measure the accuracy of congestion prediction, the mean squared error (MSE) between the node feature matrix of the predicted graph sequence and of the true sequence is used as an evaluation metric of prediction accuracy. The MSE loss measures the difference between the predicted node feature matrices $\hat{X}(T_{obs}+1), \ldots, \hat{X}\left(T_{pred}\right)$ and the true node feature matrices $X(T_{obs}+1), \ldots, X\left(T_{pred}\right)$. Denoting each element of a matrix $X(t)$ as x_{it} and $\hat{X}(t)$ as $\hat{x}_{it}$, the MSE is computed as

$$\text{MSE} = \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T_{pred}}(x_{it}-\hat{x}_{it})^2 \tag{13}$$

MSE, with the squared error, places more penalization on larger errors, and could therefore be more susceptible to outliers. As a result, another metric reported to evaluate model performance is the mean absolute error (MAE), which measures the average of magnitude difference between the prediction and the true node feature matrices:

$$\text{MAE} = \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T_{pred}}|x_{it}-\hat{x}_{it}| \tag{14}$$

Implementation Details

Unlike the trajectory generation experiments in the preceding section, the experimented models are trained from scratch (rather than pre-trained on a different dataset). The GCN-GRU model uses a 3-layer GCN to learn the spatial representations, and a 2-layer GRU to learn the temporal representations. The node features chosen are the aggregated crowd count and timestamp for each egress region, rendering a node feature length of $D = 2$. The embedding dimension of the GCN encoder is $H_{GCN} = 128$. The embedding dimension of the GRU is $H_{GRU} = 64$. We use $T_{obs} = 20$ and $T_{pred} = 20$ in the study and split the CMGraph sequences from the GCS dataset into train and test sets following a 70/30 ratio. The graph data is batched into mini-batches of size 32 for training. The Adam optimizer with a learning rate of 0.001 is used to train the GCN-GRU model as well as each baseline model for at most 40 epochs. The loss function used is MSE loss, as detailed in Equation (13).

The training and inference were conducted on the same computer, equipped with an Intel Core i7-7820X processor and an NVIDIA GeForce GTX 1080 Ti GPU. The code repository is publicly available at [46].

Experimental Results

We compare our experimental results with two baseline models: T-GCN [40] and A3T-GCN [38], which are GCN-based models previously only applied to highway vehicle flow prediction. These two baseline models are selected based on two criteria: (1) The models are among state-of-the-art graph-based traffic forecasting models. In particular, A3T-GCN is the best-performing model with the task of road taxi traffic speed [47], and (2) The models are open-sourced and therefore can be tested on the dataset of this experiment.

The results are tabulated in Table 3. It can be observed from the table that the GCN-GRU model presented in this paper outperforms the baseline models in both MSE and MAE scores, indicating that the GCN-GRU model can well learn the crowd dynamics and forecast crowd flows.

Table 3. MSE and MAE of crowd volume prediction models.

Predictor	MSE	MAE
Baseline 1 (T-GCN)	0.329	0.417
Baseline 2 (A3T-GCN)	0.320	0.403
GCN-GRU	**0.258**	**0.354**

Additionally, a sequence of predicted crowd flow is plotted in Figure 7, where it is shown that the baseline models tend to over-smoothen predictions, which may be the underlying reason why GCN-GRU outperforms in forecasting accuracy. In a place such as a train station, where there are often services such as direction guides and ticket counters needed, being able to predict crowd flows in advance will give facility operators additional time to plan for the dispatching of additional service support and potentially help reduce future congestion.

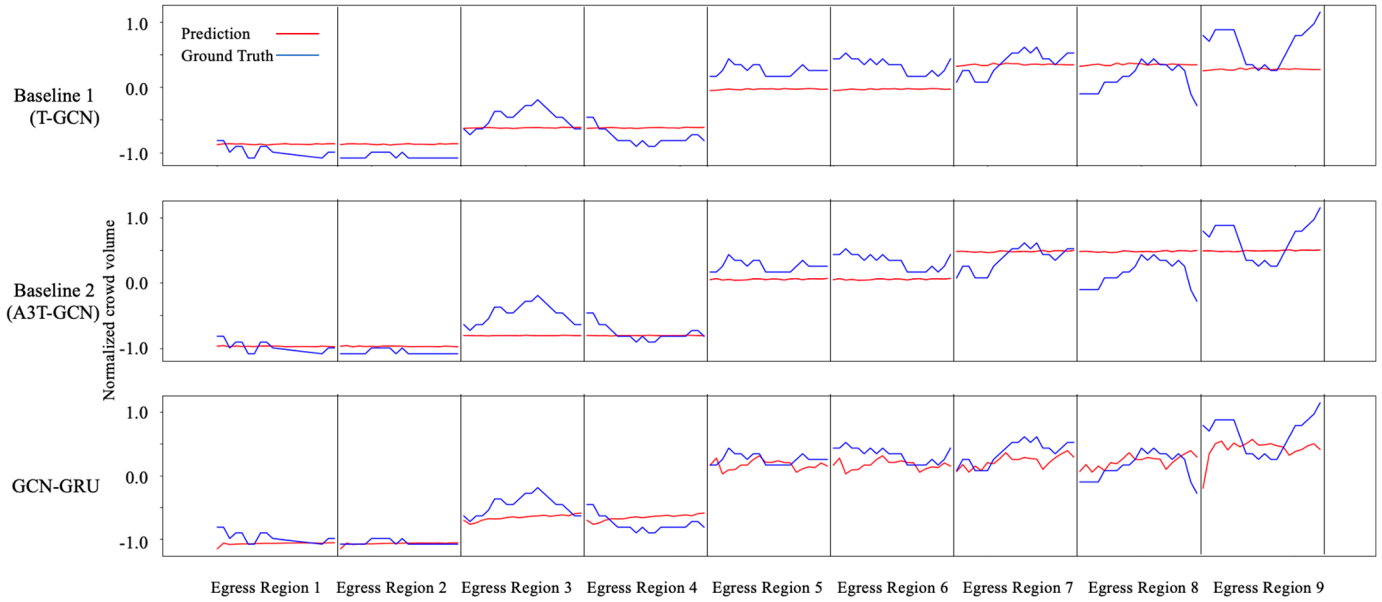

Figure 7. A sample sequence of the ground truth and predicted crowd flows at each of the nine egress regions at the Grand Central Station. The crowd volume, normalized to $[-1, 1]$ is plotted on the Y-axis.

4.1.3. Congestion Visualization

To visualize the density of pedestrian foot traffic over time, the Grand Central Station crowd flow is shown as density heatmaps in 15-min (1125 frames at 1.25 FPS) intervals in Figure 8. Observing Figure 8, we identify a number of significant qualitative observations on the locations where crowds tend to gather based on density heatmaps. In the first 45 min of the video, there appear to be fewer people in the train station, however, in the second half, a greater number of pedestrians can be seen. Regardless of the number of passengers

in the station, the heatmaps depict a large crowd forming around the ticket booth at almost all times. This indicates the necessity for faster ticket services. In addition, as the crowds become heavier, we observe more people congregating around the information center, possibly seeking assistance or direction guides. Overall, we observe that the train station requires additional services, especially as the number of passengers increases.

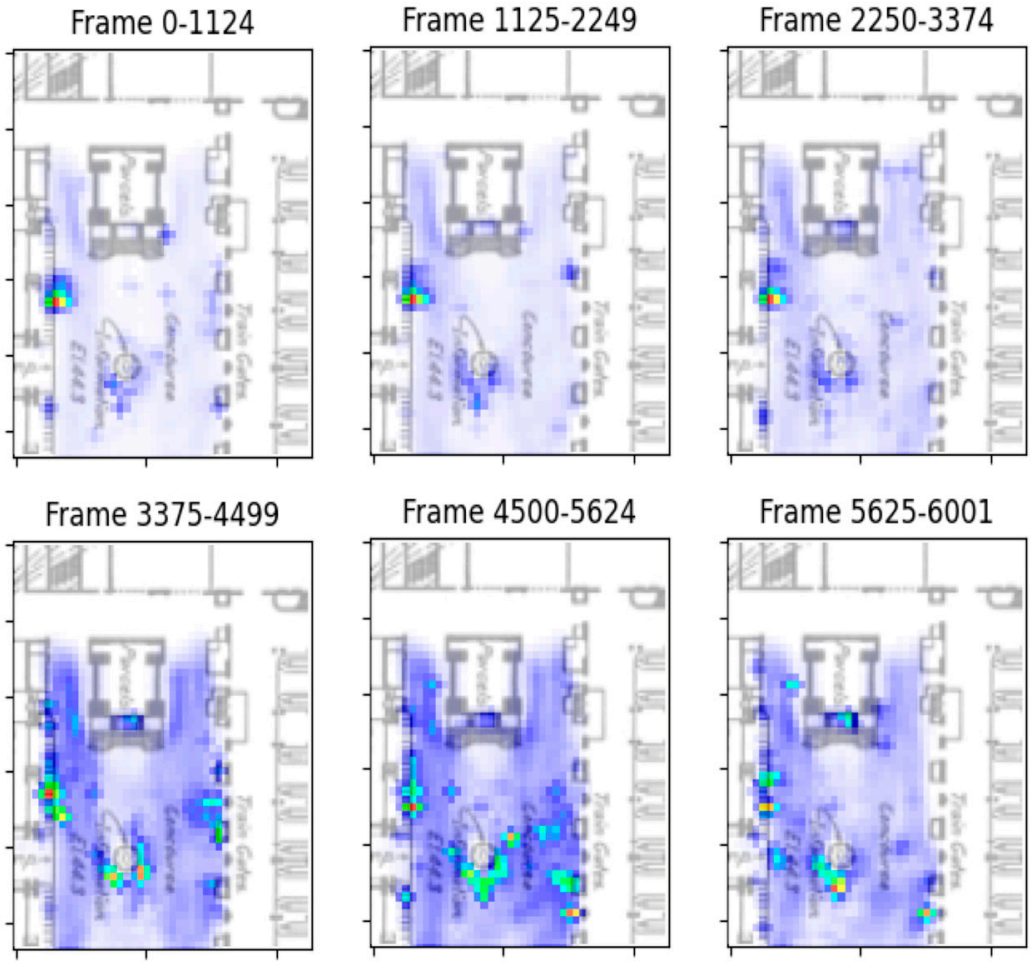

Figure 8. Heatmap visualization of crowd flow density over time in 15-min (1125–frame at 1.25 FPS) intervals. The density map colors range from blue to red with red being the highest density.

4.2. Qualitative Performance Evaluation of the Framework

As an illustration of the potential application of the suggested framework, we employ a video recorded during a football game at one of the Stanford Stadium exits. The stadium is well-known as the venue for numerous significant public events on the Stanford campus, including a variety of sporting events and annual commencement ceremonies. These public activities are certain to attract a large number of pedestrians. Therefore, we run the recorded video through our framework to extract pedestrian paths and data visualization, in order to provide potential recommendations for facility managers on how to better accommodate the congested traffic.

The video is recorded for 30 s at 29 FPS at a high-traffic intersection within the stadium. A top-down view of the recorded location is provided in Figure 9, with arrows indicating possible pedestrian flow in and out of the video frame. The location recorded is a T-shaped intersection with two bathrooms, located next to the exit. To generate human trajectories, we use the YOLOv7 detector and DeepSORT tracker, with the same hyperparameters as the Grand Central Station experiment.

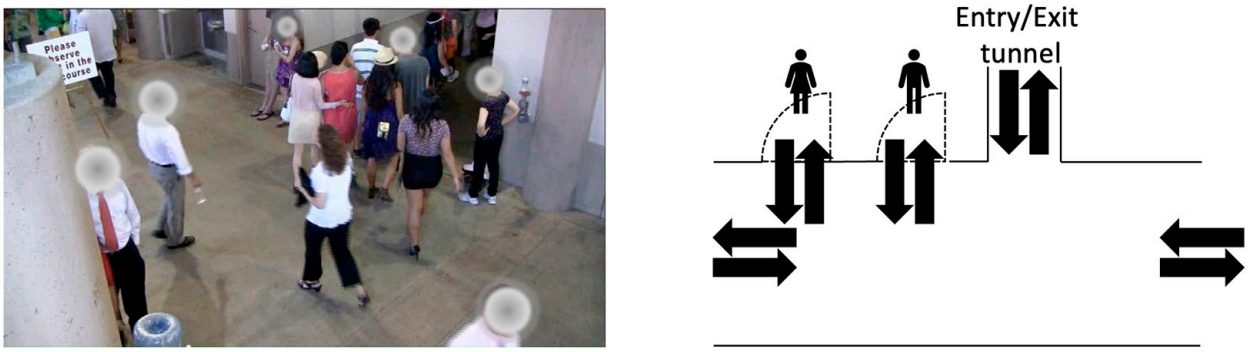

Figure 9. Top-view floor plan of the recorded location at the Stanford Stadium. Pedestrians are crowded at the entry/exit tunnel as they attempt to leave the stadium.

Figure 10 illustrates, using the real-time visualization tool, heatmaps depicting the locations of occupants in 5-s (145-frame) periods. We observe multiple areas at various times where pedestrian congestion occurs. Initially, we witness numerous passengers attempting to depart the entry/exit tunnel, resulting in congestion that took around 15 s to dissipate. This shows that residents tend to desire to evacuate the building simultaneously (for example, at the conclusion of public events). At these critical times, measures should be implemented to guide a large number of people out of the building more efficiently. Additionally, the entry to the women's restroom is crowded, with a line forming outside the bathroom door. In the planning of public areas, it is crucial to ensure that there are sufficient restrooms to prevent congestion and discomfort among the population. Particularly, the need for additional women's restroom facilities has long been observed: a study by Gwynne et al. [48] observed bathroom dwell times at the airport and show that female facilities have much longer dwell times than male facilities. Lastly, throughout the duration of the video, loitering behavior was observed near the corner of the entry/exit tunnel, where numerous individuals waited. This result indicates that planning for loitering near event exits, particularly at narrow tunnel exits, should be considered by event planners.

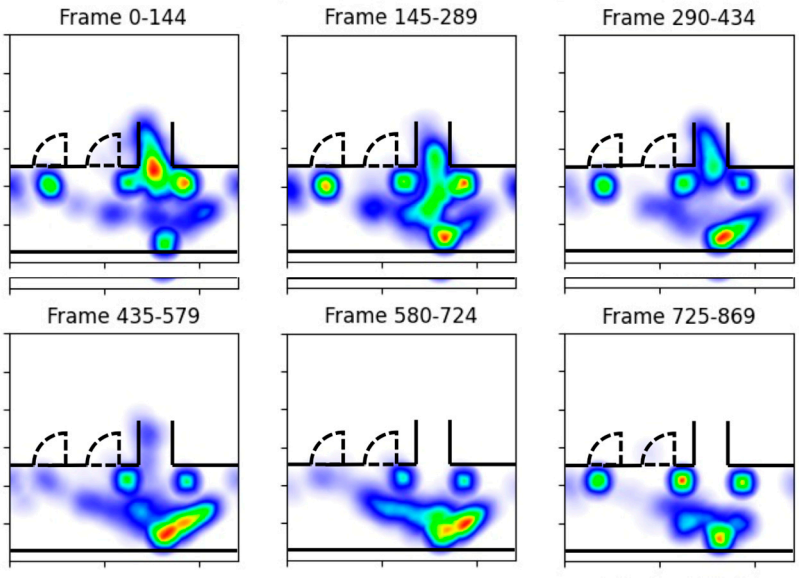

Figure 10. Heatmap visualization of crowd flow density over time in 5-s (145–frame at 29 FPS) intervals. The density map colors range from blue to red with red being the highest density.

In light of the aforementioned observations at the train station and the stadium, we have shown that the visualization and forecasting tools as discussed herein can be useful for planners and crowd managers to closely monitor congestion, as well as to analyze and improve the planning of spaces, facilities, and the distribution of occupant services. Congestion

prediction, as shown in Figure 7, is beneficial for advanced alerts of future densely crowded egress regions. Furthermore, the density heatmaps, as shown in Figures 8 and 10, are real-time congestion visualization tools that can help gain a birds-eye view of the spatial distribution of crowds, with red colors highlighting where congestions occur.

5. Conclusions

The overarching objective of this study is to develop a modular software framework that fuses spatial and temporal data for crowd congestion monitoring. The proposed framework leverages the strengths of deep learning and computer vision models for trajectory generation, and spatial connectivity of the occupied space for more effective crowd flow analysis and forecasting. To obtain a spatiotemporal mapping of each occupant in a public space, we compare empirically the use of popular CNN networks, YOLOv7 and Faster R-CNN, to automate the detection of people in CCTV videos, and then employ Kalman filter-based tracking algorithms to track people across sequences of video frames. Subsequently, using the floor plan information and homography transformation, a spatiotemporal mapping of each pedestrian is created. To capture the fused spatiotemporal information, the CMGraph is designed so that nodal data stores collective crowd flow information and graph topology represents the spatial connectedness of key egress regions. We also design a GCN-GRU model that demonstrates how CMGraphs may be utilized for spatiotemporal forecasting. To evaluate the framework, quantitative experiments are conducted on an annotated public dataset at the Grand Central Station, which has been widely used by researchers studying crowd scenes [9,43]. To further illustrate the practical application of the framework, qualitative congestion analysis is conducted on supplementary video captured at Stanford Stadium. The quantitative experimental results serve as demonstrative analysis for the plausibility of deploying the pedestrian detector, tracker, and crowd flow predictor models, whose performances are assessed with a set of standard evaluation metrics. Furthermore, qualitative visualizations such as bounding boxes, crowd count, and density heatmaps can serve as useful visual tools for crowd monitoring. Together, we demonstrate that the modular framework incorporated with machine learning models can be utilized to gather crowd mobility information by observing the spatiotemporal mapping of crowds at a public location, thereby facilitating assistance with the management of congestion hazards for stakeholders such as urban planners and infrastructure operators.

Future research could benefit from additional experiments of videos in different congested scenes. It should be noted that annotation of crowded scenes is a difficult and expensive task due to the need for intensive manual labor. Nevertheless, additional datasets can help further validate the framework's generalizability and robustness under diverse scenarios. Another line of future work could concentrate on developing methodologies for multi-camera pedestrian tracking to discover pedestrian mobility patterns between egresses that are too distant apart for a single camera to capture. Experiments could be conducted using multi-view pedestrian Re-ID models [49–51]. In order to track occupants in a large space with multiple egresses and cameras, these Re-ID models could potentially be integrated with tracking algorithms such as DeepSORT. Last but not least, to demonstrate our framework in large public spaces, especially during crowded events, additional surveillance video experiments should be conducted on larger testbeds.

Author Contributions: Conceptualization, V.W.H.W. and K.H.L.; methodology and experimentation, V.W.H.W.; writing—original draft preparation, V.W.H.W.; writing—review and editing, K.H.L.; supervision, K.H.L. All authors have read and agreed to the published version of the manuscript.

Funding: The research is supported by the Stanford Center at the Incheon Global Campus (SCIGC), which is sponsored in part under the National Program to Subsidize Attracting Foreign Educational Institution and Research Institutes published by the Ministry of Trade, Industry, and Energy of the Republic of Korea and managed by the Incheon Free Economic Zone Authority.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wang, J.; Ding, Y.N.; Liu, D.D. The Research on Early Warning of Preventing the Stampede on Crowded Places and Evacuated Technology. In Proceedings of the 2015 International Forum on Energy, Environment Science and Materials, Shenzhen, China, 25–26 September 2015; Atlantis Press: Paris, France, 2015; pp. 1544–1551.
2. Sindagi, V.A.; Patel, V.M. A Survey of Recent Advances in CNN-Based Single Image Crowd Counting and Density Estimation. *Pattern Recognit. Lett.* **2018**, *107*, 3–16. [CrossRef]
3. Lo, B.P.L.; Velastin, S.A. Automatic Congestion Detection System for Underground Platforms. In Proceedings of the 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing, ISIMP 2001 (IEEE Cat. No.01EX489), Hong Kong, China, 2–4 May 2001; pp. 158–161.
4. Martella, C.; Li, J.; Conrado, C.; Vermeeren, A. On Current Crowd Management Practices and the Need for Increased Situation Awareness, Prediction, and Intervention. *Saf. Sci.* **2017**, *91*, 381–393. [CrossRef]
5. Kizrak, M.A.; Bolat, B. Crowd Density Estimation by Using Attention Based Capsule Network and Multi-Column CNN. *IEEE Access* **2021**, *9*, 75435–75445. [CrossRef]
6. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
7. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]
8. He, K.; Gkioxari, G.; Dollar, P.; Girshick, R. Mask R-CNN. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 386–397. [CrossRef] [PubMed]
9. Zhou, B.; Wang, X.; Tang, X. Random Field Topic Model for Semantic Region Analysis in Crowded Scenes from Tracklets. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2011, Colorado Springs, CO, USA, 20–25 June 2011; pp. 3441–3448. [CrossRef]
10. Zhang, C.; Li, H.; Wang, X.; Yang, X. Cross-Scene Crowd Counting via Deep Convolutional Neural Networks. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 833–841.
11. Zhang, Y.; Zhou, D.; Chen, S.; Gao, S.; Ma, Y. Single-Image Crowd Counting via Multi-Column Convolutional Neural Network. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 589–597.
12. Boominathan, L.; Kruthiventi, S.S.S.; Babu, R.V. CrowdNet: A Deep Convolutional Network for Dense Crowd Counting. In Proceedings of the 24th ACM International Conference on Multimedia, Bangalore, India, 20–24 October 2021; ACM: Amsterdam, The Netherlands, 2016; pp. 640–644.
13. Sindagi, V.A.; Patel, V.M. CNN-Based Cascaded Multi-Task Learning of High-Level Prior and Density Estimation for Crowd Counting. In Proceedings of the 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Lecce, Italy, 29 August–1 September 2017; pp. 1–6.
14. Zeng, L.; Xu, X.; Cai, B.; Qiu, S.; Zhang, T. Multi-Scale Convolutional Neural Networks for Crowd Counting. In Proceedings of the 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 17–20 September 2017; pp. 465–469.
15. Yan, R.; Gong, S.; Zhong, S. Crowd Counting via Scale-Adaptive Convolutional Neural Network in Extremely Dense Crowd Images. *IJCAT* **2019**, *61*, 318. [CrossRef]
16. Gündüz, M.Ş.; Işık, G. A New YOLO-Based Method for Real-Time Crowd Detection from Video and Performance Analysis of YOLO Models. *J. Real-Time Image Process.* **2023**, *20*, 5. [CrossRef]
17. Magoo, R.; Singh, H.; Jindal, N.; Hooda, N.; Rana, P.S. Deep Learning-Based Bird Eye View Social Distancing Monitoring Using Surveillance Video for Curbing the COVID-19 Spread. *Neural Comput. Appl.* **2021**, *33*, 15807–15814. [CrossRef]
18. Chen, J.; Su, W.; Wang, Z. Crowd Counting with Crowd Attention Convolutional Neural Network. *Neurocomputing* **2020**, *382*, 210–220. [CrossRef]
19. Suarez, S. Grand Central Terminal's Original Lighting: Its Significance, Its Relationship with the Current Scheme, and Recommendations for Alternate Considerations. Master's Thesis, Columbia University, New York, NY, USA, 20 July 2015.
20. Sreenu, G.; Saleem Durai, M.A. Intelligent Video Surveillance: A Review through Deep Learning Techniques for Crowd Analysis. *J. Big Data* **2019**, *6*, 48. [CrossRef]
21. Fujiyoshi, H.; Hirakawa, T.; Yamashita, T. Deep Learning-Based Image Recognition for Autonomous Driving. *IATSS Res.* **2019**, *43*, 244–252. [CrossRef]
22. Belhadi, A.; Djenouri, Y.; Srivastava, G.; Djenouri, D.; Lin, J.C.-W.; Fortino, G. Deep Learning for Pedestrian Collective Behavior Analysis in Smart Cities: A Model of Group Trajectory Outlier Detection. *Inf. Fusion* **2021**, *65*, 13–20. [CrossRef]
23. Du, H.; Jin, T.; Song, Y.; Dai, Y.; Li, M. A Three-Dimensional Deep Learning Framework for Human Behavior Analysis Using Range-Doppler Time Points. *IEEE Geosci. Remote Sens. Lett.* **2020**, *17*, 611–615. [CrossRef]
24. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollar, P. Focal Loss for Dense Object Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 318–327. [CrossRef]
25. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In *Computer Vision–ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2016; Volume 9905, pp. 21–37. ISBN 978-3-319-46447-3.

26. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
27. Wang, C.-Y.; Bochkovskiy, A.; Liao, H.-Y.M. YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. *arXiv* **2022**, arXiv:2207.02696. [CrossRef]
28. Bewley, A.; Ge, Z.; Ott, L.; Ramos, F.; Upcroft, B. Simple Online and Realtime Tracking. In Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; pp. 3464–3468.
29. Wojke, N.; Bewley, A.; Paulus, D. Simple Online and Realtime Tracking with a Deep Association Metric. In Proceedings of the 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 17–20 September 2017; pp. 3645–3649.
30. Kuhn, H.W. The Hungarian Method for the Assignment Problem. *Nav. Res. Logist.* **1955**, *2*, 83–97. [CrossRef]
31. Zheng, L.; Bie, Z.; Sun, Y.; Wang, J.; Su, C.; Wang, S.; Tian, Q. MARS: A Video Benchmark for Large-Scale Person Re-Identification. In Proceedings of the Computer Vision–ECCV 2016, Amsterdam, The Netherlands, 11–14 October 2016; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 868–884.
32. Dendorfer, P.; Osep, A.; Milan, A.; Schindler, K.; Cremers, D.; Reid, I.; Roth, S.; Leal-Taixé, L. MOTChallenge: A Benchmark for Single-Camera Multiple Target Tracking. *Int. J. Comput. Vis.* **2021**, *129*, 845–881. [CrossRef]
33. Dendorfer, P.; Rezatofighi, H.; Milan, A.; Shi, J.; Cremers, D.; Reid, I.; Roth, S.; Schindler, K.; Leal-Taixé, L. MOT20: A Benchmark for Multi Object Tracking in Crowded Scenes. *arXiv* **2020**, arXiv:2003.09003. [CrossRef]
34. Su, H.; Deng, J.; Fei-Fei, L. Crowdsourcing Annotations for Visual Object Detection. In Proceedings of the Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence, Toronto, ON, Canada, 15 July 2012; Volume WS-12-08, pp. 40–46.
35. Ma, Y.; Soatto, S.; Košecká, J.; Sastry, S.S. *An Invitation to 3-D Vision*; Interdisciplinary Applied Mathematics; Springer: New York, NY, USA, 2004; Volume 26, ISBN 978-1-4419-1846-8.
36. Bazargani, H.; Bilaniuk, O.; Laganière, R. A Fast and Robust Homography Scheme for Real-Time Planar Target Detection. *J. Real-Time Image Proc.* **2018**, *15*, 739–758. [CrossRef]
37. Panagopoulos, G.; Nikolentzos, G.; Vazirgiannis, M. Transfer Graph Neural Networks for Pandemic Forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence 2021, Online, 2–9 February 2021; Volume 35, pp. 4838–4845. [CrossRef]
38. Bai, J.; Zhu, J.; Song, Y.; Zhao, L.; Hou, Z.; Du, R.; Li, H. A3T-GCN: Attention Temporal Graph Convolutional Network for Traffic Forecasting. *ISPRS Int. J. Geo-Inf.* **2020**, *10*, 485. [CrossRef]
39. Pareja, A.; Domeniconi, G.; Chen, J.; Ma, T.; Suzumura, T.; Kanezashi, H.; Kaler, T.; Schardl, T.B.; Leisersen, C.E.; Leiserson, C.E. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. *AAAI* **2020**, *34*, 5363–5370. [CrossRef]
40. Zhao, L.; Song, Y.; Zhang, C.; Zhang, C.; Liu, Y.; Wang, P.; Lin, T.; Deng, M.; Deng, M.; Deng, M.; et al. T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 3848–3858. [CrossRef]
41. Kipf, T.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
42. Cho, K.; van Merriënboer, B.; Bahdanau, D.; Bengio, Y. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In Proceedings of the SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 103–111.
43. Yi, S.; Li, H.; Wang, X. Understanding Pedestrian Behaviors from Stationary Crowd Groups. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2015, Boston, MA, USA, 7–12 June 2015. [CrossRef]
44. Bernardin, K.; Stiefelhagen, R. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. *EURASIP J. Image Video Process.* **2008**, *2008*, 1–10. [CrossRef]
45. Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
46. Github Repository. Available online: https://github.com/vivian-wong/pedestrian-test (accessed on 23 February 2023).
47. Jiang, W.; Luo, J. Graph Neural Network for Traffic Forecasting: A Survey. *Expert Syst. Appl.* **2022**, *207*, 117921. [CrossRef]
48. Gwynne, S.M.V.; Hunt, A.L.E.; Thomas, J.R.; Thompson, A.J.L.; Séguin, L. The Toilet Paper: Bathroom Dwell Time Observations at an Airport. *J. Build. Eng.* **2019**, *24*, 100751. [CrossRef]
49. Li, H.; Dong, N.; Yu, Z.; Tao, D.; Qi, G. Triple Adversarial Learning and Multi-View Imaginative Reasoning for Unsupervised Domain Adaptation Person Re-Identification. *IEEE Trans. Circuits Syst. Video Technol.* **2022**, *32*, 2814–2830. [CrossRef]
50. Tao, D.; Guo, Y.; Yu, B.; Pang, J.; Yu, Z. Deep Multi-View Feature Learning for Person Re-Identification. *IEEE Trans. Circuits Syst. Video Technol.* **2018**, *28*, 2657–2666. [CrossRef]
51. Xu, Y.; Jiang, Z.; Men, A.; Wang, H.; Luo, H. Multi-View Feature Fusion for Person Re-Identification. *Knowl.-Based Syst.* **2021**, *229*, 107344. [CrossRef]