EUROGRAPHICS 2019 / P. Alliez and F. Pellacini
(Guest Editors)

Volume 38 (2019), Number 2

Learning a Generative Model for Multi-Step Human–Object Interactions from Videos

He Wang[1†], Sören Pirk[1†], Ersin Yumer[2], Vladimir G. Kim[3], Ozan Sener[4], Srinath Sridhar[1], and Leonidas J. Guibas[1]

[1]Stanford University, [2]Uber ATG, [3]Adobe Research, [4]Intel Labs

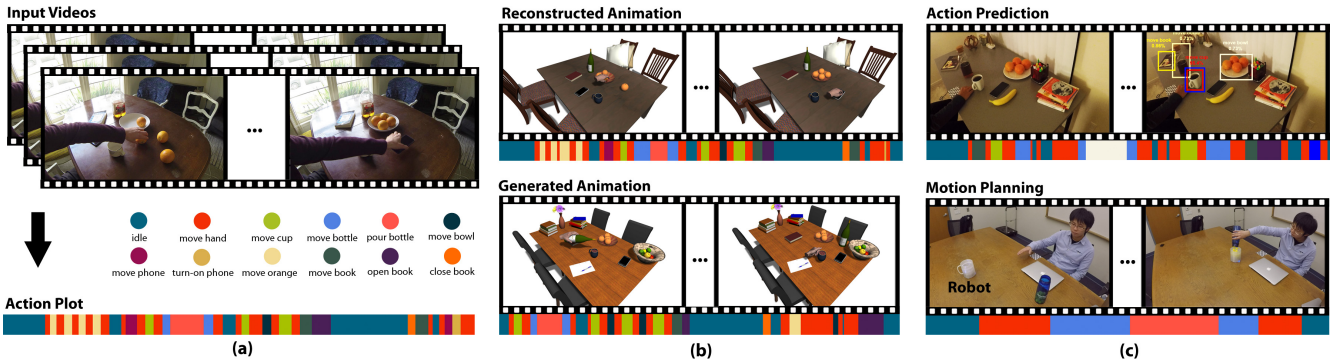

Figure 1: *Our generative model synthesizes plausible multi-step human–object interactions by learning the causal dependencies and physical constraints of individual actions from monocular RGB videos. (a) To encode interactions, we use an action plot that describes the participating objects, the object states, the elementary actions, and their causal dependencies as a set of unique operations. Based on this discretization we can efficiently learn the spatio-temporal structure of interactions to employ it for a variety of applications, including capturing the observed motions as animations (b, top), synthesizing new agent–object animation sequences (b, bottom), predicting actions that are likely to happen in the near future (c, top), and motion planning of real robotic agents (cup) in reactive environments (c, bottom).*

Abstract

Creating dynamic virtual environments consisting of humans interacting with objects is a fundamental problem in computer graphics. While it is well-accepted that agent interactions play an essential role in synthesizing such scenes, most extant techniques exclusively focus on static scenes, leaving the dynamic component out. In this paper, we present a generative model to synthesize plausible multi-step dynamic human–object interactions. Generating multi-step interactions is challenging since the space of such interactions is exponential in the number of objects, activities, and time steps. We propose to handle this combinatorial complexity by learning a lower dimensional space of plausible human–object interactions. We use action plots to represent interactions as a sequence of discrete actions along with the participating objects and their states. To build action plots, we present an automatic method that uses state-of-the-art computer vision techniques on RGB videos in order to detect individual objects and their states, extract the involved hands, and recognize the actions performed. The action plots are built from observing videos of everyday activities and are used to train a generative model based on a Recurrent Neural Network (RNN). The network learns the causal dependencies and constraints between individual actions and can be used to generate novel and diverse multi-step human–object interactions. Our representation and generative model allows new capabilities in a variety of applications such as interaction prediction, animation synthesis, and motion planning for a real robotic agent.

1. Introduction

The generation of rich *dynamic virtual environments* with complex human–object interactions is a common need in entertainment, simulation, AR/VR, and robotics. Most extant scene generation approaches, however, have focused on modeling static interaction snapshots [SCH*16], and thus have limited applicability for representing and creating dynamic scenes evolving over time. In fact, despite recent advances in activity recognition, geometric modeling and scene understanding, synthesizing plausible and meaningful interaction sequences between humans and objects remains an open and difficult problem.

† Equal contribution.

The goal of this work is to recognize and represent real world human–object interactions, and to build a model that learns the spatial, temporal, and physical constraints involved so that it can generate plausible novel dynamic interactions over time, interactions that are not directly observed in the training videos. This is challenging because, first, we need to reliably recognize objects and agent–object actions from real world observations. We use computer vision and machine learning techniques to accomplish this, learning from common RGB videos to ensure broad applicability and easy extension to future work. A second challenge lies in the complexity of learning to generate interactions in human activity spaces, due to the large variety of possible objects, their states, and their arrangements in the environment. Finally, multi-step interactions are structured over time to attain specific goals, so the generative model needs to be *consistent* with present and past states of the environment in synthesizing *plausible* action sequences.

In this paper, we present a method to automatically recognize objects and actions in RGB videos and to learn the laws governing such interactions. Based on this knowledge, we show how to encode human–object interactions using *action plots*, and how to generate plausible and diverse multi-step interactions in a variety of environments. Given an interactive environment, we describe it by: the scene that supports possible interactions (a table top setup, *e.g.,* a kitchen table, a computer desk), the objects that occur in the scene (*e.g.,* bottles, cups, books), and the actions that can change the state of the objects in the scene (*e.g.,* grasp bottle, move cup). To encode interactions, we use an *action plot* that describes the participating objects, the object states, the elementary actions, the causal dependencies of the involved actions and object states, and the individual motions necessary to perform the interaction (*e.g.,* pouring water from a bottle into a cup). Similar to a key frame in animation, each action in the plot defines a unique phase in an interaction sequence, allowing us to capture its essential features. Action plots focus on the transitions between different actions in an interaction sequence and encapsulate its key combinatorial structure. They are a symbolic and complete representation of the actions, abstracting away scene and agent specific features like motion details, appearance and geometry of the scene, thus enabling us to learn high-level semantic action constraints such as the need to grasp an object in order to move it.

We leverage state-of-the-art computer vision techniques to efficiently detect objects in videos and extend them to also detect object states and associated hand actions. To this end, we employ Faster R-CNN [RHGS15] and segment human hands using a Fully Convolutional Network (FCN) [LSD15]. Additionally, we detect state changes of objects (*e.g.,* a cup transitioning from empty to full), which helps to establish the causality of object changes and the involved actions. To recognize and temporally segment actions in videos, we employ an LSTM-based pipeline similar to Lea et al. [LVRH16]. Moreover, we learn plausible object arrangements and represent their distribution using Gaussian mixture models. Our method exhibits strong generalization behavior and is able to encode interaction in a variety of scenes with action orderings and object arrangements different from the training data. We train a generative model in the from of a Recurrent Neural Network (RNN) [CVMBB14,CGCB14] using action plots as representation. Trained using observed action plots, it can generate new plausible action plots. The RNN learns physical constraints and the causal dependencies of individual actions and can be used to predict future actions either by conditioning on a partial video as a completion task or completely from scratch as a generation task.

Finally, we show various applications of generated novel *multi-step* action sequences in prediction, animation synthesis, and semantic planning for robots. An example is illustrated in Figure 1: we observe hand–object interactions by capturing a video (top). Our method detects objects, their states, and actions allowing us to produce an action plot for reconstructing the captured motions (middle). We also build a model which can be used to generate diverse and plausible multi-step interactions (bottom), enabling previously unavailable capabilities in synthesizing dynamic scenes. In summary, our main contributions are:

- A novel representation of complex hand–object interactions as a sequence of elementary actions, *i.e.,* action plots,
- An automated method to segment actions, detect objects and their states, and hands from monocular RGB videos,
- A generative model that learns physical constraints and causal dependencies in various action sequences, and can be used to generate plausible multi-step interactions with broad applications (*e.g.,* interaction prediction, animation synthesis, robotic motion planning).

The tools developed in this work can useful be in a variety of AR/VR and robotic settings, either in isolation or as an entire system. For example, our object state estimation can inform many vision-based action detection tasks, as actions often aim to put objects in desired states. Our entire system can support applications such as (1) the generation of training videos demonstrating how certain tools can be used or complex actions performed, instantiated in a specific environment of interest; (2) the inference of interaction intent from observations and the offer of information of physical assistance to complete the interaction for the elderly or disabled; and (3) autonomous agent motion and interaction planning for specific tasks.

2. Related Work

In this section, we review related work spanning action recognition methods rooted in computer vision to activity- and animation-centric methods in computer graphics.

Human Object Interaction Perception: In the action recognition literature [HHP17], methods aim to encode the complexity of human activity spaces and the interactions therein. Although this literature is broad, it is largely based on fully supervised classification of video clips [LMSR08,EBMM03,RA09] and only a few methods exist to synthesize interactions [SPYZ11,PJZ11] or to structure the causal dependencies of human actions [FZ16].

Only more recently has it been recognized that object state plays an eminent role in learning about hand-object relationships in terms of semantic and functional state [KGS13, SS15]. Liang et al. [LZZZ16] infer the object containment relations from human actions in RGB-D videos, which can help to track highly-occluded objects [LZZ18]. Zhu et al. [ZZZ15] propose a task-oriented framework to learn the function of objects based on RGBD

videos that also considers physical concepts (*e.g.,* cracking a nut with a hammer). Several approaches examine human-object interactions. For instance, Gkioxari et al. [GGDH17] identify interaction triplets (human, verb, object) from photos by recognizing features in the appearance of humans interacting with everyday objects. Wei et al. [WZZZ17] introduce a more holistic pipeline to jointly segment objects and interaction events from video sequences. These approaches learn semantic and functional meaning from data but lack models for generating interactions.

Shape Function and Affordance: Efforts on geometric modeling and shape understanding focus on describing object relationships and object affordance through geometric features. To this end, Sharf et al. [SHL*14] introduce a method for the mobility analysis of shapes by analyzing the repetitions and relations of a shape with other objects in the scene. Bar-Avi et al. [BAR06] and Kim et al. [KCGF14] go even further and employ humanoid agents, fitted to objects, to analyze the function and affordance of shapes. Yu et al. [YDY15] reason about the object affordance of containing liquid and its best filling direction for a given 3D object, and Mottaghi et al. [MSFF17] further infer the container volume and content. More recently,Hu et al. [HvKW*16] learn a functionality model based on the co-analysis of objects and Pirk et al. [PKH*17] introduce a general purpose descriptor for understanding object function. The above methods reason about shapes and their functionality, but do not capture or understand the time-variant properties of human-object interactions.

Automated Scene Synthesis: Plausible scene generation has received considerable attention in the recent past due to its importance in graphics, robotics and computer vision. Existing methods produce plausible object arrangements for indoor scenes, by employing carefully crafted heuristics [MSL*11], automatic optimization [YYT*11], sketches [XCF*13], or procedural and probabilistic models [FRS*12, LCK*14]. Data-driven methods learn object arrangements from sensor data, by either analyzing single images [LZW*15], exploiting hybrid 2D/3D representations [MSSH14], or RGBD point clouds [SXZ*12]. In contrast, our approach models realistic dynamic scenes and plausible interactions jointly, which has previously not been addressed.

Activity-centric Scene Understanding: Commodity RGBD sensors have enabled a variety of human- and activity-centric methods for understanding and synthesizing scenes. Savva et al. [SCH*14] show that observing the behavior of humans interacting with their environment allows the capture of action maps that can be used to identify locations of interactions in unseen environments and even to infer higher-level semantic meaning of interactions [SCH*16]. Fisher et al. [FSL*15] employ 3D scans and virtual agents to model semantically-correct object arrangements allowing plausible scene synthesis. Ma et al. [MLZ*16] introduce an approach that aims to simulate the alteration of scenes through human actions. Their method learns object–object and human–object relationships from annotated photos that allows to generate messy, and thereby more realistic, yet entirely static, 3D scenes.

Human Activity Modeling and Character Animation: Due to the complexity of human motions and the variety of object arrangements, modeling human–object interactions and generating plausible animations are both difficult tasks. Many approaches exist to synthesize animations of humans [YKH04, LL04, LK05, WLO*14]. One popular approach for modeling human activities is using stochastic grammar, as in Moore et al. [ME02] for activity recognition, Qi et al. [QHWZ17] for action prediction, and in Yang et al. [YLFA15] and Dantam et al. [DS13] for robot planning and control. Motion grammar has also been used in Hyun et al. [HLL16] for character animation. These grammar-based methods are task-dependent and hence do not generalize to tasks seen in human–object interactions. Other methods explicitly define motion constraints [CH07] or physics-simulation [BSL12] to produce natural looking animations, or use probabilistic models that can be trained from small sets of manually defined example motions [LWH*12]. Agrawal et al. [AvdP16] produce high quality character locomotion by defining task-specific foot-step plans as templates for animations. Recently, Puig et al. [PRB*18] built a synthetic animation dataset of household activities and use an RNN to model and generate action programs. These approaches are similar in that they focus on human agents performing interactions, but do not learn interactions from real observations and therefore do not capture the causality between actions and object state changes.

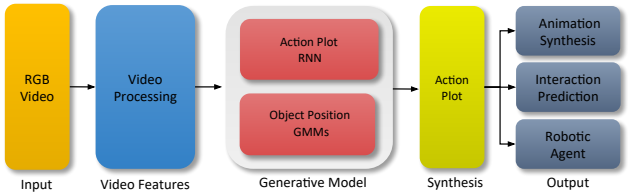

Figure 2: *Overview: we define an interaction model that allows us to generate action plots, abstract descriptions of actions that together represent an interaction. We capture RGB videos and extract features that we use to infer action labels of interaction sequences. Features and action labels are used to train an action plot RNN that allows us to generate action sequences to animate 3D animations.*

3. Overview

Our goal is to recognize and represent real world interactions, and to use them to train a generative model that can be used to produce plausible multi-step human–object action sequences. We introduce a novel representation called *action plots* to encode interactions as a set of discrete actions along with objects and object states that are involved in the action. Each action in the action plot can be seen as a key frame capturing a unique phase of the interaction. This discretization allows us to focus on the combinatorial nature of human–object interactions, while abstracting away the complexity of spatio-temporal transformations.

In Section 4, we introduce action plots and propose a novel generative model, based on a state-preserving neural network, that operates on the action plot representation. The goal of this *Action Plot RNN* is to predict action tuples including labels, target objects, state changes for hand and objects, and durations, while respecting their causal dependencies. Furthermore, we use Gaussian Mixture Models to predict object positions, while considering object-object relationships, relative distance, and movement distance.

To capture realistic properties of interactions, we use videos of people interacting with objects in common objects in indoor environ-

ments, in particular on table-top setups (*e.g.,* pouring water into a cup, putting oranges in a bowl). To create action plots that can be used to train our Action Plot RNN, we track the position, state, and motion of objects and hands and temporally segment the videos to infer action labels for the observed interaction sequences, as we describe in Section 5. We demonstrate the utility and capability of our method in Section 6 by showing how action plots and our generative model can be used in animation synthesis, interaction prediction, and the motion planning of robotic agents. Additionally, we report the results a user study in Section 6.4 to evaluate our generative model. Fig. 2 summarizes the steps of our framework.

4. Method

Human activity environments and possible human–object interactions that they can support span a large space of possibilities. While object instances, their type, or even their affordances can be described combinatorially, interacting with objects results in continuous spatio-temporal transformations, which introduces a complexity that is difficult to formalize. We account for these challenges, by proposing *action plots* as a novel representation of interactions. Each action plot is a sequence of actions performed by a human hand that causes a state change in the scene. Each action defines a unique phase in the interaction and is defined as an *action tuple* that contains an action label, duration, and participating objects with their end states and positions. Thus, a simple interaction, such as pouring water into a cup, can be described by the plot containing the following atomic actions: (1) move hand (to grasp cup), (2) move cup, (3) move hand (to grasp bottle) (4) move bottle, (5) pour water from bottle to cup, (6) move bottle.

Note that, in this illustrative example, each of these six actions is associated with a corresponding action tuple, where all six *action tuples* make up the entire *action plot*. Using the action plot allows us to abstract away the complexity of interactions in 3D spaces by a few discrete operations.

4.1. Generative Model

In this section, we discuss the details of *action plots* which we use to learn causal dependencies of action sequences and generate plausible interactions. This is based on the observation that human–object interactions can be modeled as a sequential problem. Starting with an initial state and a set of actions, we are interested in the transition probability from the given state to a subsequent state based on an action. For instance, after moving a cup, likely subsequent actions are moving the hand towards a bottle and then pouring a liquid. This is similar to other sequential problems, *e.g.,* natural language processing, where state-preserving recurrent neural networks (RNNs) have been used with success [SMH11].

Action Plots: Formally, an action tuple is defined for a single timestep as $\mathbf{T} = (a,d,o,s,p)$, where a is the action label, d is the action duration, o is the set of active objects participating, s is the end state of o, and p is the end position of o. In other words, the timestep spans duration d, and during this timestep, action a is performed on objects o and bring them to the state s and the position p. The duration, $d \in \mathbb{R}^+$, and end position, $p \in \mathbb{R}^2$, are continuous variables in time units and 2D coordinates, respectively. The action label, $a \in \mathbb{Z}_{|A|}$, participating objects, $o \in \mathbb{Z}_{|O|}$, object end state, $s \in \mathbb{Z}_{|S|}$ are defined as one-hot vectors. The size of the one-hot vectors is determined by the dictionary of all possible options for the corresponding variables A, O, and S (action label, participating objects, and object end-state dictionaries). We decouple the time-varying and the time-invariant parameters in the action tuple ($\mathbf{T} = \mathbf{L} \cup (p)$ where $\mathbf{L} = (a,d,o,s)$). This allows us to use different statistical models for each type of parameters. More specifically, we use a many-to-many the RNN to model $\mathbf{L}$ and time-independent models for p. We factorize out positions from RNN because it is very hard for RNN to encode and infer the entire 2D continuous object arrangement through the whole action sequence.

Time-Dependent Action Plot RNN for L: We structure our *Action Plot RNN* as illustrated in Fig. 3. At each timestep t, the network takes the time-dependent variables $\mathbf{L}_t$ as input and predict $\mathbf{L}_{t+1}$ for next timestep. We use the Gated Recurrent Unit (GRU) as the sequential model [CVMBB14,CGCB14]. The GRU has a latent state ρ_t that captures the information about the history of the interaction up to timestep t. In our experiments, we use a single-layer GRU with latent state size 16. Given the current inputs a_t, o_t and its latent state ρ_{t-1} from the previous timestep, the GRU outputs the updated latent state ρ_t as

$$\rho_t = g_\rho\left(a_t, o_t, \rho_{t-1}\right), \tag{1}$$

where $g(.)$ is the learned GRU function [CGCB14]. We fuse the information from GRU latent state ρ_t and other network inputs (d_t, s_t) by concatenating the features into c_t. We denote the joint state of actions, participating objects and their end object states (a, o, and s) as $z \in \mathbb{Z}_{|Z|}$ with the dictionary size $|Z| \leq |A| * |O| * |S|$. The network then predicts a probability distribution for next timestep over all valid z's as

$$P_{t+1}(z) = \frac{\exp\left(f_2^{(z)}(c_t)\right)}{\sum_{z=1}^{|Z|} \exp\left(f_2^{(z)}(c_t)\right)}, \tag{2}$$

where $f_2(.)$ outputs $|Z|$ values. Finally, our network predicts a duration $d_{t+1}^{(z)}$ for each valid joint state z as

$$d_{t+1}^{(z)} = \ln\left(1 + \exp\left(f_3^{(z)}(c_t)\right)\right), \tag{3}$$

We train the RNN using action plots extracted from real videos (see Sec. 5) of interaction sequences by associating with their $\mathbf{L}$. We use a cross-entropy loss for the predicted one-hot vector variable z, i.e. the joint state variable of (a,o,s), whereas an $L2$ loss is utilized for penalizing the difference between the ground truth duration d_{t+1} and the predicted duration of next ground truth action $\hat{d}_{t+1}^{(z_{t+1})}$. Our training loss is the sum of these two losses with the same weight. We implement the network using Tensorflow. We train the network with 64K sequences with batch size 1 and sequence length 10. Note that our RNN architecture does not take raw video frames as input by design. The use of a compact representation at input and output of the RNN module, compared to millions of variables of a video frame, significantly reduces the time spent training the RNN while increasing robustness.

RNNs are equipped with a latent state that allows them to summarize past events. Commonly, this allows RNNs to compute new

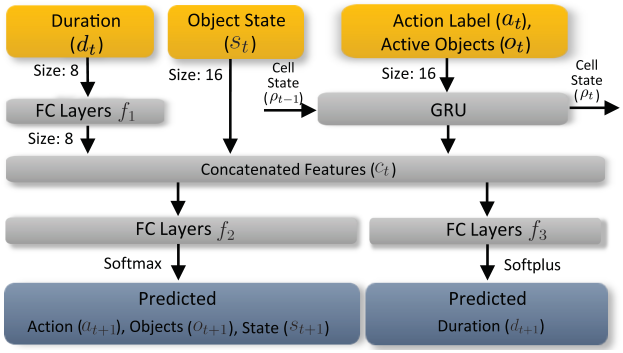

Figure 3: *Action Plot RNN: this neural network learns to predict the next joint state consisting of action labels a_{t+1}, active objects o_{t+1}, and object state s_{t+1} along with its duration d_{t+1}. The inputs (a_t, o_t, s_t, d_t) at each timestep are first embedded into vectors with the sizes illustrated in the figure. f_1, f_2, and f_3 are fully connected (FC) networks composed of three consecutive FC, ReLU, FC layers. We use softmax activation for discrete variables and softplus activation for the positive continuous variable, i.e. duration.*

states by accumulating past events with the inputs to the current cell, which results in the output and the updated cell state. When training on a small amount of data, RNNs generally tend to memorize the training sequences and hence fail to generalize to new sequences at sampling time where sequences can deviate from the training data. We hence use a small sequence length of 10, basically clearing the latent state of the RNN after 10 unroll steps. In other words, the current interaction sequence does not condition on the previous sequences and hence our RNN is free from memorizing the whole training interaction sequence. Moreover, we augment our data by on-the-fly randomly selecting action sequences when training the RNN. The random selection of action sequences also helps to reduce unnecessary long-term dependencies. As we are interested in learning short term higher-level intent, *i.e.,* one interaction sequence consisting of up to a few actions, the RNN does not need to memorize long sequences. For instance, to interact with a cup, we only need to memorize 3 actions: move hand (grasp cup), move cup, move hand (release cup). Clearing cell states and augmenting the data in the training process allows us to infer with reasonable accuracy when generating action sequences.

Time-Independent Object Position Models for p: Whenever the action is a *move* action for an object (*e.g.,* move a bottle, move a book, etc.), we need to predict a new position p_{t+1}, since it will be different from p_t. For all other actions, the object positions remain same (note that the object states might still be different). The goal of our time independent object position models is to capture the necessary information needed so as to fill-in between the beginning and end of discrete timestep for the *move* action by predicting a final destination, p_{t+1}, thus completing the entire action tuple $\mathbf{T}_{t+1} = \mathbf{L}_{t+1} \cup (p_{t+1})$.

We observe that the positions of objects are subject to a collection of strong physical priors. Certain objects may not overlap with each other, e.g. one may not find a bottle on top of a laptop. It is also common to find open bottles around cups, but uncommon

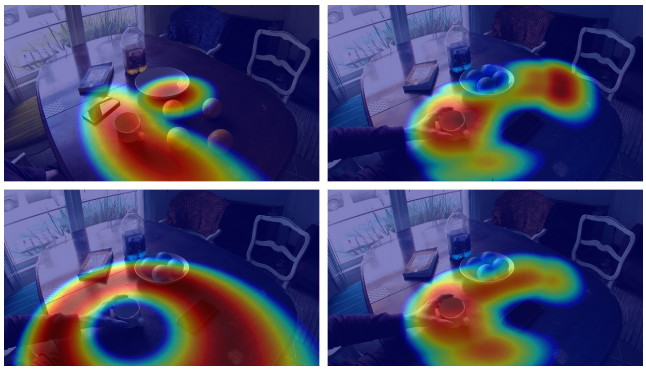

Figure 4: *Distributions as log-likelihood overlays trained through GMMs on one video. Top left: object class (cup); top right: relationship (relative distance of cup against all other); bottom left: moving distance; bottom right: final distribution used to sample a new position for objects of class cup.*

around laptops. These observations are not strongly dependent on the sequential order of previous actions. In addition, for moving an object in a given duration, the destination of this action is strongly dependent on the current object arrangement and the duration but independent of the actions that have sequentially led to the current move action. Therefore, we do not model these phenomena sequentially and instead propose the following time-independent model for the conditional probability distribution of the new position for the moving object i given the current object positions $\{p_{j,t}\}$ for each object j and the predicted duration d_{t+1} of the move action. We first discretize our activity space (*i.e.,* the table coordinate system) into K ($K = 10{,}000$ for all results in this paper) uniformly distributed square cells. Then the conditional probability for each cell position $\mathbf{x}$ is given by the following distribution:

$$P\left(p_{i,t+1} = \mathbf{x} \mid \{p_{j,t}\}, d_{t+1}\right) = \frac{1}{\mathbf{C}} P_{c_i}(\mathbf{x}) \cdot P_{c_i,\,speed}\left(\frac{||\mathbf{x} - p_{i,t}||}{d_{t+1}}\right) \cdot \prod_{j \neq i} P_{c_i,\,c_j}\left(\mathbf{x} \mid p_{j,t+1}\right), \quad (4)$$

where c_i represents the object class of the moving object i, $p_{t,i}$ is the position of the moving object i at the current timestep, d_{t+1} is the duration of the move action as predicted by the RNN. $\mathbf{C}$ is a normalization constant. P_{c_i} represents the position distribution of class c_i, and $P_{c_i,speed}$ denotes the moving speed distribution of object class c_i. P_{c_i,c_j} represents the conditional distribution of a candidate location for object class c_i, given current position $p_t^{(j)}$ of an object j in class c_j. We use Gaussian Mixture Models (GMM) for P_{c_i} and P_{c_i,c_j} with a number of mixture components between 2 and 4, whereas only a single Gaussian distribution is used for $P_{c_i,speed}$. Please see the supplementary material for analysis of these choices. Fig. 4 shows sample distributions.

4.2. Generating Interaction Sequences

We combine the trained *Action Plot RNN* and *Object Position Model* to generate interaction sequences. We first sample *Action Plot RNN* to generate a sequence of $\mathbf{L}_t$. Conditioned on each generated action and the involved objects, we use *Object Position Model*

to update the object positions p_t. Finally, with consistence check, $\mathbf{L}_t$ and p_t are coupled together to form action plot $\{\mathbf{T}_t\}$.

Sequential Tuple Generation, $\mathbf{L}_{t+1}$: We use a random sampling strategy to generate new action sequences from a trained *Action Plot RNN*.Starting from a seed action plot, the RNN can iteratively generate sequences of action plots using Eqns. 2 and Eqn. 3. At each timestep t, we feed the output $\mathbf{L}_t$ from the previous iteration as an input, generate the probability distribution of the joint states for next timestep, and then sample the distribution to get $\mathbf{L}_{t+1}$. The seed tuple can be a random but valid initial tuple to perform pure generative examples, or it can be conditioned on a few initial actions of an unseen interaction. The latter can be used to *predict the future*, and consequentially the results can be compared with the unobserved parts of the real interaction. Note that the generated actions sampled from the RNN are independent of the scene (*e.g.,* number of available objects). The output of the sampling process is a list of actions, with their associated object categories, object states, and durations. The actions are not associated with concrete objects in the scene and do not provide object positions.

Position Generation, p_{t+1}: In order to generate updated object positions, we use two modalities related to motion. First, we allow *move* actions to relocate objects in the unoccupied activity space. For a generated *move* action, we calculate the conditional probability at each cell position using Eqn. 4 and disable cells that are currently occupied by other objects in the scene. Then we sample the distribution to generate the new position. Second, we enforce object-to-object constraints. For example, in the action sequence of {move bottle, pour bottle cup, move bottle, ...}, due to the following *pour* action, we constrain the position of the *move* bottle action to be right on top of the cup. We identify these actions by scanning the next few generated states from the RNN for action labels that cause an object-object interaction (*e.g.,* pour, move in). Once such a label is detected, we select the corresponding action targets (*e.g.,* random cup, random bottle) and extend the action by adding the updated positions to the action.

Action Plot Generation, $\mathbf{T}_{t+1}$: To couple the generated actions with the scene, we convert the list of actions into an action plot. Each action in the action plot needs to be associated with an action target, the object that can be used to perform the actions, as well as a new position and a new state. As objects can be in different states, not all objects are valid action targets for a specific sequence of actions. We associate an action to an object by randomly selecting an available object in the scene that matches type and start state. Similarly, we reject actions that do not match to the scene state. For instance, the RNN might generate a *pour* action, but the scene does not contain any available empty cups. In this case, we return to the latest *move hand* action that comes before the rejected action and designate that as the entry point for resampling the RNN.

5. Learning from Videos

In this section, we describe our automatic pipeline that processes input RGB videos into an action plot that can then be used to train our Action Plot RNN. Our goal is to generate action plots containing plausible multi-step interactions that capture the physical constraints and causal dependencies in the real world. We achieve this by *automatically* learning this from RGB videos of humans interacting in a scene, providing a quick, inexpensive and versatile acquisition setup. In order to create action plots that encode complete interactions we need: (1) involved object instances, categories and positions, (2) hand position, (3) action detection and segmentation—all of which are highly challenging to extract from video. Our automatic pipeline builds upon the most recent advances in computer vision and achieves state-of-the-art accuracy on tasks such as action segmentation.

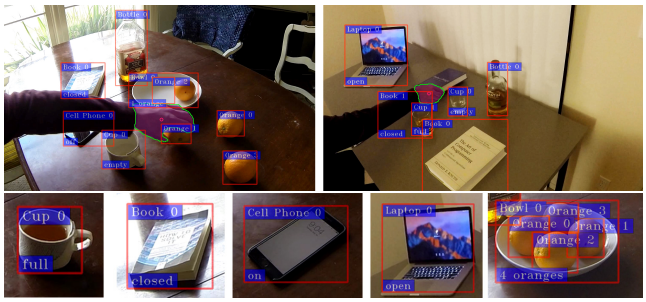

Figure 5: *Top row: two results of our tracking pipeline; our method is able to detect the type, state, and instance of objects and to segment articulated hand masks (blue overlay, green border). Bottom row: we detect object states using SVM classifiers and represent them as discrete stages.*

Data Acquisition: We acquired all videos with a GoPro Hero 5 captured at 1920×1080 px, framerate of 60 fps. In total, our dataset consists of 75 interaction videos (of up to 2 minutes), captured at 3 different locations with 3 different users. We focused on tabletop scenes since many everyday natural interaction activities occur in desk-like setups. We split the videos into a training set with 55 videos and a validation set with 20 videos. The videos show hand–object and object–object interactions with up to 10 objects and complex interactions. Our pipeline does not rely on specific lighting conditions, camera angles, or specific objects, and captures both simple hand–object (*e.g.,* moving a cup) as well as complex object-object interactions, such as stacking objects or pouring liquids. Since we are operating on monocular videos, we additionally add a checkerboard in the scenes to compute a *table coordinate frame* that allows us to register real world positions of tracked objects and hands with respect to the 3D environment.

5.1. Object and Instance Tracking

An important component of our action plots are object categories, instances, locations, and states. In order to obtain these, we first use Faster R-CNN [RHGS15] pre-trained on the Microsoft COCO dataset [LMB*14] to find candidate bounding boxes with corresponding object category labels in every frame. For instance detection, we are interested only in instances of objects that we have detected. We therefore use frame-to-frame bipartite graph matching between bounding boxes of objects in the same category. We use the Hungarian algorithm with pairwise cost (clamped to 1) based on distances between object bounding boxes. Since Faster R-CNN fails to detect objects under severe occlusions, we use a KCF tracker [HCMB12] to keep tracking these missing objects (see Fig 5). To account for the cases when people grasp and hold objects,

if the hand was within 1.5 times of the object diameter, we assume that it was grasped. In case the methods above fail to reliably detect the object, we maintain a static bounding box.

We use the camera matrix calibrated using the checkerboard to infer object position from the bottom center of its bounding box in screen space. To infer the object state we train a classifier on the content of each bounding box by using the fly-around sequences that were obtained along with the videos. Each fly-around video only shows an object in one particular state. Furthermore, we use one of the videos where the object is shown as part of an interaction, *i.e.,* we annotate object state changes and use all intermediate frames as training examples. In particular, we train a linear SVM over the features of pre-trained VGG-16 convolutional neural network [SZ14] to predict the object state.

5.2. Hand Detection

We assume that interactions are caused by human hands since they are responsible for a majority of actions in typical environments. Our goal is to infer which objects are manipulated by the hand as well as to infer object position when it is occluded by the hand. To this end, we detect 2D position of hands in the input videos. We found that Faster R-CNN is not effective at detecting hands, due to articulations, self-occlusions, and occlusions by objects. Therefore, we use a fully convolutional neural network (FCN) architecture developed for segmentation [LSD15]. We pre-train the FCN on hand masks from the GTEA dataset [FRR11] and then fine-tune on 50 frames with segmented hands from our dataset. We then run the FCN on every frame generating per-pixel masks of hands for every video frame. Finally, we compute the hand position as the centroid from the hand mask. Hand detection and object motions allow us to infer the hand state (empty, occupied), which changes once our tracking procedure detects that an object was grasped.

5.3. Action Segmentation

To generate action labels for each frame in our videos, we need to identify involved actions as well as infer the start and end times of each action (*i.e.,* action segmentation). For action segmentation, we adopt a two phased approach similar to Lea et al. [LVH16]: (1) extract meaningful image features for each frame, (2) use the extracted features to classify action labels for each frame and segment the actions. We first downsample our video data to resolution 640×360 and a framerate of 12 fps. For feature extraction, we modified the VGG-16 [SZ14] network to take a pair of RGB and motion images as the input, thereby utilizing both spatial and temporal information [LVRH16]; we use the output of block 5 as feature for each frame. The motion images consist of the difference images of 4 adjacent frames, such that for frame k, denoted by I_k, the motion image is the concatenation of $[I_{k-1}-I_k, I_{k+1}-I_k, I_{k-2}-I_k, I_{k+2}-I_k]$. The VGG-16 network is pretrained on the MS COCO dataset [LMB*14] and further fine-tuned on our videos for a frame-based action classification task. On the per-frame action detection task, we obtain an overall 85.04 % top-1 accuracy on our validation dataset among 22 action categories. Please see supplementary material for the details of data annotations.

However, frame-base action classification often suffers from over-segmentation and the lack of local temporal coherence. So, we further feed the VGG16-B5 features to a Long Short-Term Memory (LSTM) network with a cell size of 16 to aggregate temporal information temporally. Our LSTM network achieves 85.02 % top -1 accuracy. In addition to frame-wise accuracy, an edit score that takes into account the number of actions and their relative orders is commonly used in action segmentation [LVH16]. Frame-wise classification with our VGG-16 network gives an edit score of 52.47 while the LSTM network achieves an improvement with an edit score of 68.80 on the validation set. Our choice of the LSTM network was informed by experiments that showed that even state-of-the-art methods like Temporal Convolutional Network [LVH16] achieved lower frame-wise accuracy than our approach.

To further prevent any residual over-segmentation artifacts, we remove actions that are shorter than 3 frames and filter actions that involve non-existing object categories in the object detection results. This post-processing further boosts our method to a 84.41% top-1 accuracy and an edit score of 83.57, which can both be considered state-of-the-art (Figure 6).

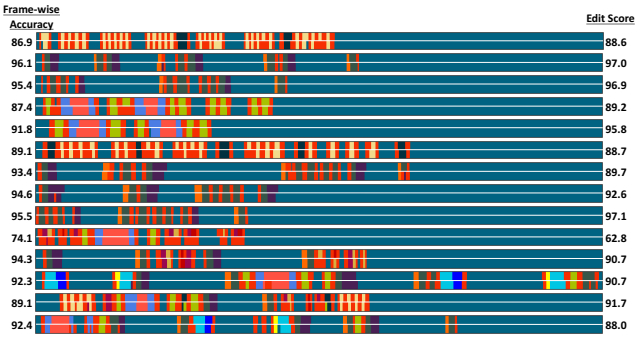

Figure 6: *The result of our action segmentation for videos in our validation set. For each row, the top half shows ground truth segmentation while the bottom half shows the prediction result.*

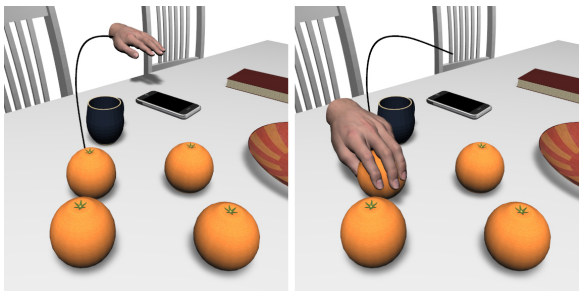

Figure 7: *A generated motion path and pose change of a grasp action: the action tuples serve as keyframes to procedurally generate motion paths based on splines (black line) and to interpolate between different hand poses that are stored with the object.*

6. Results, Experiments, and Applications

In this section, we show the results of our method and discuss new applications that are enabled by action plots and our generative model including animations synthesis, action prediction, and the motion planning for a robot. We also report the results of a user study to evaluate the quality of the generated interactions.

6.1. Animation Synthesis

Fig. 14 shows the capabilities of our framework for animation synthesis. We capture people interacting with similar objects in a tabletop configuration, but in different environments. We then show how our Action Plot RNN model learns the causal dependencies of individual actions and is able to generate novel plausible action sequences that were never observed in the data. For the example in Fig. 7, we show the motion path and articulations of a hand grasping an orange. Each action tuple can be seen as a key frame in an interaction sequence, where the corresponding object positions are used to parameterize spline functions for the animations (bottom row). Furthermore, our framework allows to capture, learn, and generate discrete state changes for various objects. While for some objects, *e.g.,* a phone, state changes are only binary, more complex interactions require multiple state and position updates (*e.g.,* put-on, bowl, pour).

Our model focuses on learning action sequences, and does not have a notion of concrete object instances. This allows us to use the generated action sequence for more objects than were used to train the RNN. This is shown in Fig. 10. We use the interaction of a bottle and two cups to generate a sequence of two bottles and five cups. Our action plot RNN also generalizes to new environments as shown in Fig. 8. Furthermore, our position generation respects static objects in the scene. While the initial scene (left) does not have any static objects, the other scenes (middle, right) contain a plate with cookies, a toaster, a notebook, stacks of books, and a fruit bowl as static objects, that block the table area for movements.

6.2. Action Prediction

As a causality-aware sequential model of interaction, our *Action Plot RNN* intrinsically has the capability to predict the near future of interactions. Fig. 11 shows an interaction prediction experiment to showcase this capability. To continue an interaction sequence observed in a video, we first select a timestep as the starting point for the prediction. For the known sequence of the video, we obtained the action labels and the active objects through our action segmentation model described in Section 5 and track all objects to store their states and positions. We then initialize the cell state of the RNN by using the observed data as input. This conditions the network on the action sequences and the available objects in the video. Finally, we generate an action plot by associating the tracked positions with the provided action labels of the known part of the video; for the predicted action labels we generate new object positions by employing our motion model described in Section 4.2. Fig. 11 shows the captured video (first row), the reconstructed sequences (second row), and two generated sequences (third and fourth row). While all sequences start with the same interactions for the first half of the sequences (indicated by the black line), the generated sequences diverge by showing different, but plausible interactions for the scene. Fig. 12 further visualizes the probabilities among the possible actions. Our model can give good action predictions with high confidence for consecutive relevant motions, e.g. the sequence shown in the Fig. 12 (move bottle close to cup, pour bottle to cup, move bottle back, then use cup).

To further evaluate the accuracy of action prediction, we evaluated

Figure 8: *An example of using the same generated action plot in multiple scenes. As the action plots only define interactions in abstracted from, we can replace objects by other – semantically similar – shapes (e.g., bottle and milk box). The position generation respects static objects in the scene. While the initial scene (left) does not have any static objects, the other scenes (middle, right) contain a plate with cookies, a toaster, a notebook, stacks of books, and a fruit bowl, that block the table area for movements.*

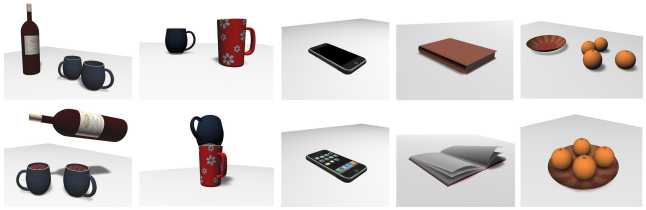

Figure 9: *State transitions for some objects: the action plot RNN is able to learn state transitions. Depending on the object class, we learn up to 5 different states. From left to right: cups change from empty to full and can be stacked, phones switch on and off, books open and close, and bowls can contain a number of objects.*

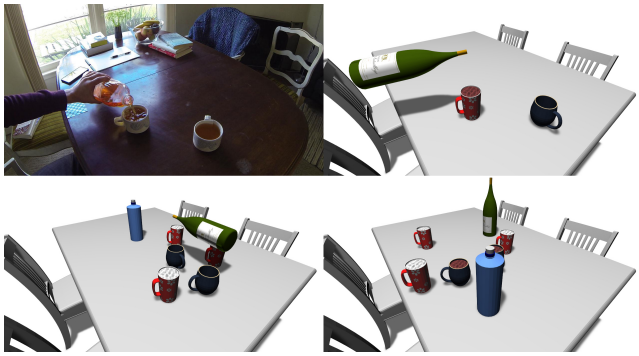

Figure 10: *As our model focuses on the causal dependencies of actions, but not on concrete object instances, a generated action sequence can be transferred to animate multiple objects of the same type. The top right figure shows reconstruction; The bottom shows transferred results on two bottles and five cups.*

our Action Plot RNN using 3 action sequences from test videos involving 113 different actions and covering all the objects and action modalities. For each action in the action sequence, we feed the observed sequence to our Action Plot RNN and forward it once more to predict the joint label of action, active object and object state. The dictionary size of the joint label of action, active object and object state is 34. We obtained 56.8% on the top-1 accuracy and 88.5% on the top-5 accuracy.

6.3. Simulation and Motion Planning for Robots

Our proposed method has applications beyond generating animations. In this section, we describe a proof-of-concept investigating

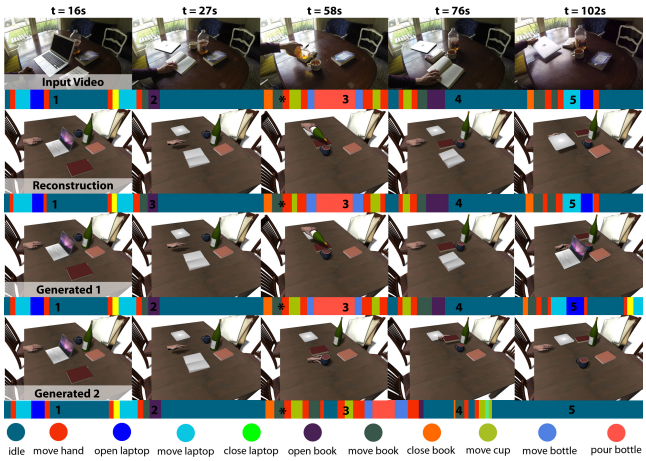

Figure 11: *We use our Action Plot RNN for continuing the interactions observed in a video (top). Our model allows to reconstruct the interaction sequence (second row) and predicts plausible actions (third and fourth row). Colored bars show the manually labeled interactions in the video, the result of the action segmentation, and the generated sequences. The black line in the action plots indicates the time before (left half) and after the sequences diverge. (right half). The ∗ indicates the position of the frame in the sequence.*

the application of our method on smart and reactive environments. The goal of such environments is not only to detect human activities but also to react to them. An important requirement is the ability to predict time critical reactions a few seconds ahead [SS15]. To demonstrate the effectiveness of our method in these situations, we developed a *smart cup*. We extended the capabilities of an ordinary cup by building a robot cup which is equipped with a remote controlled differential drive.

We implemented a simple reactive control algorithm using our method. The physical motion of the cup is computed fast using OpenMP [ŞMK12] framework and executed with a PID controller. Our object detection, state estimation, and Action Plot RNN all run at an interactive rate of 3 fps. We use the Action Plot RNN to predict future states of objects and human actions to control the robot. If the Action Plot RNN predicts an action which uses the cup as an object, we react to it based on the type of interaction involved (see Fig 13). *Unary interactions* do not involve direct hand–object manipulation but only describe indirect intent, *binary interactions* involve direct hand–object manipulation, and *ternary interactions* involve hand interacting with multiple objects.

In Fig 13, **Return Home** is an example of a unary interaction where our algorithm does not predict any direct action due the hand. However, the cup is not at a highly likely position (as described by our time-independent GMM) and therefore it moves to a higher likelihood position. **Summon Cup** (row 2, left) is an example of binary interaction where our method predicts a possible grasp of the cup by the hand. Therefore, the smart cup moves in the direction of the hand to prevent users from needing to overreach. However, if our method detects that the hand is previously holding a book (row 2, right), the smart cup does not reach since the physical constraint of holding only one object at a time is implicitly learned by our

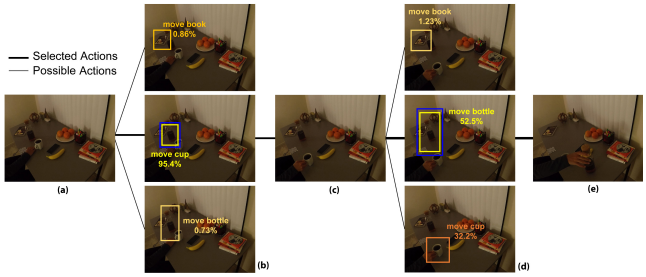

Figure 12: *Based on the current scene state, the Action Plot RNN predicts multiple possible actions that can be mapped to the detected objects in the scene. The agent selects one of the possible actions and the scene transitions to the next state, then the process starts over.*

method. Finally, **Summon Cup to Pour** (rows 3 and 4) show examples of ternary interactions where the hand, smart cup, and a bottle interact in more complex ways. When a filled bottle is moved, the smart cup automatically positions itself for easier pouring. However, when we detect that the bottle is empty, the smart cup does not react. This level of semantic planning is only possible with an understanding of complex human–object interactions.

6.4. User Study

Evaluating actions generated by our method presents a challenge since there is no correct sequence of actions but only *plausible* ones. Since this is subjective, we conducted a user study to evaluate plausibility, *i.e.,* are the motions generated by our method physically accurate and follow causal dependencies in actions. We presented participants with randomly chosen animation sequences rendered from several action plots and asked them to rate plausibility in a seven-level Likert scale. The presented action plots were randomly chosen from three conditions: (1) random action plots with no causal dependencies and physical constraints, (2) action plots reconstructed from real videos, and (3) action plots generated by our action plot RNN. Each user was presented with 16 animations in random order chosen from a set of 24 animations (8 from each action plot type). In total, 35 users (12 female, mean age was 27.9 with a stddev of 6.2) participated in our study which was administered online. The mean plausibility scores (1-strongly agree, 7-strongly disagree) were 5.1 (std=0.5, median=5.2) for random action plots, 2.7 (std=1.3, median=2.4) for reconstructed action plots, and 2.9 (std=0.7, median=2.6) for generated action plots. Two two-tailed student t-tests rejected the null hypotheses that either the reconstructed or the generated animation sequences came from the same distribution as randomly generated animations ($p < 0.003$). Furthermore, no statistically significant difference in user preference was found between the reconstructed and generated sequences. This indicates that our method is able to generate action plots that capture the causality and physical constraints of real world actions.

7. Implementation and Performance

We now discuss the performance and runtime details of our implementation which will be publicly released.

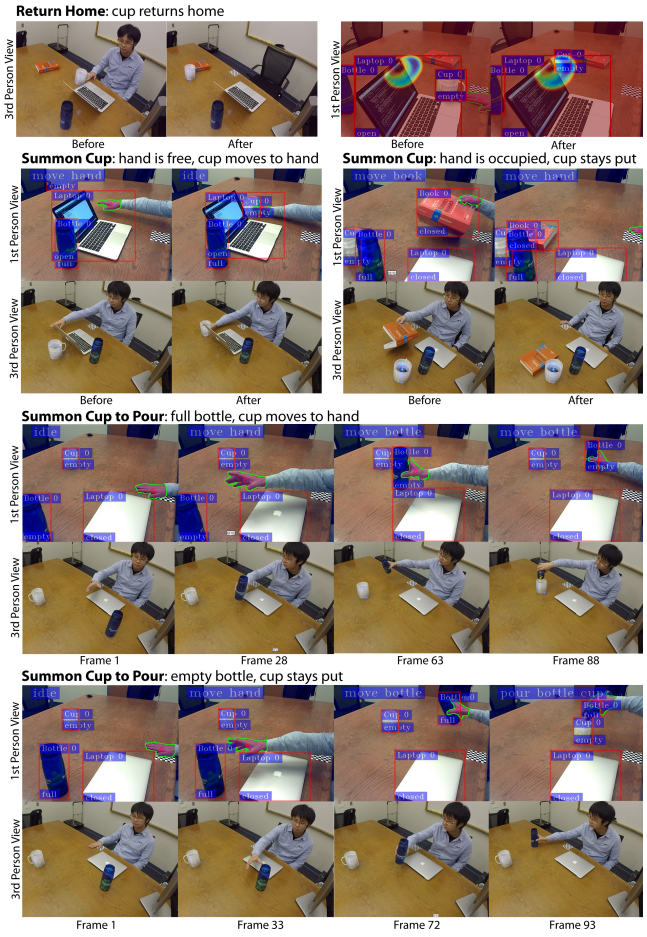

Figure 13: *We use our model in order to predict future human–object interactions and use these predictions in semantic planning of a robot cup. In summon cup (left), our method predicts grasping action and cup moves in the direction of the hand. Although hand does a similar motion in summon cup (right), the cup does not move since the hand is occupied with a book and our method does not predict grasp. In summon cup to pour (top), our method predicts pour-to action hence cup comes near the bottle. Similarly, in (bottom), bottle does a similar motion; however, our method does not predict pour-to since the bottle is empty. Hence, the cup stays put. In return home, our model does not predict an action but predicts a very low likelihood for the position of the cup; hence, cup moves to somewhere more likely when all agents leave the scene.*

Objects and Actions: We use 8 different object classes (bottle, bowl, cup, laptop, book, phone, orange, banana) and implemented 11 action modalities (idle, move, pour, drink, open, close, turn, turn-on, turn-off, use). In total, we have 22 unique interactions. While the most common action modalities are important for almost all interactions, *i.e.,* move hand and move object, others are used to initiate more complex state changes, *e.g.,* pour, open, turn-off. Please see supplementary materials for more statistics.

Scenes and Animations: Based on the scene description, we populate a 3D environment (table-top setting) with objects as observed in the input videos. We define scenes, by specifying the number and class of objects and their states. Furthermore, by detecting object type and location, we can use a scene synthesis database (*e.g.,* Fisher et al. [FSL*15] to automatically populate scenes with objects of the specified class. An action tuple describes an action to manipulate up to two participating objects. Simple actions, such as a *move*, can be resolved by selecting the object and by generating a new position based on the GMMs. More complicated actions, such as *pour*, *put-on*, or *use*, are handled individually. For a *pour* action we place the end location of the bottle to the top center of the bounding box of the participating cup and compute a rotation axis as the cross product of the direction vector and the up-axis. Similarly, we implement the other actions, *e.g.,* for stacking objects or moving objects into containers.

Hand Articulations: While we detect the position of the hand, estimating hand articulation is a highly challenging problem [SOT13] which we do not address. We instead focus on plausible hand articulations that make the synthesized animations realistic. To this end, we created a high resolution fully rigged and textured model of the hand. We use a reduced set of 26 degrees of freedom to control the articulations of the hand. For each category of object in our database, we used the Leap Motion controller to create a few exemplar grasps that can be used to hold the object. We then use the action transitions in the action plot to smoothly transition from different hand grasps using linear interpolation. The resulting animations (shown in the supplementary video) exhibit realistic hand motion without the need for full hand pose estimation.

Action Reconstruction: To reconstruct interaction sequences from video, we track the type, instance, and location of all objects in the scene. We iterate over the action labels provided with the video and keep track of state and position updates of the objects that change their location or state. Finally, we create action tuples and store them in a sequential order in the action plot.

Performance: For rendering animations, we implemented a framework in C++ using OpenGL on a desktop computer with an Intel Xenon CPU clocked at 3.7 GHz and 32 GB of RAM. We did not specifically optimize our code and rendered all results in our framework. We train the *Action Plot RNN* with 1000 epochs which allows the loss to converge within 30 minutes. Querying the network to generate 100 action events takes around 300 ms.

8. Limitations and Future Work

This work is the first step towards generating plausible dynamic virtual environment where agent actions are driven by semantic context. This problem requires jointly modeling functional and geometric aspects of the world as well as human intent, and thus many open challenges remain. First, while our current formulation of dynamic interactions into action plots makes the problem more tractable but also prevents us from modeling explicit physical laws. For instance, not every action in a plot might be physically plausible. Similarly, the scene description and the involved potential state changes due interactions are also essential for long-term planning of more structured actions. At a lower level, our method is currently limited by the restrictions imposed by capturing data from video. We do not include the orientation of objects in our action plot formulation. Finally, understanding high-level human in-

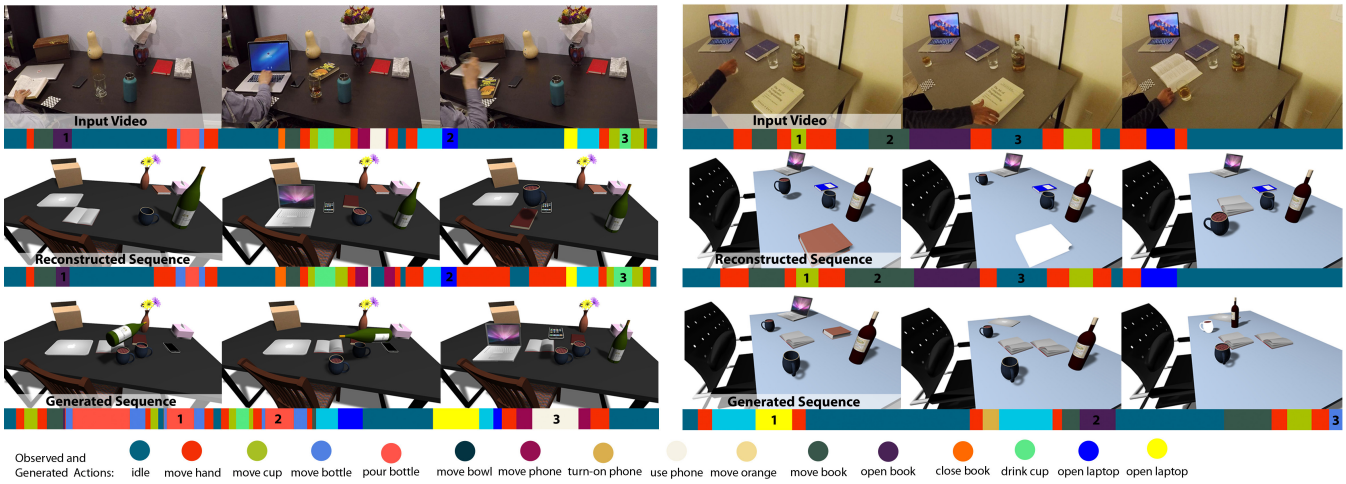

Figure 14: *Two examples highlighting the capabilities of our framework for animation synthesis. We capture videos of people interacting with objects in a table-top setup. Our method learns the causal dependencies of the individual actions from the videos (input videos) and allows to reconstruct the observed interactions (reconstructed sequence). Moreover, we generate novel interaction sequences (generated sequence) using similar objects. The colored bar under each sequence represents the ground truth labels of actions in the observation (input video), the result of the action segmentation (reconstructed sequence) and the synthesized actions (generated sequence). The numbers in the action plots indicate the frame (left to right) corresponding to this action. For the sequence at the top we also animate the hand.*

tent (*e.g.,* make breakfast) is currently not a capability we enable because of the longer time horizons invovled in these activities. While our model is currently not able to handle these cases, it can enable such semantic reasoning in the future.

9. Conclusion

We introduce a new generative model that learns to synthesie plausible human–object interactions that respect causal dependencies and physical constraints in interaction sequences. We encode interactions compactly using an *action plot* that describes a temporally sequence of atomic actions along with object positions, states and categories. We use this representation to learn interaction sequences by observing real interactions from videos. Given the initial state of a scene, our *Action Plot RNN* allows to predict actions by learning transition probabilities to subsequent states. By sampling the latent space of the RNN, we can generate new interaction sequences with similar properties as to those observed in the videos. We animate the interactions by parameterizing their state changes and motion transitions. This allows to generate plausible 3D animations of both hand–object and object–object interactions useful for creating content for synthetic scenes. It also enables previously unseen capabilities in complex motion planning for a robot in smart environments.

Acknowledgements

This research was supported by a grant from Toyota-Stanford Center for AI Research, NSF grant CCF-1514305, a Vannevar Bush Faculty Fellowship, and A Google Focused Research Award.

References

[AvdP16] AGRAWAL S., VAN DE PANNE M.: Task-based locomotion. *ACM Trans. Graph. 35*, 4 (2016), 82:1–82:11. 3

[BAR06] BAR-AVIV E., RIVLIN E.: Functional 3d object classification using simulation of embodied agent. BMVC, pp. 307–316. 3

[BSL12] BAI Y., SIU K., LIU C. K.: Synthesis of concurrent object manipulation tasks. *ACM Trans. Graph. 31*, 6 (2012), 156:1–156:9. 3

[CGCB14] CHUNG J., GULCEHRE C., CHO K., BENGIO Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014). 2, 4

[CH07] CHAI J., HODGINS J. K.: Constraint-based motion optimization using a statistical dynamic model. *ACM Trans. Graph. 26*, 3 (2007). 3

[CVMBB14] CHO K., VAN MERRIËNBOER B., BAHDANAU D., BENGIO Y.: On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014). 2, 4

[DS13] DANTAM N., STILMAN M.: The motion grammar: Analysis of a linguistic method for robot control. *IEEE Transactions on Robotics 29*, 3 (2013), 704–718. 3

[EBMM03] EFROS A. A., BERG A. C., MORI G., MALIK J.: Recognizing action at a distance. 2

[FRR11] FATHI A., REN X., REHG J. M.: Learning to recognize objects in egocentric activities. 7

[FRS*12] FISHER M., RITCHIE D., SAVVA M., FUNKHOUSER T., HANRAHAN P.: Example-based synthesis of 3d object arrangements. *ACM Trans. Graph. 31*, 6 (2012), 135:1–135:11. 3

[FSL*15] FISHER M., SAVVA M., LI Y., HANRAHAN P., NIESSNER M.: Activity-centric scene synthesis for functional 3d scene modeling. *ACM Trans. Graph. 34*, 6 (2015), 179:1–179:13. 3, 10

[FZ16] FIRE A., ZHU S.-C.: Learning perceptual causality from video. *ACM TIST 7*, 2 (2016), 23. 2

[GGDH17] GKIOXARI G., GIRSHICK R., DOLLÁR P., HE K.: Detecting and recognizing human-object interactions. *CoRR* (2017). 3

[HCMB12] HENRIQUES J. F., CASEIRO R., MARTINS P., BATISTA J.: Exploiting the circulant structure of tracking-by-detection with kernels. In *ECCV* (2012), Springer, pp. 702–715. 6

[HHP17] HERATH S., HARANDI M., PORIKLI F.: Going deeper into action recognition: A survey. *IVC 60* (2017), 4 – 21. 2

[HLL16] HYUN K., LEE K., LEE J.: Motion grammars for character animation. In *CGF* (2016), vol. 35, pp. 103–113. 3

[HvKW*16] HU R., VAN KAICK O., WU B., HUANG H., SHAMIR A., ZHANG H.: Learning how objects function via co-analysis of interactions. *ACM Trans. Graph. 35*, 4 (2016), 47:1–47:13. 3

[KCGF14] KIM V. G., CHAUDHURI S., GUIBAS L., FUNKHOUSER T.: Shape2pose: human-centric shape analysis. *ACM Trans. Graph. 33*, 4 (2014), 120:1–120:12. 3

[KGS13] KOPPULA H. S., GUPTA R., SAXENA A.: Learning human activities and object affordances from rgb-d videos. *Int. J. Rob. Res. 32*, 8 (2013), 951–970. 2

[LCK*14] LIU T., CHAUDHURI S., KIM V. G., HUANG Q.-X., MITRA N. J., FUNKHOUSER T.: Creating Consistent Scene Graphs Using a Probabilistic Grammar. *Trans. on Graph. (Proc. of SIGGRAPH Asia) 33*, 6 (2014). 3

[LK05] LAU M., KUFFNER J. J.: Behavior planning for character animation. ACM, pp. 271–280. 3

[LL04] LEE J., LEE K. H.: Precomputing avatar behavior from human motion data. pp. 79–87. 3

[LMB*14] LIN T.-Y., MAIRE M., BELONGIE S., HAYS J., PERONA P., RAMANAN D., DOLLÁR P., ZITNICK C. L.: *Microsoft COCO: Common Objects in Context*. Cham, 2014, pp. 740–755. 6, 7

[LMSR08] LAPTEV I., MARSZALEK M., SCHMID C., ROZENFELD B.: Learning realistic human actions from movies. 2

[LSD15] LONG J., SHELHAMER E., DARRELL T.: Fully convolutional networks for semantic segmentation. *CVPR* (2015). 2, 7

[LVH16] LEA C., VIDAL R., HAGER G. D.: Learning convolutional action primitives for fine-grained action recognition. In *ICRA* (2016), IEEE, pp. 1642–1649. 7

[LVRH16] LEA C., VIDAL R., REITER A., HAGER G. D.: Temporal Convolutional Networks: A Unified Approach to Action Segmentation. *ArXiv e-prints* (2016). arXiv:1608.08242. 2, 7

[LWH*12] LEVINE S., WANG J. M., HARAUX A., POPOVIĆ Z., KOLTUN V.: Continuous character control with low-dimensional embeddings. *ACM Trans. Graph. 31*, 4 (2012), 28:1–28:10. 3

[LZW*15] LIU Z., ZHANG Y., WU W., LIU K., SUN Z.: Model-driven indoor scenes modeling from a single image. In *GI* (2015), pp. 25–32. 3

[LZZ18] LIANG W., ZHU Y., ZHU S.-C.: Tracking occluded objects and recovering incomplete trajectories by reasoning about containment relations and human actions. In *AAAI Conference on Artificial Intelligence (AAAI)* (2018). 2

[LZZZ16] LIANG W., ZHAO Y., ZHU Y., ZHU S.-C.: What is where: Inferring containment relations from videos. In *IJCAI* (2016), pp. 3418–3424. 2

[ME02] MOORE D., ESSA I.: Recognizing multitasked activities from video using stochastic context-free grammar. In *AAAI/IAAI* (2002), pp. 770–776. 3

[MLZ*16] MA R., LI H., ZOU C., LIAO Z., TONG X., ZHANG H.: Action-driven 3d indoor scene evolution. *ACM Trans. Graph. 35*, 6 (2016), 173:1–173:13. 3

[MSFF17] MOTTAGHI R., SCHENCK C., FOX D., FARHADI A.: See the glass half full: Reasoning about liquid containers, their volume and content. *arXiv preprint arXiv:1701.02718* (2017). 3

[MSL*11] MERRELL P., SCHKUFZA E., LI Z., AGRAWALA M., KOLTUN V.: Interactive furniture layout using interior design guidelines. *ACM Trans. Graph. 30*, 4 (2011), 87:1–87:10. 3

[MSSH14] MAJEROWICZ L., SHAMIR A., SHEFFER A., HOOS H. H.: Filling your shelves: Synthesizing diverse style-preserving artifact arrangements. *TVCG 20*, 11 (2014), 1507–1518. 3

[PJZ11] PEI M., JIA Y., ZHU S.-C.: Parsing video events with goal inference and intent prediction. pp. 487–494. 2

[PKH*17] PIRK S., KRS V., HU K., RAJASEKARAN S. D., KANG H., YOSHIYASU Y., BENES B., GUIBAS L. J.: Understanding and exploiting object interaction landscapes. *ACM Trans. Graph. 36*, 3 (2017), 31:1–31:14. 3

[PRB*18] PUIG X., RA K., BOBEN M., LI J., WANG T., FIDLER S., TORRALBA A.: Virtualhome: Simulating household activities via programs. In *CVPR* (2018). 3

[QHWZ17] QI S., HUANG S., WEI P., ZHU S.-C.: Predicting human activities using stochastic grammar. In *International Conference on Computer Vision (ICCV), IEEE* (2017). 3

[RA09] RYOO M. S., AGGARWAL J. K.: Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities. 2

[RHGS15] REN S., HE K., GIRSHICK R. B., SUN J.: Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR* (2015). 2, 6

[SCH*14] SAVVA M., CHANG A. X., HANRAHAN P., FISHER M., NIESSNER M.: Scenegrok: Inferring action maps in 3d environments. *ACM Trans. Graph. 33*, 6 (2014), 212:1–212:10. 3

[SCH*16] SAVVA M., CHANG A. X., HANRAHAN P., FISHER M., NIESSNER M.: Pigraphs: Learning interaction snapshots from observations. *ACM Trans. Graph. 35*, 4 (2016), 139:1–139:12. 1, 3

[SHL*14] SHARF A., HUANG H., LIANG C., ZHANG J., CHEN B., GONG M.: Mobility-trees for indoor scenes manipulation. *CGF 33*, 1 (2014), 2–14. 3

[SMH11] SUTSKEVER I., MARTENS J., HINTON G. E.: Generating text with recurrent neural networks. In *ICML* (2011), pp. 1017–1024. 4

[ŞMK12] ŞUCAN I. A., MOLL M., KAVRAKI L. E.: The Open Motion Planning Library. *IEEE Robotics & Automation Magazine 19*, 4 (2012), 72–82. 9

[SOT13] SRIDHAR S., OULASVIRTA A., THEOBALT C.: Interactive markerless articulated hand motion tracking using rgb and depth data. In *ICCV* (2013), pp. 2456–2463. 10

[SPYZ11] SI Z., PEI M., YAO B., ZHU S.-C.: Unsupervised learning of event and-or grammar and semantics from video. pp. 41–48. 2

[SS15] SENER O., SAXENA A.: rcrf: Recursive belief estimation over crfs in rgb-d activity videos. Citeseer. 2, 9

[SXZ*12] SHAO T., XU W., ZHOU K., WANG J., LI D., GUO B.: An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Trans. Graph. 31*, 6 (2012), 136:1–136:11. 3

[SZ14] SIMONYAN K., ZISSERMAN A.: Very deep convolutional networks for large-scale image recognition. *CoRR* (2014). 7

[WLO*14] WON J., LEE K., O'SULLIVAN C., HODGINS J. K., LEE J.: Generating and ranking diverse multi-character interactions. *ACM Transactions on Graphics (TOG) 33*, 6 (2014), 219. 3

[WZZZ17] WEI P., ZHAO Y., ZHENG N., ZHU S.-C.: Modeling 4d human-object interactions for joint event segmentation, recognition, and object localization. *IEEE Trans. Pattern Anal. Mach. Intell. 39*, 6 (2017), 1165–1179. 3

[XCF*13] XU K., CHEN K., FU H., SUN W.-L., HU S.-M.: Sketch2scene: Sketch-based co-retrieval and co-placement of 3d models. *ACM Trans. Graph. 32*, 4 (2013), 123:1–123:15. 3

[YDY15] YU L.-F., DUNCAN N., YEUNG S.-K.: Fill and transfer: A simple physics-based approach for containability reasoning. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 711–719. 3

[YKH04] YAMANE K., KUFFNER J. J., HODGINS J. K.: Synthesizing animations of human manipulation tasks. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 532–539. 3

[YLFA15] YANG Y., LI Y., FERMÜLLER C., ALOIMONOS Y.: Robot learning manipulation action plans by" watching" unconstrained videos from the world wide web. In *AAAI* (2015), pp. 3686–3693. 3

[YYT*11] YU L.-F., YEUNG S.-K., TANG C.-K., TERZOPOULOS D., CHAN T. F., OSHER S. J.: Make it home: Automatic optimization of furniture arrangement. *ACM Trans. Graph. 30*, 4 (2011), 86:1–86:12. 3

[ZZZ15] ZHU Y., ZHAO Y., ZHU S.-C.: Understanding tools: Task-oriented object modeling, learning and recognition. pp. 2855–2864. 2