

4-24-2020

# A Semi-Automated Technique for Transcribing Accurate Crowd Motions

Alexander Fuchsberger

Brian Ricks

Zhicheng Chen

International Journal of Image and Graphics
Vol. 20, No. 2 (2020) 2050012 (18 pages)

DOI: 10.1142/S0219467820500126





# A Semi-Automated Technique for Transcribing Accurate Crowd Motions

Alexander Fuchsberger[*,‡], Brian Ricks[†,§] and Zhicheng Chen[†,¶]

[*]*College of Engineering, Bucknell University*
*701 Moore Ave, Lewisburg, Pennsylvania 17837, USA*

[†]*College of Information Science & Technology*
*University of Nebraska Omaha*
*6001 Dodge St, Omaha, Nebraska 68183, USA*
[‡]*a.fuchsberger@bucknell.edu*
[§]*bricks@unomaha.edu*
[¶]*zhichengchen@unomaha.edu*

Received 21 May 2019
Accepted 6 November 2019
Published 24 April 2020

We present a novel technique for transcribing crowds in video scenes that allows extracting the positions of moving objects in video frames. The technique can be used as a more precise alternative to image processing methods, such as background-removal or automated pedestrian detection based on feature extraction and classification. By manually projecting pedestrian actors on a two-dimensional plane and translating screen coordinates to absolute real-world positions using the cross ratio, we provide highly accurate and complete results at the cost of increased processing time. We are able to completely avoid most errors found in other automated annotation techniques, resulting from sources such as noise, occlusion, shadows, view angle or the density of pedestrians. It is further possible to process scenes that are difficult or impossible to transcribe by automated image processing methods, such as low-contrast or low-light environments. We validate our model by comparing it to the results of both background-removal and feature extraction and classification in a variety of scenes.

*Keywords*: Motion capture; pedestrian detection; agent/discrete models; real-time simulation.

## 1. Introduction

In computer animation, virtual reality and safety, models that simulate crowd behavior are increasingly used to provide a realistic representation of moving pedestrians and other types of crowds. Applications are manifold. Predictive scenarios for public building evacuations can lead to the design of safer and more

Corresponding author.

efficient layouts. Video games and movies sell better if crowds appear to be dynamic, realistic and immersive. In general, higher realism in crowd simulations translates to more trust and adaption in the industry.

A variety of models for creating synthetic crowd behavior have been investigated in recent years. Due to the dynamic and uncontrolled nature of crowds, it is, however, difficult to evaluate such models. An obvious choice is to compare synthetic motion data to original reference data. For a synthetic crowd to look realistic, it must adequately resemble the motion of real pedestrian crowds. While others have provided a means to evaluate or compare synthetic crowd data,[1] we demonstrate the production of authentic, realistic reference data that can then be used for such approaches.

Multiple models have addressed the problem of crowd segmentation and tracking, ranging from image processing techniques, such as background-removal (bg-removal) or sampling-based pedestrian detection, to sensor tracking in controlled environments. Experimental data acquisition from sensors is often not feasible to capture large crowds ($> 50$ actors), while current image processing techniques still suffer from issues such as occlusion or distortions through perspective, view angle and distance. Therefore, these methods often work well for low-density crowds, but fail in more dense or otherwise obscured scenes.

There are two current main thrusts in the area of automatic trajectory detection:

- Automated bg-removal techniques can be used to identify moving objects such as pedestrians in front of static backgrounds. These approaches are good at tracking moving objects but often fail when the crowd becomes denser and occlusion starts to appear.
- Automated feature extraction and classification models can be trained to detect pedestrians, using agent-based algorithms.[2] Although these approaches have several advantages (e.g. real-time detection, automation, or mobile camera deployment), they still suffer from severe limitations: Refining the classification process is nontrivial and specific to the features of a given scene. Some scenes may never produce satisfactory results as features to describe objects are simply lacking or inconsistent in different areas of the video frame.

Unfortunately, even the most up-to-date methods[2–5] do have considerable error ratios. While bg-removal algorithms have to deal mostly with noise (false positives), feature extraction and classification methods often cannot identify pedestrians if they are distant to the camera or lighting/contrast or occlusion in the scene do not allow them to separate objects from the background (false negatives). Our research is motivated by the goal to provide a means to achieve the highest possible detection accuracy, until a time comes when automated detection methods catch up and produce similar or superior results.

We present a technique that can avoid or minimize these issues by manually annotating (transcribing) video scenes. Against the trend of automation, we in-source the process of finding the accurate position of a person back into the human brain.

This allows us to make full use of superior cognitive abilities. The presented approach requires only a stationary video of the crowd that is to be transcribed. After the annotation process, the position of pedestrians in the scene at any given time during the clip (relative to the video frame) can be determined both relative to the screen (input) and absolute on a two-dimensional plane in the real world (output).

Unlike other methods, we intelligently place location markers not where pedestrians appear to be, but rather where they should be, based on visual clues and motion trajectories, interpreted by the person transcribing the scene.

We demonstrate the accuracy of our approach in four distinct scenes of pedestrian crowds. These scenes contain most of the previously mentioned issues such as low-contrast or occlusion, where existing image processing methods fail to produce solid results. We validate our results by applying a range of bg-removal techniques to our dataset and measuring missed agents, occluded agents and faulty artifacts. The comparison to our manual annotation technique with a zero-error tolerance shows how significantly bg-removal algorithms are actually failing in nonideal circumstances.



## 2. Related Work

In recent years, researchers have developed a variety of models for simulating realistic crowd motion.[6–9] Realism is hereby defined as the quality of motion, or how similar a computationally produced (synthetic) crowd looks to an outside observer compared to an original, human crowd. The majority of approaches are based on multiagent models, where each pedestrian is represented by a self-contained processing unit, called an agent. There are also macroscopic approaches that address crowds as single units. These are typically used in predictive analysis.[10]

Since the early multiagent models for simulating crowds,[11] agent behavior has been extensively refined. In addition to collision avoidance and basic path-finding capabilities, agents can now interact with and react to both their immediate and distant environments. Some models feature agents that diverge from another by using cultural or psychological factors, and others produce highly realistic behavior in specific scenarios, such as walking around corners,[12] or animating characters along a given trajectory line.[7]

Due to the difficulty in comparing such diversity, the field of crowd simulation research has traditionally lacked some sort of unifying standard. Researchers have attempted to generalize their research in form of frameworks.[1,13] While the industry heavily focuses on providing realism through the animation and visual appearance of characters, scientific research is primarily concerned with realism through the creation of authentic motion behavior.[13]

### 2.1. *Realism in crowd simulations*

In crowd simulation research, creating realistic looking crowds is a core objective. Much work has addressed the proper selection of parameters that define realism and

techniques for generating and simulating synthetic crowds. However, in comparison, little research has been conducted on the generation of original crowd data. As a result, the majority of works demonstrating new or enhanced methods for simulating crowds validate realism of synthetically created crowds by comparison to preexisting models. We argue that the realism of synthetic crowds is best validated by transcribing and analyzing a real human crowd in a specific scene and then replacing the agents derived from the original crowd with synthetic versions produced by a crowd algorithm. Parameter estimation and optimization can then be used to select the best fitting crowd algorithm for a given scenario.[1]

Comparing synthetically produced crowds to original crowds has a significant advantage: parameters that describe and evaluate a crowd can be equally applied to both the original and the product, resulting in an objective evaluation criteria for the realism of a crowd. Thus, unless a comparison is impossible, such as in predictive scenarios, we recommend building a crowd scenario on a real-world example. Future researchers can then base their new or enhanced algorithms on the original dataset and thus avoid a comparison between two artificial products that may not effectively be compared to each other. Many crowd simulation papers introduce algorithms that compare only to preceding works and are therefore left pointing out superiority in computational efficiency.

### 2.2. *Pedestrian detection and annotating crowds*

Identifying and segmenting moving objects in videos of dense crowds has been addressed and demonstrated in a variety of models. Typically, pedestrian motion data are generated from one of the following two sources:

- Through sensor data (experimental setting);
- Through video data (image processing).

Sensors can enable highly accurate motion tracking, but are costly to deploy and may alter pedestrian motion behavior. While micro behavior, such as collision avoidance, may not be affected, an experimental setting can make participants more determined in pursuing their objectives or altering behavior according to a given set of instructions. Because sensors are not a native part of a crowd and have to be deployed manually, they are effectively not a suitable tool for annotating crowds outside an experimental setting. Further, sensor detection is limited in scope and therefore unsuitable to capture large crowds of hundreds of people in places like airports, concerts or gatherings. Such experimental settings can be used to generate a data source for our transcription technique, but are not a requirement.

With the advancing possibilities in machine learning and a variety of applications, image processing is the primary focus in recent crowd simulation research. Source materials are significantly less expensive to acquire and produce, since they mostly consist only of video material. Detailed information can be extracted from videos.

In one study, the heart rate of people was accurately estimated solely based on videos of head motions.[14] In another study, the interaction of people and objects has been explored using a iteratively improving feature descriptor.[15] Collaborative representation classification has been used to improve classifiers for face-recognition.[16] Because of emerging topics like self-driving cars, automated pedestrian detection is an obvious area of interest.

Automated pedestrian detection methods are usually based on background-subtraction (bg-subtraction) techniques or use pre-trained models that can detect features in the video frame such as the shape of a person, even if it is partially obscured.[17–20] Because both methods are error prone to some degree we explored the idea of a semi-automated annotation technique that allows maximum accuracy, avoids errors, and is feasible to deploy on shorter video clips. It is semi-automated because only a part of the scene needs to be annotated while the missing information (agent positions) can be estimated and simulated automatically.

In bg-subtraction methods, colors and contrast for each pixel in subsequent frames are averaged to determine areas in the frame with activity. A binary mask is then applied that shows moving objects in white and the static background in black. This has several shortcomings: First, and foremost, bg-removal, despite significant advancements in recent years, remains noisy. Moving objects may not be identified in low-contrast areas of a video. Shadows can distort shapes or become new objects. For a computer, it may be hard to differentiate moving objects that are not part of the analysis, such as plants moving in the wind or cars in a crossing where pedestrians are subjects of the scene. Also, perspective becomes an issue since objects look different in size and shape, depending on distance and angle to the camera lens. Even from top–down angles with adequate viewing distance, an object's center may never resemble the actual position. Although tracking head positions is popular, we suggest that tracking pedestrians' positions at the center on the floor between their feet is a more precise estimation of their locations. Dense crowds become even more problematic because agents are likely obscuring each other.

Clustering rich sets off tracked features, such as heads, has been demonstrated as an alternative method to bg-subtraction, with decent success in handling occlusion.[21] However, the method is designed to count moving objects, rather than pinning down their exact locations. Using a tweaked body part detector that is capable of identifying only partially visible pedestrians has been proposed as an alternative means to deal with occlusion.[22] Following recent publications, we feel that deep-learning methods for automated pedestrian detection are increasingly replacing bg-removal techniques, as they allow for more than just the recognition of movement.[2–5,23–26]

In automated pedestrian detection research, popular source material often comes from stationary mounted cameras. Popular datasets, such as Caltech[27] or Kaist,[28] are typically reused to benchmark pedestrian detection algorithms. Video results in deep-learning techniques typically feature boundary boxes that show the size and position of pedestrians in the video frame. Bg-removal techniques show

moving objects in white, while the static background of the scene is colored in black. More sophisticated algorithms can filter noise, identify separate objects, and display them in different colors. In a final step, the center of such objects needs to be estimated and adjusted based on proximity to the camera, video angle and size of the object.



## 3. Research Method

In this section, we provide the details on how our video overlay method can be used to transcribe crowd motions. We address design choices and considerations and how limitations that occur in other methods can be avoided. We also briefly address the implementation and technical aspects of the technique. We conclude by describing the features of several scenes that were used as representative examples.

### 3.1. *Overview*

In Eq. (1), a given crowd $C$ is defined as a collection of markers $M$. Each marker is a quadruple containing an agent identifier $i$, and a two-dimensional coordinate $(x, y)$ that describes the position of that marker in the video at a given time $t$. More precisely, $x$ and $y$ are relative coordinates on the video frame and $t$ is measured in milliseconds since the first frame.

$$M = (i, t, x, y) \in C. \tag{1}$$

This data representation is suitable for data storage in any relational database system. Agent-based crowd simulations, however, operate on a per-agent basis. Thus, markers have to be grouped by agent identifiers and sorted by time. In Eq. (2), an agent $(A_j)$ is described as follows:

$$A_j = \{M_0, \ldots, M_n\} \leftrightarrow M.i = j. \tag{2}$$

The two nearest markers of an agent $j$ at any given time can then be determined by looping through all markers that belong to the agent $A_j$ (Algorithm 1):

**Algorithm 1.** GetClosest(Markers, time)

```
1: n ← length of agent
2: if n < 2 then return false
3: for i < n do
4:     M_0 ← Markers[i]
5:     if time <= M_0.t then return GetPos(time, M_0)
6:     M_1 ← Markers[i + 1]
7:     if ¬M_1 then return false
8:     if time < M_1.t then return GetPos(time, M_0, M_1)
```

The estimated position can then be calculated using the vector between the two marker positions, their time difference and the time of the current video frame (Algorithm 2):

**Algorithm 2.** GetPos (time, $\text{Marker}_0$, $\text{Marker}_1$)

$\text{rel}_x \leftarrow \text{Marker}_0.x$
$\text{rel}_y \leftarrow \text{Marker}_0.y$
**if** $\text{Marker}_1$ **then**
    $\text{rel}_{\text{time}} \leftarrow (\text{time} - \text{Marker}_0.t)/(\text{Marker}_1.t - \text{Marker}_0.t)$
    $\text{rel}_x \leftarrow \text{Marker}_0.x + (\text{Marker}_1.x - \text{Marker}_0.x) * \text{rel}_{\text{time}}$
    $\text{rel}_y \leftarrow \text{Marker}_0.y + (\text{Marker}_1.y - \text{Marker}_0.y) * \text{rel}_{\text{time}}$
**return** $\text{rel}_x, \text{rel}_y$

We tested Catmull–Rom splining to smooth out trajectory paths between markers.[29] In practice, however, the distortion of a basic linear path between two markers was not noticeable if the time interval between two consecutive markers was small enough. We found that a threshold of 800 milliseconds between two consecutive markers was sufficient enough to rule out any potentially noticeable visual difference in all scenes. If a pedestrian would, however, move with a speed of more than 100 pixel/second on the screen, a smaller threshold may improve localization accuracy. Accuracy improves with shorter intervals between agent markers. None of our scenes had a time gap of more than 1200 ms between two consecutive agent markers. By providing a flexible marker interval, it is possible to dynamically alter marker frequencies depending on the scene and movement paths of pedestrians. For example, agents that stand still in the video or agents that move in straight lines may require fewer markers without impacting accuracy.

### 3.2. *Transforming screen position into absolute position*

Calibrating screen position with the three-dimensional position on a world frame has been addressed in previous research.[30,31] The basic idea is that the screen position ($P_S$) can be derived from the real Position ($P$) by multiplying it with a perspective projection matrix ($M_{\text{proj}}$).

In our approach, we assume that all objects are moving on a two-dimensional plane, which simplifies the model for coordinate translation to a basic, geometric approach that does not require variables such as the field-of-view angle or camera position related to the frame. It is also assumed that a possible fish-eye effect was eliminated in a pre-processing step so that the video shows a pure perspective projection of the scene. Given enough distance from the camera to the pedestrians ($> 3$ m) a distortion of a remaining effect can be neglected. We examined a potential distortion in (Fig. 2(d)), which had the highest potential for a remaining fish-eye effect due to its top–down perspective of the area. We found that the inaccuracy of an

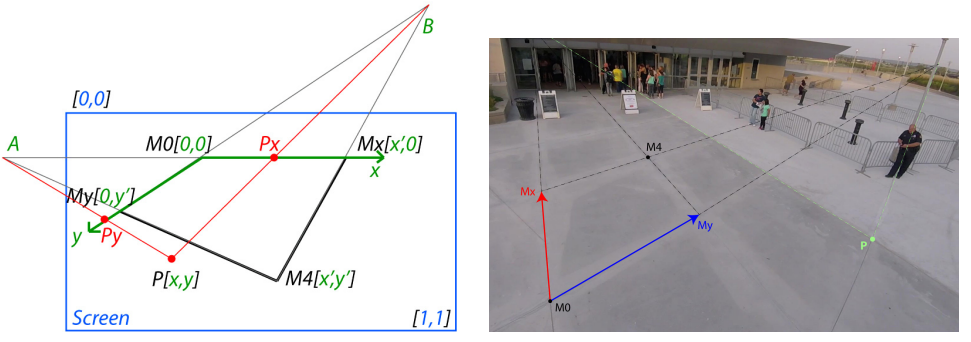



Fig. 1. Coordinate Translation. The real location of any given point ($P$) on the screen can be calculated, given a rectangle in the real scene ($M_0$, $M_x$, $M_y$, $M_4$) and its width and height (left). Demonstrated on the low-angle scene in Fig. 2(c) (right).

absolute position resulting from a misplaced marker (e.g. placing a pixel too far to the right) was more significant than a distortion from a perspective effect.

Given a rectangle on the screen ($M_0, M_x, M_y, M_4$) with a known height and width in the real world, it is possible to translate any screen coordinate into a real-world coordinate usings the cross ratio (Fig. 1).

To be able to calculate the screen positions of two vanishing points ($A$, $B$), the projected rectangle in the real world cannot also be a rectangle on screen. For simplicity, we lock the first anchor of the rectangle as the point of origin ($M_0$) in the Cartesian coordinate system. The second ($M_x$) and third ($M_y$) anchor markers indicate the direction of $x$- and $y$-axis. Both width and length of the rectangle must be known (in meters). We can then calculate the coordinates of two points ($A, B$) that mark the crossing point of the natural extensions, where an axis meets with its parallel side (Fig. 1).

To get the absolute position of any point ($P$) on the screen, we can then draw a line to $A$ and one to $B$ and calculate the intersection where $P_A$ meets $y$-axis ($P_y$) and $PB$ meets $x$-axis ($P_x$). By comparing the length of the rectangle side ($M_0, M_y$) to the intersection point ($M_0, P_y$) and ($M_0, M_x$) to ($M_0, P_x$), we can derive the Cartesian $x$- and $y$-coordinates of $P$ (Fig. 1).

This approach only works with sufficient accuracy if a rectangle can be chosen, such that $A$ and $B$ are located well outside the screen frame and every screen coordinate within the frame therefore has a valid real-world coordinate equivalent.

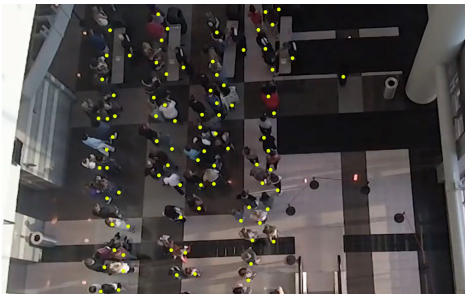
### 3.3. *Implementation*

Our application, labeled *CrowdCrush*, runs on a Ubuntu Linux server (16.04) in the *Elixir* language that is based on *Erlang*. *Elixir* was selected because it excels as a functional language for I/O intense web applications. As a code basis, we used an Model-View-Controller framework *Phoenix*. To ensure fluid animations while running simulations and instant user interface updates, we added the *React.js* front-end

framework. All simulations are conducted in real-time inside the client browser. Videos are loaded and integrated via API from YouTube to ensure that moving markers synchronize with the video frames in the background. For user authentication and session management, we utilized the *Coherence* framework. The project is released as open source under the MIT license and can be found on GitHub, https://github.com/fuchsberger/crowd-crush.

A video is transcribed in the following procedure:

- The raw film material is ideally cropped and pre-processed using a video editing software such as *Adobe Premiere CC*. We filmed all our scenes using GoPro 4 cameras. When filming a crowd from a birds-eye perspective through GoPro cameras, this can result in a distortion known as the fish-eye effect.[32] One of our sample scenes (Fig. 2(d) was affected by this distortion. We were able to significantly reduce this effect through the application of a post-processing filter onto the video track. Where appropriate, we applied this filter. We then searched the video material for scenes showing significant crowd motion or scenes including interesting crowd phenomena, such as the formation of waiting lines (Fig. 2(a)).



(a) Parallel waiting lines.

(b) Crowd in front of shop and escalator.

(c) Low-angle perspective.

(d) Partially obscured entrance.

Fig. 2. Sample scenes filmed at a stadium in a mid-sized city in the USA: (a) depicts a crowd lining up for a concert, filmed from a distant camera in a low-contrast environment (top side of frame); (b) children and parents lining up at a gift shop while others use an escalator; (c) event visitors outside the arena from a low-angle perspective; (d) pedestrians appear through glass doors inside the video frame.

We then cut a 2 min sample and rendered our output video using a dynamic frame rate @ 3 Mb/s in full HD resolution (1920 × 1080 px), or a custom resolution, resulting from the amount of cropped borders.

- In a second step, the pre-processed video clips were uploaded online. Videos can then be managed and imported on CrowdCrush. The simulation can be played, forwarded and reversed, in synchronicity with any agent markers already transcribed. The tracked locations (markers) of pedestrians are then visualized as yellow dots. A coder transcribing a video may click on the screen at the current position of a pedestrian. This will drop a marker, and the video jumps forward in time by a pre-set time interval. Clicking again would repeat the process of dropping a marker and jumping forward in time. This way an agent can be followed through its life cycle without losing focus. We have implemented keyboard controls that speed up the transcription process, such as moving forward and backward by a single time interval, or selecting, de-selecting or deleting agents. Once a video transcription is complete, it can be locked to prevent further editing.

### 3.4. *Scenes*

We chose four distinct scenes to show the flexibility of the approach (Fig. 2). All scenes were filmed at a stadium in a mid-sized urban area in the USA. We filmed at four different events with varying types of crowds. We noticed a predominantly male crowd at a wrestling match, and a younger, female crowd at a concert. One event featured a Disney show, targeting children; consequently, many single parents attended with small children. Motion behavior in those varying crowds differed significantly. Naturally, children with parents had generally slower motion and more frequent stops. At each event, we filmed crowds at the same five spots, shortly before entering the arena area. We started filming 90 min before the start of the event and continued until 30 min after the event start. From the raw material, we purposely selected scenes that included most of the common issues that image processing cannot deal with effectively.

Figure 2(a) shows the waiting lines in front of the main entrance from a top–down perspective. The camera is located about 15 m above the crowd. In this scene, it can be observed how three parallel waiting lines split at four security check points and then merge back together for the entrance gates. This scene was selected because people in the top third of the video are almost invisible because of the dark floor. The shape of pedestrians also looks significantly different in the top compared to the bottom of the video frame. This scene is also a good example for strong occlusion because of the density of the crowd, especially in the top of the video frame.

Figure 2(b) shows the formation of a crowd in front of a market stand. It also features a partially obstructed escalator where people move with constant speed out of the frame. This scene was selected because people in the crowd are moving on either a very dark or very bright background. This scene also shows how people move around obstacles and how the average closeness intensifies with proximity to the shop counter.

Figure 2(c) depicts a crowd filmed from an unusual angle. This scene was selected because the size of persons would significantly vary depending on closeness to the camera. It also shows that our model can directly locate the position of a person where their feet touch the floor, in contrast to the center of the moving object as it is calculated in most of the image processing variants.

Figure 2(d) shows a crowd that has passed the security check and enters the area from a top-down perspective. It was chosen because people do not enter the video frame from a border, as shown in all other scenes, but through glass doors within the video frame. Additionally, because of the transparency of the doors, the direction of approaching pedestrians can be tracked before they actually appear in the door. As a result, motion trajectories can start smoothly with an approaching pedestrian rather than just popping into the screen.

## 4. Results

To show the superiority of the approach, we compared the manually transcribed localization data with those produced by several bg-removal algorithms and histograms of an oriented gradients (HOG) feature extraction/support vector machines (SVM) classification. Specifically, we counted missed agents due to low contrast ($MA^{LC}$), missed agents because of occlusion overlaps ($MA^{OC}$) and the size of the video frame that was unable to identify agents reliably. We assigned each correctly detected agent a value indicating how close its shape in the bg-removal result matches the real shape observed in the original video. We further measured the processing time it took to generate an output ($T^P$) using a variety of bg-removal techniques. This includes running the bg-subtraction algorithm and merging the frames into a video. The time for automated processing ranged between 12 and 47 min per scene and mostly depended on the video frame size. Our method required between 85 and 234 min per scene, and mostly depended on the number of pedestrians visible in the scenes.

### 4.1. *Comparison to bg-removal techniques*

We used the BGSLibrary by Andrews Sobral.[33] The resulting output files of the binary foreground masks were uploaded on CrowdCrush and overlaid over the original video. This allowed us to inspect and compare the position and frequency of manually created agent markers with those visible in the synthetic data. Running the simulation showed that, in most cases, individuals and entire groups were not detected correctly due to noise, incompleteness, occlusion, or distortion.

To select the best available bg-removal algorithms for the given set of scenes, we applied each of the 40 available bg-removal algorithms from the BGSLibrary tool on the scene in Fig. 2(d). This test was performed to rule out algorithms that did not perform at all or were designed for a different purpose. Out of the remaining algorithms, we selected nine that had the strongest potential to identify moving pedestrians with as little noise as possible for all four scenes. Figure 3 depicts a side by side

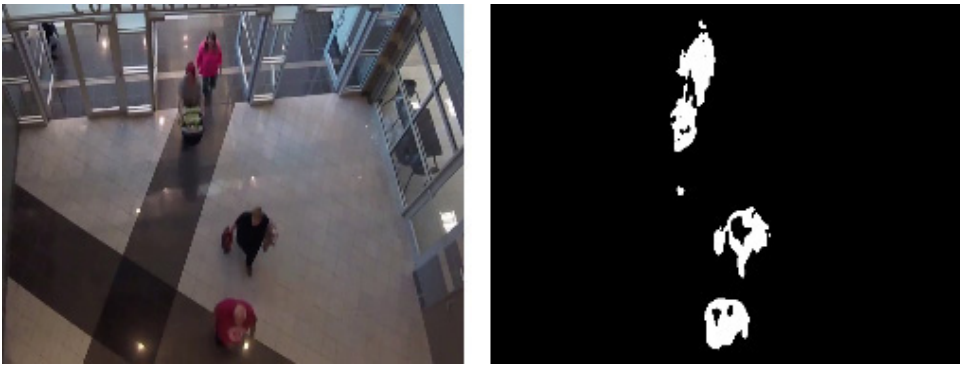


Fig. 3. Post-processing with bg-removal algorithms.

comparison of moving pedestrians in Scene D. We also ensured that at least one algorithm of each base type was present. These types included Fuzzy Algorithm, Gaussian, Frame Difference, Multimodal, and MultiLayer. Our selection included the following algorithms: MultiLayer, Static Frame Difference, LB Adaptive SOM, LG Mixture of Gaussian, Sigma Delta, KNN, Grimson GMM, and Independent Multimodal. We processed all four scenes for each of the nine algorithms and then selected the best four algorithms over all four scenes for a more detailed analysis.

We uploaded the overlays to CrowdCrush and overlayed our manual transcription on each matching bg-removal result. This revealed many encounters of missed, inaccurate or false positive agents. In our attempt to quantify these errors, we created three measures that were taken every 20 s and then averaged (six times per video and bg-algorithm):

- Missed Agents ($E^-$): This measure counts agent markers that are present in the manual transcription but missed by the bg-removal algorithm. This is usually due to low contrast to the background and/or frame issues in the algorithm to detect moving objects in time. If an object was present, but too small (less than 10% of the agent size), we considered this detection not as an agent, but noise, and therefore ignored it and counted the marker as a missed agent.
- Additional Artifacts ($E^+$): This measure counts objects detected by the bg-removal algorithm that are not actually agents. The reasons for such unwanted artifacts are primarily noise and other moving objects, such as the escalator in Fig. 2(b).
- Missed agents due occlusion ($E^0$): This measure counts agent markers that were detected in the same object by the bg-removal algorithm. Pedestrians too close to each other or partially hidden in the frame are lost and difficult to recover. There are approaches to separate such connected components.[18,22]

All measures are relative to the average number of visible agents per frame ($A^F$). We calculated an overall error ratio that sums all three measures and can be used as an

Table 1. Scene information and encountered error ratios.

| | Resolution | $M$ | $A^T$ | $A^F$ | $T^S$ | $T^A$ | A. | $E^-$ | $E^+$ | $E^O$ | $E$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| A | $782 \times 720$ | 12271 | 234 | 116.12 | 234 | 15 | 1 | 0.804 | 0.029 | 0.023 | 0.856 |
| | | | | | | 16 | 2 | 0.136 | 0.038 | 0.242 | 0.417 |
| | | | | | | 12 | 3 | 0.206 | 0.035 | 0.142 | 0.383 |
| | | | | | | 12 | 4 | 0.039 | 0.090 | 0.462 | 0.591 |
| B | $1920 \times 1080$ | 5094 | 167 | 63.74 | 124 | 45 | 1 | 0.580 | 0.029 | 0.055 | 0.663 |
| | | | | | | 47 | 2 | 0.144 | 0.047 | 0.266 | 0.457 |
| | | | | | | 34 | 3 | 0.096 | 0.029 | 0.199 | 0.324 |
| | | | | | | 35 | 4 | 0.042 | 0.185 | 0.332 | 0.559 |
| C | $1920 \times 1010$ | 3234 | 99 | 22.39 | 81 | 42 | 1 | 0.382 | 0.022 | 0.125 | 0.529 |
| | | | | | | 40 | 2 | 0.022 | 0.103 | 0.338 | 0.463 |
| | | | | | | 33 | 3 | 0.096 | 0.029 | 0.199 | 0.324 |
| | | | | | | 34 | 4 | 0.000 | 0.235 | 0.397 | 0.632 |
| D | $1072 \times 732$ | 1248 | 85 | 5.71 | 50 | 24 | 1 | 0.000 | 0.067 | 0.233 | 0.300 |
| | | | | | | 27 | 2 | 0.000 | 0.433 | 0.300 | 0.733 |
| | | | | | | 23 | 3 | 0.033 | 0.067 | 0.233 | 0.333 |
| | | | | | | 23 | 4 | 0.000 | 0.300 | 0.433 | 0.733 |

*Notes*: Automated pedestrian detection through bg-removal algorithms are compared for their detection accuracy. Contrary to an assumed error-free annotation in our model they produced at least 32% overall error ratio throughout all scenes. Algorithms tested were: (1) MultiLayer, (2) LB Adaptive SOM, (3) LG Mixture Gaussian and (4) Static Frame Difference.

$M$ is the total number of markers in scene, $A^T$ is the total number of agents in scene, $A^F$ is the average number of agents visible per frame, $T^S$ is the time to transcribe scene using video overlay technique, $T^A$ is the time to produce bg-removal overlay, $E^-$ is the ratio of undetected pedestrians, $E^+$ is the ratio of artifacts that are not pedestrians, $E^0$ is the ratio of obscured pedestrians, $E$ is the overall error ratio in correctly detecting pedestrians.

overall indicator for the reliability of a given bg-removal algorithm. We also measured the time it took for the computer to transcribe and produce the binary mask video of the scenes ($T^A$), as well as the time it took for us to manually transcribe the videos ($T^S$). Table 1 summarizes the results.

To guarantee accurate counting of agents in our measures, we colored connected components in the bg-removal overlay videos to identify which agents were occluded (Fig. 4).

To ensure the coordinate translation process produced accurate world coordinates, we captured pictures and measured the exact distances from objects in the frame to a specified point of origin (Fig. 1). We then compared the results of the real measures with the distances produced by our algorithm and found that the basic concept is working as expected. However, a slight inaccuracy of a few pixels translates already to offset coordinates, amplified with the distance to the point of origin on the screen. Any unaccounted fish-eye effect distorts the coordinate result further.

### 4.2. *Comparison to feature extraction and classification*

We attempted to extract agent positions from our scenes using feature extraction and classification. We implemented a basic HOG feature descriptor[34] and the linear SVM model.[35] We used the code-basis published in the open-cv library.[36]

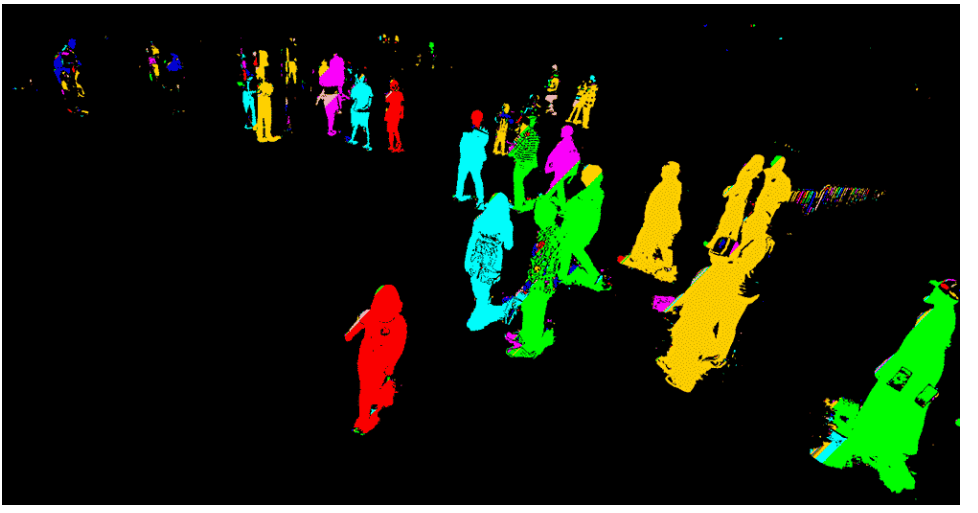


Fig. 4. Connected components help to identify and count occluded agents, reduce noise and remove artifacts.

Although we did try, we could not tweak the descriptor to produce satisfactory comparison results. Refining the classification process was very challenging due to the diversity of the scenes and the unique features within each scene. In a second approach we zoomed into scenes to allow objects to become bigger. This was somewhat more successful and we conclude that pedestrians in several of our scenes were too small to be correctly identified. The classifier was further handicapped by the inconsistency of the background, such as black areas switching into bright gray sections. Also the different camera angles let to agents occluding each other in more dense scenes.

We conclude that the scenes would have to be specifically trained to detect pedestrians, thereby ruling out a solution that is generally applicable. With an error ratio of undetected pedestrians ($E^-$) of close to 100%, we decided to omit the results from Table 1, as they are not giving any meaningful insight.

## 5. Conclusion

Regardless of the scene, no bg-removal algorithm could match the accuracy of our manual annotation method. Overall error ratios ranged between 30% and 85% compared to a 0% error ratio in our approach. While some algorithms performed well in not missing agents (Adaptive SOM, Static Frame Difference), others performed better in avoiding noise (MultiLayer). Overall, Mixture Gaussian performed best with an error ratio of 32.7% over all scenes.

We measured the time each algorithm required to produce binary mask videos of the given scenes. This time includes the time for producing png files of each frame with the BGS Library, loading them into an image processing tool (ImageJ) and then

producing an .AVI video file @30 fps. It was revealed that for a video with a given duration, the processing time of bg-subtraction was primarily influenced by the video resolution and frame rate and secondarily by the selected algorithm. In contrast, the time it took to manually annotate videos was primarily determined by the number of agents and markers.

To compare against automated detection techniques, we only tested HOG+ LVM as one of the currently leading pedestrian detection algorithms. Given the challenging scene features, a pre-trained model could not provide any meaningful results. Manually training the model for each scene would be possible, but would defeat the purpose of automated detection.

We conclude that our method for transcribing crowds is a feasible alternative to bg-removal or automated detection, if precision and a zero-error tolerance are important criteria. We directly capture the position of a pedestrian where their vertical axis meets the floor. Unlike other methods, we do not require error-prone guessing of an agent's original position. Pre-processing and supplementary tasks, such as bg-removal, component identification, and machine learning, can be avoided altogether if the primary objective is to locate pedestrians in a video frame. Manual annotation might even be faster than complex and multidimensional alternatives because of the simplicity of the work flow. Our coordinate translation technique performed very well in scenes (c) and (d), where clear environment references to initialize the rectangle were available. However, in scene (a) coordinates were slightly offset because of the camera distance and low resolution of the video, resulting in inaccurate positioning of the exact reference rectangle. In scene (b) coordinates appeared correctly, but agents on the elevated escalator could not be used. We conclude that the quality of the results are based on scenery and improve with larger video/monitor resolution.

## 6. Limitations

We are aware that our manual transcription technique is going against the trend of automated pedestrian detection. However, given the lack of an accurate universally applicable solution that produces accurate results, we feel a need to provide a temporary solution until automated detection advances to the point of superiority.

Because the transcription process is performed manually, it takes significantly more time to locate agent positions. Table 1 shows the time it took to transcribe each of the scenes. Since a human coder is required and time is the limiting resource, our approach is not feasible for long videos and videos with hundreds or thousands of agents in the frame (such as view of a stadium tribune). For the same reason, it cannot be applied in real time.

The monitor screen size, camera resolution and the distance of the camera to the pedestrians is another limiting factor. We filmed in HD-resolution and had no problems transcribing scenes where 120 actors were present at a time. We suspect that over a threshold of 150 agents/frame, the transcription process might become tedious and error-prone due to small agent sizes.

We initially considered analyzing the fit between an identified agent in a bg-removal tool with the shape of the real pedestrian in the video. Such a metric would measure the quality of the match and therefore provide another indicator for the reliability of the bg-algorithm. We were unable to reliably automate the matching and comparison and left this for a future study.



## Acknowledgments

This project is supported by NSF grant 1718139: *A Perceptual-based Approach to improve Synthetic Crowds.*[a] We thank the students of our data-science lab for their effort in filming crowds.

## References

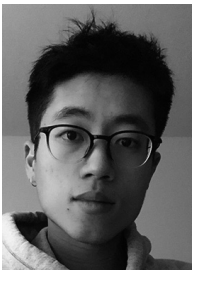
1. D. Wolinski *et al.*, "Parameter estimation and comparative evaluation of crowd simulations," *Comput. Graphics Forum* **33**, 2 (2014).
2. J. Mao, T. Xiao, Y. Jiang and Z. Cao, "What can help pedestrian detection?," in *IEEE Conf. Computer Vision and Pattern Recognition* (2017).
3. D. Xu, W. Ouyang, E. Ricci, X. Wang and N. Sebe, "Learning cross-modal deep representations for robust pedestrian detection," in *IEEE Conf. Computer Vision and Pattern Recognition* (2017).
4. Q. Ye *et al.*, "Self-learning scene-specific pedestrian detectors using a progressive latent model," in *IEEE Conf. Computer Vision and Pattern Recognition* (2017).
5. S. Huang and D. Ramanan, "Expecting the unexpected: Training detectors for unusual pedestrians with adversarial imposters," in *IEEE Conf. Computer Vision and Pattern Recognition* (2017).
6. D. Wolinski, M. C. Lin and J. Pettré, "WarpDriver: Context-aware probabilistic motion prediction for crowd simulation," *ACM Trans. Graphics* **35**, 6 (2016).
7. D. Holden, J. Saito and T. Komura, "A deep learning framework for character motion synthesis and editing," *ACM Trans. Graphics* **35**, 4 (2016).
8. G. Lu, L. Chen and W. Luo, "Real-time crowd simulation integrating potential fields and agent method," *ACM Trans. Mod. Comput. Simul.* **26**, 4 (2016).
9. A. Ansar and K. Daniilidis, "Crowd-driven mid-scale layout design," *IEEE Trans. Pattern Anal. Machine Intell.* **35**, 4 (2016).
10. N. Fridman and G. A. Kaminka, "Using qualitative reasoning for social simulation of crowds," *ACM Trans. Intell. Syst. Technol.* **4**, 3 (2013).
11. C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM SIGGRAPH Comput. Graphics* **21**, 4 (1987).
12. G. He *et al.*, "Shadow obstacle model for realistic corner-turning behavior in crowd simulation," *Front. Inf. Technol. Electron. Eng.* **17**, 3 (2016).
13. A. Fuchsberger, N. Tahmasbi and B. Ricks, "A framework for achieving realism in agent-based pedestrian crowd simulations," in *Americas Conf. Information Systems* (2017).
14. A. Al-Naji and J. Chahl, "Contactless cardiac activity detection based on head motion magnification," *Int. J. Image Graphics (IJIG)* **17**, 1 (2016).
15. M. F. Alcantara, H. Pedrini and Y. Cao, "Human action classification based on silhouette indexed interest points for multiple domains," *Int. J. Image Graphics (IJIG)* **17**, 3 (2017).

[a] https://www.nsf.gov/awardsearch/showAward?AWD_ID=1718139.

16. C. Tian, G. Sun, Q. Zhang *et al.*, "Integrating sparse and collaborative representation classifications for image classifications," *Int. J. Image Graphics* **17**, 2 (2007).
17. A. B. Chan, Z. J. Liang and N. Vasconcelos, "Privacy preserving crowd monitoring: Counting people without people models or tracking," in *Conf. Computer Vision and Pattern Recognition* (IEEE, 2008).
18. B. Leibe, E. Seemann and B. Schiele, "Pedestrian detection in crowded scenes," in *Conf. Computer Society* (IEEE, 2005).
19. W. Ge and R. T. Collins, "Evaluation of sampling-based pedestrian detection for crowd counting," in *Workshop on Performance Evaluation of Tracking and Surveillance* (IEEE, 2009).
20. W. Ge and R. T. Collins, "Marked point processes for crowd counting," in *Conf. Computer Vision and Pattern Recognition* (IEEE, 2009).
21. V. Rabaud and S. Belongie, "Counting crowded moving objects," in *Conf. Computer Vision and Pattern Recognition* (IEEE, 2006).
22. L. Wang, X. Ji, Q. Deng and M. Jia, "Deformable part model based multiple pedestrian detection for video surveillance in crowded scenes," in *Conf. Computer Vision, Theory and Applications* (IEEE, 2014).
23. S. Zhang, R. Benenson and B. Schiele, "CityPersons: A diverse dataset for pedestrian detection," in *IEEE Conf. Computer Vision and Pattern Recognition* (2017).
24. D. Li, X. Chen, Z. Zhang and K. Huang, "Learning deep context-aware features over body and latent parts for person re-identification," in *IEEE Conf. Computer Vision and Pattern Recognition* (2017).
25. V. Kumar, A. Namoboodiri, M. Paluri and C. Jawahar, "Pose-aware person recognition," in *IEEE Conf. Computer Vision and Pattern Recognition* (2017).
26. T. Bagautdinov, A. Alahi, F. Fleuret, P. Fua and S. Savarese, "Social scene understanding: End-to-end multi-person action localization and collective activity recognition," in *IEEE Conf. Computer Vision and Pattern Recognition* (2017).
27. P. Dollar, C. Wojek, B. Schiele and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Trans. Pattern Anal. Machine Intell.* **34**, 4 (2012).
28. S. Hwang, J. Park, N. Kim, Y. Choi and I. S. Kweon, "Multispectral pedestrian detection: Benchmark dataset and baseline," in *IEEE Conf. Computer Vision and Pattern Recognition* (2015).
29. Y. Wang, D. Shen and E. K. Teoh, "Lane detection using catmull-rom spline," in *IEEE Int. Conf. Intelligent Vehicles* (1998).
30. J. Heikkila, "Geometric camera calibration using circular control points," in *IEEE Trans. Pattern Anal. Machine Intell.* **22**, 10 (2000).
31. A. Ansar and K. Daniilidis, "Linear pose estimation from points or lines," in *IEEE Trans. Pattern Anal. Machine Intell.* **25**, 5 (2003).
32. X. Ying, Z. Hu and H. Zha, "Fisheye lenses calibration using straight-line spherical perspective projection constraint," in *Comput. Vis. ACCV* **3852** (2006).
33. A. Sobral, "BGSLibrary: An openCV C++ background subtraction library," *IX Workshop de Visão Computacional* (2013).
34. N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conf. Computer Vision and Pattern Recognition* (2005).
35. M. Bertozzi *et al.*, "A pedestrian detector using histograms of oriented gradients and a support vector machine classifier," in *2007 IEEE Intelligent Transportation Systems Conf.* (2007).
36. A. Rosebrock, "HOG detectMultiScale parameters explained," https://www.pyimagesearch.com/2015/11/16/hog-detectmultiscale-parameters-explained/.

**Alexander Fuchsberger** received his M.S. in Management of Information Systems and Technology at the University of Nebraska Omaha, USA, in 2014 and his M.A. in Management, Communications & Information Technology at the Management Center Innsbuck, Austria, in 2014. He is currently pursuing his Ph.D. in Information Technology at the University of Nebraska Omaha. Since 2018 he is working at Bucknell University, Pennsylvania, USA, where he teaches courses in Computer Science.

**Brian Ricks** is the director of the Bricks Lab at the University of Nebraska at Omaha, which focuses on simulation and virtual worlds. His research interests include working directly with building managers and designers to identify fundamental research questions in the field of crowd simulation. His work has been published in Transactions on Visualization and Computer Graphics, the Visual Computer, and in numerous conferences. His work is currently funded by the National Science Foundation.



**Zhicheng Chen** is a second year Master student in Computer Science at the University of Nebraska Omaha, USA. He is a member of the Bricks lab and focus on developing queuing behavior for agents in real-time crowd simulations. He is also a skilled programmer using a variety of languages, frameworks and tools.