The Supplementary Materials for UnrealPerson: An Adaptive Pipeline towards Costless Person Re-identification

Tianyu Zhang, Lingxi Xie, Longhui Wei, Zijie Zhuang, Yongfei Zhang, Bo Li, Qi Tian

In the supplementary materials, we introduce more details of our data synthesis system and show some high-resolution synthesized images. We present detailed steps in Fig. 1 and elaborate in the following sections.

A. Preparing Human Models and Animations

Human model production is conducted with Makehuman [1] and Mixamo [2]. Makehuman is an open-source system and has many free assets provided by the community. Makehuman supports generating human models with adjustable parameters, such as age, weight, height, body shape, gender, ethnicity, hair, and so on. Based on a community plugin, *massproduce*, we refine it to generate human models with randomized clothes and accessories. We download all available clothing assets provided by Makehuman community and, after filtering out uncommon or unsuitable ones, we classify each clothing into fine-grained categories, *e.g.*, female-upper clothes, male-full clothes, female/male hats, *etc*. The numbers of each clothing type we use in our data synthesis process are shown in Tab. 1.

After obtaining those 3D human models, we use Mixamo, an online platform for animating static characters, to produce a rigged human skeleton that fits most of the animations provided on Mixamo. Note that all human models share the same skeleton, although they may differ in height and body shape. Therefore, we just need to animate one 3D model on Mixamo, and the rigged skeleton naturally fits all other 3D models. In other words, we are able to obtain various animations that can be used on all human models.

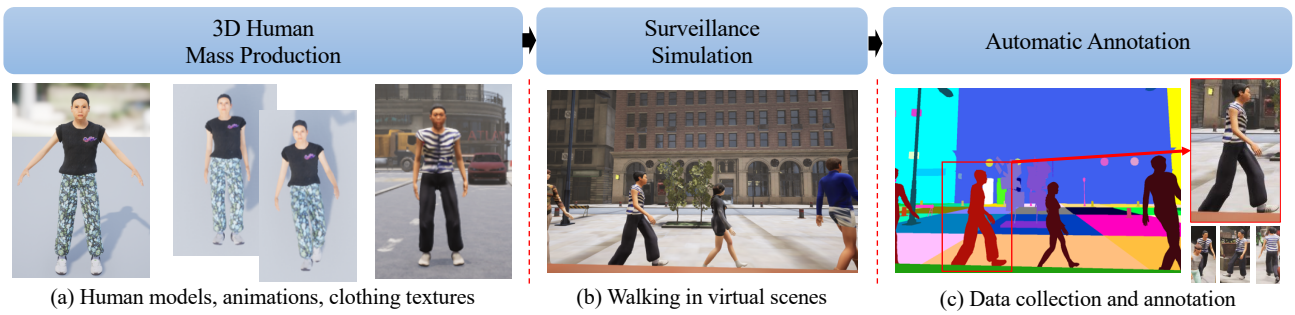
Type	Male	Female	M & F	Notes
Headwear	6	5	9	hat, flower, earphone, hairpin
Upper clothes	11	30	12	shirt, sweater, blouse, vest
Lower clothes	7	26	21	pants, skirt, shorts, jeans, tights
Full clothes	9	65	1	robe, dress, outfit
Outer clothes	1	3	2	jacket, coat, long cardigan
Shoes	2	6	15	-
Glasses	0	0	7	-
Scarf	0	0	2	-
Mask	0	0	3	-
Backpack	0	0	1	-
Handheld	0	0	4	handbag, umbrella, suitcase

Table 1. The numbers of clothes we use in data synthesis.

This saves us much effort animating static models, which usually costs a lot of time and money. We choose 4 walking and 2 idling animations for our 3D humans. In fact, there are hundreds of animations on Mixamo, including running, talking, arguing, crawling, *etc*, which can be further added to our system and enhance the reality of synthesized ReID data.

B. Preparing Clothing Textures

One of the major differences that advances the performance of our synthesized data is that we use real-world clothing textures instead of generated textures. The diversity of ready-made clothes is not enough to support large-scale ReID datasets. To address this issue, we change colors and patterns on clothes to enrich the appearance distribution of clothing templates. The appearance of clothes is



(a) Human models, animations, clothing textures (b) Walking in virtual scenes (c) Data collection and annotation

Figure 1. The steps of our data synthesis system.

defined by texture images. A pre-defined mapping function will be applied to 3D models to show the texture image on the clothes region. To keep realistic, we use real-world clothing images of two datasets, Clothing-co-parsing [7] and DeepFashion [4], as the texture images. Specifically, image patches are cropped from clothing images according to pixel-level segmentation annotation. The largest rectangle of each clothing area is cropped as a patch. These patches are also classified into several categories based on the clothing type. The extra texture images are only applied to corresponding texture slots on 3D humans. For example, if a texture image is cropped from T-shirts, it will only be used in upper body clothes of 3D humans. In total, we obtain more than 10;000 clothing textures. Later, they will be used on human models in Unreal Engine 4.

So far, all steps above, including preparing human models, animations, and clothing textures, are totally free. Moreover, if not necessary, there is no need to repeat the steps of producing human models and animations. Other foreground variations, such as poses and walking trajectories, are all implemented in Unreal Engine 4 [3].

C. Surveillance Simulation in UE4

In fact, the surveillance simulation is implemented in the same way as game development. In Unreal Engine 4, the scripts controlling game objects and scenes are implemented in *Blueprints*. We design an **actor blueprint** for our surveillance simulation. A list of 3D human models is contained in the actor blueprint, together with several lists of clothing textures. Each actor first initializes its appearance by selecting assets from the human model list and texture lists according to the input identity index. The paths that actors walk along consist of a series of target points in the game scene. The actors automatically find paths to the next target point once arrived, otherwise teleport to the next point if stuck for a long time. With our customized actor blueprint, we can generate thousands of different identities with this actor blueprint by inputting different identity indexes and target point lists.

Moreover, we implement a **level blueprint** in UE4. It has some useful functions, which we can call anytime during the game running. The level blueprint is in charge of actor spawning/deleting, game pausing/resuming, and skylight changing. We purchase four scenes from the unreal engine marketplace (as shown in Fig. 2). For a new virtual scene, we only conduct two steps to fit our level blueprint. First, we build the navigation on this scene for our actors to find paths. Second, we select some target points and assign spawning areas. In summary, it is very convenient to add new identities and new virtual scenes in our data synthesis system.

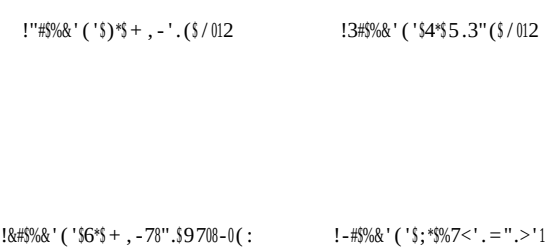


Figure 2. The four scenes used in our synthesized data.

D. Data Collection and Annotation

We design an automatic python script based on *UnrealCV* [5, 6], a plugin that interacts with UE4. This script sends commands to spawn actors in the virtual scenes. Then, they will follow the pre-defined paths walking around. The script can control the camera location and rotation, and collect RGB images together with pixel-level instance segmentation images. UE4 regards every game object as an instance, and UnrealCV provides segmentation as a display mode, which we can easily use as our annotation to separate different identities. Human models are spawned and deleted by group, *e.g.*, 200 identities as a group, to simulate crowdedness in the real scenarios. At each camera location, we change the skylight occasionally. Once we start the UE4 game and the script, everything will be done automatically. This data collection and annotation process is costless and fast compared to real-world data annotation.

References

[1] Makehuman community. Makehuman, 2020. http://www.makehumancommunity.org. 1

[2] Adobe Systems Incorporated. Mixamo, 2020. https://www.mixamo.com. 1

[3] Epic Games Incorporated. Unreal engine, 2020. https://www.unrealengine.com. 2

[4] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *CVPR*, 2016. 2

[5] Weichao Qiu and Alan Yuille. Unrealcv: Connecting computer vision to unreal engine. In *ECCV*, 2016. 2

[6] Weichao Qiu, Fangwei Zhong, Yi Zhang, Siyuan Qiao, Zihao Xiao, Tae Soo Kim, and Yizhou Wang. Unrealcv: Virtual worlds for computer vision. In *ACMMM*, 2017. 2

[7] Wei Yang, Ping Luo, and Liang Lin. Clothing co-parsing by joint image segmentation and labeling. In *CVPR*, 2013. 2