GATA: Supplementary Material

1. Automatic Large-Scale Data Collection

As described in Section 3.1 of the paper, we develop an automatic data collection pipeline, we illustrate it in Figure 1. Here, we detail all of the collection procedures, including sense setting generation, scene rendering and post-processing.

1.1. Scene Setting Generation

Scene setting defines an instance of a random scene for random actors playing a target animation. It includes random factors like scene location, time of day, scene weather, camera view, and human actors. Their values are randomly sampled within their own promising prior domain. Scene setting is also designated a specific target animation name which defines a semantic action to be transformed by the graphic engine as an action video clip.

Character Animation. There are 156,119 animation clips in version [1] of GTA-V that we used. We keep animations that can be played validly through game Mods only. Some of the remaining animation clips are visually the same or quite similar, and we merge them as the same animation based on the similarity of the 3D human pose sequence like [2]. Overall, the number of remaining targeting animation clips is 27814.

Human Character. There are more than 800 pedestrian models with optional clothes and accessories variations. These models include variations of gender, age and ethnic group. We spawn random characters directly after scene creation without a configurable value in the scene setting generation step.

Scene Location, Weather and Time of Day. Scene locations are randomly selected in the large map area of the whole game world. However, a location will be preferred like a road or square, which has less occlusion during the scene rendering stage. There are 9 distinct weather conditions, such as "CLEAR", "RAIN", and "FOGGY", to be randomly set with a higher probability for common weather. The time of day in the virtual world can be the factor influencing the environmental light of the scene. We randomly sample all the time but with a higher probability for daytime than nighttime.

Character Placement and Camera View. Characters or actors can be spawned randomly around the center of scene location as readers can find in column (a) of Figure 3. For the more efficient synthesis of the same action instance, we can place multiple actors in the same scene at the same time like (b) of Figure 3, which is referred as *ground* mode. However, mutual occlusion under various camera views is inevitable in this setting. Therefore, we propose a floating multi-person placement strategy as a practical solution like (c) of Figure 3, which is refer as *floating* mode. For the *ground* mode, the camera is randomly placed in an upper hemisphere centered at the scene location with a proper radius. For the *floating* mode, we put the camera in front of the scene location with a fixed distance r but lift it higher randomly in the range of $[\frac{r}{10}, \frac{r}{2}]$ relevant to the center location, which gives the pitch rotation variation between the camera and human actors. The yaw rotation is randomly set by human heading direction relevant to camera position between $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

For each target animation clip, we randomly generate a 12 (4 Location $\times$ 3 Camera) scene configuration with 32 human characters inside the same view, which results in a large-scale video synthesis task of $12 \times 27814 = 333768$ videos. The large number of scene settings are automatically split into multiple workers to run the game with Mods in parallel. A manager server handles the configuration generation, configuration dispatch and task launch in all worker nodes.

1.2. Scene Rendering

For each worker node, a task monitor process is launched by the manager server. It launches a game instance with the game Mods to start the data synthesis task automatically. It is also responsible for automatic failure recovery.

The game Mods, which is used to control the virtual game world, can load the scene configurations, render the scene as configured individually, and write out frame images and annotations automatically. The game Mods is implemented based on Script Hook V [3], a C++ library for developing game plugins and two game Mods [4] [5].

The Data Sender module encodes all video frames to videos for data compression. The resulting videos, packed together with compressed annotations and task logs, are then transferred from the worker to the data server.

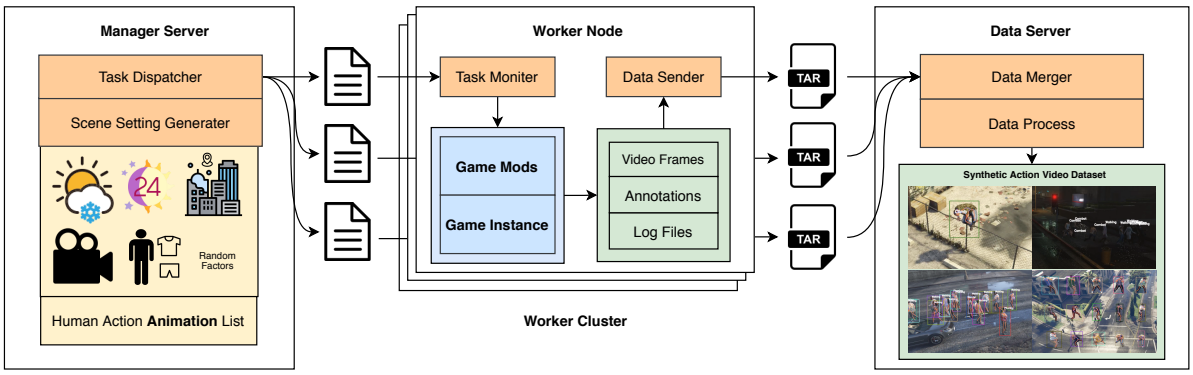

Figure 1. The data collection pipeline.

1.3. Post-Processing

A data server gathers all generated data from all worker machines. The raw synthetic data and labeling are then automatically processed for data cleaning. As the final scene location and camera are randomly sampled in the prior valid range, there is some scene occlusion in front of the camera inevitably. We filter out all the occluded instances by keypoint visibility.

We run the whole pipeline automatically using 8 worker machines and take 20 days to finish all the rendering tasks and data collection. The rendering task runs at 15 FPS with a spatial resolution of 1920×1080 for a single worker.

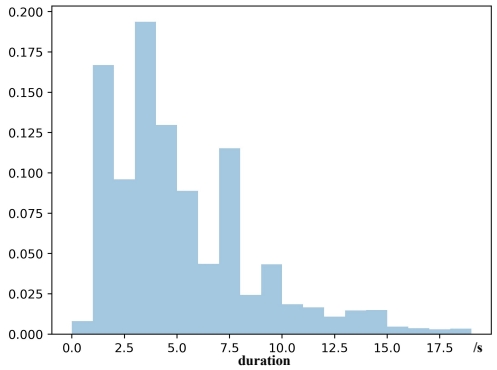

Figure 2. Duration distribution of GATA.

2. More Details about GATA

We choose the *floating* mode 3 (c) for generating the dataset used in this work. As it makes a large-scale data synthesis more feasible and gains higher qualities data instance for mutual occlusion alleviated. During training and testing, we crop the action instances from the origin videos for human-centric action understanding.

Table 1. Summary of GATA.

#Class	#Clip	#Video	Duration
27814	8.1M	256698	∼10K hours
Mean length	FPS	Res.	#Human model
4.67s	15	1080p	∼800

The numbers of clips in training/testing sets are 8002501 and 94985 respectively. Clips of different splits come from different origin videos. In Table 3 of the paper, we report the accuracy of the testing set of GATA. However, we do not pay much attention to the metric because our goal is to improve the performance of the downstream tasks. For fine-tuning on downstream tasks, the model is pretrained over the full 8.1M clips.

In addition to the properties related to the representation learning, we illustrate some other information about our GATA in Table 1. We show the distribution of the duration in Figure 2, where *Res.* means the spatial resolution.

Besides, our GATA provides informative annotations, such as class labels, 2D/3D bounding boxes, 2D/3D key points with visibility and camera parameters. We visualize this information in Figure 4. We provide some videos and annotations in the supplementary materials. Due to the limitation of the file size, we only provide the annotation of one of the videos and omit the annotation information of the other videos. Please refer to the **examples** folder for more details.

3. List of Class in Charades-Motion

We extract verbs from the category in Charades [6] and remove some abstract concepts, such as *take*. The final class is listed in Table 2.

Table 2. Class list of Charades-Motion.

hold	put	throw	undress
tidy	wash	close	sneeze
fix	open	sit	run
work	smile	watch/read/look	dress
eat	make	walk	cook
turn on	turn off	drink	grasp
pour	lie	laugh	awake

References

[1] Grand Theft Auto V Animations List (as of game patch v1734), https://alexguirre.github.io/animations-list. 1

[2] https://github.com/facebookresearch/fairmotion. 1

[3] http://www.dev-c.com/gtav/scripthookv/. 1

[4] https://github.com/fabbrimatteo/JTA-Mods. 1

[5] https://github.com/elsewhat/gtav-mod-scene-director. 1

[6] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, pages 510–526. Springer, 2016. 2

(a) single-person (b) multi-person (c) multi-person (floating)

Figure 3. Three rendering modes.

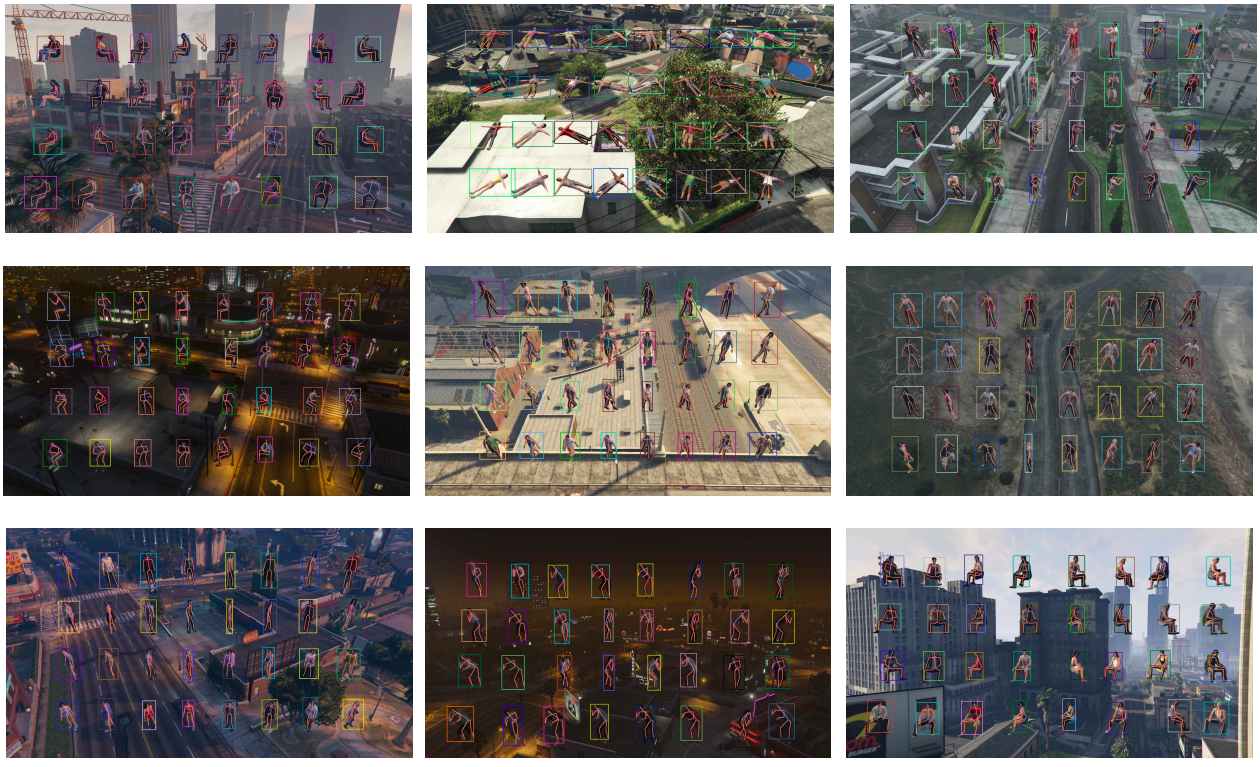

Figure 4. Informative annotations of GATA.

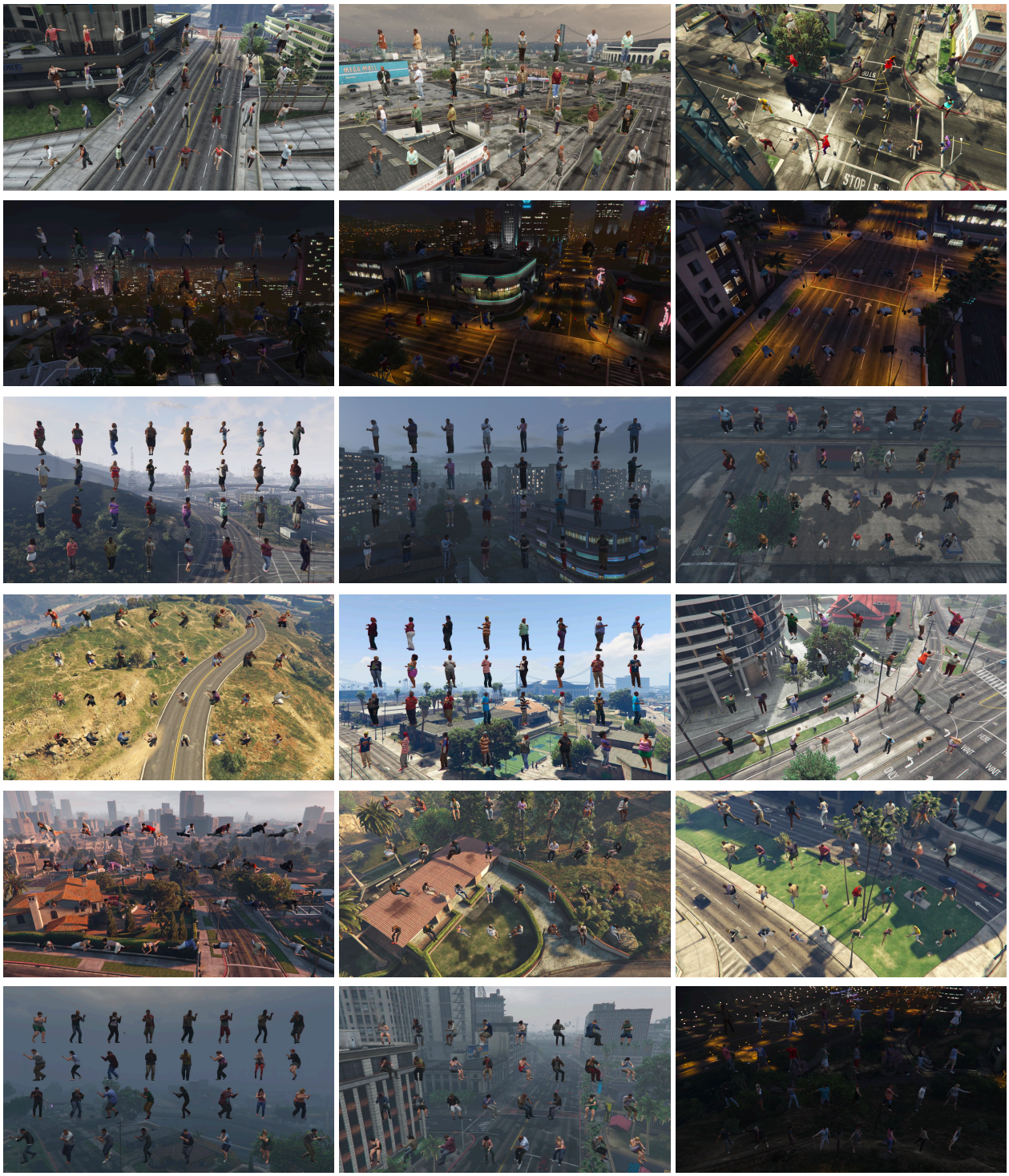

Figure 5. More data examples of GATA.