Advanced Engineering Informatics 73 (2026) 104583

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

TrajGATFormer: A graph-based transformer approach for worker and obstacle trajectory prediction in off-site construction environments☆

Mohammed Alduais [a], Xinming Li [b], Qipei Mei [a,*]

[a] Department of Civil & Environmental Engineering, University of Alberta, Edmonton, T6G 2R3, AB, Canada
[b] Department of Mechanical Engineering, University of Alberta, Edmonton, T6G 2R3, AB, Canada

ARTICLE INFO

Keywords:
Safe workplace
Trajectory prediction
Offsite construction
Data-driven approaches
Transformer

ABSTRACT

As the demand grows within the construction industry for processes that are not only faster but also safer and more efficient, offsite construction has emerged as a solution, though it brings new safety risks due to the close interaction between workers, machinery, and moving obstacles. Predicting the future trajectories of workers and taking into account social and environmental factors is a crucial step for developing collision-avoidance systems to mitigate such risks. Traditional methods often struggle to adapt to the dynamic and unpredictable nature of construction environments. Many rely on simplified assumptions or require hand-crafted features, limiting their ability to respond to complex interactions between workers and moving obstacles. While recent data-driven methods have improved the modeling of temporal patterns, they still face challenges in capturing long-term behavior and accounting for the spatial and social context crucial to collision risk assessment. To address these limitations, this paper proposes a comprehensive framework integrating YOLOv10n and DeepSORT for precise detection and tracking, alongside two novel trajectory prediction models: TrajGATFormer and TrajGATFormer-Obstacle. YOLOv10n serves as the backbone for object detection, reliably identifying workers and obstacles in diverse construction scenes, while DeepSORT efficiently tracks each detected entity over time, assigning unique IDs to maintain continuity across frames. The two predictive models leverage a transformer encoder–decoder architecture with Graph Attention Networks (GAT) to capture both temporal dependencies and spatial interactions. TrajGATFormer focuses on predicting worker trajectories, achieving an average displacement error (ADE) of 1.25 m and a final displacement error (FDE) of 2.35 m over a 4.8-second prediction horizon. TrajGATFormer-Obstacle extends this framework to handle both worker and obstacle predictions, achieving even higher accuracy, with an ADE of 1.03 m and an FDE of 2.06 m. Comparative analysis showed that both models outperformed traditional methods, reducing ADE and FDE by up to 35% and 38%, respectively. These models can serve as the backbone for constructing collision-avoidance alarm systems that can be installed at offsite construction facilities.

1. Introduction

The construction industry is considered as one of the major drivers of the global economy in the coming few years. According to a report by Oxford Economics, the global construction industry is projected to grow by 42% this decade, reaching 15.2 trillion USD compared to 10.7 trillion USD in the previous decade [1]. Furthermore, total global spending on construction accounted for 13% of global GDP in 2020 and is projected to rise to 13.5% by 2030 [1]. Although the construction sector contributes significantly to economic growth, it is also one of the most hazardous industries. In 2020 alone, the construction industry in the United States was responsible for 1034 fatal injuries and more than 70,000 non-fatal injuries [2]. Between 2011 and 2021, more than 10,000 construction workers were killed on the job due to exposure to various hazardous situations, including (1) falls to lower levels, (2) being struck by objects, (3) electrocution, (4) caught-in/between incidents, and (5) other hazards [2]. According to OSHA, falls to lower levels, being struck by objects, electrocution, and caught-in/between incidents are classified as "Focus Four Hazards", whereas all other hazards are classified as "Non-Focus Four Hazards" [3]. Focus Four hazards are the primary cause of fatal injuries, accounting for up to 65.5% of all fatal incidents, which corresponds to approximately 6900 deaths. Similarly, from 2011 to 2020, Focus Four injuries accounted for over 40.4% of non-fatal injuries, totaling more than 315,000 incidents. Struck-by accidents accounted for more than 17% of fatal injuries,

☆ This article is part of a Special issue entitled: 'ADVEI_AI & Robotics for IC' published in Advanced Engineering Informatics.

* Corresponding author.
E-mail address: qipei.mei@ualberta.ca (Q. Mei).

https://doi.org/10.1016/j.aei.2026.104583
Received 18 October 2025; Received in revised form 26 February 2026; Accepted 8 March 2026
Available online 17 March 2026

accounting for 1700 injuries, and more than 20% non-fatal injuries, accounting for more than 167,000 injuries.

As a result, off-site construction emerges as a possible approach to improve worker safety. Off-site construction provides a faster, safer and more controlled working environment with improved quality control practices, and is less harmful to the environment compared to traditional on-site construction [4–7]. According to the Mackenzie report, off-site construction can reduce construction time by 50% and reduce cost in some projects by 20% [8]. However, off-site construction facilities are dynamic environments in which workers operate in congested areas and in close proximity to construction equipment such as lifting cranes and robotic machines, which can lead to unsafe situations [9]. Furthermore, the lack of positional awareness among workers when operating construction machinery can be attributed to abrupt movements and rotations of the machinery, obscured sight lines, and the noisy working environment [9]. Therefore, to mitigate such risks, the prediction of the worker's trajectory, which can be defined as the prediction of future positions based on their previous observations and social and environmental interactions [10], plays a crucial role in providing essential information for the development of a struck-by alarm system that alerts workers when a potential hazard situation can occur [11,12].

Recent advances in trajectory prediction have introduced various approaches that range from traditional statistical models to sophisticated machine learning techniques [10]. Traditional methods, like Kalman Filters (KF) [13], and Hidden Markov Models (HMMs) [14], rely on predefined assumptions and historical data, which often limit their adaptability in highly dynamic and unpredictable settings like construction sites [10]. More recently, data-driven approaches, such as Long Short-Term Memory (LSTM) networks [15], have become popular for their ability to capture temporal patterns in sequential data. LSTMs have shown promise in predicting worker movement by leveraging historical and contextual information, they still struggle in complex environments that require accurate long-term predictions [16].

Transformer models have gained attention for trajectory prediction due to their inherent capability to capture complex dependencies across long time-frames, without the limitations of traditional recurrent models [17,18]. While their initial success was demonstrated in natural language processing, their versatility has made them popular in various domains, including trajectory prediction [19]. Additionally, Graph Attention Networks (GATs) have been applied to model spatial relationships between entities [20], making them well-suited for dynamic environments where interactions between workers and machinery are crucial. Combining transformers with GATs offers a comprehensive approach to trajectory prediction by accounting for both temporal and spatial interactions.

In this paper, we propose a trajectory prediction framework that includes two distinct models. The first model is designed specifically for predicting worker trajectories, while the second model extends this capability by incorporating both workers and moving obstacles. The framework begins by detecting and tracking objects using a surveillance camera, followed by preprocessing the acquired data. Each model employs a hybrid approach that combines Transformer networks and GATs. By leveraging the strengths of both architectures, the models effectively capture the spatial and temporal complexities of dynamic environments, ensuring more accurate trajectory predictions.

The paper is organized as follows: Section 2 reviews the literature and methods used for trajectory prediction. Section 3 introduces the proposed framework, proposed model architectures, and the implementation details. Section 4 discusses the datasets used. Section 5 presents the results of the proposed models along with different trajectory prediction methods. Section 6 provides a conclusion, and Section 7 presents limitations of the proposed framework and suggests future direction.

2. Related work

With the rapid development in the fields of computer vision and autonomous vehicles, trajectory prediction emerges as one of the fundamental challenges that need to be solved and studied [12]. There are two approaches to solving the problem of trajectory prediction.

The first approach is the knowledge-based approach [10], which is referred to by multiple names in the literature, for example physics-based [21], expert-based [22], reasoning-based [23], or traditional approach [24]. In this approach, models need predefined rules or functions by humans that try to interpret or represent the physical, social, and physiological information of the target [10]. For example, Constant-Velocity method, is considered as one of the simplest yet effective models for trajectory prediction [25]. This method assumes that motion of vehicles/pedestrians continues without taking into consideration any other factors such as social interactions. Kalman Filtering (KF) [13] which was introduced in 1960, can be used to introduce uncertainty information to kinematic models [26]. For example, Elnagar [27], proposed an algorithm to predict the trajectories of moving obstacles combining the KF and the constant acceleration model. However, one of the main disadvantages of these methods is the assumption of constant motion, which results in poor performance when considering long term predictions ($> 2s$) [26]. Hidden Markov Models (HMMs) are recognized as probabilistic models used for the analysis of sequential data [14]. These models operate under the assumption that the data follows a Markov process with states that are not directly observable. A model proposed by Ye et al. [28], used HMMs with two layers of hidden states for vehicle trajectory prediction. The authors used statistical methods to determine the HMM parameters. In the prediction phase, the authors introduced the Viterbi algorithm to solve the decoding of HMM and find the optimal sequence of hidden states. The social force model (SFM) proposed by Helbing and Molnar [29], was one of the pioneering works to model pedestrian interaction and inspired many researches to build on this concept. This approach describes that the movement of pedestrians is based on a social force that drives them to perform certain actions, in this case their movement.

The second approach is the data-driven approach, also known as supervised deep learning methods [10]. This approach is widely used in fields such as self-driving cars, robotics, and pedestrian trajectory prediction [10]. Unlike the knowledge-based approach, this method does not require manual definition of human behavior. Instead, it offers more flexible models capable of adapting to complex scenarios, although it relies on large amounts of data for training [10]. An example of data-driven approach is Convolution Neural Networks (CNNs), which is a famous method used in computer vision problems such as image classification [30]. A model proposed by Nikhil and Morris [31], where they used a CNN-based model to predict the trajectories of pedestrians. They used a fully connected layer to embed the trajectories, followed by multiple layers of CNN that are used to extract the temporal information. A fully connected layer was added to the end of the model architecture that takes the features from the CNN layers to generate future trajectories. However, this model did not consider social and spatial information for its prediction. To overcome this issue LSTM methods are introduced, which was used by many researchers in the application of trajectory prediction [10]. One of the pioneers in this field is the Social-LSTM model that was proposed by Alahi et al. [32]. The main break throw in his proposed method was the introduction of a social pooling layer that takes into account the influence of the nearby pedestrians. With the success of Social-LSTM, LSTM gained popularity among researchers to use it to solve trajectory prediction. Graph Neural Networks (GNNs) on the other hand have been used in various applications, such as road traffic prediction [33,34], text classification [35], and in machine translation [17,36,37]. In the domain of trajectory prediction, Social BiGAT which was proposed by Kosaraju et al. [38], Incorporated Graph Attention Networks (GATs) to enhance the ability

Table 1
A summary of trajectory prediction categories and their corresponding methods.

Category	Method	Input	Reference
Knowledge-Based	Constant-Velocity	Past trajectories	[25]
	Kalman filter	Past trajectories and velocity	[27]
	Hidden Markov Models	Vehicle past trajectories	[28]
	Social Force Model	Past and neighbor trajectories	[29]
Data-driven	Convolutional Neural Networks	Displacement volume	[31]
	Recurrent Neural Networks	Past and neighbor trajectories	[32]
	Graph Neural Networks	Past trajectories and scene information	[38]
	Transformer Networks	Past trajectories	[19]

of the model to capture the social interactions between pedestrians, while the main model architecture uses GAN. However, their limitation lies in the inability to capture temporal dependencies, which are essential for predicting movement over time [20]. Since Transformer Networks are capable of capturing long-range dependencies in sequential problems [18], it was utilized as a possible alternative. One of the pioneering works that introduced transformer networks to pedestrian trajectory prediction was the trajectory transformer (TF) by Giuliari et al. [19]. They used the same transformer encoder–decoder [17], which can predict future trajectories of multiple pedestrians at once and reported promising results compared to other trajectory prediction models. However, their approach models each pedestrian without considering any social or scene interactions.

There are several studies that proposed trajectory predication methods in the construction industry. For example, a model was presented by Zhu et al. [39] that was designed to predict workers and moving equipment trajectories based on inputs from two video cameras. This model was designed by utilizing Kalman filters and relies on past trajectories without taking into account other important information, such as the influence of other entities on each other. Wang et al. [40] used a deep region-based convolutional neural network (R-CNN) to detect and classify workers and construction equipment and then designed a CNN-based model for the prediction of the trajectory of these objects. Later, Cai et al. [12] proposed context-augmented LSTM model that predicts workers trajectories in dynamic construction sites by integrating individual movement with contextual information. However, one of the drawbacks that faced this model that they considered the final destination as a prior knowledge. Another models proposed by Yang et al. [11], proposed an environment-aware worker trajectory prediction model in a modular construction site, using LSTM, and it takes into account the following: (1) worker movement, (2) worker-to-worker and (3) environment-to-worker interactions. However, these models did not consider the influence of dynamic obstacles but considered the use of stationary obstacles as a fixed coordinates. Recent studies have increasingly explored deep learning–based trajectory prediction tailored to construction environments. Notably, Heravi et al. [41] proposed an activity-aware deep learning framework for 3D human motion trajectory prediction in construction settings, where worker activities are explicitly incorporated to enhance prediction accuracy. Their work demonstrates that integrating contextual and semantic information, such as task type and motion intent, can significantly improve motion forecasting performance in complex construction scenarios. From a site-context perspective, Duan et al. [42] proposed a risk-driven, block-based worker trajectory prediction approach that quantifies safety risk across site blocks and trains a Transformer–LSTM model to predict the worker's next block. In addition, Bao, Bu, and Rui [43] introduced a computer vision–based approach to recognize the spatial relevance between engineering vehicles and workers for monitoring struck-by accidents, supporting proactive safety monitoring in dynamic construction environments. See Table 1 for a summary of some of the trajectory prediction methods.

3. Proposed framework

The proposed framework, illustrated in Fig. 1, includes four primary components: (1) Object Detection & Tracking, (2) Data Pre-processing, (3) Model Development, and (4) Model Evaluation. Firstly, in the Object Detection & Tracking phase, the YOLOv10n model is used to detect both workers and obstacle in the scene. Using the results from the YOLOv10n model, a DeepSORT algorithm is used to track workers and obstacle. In this study, the term "obstacle" is used from the worker-trajectory modeling perspective to denote a dynamic non-worker entity that influences worker trajectories, and the moving panel is used as one example of such an obstacle in the experiments. Secondly, data pre-processing steps are implemented in order to modify the data to match the input format of the proposed model. Thirdly, the model development is introduced with the two proposed models. Fourthly, the model is evaluated using specific metrics. Each component will be explained in detail in the following sections.

3.1. Object detection and tracking

To automate the detection and tracking of workers and obstacle, the YOLOv10n model was trained on a real-world construction dataset. YOLOv10n is an object detection algorithm that was published by [44]. This model is commonly used in object detection applications as it provides a real-time, efficient, and accurate object detection [44]. YOLOv10 has many model types based on its size. In this paper, we used YOLOv10n, which is a smaller model with about 2.3 million parameters. In addition to using YOLOv10n, the DeepSORT algorithm [45] was utilized to add another parameter to the dataset, which is the tracking ID. DeepSORT uses a pre-trained feature extractor that was originally trained on large-scale pedestrian re-identification datasets. In this study, the model was used without retraining, and the results were checked manually to verify the outputs. This will help to prepare the data to match the input format.

3.2. Data pre-processing

In this step, the acquired data from step one is transformed into the following format:

[Frame number, Class, ID, pos_x, pos_y]

Where Frame number represent the current frame number, Class represents the class of the detected object (i.e. Worker or Obstacle), ID represents the ID of the current tracked object, pos_x represents the x-coordinates for the tracked object, and pos_y represents the y-coordinates for the tracked object. The output from YOLOv10n (for object detection) and DeepSORT (for object tracking) provides object positions in terms of pixel coordinates in the image. However, for applications that require spatial reasoning or real-world measurements, it is crucial to convert these pixel coordinates to real-world coordinates.

The conversion from pixel coordinates to real-world coordinates is achieved by estimating a homography matrix using four known reference points in the scene. In this study, these reference points

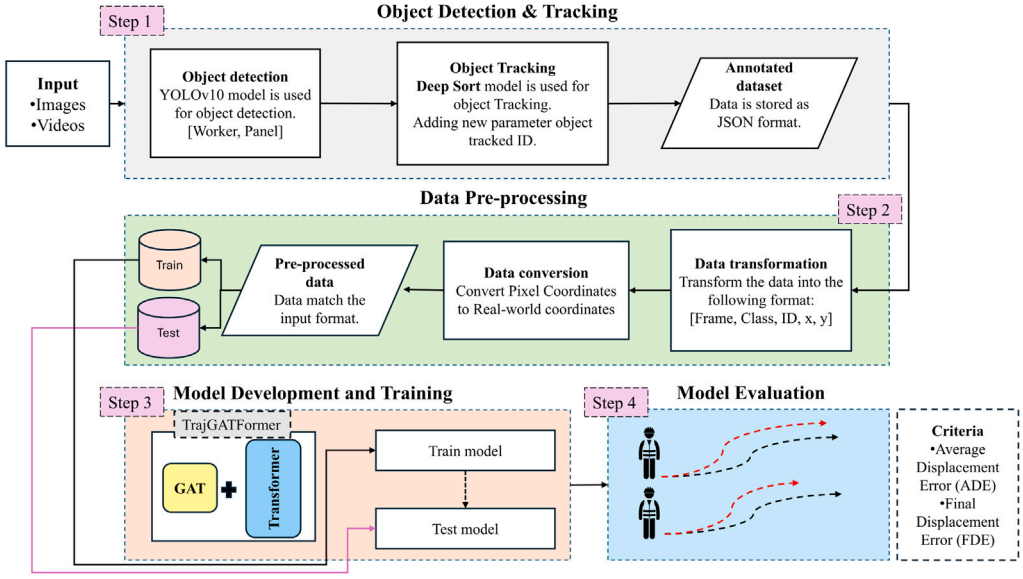

Fig. 1. Overview of the proposed framework.

correspond to known real-world coordinates of panel corners within the facility, along with their associated pixel coordinates in the image. This approach assumes a planar workspace and a fixed camera viewpoint, which is representative of controlled off-site construction environments where workstations are typically flat and camera installations remain static. Prior to deployment, camera calibration can be performed using standard procedures, such as checkerboard-based calibration, to ensure accurate homography estimation [46]. The function cv2.getPerspectiveTransform() in OpenCV is used to compute the homography matrix. Once the homography matrix is obtained, the pixel coordinates can be mapped to their corresponding real-world positions.

3.3. Model development

Inspired by the work of [18], the proposed model utilizes the transformer encoder–decoder architecture. The main components of transformer models include the *Encoder*, *Decoder*, *Attention Mechanism*, *Multi-head Attention*, and *Positional Encoding*.

The primary function of the encoder is to encode the source inputs into representations that can later be decoded into meaningful target sequences. The encoder consists of N identical layers (commonly $N = 6$), each of which includes two main components: *Multi-head Attention* and *Fully Connected Feed-Forward Network (FFN)*. These components are connected by residual connections, which add the input of each layer to its output, helping to stabilize the training [17]. Layer normalization is then applied to the output of each sub-layer to ensure smooth gradient flow.

The decoder generates the target sequence by utilizing the representations produced by the encoder. Its structure is similar to that of the encoder but includes an additional attention module. The first attention module processes the encoded information from the encoder, while the second attention module applies masking to the output embeddings, ensuring that the model attends only to previous tokens in the sequence. This masking helps preserve the autoregressive property of sequence generation, where the model generates one token at a time based on the preceding context.

In the attention modules, the attention mechanism is used to help the model focus on relevant parts of the input sequence. The attention scores are computed based on *query (Q)*, *key (K)*, and *value (V)* matrices using Eq. (11).

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

Here, d_k represents the dimension of the key vector. The softmax function ensures that the attention scores are normalized into a probability distribution, allowing the model to weigh different parts of the input sequence appropriately.

To further enhance the model's ability to focus on various aspects of the input, transformers employ *multi-head attention*, where multiple attention mechanisms run in parallel. Each head attends to different subspaces of the input, allowing the model to capture various aspects of the input sequence. The outputs from all attention heads are concatenated and passed through a linear transformation to integrate the information.

Since transformers do not have an inherent mechanism to process sequential order, *positional encoding* is added to the input embeddings to encode the position of each token in the sequence. The positional encoding function is defined by Eqs. (2) and (3).

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \tag{2}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i+1}{d}}}\right) \tag{3}$$

Where pos is the position of the token in the sequence, i is the dimension index, and d is the model dimension. This encoding ensures that each token's position is uniquely represented in the model's input embeddings, allowing the transformer to incorporate the sequential nature of the data.

However, to better understand the relationships between workers, GAT [47], was implemented to model the social interactions between them. GAT is used on graph-based data, which uses self-attention to calculate the importance of each neighbor on the target node. GAT have two main components: (1) Nodes, and (2) Edges. where nodes can represent individual points in the data and the edges represent the relationship between the target node and its neighbor.

Now lets assume we have a graph $G = (V, E)$, where V, E represents the set of nodes (i.e. workers) and set of edges (i.e. Euclidean distance), respectively. Each node $i \in V$ is associated with a feature vector $\tilde{h}_i \in \mathbb{R}^F$, where F is the feature dimension. In this study, $\tilde{h}_i$ represents the worker's motion-state feature for node i, derived from the worker's observed 2D trajectory coordinates (position information) after projection into the model feature space. The feature now are transformed into high level features, using linear transformation with

a learnable parameter matrix W, where $W \in \mathbb{R}^{F' \times F}$ and F' is the new dimension after transformation see Eq. (4).

$$\tilde{h}_i' = W\tilde{h}_i, \quad \tilde{h}_i' \in \mathbb{R}^{F'} \tag{4}$$

The attention coefficient for the node pair (i, j) at time step t is computed using Eq. (5).

$$\alpha_{ij}^t = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^T[W\mathbf{h}_i^t \parallel W\mathbf{h}_j^t]))}{\sum_{k\in\mathcal{N}_i}\exp(\text{LeakyReLU}(\mathbf{a}^T[W\mathbf{h}_i^t \parallel W\mathbf{h}_k^t]))} \tag{5}$$

Where $a \in \mathbb{R}^{2F'}$ is the weight vector of a single-layer feedforward neural network, $\parallel$ denotes the concatenation operation, $\mathcal{N}_i$ represents the neighbors of node i on the graph, exp denotes the exponential function, and LeakyReLU is the Leaky Rectified Linear Unit activation function [48]. Finally, the output of a single GAT layer for a given node i at time t is defined by Eq. (6).

$$\hat{h}_i = \sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}W\tilde{h}_j'\right) \tag{6}$$

Where σ represents a non-linear activation function and $\tilde{h}_j'$ represents the transformed feature vector of node j.

In the form of trajectory prediction, for a given person or obstacle i, we define the observed trajectory sequences as:

$$T_{\text{obs}} = \{x_t^{(i)}\}_{t=-(T_{\text{obs}}-1)}^{0} \tag{7}$$

Where $x_t^{(i)} \in \mathbb{R}^2$ represents the Cartesian coordinates (position) at time t, and T_{obs} is the number of observed time steps. Before feeding into the transformer, the 2D positions are projected into a higher-dimensional space D:

$$e_{\text{obs}}^{(i,t)} = x_t^{(i)}W_x \tag{8}$$

Where $W_x \in \mathbb{R}^{2\times D}$ is a learned weight matrix, and $e_{\text{obs}}^{(i,t)} \in \mathbb{R}^D$ is the transformed embedding of the observed position. Since Transformers do not have inherent order-awareness, a positional encoder is added to provide temporal context.

Each input embedding $e_{\text{obs}}^{(i,t)}$ is timestamped as:

$$\xi_{\text{obs}}^{(i,t)} = PE + e_{\text{obs}}^{(i,t)} \tag{9}$$

Where PE is the positional encoding vector of the same dimension D. The positional encoding is defined in Eqs. (2) and (3). Once the input embeddings are calculated for both workers and obstacles we get: $\xi_{\text{worker,obs}}^{(i,t)}$ for worker input embeddings and $\xi_{\text{obstacle,obs}}^{(i,t)}$ for obstacle input embeddings. Now for GAT model each agent i at time t has a corresponding node embedding $\xi_{\text{worker,obs}}^{(i,t)}$. The GAT computes a new representation for each node using self-attention over its neighbors. First, each node embedding is transformed using a learnable weight matrix using Eq. (4). The importance of a neighboring node j to node i is computed using Eq. (5). The new node representation is computed as a weighted sum of its neighbors' embeddings using Eq. (6) and we get:

$$\hat{h}_{worker,i} = \sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}W\tilde{h}_j'\right) \tag{10}$$

Now for the input to the transformer encoder we have $\xi_{\text{worker,obs}}^{(i,t)}$ as the input for the worker encoder. This input sequence is passed through L Transformer encoder layers. Within each layer, the self-attention mechanism computes queries (Q), (K), and (V) by multiplying the input sequence $\xi_{\text{obs}}^{(i,t)}$ with learned weight matrices W_Q, W_K, and W_V, respectively. The attention is computed using a scaled dot-product using Eq. (1), and the results are weighted and aggregated through multi-head attention. The attention output is normalized using a LayerNorm operation. The normalized output is then passed through a feedforward network, where it undergoes a non-linear transformation with ReLU activation. After processing, the layer output is again normalized. This process is repeated through all layers of the encoder, resulting in the final encoded sequence. The final output of the Transformer worker encoder is denoted as $\hat{h}_{\text{worker,encoder}}^{(L)}$, which represents motion dynamics learned solely from historical trajectory information. A similar process is applied to the obstacle encoder, resulting in $\hat{h}_{\text{obstacle,encoder}}^{(L)}$. The final encoder output is shown in Eq. (11). The addition of $\hat{h}_{\text{worker,encoder}}^{(L)}$ and $\hat{h}_{\text{worker},i}$ ensures that the same dimensions are used. The encoded sequence H_{encoded} is passed as the input to the decoder. The decoder uses this encoded representation to generate the predicted future trajectories. It is important to note that the proposed framework performs multi-agent trajectory prediction rather than single-agent prediction. At each inference step, the model jointly predicts future trajectories for all workers and moving obstacles present in the scene. Each agent (worker or obstacle) is treated as an independent entity with its own observed trajectory sequence, while interactions are captured through the GAT and shared latent representations. The decoder outputs a set of predicted trajectories simultaneously, enabling concurrent prediction of multiple workers and obstacles within a single forward pass.

$$H_{\text{encoded}} = \text{Concat}\left(\hat{h}_{\text{worker,encoder}}^{(L)} + \hat{h}_{\text{worker},i}, \hat{h}_{\text{obstacle,encoder}}^{(L)}\right) \tag{11}$$

3.3.1. TrajGATFormer model

TrajGATFormer is designed to process worker trajectory information. It will take the output from Eq. (10) and add it to the output from the worker encoder $\hat{h}_{\text{worker,encoder}}^{(L)}$. The results will be used as an input to the decoder. Fig. 2, shows overview of TrajGATFormer model architecture.

3.3.2. TrajGATFormer-obstacle model

Since TrajGATFormer did not incorporate the effect of moving obstacles on the workers trajectory prediction. It only encoded the past trajectories of workers and ignored a critical object that they normally work in proximity with. Therefore, to enhance the model's ability to represent real-world scenarios on construction sites, the proposed approach integrates information about moving obstacles, specifically floor or roof panels, into the trajectory prediction framework. The core of the model remains inspired by TrajGATFormer; however, a significant modification is made by introducing a dedicated encoder to process obstacle information. The worker encoder and obstacle encoder do not share parameters and are trained to learn entity-specific temporal representations. Interaction between workers and obstacles is modeled at the latent level by concatenating the encoded obstacle representations with the worker representations augmented by the GAT-based interaction features. Specifically, the final worker encoder output is combined with the GAT output and concatenated with the obstacle encoder output to form a shared latent representation H_{encoded}, which is then provided to a single decoder. This design allows the model to preserve independent feature extraction for workers and obstacles while enabling joint reasoning over their interactions during trajectory prediction. In the current architecture, the GAT is used to model worker-worker interactions only, while obstacle trajectories are encoded separately and fused at the latent level. This design preserves entity-specific temporal feature learning and avoids a more complex heterogeneous graph formulation (with node/edge-type definitions and additional parameters), which was not adopted given the limited size of the available obstacle-motion data. Fig. 3, shows overview of TrajGATFormer-Obstacle worker model architecture.

3.4. Model evaluation

In this section, baseline models and the evaluation metrics are discussed.

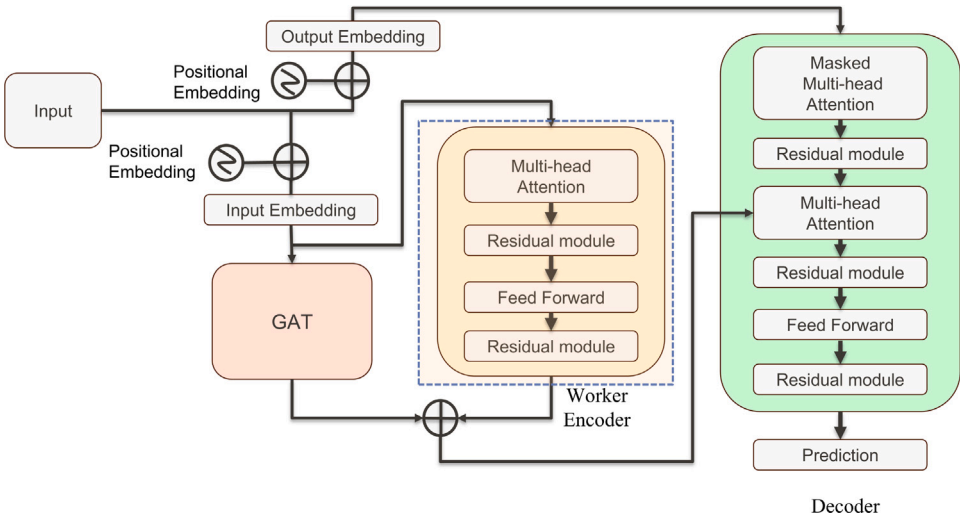

Fig. 2. Overview of our TrajGATFormer worker trajectory prediction model.

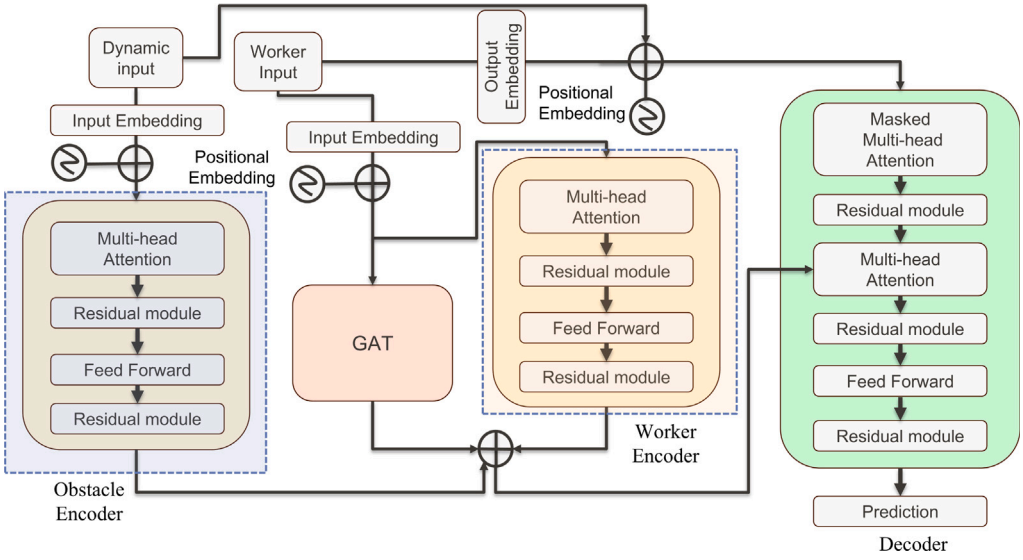

Fig. 3. Overview of our TrajGATFormer-Obstacle worker and obstacle trajectory prediction model.

3.4.1. Baseline models

In this paper different models were used to compare the performance of our model. The first model is Social-LSTM which was proposed by Alahi et al. [32]. The second model Social-GAN which was proposed Gupta et al. [49]. The third and forth models are EA-Distance and EA-Direction proposed by Yang et al. [11]. The proposed model is used for worker trajectory prediction in a modular construction site, using LSTM, and it takes into account the following: (1) worker movement, (2) worker-to-worker and (3) environment-to-worker interactions. Using these models we can see whether TrajGATFormer is able to outperform them or not.

3.4.2. Evaluation metrics

The evaluation metrics used in this paper are: (1) Average Displacement Error (ADE) and (2) final displacement error (FDE). ADE is defined as the mean square error between all the predicted trajectories and the corresponding ground-truths and can be calculated using Eq. (12).

$$\mathrm{ADE} = \frac{\sum_{i=1}^{N}\sum_{t=T_{\mathrm{obs}}+1}^{T_{\mathrm{pred}}}\left\|(\hat{x}_i^t,\hat{y}_i^t)-(x_i^t,y_i^t)\right\|}{N\times(T_{\mathrm{pred}}-T_{\mathrm{obs}}-1)} \tag{12}$$

Where N represents the number of workers or obstacle, T_{pred} denote the predicted coordinates of workers or obstacle i at the time instant t, $(\hat{x}_i^t,\hat{y}_i^t)$ denotes the predicted coordinates, (x_i^t,y_i^t) denote the corresponding ground-truth coordinates, and $\|\cdot\|$ is the Euclidean distance. FDE can be defined as the mean square error between the last coordinates of the predicted trajectories and the corresponding ground-truth and can be calculated using Eq. (13). The units of both ADE and FDE are in meters as they use real-world coordinates.

$$\mathrm{FDE} = \frac{\sum_{i=1}^{N}\left\|\left(\hat{x}_i^{(T_{\mathrm{pred}})},\hat{y}_i^{(T_{\mathrm{pred}})}\right)-\left(x_i^{(T_{\mathrm{pred}})},y_i^{(T_{\mathrm{pred}})}\right)\right\|}{N} \tag{13}$$

ADE and FDE are used in this work because they are widely adopted standard metrics in trajectory prediction and enable fair comparison with prior studies. ADE measures the average error over the prediction horizon, while FDE captures the final-position error, which is particularly relevant for safety-oriented forecasting. These two metrics have been extensively reported in prior work on pedestrian/agent trajectory prediction, including Social-LSTM [32], Social-GAN [49], Transformer-based trajectory models [19], and AgentFormer [50].

Table 2
TrajGATFormer model parameters.

Model parameter	Value
Worker Encoder Layers	1
Decoder Layers	1
d_{model}	512
Attention Heads	8
Worker d_k	256
Dropout	0.1

3.5. Implementation details

The implementation details of the proposed framework will be explained separately for Object detection and tracking, TrajGATFormer, and TrajGATFormer-Obstacle. However, the same hardware were used for training all of them. The hardware used for training is NVIDIA RTX 3060 GPU with 12 GB of memory, an AMD Ryzen 9 5900 12-Core Processor at 3.00 GHz, and 32 GB of RAM. The datasets used were split into three subsets, with a split of 70%, 20%, and 10% for training, validation, and testing, respectively.

First, object detection for both workers and obstacles was performed using the YOLOv10n model. Manual annotation was conducted only to create ground-truth labels for detector training and was not required during inference. A total of 2163 frames were manually annotated and expanded to 9288 training samples through data augmentation techniques, including brightness adjustment (−10% to +10%), horizontal flipping, and vertical flipping. This model was trained for 400 epochs, batch size of 4, image size of 640 pixels, and Adam optimizer.

Second, the TrajGATFormer worker trajectory prediction model. A summary of the model hyper-parameters is shown in Table 2. Regarding the d_{model}, its value follows the proposed value in [17], which is 512. However, for the Worker d_k value is 256, as proposed in several papers [50].

The loss function used in the model is the L2 loss between the $(x_{\text{pred}}, y_{\text{pred}})$ coordinates predicted by the model and the real $(x_{\text{real}}, y_{\text{real}})$ coordinates. Table 3 contains the optimizer parameters. The Adam optimizer was chosen for this model with 4000 warm-up steps for the learning rate; after that, it starts to decay. The learning rate equation used in the proposed model is as follows:

$$\text{Learning Rate} = \text{factor} \times \frac{1}{\sqrt{d_{\text{model}}}} \times \min\left(\frac{1}{\sqrt{\text{step number}}}, \frac{\text{step number}}{\text{warmup steps}^{1.5}}\right) \tag{14}$$

Similarly to what was proposed in [17], β_1 and β_2 are 0.9 and 0.98, respectively as values for the optimizer. The training of the models was done in 600 epochs and the batch size used is 4, as the GPU runs out of memory if the batch size is increased. A summary of TrajGATFormer and TrajGATFormer-Obstacle optimizer parameters is shown in Table 3.

Transfer learning was investigated to see whether it improves the performance of the model or not. Two models were produced sharing the same model architecture and vary by training process. The first model was initially trained using the ETH dataset, in order to initialize the parameter weights, resulting in a base model. Then retrained using the construction dataset. The second model was trained using only the construction dataset without applying transfer learning. The first model will use the trained parameters of the base model to initialize its weights; no weight randomization is needed.

Third, the TrajGATFormer-Obstacle worker and obstacle trajectory prediction model. A summary of the model parameters used is shown in Table 4. The training of the models was done in 600 epochs, and the batch size used is 3 for workers and 1 for panels. For the observation and prediction window, it is commonly used in the literature to use an observation length of 8 frames and predict the future 12 frames.

Table 3
Proposed models (TrajGATFormer and TrajGATFormer-Obstacle) optimizer parameters.

Optimizer parameter	Base model	Proposed models
Optimizer	Adam	Adam
Initial Learning Rate	0	0
Learning Rate Warm-up Steps	4000	4000
Scaling Factor	1	1
β_1	0.9	0.9
β_2	0.98	0.98
ϵ	1×10^{-9}	1×10^{-9}
Batch Size	32	4

Table 4
TrajGATFormer-Obstacle model parameters.

Model parameter	Value
Worker Encoder Layers	1
Panel Encoder Layers	1
Decoder Layers	1
d_{model}	512
Attention Heads	8
Worker d_k	256
Dropout	0.1

4. Datasets

In this paper, two datasets namely ETH and construction datasets were used. Each dataset is split into three subsets, training, validation, and testing in an 80/10/10 split. The first dataset described in the following section will be used to train a base model to initialize the model weights. Then a new construction dataset is created to be used to fine-tune the base model using transfer learning.

4.1. ETH dataset

The name "ETH" was derived from ETH Zurich [51], this dataset contains two scenes in a birds eye view of RGB cameras that are usually used for surveillance purposes: (1) Eth; and (2) Hotel. ETH scene was taken from the top of the ETH main building and the annotation was done at 2.5 frames per second (fps), which means that one frame is annotated every 0.4 s. This part consists of a total of 365 different pedestrian trajectories. For Hotel, the scene was taken from the 4th floor of a hotel in Bahanhofstr in Zurich and the video was taken at a 25 fps but the annotation was done at 2.5 fps for consistency. The image resolution of both datasets is 640 × 640 pixels.

4.2. Construction dataset

The second dataset used in this paper is a real-world construction dataset. The scenes were taken from a surveillance camera inside an off-site construction facility in Edmonton, Alberta. The construction dataset was temporally downsampled to 2.5 fps during preprocessing to match the ETH dataset, resulting in a prediction setup of 8 observed frames and 12 predicted frames corresponding to a 4.8-second prediction horizon. During training phases, the construction dataset is used, which contains two scenes from the same work station but working on different parts. In the first scene, 1907 frames were annotated with up to 3 workers and 1 moving obstacle. In the second scene, 409 frames were annotated with up to 3 workers and 1 moving obstacle. The first scene was used to train and test TrajGATFormer, and the second scene is used in addition to the first scene to train and test TrajGATFormer-Obstacle. As the first scene was focused mainly on the workers' movement in the factory, which is the focus of TrajGATFormer, and the second scene we tried to focus more on video portions where the obstacle is actually moving to add more data for the model to learn.

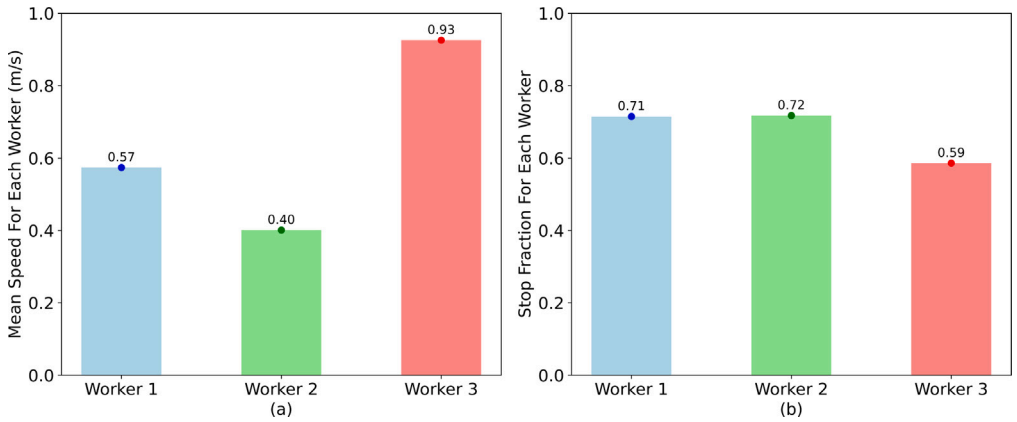

Fig. 4. Statistical analysis of the construction dataset.

4.3. Statistical analysis

To better understand the proposed construction dataset, a statistical analysis of the recorded trajectories is presented. This analysis is performed on scene 1 from the construction dataset. To analyze the dataset two factors were considered: (1) Workers mean walking speed, and (2) workers stop fraction. The mean walking speed of a trajectory is calculated by averaging the speed between consecutive points. The speed between two consecutive points is given by the Euclidean distance between those points, multiplied by fps. For a trajectory with n points, the mean speed can be calculated as follows:

$$v_{\text{mean}} = \frac{1}{n-1} \sum_{i=1}^{n-1} \|\mathbf{p}_{i+1} - \mathbf{p}_i\| \cdot \text{fps}$$

Where v_{mean} is the mean speed, $\mathbf{p}_i$ is the position of the ith point in the trajectory, f_{rate} is the frame rate (frames per second), $\|\mathbf{p}_{i+1} - \mathbf{p}_i\|$ is the Euclidean distance between consecutive points. The stop fraction is the proportion of the total time during which the speed is considered low and the worker is deemed as stopped. To categorize whether the person is stopped researches considered using thresholds as seen in [52], where the author used a threshold of 0.5 m/s for walking pedestrians. However, this might not be the case for walking workers in an offsite construction facility as they normally work in a set of working stations, which might reduce the overall walking speed and consonantly the stopping threshold. Therefore, we decided to use a bit lower threshold than pedestrians and used a threshold of 0.4 m/s. The stop fraction is calculated as:

$$\text{Stop Fraction} = \frac{\sum_{i=1}^{n-1} \mathbb{1}(\|\mathbf{p}_{i+1} - \mathbf{p}_i\| \cdot \text{fps} < \text{thresh})}{n-1}$$

Wher:

- $\mathbb{1}(\cdot)$ is the indicator function that is 1 if the condition inside is true (speed is below the stop threshold), and 0 otherwise.

- thresh is the threshold speed below which the worker is considered to have stopped.

As we can see from Fig. 4, the workers mean walking speed is plotted on (a) and the workers stop fraction is plotted on (b). From this figure assuming a threshold of 0.4 m/s worker 1 and 2 are mostly not moving during the recorded period. While worker 3 has been actively walking with a mean walking speed of 0.93 m/s. This is shown in the stop fraction figure, where workers 1 and 2 are considered stopped for more than 70% of the time, while worker 3 is considered stopped for around 59% of the time. Despite these results, it actively mirrors the real-world scenario where the workers are spending most of their time working in the working station.

5. Results and discussion

The results of the proposed TrajGATFormer and TrajGATFormer-Obstacle worker and/or obstacle trajectory prediction models are presented in this section.

5.1. Object detection and tracking results

The detection model was able to achieve high detection results. The results are 99.39% for precision, 99.4% for recall, 93.7% for mean Average Precision (mAP), and 99.4% for mean Average Precision at 50 (mAP50). For training, their are three training losses that was monitored: (1) Box loss, (2) Class loss, and (3) DFL loss. Despite the high performance of the detection model, the model faces challenges when exposed to drastically different environment. There are some reasons that can justify this, as when applied to different working environment several factors change, such as lighting, recording equipment, camera location.

5.2. Training results of TrajGATFormer and TrajGATFormer-obstacle

The training results of TrajGATFormer with transfer learning show that the model in the early epoch exhibits an over-fitting signs; however, after letting the model train for more epochs the model performance improved. The comparison between TrajGATFormer with transfer learning and without transfer learning is shown in Fig. 5. It can be seen from the figure that training with transfer learning starts with a lower training loss compared to TrajGATFormer trained without transfer learning. However, TrajGATFormer trained without transfer learning was able to converge very fast within a few epochs and followed the training loss of the other model. From this figure, we can see that the model with transfer learning resulted in overall lower training error.

In TrajGATFormer-Obstacle, transfer learning was applied to the worker encoder by reusing pretrained weights from TrajGATFormer. The obstacle encoder was trained from scratch, as it represents a newly introduced module with different input characteristics and no pretrained counterpart".

5.3. Performance evaluation of TrajGATFormer and TrajGATFormer-obstacle

The performance of the proposed models is evaluated against the baseline methods using identical experimental settings. All models use the same detection, tracking, and preprocessing pipeline based on YOLOv10n and DeepSORT, ensuring consistent trajectory inputs across methods. The observation window is fixed at 8 frames, and the prediction horizon is 12 frames. Training is conducted for 600 epochs

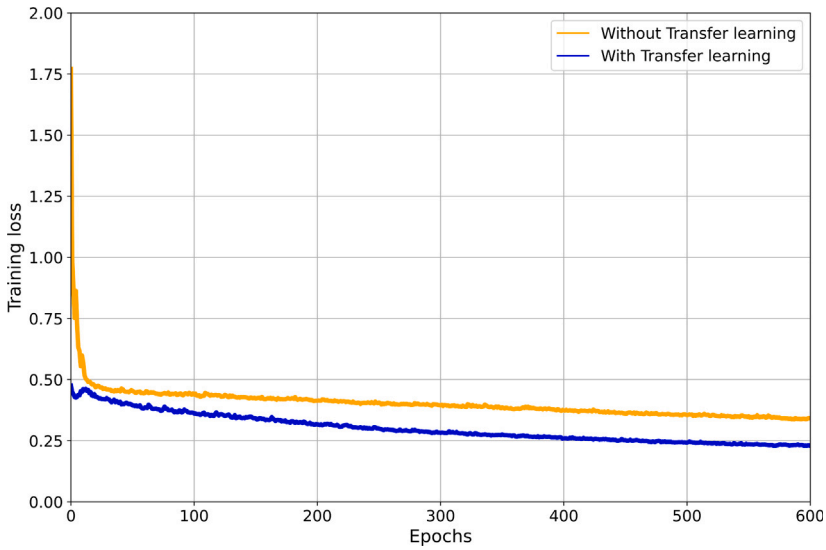

Fig. 5. Training loss curves comparison between training with transfer learning and without for TrajGATFormer.

with a batch size of 4. This setup ensures that performance differences arise from the trajectory prediction models themselves rather than variations in data preprocessing.

Table 5, shows a summary of the worker trajectory results from all models. While Table 6, shows the trajectory predictions of the moving obstacle in our case a moving panel. The presented results showed that TrajGATFormer outperformed all baseline models with an improvement percentage range of approximately [27% – 31%] for ADE and a range of approximately [26% – 29%] for FDE. However, TrajGATFormer-Obstacle results showed a significant improvement compared to baseline models, up to more than 40% improvement in ADE and 38% in FDE compared to LSTM. In addition, when comparing TrajGATFormer-Obstacle and TrajGATFormer, the results showed an improvement. TrajGATFormer-Obstacle has an ADE of 1.03, while TrajGATFormer has an ADE of 1.25. This represents an improvement of about 17.6% in TrajGATFormer-Obstacle, which means it is more accurate in predicting the overall trajectory. TrajGATFormer-Obstacle has an FDE of 2.06, compared to 2.35 for TrajGATFormer. This results in an improvement of around 12.34% in terms of the accuracy of the final prediction. To further assess the model's capacity to generalize beyond the exact training trajectories, we conducted additional experiments using trajectory-level data augmentation that introduce spatial perturbations, scaling, and rotation while preserving interaction structure. When retrained using the augmented dataset, the model achieved improved performance (ADE = 1.31, FDE = 1.40), demonstrating stable behavior and improved robustness under increased trajectory variability. The comparison between the original and augmented training results indicates that the model does not collapse or degrade when exposed to distributional variation, suggesting that it does not rely on memorization of specific scene geometry or motion patterns. Instead, the performance gains observed in both settings support the claim that the proposed framework learns transferable representations of motion and interaction. While broader validation on multi-site datasets would further strengthen empirical generalization claims, the consistency of performance improvements across both original and augmented training setups provides evidence that the proposed framework has the capacity to generalize and should be interpreted as a scalable proof of concept rather than a scene-specific solution. Despite the overall performance gains, Table 6 shows that obstacle trajectory prediction remains less accurate than worker prediction. This performance gap does not indicate a bias of the model toward worker trajectories, as worker and obstacle motions are encoded independently using two dedicated transformer encoders. As defined in Eq. (11), the obstacle

Table 5
Results for worker trajectories.

Model name	ADE	FDE
LSTM	1.753	3.332
SGAN	1.767	3.343
EA-Distance	1.825	3.349
EA-Direction	1.734	3.201
TrajGATFormer	1.250	2.350
TrajGATFormer-Obstacle	1.030	2.060

Table 6
Results for panel trajectories.

Model name	ADE	FDE
TrajGATFormer-Obstacle	1.630	3.790

encoder output $\hat{h}^{(L)}_{\text{obstacle,encoder}}$ is concatenated with the worker representation to form the shared latent representation passed to the decoder, allowing joint prediction while preserving modality-specific features. The lower accuracy for panel prediction is primarily attributed to data-related factors rather than architectural limitations, including the smaller number of obstacle motion instances and the intermittent, stop–and–go movement patterns of panels compared to workers.

Fig. 6, compares the predicted trajectories between TrajGATFormer and TrajGATFormer-Obstacle. The predicted trajectories are shown for the same set of frames (i.e., they share the same frame for each trajectory prediction point). The results showed improvement for all workers. However, it showed that the TrajGATFormer-Obstacle results follow the general direction of the ground-truth but have shorter trajectories. This can be seen in the trajectories of the worker on the left side. Fig. 7, shows a comparison between a prediction made by TrajGATFormer for a time stamp and the corresponding ground-truth trajectories for long trajectories with trajectories in the form of real-world coordinates. The results demonstrate that the model effectively captures the general movement patterns of workers, even in instances of irregular motion, such as the worker on the left side. However, in some cases, the predicted trajectory lengths deviate from the ground-truth. This discrepancy may stem from the dataset's inherent structure. Unlike pedestrians, workers exhibit distinct movement patterns, often pausing for extended periods to complete tasks before proceeding to their next location.

Furthermore, Fig. 7 illustrates the actual trajectories of three workers and one moving panel, along with their predicted trajectories and

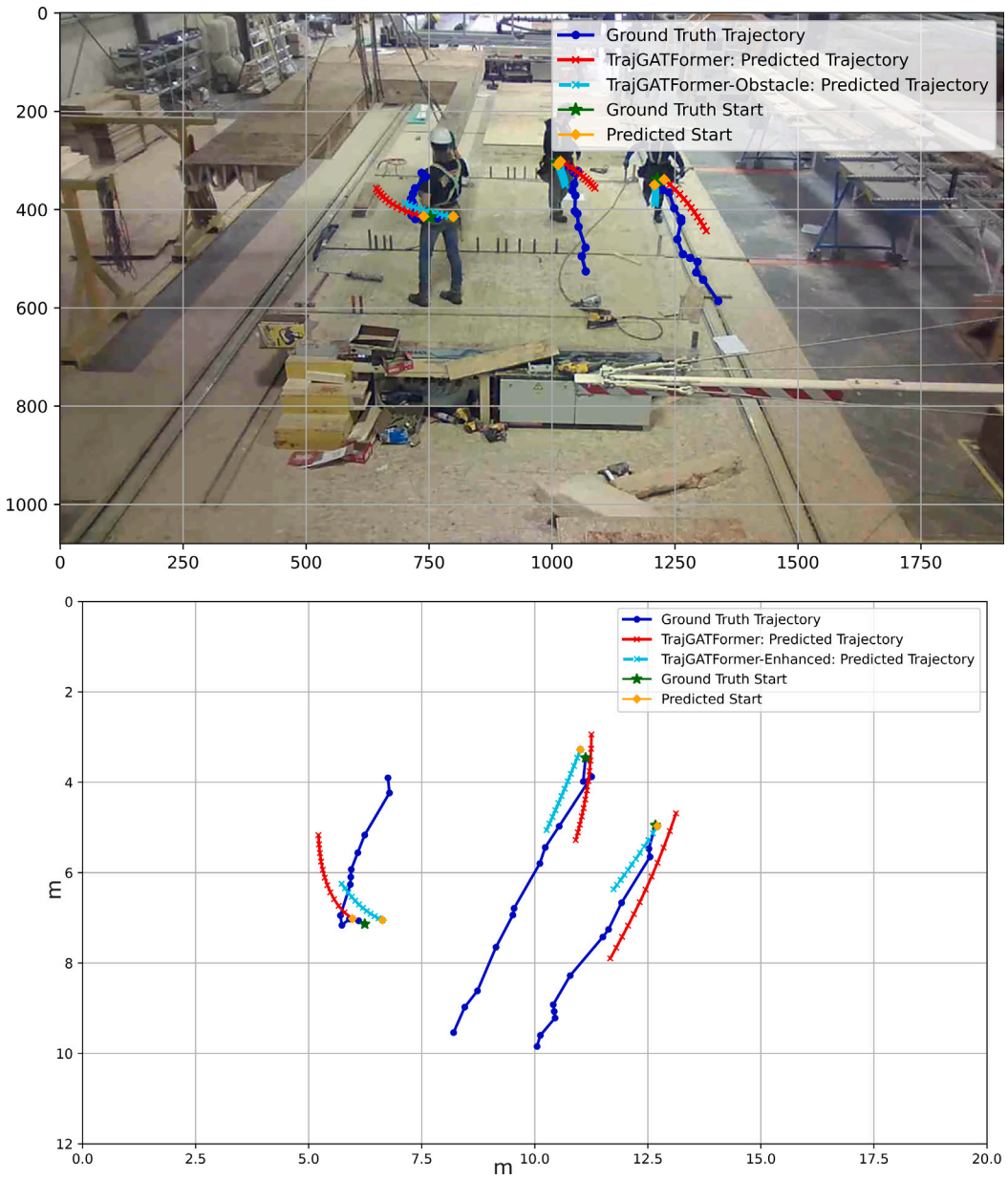

Fig. 6. Comparison of single-sample trajectory prediction results produced by TrajGATFormer and TrajGATFormer-Obstacle.

their corresponding real-world coordinates. The overall performance of TrajGATFormer-Obstacle is better compared to TrajGATFormer. However, the model still suffers from long trajectories, as seen in Fig. 8. It is clear that the model can predict the direction of the movement accurately, but it cannot predict the full length of the ground truth, as shown in the trajectories of workers in the middle and right sides. In addition, the model still suffers from the sudden or sharp turns of the workers, as shown in the trajectory of the worker on the left side. For the panel trajectory, the model can slightly follow the length and direction of long step movements. However, the trajectories are much worse than those of the workers, as shown in Table 6. For panels, the poor performance can be explained by the limited annotated data that includes moving objects. However, when checking the performance of the models for short step movements, the models were able to accurately predict the direction and length of the trajectories as shown in Fig. 9. In addition, the trajectory of the worker in Fig. 10 shows that the ground truth trajectory has a sudden turn, which caused the poor performance of the predicted trajectory, even though in the earlier steps, the predicted trajectories follow the ground truth until the sudden turn.

6. Conclusions

In conclusion, to mitigate hazards in offsite construction where workers operate in crowded environments with both static and dynamic obstacles, accurate trajectory prediction is essential for developing effective struck-by alarm systems. Traditional approaches have largely overlooked the dynamic nature of obstacles, limiting their ability to capture workers movements. In response, this paper proposes two trajectory prediction models, TrajGATFormer and TrajGATFormer-Obstacle. The first part of the work focused on the development of TrajGATFormer, which demonstrated improved prediction of worker trajectories using transformer-based layers and GATs. This model effectively captured the temporal and spatial dependencies of worker movements, significantly reducing prediction errors compared to traditional models. TrajGATFormer achieved a reduction in ADE by 27.92% to 31.51% and FDE by 26.57% to 29.84% compared to models like

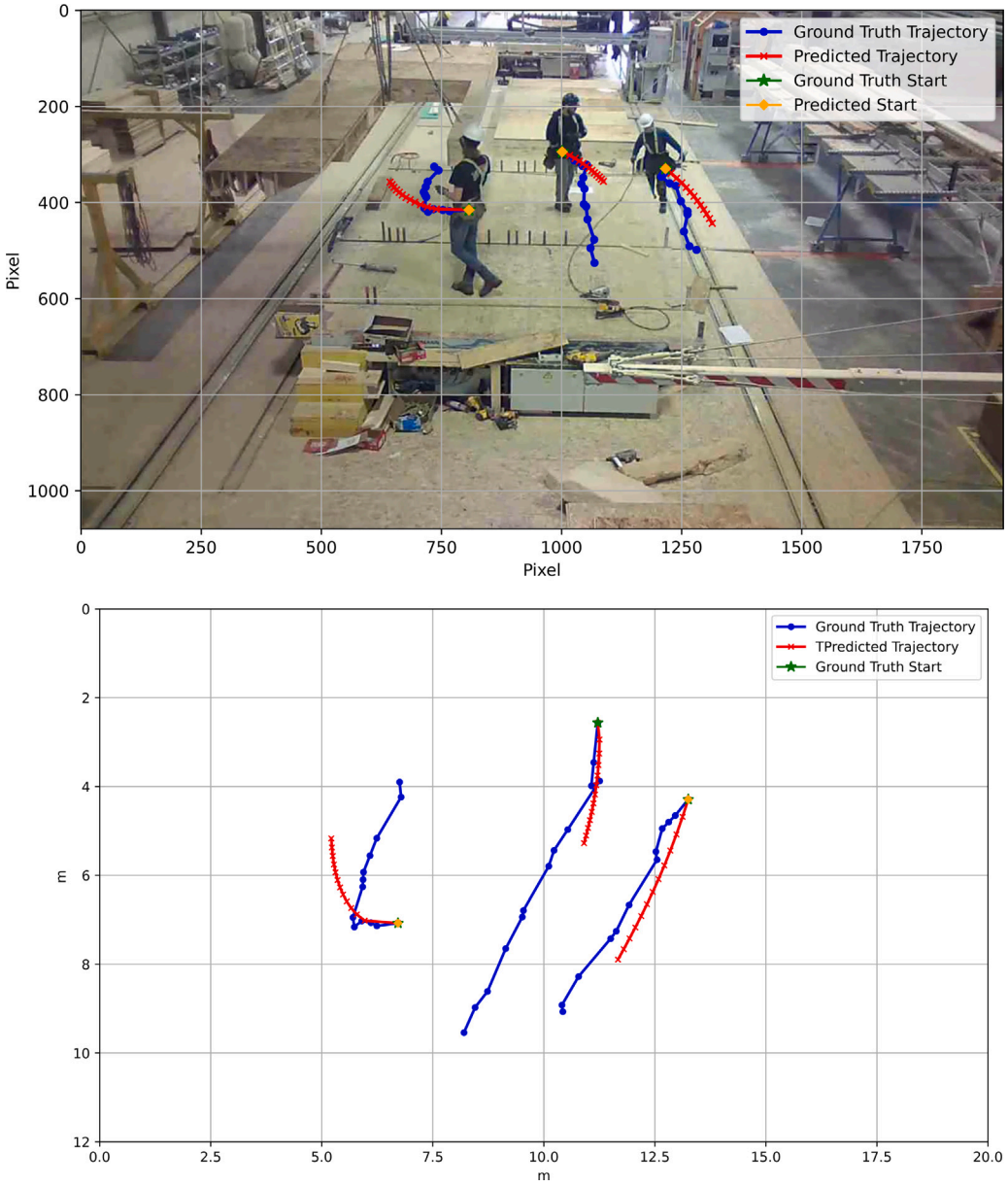

Fig. 7. Comparison between a TrajGATFormer prediction at a representative timestamp and the corresponding ground-truth long trajectories.

LSTM and SGAN. The second part extended the initial framework to include both workers and obstacles with TrajGATFormer-Obstacle. By incorporating an additional transformer encoder for obstacle tracking, this model further improved accuracy in predicting interactions between workers and dynamic objects. TrajGATFormer-Obstacle achieved a reduction in ADE by 17.6% and FDE by 12.34% compared to TrajGATFormer. While the model demonstrates significant improvements in handling short-term trajectories, performance declines for long trajectories and abrupt movement changes, highlighting the need for more diverse training data and adaptive learning strategies. Future work should focus on addressing these challenges to further refine prediction accuracy and robustness in complex construction scenarios.

7. Limitations and future direction

This research, while successful in developing an advanced framework for trajectory prediction, faced several limitations that provide avenues for future improvement. First, the dataset used for model training was limited in both scope and diversity, particularly with regard to obstacles types, worker interactions, and the working environment. This constraint may impact the model's performance when deployed in highly varied or complex scenarios, where prediction accuracy for rare or complex interactions is essential. Second, while the models performed exceptionally well in short-term trajectory prediction, their predictive accuracy decreases in long-term scenarios or when workers exhibit sudden, unpredictable changes in movement or prolonged inactivity. Third, the framework relies on single-camera tracking, which is vulnerable to occlusions and ID switches in multi-worker scenes. Such issues were observed during experiments and may propagate errors into trajectory prediction.

To build on the advancements made in this research, several promising directions for future work are identified. First, expanding data collection efforts to cover a wider range of construction sites, obstacle types (e.g., forklifts, cranes, handcarts), and interaction scenarios would significantly improve the adaptability of the model in different

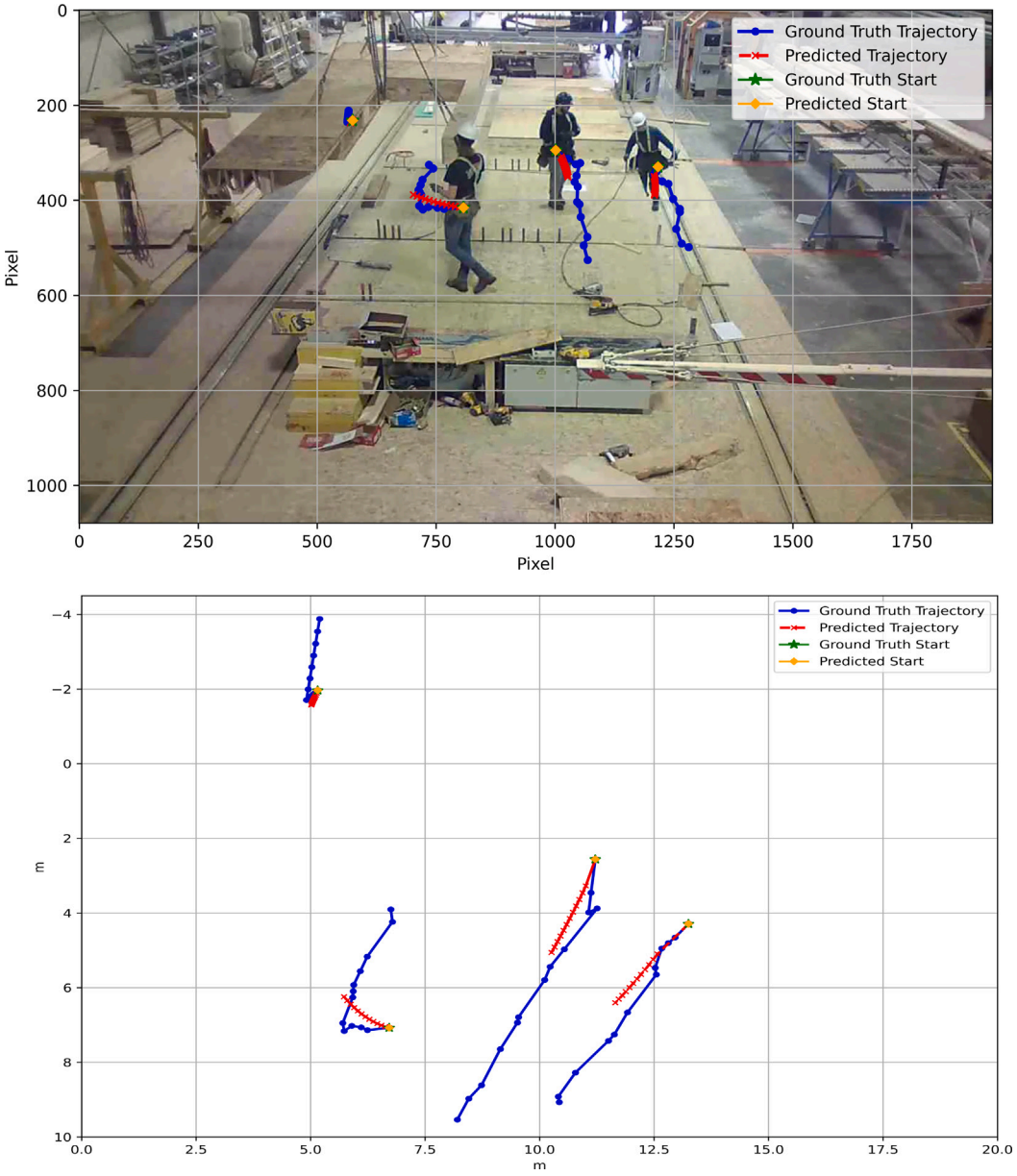

Fig. 8. Comparison between a TrajGATFormer-Obstacles prediction at a representative timestamp and the corresponding ground-truth long trajectories.

environments, especially when dealing with unusual but high-risk interactions. Second, implementing the framework on live construction sites would provide valuable practical insights and allow real-world testing of the performance, robustness, and responsiveness of the model, particularly when faced with real-time constraints and hardware limitations. Another important future direction is the integration of the trajectory prediction framework with safety-oriented risk assessment modules. In such extensions, predicted worker and obstacle trajectories could be combined with rule-based or learning-based mechanisms to generate collision warnings or safety alerts using metrics such as proximity thresholds, time-to-collision, or probabilistic risk scores, enabling proactive safety interventions. In addition, the prediction model can be strengthened by incorporating construction-specific contextual inputs, such as distance-to-obstacle features, destination (goal) intent, and hazard-zone awareness, to produce trajectories that better reflect site constraints and safety-critical interactions. Furthermore, multi-modal prediction can be explored to generate multiple plausible future paths, which is particularly important when worker motion is inherently ambiguous and several feasible routes exist around obstacles or hazard zones.

CRediT authorship contribution statement

Mohammed Alduais: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation. **Xinming Li:** Writing – review & editing, Supervision, Funding acquisition, Data curation, Conceptualization. **Qipei Mei:** Writing – review & editing, Supervision, Methodology, Funding acquisition, Conceptualization.

Funding

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) through the Alliance Grants with Alberta Innovates [File No. ALLRP 561120 - 20; ALLRP 576826 - 22].

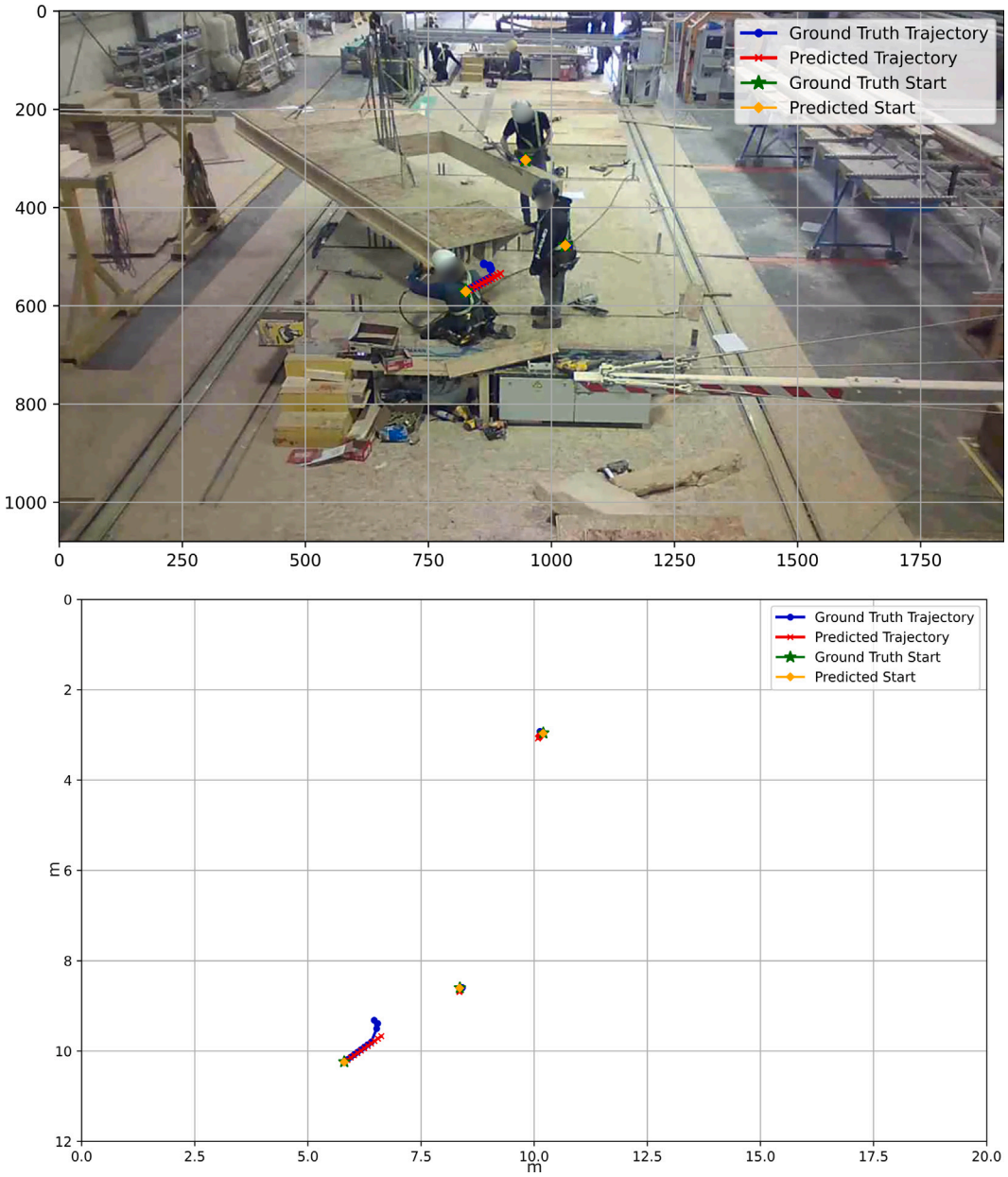

Fig. 9. Predicted vs. ground-truth short trajectories for TrajGATFormer.

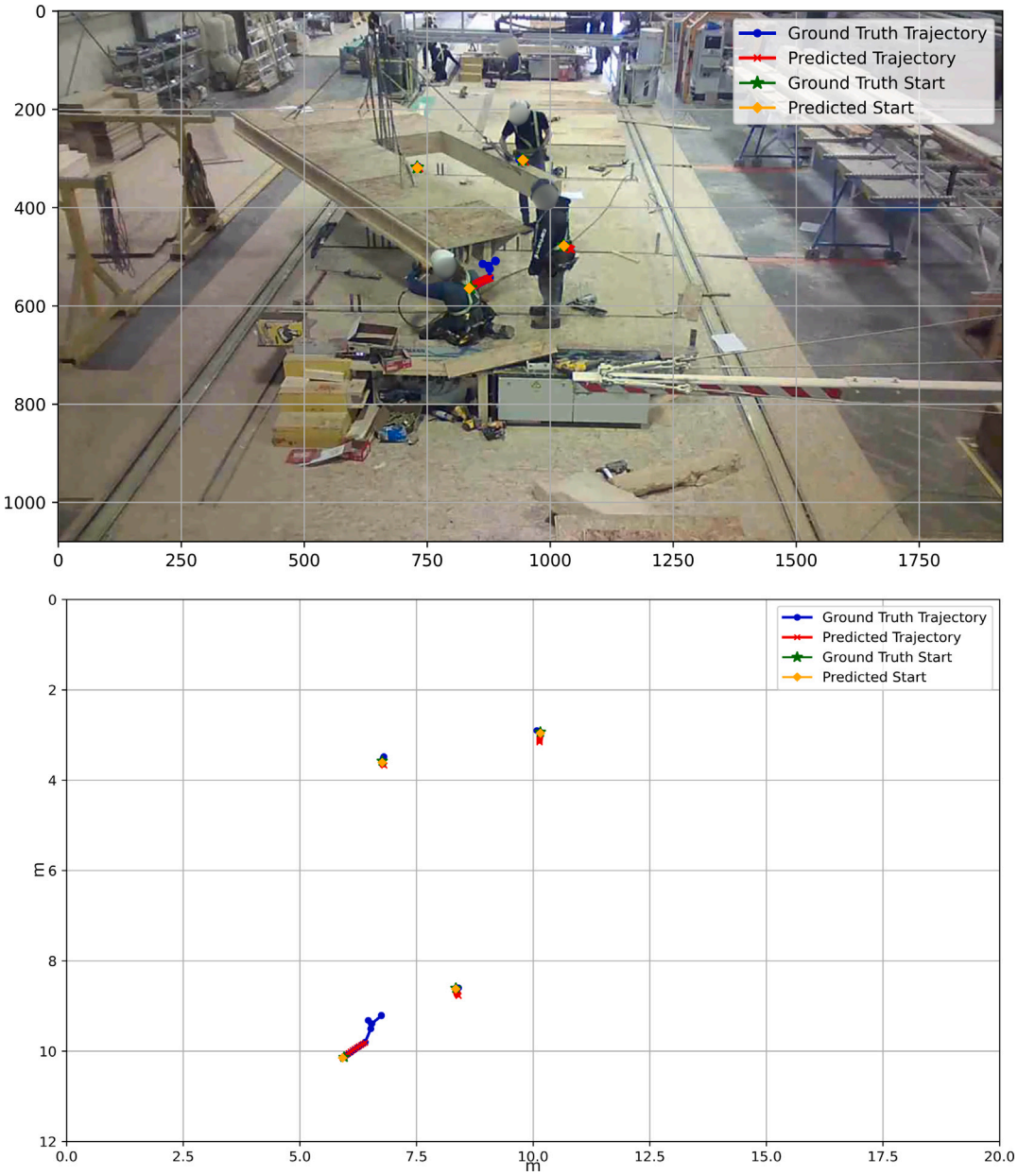

Fig. 10. Predicted vs. ground-truth short trajectories for TrajGATFormer-Obstacles.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] Oxford Economics, The future of construction: A global forecast for construction to 2030, 2021.

[2] W. Harris, T. Yohannes, A.B. Trueblood, Data bulletin: Focus four injuries in construction, 2023, (Accessed 18 June 2024).

[3] American Industrial Hygiene Association, Focus four for health: An initiative to address four major construction health hazards, 2019, Guidance Document, Last Updated 2019. URL https://www.aiha.org.

[4] R.M. Lawson, P.J. Grubb, J. Prewer, P.J. Trebilcock, Modular construction using light steel framing: An architect's guide, The Steel Construction Institute, Silwood Park, Ascot, Berkshire SL5 7QN, 1999.

[5] Y. Yu, Z. Chen, Rigidity of corrugated plate sidewalls and its effect on the modular structural design, Eng. Struct. 175 (2018) 191–200.

[6] D. Stern, Steel-framed modular construction for high-rise hotels, 2017, (Accessed 18 June 2023).

[7] R. Jin, J. Hong, J. Zuo, Environmental performance of off-site constructed facilities: A critical review, Energy Build. 207 (2020) 109567.

[8] N. Bertram, S. Fuchs, J. Mischke, R. Palter, G. Strube, J. Woetzel, Modular construction: From projects to products, 2019.

[9] J. Teizer, T. Cheng, Proximity hazard indicator for workers-on-foot near miss interactions with construction equipment and geo-referenced hazard areas, Autom. Constr. 60 (2015) 58–73, http://dx.doi.org/10.1016/j.autcon.2015.09.003.

[10] R. Korbmacher, A. Tordeux, Review of pedestrian trajectory prediction methods: Comparing deep learning and knowledge-based approaches, 2022, ArXiv Preprint ArXiv:2111.06740v2. URL https://arxiv.org/abs/2111.06740v2.

[11] Q. Yang, Q. Mei, C. Fan, M. Ma, X. Li, Environment-aware worker trajectory prediction using surveillance camera in modular construction facilities, Buildings 13 (6) (2023) http://dx.doi.org/10.3390/buildings13061502, URL https://www.mdpi.com/2075-5309/13/6/1502.

[12] J. Cai, Y. Zhang, L. Yang, H. Cai, S. Li, A context-augmented deep learning approach for worker trajectory prediction on unstructured and dynamic construction sites, Adv. Eng. Inform. 46 (2020) 101173, http://dx.doi.org/10.1016/j.aei.2020.101173.
[13] R.E. Kalman, A New Approach to Linear Filtering and Prediction Problems, J. Basic Eng. 82 (1) (1960) 35–45, http://dx.doi.org/10.1115/1.3662552.
[14] L. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proc. IEEE 77 (2) (1989) 257–286, http://dx.doi.org/10.1109/5.18626.
[15] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Comput. 9 (8) (1997) 1735–1780, http://dx.doi.org/10.1162/neco.1997.9.8.1735.
[16] C. Cai, Y. Tao, T. Zhu, Z. Deng, Short-term load forecasting based on deep learning bidirectional LSTM neural network, Appl. Sci. 11 (17) (2021) http://dx.doi.org/10.3390/app11178129, URL https://www.mdpi.com/2076-3417/11/17/8129.
[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017.
[18] L. Franco, L. Placidi, F. Giuliari, I. Hasan, M. Cristani, F. Galasso, Under the hood of transformer networks for trajectory forecasting, Pattern Recognit. 138 (2023) 109372, http://dx.doi.org/10.1016/j.patcog.2023.109372.
[19] F. Giuliari, I. Hasan, M. Cristani, F. Galasso, Transformer networks for trajectory forecasting, 2020, arXiv:2003.08111.
[20] Y. Huang, H. Bi, Z. Li, T. Mao, Z. Wang, STGAT: Modeling spatial-temporal interactions for human trajectory prediction, in: 2019 IEEE/CVF International Conference on Computer Vision, ICCV, 2019, pp. 6271–6280, http://dx.doi.org/10.1109/ICCV.2019.00637.
[21] A. Tordeux, M. Chraibi, A. Seyfried, A. Schadschneider, Prediction of pedestrian dynamics in complex architectures with artificial neural networks, J. Intell. Transp. Syst. 24 (6) (2020) 556–568, http://dx.doi.org/10.1080/15472450.2019.1621756.
[22] H. Cheng, F.T. Johora, M. Sester, J.P. Mueller, Trajectory modelling in shared spaces: Expert-based vs. Deep learning approach? in: Multi-Agent-Based Simulation XX: International Workshop, MABS 2020, 2020, pp. 1–12.
[23] T. Kruse, A.K. Pandey, R. Alami, A. Kirsch, Human aware robot navigation: A survey, Robot. Auton. Syst. (2013).
[24] A. Bighashdel, G. Dubbelman, A survey on path prediction techniques for vulnerable road users: From traditional to deep-learning approaches, in: 2019 IEEE Intelligent Transportation Systems Conference, ITSC, IEEE. IEEE, 2019, pp. 1039–1046.
[25] C. Schöller, V. Aravantinos, F. Lay, A. Knoll, What the constant velocity model can teach us about pedestrian motion prediction, 2020, arXiv:1903.07933.
[26] P. Karle, F. Török, M. Geisslinger, M. Lienkamp, MixNet: Physics constrained deep neural motion prediction for autonomous racing, IEEE Access 11 (2023) 85914–85926, http://dx.doi.org/10.1109/access.2023.3303841.
[27] A. Elnagar, Prediction of moving objects in dynamic environments using Kalman filters, in: Proceedings 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (Cat. No.01EX515), 2001, pp. 414–419, http://dx.doi.org/10.1109/CIRA.2001.1013236.
[28] N. Ye, Y. Zhang, R. Wang, Vehicle trajectory prediction based on hidden Markov model, KSII Trans. Internet Inf. Syst. 10 (2016) 3150–3170, http://dx.doi.org/10.3837/tiis.2016.07.016.
[29] D. Helbing, P. Molnár, Social force model for pedestrian dynamics, Phys. Rev. E 51 (5) (1995) 4282–4286, http://dx.doi.org/10.1103/physreve.51.4282.
[30] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, NIPS '12, Curran Associates Inc., Red Hook, NY, USA, 2012, pp. 1097–1105.
[31] N. Nikhil, B.T. Morris, Convolutional neural network for trajectory prediction, 2018, arXiv:1809.00696.
[32] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, S. Savarese, Social LSTM: Human trajectory prediction in crowded spaces, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2016, pp. 961–971, http://dx.doi.org/10.1109/CVPR.2016.110.
[33] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, 2017, arXiv preprint arXiv:1707.01926.
[34] Z. Cui, K. Henrickson, R. Ke, Y. Wang, High-order graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting, 2018, arXiv preprint arXiv:1802.07007.
[35] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2017, arXiv:1609.02907.
[36] P. Shaw, J. Uszkoreit, A. Vaswani, Self-attention with relative position representations, in: Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2018.
[37] C. Gulcehre, M. Denil, M. Malinowski, A. Razavi, R. Pascanu, K.M. Hermann, P. Battaglia, V. Bapst, D. Raposo, A. Santoro, N. de Freitas, Hyperbolic attention networks, 2018, arXiv preprint arXiv:1805.09786.
[38] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, S.H. Rezatofighi, S. Savarese, Social-BiGAT: Multimodal trajectory forecasting using bicycle-GAN and graph attention networks, 2019, arXiv:1907.03395.
[39] Z. Zhu, M.-W. Park, C. Koch, M. Soltani, A. Hammad, K. Davari, Predicting movements of onsite workers and mobile equipment for enhancing construction site safety, Autom. Constr. 68 (2016) 95–101, http://dx.doi.org/10.1016/j.autcon.2016.04.009.
[40] M. Wang, P. Wong, H. Luo, S. Kumar, V. Delhi, J. Cheng, Predicting safety hazards among construction workers and equipment using computer vision and deep learning techniques, in: ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction, vol. 36, IAARC Publications, 2019, pp. 399–406.
[41] L. Fan, L. Zhou, M. Wang, Optimizing data regeneration and storage with data dependency for cloud scientific workflow systems, Expert Syst. Appl. 238 (2024) 121984, http://dx.doi.org/10.1016/j.eswa.2023.121984.
[42] P. Duan, J. Zhou, Y. Qiao, P. Guo, Block-based construction worker trajectory prediction method driven by site risk, Autom. Constr. 167 (2024) 105721, http://dx.doi.org/10.1016/j.autcon.2024.105721.
[43] S. Bao, H. Bu, N. Rui, Computer vision–based recognition of spatial relevance between engineering vehicles and workers for monitoring struck-by accidents, J. Constr. Eng. Manag. 151 (11) (2025) 04025163, http://dx.doi.org/10.1061/JCEMD4.COENG-16432.
[44] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, G. Ding, YOLOv10: Real-time end-to-end object detection, 2024, arXiv preprint arXiv:2405.14458.
[45] N. Wojke, A. Bewley, D. Paulus, Simple online and realtime tracking with a deep association metric, in: 2017 IEEE International Conference on Image Processing, ICIP, IEEE, 2017, pp. 3645–3649, http://dx.doi.org/10.1109/ICIP.2017.8296962.
[46] Z. Zhang, A flexible new technique for camera calibration, IEEE Trans. Pattern Anal. Mach. Intell. 22 (11) (2000) 1330–1334, http://dx.doi.org/10.1109/34.888718.
[47] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, 2018, arXiv:1710.10903.
[48] V. Nair, G.E. Hinton, Rectified linear units improve restricted boltzmann machines, in: ICML 2010, 2010, pp. 807–814.
[49] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, A. Alahi, Social GAN: Socially acceptable trajectories with generative adversarial networks, 2018, arXiv:1803.10892.
[50] Y. Yuan, X. Weng, Y. Ou, K. Kitani, AgentFormer: Agent-aware transformers for socio-temporal multi-agent forecasting, 2021, arXiv:2103.14023.
[51] S. Pellegrini, A. Ess, K. Schindler, L. van Gool, You'll never walk alone: Modeling social behavior for multi-target tracking, in: 2009 IEEE 12th International Conference on Computer Vision, IEEE, 2009, pp. 261–268.
[52] K. Okamoto, A. Utsumi, T. Ikeda, H. Yamazoe, T. Miyashita, S. Abe, K. Takahashi, N. Hagita, Classification of pedestrian behavior in a shopping mall based on lrf and camera observations, 2011.