Signal, Image and Video Processing (2019) 13:643–650
https://doi.org/10.1007/s11760-018-1392-8

ORIGINAL PAPER

Augmentation of virtual agents in real crowd videos

Yalım Doğan[1] · Serkan Demirci[1] · Uğur Güdükbay[1] · Hamdi Dibeklioğlu[1]

Received: 26 September 2018 / Revised: 7 November 2018 / Accepted: 8 November 2018 / Published online: 22 November 2018
© Springer-Verlag London Ltd., part of Springer Nature 2018

Abstract
Augmenting virtual agents in real crowd videos is an important task for different applications from simulations of social environments to modeling abnormalities in crowd behavior. We propose a framework for this task, namely for augmenting virtual agents in real crowd videos. We utilize pedestrian detection and tracking algorithms to automatically locate the pedestrians in video frames and project them into our simulated environment, where the navigable area of the simulated environment is available as a navigation mesh. We represent the real pedestrians in the video as simple three-dimensional (3D) models in our simulation environment. 3D models representing real agents and the augmented virtual agents are simulated using local path planning coupled with a collision avoidance algorithm. The virtual agents augmented into the real video move plausibly without colliding with static and dynamic obstacles, including other virtual agents and the real pedestrians.

Keywords Data-driven simulation · Pedestrian detection · Pedestrian tracking · Crowd simulation · Collision avoidance · Augmented reality

1 Introduction

Crowd simulation research aims to generate plausible and realistic virtual crowd animations for various application areas, including the entertainment (movies and games) and safety. The simulated scenarios vary from emergency evacuation simulation on a building to artificial crowds generated in video games for creating a realistic environment for the player. The most important aspect of these simulations is how closely they resemble the real crowds in the sense of their behavior and appearance. The visual quality of the simulations depends on the modeling and behavior of the agents, which include geometry, motion, personalities, and emotions, as well as the environment, including static and dynamic obstacles.

Augmented reality is widely used in many application domains such as gaming, social platforms, education, and design. In some of these applications, there is a demand for crowd videos including real and virtual agents seamlessly integrated together. For example, for first-aid training, one can incorporate virtual agents representing paramedics into a crowd video (e.g., a crowd in a stadium or a concert area) that are applying first aid procedures (i.e., artificial respiration) to wounded individuals in the real video. Another application would be training security personnel for terrorist attacks on crowds. One can insert security officers and terrorists into a crowd video of such an incident and simulate their actions for training security officers.

We propose a framework for seamlessly augmenting virtual agents into crowd videos and moving them around realistically during the simulations without colliding with each other and static and dynamic obstacles in the environment, including pedestrians in the video feed. We tested our framework with the videos of semi-crowded areas with 30-50 pedestrians. We record the videos with a monocular and somewhat static camera, only slightly moving because of human intervention. For such scenarios, we record videos of a pedestrian area from a high altitude but still close enough to clearly identify the pedestrians. The recorded videos do not contain any dynamic obstacles, such as cars. We will provide the recorded videos to the research community.

The rest of the paper is organized as follows: We discuss related work on pedestrian detection and tracking and crowd simulations in Sect. 2. Section 3 provides an overview of the framework. In Sect. 4, pedestrian detection and track-

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s11760-018-1392-8) contains supplementary material, which is available to authorized users.

✉ Uğur Güdükbay
gudukbay@cs.bilkent.edu.tr

[1] Department of Computer Engineering, Bilkent, 06800 Ankara, Turkey

ing processes are discussed together with results for our test video footages. Section 5 describes our method for projecting pedestrians in the videos into the simulation environment, augmenting the real videos with virtual agents, including the local path planning implementation for moving the synthetic agents around realistically without colliding with each other and static and dynamic obstacles, including real pedestrians in the video feed. Section 6 provides experimental results, including the visual results. Section 7 provides conclusions and possible future extensions.

2 Related work

The proposed approach can be regarded as data-driven simulation because it uses detection and tracking data from real crowd videos to project pedestrians into our simulation environment. Usually, data-driven approaches in crowd simulations use the detection data from videos but without directly projecting the agents accordingly. Lerner et al. [1] extract the trajectory of real pedestrians in a given video in order to create a behavior database. By forming a query for each virtual agent in the simulation to the database, agent trajectories are adjusted by resulting influence forces, ending up with a realistic movement. Charalambous et al. [2] use machine learning techniques to analyze the crowd trajectories in the simulation or video referring to the reference simulation. Musse et al. [3] follow an approach similar to ours where the pedestrians in the video feed are tracked and their trajectories are used for simulating the environment. However, they do not directly project the pedestrians onto the video; they extract trajectory patterns of pedestrians to derive velocities of the virtual agents in the simulation.

Kim et al. [4] propose an approach for obtaining a realistic crowd simulation by learning from trajectories in a real-life crowd video. The method is fully automatic, and it can define the dynamics of agents in the video that can be used to synthesize a new and adaptive crowd usable in arbitrary environments. Different from our approach, they do not project the agents in the video to the 3D simulation space, which means there is no augmentation. In addition, our approach provides an interface for fine-grained control of synthetic agents in the video.

A study where the pedestrians are directly projected is performed by Narahara and Kobayashi [5] for walkthrough in architectural structures using real projectors and 3D building models. Bulbul and Dahyot [6] use social media data to populate virtual cities with virtual agents. In contrary to our framework, they use images on social media platforms, rather than a given video feed of the simulation area.

There are other data-driven simulations that extract information from videos to generate models from the extracted parameters to represent virtual environments, rather than augmenting real videos with virtual agents. Turkay et al. [7] use information theory to create a behavioral model of agents, which is related to their aggressiveness.

There is some notable research to insert virtual agents into videos. Thalmann et al. [8] simulate a virtual agent in the manually constructed simulation environment by extracting camera calibration for positioning and occlude the model by masking and animating with basic path planning. In order to increase the efficiency of managing virtual crowds, Zheng and Li [9] use real-life markers for the generation, manipulation, and interaction with the crowd. Obaid et al. [10] use virtual reality headsets together with several sensors to test the effect of non-verbal attributes of a virtual agent on the user physiologically for human–machine interactions.

Fernandez et al. [11] propose a framework to use natural language processing (NLP) for the creation and manipulation of virtual agents by the user. The behavior of the virtual agents is based on certain rules and reactions that are associated in a tree-like structure, *situation graph tree*. In this framework, different from the previous ones, the virtual agents interact with real agents in the video feed, by interpreting their actions using fuzzy logic.

Baiget et al. [12] simulate the virtual–real agent interactions in real time, using a baseline similar to [11], together with various position and motion detection algorithms. The framework also contains non-pedestrian agents, such as cars, that participate in the simulation by actively interacting with agents in the scene. Yet, their framework only simulates with a small bunch of agents compared to ours with an ad hoc solution to collision detection, i.e., evaluating the distance between obstacles and the agent of interest.

Bera et al. [13] extend the collision avoidance between virtual and real agents by introducing personal spaces. By estimating the current behavior of agents in terms of reciprocal velocity obstacles (RVO) [14] parameters, their individual personalities are obtained, which is then used to predict their individual trajectories.

Zhang et al. [15] simulate virtual agents in an environment. Although they do not simulate a crowd, they generate the environment automatically by estimating the camera parameters and focal length. They use a freely moving camera, which is able to see the area from different views. Frames taken from the camera are used to position a point on the 2D image plane into the 3D simulation area that represents the visible area by the camera. Because the simulation area is represented in 3D, the real and virtual objects (agents) both cast and obtain shadows, in addition to occlusion.

For local path planning of virtual agents, we use RVO. In this way, virtual agents plausibly move without colliding with static and dynamic obstacles in the environment, including virtual agents and real pedestrians in the video. For global navigation, we use Unity's pathfinder algorithm [16]. We do not use Unity's navigation system; rather, we use our own

navigation algorithm using navigation meshes. We also test another approach [17] that uses a hierarchy of navigation meshes of the simulation area, which brings an improvement on the navigation speed of virtual agents.

Olivier et al. [18] investigate if virtual reality (VR) is a reliable medium for interactive locomotion environments where a real human participant is set to avoid colliding with a virtual agent in the simulation. The effect of VR on the perception of the virtual agent's navigation is compared to its real-life counterpart, to see if VR impaired the required information for collision avoidance. In addition, a number of locomotion interfaces are presented to the participants and their usability are assessed. However, the collision avoidance scenarios only include a single virtual agent which cause reduced scalability to large crowd simulations.

In contrast to previous studies, we provide an interactive user interface that enables the users to modify the simulation with regard to the behavior and movement of the pedestrians and position the navigable area on top of the real video, which makes our framework applicable to different real crowd videos.

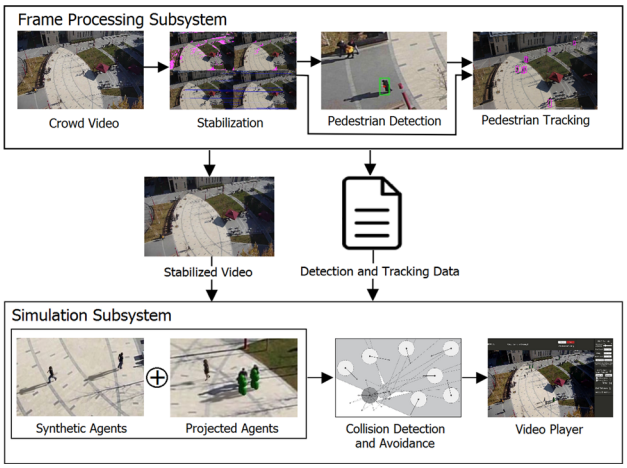

Fig. 1 The proposed framework: Video processing subsystem stabilizes video frames and detects/tracks the pedestrians. Simulation subsystem projects the locations of the real pedestrians found in the video and simulates them with simple artificial agents. However, these agents are not displayed, but rather the input video is played at the background, with virtual agents augmented into the environment

3 Framework overview

Our framework consists of two subsystems: (1) video processing subsystem and (2) simulation subsystem. The video processing subsystem detects and tracks the pedestrians in the input video. Obtained detection/tracking results are fed to the simulation subsystem. The simulation subsystem is for augmenting the virtual agents into the simulation environment and seamlessly integrating them with the real video. The proposed framework is depicted in Fig. 1. The work flow is described as follows:

- Once a crowd video is input to the system, each of its frames is stabilized since the video may be disrupted and that can adversely affect the accuracy of detection and tracking processes. We utilize the homography, namely the correspondence between projections of points on a planar surface for different camera angles, between the current frame and a reference frame (first frame) for the stabilization process.
- The positions and scales of pedestrians are detected and tracked using background subtraction and a histogram of oriented gradients (HOG) + support vector machine (SVM)-based pedestrian detector [19].
- The stabilized video is provided to the user where s/he can choose to include the detection boxes, the tracking status of pedestrians, and the mean frame rate, i.e., frames per second (fps), of the detection process. Then, using the stabilized video and the computed tracking data of pedestrians, the agents representing real pedestrians are projected into the simulation environment. Furthermore, virtual agents that are added by the user are augmented into the environment.
- Both types of agents (agents representing real pedestrians and augmented virtual agents) are simulated together by using the RVO algorithm for local collision avoidance so that the agents move around without colliding with each other and any other obstacle. In the current implementation, the mesh of the navigation area, the position of the camera, and the light sources in the simulation environment are interactively generated with the user assistance.

4 Pedestrian detection and tracking

We use a HOG + SVM-based pedestrian detector in conjunction with background subtraction to reduce the errors caused by false positive detections. If a detection does not contain a foreground object, it is removed from the detection list (cf. Fig. 2). We use a Gaussian mixture-based background segmentation algorithm to segment frames into the foreground and background regions [20]. The detector is applied only in the regions around foreground objects. Furthermore, searching only foreground regions of the frame improves the computational performance of the detector.

In the tracking stage, pedestrian trajectories are calculated by building a model for each pedestrian in the video feed. Each tracker stores a probabilistic spatial model (Kalman filter with a constant acceleration model) and the histogram of the region as the visual feature for each tracked

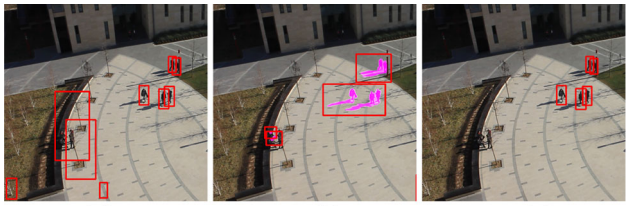

Fig. 2 Pedestrian detection and background subtraction: **a** the output of the pedestrian detector, **b** the output of the background subtractor, **c** the output of the pedestrian detector with background subtraction

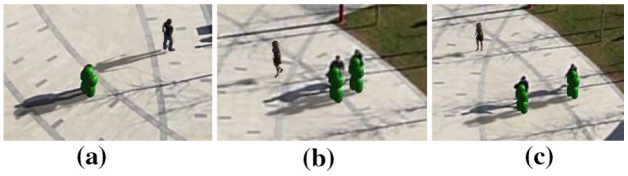

Fig. 3 The projection of pedestrians onto the simulation area: **a** the projection example of a single pedestrian, together with an virtual agent above, **b** a pair of pedestrians successfully projected separately as two agents, **c** the pedestrians go different ways where a virtual agent stays at the upper left corner

pedestrian. The spatial model is used to predict the position of the pedestrian for subsequent frames. At each frame, trackers are associated with pedestrian detections. The predicted position and the visual features of the tracker are used for pedestrian tracking. In the association process, a score is calculated for each tracker–detection (t, d) pair. The tracker–detection pair with the highest score is associated with each other. A greedy assignment algorithm is used for the association process; it is completed either when all trackers and detections are assigned to each other or all of the remaining trackers and detections have scores that are less than a pre-specified threshold, which is determined experimentally. The score function is a composition of the positional and visual score functions.

$$\text{score}(t, d) = \alpha \times \text{score}_{\text{vis}}(t, d) + (1 - \alpha) \times \text{score}_{\text{pos}}(t, d),$$

where α is the weight parameter. The visual score is the L2 distance between the color histograms (in Lab space) of pedestrian detections in consecutive frames. The positional score function is based on the predicted L2 distance of the tracker and detected position. As the distance between tracker and detected position increases, the score decreases. If the distance between tracker and detection position is greater than a pre-specified threshold, D, which is determined experimentally, the positional score function becomes zero. We take the values of D as 50 and α as 0.8.

$$\text{score}_{\text{pos}}(t, d) = \max\left(0, 1 - \frac{||\text{pos}_t - \text{pos}_d||_2}{D}\right)$$

5 Agent projection and simulation

5.1 Agent projection

The agent projection solely depends on the detection and tracking data that are output from the frame processing subsystem. The output file will be referred to as the detection result in the rest of the paper.

For each processed frame, the detection result contains the identification number (id) for the tracked (or newly detected) pedestrian, which is unique through the simulation, even when the tracked pedestrian is lost. This id is used to identify and control the projected agent's movement. For each id found in the detection result, the perspectively projected position, velocity, and the detection window dimensions are specified.

At each frame, the list of agents to be projected is checked based on their identification numbers in the detection result. Their relative positions on the navigable area are calculated by sending rays from the camera to centers of the foot positions on the view plane (assumed to be the bottom center of each detection window) and finding their intersections with the 3D mesh of the navigable area (cf. Fig. 3).

Sometimes the detection may lack the next position of the tracked pedestrian because of occlusion in the video, unstable frame, and so on. In such cases, the projected agent gets unsynchronized with the tracking data. In order to avoid lost agents, they continue to appear for a few (typically 1–2) seconds before they are automatically destroyed, instead of waiting for reconnection. In addition, the velocities of projected agents are checked at every frame for whether they exceed an experimentally determined threshold. If so, it means that the detection data about that agent are considered as noise and the agent gets deleted. This protects the simulation from false positive errors in the detection stage. However, the efficiency solely depends on the selected threshold value. If the threshold is selected small, compared to the perceived speed of projected agents, most of them would be deleted. If a large threshold is used, it cannot detect very fast agents, degrading the realism.

For the virtual agents augmented onto the video, we use custom-generated, realistic humanoid models that are almost indistinguishable from the real pedestrians in the video. We use a simple toy humanoid model to represent the projected pedestrians, which makes it easy to distinguish them from virtual ones, hence increasing the user experience. For the final augmented rendering of the simulation, we do not show the simple toy models representing projected pedestrians but rather we show the video augmented with virtual agents.

The height of virtual agents is determined by the height of the pedestrians in the video. After finding the 3D head

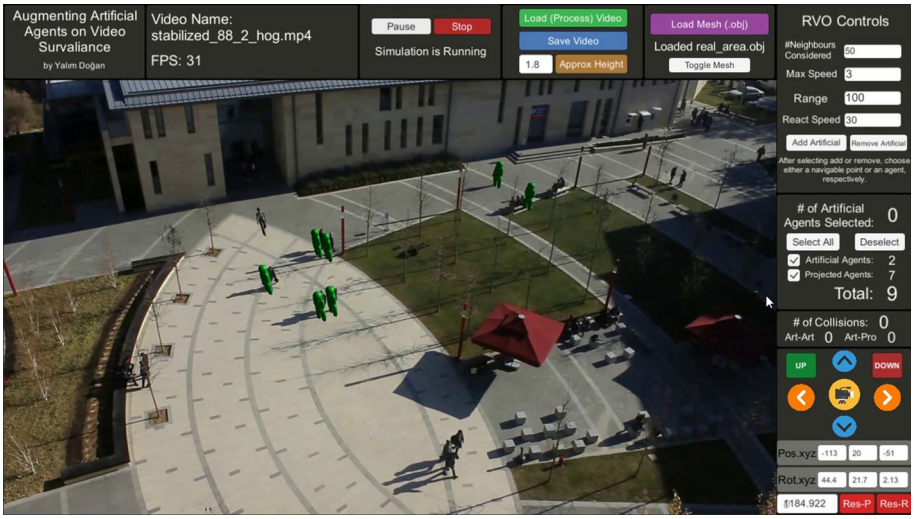

Fig. 4 The user interface of the framework

position of each detection using rays, a proposed height is obtained as the distance from head to 3D feet position, which is already known. The average of all such heights is set as the global height of virtual agents.

5.2 Collision detection and avoidance

We implemented RVO [14] for collision avoidance between virtual agents, as well as between virtual agents and real pedestrians represented according to the detection and tracking results. RVO uses Minkowski sums and velocity obstacles to determine the set of admissible velocities for each agent to obtain a collision-free movement. For each pair of agents, their Minkowski sum, which represents the area where the acting agent should avoid in order not to collide, is calculated. It is then used to calculate the resulting velocity obstacle between them. The new velocity is selected outside of the velocity obstacle, which respects kinematic constraints (such as the turning speed, maximum speed, and acceleration) and the preferred velocity of the agent.

The usage of velocity obstacles may cause the agent to oscillate between two velocities while trying to avoid collisions. To alleviate this problem, Van den Berg et al. [14] propose to divide the responsibility of adjusting velocities between the agents equally. This prevents an agent to use its previous velocity; new velocities stay optimal for reaching the goal. The approach of dividing the work between the agents requires the usage of a new parameter, which enables us to change the proportion for each. Each agent takes a part of the responsibility for changing the velocities and adjusts its own velocity. In our implementation, the projected agents are unresponsive as they are directed by the given data. Only the virtual agents change their velocity; collision avoidance algorithm (RVO) treats the projected agents as dynamic obstacles. RVO makes sure a collision-free movement for the virtual agents over a certain time period, τ. If a collision is unavoidable, agents can select velocities inside the RVO region; they receive a penalty in such cases.

5.3 The user experience

In order to simulate arbitrary scenarios, the user can provide an input video together with its detection result (from frame processing subsystem). The user can also provide an object file that includes the mesh representing the navigable area. The mesh is placed at a default location in the environment, and its resulting navigational mesh is automatically constructed via the Unity Engine. The navigable area is a flat surface; however, the user can provide any kind of mesh that is acceptable for the construction of a navigation mesh.

As the mesh and the video are related through the camera, its placement and orientation play an important role. Therefore, the user is provided with several tools for camera placement in which the loaded mesh and the input video align in the camera's perspective view. When the user is satisfied with the placement, the simulation can be started.

The user can specify various properties of virtual agents in the scene and adjust their collision avoidance (RVO) parameters through the user interface provided by the implementation (cf. Fig. 4). The user can specify

- the number of neighbors considered for each agent,
- starting positions and destinations for virtual agents,
- the maximum speed of the agent,
- the range of the circular area where the neighbor agents are taken into consideration, and
- the minimum amount of time that the calculated velocity of the agent is safe with respect to the other agents, called *reaction speed*.

Upon changing these parameters, the user can visualize the resulting behavior of virtual agents during the simulation. The user can select individual virtual agents or select all of them to change their properties using the provided interface elements. The number of collisions for virtual agents is displayed during the simulation, both with the agents projected from the real video and other virtual agents. The number of collisions with the projected agents reflects the number of frames where a collision occurs, rather than just the collision itself. This indicates how fast a virtual agent can get out of the collision status. The user can also play/pause, stop, and restart the simulation together with the input video.

New virtual agents can be added in or removed from the simulation while the simulation is running, paused, or stopped. The newly added agents are represented using different models randomly in order to provide heterogeneity. The video is converted to a format compatible with our simulation environment defined in Unity and played in the back of the navigation area.

6 Experiments and discussion

We use Unity Game Engine as our simulation environment and record our simulations on a laptop computer with Intel® Core™ i7-4500U CPU @1.8 GHz, 8 GB RAM and NVIDIA 740 M graphics chip.

We recorded two test videos of the pedestrian area in front of the Engineering Building of Bilkent University including 20–30 pedestrians. We used a consumer-grade monocular camera, stabilized by a tripod. Each video contains a sparse crowd of pedestrians on a plain surface with no additional dynamic obstacles. We record the videos from a high altitude, higher than a surveillance camera, but not perpendicular. Both videos belong to the same area, but they are from slightly different angles. Both videos have a duration of approximately one and a half minute.

We tested our framework on a static video taken from the MOT15 Database [21] and two videos that we recorded for which we manually prepared the ground truth. Each video contains a sparse crowd of pedestrians on a plain surface with no additional dynamic obstacles. Among the tested videos, the video taken from the MOT15 database is the easiest when it comes to identifying pedestrians clearly because the camera is located closer to the area of interest. In almost all of the frames in our videos, pedestrians are still distinguishable; if not always clearly. In our experiments, it is expected to have a lower recall rate in our videos than the video taken from the MOT15 database. We provide the qualitative and quantitative results of pedestrian detection and tracking in "Electronic Appendix."

In our simulation, it is important to have a high recall rate rather than a high precision rate because false positives (FPs) reflect it as additional people that our agents need to take into account while avoiding collisions. On the contrary, false negatives (FNs) are ignored by our agents during navigation. Having a higher number of FPs may cause our agents to have limited movement or not navigate at all, but compared to "going through" people in the video, it does not disturb the realism as much.

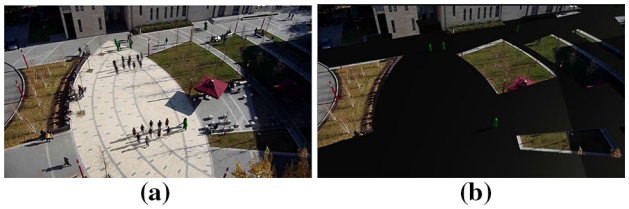

Fig. 5 Navigation mesh placement: **a** a still frame from the video and **b** the positioning and orientation of the navigation mesh on top of the video frame

Figure 5 shows the positioning and orientation of the navigation mesh on top of a recorded video. Higher resolution images for all videos are provided in Electronic Appendix. We have the mesh model of the pedestrian area generated from the floor plan. We position and orient the navigation mesh in our simulation area according to the camera position and orientation of the video used in the simulation.

We calculate the velocities of real pedestrians by interpolating their positions at the video frames. We integrate RVO into our system for collision avoidance, rather than directly using the Unity's obstacle avoidance mechanism. Unity also uses RVO for local collision avoidance; however, we want to be able to modify the RVO parameters in real-time and separate pathfinding from collision avoidance so that these parameters can be specified using different techniques for the requirements of a new simulation.

As it can be seen in the supplementary video (cf. Fig. 6), collisions between virtual agents and with real agents are very rare for sparse crowds. In dense crowds with a high number of real agents or if too many virtual agents are augmented into the environment, the number of collisions will increase. The detection and tracking errors may cause collisions with virtual agents because of the wrong positioning of real agents in the environment. Because we cannot control the behavior of the real agents in the video in terms of their collision avoidance behavior, virtual agents handle the collision avoidance. This causes some strange behaviors, such as sudden changes in the position of virtual agents when they are faced with a high number of projected agents.

To increase the realism of our augmented videos, we generate shadows of the augmented agents according to the manually placed directional light source in the scene. The shadows are made visible on the invisible mesh using an open-source shader for Unity [22].

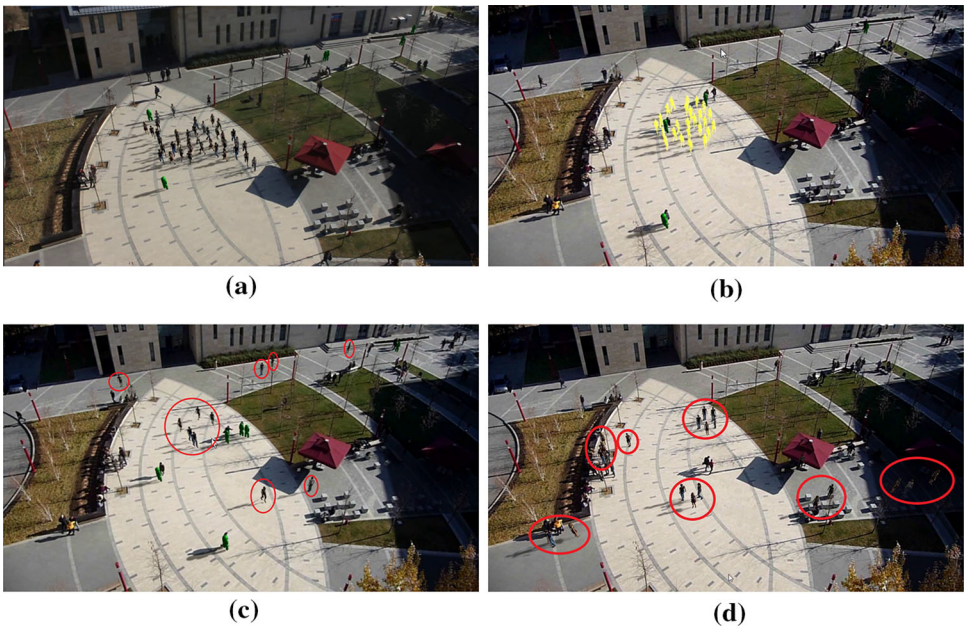

Fig. 6 Still frames from the simulations: **a** Two virtual human groups navigate in opposite directions in the middle area. **b** A large virtual human group in the selected state (as yellow) that goes through a number of real agents. **c** A scenario containing several virtual agents shown with red circles. **d** Another scenario with several groups virtual agents shown with red circles. In **a**, **b** and **c**, the detected pedestrians are shown with simple green 3D models whereas in **d** real pedestrians in the video are shown. Please note that some pedestrians are not detected (false negatives), as well as there could be extra artificial agents representing real pedestrians because of false positive detections (color figure online)

We compare and contrast the proposed approach to the state-of-the-art approaches presented in [11,12]. Fernández et al. [11] augment virtual agents into the existing video surveillance footage for incremental event evaluation, similar to our approach. Because their aim is incremental user evaluation, they propose a natural language interface to specify the desired contents. They test their approach on indoor, street, and soccer environment video sequences.

Baiget et al. [12] estimate real agent motion in real-time using a multi-object tracking algorithm and augment virtual agents with behavior models into the video by taking into account their interaction with other virtual and real agents. They test their framework with a small number of agents (at most seven people and seven cars) and achieved frame rates up to 25 fps. We simulate a pedestrian area, which has no dynamic obstacles, such as cars, as opposed to this approach. However, any dynamic objects detected and tracked by our video processing system could be easily incorporated into the simulation. Baiget et al. obtain the ground plane by applying homography to the video frames and place virtual agents accordingly. They use a simple collision avoidance method that evaluates the distance of a virtual agent to the real objects and other dynamic obstacles in the real video. Because they compose the virtual agents with the video feed on-the-fly, detection and tracking errors may also adversely affect the composition, such as a virtual agent is entering the real agent in the video. We use a collision detection and avoidance method based on RVO to arrange the velocities of the virtual agents so that they do not collide with other virtual and real agents. We represent the real pedestrians in the video using rectangular bounding boxes and feed these into the collision avoidance module. Because we process the real and virtual agents as well as the dynamic obstacles uniformly, the virtual agents move around plausibly without colliding, or colliding rarely, with each other and with real pedestrians. However, the detection and tracking errors still may cause collisions because of the wrong positioning of real agents in the simulation environment.

Our approach is currently two stage: The simulation stage follows the video processing stage; hence, it is not real time because of the requirement of extracting the navigable area and the navigation mesh, which is not done fully automatically. The navigable area and the navigation mesh are required for global path planning of the virtual agents, as well as local path planning (collision avoidance).

7 Conclusion and future work

We propose a framework for generating a crowd simulation from the pedestrians detected in a given video, augmented

with virtual agents. Our system uses pedestrian detection and tracking algorithms to project the pedestrians from the video onto the simulation environment. The virtual agents move around plausibly without colliding with each other and with the projected ones, thanks to the usage of RVO for local collision avoidance. The user can control the simulation through the provided user interface by adding and removing virtual agents, specifying their origins and destinations for the global movement, and their velocities.

The proposed framework can be extended in the following directions. Better pedestrian detection and tracking algorithms, such as [23], could be employed. The navigable area and the navigation mesh could be automatically generated from the video on-the-fly, which facilitates the online application of the proposed framework by coupling the detection and tracking stage with the simulation stage. When the detection and tracking data are extracted, agents can be directly projected in the simulation. The approach proposed by Zhang et al. [15] could be adapted to generate the navigable area automatically.

In order to make the augmented crowd appear more realistic, the dress code of the environment can be extracted from the pedestrians in the video and virtual agent models can be selected based on this information. The behavior of virtual agents, such as personalities, trajectories, walking speed, can also be adjusted according to that of pedestrians in the video. The studies such as [3,7,13] can be used to extract such information.

References

1. Lerner, A., Chrysanthou, Y., Lischinski, D.: Crowds by example. Comput. Gr. Forum **26**(3), 655–664 (2007)
2. Charalambous, P., Karamouzas, I., Guy, S.J., Chrysanthou, Y.: A data-driven framework for visual crowd analysis. Comput. Gr. Forum **33**(7), 41–50 (2014)
3. Musse, S.R., Jung, C.R., Jacques, J., Braun, A.: Using computer vision to simulate the motion of virtual agents. Comput. Anim. Virtual Worlds **18**(2), 83–93 (2007)
4. Kim, S., Bera, A., Best, A., Chabra, R., Manocha, D.: Interactive and adaptive data-driven crowd simulation. In: Proceedings of IEEE Virtual Reality, ser. VR'16, pp. 29–38 (2016)
5. Narahara, T., Kobayashi, Y.: Crowd mapper: projection-based interactive pedestrian agents for collective design in architecture. In: Proceedings of the eCAADe 33rd Annual Conference. Education and Research in Computer Aided Architectural Design in Europe (eCAADe), pp. 191–200 (2015)
6. Bulbul, A., Dahyot, R.: Populating virtual cities using social media. Comput. Anim. Virtual Worlds **28**(5), e1742 (2017)
7. Turkay, C., Koc, E., Balcisoy, S.: Integrating information theory in agent-based crowd simulation behavior models. Comput. J. **54**(11), 1800–1820 (2011)
8. Thalmann, N.M., Thalmann, D.: Animating virtual actors in real environments. Multimed. Syst. **5**(2), 113–125 (1997)
9. Zheng, F., Li, H.: ARCrowd-a tangible interface for interactive crowd simulation. In: Proceedings of the 16th International Conference on Intelligent User Interfaces. ACM, pp. 427–430 (2011)
10. Obaid, M., Damian, I., Kistler, F., Endrass, B., Wagner, J., André, E.: Cultural behaviors of virtual agents in an augmented reality environment. In: Proceedings of the International Conference on Intelligent Virtual Agents. Springer, pp. 412–418 (2012)
11. Fernández, C., Baiget, P., Roca, F.X., Gonzàlez, J.: Augmenting video surveillance footage with virtual agents for incremental event evaluation. Pattern Recognit. Lett. **32**(6), 878–889 (2011)
12. Baiget, P., Fernández, C., Roca, X., Gonzàlez, J.: Generation of augmented video sequences combining behavioral animation and multi-object tracking. Comput. Anim. Virtual Worlds **20**(4), 473–489 (2009)
13. Bera, A., Randhavane, T., Prinja, R., Manocha, D.: Sociosense: robot navigation amongst pedestrians with social and psychological constraints. arXiv:1706.01102 (2017)
14. Van den Berg, J., Lin, M., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: Proceedings of the IEEE International Conference on Robotics and Automation, ser. ICRA'08. IEEE, pp. 1928–1935 (2008)
15. Zhang, G., Qin, X., An, X., Chen, W., Bao, H.: As-consistent-as-possible compositing of virtual objects and video sequences. Comput. Anim. Virtual Worlds **17**(3–4), 305–314 (2006)
16. Unity Technologies, Inc., Unity manual: navigation and pathfinding. [Online]. https://docs.unity3d.com/Manual/Navigation.html
17. Pelechano, N., Fuentes, C.: Hierarchical path-finding for navigation meshes (HNA*). Comput. Gr. **59**, 68–78 (2016)
18. Olivier, A.-H., Bruneau, J., Kulpa, R., Pettré, J.: Walking with virtual people: evaluation of locomotion interfaces in dynamic environments. IEEE Trans. Vis. Comput. Gr. **24**(7), 2251–2263 (2018)
19. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, ser. CVPR'05, IEEE, vol. 1, pp. 886–893 (2005)
20. Zivkovic, Z., van der Heijden, F.: Efficient adaptive density estimation per image pixel for the task of background subtraction. Pattern Recognit. Lett. **27**(7), 773–780 (2006)
21. Leal-Taixé, L., Milan, A., Reid, I., Roth, S., Schindler, K.: MOTChallenge 2015: towards a benchmark for multi-target tracking. arXiv:1504.01942 *[cs]* (2015)
22. Takahashi, K.: Shadowdrawer (2015). [Online]. Available: https://github.com/keijiro/ShadowDrawer
23. García-Martín, Á., Sánchez-Matilla, R., Martínez, J.M.: Hierarchical detection of persons in groups. Signal Image Video Process. **11**(7), 1181–1188 (2017)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.