Generating Activity Snippets by Learning Human-Scene Interactions (Supplementary Material)

CHANGYANG LI, George Mason University, USA
LAP-FAI YU, George Mason University, USA

CCS Concepts: • **Computing methodologies → Graphics systems and interfaces**.

Additional Key Words and Phrases: graph generation, behavior synthesis, character animation, mixed reality

ACM Reference Format:
Changyang Li and Lap-Fai Yu. 2023. Generating Activity Snippets by Learning Human-Scene Interactions (Supplementary Material). *ACM Trans. Graph.* 42, 4, Article 1 (August 2023), 6 pages. https://doi.org/10.1145/3592096

1 ADDITIONAL CONSTRAINTS

We provide additional constraints that help to better pose instances in the scene during the optimization.

Repulsion. This type of constraint is a variation of position-based constraints. In our work, position-based constraints serve to keep associated objects close to each other. However, in special cases where a position-based constraint exists in a keyframe but is removed in the next keyframe, a repulsion constraint is automatically added into the next keyframe to repel them away from each other. For example, a character is *on the side of* a table at keyframe k, but the relation is deleted at keyframe $k+1$, a repulsion constraint is applied at keyframe $k+1$ since we expect the character to be no longer on the side of the table.

Concretely, for a position-based constraint defined as $C_{\mathrm{p}}(u,v) = 1 - e^{\lambda D - d(u,v)}$ (refer to Equation 6 in our main paper) with a target distance D and a distance function $d(u,v)$, its accompanying repulsion constraint, on the contrary, encourages instances u and v to keep a distance larger than D, and thus is defined as:

$$C_{\mathrm{r}}(u,v) = \max\left(1 - e^{d(u,v) - \frac{1}{\lambda}D}, 0\right) \tag{1}$$

Note that theoretically, this is also a position-based constraint and thus is considered during the coarse 2D optimization.

Movement continuity. This is an optional constraint. Different from previously discussed constraints, this type of constraint considers a character's position change between two adjacent keyframes. The motivation for introducing this constraint is that while a character can move from the position in a previous keyframe due to changes in associated position-based constraints, we expect such movements to be minor in specific cases to preserve some continuity. To illustrate, we use the same example as in the discussion of repulsion constraints: When the barber is repelled such that the customer is not reachable, the optimizer may move this barber far away from the original position, which can be counter-intuitive to real-life experiences. The movement continuity constraint regarding a character u at keyframe k is:

$$C_{\mathrm{mc}}(u_k, u_{k+1}) = \left[\frac{d(u_k, u_{k+1})}{d(u_k, p_k^{\dagger})}\right]^2, \tag{2}$$

where the distance function $d(u_k, u_{k+1})$ computes the walking distance for u considering its positions in keyframe k and k+1. $p_k^{\dagger}$ refers to the farthest position in the scene from u's position at keyframe k, thus $d(u_k, p_k^{\dagger})$ denotes the upperbound of u's walking distance during keyframe k to $k+1$. The walking distance is computed using an A* shortest path algorithm. This is a position-based constraint and is considered during the coarse 2D optimization.

Object orientation alignment. This is an optional constraint. To keep the tidiness of the scene, it is sometimes desired to align objects with respect to their orientations. For example, in the cooking activity in Figure 8 shown in our main paper, tables near the top side are constrained to face down and tables near the right side are constrained to face left. This constraint can be directly modified from Equation 7 in the main paper such that:

$$C_{\mathrm{oa}}(u) = \frac{1 - s(\mathbf{f}_u, \mathbf{f}'_u)}{2}, \tag{3}$$

where the orientation suitability function $s(\mathbf{f}_u, \mathbf{f}'_u)$ here evaluates the suitability between object u's forward direction $\mathbf{f}_u$ and its expected forward $\mathbf{f}'_u$. This is an orientation-based constraint and is only considered during the fine 3D optimization.

Object position alignment. This is an optional constraint. Similar to the discussion about orientation alignment, positions can also be constrained for better alignments. For example, in the "barber service" activity in Figure 8 shown in our main paper, an object position alignment constraint is applied for the three chairs such that they are placed in a line and are evenly distributed. Assuming a set of objects $\{O\}$ are expected to face the same direction using the object orientation alignment constraint, the position alignment constraint is associated with the whole group and is defined as:

$$C_{\mathrm{pa}}(\{O\}) = 1 - \frac{1}{2}\left(e^{SD_{\{O\}}^{x/y}} + e^{SD_{\{O\}}^{dis}}\right), \tag{4}$$

where $SD_{\{O\}}^{x/y}$ is the standard deviation of their coordinates on either x or y axis on the 2D floor plane depending on their expected facing direction. For example, suppose the three chairs in the "barber service" activity in Figure 8 of our main paper are expected to

Authors' addresses: Changyang Li, George Mason University, USA, cli25@gmu.edu; Lap-Fai Yu, George Mason University, USA, craigyu@gmu.edu.

0730-0301/2023/8-ART1
https://doi.org/10.1145/3592096

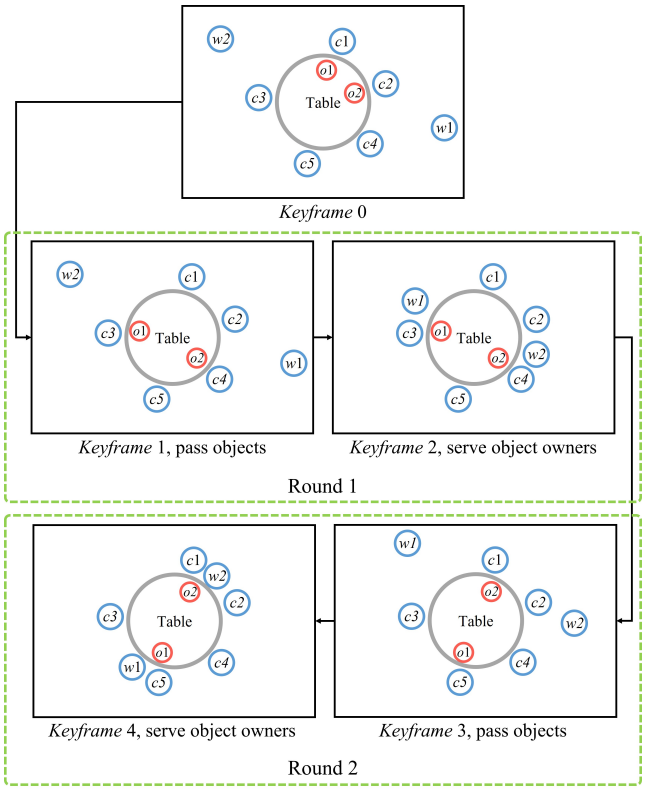

Fig. 1. An illustration of the scalable "pass object" test scenario. In each round, the objects are first passed from the two previous owners to the two next customers, and then the two waiters go to serve the new object owners.

align along the x axis. As a result, their y coordinates should be similar so their $SD^{y}_{\{O\}}$ is small. $SD^{dis}_{\{O\}}$ is the standard deviation of distances between pairwise adjacent objects. This is a position-based constraint and is considered during the coarse 2D optimization.

2 PROGRESSIVE OPTIMIZATION

In this section, we discuss our preliminary study of devising the progressive optimization framework. We create a scalable test scenario for a "pass object" game: there are five customers $c1$, $c2$, $c3$, $c4$, and $c5$, sitting near a table, and two waiters $w1$ and $w2$ who are ready to serve the customers. Two objects, $o1$ and $o2$ are placed on the table. The rule is that the waiters always go to serve customers who currently own the objects, and stay *on the side of* them. Initially, $o1$ and $o2$ *belong* to $c1$ and $c2$ respectively (such that they are within a reachable distance from $c1$ and $c2$; refer to details in Section 6.1 in the main paper). After the initial keyframe, the activity can be infinitely scaled up by iteratively extending the number of rounds r:

(1) At keyframe k, the two objects are passed to the next two customers. If they *belong* to $c4$ and $c5$ at keyframe $k-1$, the old relations are deleted, and new relations $o1$ *belongs* to $c1$ and $o2$ *belongs* to $c2$ are created at keyframe k. Meanwhile, the relations $w1$ *on the side of* $c4$ and $w2$ *on the side of* $c2$ are also deleted, thus $w1$ and $w2$ should be repelled from $c4$ and $c5$ by the repulsion constraint described in Equation 1.
(2) At keyframe $k+1$, the two waiters who were serving the previous owners of the objects, move to serve the new owners determined at keyframe k. New relations *on the side of* are added between the waiters and the new owners.

Figure 1 illustrates the setting. A scalable activity of r rounds contains $2r+1$ keyframes and $4r+9$ constraints (9 from the inital keyframe and 4 for every round). Note that when counting the number of constraints here, we only count the ones that are considered in practice because some constraints consistently appear throughout continuous keyframes, and thus can be excluded from the optimization to reduce unnecessary computations. Refer to discussions in Section 6.2.1 our main paper.

Table 1 shows quantitative results of optimizing instances' positions in the scalable activities, comparing the standard simulated annealing and our progressive optimization. The results demonstrate the effectiveness of the progressive optimization: using the standard simulated annealing, the optimization could easily get stuck at local minimums, while our progressive solver got apparently lower final costs when the activity scaled up.

Table 1. Quantitative results of optimizing instances' positions in a series of scalable activities, comparing the standard simulated annealing (SA) and our progressive optimization (ours). "Ours (X)" indicates a progressive optimization with X phases.

Rounds	No. of Constraints	No. of Iterations	Final Cost	
			Standard	Ours (X)
10	49	15,000	0.02	0.02 (3)
20	89	15,000	1.98	**0.23** (4)
50	209	15,000	11.37	**4.77** (4)
100	409	23,000	60.64	**24.09** (5)

3 ADDITIONAL EXPERIMENT DETAILS AND RESULTS

3.1 Generating and Instantiating Activity Snippets

In addition to the generated activity snippets presented in our main paper, we provide four additional activity snippets in Figure 6. Full activities with animations are visualized in our supplementary video.

We show another instantiation of the "serve food" activity (Figure 8 in our main paper) in Figure 4, where objects were substituted with objects with the same semantic labels but different geometries. We also demonstrate an example of activity transition in Figure 5, which shows a follow-up "clean up table" activity to the "serve food" activity (Figure 8 in our main paper). When generating keyframe descriptions for the "clean up table" activity, the last keyframe of the "serve food" activity was used as an initial keyframe. Furthermore, the initial 3D placements of instances were also inherited from the last keyframe of "serve food" activity and were fixed when instantiating the subsequent keyframes.

3.2 Data Preparation for Synthetic Activities

Our training data of synthetic activities is created via a simulation based on the rules and recipes. In *Overcooked*, symbolic actions such as getting or putting down an object at a locationare allowed,

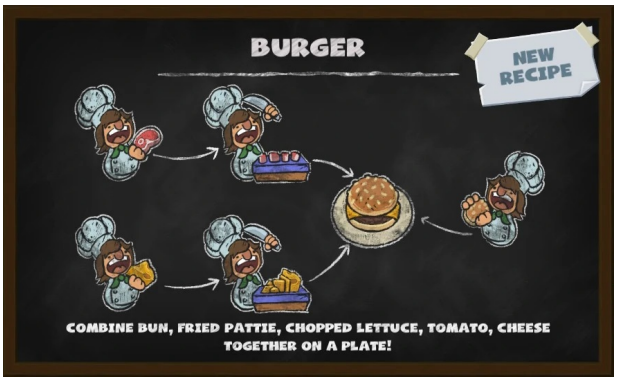

Fig. 2. The recipe of preparing a burger in the video game *Overcooked*.

triggering the functionalities of objects when required ingredients are ready. Like the example shown in Figure 2, recipes can be represented as directed graphs, in which nodes of zero in-degree are practicable tasks currently, and the node of zero out-degree is the target dish. While different types of ingredients in *Overcooked* have their own location to be accessed, we simplify the scenario and combine the source locations of all raw ingredients as "refrigerator". In our work, we only assume there are two chefs in the kitchen. We create two sets of cooking tools (e.g., stoves, chopping boards) by default such that they can collaborate with no tool usage conflicts.

When the simulation starts, both chefs are idle. Each chef randomly picks a practicable task by checking the recipe graph continuously until the whole cooking task is completed. In case a chef cannot find a practicable task, the chef stays idle. When a practicable task is done, it is removed from the graph, and its successor could become a new practicable task if its in-degree becomes zero. In practice, practicable tasks are further converted into lower-level symbolic actions. For example, for the "get beef" task, the symbolic actions should include: (1) Go to the refrigerator; (2) Get a slice of beef from the refrigerator; (3) Go to a chopping board; (4) Put the beef on the chopping board. Note that (3) and (4) are determined based on the recipe: in the example shown in Figure 2, the next step after getting the beef is to chop it. It is common that the final action sequences of the two chefs are of different lengths, and we add random "idle" actions to keep the sequences the same length.

We included 14 recipes as the activity labels in this experiment. For each label, different orders of action sequences (including random "idle" actions) allow various valid activity advancements, and we synthesized 500 pieces of training data in total.

3.3 Perceptual Study

Statistical results of ratings for each activity snippet individually are shown in Table 2. For all snippets on all metrics, the median values are all 4, and the mean values are close to 4.00. In conclusion, most participants gave position ratings on the qualities of generated activity snippets. Figure 7 shows a screenshot of the survey shown to our participants.

Table 2. Statistical results of participants' ratings for each of the 10 generated activity snippets on the four metrics. For each activity snippet, Each metric is computed using 300 ratings based on a 5-point Likert scale (1 meaning "the lowest" and 5 meaning "the highest").

Snippet No.		1	2	3	4	5	6	7	8	9	10
Reasonableness	Mean	3.77	3.82	3.90	3.87	3.91	3.88	3.93	3.96	3.81	4.00
	Median	4	4	4	4	4	4	4	4	4	4
	Standard Deviation	0.93	0.96	0.93	0.94	0.89	0.90	0.85	0.93	0.94	0.89
Intuitiveness	Mean	3.64	3.64	3.76	3.68	3.80	3.75	3.94	3.91	3.73	3.84
	Median	4	4	4	4	4	4	4	4	4	4
	Standard Deviation	0.88	0.95	0.89	0.88	0.89	0.92	0.86	0.92	0.92	0.90
Placement plausibility	Mean	3.84	3.80	3.87	3.87	3.89	3.88	3.98	3.93	3.83	3.97
	Median	4	4	4	4	4	4	4	4	4	4
	Standard Deviation	0.80	0.95	0.86	0.87	0.94	0.85	0.80	0.89	0.94	0.90
Overall plausibility	Mean	3.76	3.81	3.86	3.80	3.97	3.93	3.96	3.92	3.90	3.82
	Median	4	4	4	4	4	4	4	4	4	4
	Standard Deviation	0.97	0.95	0.92	0.93	0.83	0.90	0.85	0.89	0.89	0.89

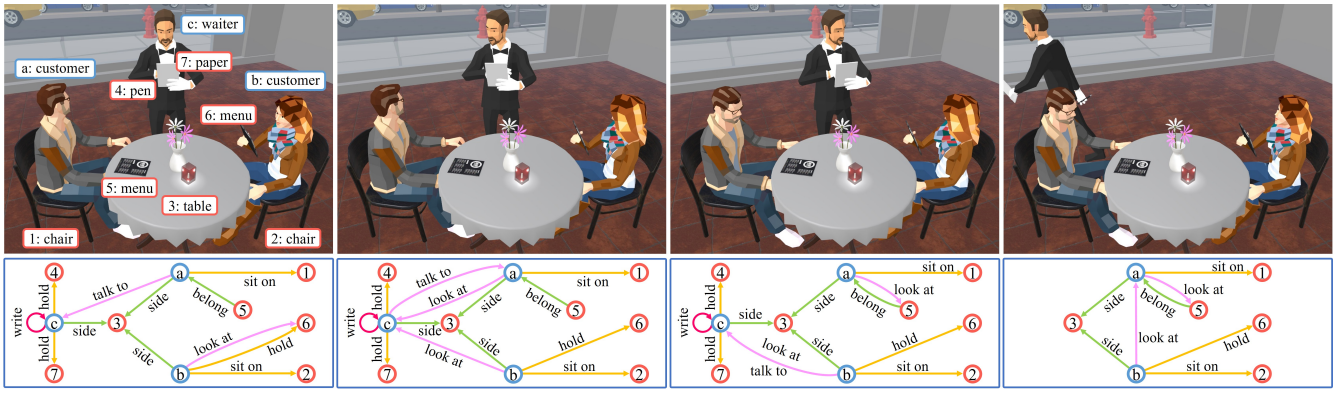

Fig. 3. Details of keyframe descriptions of the "take orders" activity shown in Figure 1 of the main paper.

Fig. 4. Substituting objects in the "serve food" activity (Figure 8 in our main paper) with ones with the same semantic labels but different geometries.

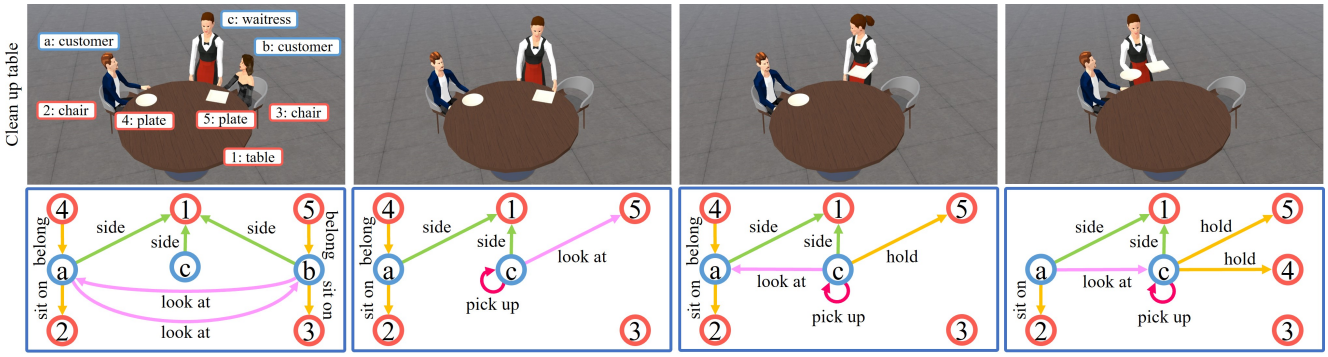

Fig. 5. A "clean up table" activity following the "serve food" activity shown in Figure 8 of our main paper.

Fig. 6. Four additional generated activity snippets. Selected keyframes are presented.

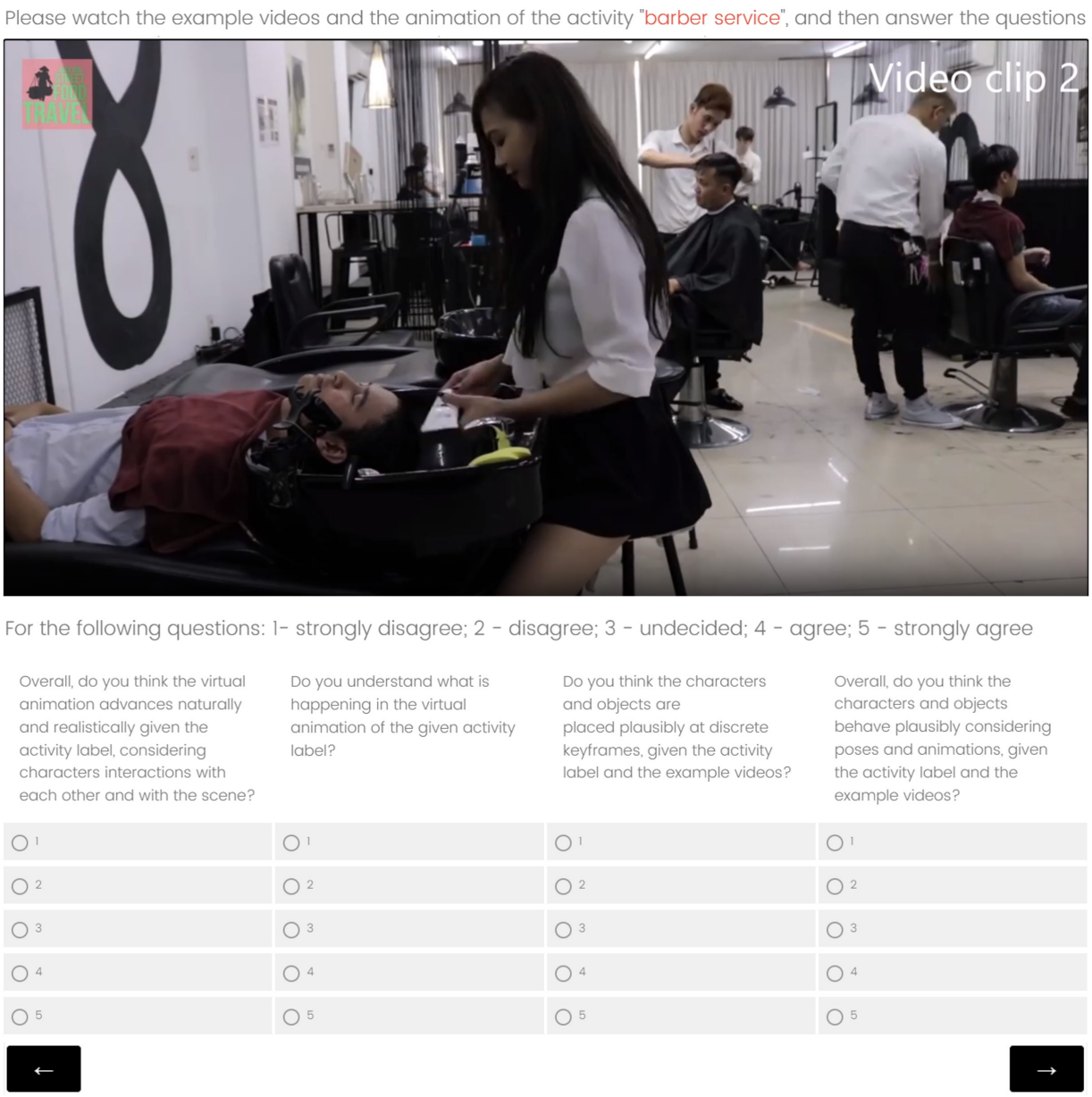

Fig. 7. A screenshot of the survey shown to participants in our user study.