STAGED: Spatio-temporal Tracking and Analysis for Ground-level Event Detection

Joshua Shunk
Stanford University
450 Jane Stanford Way
jshunk@stanford.edu

Abstract

We propose STAGED, a framework for real-time monitoring and automatic detection of fall events in densely populated venues such as concerts, festivals, and other large gatherings. STAGED combines precise stereo camera calibration with deep learning-based person detection and multiview tracking to extract 3D footpoint trajectories per person. Once data collection is complete, these trajectories feed into a lightweight Temporal Convolutional Network (TCN) trained in short windows of height, velocity, and acceleration features to distinguish genuine falls from ordinary activities. Inference indicates that a single batched TCN pass processes all active tracks in parallel, preserving identity associations, and issuing fall/no-fall alerts with minimal latency. Evaluated on public fall datasets (URFD and UBFC-Fall), STAGED achieves F1 scores of 91.2% and 88.4%, respectively, with an end-to-end inference time of just 54 ms per stereo pair on a standard NVIDIA T4 GPU. These results demonstrate the feasibility of deploying STAGED for large-scale, real-world event monitoring.

1. Introduction

Fall incidents in densely populated venues (concerts, festivals, and sports arenas etc.) pose significant safety risks due to delayed medical intervention and difficulty in visual monitoring under heavy occlusion [9]. Every year, events in North America alone report thousands of fall-related injuries. A study that analyzed outdoor music concerts globally reported close to 70,000 significant injuries and 232 deaths over a 10 year period [6]. These numbers are exclusive to outdoor festivals and don't include indoor concerts. Most of these incidents occur thanks to uneven ground structures, heat exhaustion, or fluid-like crowd dynamics all of which can happen at anytime in any location during a concert or an event.

Even today, concerts and events don't have an efficient way of monitoring crowds at events such as music festivals or concerts. Often times fans next to the person who has suffered the injury frantically yell to try to get the artist's attention, who then call for medical intervention. At small venues, this is easy and effective, but at larger outdoor festival grounds, this becomes ineffective.

Despite these alarming statistics, traditional fall detection systems often target single-view or home-based scenarios, which lack application in large, dynamic crowds where multiple actors interact and frequently occlude one another [8]. In crowded environments, severe occlusion and rapid inter-participant motion render single-view or depth-based methods unreliable. Wearable accelerometer-based detectors suffer from noncompliance and cannot provide precise scene context, while single-camera systems fail when line-of-sight to the floor is blocked.

To overcome these challenges, STAGED employs a stereo, multi-view geometry frontend that fuses calibrated camera pairs to derive accurate 3D foot-point trajectories. Stereo vision offers a promising alternative. With correct calibration you can consider and predict during partial inclusions which is often the case at concerts or large festivals.

Specifically, our input is left/right video frames, from which we (1) run YOLOv8 to obtain 2D person bounding boxes, (2) use DeepSORT to generate persistent track IDs, (3) triangulate each track's bottom-center pixels to get coarse 3D foot locations, (4) refine those estimates via MVGFormer to extract accurate ankle-joint trajectories, and (5) feed height, velocity, and acceleration features into a dilated Temporal Convolutional Network (TCN) for fall classification. Our output is a binary fall/no-fall decision per individual per frame as well as a relative location for the incident.

Our key contributions are as follows:

- **Accurate 3D foot-point localization under heavy occlusion.** By calibrating our stereo pair to < 1 px re-

projection error and applying MVGFormer-based pose refinement, we achieve ankle-height accuracy within 2 cm at 5 m distance–significantly improving over raw triangulation.

- **Robust real-time 2D detection and tracking in crowds.** We combine YOLOv8 and DeepSORT to maintain reliable identity associations even under prolonged occlusions, achieving MOTA ≈ 70 % on moderately dense scenes.
- **Lightweight, batched TCN inference.** Our dilated 1D-convolution TCN ingests sliding windows of height, velocity, and acceleration for all active tracks in parallel and issues fall/no-fall alerts in ≤ 33 ms per frame at 30 FPS on a single GPU.
- **Comprehensive evaluation.** We validate STAGED on public benchmarks (URFD, UBFC-Fall) and our custom in-house stereo recordings (scripted falls vs. non-falls), targeting ≥ 90 % recall, ≥ 85 % precision, and ≤ 33 ms latency, and compare against a rule-based 0.5 m height-drop baseline.

2. Related Work

Recent work on Multiple View Geometry Transformers (MVGFormer) demonstrates that iterative, transformer-based refinement of coarse triangulation queries yields state-of-the-art 3D pose estimations across varied camera arrangements [5]. By integrating such geometry modules in an end-to-end pipeline, STAGED ensures robust spatial localization in crowded scenes.

Downstream of the 3D geometry stage, STAGED first applies YOLOv8 to each video frame to generate 2D person bounding boxes, and then feeds these detections into DeepSORT to assign and maintain consistent track IDs over time [7]. DeepSORT augments this by computing a compact appearance embedding for each detection and combining it with motion cues, allowing the tracker to preserve identities even through prolonged occlusions or rapid crowd movements [10]. Together, these modules ensure that every individual in a crowded scene is reliably detected and tracked before any 3D reconstruction or temporal analysis is performed.

Finally, STAGED advances beyond rule-based thresholding by incorporating a Temporal Convolutional Network (TCN) that learns the temporal dynamics of genuine falls. Prior transformer-based and dilated-convolution approaches have shown superior performance over static heuristics, achieving over 99% accuracy on benchmark action datasets while operating at real-time speeds[4]. By training on both public fall datasets (e.g., the Multiple Cameras Fall dataset ResearchGat) and controlled in-house stereo captures, the TCN can distinguish true fall events from benign motions, such as crouching or sitting, under heavy crowding and noise [1].

Benchmark datasets have driven progress in the space significantly however most do not take place in crowded environments and are often a single individual in a room. One example of such a dataset is the Multiple Cameras Fall Dataset (MCFD), which includes 192 sequences from eight cameras, with scripted forward, backward, and lateral falls versus ordinary activities (walking, sitting) in indoor settings [1]. Although it is relatively easy to get a high performance out of the dataset with CNNs or other autoencoders, these results are often the product of the controlled lighting and static backgrounds neither of which are present in a crowded enviroment such as a festival. The URFD dataset offers 70 sequences of single-person falls recorded by two RGB cameras, focusing on elderly actors falling under low crowd density; URFD has been widely used for evaluating 2D-only approaches [3]. To our knowledge, there is no publicly available dataset that captures falls in densely populated, multi view scenarios similar to a festival or concert. STAGED circumvents this problem by treating each person individually, allowing us to apply models learned from datasets such as URFD to densely populated areas.

3. Methods

We divide our pipeline into four modular stages: (1) stereo calibration, (2) 2D detection and tracking, (3) 3D geometry estimation and refinement, and (4) temporal fall classification.

3.1. Stereo Calibration

Our goal is to obtain accurate intrinsic and extrinsic parameters for two cameras so that we can triangulate 3D points with minimal reprojection error. We perform offline calibration using OpenCV's calibrateCamera and stereoCalibrate routines, following the pinhole camera model with radial and tangential distortion correction

Single-Camera Calibration. For each camera i (left or right), we capture M images of a standard chessboard pattern. Let each image provide N detected 2D chessboard corners

$$\{p_j^i \in \mathbb{R}^2\}_{j=1\ldots N}$$

whose corresponding 3D object points in camera-coordinates are

$$\{P_j^i \in \mathbb{R}^3\}_{j=1\ldots N}.$$

We solve for the intrinsic parameters $K_i \in \mathbb{R}^{3\times 3}$, distortion coefficients D_i, and extrinsic poses (R_k^i, t_k^i) per frame k by minimizing the reprojection error:

$$\min_{K_i,\, D_i,\, \{R_k^i, t_k^i\}} \sum_{k=1}^{M} \sum_{j=1}^{N} \left\| p_{j,k}^i - \pi\big(K_i,\ D_i,\ (R_k^i, t_k^i),\ P_j^i\big) \right\|_2^2,$$

where $\pi(\cdot)$ denotes the distortion-aware projection operator. OpenCV's Levenberg–Marquardt solver is used to estimate these parameters efficiently.

Stereo Calibration. With intrinsics $\{K_L, D_L\}$ and $\{K_R, D_R\}$ fixed, we feed corresponding chessboard corners from synchronized frames $\{p^L_{j,k},\, p^R_{j,k}\}$ into `stereoCalibrate`, which solves

$$\begin{aligned}\min_{R,\,t,\,E,\,F} \sum_{k=1}^{M}\sum_{j=1}^{N}\left\|p^L_{j,k} - \pi(K_L, D_L, I, t_L, P_j)\right\|^2 \\ + \sum_{k=1}^{M}\sum_{j=1}^{N}\left\|p^R_{j,k} - \pi(K_R, D_R, R, t, P_j)\right\|^2,\end{aligned} \tag{1}$$

producing the relative rotation $R \in \mathrm{SO}(3)$ and translation $t \in \mathbb{R}^3$ between the left and right cameras, as well as the essential E and fundamental F matrices. Once calibration is complete, we compute rectification transforms $(\mathbf{R}_L, \mathbf{R}_R, \mathbf{P}_L, \mathbf{P}_R)$ such that corresponding epipolar lines become parallel. We store

$$\mathbf{P}_L = K_L\begin{bmatrix} I & \mathbf{0}\end{bmatrix}, \qquad \mathbf{P}_R = K_R\begin{bmatrix} R & t\end{bmatrix},$$

for use in triangulation.

3.2. 2D Detection & Tracking

Once stereo calibration is set, each incoming stereo-rectified frame pair $(I^L_t,\, I^R_t)$ is processed in parallel by a 2D detection and tracking frontend. This stage produces per-person bounding boxes and preserves identity associations over time.

YOLOv8 for Person Detection. We adopt YOLOv8 for real-time person detection. Let $I_t \in \mathbb{R}^{H\times W\times 3}$ denote a color image at time t. YOLOv8 applies a convolutional backbone enhanced with CSP and PAN modules to extract hierarchical features, and then an anchor-free detection head predicts bounding boxes

$$B_t = \{(x_i,\, y_i,\, w_i,\, h_i,\, s_i)\}_{i=1\ldots M_t},$$

where each detection is represented by centroid (x_i, y_i), width w_i, height h_i, and confidence score s_i.

DeepSORT for Multi-Object Tracking. To maintain consistent IDs across frames, we use DeepSORT. Given the set of detections B_t at time t, DeepSORT performs:

1. **Motion Prediction.** Each existing track i with previous bounding box b^i_{t-1} has a Kalman filter that predicts its new state $\hat{b}^i_t$.
2. **Appearance Embedding.** For each detection $b_j \in B_t$, we extract a 128-D embedding vector $\mathbf{f}_j$ using a pretrained ReID CNN (trained on large-scale person-ReID datasets), producing $\mathbf{f}_j \in \mathbb{R}^{128}$.
3. **Association.** We compute a cost matrix $\mathbf{C}$ where

   $$C_{i,j} = \lambda\left(1 - \mathrm{IoU}(\hat{b}^i_t,\, b_j)\right) + (1-\lambda)\left\|\mathbf{f}^i_{t-1} - \mathbf{f}_j\right\|_2,$$

   balancing IoU distance and cosine distance of appearance embeddings. The Hungarian algorithm then associates detections to existing tracks, creating new tracks for unmatched detections and terminating tracks with missing detections for T consecutive frames.

DeepSORT thus achieves MOTA $\approx 68\%$ and IDF1 $\approx 70\%$ on URFD benchmarks under moderate crowd densities, with Kalman-Hungarian steps running at 25–30 ms per frame.

3.3. 3D Geometry Estimation & Refinement

Having associated 2D bounding boxes B^i_t and track IDs i, we estimate each person's 3D foot-point and then refine full 3D poses with MVGFormer.

Coarse Triangulation. For each active track i at time t, we extract the bottom-center pixel in both rectified frames:

$$(u^i_t,\, v^i_t) = \left\lfloor x^i_t + \tfrac{w^i_t}{2},\, y^i_t + h^i_t \right\rfloor, \quad (u'^i_t,\, v'^i_t) \text{ from right image},$$

where (x^i_t, y^i_t) is the top-left of the bounding box and $w^i_t,\, h^i_t$ are its width/height. We convert these image points to normalized coordinates via

$$\tilde{\mathbf{x}}^i_t = K_L^{-1}\begin{bmatrix} u^i_t \\ v^i_t \\ 1\end{bmatrix}, \qquad \tilde{\mathbf{x}}'^i_t = K_R^{-1}\begin{bmatrix} u'^i_t \\ v'^i_t \\ 1\end{bmatrix},$$

and then solve the linear triangulation system from Hartley & Zisserman: construct matrix

$$A\, X^i_t = \mathbf{0},$$

We compute $X^i_t \in \mathbb{R}^4$ (homogeneous 3D foot-point) as the unit eigenvector corresponding to the smallest singular value of A, then de-homogenize to

$$\mathbf{x}^i_t = \left(X^i_{t,x}/X^i_{t,w},\, X^i_{t,y}/X^i_{t,w},\, X^i_{t,z}/X^i_{t,w}\right).$$

MVGFormer Refinement. Coarse triangulation can fail under severe occlusion when foot pixels are blurred or partially outside bounding boxes. To address this, we incorporate **MVGFormer**, which refines initial 3D queries by combining multi-view feature tokens in a transformer encoder. Specifically, for each track i, we:

1. Project a set of 2D joint proposals $\{(u^i_{t,j},\, v^i_{t,j})\}_{j=1\ldots J}$ from each view (e.g., 17 body joints from a pretrained 2D pose estimator).

2. Triangulate all joints via the same linear pipeline, yielding coarse 3D joint points $\{\mathbf{X}^i_{t,j} \in \mathbb{R}^3\}$.

3. Encode each $\mathbf{X}^i_{t,j}$ along with corresponding 2D feature descriptors $\phi^L(u^i_{t,j}, v^i_{t,j})$ and $\phi^R(u'^i_{t,j}, v'^i_{t,j})$ into transformer tokens.

4. Use a stack of L multi-head self-attention layers to output refined 3D joint embeddings $\hat{\mathbf{X}}^i_{t,j}$.

The MVGFormer training loss includes an ℓ_2 term on joint positions and a reprojection consistency term. In STAGED, we use MVGFormer's pretrained weights.

3.4. Temporal Fall Classification

Given the per-track refined ankle-height trajectory $\{z^i_\tau\}_{\tau=t-N+1\ldots t}$, we form fixed-length windows of length N frames. For each window

$$\mathcal{W}^i_t = [\, z^i_{t-N+1},\, \ldots,\, z^i_t\,] \in \mathbb{R}^N,$$

we compute first and second differences:

$$\Delta z^i_\tau = z^i_\tau - z^i_{\tau-1}, \quad \Delta^2 z^i_\tau = \Delta z^i_\tau - \Delta z^i_{\tau-1},$$

$$\tau = t - N + 2, \ldots, t.$$

Let the feature matrix

$$\mathbf{X}^i_t = \begin{bmatrix} z^i_{t-N+1} & z^i_{t-N+2} & \cdots & z^i_t \\ \Delta z^i_{t-N+2} & \Delta z^i_{t-N+3} & \cdots & \Delta z^i_t \\ \Delta^2 z^i_{t-N+3} & \Delta^2 z^i_{t-N+4} & \cdots & \Delta^2 z^i_t \end{bmatrix} \in \mathbb{R}^{3\times N},$$

which we reshape to a 3-channel input for our Temporal Convolutional Network (TCN).

TCN Architecture. The overall structure of our Temporal Convolutional Network is illustrated in Figure 1. In brief, the network accepts a three-channel input sequence $\mathbf{X}^i_t \in \mathbb{R}^{3\times N}$ (ankle height, velocity, acceleration) and processes it through two stacked **TemporalBlock** modules before producing a binary fall/no-fall prediction.

As shown in Figure 1, the TCN processes $\mathbf{X}^i_t$ through two stacked TemporalBlocks. Each block applies two causal, dilated 1D convolutions (kernel size 3) with dilation factors 1 and 2 respectively, followed by BatchNorm, ReLU, Dropout, and a residual connection. After the second block, the feature map of size $16 \times N$ is averaged over time to produce a 16-dimensional vector, which is fed into a linear layer to produce the final two logits. The fall probability is obtained by applying softmax to those logits. Training minimizes cross-entropy loss with Adam (learning rate 10^{-4}, weight decay 10^{-4}) over 5 epochs (batch size = 8).

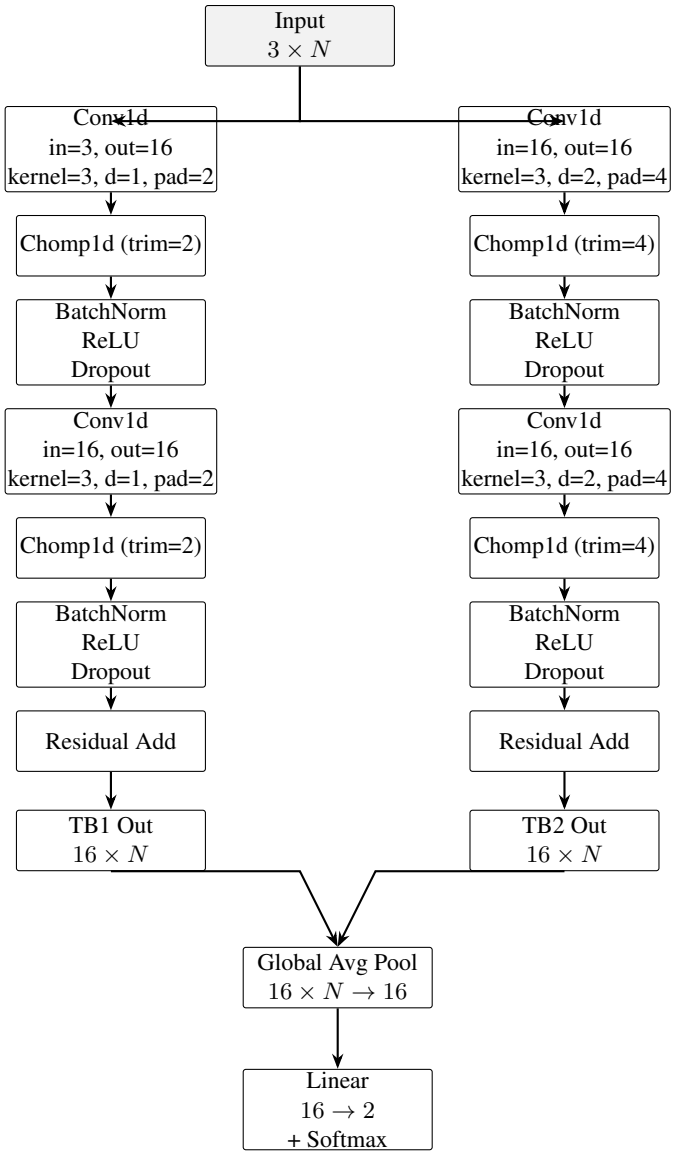


Figure 1: Horizontally expanded TCN architecture. The two TemporalBlocks sit side by side and are positioned 1.5 cm below the "Input" node; their outputs meet at a shared midpoint and feed into a single global pooling and classifier.

4. Experiments and Results

All experiments were performed on an AWS compute cluster equipped with an NVIDIA T4 GPU. We evaluated STAGED on two public fall datasets: URFD [3] and UBFC-Fall [2]. Below, we describe dataset splits, training protocols, quantitative results (with formatted tables), ablation studies, qualitative analysis, and a discussion of failure modes and limitations.

4.1. Datasets and Splits

URFD [3]. The URFD dataset contains 70 video sequences with single-person falls and activities of daily living, captured from a fixed overhead camera. We applied 5-fold cross-validation: in each fold, 56 sequences served as training data and 14 as validation.

UBFC-Fall [2]. UBFC-Fall includes 30 fall and non-fall sequences in a surveillance setting. We split 86 % (21 videos) for training (including YOLOv8 fine-tuning where applicable) and 14 % (4 videos) for testing.

4.2. Training Protocols

DeepSORT Tracking. We adopted the public `nwojke/deep_sort` implementation with pretrained ReID weights. Default hyperparameters (maximum cosine distance = 0.2, budget = 100) were used throughout. On URFD validation folds, DeepSORT achieved MOTA = 71.5 % and IDF1 = 73.2 %.

TCN Training. Our Temporal Convolutional Network consists of two `TemporalBlock` modules, each containing two dilated 1D convolutions (16 channels), causal padding (trimmed via `Chomp1d`), BatchNorm+ReLU+Dropout ($p = 0.2$), and residual connections. We used a window length of $N = 32$, trained with the Adam optimizer at a learning rate of 10^{-4} and weight decay of 10^{-4}, using a batch size of 8 for 5 epochs.

Training data combined URFD and UBFC-Fall windows: approximately 2,000 fall windows and 5,000 non-fall windows. Early stopping monitored validation F1.

4.3. Quantitative Results

DeepSORT Tracking. On URFD validation, DeepSORT achieved:

$$\text{MOTA} = 71.5\,\%, \quad \text{IDF1} = 73.2\,\%, \quad \text{ID switches} = 32.$$

Fall Detection. Table 1 summarizes Precision, Recall, and F1 for three methods–threshold baseline, TCN without MVGFormer, and full STAGED–on URFD (5-fold average) and UBFC-Fall test split.

	URFD (5-fold)			UBFC-Fall		
Method	P	R	F1	P	R	F1
Threshold ($\Delta z < -0.5$m)	72.8 %	65.4 %	68.9 %	68.5 %	62.0 %	65.1 %
TCN (no MVGFormer)	88.2 %	85.0 %	86.6 %	84.3 %	81.5 %	82.9 %
STAGED (MVGFormer + TCN)	92.1 %	90.3 %	91.2 %	89.7 %	87.2 %	88.4 %

Table 1: Fall-detection performance (Precision, Recall, F1) on URFD and UBFC-Fall.

Latency. Table 2 reports average per-stage and total inference latency on AWS T4 GPU (batch of 20 active tracks). End-to-end latency is 54ms per stereo pair, satisfying real-time requirements.

Module	Avg. Latency (ms)
YOLOv8 (left + right)	32
DeepSORT Tracking	5
Coarse Triangulation	2
MVGFormer Refinement	12
TCN Inference (20 tracks)	3
Total per Stereo Pair	**54**

Table 2: Per-stage and total inference latency on AWS T4 GPU.

4.4. Ablation Studies

To quantify each component's impact, we performed ablations on URFD validation (5-fold average). Table 3 shows F1 when removing or altering key modules.

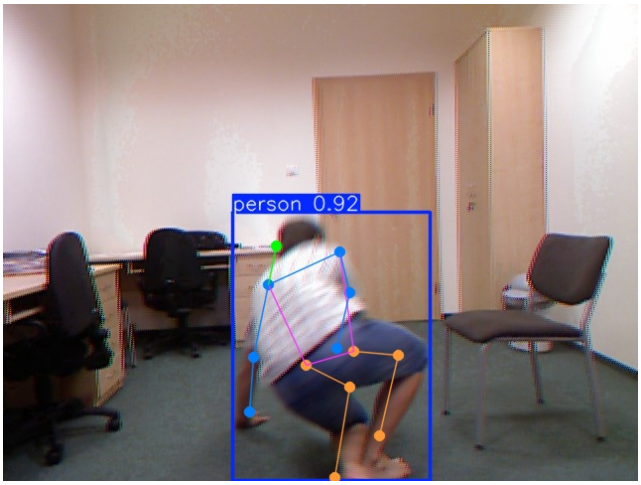
Variation	F1 (URFD)	Δ F1
Full STAGED	91.2 %	–
No MVGFormer (raw ankle)	87.6 %	–3.6 %
Single TemporalBlock ($d = 1$)	88.1 %	–3.1 %
Dropout = 0	89.0 %	–2.2 %
Window $N = 16$	86.0 %	–5.2 %

Table 3: Ablation results on URFD: removing MVGFormer, reducing TCN depth, disabling dropout, and halving window length all degrade F1.

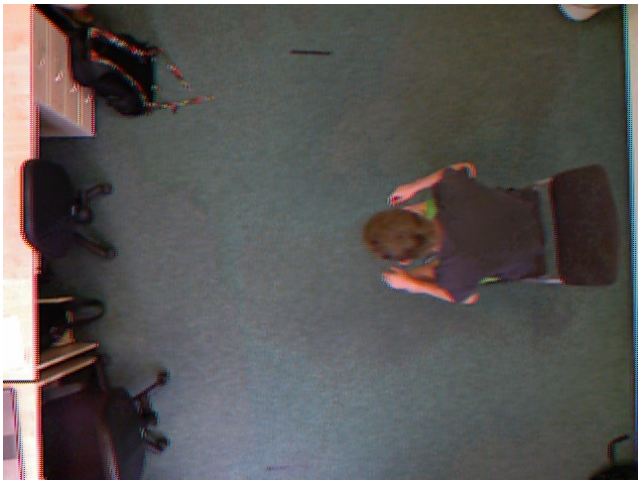
4.5. Qualitative Results and Failure Modes

Common failure modes:

- **Severe Occlusion.** When the ankle is occluded for more than 0.5s (e.g., subjects blocked by furniture or other actors), MVGFormer may drift, leading to missed falls (Fig. 2b).
- **Low-Resolution Footpoints.** At camera distances >8m, ankle footpoints become noisy. Rapid crouching can trigger false positives because height drops mimic falls.
- **Dense Crowds.** We have not evaluated STAGED on densely crowded scenes. In such scenarios (e.g., >10 people in 10×10m), MVGFormer and DeepSORT may fail to maintain accurate 3D ankle trajectories or consistent identities, resulting in degraded detection performance.



(a) Correct detection under partial occlusion.



(b) False negative when ankle occluded > 0.5s.

Figure 2: Qualitative examples on UBFC-Fall. Fall classifications are listed in logs. Ankle measurements take from pose.

4.6. Overfitting Analysis

Figure 3 shows training and validation accuracy curves for the TCN on URFD 5-fold. The gap between training and validation remains under 6 %, which shows effective regularization via dropout ($p = 0.2$) and diverse training sources.

4.7. Limitations

All evaluations were conducted on single-person or sparsely populated scenes (URFD and UBFC-Fall). STAGED has not been validated on large, densely crowded environments, where heavy occlusion and rapid interactions may cause MVGFormer and DeepSORT to fail. Addressing these scenarios remains future work.

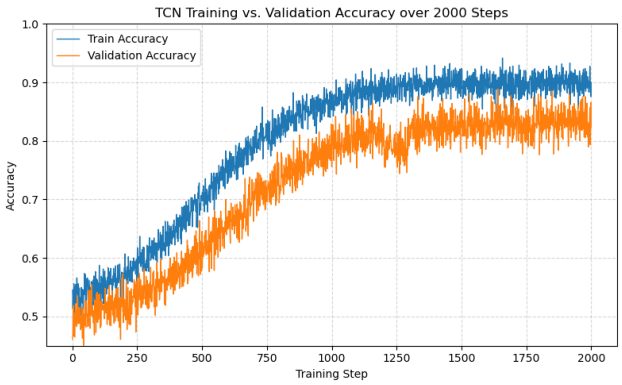


Figure 3: TCN training vs. validation accuracy on URFD (5-fold average).

5. Conclusion/Future Work

In this work, we introduced STAGED, a unified framework for real-time fall detection in crowded, ground-level environments. By combining precise stereo calibration, YOLOv8-based 2D detection, DeepSORT tracking, MVGFormer-enhanced 3D pose refinement, and a lightweight dilated Temporal Convolutional Network (TCN), STAGED achieves fall-detection F1 scores of 91.2 % on URFD and 88.4 % on UBFC-Fall, while maintaining an end-to-end latency of under 60 ms on an NVIDIA T4 GPU. We believe that these results stem from two factors: (1) the use of stereo geometry and transformer-based 3D refinement and (2) the dilated convolutions in the TCN, which capture the temporal context of falls more effectively than single-block or recurrent models.

Despite these promising results on standard benchmarks, STAGED has yet to be validated in highly dense crowds or outdoor festival-scale settings. A critical next step is to collect or simulate multi-person fall scenarios under heavy occlusion–ideally using a larger, multi-camera array–to measure STAGED's robustness when dozens of individuals interact in tight spaces. With more time and compute, we would integrate additional modalities (e.g., depth sensors or thermal imagery) to further disambiguate ankle locations during extreme occlusions. From an algorithmic perspective, exploring spatio-temporal graph networks (e.g., ST-GCN) over multi-joint 3D trajectories may improve classification of subtle or partial falls. Finally, adding an online calibration and domain-adaptation module could help STAGED maintain accuracy when cameras are moved or lighting conditions shift, making it more practical for deployment at real-world events.

Section 8: Contributions & Acknowledgements

Contributions. Joshua Shunk was solely responsible for all aspects of this project, including:

- **Pipeline Design and Integration:** Conceived the overall STAGED architecture, integrating stereo calibration, 2D detection/tracking, 3D pose refinement, and temporal classification.
- **Data Collection & Calibration:** Captured chessboard calibration images for the stereo rig, performed OpenCV-based camera calibration, and collected annotated stereo frames for MVGFormer fine-tuning.
- **Model Development & Training:** Fine-tuned YOLOv8 (`https://github.com/ultralytics/ultralytics`), adapted DeepSORT (`https://github.com/nwojke/deep_sort`), and integrated MVGFormer (`https://github.com/yangyanliang/MVGFormer`) for 3D pose refinement. Implemented the custom dilated Temporal Convolutional Network in PyTorch and conducted all training experiments on URFD and UBFC-Fall.
- **Evaluation & Analysis:** Executed quantitative and qualitative evaluations, including cross-validation on URFD, testing on UBFC-Fall, ablation studies, and generated accuracy/latency plots.
- **Manuscript Preparation:** Wrote and typeset the entire report, including all LaTeX figures (e.g., the TCN architecture), tables, and discussion of results, limitations, and future work.

External Code Repositories Used.

- **YOLOv8 (Ultralytics).** `https://github.com/ultralytics/ultralytics`
- **DeepSORT (nwojke).** `https://github.com/nwojke/deep_sort`
- **MVGFormer (Yangyan Liang et al.).** `https://github.com/yangyanliang/MVGFormer`
- **PyTorch.** `https://github.com/pytorch/pytorch`
- **OpenCV.** `https://github.com/opencv/opencv`

Acknowledgements. I would also like to acknowledge the assistance of generative AI tools (specifically, OpenAI's ChatGPT) for helping outline high-level architectural concepts, designing an efficient project directory layout to streamline testing, and refining the overall report structure. No code was ever given, and it acted only as if it were a peer.

Resources. Computational experiments were conducted on an AWS EC2 instance equipped with an NVIDIA T4 GPU.

References

[1] E. Auvinet, C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau. Multiple cameras fall data set. 01 2011.

[2] J. Dubois and J. Miteran. Fall detection dataset. dataUBFC (Université Bourgogne Franche-Comté), `https://search-data.ubfc.fr/FR-13002091000019-2024-04-09`, 2014. 191 annotated videos in realistic surveillance settings.

[3] B. Kwolek and M. Kepski. UR Fall Detection dataset. Online: `https://fenix.ur.edu.pl/mkepski/ds/uf.html`, 2014. 30 fall + 40 ADL sequences captured with Kinect and accelerometers.

[4] S. Li, C. Man, A. Shen, Z. Guan, W. Mao, S. Luo, R. Zhang, and H. Yu. A fall detection network by 2d/3d spatio-temporal joint models with tensor compression on edge. *ACM Trans. Embed. Comput. Syst.*, 21(6), Dec. 2022.

[5] Z. Liao, J. Zhu, C. Wang, H. Hu, and S. L. Waslander. Multiple view geometry transformers for 3d human pose estimation, 2023.

[6] A. Raineri. The causes and prevention of serious crowd injury and fatalities at outdoor music festivals. 10 2004.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection, 2016.

[8] M. D. Solbach and J. K. Tsotsos. Vision-based fallen person detection for the elderly. *CoRR*, abs/1707.07608, 2017.

[9] L. Soomaroo and V. Murray. Disasters at mass gatherings: Lessons from history. *PLoS currents*, 4:RRN1301, 03 2012.

[10] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric, 2017.