Title
VRKitchen: A 3D Dynamic Interactive Environment for General Computer Vision Research

Author
Gong, Ran

Publication Date
2019

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

VRKitchen:
A 3D Dynamic Interactive Environment
for General Computer Vision Research

A thesis submitted in partial satisfaction
of the requirements for the degree Master of Science
in Computer Science

by

Ran Gong

2019

ABSTRACT OF THE THESIS

VRKitchen:
A 3D Dynamic Interactive Environment
for General Computer Vision Research

by

Ran Gong
Master of Science in Computer Science
University of California, Los Angeles, 2019
Professor Song-chun Zhu, Chair

One of the main challenges of advancing task-oriented learning such as visual task planning and reinforcement learning is the lack of realistic and standardized environments for training and testing AI agents. Previously, researchers often relied on ad-hoc lab environments. There have been recent advances in virtual systems built with 3D physics engines and photo-realistic rendering for indoor and outdoor environments, but the embodied agents in those systems can only conduct simple interactions with the world (e.g., walking around, moving objects, etc.). Most of the existing systems also do not allow human participation in their simulated environments. In this work, we design and implement a virtual reality (VR) system, VRKitchen, with integrated functions which i) allow human teachers to perform demonstrations to train agents (i.e., learning from demonstration). ii) allow users to collect multi-modal sensor data to perform other computer vision tasks. We provide standardized evaluation benchmarks and data collection tools to facilitate a broad use in research on general computer vision research. Especially, we evaluate our collected data on human attention prediction task.

The thesis of Ran Gong is approved.

Demetri Terzopoulos

Kai-wei Chang

Song-chun Zhu, Committee Chair

University of California, Los Angeles

2019

To my family and friends who made this possible

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I'd like to express my sincere gratitude towards my advisor Dr. Song-Chun Zhu for his guidance and mentorship. I am especially grateful for providing me an opportunity to work with a group of extremely talented students. I am thankful for Xue Xie and Qing Li for a lot of helpful discussions. I am also very thankful for Xiaofeng Gao for his detailed guidance and long time support.

CHAPTER 1

Introduction

Thanks to the recent success in many domains of AI research, humans now have built machines that can accurately detect and recognize objects [KH12, HGD17], generate vivid natural images [BDS18], and beat human Go champions [SSS17]. However, a truly intelligent machine agent should be able to solve a large set of complex tasks in the physical world by adapting itself to unseen surroundings and planning a long sequence of actions to reach the desired goals, which is still beyond the the capacity of current machine models. To achieve the state of the art results, these systems often need a simulation environment which agents can interact with. However, in some domains, collecting dataset for agents is very expensive, slow and often inaccurate. This gives rise to the need of an environment capable of synthesizing interactive dataset for different tasks. In particular, we are interested in the following two dataset generation approaches for the present work.

1.1 Generating multi-modal dataset in a dynamic environment.

According to psychology studies [SG05], humans learn from multi-modal inputs (Vision, Sound, Touches etc.). Different input modules self-teach each other so that infants can obtain a rich and compact experience about the world. Recent works [NKK11] have been using sensor fusion-like algorithm to merge different modality of inputs. Therefore, an environment that can obtain different modality of sensor inputs with interactivity should be designed and implemented.

Researchers [SG05] also indicate that learning experience for infants are physical, and infants often explore around the environment to find out what task to be learned and the solutions to these tasks. To better simulate the real world scenario where the appearance of the same object may change dramatically as a result of actions [ILA15, FR13, LWZ17], the

dataset generation environment needs to have rich fluent changes. To capture such variation in object appearance, the agent is required to have a better visual representation of the environment dynamics. For example, the agent should recognize the tomato even if it is cut into pieces and put into container. To acquire such visual knowledge, it is important for an agent to learn from physical interactions and reason over the underlying causality of object state changes. Therefore, it is critical to have an interaction-based dynamic world.

Long sequences of events are often needed for certain tasks: for example, human attention prediction and human intention prediction. In order to perform these tasks, current systems often need a large amount of annotated data. Therefore, a system that is capable of generating a large number of highly customizable annotated data will potentially be helpful to the future research.

There have been work on implementing interaction-based learning in lab environments [LGF16, ACM15, HKB15], but the limited scenarios greatly restrict scalability and reproducitibility of prior work, plus the ad-hoc environments often do not come with dataset generation capability.We believe that building a realistic simulation platform is a good alternative since i) performance of different algorithms can be easily evaluated and benchmarked, ii) a large set of diverse and realistic environments and tasks can be designed and customized. iii) customizable multi-modal data can be relatively easily generated (Figure 1.1, Figure 1.2, Figure 1.3).


Figure 1.1: RGB

1.2 Collecting human demonstrations to bootstrap agents' models.

Training an agent from scratch is extremely difficult in complex environments. To bootstrap the training, it is common to let an agent to imitate human experts by watching human

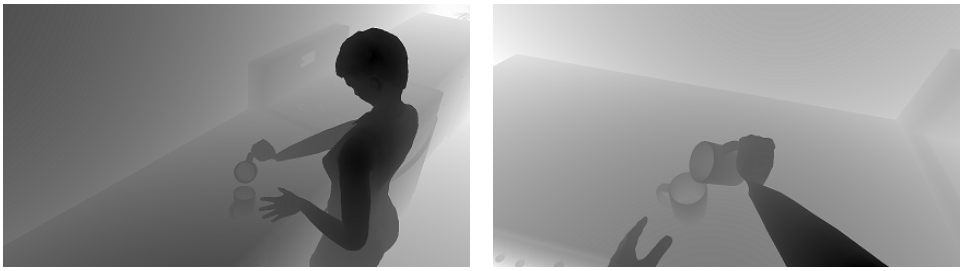

Figure 1.2: Depth

Figure 1.3: Segmentation

demonstrations [NR00, ZMB08, GGC16]. Previous work has shown that learning from demonstrations (or imitation learning) significantly improves the learning efficiency and achieves a higher performance than reinforcement learning does [ZGK17, HVP17]. However, it is expensive and time consuming to collect diverse human demonstrations with high qualities. We believe that virtual reality games can provide us with an ideal medium to crowd source demonstrations from a broad range of users [AD08].

In this work, we focus on simulating cooking activities in a virtual kitchen environment, VRKitchen. We illustrate how this system can address the emerging needs for the learning problems in an example shown in Figure 1.4, where an agent makes a sandwich in one of the kitchens created in our system.

- The environment allows the agent to interact with different *tools* and *ingredients* and simulates a variety of object changes. E.g., the bread changes its color when it is being heated in the oven, and the tomato turns into slices after it is cut. The agent's interactions with the physical world when performing cooking tasks will result in large variations and temporal changes in objects' appearance and physical properties, which calls for a task-oriented visual representation.
- To make a sandwich, the agent needs to perform a long sequence of actions, including

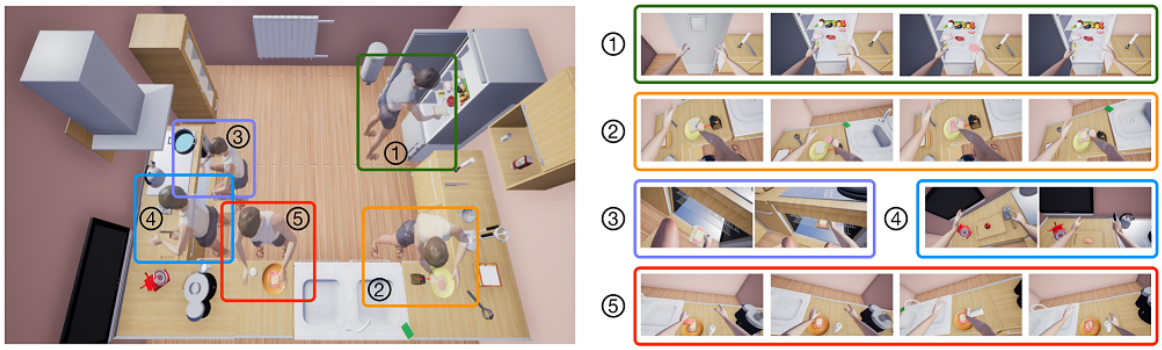

Figure 1.4: A sample sequence of an agent making a *sandwich*. Rectangles on the left graph represents five necessary sub-tasks, including (1) taking ingredients from fridge, (2) putting ham and cheese on the bread, (3) use the oven, (4) cut tomato and (5) add some sauce. Each rectangle on the right graph indicates atomic actions required to finish a sub-task.

taking ingredients from a fridge, putting cheese and ham on the bread, toasting the bread, adding some sliced tomato and putting some sauce on the bread. To quickly and successfully reach the final goal, it is necessary to equip the agent with the ability to conduct long-term planning.

- We build two interfaces to allow an AI algorithm as well as a human user to control the embodied agent respectively, thus humans can give demonstrations using VR devices at any places in the world, and the AI algorithms can learn from these demonstrations and perform the same tasks in the same virtual environments.

In summary, our main contributions are:

- A configurable virtual kitchen environment in a photo-realistic 3D physical simulation which enables a wide range of cooking tasks with rich object state changes and compositional goals;
- A toolkit including a VR-based user interface for collecting human demonstrations, and a Python API for training and testing different AI algorithms in the virtual environments.
- A new human demonstration dataset of various cooking tasks – UCLA VR chef dataset.
- A multi-view dataset automatically generated from VRKitchen with automatically generated annotations.

CHAPTER 2

Related Work

Env.	Large-scale	Physics	Realistic	State	Manipulation	Avatar	Demon
Malmo [JHH16]	√			√			
DeepMind Lab [BLT16]							
VizDoom [KWR17]							
MINOS [SCD17]	√		√				
HoME [BPA17]	√	√	√				
Gibson [XZH18]	√	√	√			√	
House3D [WWG18]	√	√	√				
AI2-THOR [KMG17]		√	√	√			
VirtualHome [PRB18]			√	√	√	√	
SURREAL [FZZ18]		√			√		√
VRKitchen (ours)		√	√	√	√	√	√

Table 2.1: Comparison with other 3D virtual environments. Large-scale: a large number of scenes. Physics: physics simulation. Realistic: photo-realistic rendering. State: changeable object states. Manipulation: enabling object interactions and manipulations. Avatar: humanoid virtual agents. Demonstration: user interface to collect human demonstrations.

2.1 Simulation platforms

Traditionally, visual representations are learned from static datasets. Either containing pre-recorded videos [RAA12] or images [JWS09a], most of them fail to capture the dynamics in viewpoint and object state during human activities, in spite of their large scale. Some early systems [QT08, RT00, TR95, LGS16] try to simulate the dynamics of human activities in order to support the development of smart visual surveillance systems and research into ac-

tive computer vision for navigation. However, agents in the environment cannot be trained in a fined grained level for compositional tasks, and environments often do not have a lot of dynamic changes caused by agents' actions.

To address this issue, there has been a growing trend to develop 3D virtual platforms for training embodied agents in dynamic environments. Typical systems include 3D game environments [KWR17, BLT16, JHH16], and robot control platforms [TET12, CB16, FZZ18, PAR18]. While these systems offer physics simulation and 3D rendering, they fail to provide realistic environments and daily tasks humans face in the real world.

More recently, based on 3D scene datasets such as Matterport3D [CDF18] and SUNCG [SYZ17], there are have been several systems simulating more realistic indoor environments [BPA17, WWG18, SCD17, MHL17, XZH18] for visual navigation tasks and basic object interactions such as pushing and moving funitures [KMG17]. While the environments in these systems are indeed more realistic and scalable compared to previous systems, they still can not simulate complex object manipulation that are common in our daily life. [PRB18] took a step forward and has created a dataset of common household activities with a larger set of agent actions including pick-up, switch on/off, sit and stand-up. However, this system was only designed for generating data for video understanding. In contrast, our system emphasizes training and evaluating agents on virtual cooking tasks, which involves fine-grained object manipulation on the level of object parts (e.g., grasping the handle of a knife), and flexible interfaces for allowing both human users and AI algorithms to perform tasks. Our system also simulates the animation of object state changes (such as the process of cutting a fruit) and the gestures of humanoid avatars (such as reaching for an object) instead of only showing pre-conditions and post-effects as in [KMG17]. A detailed comparison between our system and other virtual environments is summarized in Table 2.1.

2.2 Imitation learning

Learning from demonstration or imitation learning is proven to be an effective approach to train machine agents efficiently [AN04, SS08, RGB10]. Collecting diverse expert demonstrations with 3D ground-truth information in real world is extremely difficult. We believe the VR interface in our system can greatly simplify and scale up the demonstration collec-

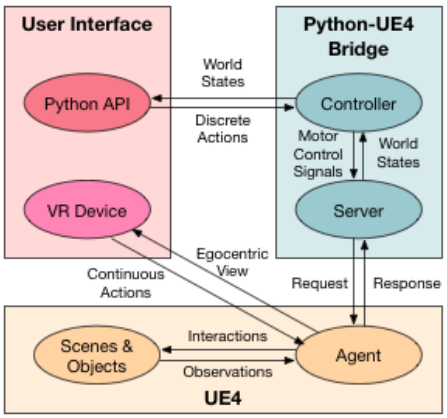

Figure 2.1: Architecture of VRKitchen. Users can either directly teleoperate the agent using VR device or send commands to the agent by Python API.
tion.

2.3 VR for AI

VR provides a convenient way to evaluate AI algorithms in tasks where interaction or human involvement is necessary. Researches have been conducted on many relevant domains, including physical intuition learning [LGF16], human-robot interaction [LRM17, GRO17], learning motor control from human demonstrations [HKB15, KNM01, BCC01]. Researchers have also used VR to collect data and train computer vision models. To this end, several plugins for game engines have been released, such as UETorch [LGF16] and UnrealCV [QY16]. To date, such plugins only offer APIs to control game state and record data, requiring additional packages to train virtual agents, or to gather data for other computer vision tasks.

2.4 Datasets for computer vision tasks

Deng et al.[JWS09b] started the big data era for modern computer vision research. With the increasing popularity of deep learning, there are a lot of datasets catering for different tasks. For human attention predictions, CAD120 [KGS13], has been commonly used. However, creating these dataset will require a significant amount of man power to design tasks, record videos, and annotate data. Amazon Mechanical Turks are commonly used for annotation purposes; however, according to our own experiences, turker annotations are often noisy, so

it takes a large amount of time for researchers to design a protocol to make sure annotations gathered from turkers are reliable. For tasks like object recognition, researchers often want to label the smallest bounding box around the object; however, in real annotation scenarios, it is really hard for humans to find the smallest bounding box around the object. This might due to the fact that turkers are paid by the number of images they annotated not by how accurate their boxes are. As long as, their annotations are reasonable, researchers will often give them a pass. However, recent studies [RBE17] has demonstrated that inaccuracy in the data annotations can often produce meaningful differences in the final model. Therefore, it is crucial to have accurate annotations for the dataset. In an virtual environment, since we have all the information about the object model, it is relatively easy to obtain ground truth information through transformation matrices and projection matrix.

CHAPTER 3

VRKitchen Environment

Our goal is to enable better learning of autonomous agents for tasks with compositional goals and rich object state changes. To this end, we have designed VRKitchen, an interactive virtual kitchen environment which provides a testbed for training and evaluating various learning and planning algorithms in a variety of cooking tasks. With the help of virtual reality device, human users serve as teachers for the agents by providing demonstrations in the virtual environment.

3.1 Architecture Overview

Figure 2.1 gives an overview of the architecture of VRKitchen. In particular, our system consists of three modules: (1) the physics engine and photo-realistic rendering module consists of several humanoid agents and kitchen scenes, each has a number of ingredients and tools necessary for performing cooking activities; (2) a user interface module which allows users or algorithms to perform tasks by virtual reality device or Python API; (3) a Python-UE4 bridge, which transfers high level commands to motor control signals and sends them to the agent.

3.2 Physics Engine and Photo-realistic Rendering

As a popular game engine, Unreal Engine 4 (UE4) provides physics simulation and photo-realistic rendering which are vital for creating a realistic environment. On top of that, we design humanoid agents, scenes, object state changes, and fine-grained actions as follows.

Figure 3.1: Four humanoid avatars designed using MakeHuman .

Figure 3.2: VRKitchen scenes

3.2.1 Humanoid Agents

Agents in VRKitchen have human-like appearances (shown in Figure 3.1) and detailed embodiment representations. The animation of the agent can be broken into different states, e.g. *walking, idle*. Each agent is surrounded by a capsule for collision detection: when it's *walking*, it would fail to navigate to a new location if it collides with any objects in the scene. When it is *idle*, the agent can freely interact with objects within certain range of its body.

There are in total 12 different animation states as shown in Figure 3.3. Each animation state has an associated animation. The transitions of the animation states are determined by python api data. When appropriate, the python api will issue an animation transition command. There are two types of animations: i) node manipulation through inverse kinematics(IK) ii) blended animations from different online resources. IK systems are more flexible; however the final animation might not look natural at all. This is because the IK system will try to reach the location of specification regardless of the pose of the character. Blended animations, on the other hand, are more natural, but can not reach any arbitrary position in the environment. Therefore, for animation states that require object manipulations or related to object manipulations: reaching for an object, crouching down, we are using IK system. For animation states like holding object and walking, turning around, standing

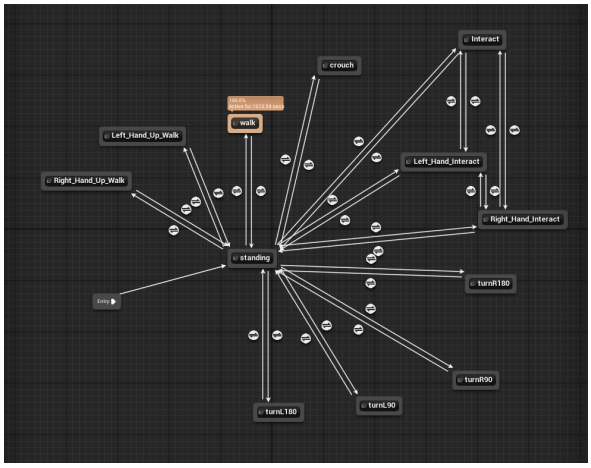

Figure 3.3: animation states for our agents

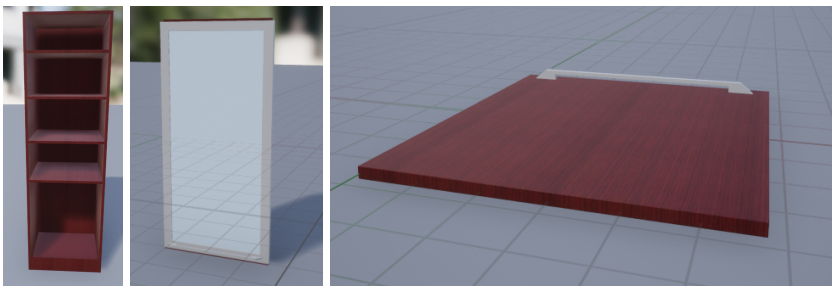

Figure 3.4: Sample decomposed kitchen cabinet. Manually decomposed through blender.

up, we are using blended animations. The transition between animation states are blending using the tool provided by UE4. We chose the Hermite cubic transition mode with a transition duration between 0.2 seconds to 0.5 seconds based on different transitions. We manually tried different modes and transition duration, and found out this parameter setting looks more natural. Even though all agents share the same animation states, different agents may exhibit different behaviors when using IK animations. This is because different agents have different limb length, so the computed IK trajectory might be different.

3.2.2 Scenes

VRKitchen consists of 16 fully interactive kitchen scenes as shown in Figure 3.2. Agents can interact with most of the objects in the scenes, including various kinds of *tools, receptacles* and *ingredients*. Each kitchen is designed and created manually based on common household setting. 3D models of furniture and appliances in kitchens are first obtained from the SUNCG dataset [SYZ17]. SUNCG dataset provides a script to create an entire kitchen

Figure 3.5: Small window shows a tomato before a cutting action

Figure 3.6: Small window shows a tomato after a cutting action

from different 3D models. However, the created kitchens do not support any interactions at all. For example, agents cannot open the doors in the kitchen (stove door, cabinet doors etc.), because doors are fixed, not movable. To solve this issue, we use blender to manually separate door from the rest of the 3D model for various different 3D models as shown in Figure 3.4. We also decompose other parts of the objects according to our need. Sometimes decomposing objects are not enough. A decent amount of SUNCG models do not have interiors at all. In order to make it functional in our kitchen setting, we also manually design and create the functional interiors for our 3D models using blender. After we have basic furniture and appliances in the scene, we then add cooking *ingredients* and *tools*. Instead of sampling their locations randomly, we place the objects according to their utility, e.g. *tools* are placed on the cabinets while perishable *ingredients* such as fruits and vegetables are available in the fridge. On average, there are 55 interactive objects in a scene.

3.2.3 Object State Changes

One key factor of VRKitchen is the ability to simulate state changes for objects. Instead of showing only pre-conditions and post effects of actions, VRKitchen simulates the continuous geometric and topological changes of objects caused by actions. This leads to a great number of available cooking activities, such as roasting, peeling, scooping, pouring, blending, juicing, etc. Overall, there are 18 cooking activities available in VRKitchen.

The environment mainly consists discrete changes as well as shown in Figure 3.5 and Figure 3.6. We believe for most tasks, discrete fluent changes that specifying pre-conditions and post-effects are sufficient for task planning.

3.2.4 Fine-grained Actions

In previous platforms [KMG17, BPA17], objects are typically treated as a whole. However, in real world, humans apply different actions to different parts of objects. E.g. to get some coffee from a coffee machine, a human may first press the power button to open the machine, and press the brew button afterwards to brew coffee. Thus we design the objects in our system in a compositional way, i.e., an object has multiple components, each of which has its own affordance. This extends the typical action space in prior systems to a much larger set of fine-grained actions and enables the agents to learn object-related causality and commonsense.

3.3 User Interface

With a detailed human embodiment representation, multiple levels of human-object-interactions are available. In particular, there are two ways for users to provide such demonstrations:

(1) Users can directly control the agent's head and hands. During teleportation, actions are recorded using a set of off-the-shelf VR device, in our case, an Oculus Rift head-mounted display (HMD) and a pair of Oculus Touch controllers. Two Oculus constellation sensors are used to track the transforms of the headset and controllers in 3D spaces. We then apply the data to a human avatar in the virtual environment: the avatar's head and hand movements correspond to the human user's, while other parts of its body are animated through a

built-in Inverse Kinematics solver (Forward And Backward Reaching Inverse Kinematics, or FABRIK). Human users are free to navigate the space using the Thumbsticks and grab objects using the Trigger button on the controller. Figure **??** gives an example of collecting demonstrations for continuous actions.

(2) The Python API offers a way to obtain discrete action sequences from users. In particular, it provides world states and receives discrete action sequences. The world state is comprised of the locations and current states of nearby objects and a RGB/depth image of agent's first person view. Figure 3.7 and Figure 3.8 show examples of recorded human demonstrations for tasks *pizza* and *roast meat* from a third person view.

3.4 Python-UE4 Bridge

The Python-UE4 bridge contains a communication module and a controller. The Python server communicates with the game engine to receive data from the environment and send requests to the agent. It is connected to the engine through sockets. To perform an action, the server sends a command to UE4 and waits for response. A client in the game engine parses the command and applies the corresponding animations to the agent. A payload containing states of nearby objects, agent's first person camera view (in terms of RGB, depth and object instance segmentations) and other task-relevant information are sent back to the Python server. The process repeats until terminal state is reached.

The controller enables both low level motor controls and high level commands. Low level controls change local translation and rotation of agent's body, heads and hands, while other body parts are animated using FABRIK. High level commands, which performs atomic actions such as taking or placing an object, are further implemented by taking advantage of the low level controller. To cut a carrot with a knife, for example, the high level controller iteratively updates the hand location until the knife reaches the carrot.

3.5 Performance

We run VRKitchen on a computer with Intel(R) Core(TM) i7-7700K processor @ 4.50GHz and NVIDIA Titan X (Pascal) graphics card. A typical interaction, including sending com-

Figure 3.7: An example of human demonstrations for making a *pizza*.

mand, executing the action, rendering frame and getting response, takes about 0.066 seconds (15 actions per second) for a single thread. The resolutions for RGB, depth and object segmentation images are by default 84×84, but can be changed to any resolution if needed(will affect performance).

3.6 Environment Interactions

In VRKitchen, we design all atomic actions and object state changes available in several dish preparing tasks. Using these atomic actions, the agent can interact with the environments until a predefined goal is reached. Figure 3.9 shows some examples of dishes.

3.6.1 Atomic Actions

Each atomic action listed below can be viewed as a composition of a verb (action) and a noun (object). Objects can be grouped into three types: *tools, ingredients and receptacles*. (1) *Ingredients* are small objects needed to make a certain dish. We assume that the agent can hold at most one *ingredient* at a time. (2) For *receptacles*, we follow the definition in [KMG17]. They are defined as stationary objects which can hold things. Certain *receptacles* are called *containers* which can be closed and agents can not interact with the objects within them until they are open. (3) *Tools* can be used to change the states of certain *ingredients*. Atomic actions and object affordance are defined in a following way:

- `Take` {*ingredient*}: take an *ingredient* from a nearby *receptacle*;
- `Place into` {*receptacle*}: put a held *ingredient* into a nearby *receptacle*;
- `Use` {*tool*}: use a *tool* to change the state of a *ingredient* in a nearby *receptacle*;
- `Go To` {*tool, receptacle*}: move to a *tool* or *receptacle*;
- `Toggle (open/close)` {*container*}: change state of a *container* in front of the agent.
- `Turn`: rotating the agent's facing direction by 90 degrees.

Note that actions including `Take, Place into, Use`, and `Toggle` would fail if the agent is not near the target object.

3.6.2 *Ingredient* Sets and States

Meanwhile, there are seven sets of *ingredients*, including *fruit, meat, vegetable, cold-cut, cheese, sauce, bread* and *dough*. Each set contains a number of *ingredients* as variants: for example, *cold-cut* can be ham, turkey or salami. One *ingredient* may have up to four types of state changes: *cut, peeled, cooked* and *juiced*. We manually define affordance for each set of *ingredients*: e.g. *fruit* and *vegetable* like oranges and tomatoes can be juiced (using a juicer) while bread and meat can not. *Tools* include grater, juicer, knife, oven, sauce-bottle, stove and *receptacles* are fridge, plate, cut-board, pot and cup.

Task	Goal states	Target location
Fruit juice	fruit1: cut, juiced; fruit2: cut, juiced	cup
Roast meat	fruit: cut, juiced, cooked; meat: cooked	pot
Stew	veg: cut, cooked; meat: cooked	pot
Pizza	veg: cut, cooked; cold-cut: cooked; cheese: cooked; sauce: cooked; dough: cooked	plate
Sandwich	veg: cut; sauce; cold-cut: cooked; cheese: cooked; bread: cooked	plate

Table 3.1: The goals for five available dishes. In each task, the agent should change required *ingredients* to the goal states and move them to a target location.

3.6.3 Goals

Based on the atomic actions defined in 3.6.1, agents can prepare five dishes: *fruit juice, stew, roast meat, sandwich* and *pizza*. Goals of each tasks are compositionally defined upon (1) goals states of several sets of ingredients and (2) target locations: to fulfill a task, all required *ingredients* should meet the goal states and be placed in a target location. For example, to fulfill the task *fruit juice*, two *fruits* should be cut, juiced and `place into` the same cup. Here, the target locations are one or several kinds of *containers*. Table 3.1 defines the goal states and target locations of all tasks.

Figure 3.8: An example of human demonstrations for making *roast meat*.

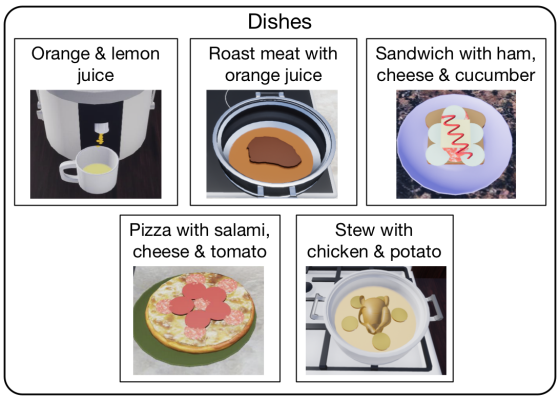

Figure 3.9: Examples of dishes made in VRKitchen. Note that different *ingredients* leads to different variants of a dish. For example, mixing orange and kiwi juice together would make *orange & kiwi juice*.

CHAPTER 4

Data Generation from Virtual Environment

4.1 Gather Data from Human Demonstrations

Gathering Human demonstrations is an essential component for policy-learning tasks. DAGGER [RGB10] requires humans continuously provide feedback to the learned policy, and take over when human sees a mismatch between agents' policy and human's belief. A distributed data collection tool also provide future opportunity for crowd sourcing human demonstrations. Here, we provide two different ways for human to take over when appropriate: i) through VR device 2) through a web-based interface in case not having a VR device available.

4.1.1 Gather Human Demonstration from VR Device

UE4 has built-in VR support. Here we use Oculus Rift to perform our experiment. In the VR Setting, a lot of actions are continuous; however, in VR Kitchen, atomic actions are discrete. In order to mitigate this difference, we propose to decompose continuous actions into discrete ones. Users use Touch Pad "A" Button and pointers to navigate around the world to make appropriate actions. As long as users have VR device, they are not constraint on their locations as shown by Figure 4.1 and Figure 4.2 .

i) When the user pressed "A" button, provided the user is far away from the location of interest, the system will teleport the user to the nearest valid location around his pointer. In the back end, the system will interpret this user action as GoTo *location*.

ii) When the user pressed "A" button, provided the user is close to the location of interest, the system will interpret the action as one of the following according to the current state of the agent: Use *item*, Take *item*, Place into *item*, Toggle(Open/Close) *item*.

Figure 4.1: Using a VR device at home

Figure 4.2: Using a VR device in office

4.1.2 Gather Human Demonstration from Web Interface

VR device is still quite expensive until this date. Therefore, not everyone has VR device available in their home. In order to alleviate this problem, we propose to use a web-based interface for the purpose of data collection for the general public. The web-based interface is built using Flask, JavaScript, HTML5 and CSS.

At first, the web-based interface will provide some initial text-based instructions about the task as shown in figure 4.3, and the users are asked to solve this problem with their commonsense knowledge.

After users choose a task and a scene id, the web-based interface will provide a short and quick demo about the environment set-up, and a sample demonstrations done by the

Figure 4.3: web interface instructions

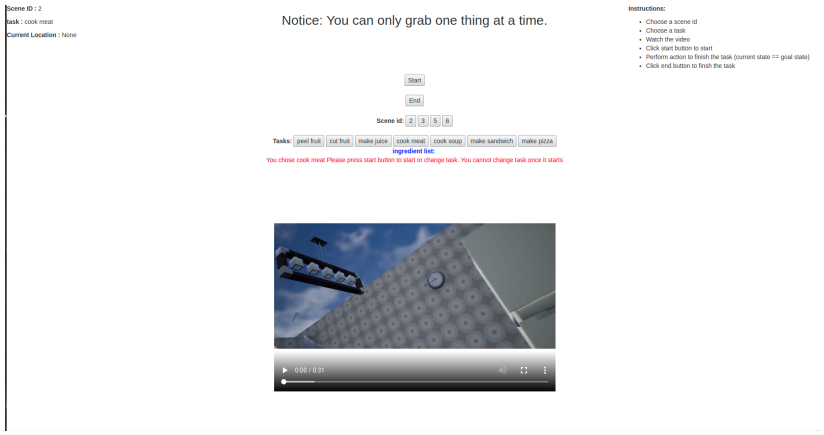

Figure 4.4: web interface tutorial

machine for that task. As shown by figure 4.4. However, the ingredients are randomized. Therefore, users are most likely to use a different set of ingredients.

Then the web-based interface will display all the valid atomic actions to the users and provide users with a text-based description about the goal state of the current task and the user's current state as shown in figure 4.5. Users can simply click buttons on a web browser to execute an action. All actions are recorded in the back-end. The actions and the associated user RGB image can be used for imitation learning.

In the data collection process, users will first solve a simple task "Cut Fruit", which is not recorded so that users are familiar with the tools and the environment.

We record the human demonstrations from 9 different users, and we found out that user's background has a significant impact on task completion steps. For example, users' from a western background can finish the "Make Pizza" task way faster than others. We also find

Figure 4.5: web interface atomic actions

Task	Average(Number of steps to solve the task)	Median	Variance
Cook Meat	16.67	16	7.8
Cook Soup(only 8 data points are valid)	15.38	14.5	9.9
Make Juice	17.67	17	6.9
Make Pizza	34.11	33	28.8
Make Sandwich	30.67	29	12.67

Table 4.1: Human Demonstration Statistics

out that simple tasks tend to have a lower variance. Detail statistics are shown in Table 4.1

4.2 Generate Data from Ground Truth

In General Computer vision tasks, we often need to annotate data. With our environment, we can automatically provide some annotated data with low noise. We can provide RGB image, depth image and instance segmentation from multi-view cameras. We also has the capacity to automatically generate bounding boxes on a 2D image. Apart from those general computer vision annotations, we can also provide annotations tailored to a specific computer vision task.

Here we demonstrate our environment's capacity through a AttentionObject-VR dataset.

4.2.1 Human Attention

Attention is an important topic in the computer vision field and has been widely used for object detection, video tracking , image retrieval, and other applications. Eye fixation saliency map estimation and saliency object estimation are two important problems in the study of visual attention, and their focus is inferring saliency regions or objects in an image that draw the attention of the human (outside the image) who is looking at the image. In this paper, we study the attention of a human inside a third-person view video, we call it Inside-video human attention. To infer human attention, the foremost thing is to make clear what the human attention is. Originally, attention is a concept in philosophy. Nowadays, it is well known as a concept in psychology. One dominant definition in psychology is that attention is the process of attending to objects. This definition indicates that the attention is based on objects. Actually, some studies [Che12, CY12, PR14] in psychophysics and biology fields as well as some inter-discipline studies in neuro image filed and brain image field also claim the object-based attention. These studies provide the strong theory support for defining human attention as objects. Another widely accepted definition in psychology is that attention is something that happens in the mind - a mental ”inside” which is linked with the perceivable ”outside” [See11]. This definition indicates that attention is related with the high-level invisible information in human mind.

Based on these studies, we define human attention as the attentional objects that coincide with the task a human is doing. With a task in the mind, a human finishes the task by doing several sub-tasks in certain temporal order. For example, when a human is doing the task of take the water from the drinking fountain”, the human firstly finds the cup, then goes to the drinking fountain, and finally takes the water. To finish each sub-task, a human behaves purposely to operate on or approach to the attentional objects. For example, when the human is doing the sub-task of finding the cup”, the human uses the hand to catch the cup. When the human is doing the sub-task of going to the drinking fountain”, the human walks to approach to the drinking fountain.

4.2.2 Dataset Overview

Though there exists a large number of datasets for the studies of human gaze, visual attention, and human-object interaction, to our best knowledge, no publicly available dataset is targeted for inferring the task-driven inside video human attention. Therefore, we collect a video dataset in VR (Virtual Reality) scenes. With the development of VR technique, the VR data has become extremely life-like as real data. In VR scenes, all objects are configured with accurate locations and sizes, allowing the automatic object annotations and large-scale data collection. To collect the dataset, we use 8 different existing kitchen scenes. In each scene, many furniture and objects are configured, objects can be divided into two categories: tools (e.g., knife, juicer, oven, etc.) and ingredients (e.g. bread, orange, tomato, etc.). A human can use tool to change the state of an ingredient. For example, to do the task of making orange juice, a human uses a knife to cut an orange into halves and put them into a juicer to get juice.

The dataset has several characteristics:

- Diverse and large. The dataset consists of 8 scenes, 10 tasks, 33 subtasks, and 4 humans. As shown in Figure 4.6, different scenes vary significantly in the scene scale, furniture configuration, and object placement. For each scene, we collect videos from 3 different camera views to make the data more diverse. The camera views are fixed in certain scene, and are manually chosen to make the views cover the 360-degree scene. the images of different camera views notably differ from each other. The 10 tasks are: bake bread, cook soup, cut meat, fry steak, make coffee, make juice, make sandwich, microwave food, pour coke, and turn on light. The dataset is large, consisting of 133,419 images and 1,887,858 object annotations in total. Averagely, each video consists of 171 images. The video resolution resolution is 1280×720

- Well-organized. To make the dataset qualified for inferring human attentional objects, it is necessary to guarantee humans and attentional objects are inside images. Therefore, we remove the images and videos that do not satisfy this requirement. To divide the dataset into training set and testing set, the data collected in scene 7 and scene 8 is used for testing, and the data collected in other scenes is used for training.

- Well-annotated. Figure 4.7 shows an example of annotating a video. Given a video with a task label, it is segmented as several sub-tasks to guarantee that the attentional object in each sub-task is determinate. To accurately segment a task into several sub-tasks, three volunteers are asked to find the key frames in a video to segment sub-tasks. For most cases, the key-frame is not controversial. For controversial ones, the average key-frame is taken as the final key-frame. For each frame, the attentional objects and non-attentional objects are annotated with the detailed information like the location, size, and types. Averagely, each image contains 1.16 attentional object annotations (two attentional objects are annotated in some images) and 13 non-attentional object annotations. Benefiting from the good annotations, the dataset can also be used for other studies like task/event recognition, video segmentation, and action recognition.

-	Vidoes	Images	Attentional Objects	Other Objects
Train	596	100,951	117,643	1,330,431
Test	184	32,468	37,211	402,573
Total	780	133,419	154,854	1,733,004

Table 4.2: The statistics of the AttentionObject-VR dataset. Videos: video number, Images: image number, Attentional objects: attentional object annotation number, other objects: non-attention object annotation number

Methods	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	All
PRNet	0.41	0.28	0.29	0.28	0.26	0.29	0.31	0.34	0.27	0.07	0.30
Hopnet	0.54	0.36	0.36	0.37	0.17	0.37	0.39	0.39	0.29	0.00	0.35
ResNet-BinCls	0.49	0.51	0.46	0.55	0.19	0.53	0.48	0.71	0.50	0.48	0.48

Table 4.3: Accuracy of different methods on the AttentionObject-VR dataset. ”All” corresponds to the overall accuracy. T1 to T10 correspond to the accuracy on different tasks. T1: bake bread, T2: cook soup, T3: cut meat, T4: fry steak, T5: make coffee, T6: make juice, T7: make sandwich, T8: microwave food, T9: pour coke, and T10: turn on light.

4.2.3 Dataset Benchmark Results

We run object detection algorithm RetinaNet [LGG17] on this dataset, and the results are in Table 4.7. Then we benchmark our dataset on human attention task.

We study the problem of inferring the task-driven attentional objects of a human inside third-person view videos, to our best knowledge, there does not exist exactly same work with ours. The most related work is to estimate where a human is looking. Therefore, we select two state-of-the-art human face and head direction estimation methods as baselines. We briefly describe the three baseline methods as follows.

- PRNet [FWS18]. PRNet is a face alignment method that can estimate human face direction. It takes the raw image and human face as input, and the output is the dense (more than 40K) aligned face key points. These dense points are compared with a pretrained model to compute the camera matrix, which is further combined with 68 facial key points to estimate the human face direction.

- Hopenet [RCR18]. Hopenet is a head pose estimation method. It takes the raw image and human face as input, and the output is the three Euler angles that signal human head direction.

- ResNet-BinCls [HZR16]. ResNet-BinCls is a binary classification method based on ResNet-18 [HZR16]. It first detects the objects in an image, then a binary classifier estimates the scores of each object being and not being the attentional object. To estimate the score of a candidate object, the human skeleton and the candidate object are represented as a binary $1 \times H \times W$ mask, which is concatenated with $3 times$H$times$W raw image to serve as the input of the trained binary classifier. The RetinaNet model [LGG17] and OpenPose model [CSW17] are respectively used for attentional object candidate detection and human pose estimation.

Benchmark results are shown in Table 4.3, and qualitative results are shown in Figure 4.8 .

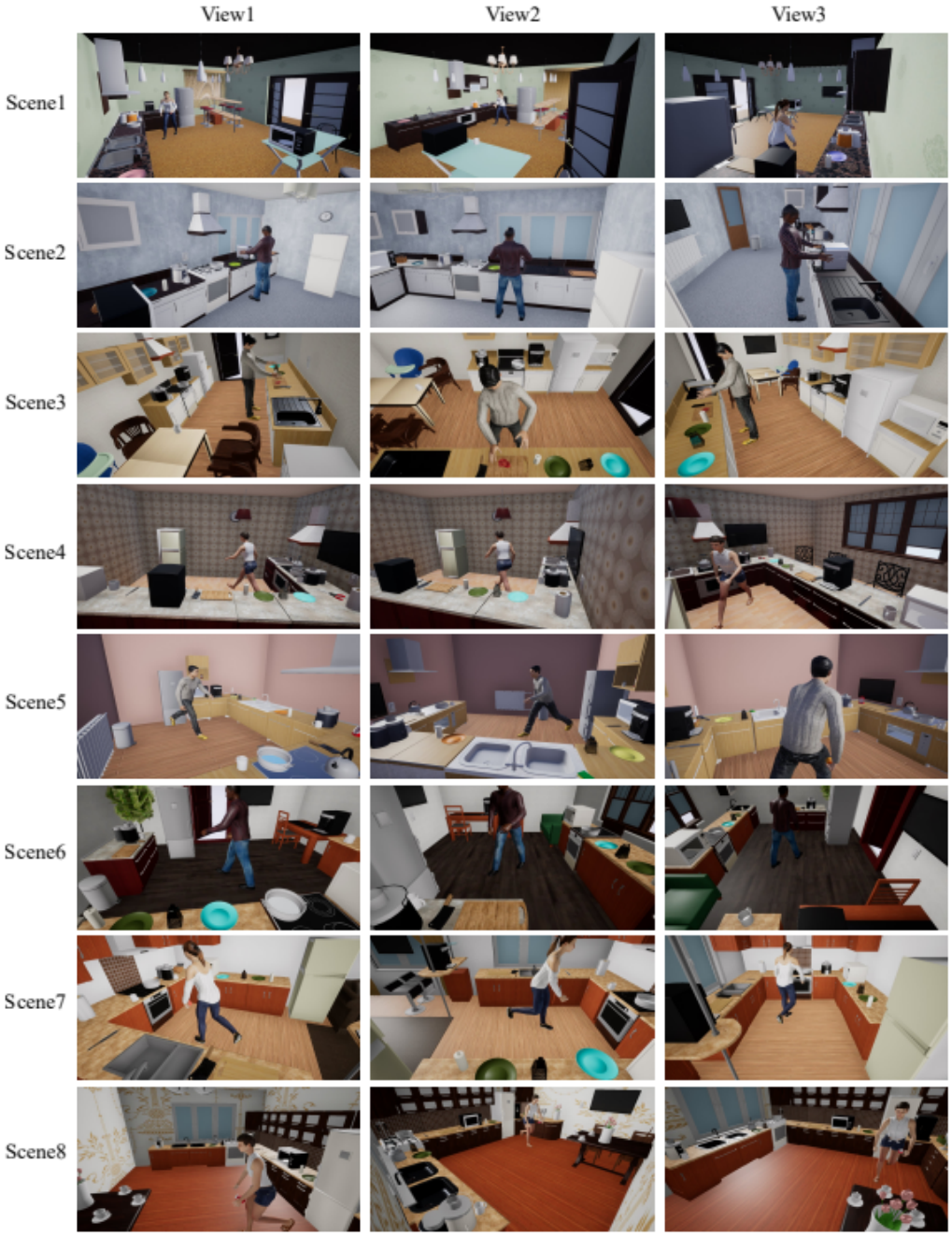

Figure 4.6: Samples of the AttentionObject-VR dataset. The dataset is collected in eight scenes. In each scene, videos are captured from three difierent camera views. In this figure, each row shows three images from the three camera views at the same time in the same scene

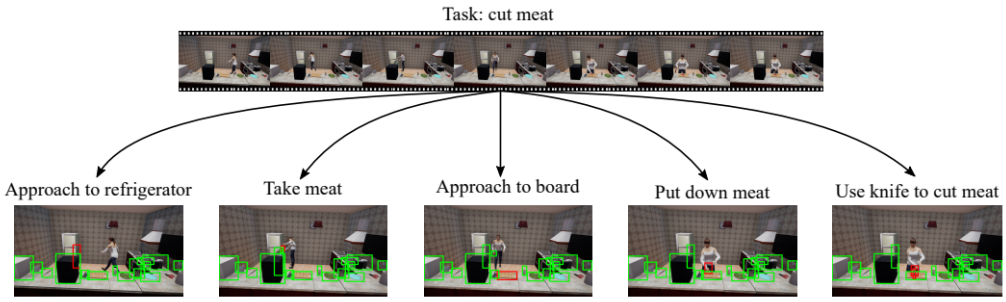

Figure 4.7: An example of annotating a video. Given a video with the task label of meat", the video is segmented as several sub-tasks ("approach to refrigerator", "take meat", "approach to board", "put down meat", and "use knife to cut meat"). In each sub-task, the attentional object (red bounding boxes) and other non-attentional objects (green bounding boxes) are annotated. To conclude, the annotations include task label, sub-task labels, attentional objects, and non-attentional objects.

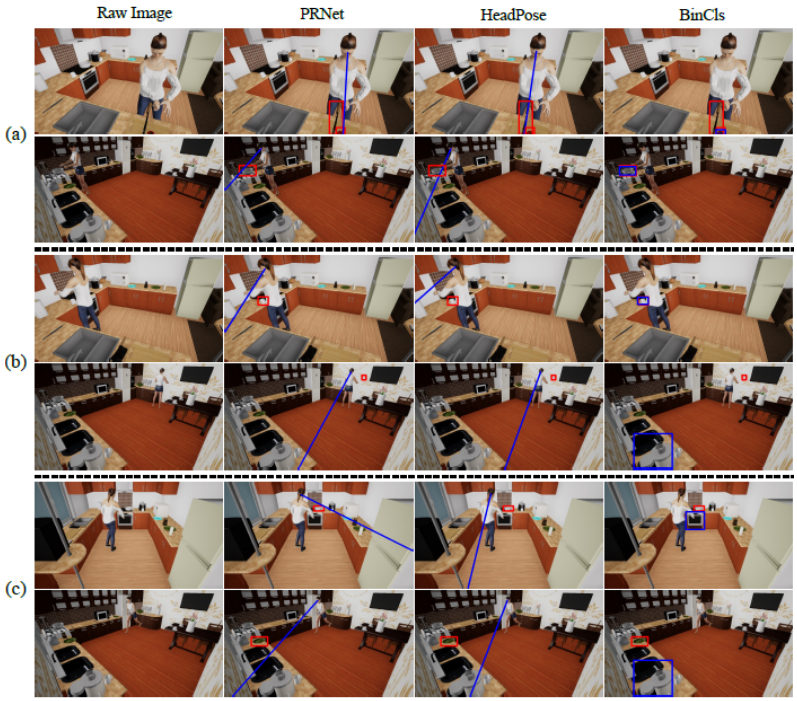

Figure 4.8: Samples of qualitative results of different methods in three typical scenarios. (a) Human facial information is available and conveys the distinct cue to infer attentional objects. (b) Human facial information is not available, but the human pose provides the informative cue to infer attentional objects. (c) Human facial cue and human pose cue are not sufficient, and invisible high-level task information is needed to infer attentional objects. In this Figure, the red bounding boxes represent the ground truth attentional object annotations, the blue lines represent the face and head directions estimated by the PRNet model and Hopenet model, and blue bounding boxes represent the attentional objects estimated by the ResNet-BinCls method.

Task	Task definition(attention objects in parenthesis)
Make Coffee	Approach cup (cup)
	Take cup (cup)
	Approach coffee machine(coffee machine)
	Put the cup under coffee machine (cup and coffee machine)
	Press make button to make coffee(cup and coffee machine)
Microwave Food	Approach fridge (fridge)
	Open door of fridge (fridge)
	Take the bread(bread)
	Close fridge(fridge)
	Approach plate(plate)
	Put bread on plate (bread and plate)
	Take the plate and bread to approach microwave(microwave)
	Use microwave(microwave)
Cook Soup	Approach tomato (tomato)
	Use knife (knife and tomato)
	Pick up tomato (tomato)
	Approach pot (pot)
	Put the tomato into pot (pot and tomato)
Pour Coke	Approach fridge (fridge)
	Take coke from fridge (coke)
	Close fridge (fridge)
	Approach a cup (cup)
	Pour coke into the cup (cup and coke)

Table 4.4: Dataset Tasks part1

Task	Task definition(attention objects in parenthesis)
Make Juice	Approach Fridge (Fridge)
	Open Fridge (Fridge)
	Take up orange (orange)
	Close Fridge(Fridge)
	Approach board (board)
	Use knife (knife and orange)
	Take the orange (orange)
	Approach juicer (juicer)
	Put the orange into juicer (orange and juicer)
Fry Steak	Approach fridge (fridge)
	Open door of fridge (fridge)
	Take the steak(bread)
	Close fridge(fridge)
	Approach pot (pot)
	Put the steak into pot (pot and steak)
	Operate stove (stove)
Make Sandwich	Approach fridge (fridge)
	Open door of fridge (fridge)
	Take the bread(bread)
	Close fridge(fridge)
	Approach plate(plate)
	Put the bread onto plate(bread and plate)
	Approach fridge (fridge)
	Open door of fridge (fridge)
	Take the ham(ham)
	Close fridge(fridge)
	Approach bread(bread)
	Put the ham onto bread (ham and bread)

Table 4.5: Dataset Tasks part2

Task	Task definition(attention objects in parenthesis)
Bake bread	Approach fridge (fridge)
	Open door of fridge (fridge)
	Take the bread(bread)
	Close fridge(fridge)
	Approach oven (stove)
	Use oven (stove, bread)
Cut Meat	Approach Fridge(Fridge)
	Open fridge door(fridge)
	take the beef(beef)
	Close fridge door(fridge)
	Approach Board(Board)
	use Knife (knife, beef)
Turn on the light	Approach light switch (light switch)
	push/pull switch (light switch)

Table 4.6: Dataset Tasks part3

Class	Num. of Instances	Accuracy(mAp)
Bread	4409	0.1665
Cut Board	29828	0.0340
Microwave	26144	0.4670
Fridge	28912	0.7911
Light Switch	8146	0.4900
Coke	3014	0.2539
Stove	28640	0.6707
Juicer	30402	0.9961
Coffee Machine	21487	0.0351
Plate	57606	0.7812
Ham	838	0.0021
Beef	2602	0.0987
Tomato	1780	0.1883
Cup	61534	0.5512
Pot	30678	0.6622
Eggplant	1129	0.0153
Knife	29488	0.2207
Orange	1417	0.1404
Average by class		0.3647
Weighted Average		0.5390

Table 4.7: Object detection results

CHAPTER 5

Conclusion

We have designed a virtual reality system, VRKitchen, which offers physical simulation, photo-realistic rendering of multiple kitchen environments, a large set of fine-grained object manipulations, and embodied agents with human-like appearances and gestures. We have implemented toolkits for training and testing AI agents as well as for collecting human demonstrations in our system. We are also able to compile a video dataset of human demonstrations of the cooking tasks using the user interface and scripting files in the system. In the future, we plan to enrich the simulation in our system and conduct more experiments including visual representation learning, world model learning, reinforcement learning, imitation learning, visual task planning, language grounding etc.

REFERENCES

[ACM15] Pulkit Agrawal, Joao Carreira, and Jitendra Malik. "Learning to see by moving." In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[AD08] Luis von Ahn and Laura Dabbish. "Designing games with a purpose." *Communications of the ACM*, 2008.

[AN04] Pieter Abbeel and Andrew Y. Ng. "Apprenticeship learning via inverse reinforcement learning." In *Twenty-first international conference on Machine learning - ICML '04*, p. 1, 2004.

[BCC01] Igor R. Belousov, Ryad Chellali, and Gordon J. Clapworthy. "Virtual reality tools for Internet robotics." *Proceedings - IEEE International Conference on Robotics and Automation*, 2001.

[BDS18] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis." *CoRR*, **abs/1809.11096**, 2018.

[BLT16] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, Julian Schrittwieser, Keith Anderson, Sarah York, Max Cant, Adam Cain, Adrian Bolton, Stephen Gaffney, Helen King, Demis Hassabis, Shane Legg, and Stig Petersen. "DeepMind Lab." pp. 1–11, 2016.

[BPA17] Simon Brodeur, Ethan Perez, Ankesh Anand, Florian Golemo, Luca Celotti, Florian Strub, Jean Rouat, Hugo Larochelle, and Aaron Courville. "HoME: a Household Multimodal Environment." Number Nips, pp. 1–6, 2017.

[CB16] E Coumans and Y Bai. "Pybullet, a python module for physics simulation for games, robotics and machine learning." *GitHub repository*, 2016.

[CDF18] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. "Matterport3D: Learning from RGB-D data in indoor environments." *Proceedings - 2017 International Conference on 3D Vision, 3DV 2017*, pp. 667–676, 2018.

[Che12] Zhe Chen. "Object-based attention: A tutorial review." *Attention, Perception, & Psychophysics*, **74**(5):784–802, 2012.

[CSW17] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. "Realtime multi-person 2d pose estimation using part affinity fields." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7291–7299, 2017.

[CY12] Wei-Lun Chou and Su-Ling Yeh. "Object-based attention occurs regardless of object awareness." *Psychonomic bulletin & review*, **19**(2):225–231, 2012.

[FR13] Alireza Fathi and James M. Rehg. "Modeling actions through state changes." In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2013.

[FWS18] Yao Feng, Fan Wu, Xiaohu Shao, Yanfeng Wang, and Xi Zhou. "Joint 3d face reconstruction and dense alignment with position map regression network." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 534–551, 2018.

[FZZ18] Linxi Fan, Yuke Zhu, Jiren Zhu, Zihua Liu, Orien Zeng, Anchit Gupta, Joan Creus-Costa, Silvio Savarese, and Li Fei-Fei. "SURREAL: Open-Source Reinforcement Learning Framework and Robot Manipulation Benchmark." (CoRL), 2018.

[GGC16] A. Giusti, J. Guzzi, D. C. Cireşan, F. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella. "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots." *IEEE Robotics and Automation Letters*, **1**(2):661–667, July 2016.

[GRO17] Andrea de Giorgio, Mario Romero, Mauro Onori, and Lihui Wang. "Human-machine Collaboration in Virtual Reality for Adaptive Production Engineering." *Procedia Manufacturing*, 2017.

[HGD17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. "Mask R-CNN." *CoRR*, **abs/1703.06870**, 2017.

[HKB15] Andrei Haidu, Daniel Kohlsdorf, and Michael Beetz. "Learning action failure models from interactive physics-based simulations." In *IEEE International Conference on Intelligent Robots and Systems*, 2015.

[HVP17] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. "Learning from Demonstrations for Real World Reinforcement Learning." *CoRR*, **abs/1704.03732**, 2017.

[HZR16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[ILA15] Phillip Isola, Joseph J. Lim, and Edward H. Adelson. "Discovering states and transformations in image collections." In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015.

[JHH16] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. "The malmo platform for artificial intelligence experimentation." *IJCAI International Joint Conference on Artificial Intelligence*, **2016-Janua**:4246–4247, 2016.

[JWS09a] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "ImageNet: A large-scale hierarchical image database." In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[JWS09b] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "ImageNet: A large-scale hierarchical image database." In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. IEEE, jun 2009.

[KGS13] Hema Swetha Koppula, Rudhir Gupta, and Ashutosh Saxena. "Learning human activities and object affordances from rgb-d videos." *The International Journal of Robotics Research*, **32**(8):951–970, 2013.

[KH12] Alex Krizhevsky and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In *Neural Information Processing Systems*, 2012.

[KMG17] Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. "AI2-THOR: An Interactive 3D Environment for Visual AI." pp. 3–6, 2017.

[KNM01] H. Kawasaki, K. Nakayama, T. Mouri, and S. Ito. "Virtual teaching based on hand manipulability for multi-fingered robots." *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, 2001.

[KWR17] Michal Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaskowski. "ViZDoom: A Doom-based AI research platform for visual reinforcement learning." *IEEE Conference on Computatonal Intelligence and Games, CIG*, 2017.

[LGF16] Adam Lerer, Sam Gross, and Rob Fergus. "Learning Physical Intuition of Block Towers by Example." Technical report, 2016.

[LGG17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. "Focal loss for dense object detection." In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

[LGS16] Jenny Lin, Xingwen Guo, Jingyu Shao, Chenfanfu Jiang, Yixin Zhu, and Song-Chun Zhu. "A virtual reality platform for dynamic human-scene interaction." In *SIGGRAPH ASIA 2016 virtual reality meets physical reality: Modelling and simulating virtual humans and environments*, p. 11. ACM, 2016.

[LRM17] Oliver Liu, Daniel Rakita, Bilge Mutlu, and Michael Gleicher. "Understanding human-robot interaction in virtual reality." In *RO-MAN 2017 - 26th IEEE International Symposium on Robot and Human Interactive Communication*, 2017.

[LWZ17] Yang Liu, Ping Wei, and Song Chun Zhu. "Jointly Recognizing Object Fluents and Tasks in Egocentric Videos." In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-Octob, pp. 2943–2951, 2017.

[MHL17] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J. Davison. "SceneNet RGB-D: Can 5M Synthetic Images Beat Generic ImageNet Pre-training on Indoor Segmentation?" In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-Octob, pp. 2697–2706, 2017.

[NKK11] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. "Multimodal deep learning." In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 689–696, 2011.

[NR00] Andrew Y. Ng and Stuart Russell. "Algorithms for inverse reinforcement learning." In *International Conference on Machine Learning (ICML)*, 2000.

[PAR18] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. "Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research." feb 2018.

[PR14] Arezoo Pooresmaeili and Pieter R Roelfsema. "A growth-cone model for the spread of object-based attention during contour grouping." *Current Biology*, **24**(24):2869–2877, 2014.

[PRB18] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. "VirtualHome: Simulating Household Activities via Programs." jun 2018.

[QT08] Faisal Qureshi and Demetri Terzopoulos. "Smart camera networks in virtual reality." *Proceedings of the IEEE*, **96**(10):1640–1656, 2008.

[QY16] Weichao Qiu and Alan Yuille. "UnrealCV: Connecting computer vision to unreal engine.", 2016.

[RAA12] Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele. "A database for fine grained activity detection of cooking activities." In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1194–1201, 2012.

[RBE17] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. "Snorkel: Rapid training data creation with weak supervision." *Proceedings of the VLDB Endowment*, **11**(3):269–282, 2017.

[RCR18] Nataniel Ruiz, Eunji Chong, and James M Rehg. "Fine-grained head pose estimation without keypoints." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 2074–2083, 2018.

[RGB10] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning." In *Proceedings of AISTATS*, volume 15, pp. 627–635, 2010.

[RT00] Tamer F Rabie and Demetri Terzopoulos. "Active perception in virtual humans." In *Vision Interface*, volume 2000, 2000.

[SCD17] Manolis Savva, Angel X. Chang, Alexey Dosovitskiy, Thomas Funkhouser, and Vladlen Koltun. "MINOS: Multimodal Indoor Simulator for Navigation in Complex Environments." pp. 1–14, 2017.

[See11] Axel Seemann. *Joint attention: New developments in psychology, philosophy of mind, and social neuroscience*. MIT Press, 2011.

[SG05] Linda Smith and Michael Gasser. "The development of embodied cognition: Six lessons from babies." *Artificial life*, **11**(1-2):13–29, 2005.

[SS08] Umar Syed and Robert E Schapire. "A Game-Theoretic Approach to Apprenticeship Learning." In *Advances in Neural Information Processing Systems 20*, volume 20, pp. 1–8, 2008.

[SSS17] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. "Mastering the game of Go without human knowledge." *Nature*, **550**(7676):354–359, oct 2017.

[SYZ17] Shuran Song, Fisher Yu, Andy Zeng, Angel X. Chang, Manolis Savva, and Thomas Funkhouser. "Semantic scene completion from a single depth image." In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pp. 190–198, 2017.

[TET12] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control." In *IEEE International Conference on Intelligent Robots and Systems*, 2012.

[TR95] Demetri Terzopoulos and Tamer F Rabie. "Animat vision: Active vision in artificial animals." In *Proceedings of IEEE International Conference on Computer Vision*, pp. 801–808. IEEE, 1995.

[WWG18] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. "Building Generalizable Agents with a Realistic and Rich 3D Environment." jan 2018.

[XZH18] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. "Gibson Env: Real-World Perception for Embodied Agents." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[ZGK17] Yuke Zhu, Daniel Gordon, Eric Kolve, Dieter Fox, Li Fei-Fei, Abhinav Gupta, Roozbeh Mottaghi, and Ali Farhadi. "Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning." *Proceedings of the IEEE International Conference on Computer Vision*, **2017-Octob**(1):483–492, 2017.

[ZMB08] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. "Maximum Entropy Inverse Reinforcement Learning." In *Proc. AAAI*, pp. 1433–1438, 2008.