Synthetic Crowd Videos For Machine Learning

Anson, Tsan Kwong Wong

11 December 2016

1 Background

This report describes the synthetic crowd videos generation project I was involved. Originally I planned to work on crowd tracking with front view and dynamic obstacles. However, the learning curve and engineering work is too non-trivial, which I can hardly come up with result in three months. I then chose to work on synthetic crowd videos generation. The objective of two topic is the same: to enhance crowded scene understanding.

2 Introduction

The accessibility of commodity cameras has lead to wide availability of crowd videos. In particular, videos of crowds consisting of tens or hundreds (or more) of human agents or pedestrians are increasingly becoming available on the internet, e.g. YouTube.

One of the main challenges in computer vision and related areas is crowded scene understanding or crowd video analysis. There are a range of sub-problems in crowd understanding and analysis, including crowd behavior analysis, crowd tracking, crowd segmentation, crowd counting, abnormal behavior detection, crowd prediction, etc.

The problems related to crowded scene understanding have been extensively studied. Many solutions for each of these sub-problems have been developed by using crowd video datasets along with different techniques for computing robust features or learning the models. However, most of these datasets are limited, either in terms of different crowd behavior or scenarios, or the accuracy of the labels.

Machine learning methods, including deep learning, usually require a large set of labeled data to avoid overfitting and to compute accurate results. A large fraction of crowd videos available on the Internet are not labeled or do not have ground truth or accurate information about the features. There are many challenges that arise in terms of using these Internet crowd videos for scene understanding:

- The process of labeling the videos is manual and can be very time consuming.

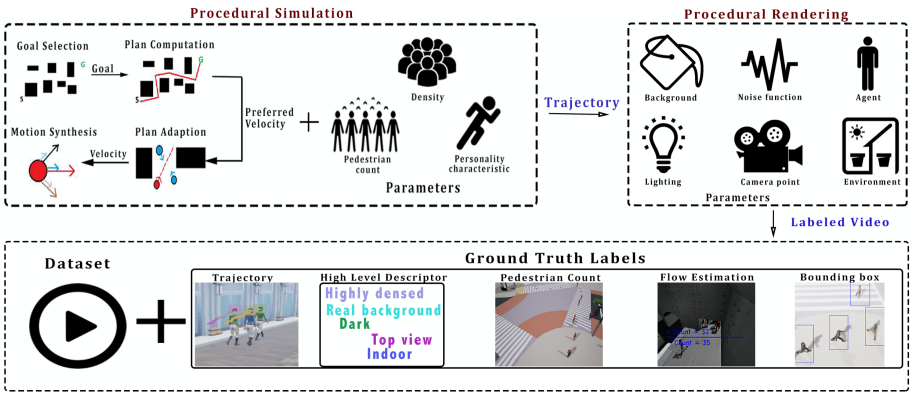

Figure 1: The framework consists of two components: procedural simulation (top left) and procedural renderer (top right). There are several parameters as show in the figure can be adjusted during the Procedural Simulation and Procedural Rendering to produce videos with a range of variety. Each final video/image consists of a number of ground truth labels (bottom). Our approach can automatically generate different videos with accurate labels.

- There may not be a sufficient number of videos available for certain crowd behaviors (e.g., panic evaluation from a large building or stadium) or for certain cultures (e.g., crowd gatherings in remote villages in the developing world). Most Internet-based videos are limited to popular locations or events.

- The classification process is subject to the social-cultural background of the human observers and their intrinsic biases. This can result in inconsistent labels for similar behaviors.

- In videos corresponding to medium and high density crowds, it is rather difficult to count the number of pedestrians exactly or classify their behaviors or tracks. This complexity is highlighted in one of the sample images in the UCF Crowd counting dataset, shown in Figure. Similar problems can arise in noisy videos or the ones recorded in poor lighting conditions.

As a result, instead of pursuing a crowd tracking algorithm, we believe that synthetic crowd videos can do the job, which provides ground truth information, for example, agents location, height, density of crowds, etc. In this report, a synthetic data generation framework will be introduced, with post-modification on agents' feature for machine learning result improvement.

3 First version - Procedural synthetic data generation framework

Crowds are observed in different situations in the real-world, including indoor and outdoor scenarios. The procedural framework is expected to be capable of

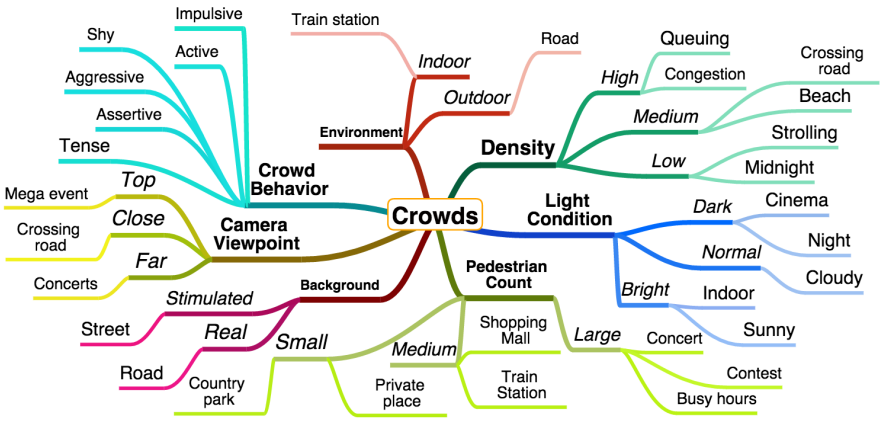

Figure 2: Hierarchical and parametric classification of crowd behaviors and renderings. Attribute Labels includes: Background, Crowd Behaviour (Personality), Camera Viewpoint, Density, Environment, Light Condition and Pedestrian Count. We can use these labels to classify different characteristics of crowds and use them in our procedural framework. Note that the list above is not exhaustive, as we can further extends attribute like Density to medium-high, medium-low, etc.

providing all types of crowd videos with appropriate ground truth labels. In this section, I will give an overview of our framework and the various parameters used to generate the videos.

3.1 Crowd Generator

Modeling the behavior of large, heterogeneous crowds is important in various domains including psychology, robotics, pedestrian dynamics, and computer graphics. There is more than a century of work in social sciences and psychology on observing and classifying crowds. Social scientists have classified the crowds in terms of behaviors, size, and distributions. According to *Convergence Theory*, crowd behavior is not a product of the crowd itself; rather it is carried into the crowd by the individuals. These observations have been used to simulate different crowd behaviors and flows.

The procedural crowd simulation framework builds on these prior observations in social sciences and on simulation methods. The overall hierarchical classification is shown in Fig. 2. Each of these labels is used to describe some attributes of the crowds or the pedestrians. In addition to the labels that govern the movements or trajectories of each agent or pedestrian, we also use a few labels that control the final rendering (e.g., lighting, camera position, brightness, field of view) of the crowd video by using appropriate techniques from computer graphics.

Finally, we can also add some noise (e.g. Gaussian noise) to degrade the final rendered images, so that these images are similar to those captured using

Figure 3: (a) Generation consists of 858 agents. (c) From a dataset, UCF dataset, it is very hard to accurately count the number of pedestrians or classify other characteristics such as density or behavior in this real-world image (i.e. generate accurate labels). In contrast, our method can automatically generate accurate labels as demonstrated in (b).

video cameras which actually have such type of noise.

Framework Design: The framework has two major components: procedural simulation and procedural rendering. The procedural simulator takes as input the number of agents or pedestrians, densities, behavior (personality of agent groups) and flow characteristics and computes the appropriate trajectories and movements of each agent corresponding to different frames. Given the position of each agent during each frame, the procedural renderer generates the image frames based on the different parameters that control the lighting conditions. Each of these input parameters corresponds to a label of the final video.

Procedural Crowd Simulation: In this section, I will give an overview of our procedural crowd simulator. Menge, a crowd simulation engine, which makes it possible to combine state-of-the-art crowd modeling techniques addressed in the previous section. In particular, we have picked ORCA as our navigation algorithm. Given the labels or high-level descriptors, the method can generate crowd movements or behaviors that fit those descriptions. These include the total number of agents or pedestrians in the scene, their density over different parts of the environment or the scene, their global and local movements, and the behavior characteristics.

Global Crowd Characteristics: In the simulation stage, we vary the global parameters, including the personality settings of different agents, density, and the number of agents used to generate different types of trajectories. The number of agents will control how many pedestrians are in the scene, and the density factor decides whether or not the pedestrians would be located very close to each other. It is essential to include different levels of overlapping in the training data set to avoid overfitting. The personality parameters allow the crowd behavior to be more natural-looking.

Crowd Movement and Trajectory Generation: A key component is simulating the crowd or pedestrian movement. Building on prior research in crowd movement simulation, we can use the property that movement specification can be decomposed into three major subproblems: agent goal selection, global plan computation, and local plan adaptation (see Fig. 1). We further elaborate on each of these components and give various possible solutions for each of these subproblems, based on the different crowd behavior classifier.

Goal Selection: In the goal selection module, we specify the high-level goal of each pedestrian. For example, the agent may want to go to a particular location in the scene or just visit a few areas, etc. It is expected that the goal can change across time and is affected by the intrinsic personalities or characteristics of the agents or the environmental factors. There is extensive literature on goal selection methods and we can use these methods in our procedural simulation framework.

Global Path Planning: Given the goal specification of each agent, we compute a collision-free trajectory to achieve that goal. The path-planning module is a high-level module that computes a preferred velocity or direction for each agent for a given time-step. We use techniques based on potential field methods, roadmaps, navigation meshes, and corridor methods.

Local Plan Adaptation: Since the path computed in the previous stage usually considers only the static obstacles, we need to adapt the plan to tackle the dynamic obstacles or other agents or pedestrians in the scene. We transform the preferred velocity computed into a feasible velocity that can avoid collisions in real time with multiple agents. Some of the commonly used motion models for local plan adaptation are based on social forces, rule-based methods, and reciprocal velocity obstacles.

Full Human Motion Computation: The main goal of high degree of freedom human motion computation or motion synthesis is to compute the locomotion or position of each agent in terms of the joint positions, corresponding to the walk cycle as well as to the motion of the head and upper body. We use standard techniques from computer animation based on kinematic, dynamics and control-based methods to generate the exact position of each pedestrian in the scene.

3.2 Procedural Rendering:

After computing the trajectory or movement specification characterized by the global parameters for each pedestrian in the video, we generate an animation and render it using different parameters. We can control the lighting conditions, resolution, viewpoint, and the noise function to lower the image quality, as needed.

Animated Agent Models: We use a set of animated agent models, that include the gender, skin color, clothes and outlook. We randomly assign these models to every agent. Furthermore, we may associate some objects in the scene with each agent or pedestrian. For example, in the case of a shopping mall, a customer may carry items he or she bought in bags; and in a theme park, there may be parents walking along with the children. These attached items could potentially obstruct the agent and change its appearance.

3D Environments and Backgrounds: Our background scenes include both indoor and outdoor environments. Ideally, we can import any scene with a polygonal model representation and/or textures and use that to represent the environment. We can also vary the lighting conditions to model different weather conditions: a sunny day could have a huge difference in appearance compared to a gloomy day. On top of that, we can also add static and dynamic obstacles to model real world situations. For instance, we could add moving vehicles into a city map and animated trees into a countryside map.

Image-space Projection and and Noise Functions: In order to render the 3D virtual world and the animated agent model, we render the image using a camera projection model: perspective projection or orthogonal projection. Typically, we render the videos with perspective projections to simulate real world videos. At this stage, we use different parameters to the projection model to obtain the videos captured from different viewpoints. In practice, video and images collected from different camera views could result in significant differences in the appearance. We also add a Gaussian noise filter with varying levels of standard deviation to emulate the digital noise in a video frame.

In our current implementation, we use the Unreal game engine for rendering and generating each video frame. It is an open source engine and we can easily specify the geometric representation, lighting and rendering information, and generate images/videos at any resolution or add noise functions. We can easily adjust the different rendering parameters available in Unreal Engine to control the final crowd rendering.

3.3 Ground Truth Labels

The two main labels related to such datasets are the pedestrian count and the trajectory of each pedestrian. And a single video can provide both kind of labels or even more when the framework is extended in the future.

This can be rather challenging for dense crowds, where generating such a labeled dataset is a major challenge. In order to accurately generate such labels, we consider each head of an agent in the video that is not obstructed by other scene objects. We compute the screen-space coordinates during each frame for every agent using the given camera parameters. We can also compute the position of lower body or full body contours. Given these head and lower body information, we can accurately compute the count and the trajectories.

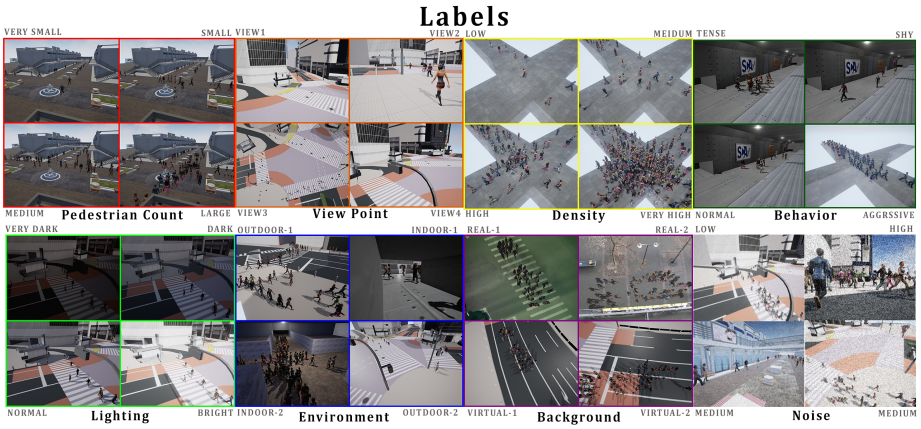

Figure 4: **Parameters used in LCrowdV**: Samples of images that illustrate the effect of changing different high level parameters in the scene. These parameters are also used as the labels.

Apart from the trajectories of the head, we also use the bounding boxes for pedestrian detection. Using the same technique mentioned above, we compute the bounding box for each pedestrian, which is centered at the centroid of the model used for each agent. This is more accurate than annotating the bounding boxes manually, especially for high density scenes.

Another major problem in crowd scene analysis is computing the crowd flows. The goal is to estimate the number of pedestrians crossing a particular region in the video. For real videos with dense pedestrian flows, it is difficult and labor intensive to compute such flow measures. This is due to the fact that there may be partial occlusion and a human operator needs to review each frame multiple times to obtain this information accurately. On the other hand, we can easily count how many agents are crossing a line or a location in the scene. However, in some of the pedestrian videos, agents could walk around or over the counting line because of collision avoidance. If we can count every agent that crosses the line, this count could increase when an agent is close to the counting line or when an agent repeatedly crosses the line. Therefore, we define an agent as crossing the line only if it has passed a particular tolerance zone or region in the scene.

In addition to these labels corresponding to the tracks, bounding boxes, flows, etc. we also keep track of all the parameters used by the procedural framework, i.e. the seven different high level parameters. As mentioned in the previous section, we have generated the videos using seven different parameters. These parameters can be used to describe the video in a high level manner, as shown in Fig. 2 and Fig. 4.

Figure 5: **An example of randomized appearance**

4 Second version - Rendering-based framework

A modification of the framework is explained below. The previous framework aims to produce crowd videos, in which high fps is a must. As a result, resolution suffers as a trade-off. The new version, in contrast, aims to generate crowd images, specific for crowd counting application. It no longer requires trajectory from Menge, instead it utilizes Unreal game engine to produce high-resolution images.

4.1 Feature modification

As mentioned previously, the appearance and walk cycle of the pedestrians play an important role for learning. Here I aim to improve the learning result by random appearance, and diverse and realistic walk cycle.

Random Appearance: Each agent's construction script consists of a set of rules, which determine the outlook of that agent. Using masks and filters, we are able to randomize the agents' skin and clothing color upon spawning. With a single character model, we will be able to obtain more than 50 different kind of combination. This enhancement increases the overall efficiency, as it saves a lot of effort and increases variety of agents. Fig. 5 shows an examlpe of using one single model to generate characters with different clothing and skin color.

Diverse and realistic walk cycle: 12 new walk cycles were added to the simulation framework, as compared to only having one in the previous version. Some of the new walk cycles even introduce the use of external objects, like phones. This modification helps to create natural and diverse motion among all

Figure 6: **An example of the resulting dataset**

agents, and avoid overfitting due to duplicate walking motion.

4.2 Image generation

A script is used to generate images with high resolution, by basically

- Identify obstacles and define walkable areas for agents to spawn.
- Spawn agents randomly with respect to appearance, motion and direction they are facing.
- Take a high-resolution screenshot with Unreal game engine built-in function, and destroy all agents.
- Repeat the process.

Fig. 6 show an example of the resulting images obtained.

5 Conclusion

This report describes two different methods for the synthetic crowd videos generation. The resulting dataset can be useful for machine learning, to enhance crowded scene understanding. I will treat the learning part as future work, and continue this project and analyze the learning result.