UnrealCV: Virtual Worlds for Computer Vision

Weichao Qiu[1], Fangwei Zhong[2], Yi Zhang[1], Siyuan Qiao[1],
Zihao Xiao[1], Tae Soo Kim[1], Yizhou Wang[2], Alan Yuille[1]
1. Johns Hopkins University, 2. Peking University
{qiuwch,zfw1226,edwardz.amg,joe.siyuan.qiao}@gmail.com
{zxiao10,tkim60}@jhu.edu,yizhou.wang@pku.edu.cn,alan.l.yuille@gmail.com

ABSTRACT

UnrealCV[1] is a project to help computer vision researchers build virtual worlds using Unreal Engine 4 (UE4). It extends UE4 with a plugin by providing (1) A set of UnrealCV commands to interact with the virtual world. (2) Communication between UE4 and an external program, such as Caffe. UnrealCV can be used in two ways. The first one is using a compiled game binary with UnrealCV embedded. This is as simple as running a game, no knowledge of Unreal Engine is required. The second is installing UnrealCV plugin to Unreal Engine 4 (UE4) and use the editor of UE4 to build a new virtual world. UnrealCV is an open-source software under the MIT license. Since the initial release in September 2016, it has gathered an active community of users, including students and researchers.

CCS CONCEPTS

• **Software and its engineering** → **Software libraries and repositories**; • **Computing methodologies** → *Computer vision*; Reinforcement learning;

KEYWORDS

computer vision, unreal engine, virtual reality

1 INTRODUCTION

Realistic virtual worlds are created by the movie and game industry to tell stories that are difficult to see in the real world. These photo-realistic virtual worlds are also useful for research, they can be used for many tasks that the real world is not capable of, such as generating large number of images with hard-to-get annotations[2], simulate images of extreme weather conditions[8], stress test a vision algorithm by creating hazardous factors[10] and do expensive robotics training[11].

Constructing a *virtual world* from a high-fidelity 3D video game or CG movie is attractive for computer vision researchers. This motives the creation of OpenAI universe [1], Malmo [3], VizDoom [4], etc. But modifying a video game or CG movie has two limitations. First, open source video games, such as Doom, have limited visual realism, while the newest video games such as GTA do not have good APIs to access its internal data and sometimes have license issues. Second, the modifications of one video game can not be transfered to others and needs to be redone case by case. In order to use a video game as a virtual world for computer vision researchers, some extensions need to be done: (1) the video game needs to be programmly accessible through an API, so that an AI agent can communicate with it (2) the information in the virtual worlds need to be extracted to achieve certain tasks, such as ground truth generation or giving rewards based on the action of agents.

Constructing a virtual world using a game engine can solve the problems in modifying a game. *Game engine* is a software framework for creation of video games. If the extension is done in the game engine, then the game engine can be used to produce a video game (virtual world) that can meet the requirements of researchers. Unreal Engine (UE4) is a popular game engine for creating high-fidelity video games and one of the best choices for virtual reality developments. The rich 3D resources and full access to its source code motivates us to extend it for creating virtual worlds that are useful for researchers. This idea leads to the creation of UnrealCV.

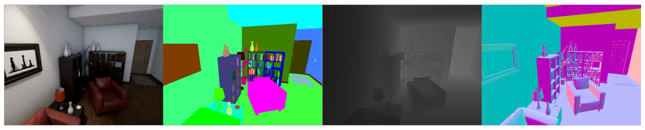

Figure 1: A synthetic image and its ground truth generated using UnrealCV. This virtual room is from the technical demo RealisticRendering created by Epic Games. From left to right are the synthetic image, object instance mask, depth, surface normal

UnrealCV extends UE4 to make it able to create virtual worlds that can meet requirements of researchers. Two additional features are provided to UE4. First, it provides a communication layer, so that a program can communicate with UE4 to extract information. Second, commonly used computer vision features are provided, such as ground truth generation shown in Fig. 1. A preliminary version of UnrealCV is published in [7]. The design and implementation of UnrealCV satisfies these requirements: (1) Easy to install and use and can support major platforms and languages for research, such as Python and Linux. Researchers are not game developers and UnrealCV can be used without knowing anything about UE4 and game design. (2) Extensible to include more features that researchers need. (3) Compatible with the up-to-date version of Unreal Engine to benefit from the improvements from the upstream.

Virtual worlds have been attractive for AI researchers since the birth of AI, but the poor visual realism limited the adoption. Creating realistic virtual worlds is difficult and expensive before,

[1]https://unrealcv.org

ACM ISBN ISBN 978-1-4503-4906-2/17/10…$15.00
https://doi.org/10.1145/3123266.3129396

but this has been changed in the past few years. The cost has been hugely reduced by 3D marketplace, better 3D reconstruction tools. The prosper of virtual reality encourages more people to create and share high quality 3D contents. The visual realism improvement and cost reduction is critical for using virtual worlds in vision research. Unreal Engine is one of the best choices for virtual reality development and very open to the community. It hugely reduce the cost for accessing high-quality computer graphics. UnrealCV provides a bridge between computer vision and Unreal Engine to help researchers easily utilize these high quality computer graphics resource.

2 HIGHLIGHTS OF UNREALCV

UnrealCV provides a set of tools to make the creation and sharing of virtual worlds easier. It extends Unreal Engine to make the popular game engine able to construct virtual worlds that can meet requirements of researchers out-of-the-box. It has the following advantages.

Easy to use Game binaries created by UnrealCV are provided, so that researchers can start using a virtual world without any knowledge of UE4. Using a virtual world is as simple as download a game and run it. These game binaries are organized with a model zoo.

UnrealCV supports popular platforms (Linux, Windows) and Python. Experimental support for Mac and MATLAB are included. Any language that can use socket can be easily supported. It is compatible with a wide range of UE4 versions. It can also be used together with UE4 Editor to design new virtual worlds.

In order to make the virtual environment easy to deploy, we also provide docker images to simplify the installation and creation of distributed learning systems. The docker image is also used for automatically building and testing of the software.

Powerful and extensible UnrealCV makes the information of a virtual world accessible through URI (unique resource identifier), each resource in the virtual world is associated with a URI. For example, the object location can be accessed with `/object/[id]/location`. More resources of the virtual world can be exposed by extending the UnrealCV.

The URI is defined in a hierarchical modular way. For example, `/light/[name]/intensity` is used for the light intensity, a new URI `/light/[name]/color` to access the light color can be added without affecting existing ones.

Some information of the virtual world is not directly accessible, but needs to be computed by UnrealCV, for example `/camera/[id]/object_mask` is the object segmentation mask shown in Fig. 1. There are also some abstract resource for special purposes, such as `/action/keyboard` to simulate a keyboard input. The URI is used in the RESTful commands of UnrealCV which is described in Sec. 3.2. The commands can be used to achieve simple tasks, such as generating an image dataset, or be combined to achieve a complex task such as reinforcement learning, which is described in Sec. 4.

We are not only actively working with our collaborators to add new features, but also provide documentation to show how to implement new commands[2]. UnrealCV is open-source and can be extended by anyone. Unit tests are provided to make this easier.

[2]http://docs.unrealcv.org/en/master/plugin/develop.html

Well documented and tested Tutorials are provided from simple task such as image dataset generation to complex task such as reinforcement learning. API documentation of UnrealCV is provided to help others extend the software.

A lot of tutorials of designing video games can be found for Unreal Engine, but not very relevant for the research purpose. In the UnrealCV project, we also organized relevant tutorials to research. A list of research projects that using virtual worlds for computer vision is maintained by the team as a separate project[3] to help researchers to compare and find useful tools.

Model zoo Inspired by the idea of model zoo from Caffe, we created a model zoo for virtual worlds. It is a place for sharing virtual worlds.

The model zoo makes it easy to use a virtual world without the knowledge of UE4 and no need to purchase 3D models. According to the license, it is not possible to distribute source content directly, but the binary can be released. The link to the original source content will be provided, so that researchers who require the source content can buy the 3D models and use them within the UE4 editor.

Initially, six virtual worlds are released for data generation and algorithm diagnosis[7, 10]. The instruction for releasing new virtual environments are included to encourage the community to create new virtual worlds and share them[4].

A virtual world can be used for many different tasks. Sharing virtual worlds is a new concept for computer vision dataset. We will also share images we generated from the virtual world. Moreover, scripts are provided to generate more images from the virtual worlds. A virtual world also enables new tasks that traditional image/video datasets can not support, such as reinforcement learning and active vision. UnrealCV is an underlying tool to make virtual worlds easy to construct and share.

2.1 Compare to Related Software

Unreal Engine has been used in a few research projects [5, 9]. The game engine was extended to meet the requirements of a specific research project. But customization of Unreal Engine makes it harder to adapt to newer versions. There is no model zoo for sharing virtual worlds, so the usage of UE4 requires a considerate amount of preparation.

We aim to provide a flexible and extensible API that can be combined to achieve different goals. While AirSim is focused on robotics simulation, which requires more customization with UE4 and more complex to get started. The design principles and goal of UnrealCV and AirSim are different, but there are many components are in common and a collaboration between these two projects will be beneficial for the community. ISSAC is also a robot simulator based on UE4. It is recently announced by NVidia[5], but no technical details are available yet.

3 ARCHITECTURE

UnrealCV provides an easy way to read and modify resources of a virtual world. The user imports the *UnrealCV client* as a library into his code. The UnrealCV client will be used to request information from the virtual world, for example, reading the 3D location of an object. This is done through sending an *UnrealCV command* to the

[3]https://github.com/unrealcv/synthetic-computer-vision
[4]http://docs.unrealcv.org/en/master/plugin/package.html
[5]https://www.nvidia.com/en-us/deep-learning-ai/industries/robotics/

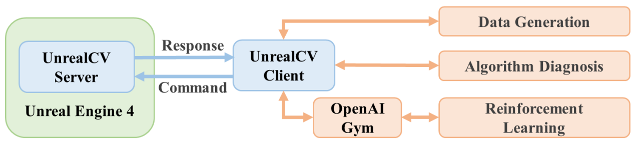

Figure 2: UnrealCV consists of the UnrealCV server which is embedded into a game during compilation. External programs use the UnrealCV client to communicate with UE4 games. See text for details.

virtual world. The *UnrealCV server* processes the command and returns requested information. This procedure is shown in Fig. 2.

3.1 UnrealCV Server and Client

UnrealCV has two components, the server and the client. The server uses the C++ API of Unreal Engine to access information of the virtual world. The client is a library to communicate with the server by sending commands and parsing responses. The server and client communicates using TCP. The communication protocol is defined by a set of UnrealCV commands, described in Sec. 3.2.

The server implements features useful for computer vision researchers, such as ground truth generation. These information can be computed from the internal data of Unreal Engine, but they are missing from UE4.

The client is a small library that can be easily integrated into other code. We demonstrate this with a simple image generation tutorial which uses the client in a short script and a complex reinforcement learning demo, which involves tensorflow and needs to be run for days in Sec. 4. Python version of UnrealCV client is provided and can be installed through the package manager. Experimental support for MATLAB is provided. The communication protocol is documented, making it easier to implement a client for a different language that can support socket.

The communication between the server and the client are bidirectional. The client requests information from the server and the server can also send a message to the client to notify an event. This makes it easier to design tasks which require triggering events, for example, sending collision notification in a reinforcement learning task. Currently only one concurrent client is supported. Multiple clients will be useful to simulate multiple intelligent agents interacting in one virtual environment. This will be our future work.

3.2 Command System

The resources of a virtual world are exposed through a set of commands. The command is designed in a RESTful style. A typical command, for example, `vset /camera/[id]/location [x] [y] [z]`, consists of three parts. The first part is the operation to perform, it can be either `vget` or `vset`. The second part is an URI indicating the resource of the 3D world to operate with. The third part is extra parameters, such as the new location of an object.

These commands are documented[6] and tutorials are provided to show how to combine commands to achieve certain research tasks.

The command is an abstraction of what is needed from a virtual world. This concept can be applied to other softwares, creating a unified API for the research community, which is beyond Unreal Engine.

[6]http://docs.unrealcv.org/en/master/reference/commands.html

3.3 Virtual World Creation using UnrealCV

As a game engine, Unreal Engine provides tools for packaging a game. *Packaging* produces a video game binary containing rendering and physics simulation code, 3D models and tasks for players. UnrealCV adds computer vision related code to the game binary during packaging, making the binary can be used as a virtual world. The information of the virtual world will be exposed through UnrealCV commands.

3.4 UE4 Editor Plugin

Researchers will not be satisfied with existing virtual worlds once they have an idea that existing ones cannot support. For example, we may want to place a glass door into the room to see whether the robot can successfully avoid it, but no existing virtual worlds provide such a setup. Therefore it is necessary to have a tool for designing virtual worlds.

Unreal Engine Editor (UE4Editor) is a useful tool for editing a 3D environment, similar to 3DS max and Maya. It can be used to add/delete objects, modify details of the scene. UnrealCV does not provide the complex 3D editing that UE4Editor can provide, but UnrealCV can be used as a plugin of the editor, enabling researchers to combine the power of both. With UnrealCV plugin installed, UnrealCV server code will run inside the UE4Editor to extend its functions.

4 APPLICATIONS AND EXAMPLES

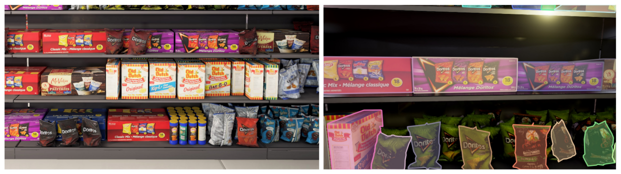

Figure 3: A virtual supermarket shelf and its annotation visualized by MSCOCO API. The left is the synthetic image and the right is the segmentation mask, see [6] for more details.

Data Generation A virtual world can be used to generate large amount of images with ground truth. A virtual world can also produce ground truths that are hard to annotate in a real image. For example, for the grocery shelf image in Fig. 3, it is very time consuming to annotate the object mask of individual items. It is even more challenging to annotate the occluded regions of objects. In order to get a benchmark image dataset of supermarket for training, a virtual supermarket was constructed in [6] using UnrealCV. The object placement, lighting were randomized to increase the variety of the data and prevent over-fitting for training object detectors. A tool for randomly placing objects on the shelf is provided, which can be modified to randomly generate other scenes. This tool is also included in UnrealCV. The object detector is trained only with synthetic data and can work well on real images, see [6] for detailed results. The synthetic image and visualization of ground truth is shown in Fig. 3.

Algorithm Diagnosis The preliminary paper of UnrealCV [7] gave an example of diagnosing an object detector. It showed the performance degeneration when viewpoint varied.

A more extended diagnosis study was done for stereo algorithms in [10]. In a virtual world, it is convenient to adjust material properties, such as texture, surface reflectance and transparency, to create certain degrees of hazardous regions for stereo matching algorithms. This enables us to densely sample the degrees of hazardous factors to analyze their effect on stereo algorithms, see Fig. 4. Findings on synthetic datasets are further verified by constructing small real world datasets tuned to these precise degrees of hazard.

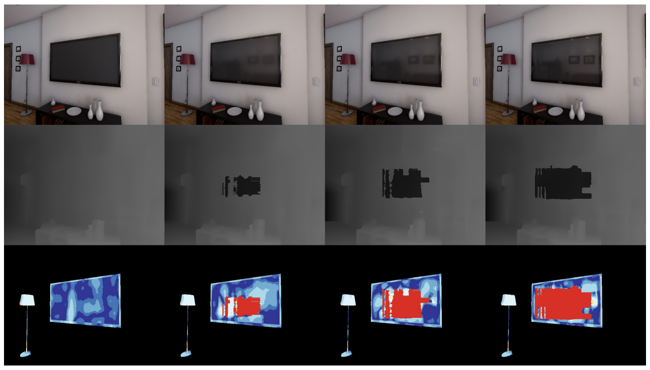
Figure 4: Controlling the specularity of the TV, from top to bottom are the input image, disparity estimation and error compared with ground truth. The subtle visual difference in the first row is challenging for state-of-art methods, see [10] for more details.

Reinforcement Learning It is expensive, time-consuming and in some cases dangerous to train agents by trial-and-error in the real world. So it is attractive to train agents in a virtual world. But most of virtual worlds for reinforcement learning are far different from the real in terms of appearance. A realistic virtual world is important to help robots evolve from playing games to solving real-world problems, like grabbing objects or visual navigation.

UnrealCV provides a new way to construct realistic virtual environments. The high-fidelity rendering makes it easier to transfer from virtual to real. We provide an UnrealCV based OpenAI Gym interface to help researchers integrate RL algorithms with a virtual world. This reinforcement learning interface can be used without knowledge of UE4. Rich internal data from UnrealCV makes it flexible to design reward function for various tasks. To help users get started with the installation and usage, we provide a tutorial and a visual navigation example shown in Fig. 5, in our project page.

5 AVAILABILITY

UnrealCV is developed in Windows and tested for Windows and Linux. The project is under MIT license and all source code can be accessed in the Github.

6 CONCLUSION

This paper presented UnrealCV, a tool to use UE4 to construct realistic virtual worlds. Realistic virtual worlds can be used for generating large image/video datasets with detailed ground truth. They can also be used for tasks that image/video datasets can not support, such as reinforcement learning and active vision. UnrealCV helps computer vision researchers use realistic graphics resources from the game, virtual reality and architecture visualization industries. It

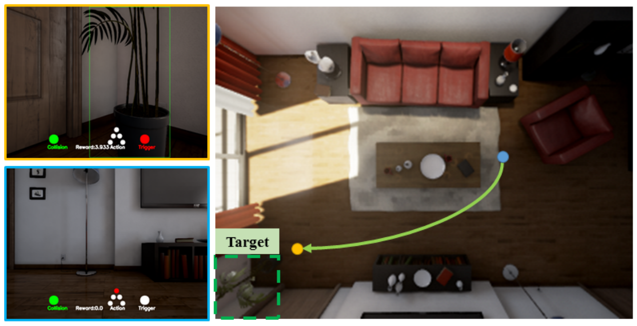

Figure 5: Visual navigation in a realistic room. The top view shows the room layout, target object and a path learned by DRL to find target while avoiding collision. The left images show agent's first-person view at the begin and end of the path.

provides an easy-to-use and extensible API for researchers with no UE4 knowledge and hosts a model zoo to make virtual worlds easy to create and share. We hope it can be useful for the community and welcome feedback and contribution.

ACKNOWLEDGMENTS

We would like to thank the open source community for the contribution, feedback and suggestion. This work is supported by NSF award CCF-1317376 and NSFC-61273304

REFERENCES

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. (2016). arXiv:arXiv:1606.01540
[2] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. 2012. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision*. Springer, 611–625.
[3] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. 2016. The malmo platform for artificial intelligence experimentation. *IJCAI International Joint Conference on Artificial Intelligence* 2016-January (2016), 4246–4247.
[4] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. 2016. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*. IEEE, 1–8.
[5] Adam Lerer, Sam Gross, and Rob Fergus. 2016. Learning Physical Intuition of Block Towers by Example. (2016). arXiv:1603.01312 http://arxiv.org/abs/1603.01312
[6] Siyuan Qiao, Wei Shen, Weichao Qiu, Chenxi Liu, and Alan Yuille. 2017. ScaleNet: Guiding Object Proposal Generation in Supermarkets and Beyond. *arXiv preprint arXiv:1704.06752* (2017).
[7] Weichao Qiu and Alan Yuille. 2016. UnrealCV: Connecting Computer Vision to Unreal Engine. In *Computer Vision–ECCV 2016 Workshops*. Springer International Publishing, 909–916.
[8] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. 2016. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3234–3243.
[9] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. 2017. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. (2017). arXiv:arXiv:1705.05065
[10] Yi Zhang, Weichao Qiu, Qi Chen, Xiaolin Hu, and Alan Yuille. 2016. UnrealStereo: A Synthetic Dataset for Analyzing Stereo Vision. *arXiv preprint arXiv:1612.04647* (2016).
[11] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. 2017. Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. In *International Conference on Robotics and Automation (ICRA)*.