UnrealCV

Release 0.3.8

Jun 14, 2017

Tutorials

UnrealCV helps 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 or tensorflow.

Fig. 1: Images generated from the technical demo *RealisticRendering*

Currently, UnrealCV plugin supports UE4 (4.12, 4.13, 4.14) in Windows, Linux and Mac, more details about supported platforms in *this page*. UnrealCV client supports python and MATLAB (experimental). Read *getting started* to learn how to use UnrealCV.

Tutorials are provided to help you get familiar with UnrealCV quickly and more technical details are documented in the reference section.

CHAPTER 1

Tutorials

- *Getting started* - The basics of using UnrealCV, start from here.
- *Generate Images* - Generate an image dataset with ground truth.
- *Integration with an AI program* - Show how to use images from a video game for testing faster RCNN.
- *Install UnrealCV Plugin* - How to install UnrealCV plugin into UE4.

CHAPTER 2

Reference

- *The architecture of UnrealCV* - Technical details of UnrealCV.
- *Command System* - All available commands provided in UnrealCV.

For the doc of a specific version of unrealcv, please see the version page

Getting started

This page introduces *UnrealCV commands* and how to use them to perform basic tasks. We also show how to use a python script to control an UnrealCV embedded game through these commands.

Download a game binary

This tutorial will use a game binary to demonstrate UnrealCV commands. You can also *create your own game using UnrealCV plugin*.

First you need to download a game binary from *our model zoo*. For this tutorial, please download *RealisticRendering*. After unzip and run the binary, you are expected to see a screen like this. The game will be started in a window mode with resolution 640x480, you can change the resolution by *changing the configuration file* of UnrealCV.

RealisticRendering

Use mouse to look around and use keys `w a s d` to navigate, use `q e` to level the camera up and down. If you want to release mouse cursor from the game, press ‘ (the key on top of tab).

UnrealCV commands

UnrealCV provides a set of commands for computer vision research. These commands are used to perform various tasks, such as control camera and get ground truth. The table below summaries commands used in this tutorial. The complete list can be found in *the command list* in the reference section.

Fig. 2.1: Initial screen of the game

Command	Help
vset /viewmode [viewmode_name]	Set ViewMode to (lit, normal, depth, object_mask)
vget /camera/0/lit	Save image to disk and return filename
vset /camera/0/location [x] [y] [z]	Set camera location

Try UnrealCV commands

Unreal Engine provides a built-in console to help developers to debug games. This built-in console is a convenient way of trying UnrealCV commands. To open the console, press ‘ (the key on top of tab) twice, a console will pop out, as shown in Fig. 2.2. Type in `vset /viewmode object_mask` you are expected to see the object instance mask. Use `vset /viewmode lit` to switch back to normal rendering setting.

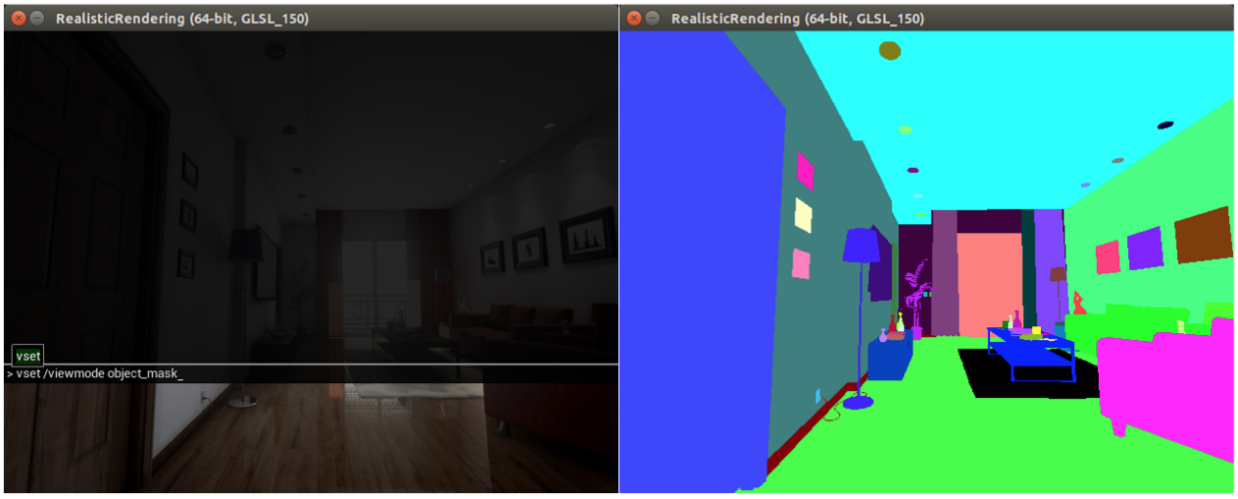


Fig. 2.2: Use console to try UnrealCV commands

Use python client to execute commands

If we want to generate a large-scale synthetic dataset, or do active tasks, such as reinforcement learning, in this virtual world. We need to allow an intelligent agent to perceive, navigate and interact in the scene. We provide UnrealCV client to enable other programs to communicate with this virtual world. The client will use a *plain-text protocol* to exchange information with the game.

Here we use the python client for illustration. If you are looking for a MATLAB client, please see *the MATLAB client*.

First, we need to install the python client library.

Install UnrealCV python library

```
pip install unrealcv
```

Generate some images from the scene

```
from unrealcv import client
client.connect() # Connect to the game
if not client.isconnected(): # Check if the connection is successfully established
  print 'UnrealCV server is not running. Run the game from http://unrealcv.github.io␣
↪first.'
else:
  filename = client.request('vget /camera/0/lit')
  filename = client.request('vget /camera/0/depth depth.exr')
```

You can find this example in client/examples/10lines.py.

If you encountered any errors following this tutorial, please see *the diagnosis* page to find a solution.

Next: Use UnrealCV in the game mode or plugin mode?

For the game mode, you can use a compiled game binary. You can freely control the camera in this game and generate images and ground truth from it. But it is not easy to change the scene, such as add more objects or change the material properties. If you have access to an UE4 project and know how to use the UE4Editor, you can install the plugin to UE4Editor, so that you can combine the power of UE4Editor and UnrealCV to create new virtual worlds for research.

Tutorials

- *How to generate an image dataset*
- *Integrate with a deep learning framework*
- *Use the plugin in UE4Editor*
- *Modify code and add a new command*

Articles

- To fully understand how does UnrealCV work and the technical details, please read its *architecture* or our paper. For a complete list of available commands, please read *the command list* in the reference section.

Generate Images

This ipython notebook demonstrates how to generate an image dataset with rich ground truth from a virtual environment.

Load some python libraries The dependencies for this tutorials are PIL, Numpy, Matplotlib

```
from __future__ import division, absolute_import, print_function
import os, sys, time, re, json
import numpy as np
import matplotlib.pyplot as plt

imread = plt.imread
def imread8(im_file):
    ''' Read image as a 8-bit numpy array '''
    im = np.asarray(Image.open(im_file))
    return im

def read_png(res):
    import StringIO, PIL.Image
    img = PIL.Image.open(StringIO.StringIO(res))
    return np.asarray(img)

def read_npy(res):
    import StringIO
    return np.load(StringIO.StringIO(res))
```

Connect to the game

Load unrealcv python client, do `pip install unrealcv` first.

```
from unrealcv import client
client.connect()
if not client.isconnected():
    print('UnrealCV server is not running. Run the game downloaded from http://
↪unrealcv.github.io first.')
```

Make sure the connection works well

```
res = client.request('vget /unrealcv/status')
print(res)
```

Out:

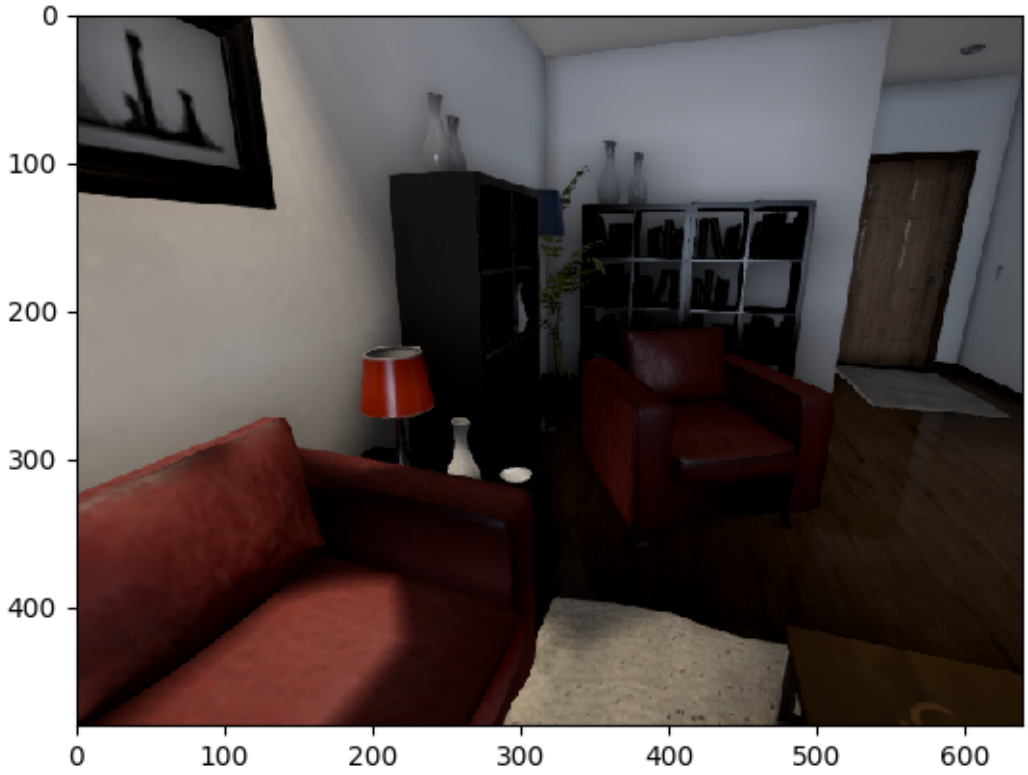
```
Is Listening
Client Connected
9000
Configuration
Config file: /home/unrealcv/LinuxNoEditor/RealisticRendering/Binaries/Linux/unrealcv.
↪ini
Port: 9000
Width: 640
Height: 480
```

Load a camera trajectory

```
traj_file = './camera_traj.json' # Relative to this python script
import json; camera_trajectory = json.load(open(traj_file))
# We will show how to record a camera trajectory in another tutorial
```

Render an image

```
idx = 1
loc, rot = camera_trajectory[idx]
# Set position of the first camera
client.request('vset /camera/0/location {x} {y} {z}'.format(**loc))
client.request('vset /camera/0/rotation {pitch} {yaw} {roll}'.format(**rot))

# Get image
res = client.request('vget /camera/0/lit lit.png')
print('The image is saved to %s' % res)

# It is also possible to get the png directly without saving to a file
res = client.request('vget /camera/0/lit png')
im = read_png(res)
print(im.shape)

# Visualize the image we just captured
plt.imshow(im)
```



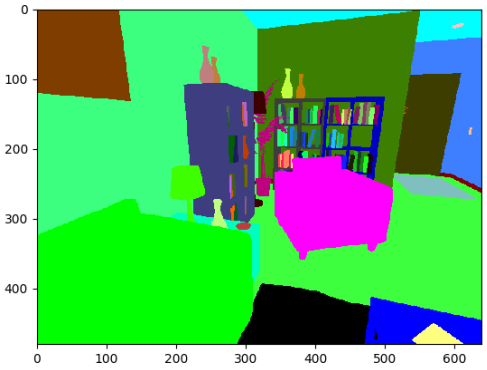
Out:

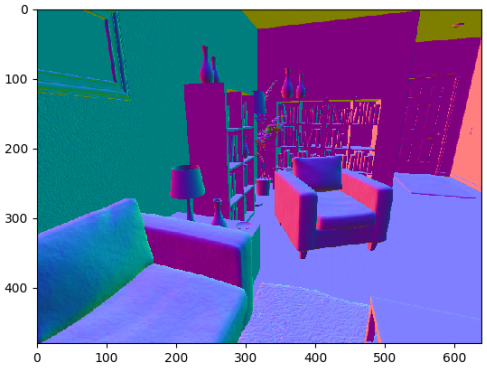
```
The image is saved to /home/unrealcv/LinuxNoEditor/RealisticRendering/Binaries/Linux/
↪lit.png
(480, 640, 4)
```

Ground truth generation

Generate ground truth from this virtual scene

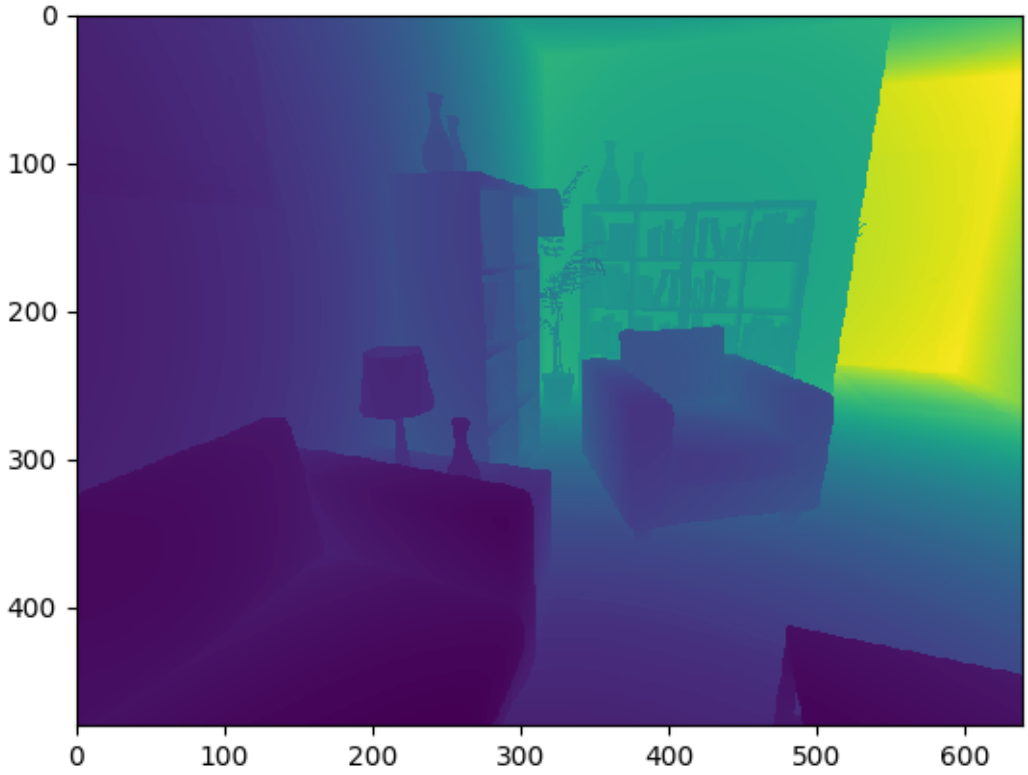
```
res = client.request('vget /camera/0/object_mask png')
object_mask = read_png(res)
res = client.request('vget /camera/0/normal png')
normal = read_png(res)

# Visualize the captured ground truth
plt.imshow(object_mask)
plt.figure()
plt.imshow(normal)
```





Depth is retrieved as a numpy array For UnrealCV < v0.3.8, the depth is saved as an exr file, but this has two issues. 1. Exr is not well supported in Linux 2. It depends on OpenCV to read exr file, which is hard to install

```
res = client.request('vget /camera/0/depth npy')
depth = read_npy(res)
plt.imshow(depth)
```



Get object information

List all the objects appeared in the virtual scene

```
scene_objects = client.request('vget /objects').split(' ')
print('There are %d objects in this scene' % len(scene_objects))

# TODO: replace this with a better implementation
class Color(object):
    ''' A utility class to parse color value '''
    regexp = re.compile('\(R=(.*),G=(.*),B=(.*),A=(.*)\)')
    def __init__(self, color_str):
        self.color_str = color_str
        match = self.regexp.match(color_str)
        (self.R, self.G, self.B, self.A) = [int(match.group(i)) for i in range(1,5)]

    def __repr__(self):
        return self.color_str

color_mapping = {}
inverse_color_mapping = {}
num_objects = len(scene_objects)
for idx in range(num_objects):
    objname = scene_objects[idx]
    color = Color(client.request('vget /object/%s/color' % objname))
```

```
    idx = color.R * 256 * 256 + color.G * 256 + color.B
    color_mapping[objname] = idx
    inverse_color_mapping[idx] = objname

    if idx % (num_objects / 10) == 0:
        sys.stdout.write('.')
        sys.stdout.flush()
```

Out:

```
There are 296 objects in this scene
.
```

How many objects in this frame

```
mask = object_mask
mask_idx = mask[:,:,0] * 256 * 256 + mask[:,:,1] * 256 + mask[:,:,2]

unique_idx = list(set(mask_idx.flatten()))
print('There are %d objects in this image' % len(unique_idx))

obj_names = [inverse_color_mapping.get(k) for k in unique_idx]
print(obj_names)
```

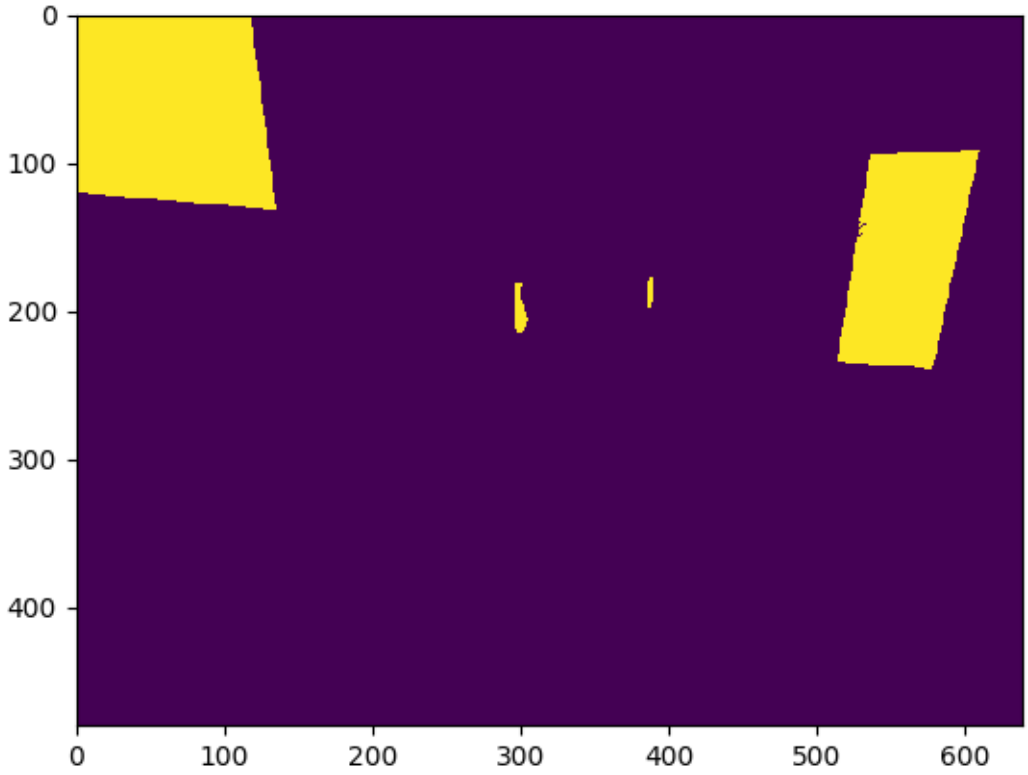
Out:

```
There are 48 objects in this image
[None, 'Mug_30', 'Carpet_5', 'BookLP_142', None, 'BookLP_140', 'Couch_13', None, 'SM_
↪Shelving_10', 'BookLP_141', None, None, 'SM_Railing_35', 'BookLP_176', None, 'SM_
↪Railing_33', 'Switch_2', 'BookLP_104', 'SM_CoffeeTable_14', None, 'SM_Railing_34',␣
↪None, 'SM_Shelving_9', None, None, 'SM_Shelving_8', None, None, None, None, None,␣
↪None, None, None, None, 'BookLP_108', 'BookLP_106', 'EditorPlane_34', 'BookLP_144',␣
↪None, 'SM_Room_7', None, 'BookLP_105', 'EditorPlane_24', None, 'EditorPlane_31',␣
↪None, 'EditorPlane_25']
```

Show info of an object

Print an object

```
obj_idx = 0
obj_name = obj_names[obj_idx]
print('Show the object mask of %s' % obj_name)
mask = (mask_idx == unique_idx[obj_idx])
plt.imshow(mask)
```



Out:

```
Show the object mask of None
```

Clean up resources

```
client.disconnect()
```

Total running time of the script: (0 minutes 11.202 seconds)

Download Python source code: generate_images_tutorial.py

Download Jupyter notebook: generate_images_tutorial.ipynb

Generated by Sphinx-Gallery

Integration with an AI program

This is a video showing connecting Faster-RCNN with a video game for diagnosis.

Warning: We are working on a python layer for Caffe and will include it in the next release.

Read *getting started* first and know how to use the compiled binary. Faster-RCNN is an algorithm to detect objects in an image. The original paper is published here. In this tutorial we show how to run Faster-RCNN in the virtual scene. You can observe the Faster-RCNN can do a pretty good job, but still consistently making some mistakes, for example, detect photos as TV monitor.

Run the demo

To run the program illustrated in the video.

1. Install py-faster-rcnn. Follow the instruction to setup. Make sure you can run `tools/demo.py`
2. Run RealisticRendering in the *RealisticRendering*
3. Modify `rcnn_path` in the script `client/examples/faster-rcnn.py` to point to your Faster-RCNN installation, then run it.

```
cd client/examples
python faster-rcnn.py
```

After running `faster-rcnn.py`, you are expected to see a message showing the client successfully connected to the game. When you clicked mouse in the game, a frame will be sent to Faster RCNN for detection. Using the same technique shown in this tutorial, algorithms used for depth estimation, object recognition can be also easily integrated.

Here we show how to do testing in a virtual scene. If you are interested in training model with synthetic images. Please refer to the tutorial about *generating images*.

Install UnrealCV Plugin

This page briefly describes how to install UnrealCV as a UE4 plugin. Make sure you read *getting started* before trying to use the plugin.

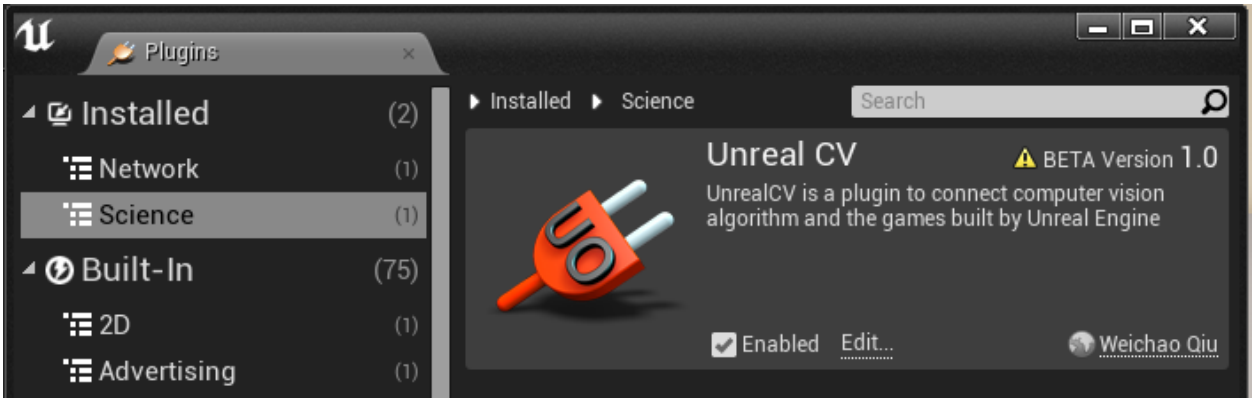
Use compiled plugin binary

You can download compiled UnrealCV binaries from our github release page. Then copy the compiled binaries to the plugins folder to install it. Build it yourself by following the *Compile from source code*. You can install the plugin to either a game project or to UE4 engine.

- **Install to project**
  - Go to project folder which contains `[ProjectName].uproject`
  - Create a folder called `Plugins`
  - Put `UnrealCV` folder into the `Plugins` folder.
- **Install to Unreal Engine**
  - Go to the plugin folder of Unreal Engine which is `Engine/Plugins`
  - Put `UnrealCV` folder into the `Plugins` folder.

Note: If you choose to install to Unreal Engine, please make sure the version of the Unreal Engine is identical to the version of the compiled binaries downloaded.

Open `Menu -> Edit -> Plugins`, make sure UnrealCV is installed and enabled. You have to be in play mode before you type the commands.



Install from UE4 marketplace (coming)

For Windows and Mac user, UnrealCV will be released to the UE4 marketplace. We are still finalizing the submission to the UE4 marketplace and it will be available soon.

Compile from source code

If you want to try a version of UnrealCV not provided in our github release page, for example, you want to try some experimental features not released yet. Compiling the plugin code from source code is the only choice.

To compile UnrealCV plugin, use `build.sh` for linux and mac, use `build.bat` for windows, remember to set the path of Unreal Engine by following instructions of the script. After running this command you should see `Automation.Execute: BUILD SUCCESSFUL` and the plugin binaries will be produced in the `Plugins/UnrealCV` folder. Then you can copy the compiled plugin to a UE4 project.

Note: It will take some time to run this command. For windows, we suggest using `cmd` to run *build.bat* to make sure the path is set correctly.

If you want to modify UnrealCV code and add new features. Please refer to the *development setup*. Setting up a dev environment takes a bit more time but will make it much easier to debug and modify code.

Note: When using the plugin in the editor, it is strongly recommend to turn off the setting `Editor Preference -> General -> Misc. -> Use Less CPU when in Background`.

Special tips for Linux

In Linux, the Unreal Engine needs to be built from source code. How to compile from source code can be found in this official document Building On Linux. But the Linux version currently does not contain OpenEXR support, which is required for getting accurate depth.

To solve this, download our OpenEXR patch for linux and run `git apply 0001-Fix-openexr-support-for-linux-version.patch` after running `./GenerateProjectFiles.sh`. This dependency will be removed later.

Usage in the Editor

In UE4 editor, you can edit the scene and change the material properties. We show a few examples here using the scene RealisticRendering.

Edit Object Specularity

Select an object, e.g. the wooden floor, you want to edit in UE4 editor, and double click the image of `Element 0` at `Details -> Materials` tab to edit its property.

Edit *SpecularWood* property, e.g. increase the value to make it more specular.

The results are as follows,

Edit Object Color

Following similar steps, you can edit the color of an object.

The results are as follows,

Development

author @Ized06, @qiuwch

UnrealCV can be compiled as a plugin as shown in the *Compile from source code*, but it you want to modify the plugin source code, compiling it together with an UE4 C++ project will make debug much easier.

Create a C++ game project

UE4 has two types of game projects. Blueprint project and C++ project. We need a C++ project.

In a C++ project, the plugin code will be compiled together with the game project.

The simplest way to start is using the playground project. Playground project is a simple UE4 project to show basic features or UE4 and UnrealCV, it serves as a development base and test platform for UnrealCV team.

Get the playground project by

```
git clone --recursive https://github.com/unrealcv/playground.git
```

UnrealCV is a submodule of this repository. If you cloned the project without `--recursive` and found the folder `Plugins/unrealcv` empty. You can use `git submodule init; git submodule update` to get the UnrealCV code.

Compile the C++ project

Windows

- Install visual studio.
- To generate visual studio solution, right click `playground.uproject` and choose `Generate Visual Studio project files`.

- Open the `*.sln` solution file and build the solution. The plugin code will be compiled together with the game project.

Linux

- Compile UE4 from source code following this official instruction
- Put this to your *.bashrc* or *.zshrc*

```
# Modify to a path that specified in step 1
export UE4=/home/qiuwch/UnrealEngine/4.13
UE4Editor() { bin=${UE4}/Engine/Binaries/Linux/UE4Editor; file=`pwd`/$1; $bin $file; }
UE4GenerateProjectFiles() {
  bin=${UE4}/GenerateProjectFiles.sh; file=`pwd`/$1;
  $bin -project=${file} -game -engine;
}
```

- Generate project file and use Makefile

```
UE4GenerateProjectFiles playground.uproject
make playgroundEditor
# or make playgroundEditor-Linux-Debug
UE4Editor playground.uproject
```

Mac

Note: Need help for writing this section

Useful resources for development include:

- *The code API documentation*
- *UnrealCV architecture*

Add a new command

UnrealCV provides a set of commands for accomplishing tasks and the list is growing. But it might not be sufficient for your task. If you need any functions that is missing, you can try to implement it yourself.

The benefit of implementing an UnrealCV command are:

1. You can use the communication protocol provided by UnrealCV to exchange data between your program and UE4.
2. You can share your code with other researchers, so that it can be used by others.

Here we will walk you through how to implement a command `vset /object/[id]/rotation` to enable you set the rotation of an object.

`FExecStatus` return the exec result of this command. The result will be returned as a text string.

Available variables for a command are `GetWorld()`, `GetActor()`, , `GetLevel()`.

A new functions will be implemented in a CommandHandler. CommandDispatcher will use CommandHandler.

The configuration file

Start from UnrealCV `v0.3.1`, the config of UnrealCV can be configured in a configuration file. Use `vget / unrealcv/status` to see current configuration.

Change game resolution

The output resolution of UnrealCV is independent of the window size.

If you want to change the display resolution. In game mode, use console command `r.setres 320x240`

When use `play -> selected viewport` the resolution can not be changed, use *play -> new window editor* instead.

Change the server port

Use `vget /unrealcv/status` to get the directory of the configuration file. Then open the configuration file and modify the server port.

Package a new game

In some scenarios you will want to package your game projects into a binary, instead of using it in the editor, for example you want to deploy the game to a deep learning server without UE4 installed or share your game project with other researchers

This page briefly describes how to create a game binary with UnrealCV embedded. Some game binaries can be found in the *model zoo*

Guide to submit a binary

1. Modify UE4 config file
2. Package your game project into a binary
3. Make a pull request to modify the *Model Zoo*
4. We will review your pull request and merge the changes to the UnrealCV website

1. Modify an UE4 config file

First, you need to add a line to UE4 engine config file `Engine\Config\ConsoleVariables.ini` by adding this line to the end.

```
r.ForceDebugViewModes = 1
```

If this line is missing, UnrealCV commands of the packaged game will not work correctly.

2. Package your game project

UE4 makes it easy to release your project as a game binary. You can use the editor menu to package a game. Many related blog posts can be found online.

- Use UE4 Editor to package a game
- (advanced) use script to package a game

Another approach is used the packaging script contained in unrealcv repository.

3. Make a pull request

The last step is making a pull request to modify the *model zoo page* and add your content. We list some information that should be provided in the pull request. These information can help others better utilize the game binary. This is *an example*.

Binary name (required): An easy to remember name that people can refer to the binary you created

Author information (required): Author name and contact information

Binary description (required): What this virtual world is designed for? Generating dataset, reinforcement learning, etc.??

UnrealCV version (required): It can be a release version such as v0.3, a git short sha version, or a pointer to a commit of your fork. This information is to help others find which API is available and the corresponding documentation.

Download link (required): Please host binaries in your website, if you have difficulties finding a place to host content, we can help you find some free solutions. Please also include which platform (win,mac,linux) your binaries are built for.

Related paper or project page (optional): If this game binary is related to a research project, make a link to here.

Packaging binaries automatically

In the UnrealCV team, we use a set of packaging scripts to automate the packaging and testing. These scripts are hosted in <qiuwch/unrealcv-packaging>.

The architecture of UnrealCV

The focus of UnrealCV is to do IPC (Inter Process Communication) between a game and a computer vision algorithm. The communication can be summarized by Fig.1. A game created by Unreal Engine 4 will be extended by loading UnrealCV server as its module. When the game launches, UnrealCV will start a TCP server and wait for commands. Any program can use UnrealCV client code to send plain text UnrealCV command to control the scene or retrieve information. The design of UnrealCV makes it cross-platform and support multiple programming languages. The command system is designed in a modular way and can be easily extended.

Unrealcv consists two parts:

1. *unrealcv server* which is embedded into a video game.
2. *unrealcv client* uses commands to control the server.

The initialization procedure of UnrealCV. The module loaded and start a TCP server waiting for connection. Commands will be parsed by regexp.

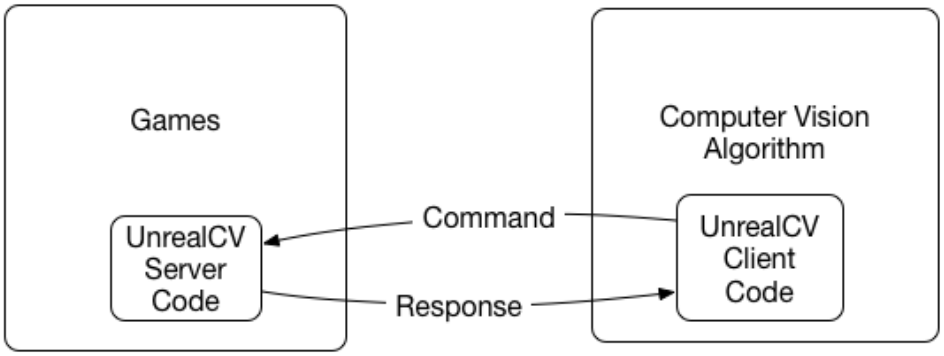


Fig. 2.3: Fig.1: Architecture of UnrealCV

Project Layout

```
client/             # Client code for Python and MATLAB
    examples/       # Examples showing how to use client code to do tasks
    matlab/         # MATLAB client
    python/         # Python client
    scripts/        # Scripts for tasks, such as building and packaging
Content/            # Plugin data
docs/               # Documentation of UnrealCV
Resources/          # Plugin resource
Source/             # Plugin C++ source code
test/               # Test code
UnrealCV.uplugin    # Descriptor file for an UE4 plugin
README.md
```

Command System

Unreal Engine 4 has some built-in commands to help game development. These commands can be typed into a built-in console. Using these commands, a developer can profile the game performance and view debug information. To invoke the built-in console of a game, type the ‘ key (the key above tab).

UnrealCV provides commands useful for computer vision researchers. What is more, these commands can be used by an external program. A built-in command can also be used using the special command `vrun`.

Command cheatsheet

Reply [this thread](https://groups.google.com/d/topic/unrealcv/EuJlibmTN3c/discussion) to tell us what missing functions are needed for your project. We will consider adding it in the future release.

1. Camera operation

See `Source/UnrealCV/Private/Commands/CameraHandler.h(.cpp)` for more details.

vget /camera/[id]/location (v0.2) Get camera location [x, y, z]

vget /camera/[id]/rotation (v0.2) Get camera rotation [pitch, yaw, roll]

vset /camera/[id]/location [x] [y] [z] (v0.2) Set camera location [x, y, z]

vset /camera/[id]/rotation [pitch] [yaw] [roll] (v0.2) Set camera rotation [pitch, yaw, roll]

vget /camera/[id]/[viewmode] (v0.2) Get [viewmode] from the [id] camera, for example: vget /camera/0/depth

vget /camera/[id]/[viewmode] [filename] (v0.2) Same as the above, with an extra parameter for filename

> **filename** Filename is where the file will be stored.
>
> **example** `vget /camera/0/lit lit.png`

vget /camera/[id]/[viewmode] [format] (v0.3.7) Support binary data format

> **format** If only file format is specified, the binary data will be returned through socket instead of being saved as a file.
>
> **example** `vget /camera/0/lit png`

vget /camera/[id]/object_mask (v0.2) The object mask is captured by first switching the viewmode to object_mask mode, then take a screenshot

vset /viewmode [viewmode] (v0.2) Set ViewMode to (lit, normal, depth, object_mask)

vget /viewmode (v0.2) Get current ViewMode

2. Object interaction

See *Source/UnrealCV/Private/Commands/ObjectHandler.h(.cpp)* for more details

vget /objects (v0.2) Get the name of all objects

vget /object/[obj_name]/color (v0.2) Get the labeling color of an object (used in object instance mask)

vset /object/[obj_name]/color [r] [g] [b] (v0.2) Set the labeling color of an object

3. Plugin commands

See *Source/UnrealCV/Private/Commands/PluginHandler.h(.cpp)* for more details.

vget /unrealcv/status (v0.2) Get the status of UnrealCV plugin

vget /unrealcv/help (v0.2) List all available commands and their help message

4. Action commands

See `Source/UnrealCV/Private/Commands/ActionHandler.h(.cpp)`

vset /action/keyboard [key_name] [delta] (v0.3.6) Valid key_name can be found in here

Run UE4 built-in commands

vrun [cmd] (v0.3) This is a special command used to execute Unreal Engine built-in commands. UE4 provides some built-in commands for development and debug. They are not very well documented, but very useful.

A few examples are:

- `stat FPS` - show current frame rate
- `shot` - take a screenshot

- `show Material` - toggle the display of Material

These commands can be executed in the UE4 console. If you want to use them in UnrealCV, you can prefix these commands with *vrun stat FPS*.

Run Blueprint commands

vexec [cmd] TODO

Model Zoo

We provide compiled virtual worlds to play with. All the digital contents belong to the original author. If the program crash for any reason, you can report an issue.

Run the downloaded binary

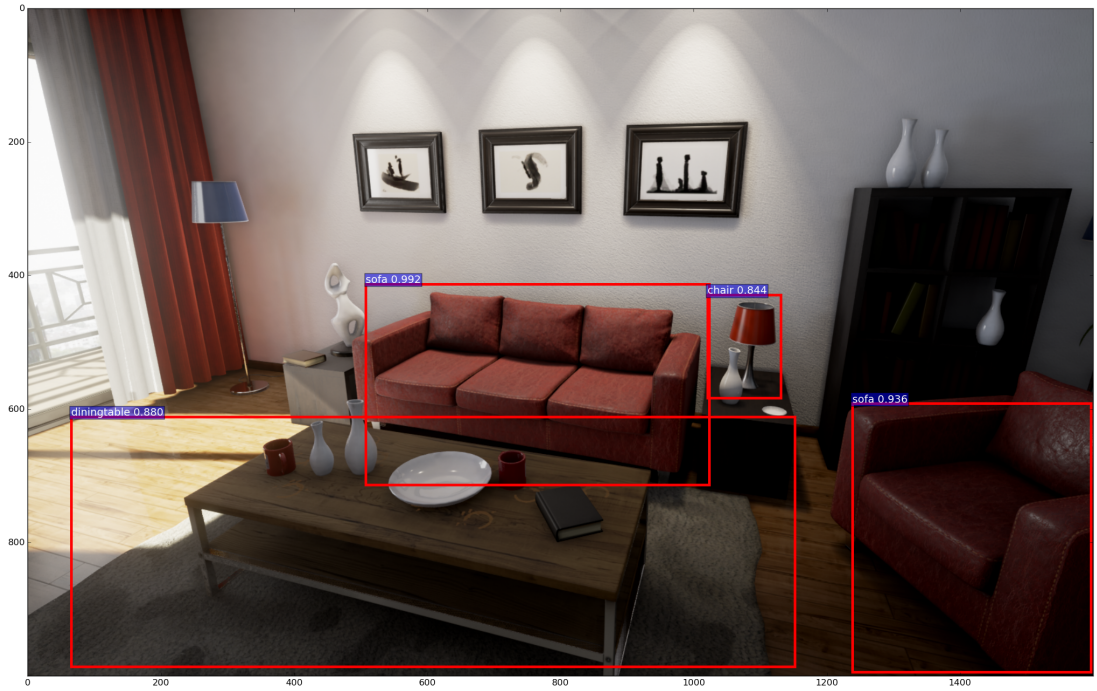
- In Mac: Run `[ProjectName].app`
- In Linux: Run `./[ProjectName]/Binaries/Linux/[ProjectName]`
- In Windows: Run `[ProjectName]/[ProjectName].exe`

Read *Package a new game*, if you are interested in sumbitting a binary to the model zoo.

RealisticRendering

Source https://docs.unrealengine.com/latest/INT/Resources/Showcases/RealisticRendering/

Preview



Description Realistic Rendering is a demo created by Epic Games to show the rendering power of Unreal Engine 4. Here we provide an extended version of this demo with UnrealCV embedded.

Download Windows, Linux (tested in Ubuntu 14.04), Mac

- *Docker*

```
`nvidia-docker run -it --rm --env="DISPLAY=:0.0" --volume="/tmp/.X11-unix:/tmp/.X11-
↪unix:rw" -p 9000:9000 -v /home/qiuwch/workspace/unrealcv-develop-branch/test/
↪output:/home/unrealcv/LinuxNoEditor/RealisticRendering/Binaries/Linux/output qiuwch/
↪rr /home/unrealcv/LinuxNoEditor/RealisticRendering/Binaries/Linux/
↪RealisticRendering`
```

ArchinteriorsVol2Scene1

Source https://www.unrealengine.com/marketplace/archinteriors-vol-2-scene-01

Preview

Description ArchinteriorsVol2Scene1 is a scene of a 2 floors apartment.

Download Windows, Linux

ArchinteriorsVol2Scene2

Source https://www.unrealengine.com/marketplace/archinteriors-vol-2-scene-02

Preview

Description ArchinteriorsVol2Scene2 is a scene of a house with 1 bedroom and 1 bathroom.

Download Windows, Linux

ArchinteriorsVol2Scene3

Source https://www.unrealengine.com/marketplace/archinteriors-vol-2-scene-03

Preview

Description ArchinteriorsVol2Scene3 is a scene of an office.

Download Windows, Linux

UrbanCity

Source https://www.unrealengine.com/marketplace/urban-city

Preview

Description UrbanCity is a scene of a block of street.

Download Windows, Linux

First Person Shooter

Trouble Shooting

We tried our best to make the software stable and easy to use, but accidents sometimes happen. Here are a list of issues that you might find. Use the search function `ctrl+f` to search your error message. If you can not find useful information here, post a new issue.

- The binary can not run

python3 support. See issue #49, Thanks to @jskinn for the idea!

Supported Unreal Engine Version

UE4 (4.12, 4.13, 4.14)

Client

Python MATLAB

4.14

Windows Linux Mac

Verified projects

Unreal Engine projects are of dramatic different scales and complexity. It can be as simple as just a single room, or be a large city or outdoor scene. UnrealCV is far from perfect and it has compatible issues with some Unreal projects. Here are a few projects we are currently using and have verified that UnrealCV can work well with. If you want us to test a map (project), please let us know.

Here are a list of Unreal projects that we tried and verified.

- Playground, tested by @qiuwch.
- Realistic Rendering, tested by @qiuwch.
- CityScene, tested by @qiuwch, @edz-o
- SunTemple, tested by @edz-o

Known issues

- The binary can not run

  For example an error like this.

```
[2017.05.25-04.14.33:476][  0]LogLinux:Error: appError called: Assertion failed:␣
↪Assertion failed:  [File:/UE4/Engine/Source/Runtime/OpenGLDrv/Private/Linux/
↪OpenGLLinux.cpp] [Line: 842]
```

Unable to dynamically load libGL: Could not retrieve EGL extension function eglQueryDevicesEXT

It is very likely an issue with the OpenGL of the system.

`sudo apt-get install mesa-utils` and run `glxgears`. Make sure you can see a window with gears running in it.

- The screen resolution is not what I want
  - In editor, change *Editor preference -> Level Editor -> Play*
  - In the game mode, use console command *setres 640x480*
  - Change the config file shown in *vget /unrealcv/status*
- The speed of UnrealCV
- The OpenEXR requirement
- The Unreal Engine config file not changed
- The image and ground truth not aligned

Build

Mac

Native error= Cannot find the specified file

https://answers.unrealengine.com/questions/574562/cannot-package-a-plugin-in-415mac.html

Invalid SDK MacOSX.sdk, not found in /Library/Developer/CommandLineTools/Platforms/MacOSX.platform/Developer/SDKs

https://answers.unrealengine.com/questions/316117/missing-project-modules-1.html https://github.com/nodejs/node-gyp/issues/569#issuecomment-255589932

Issues and workarounds

Issues and workaround can be found in [issue tracker](https://github.com/unrealcv/unrealcv/issues). Please use the search function before posting your issue, your problem might have already been answered. Also you are welcome to chat with us in our [gitter channel](https://gitter.im/unrealcv/unrealcv).

If the plugin crashed the editor, please send us your crash log to help us figure out what is going wrong. The crash log can be found in *Saved/CrashReport*. If you can not find the crash report, you can also send us the core dump file.

- Can not connect to the binary

Use `vget /unrealcv/status` to make sure the server is listening and no client is connected.

Subscribe to an issue if you want to get future notification.

API

The C++ code of the plugin is documented with doxygen. You can generate the API document with `cd docs/doxygen; doxygen`. An online version is hosted in here.

Python

The python client UnrealCV === Provides functions to interact with games built using Unreal Engine.

```
>>> import unrealcv
>>> (HOST, PORT) = ('localhost', 9000)
>>> client = unrealcv.Client((HOST, PORT))
```

class `unrealcv.`**`Client`**(*endpoint*, *message_handler=None*)

Client can be used to send request to a game and get response Currently only one client is allowed at a time More clients will be rejected

`request`(*message*, *timeout=5*)

Send a request to server and wait util get a response from server or timeout.

Parameters — cmd : string, command to control the game More info can be seen from http://unrealcv.github.io/commands.html

Returns — response: plain text message from server

Examples — >>> client = Client('localhost', 9000) >>> client.connect() >>> response = client.request('vget /camera/0/view')

class `unrealcv.`**`BaseClient`**(*endpoint*, *raw_message_handler*)

BaseClient send message out and receiving message in a seperate thread. After calling the *send* function, only True or False will be returned to indicate whether the operation was successful. If you are trying to send a request and get a response, consider using *Client* instead. This class adds message framing on top of TCP

`connect`(*timeout=1*)

Try to connect to server, return whether connection successful

`send`(*message*)

Send message out, return whether the message was successfully sent

MATLAB

The matlab client

Core

FUE4CVServer uses *FCommandDispatcher* to execute commands. All CommandHandlers are registered.

class `FUE4CVServer`

UnrealCV server to interact with external programs.

Inherits from FTickableGameObject

Public Functions

void **`SendClientMessage`**(FString *Message*)

Send a string message to connected clients.

APawn ***`GetPawn`**()

Return the Pawn of this game.

Only available during game play.

virtual void **Tick**(float *DeltaTime*)
Implement ticking function of UE4CVServer itself.

void **RegisterCommandHandlers**()
For UnrealCV server, when a game start:

1.Start a TCPserver.

2.Create a command dispatcher

3.Add command handler to command dispatcher, CameraHandler should be able to access camera

4.Bind command dispatcher to TCPserver

5.Bind command dispatcher to UE4 console

When a new pawn is created.

1.Update this pawn with GTCaptureComponent

bool **InitWorld**()
Make sure UE4CVServer correctly initialized itself in the GameWorld.

Make sure the UE4CVServer is correctly configured.

UWorld ***GetGameWorld**()
Return the GameWorld of the editor or of the game.

Public Members

FCommandDispatcher ***CommandDispatcher**
The CommandDispatcher to handle a pending request.

FServerConfig **Config**
The config of UE4CVServer.

UNetworkManager ***NetworkManager**
The underlying class to handle network connection, ip and port are configured here.

Public Static Functions

FUE4CVServer &**Get**()
Get the singleton.

Warning: doxygenclass: Cannot find class "UNetworkManager" in doxygen xml output for project "unrealcv" from directory: ./doxygen/xml/

class FExecStatus
Present the return value of a command.

If the FExecStatusType is pending, check the promise value.

class FPromise
Return by async task, used to check status to see whether the task is finished.

class FCommandDispatcher
Engine to execute commands.

class FAsyncWatcher
Inherits from FRunnable

The execution result of a command

Basic

class AUE4CVPawn
UE4CVPawn can move freely in the 3D space.

Inherits from ADefaultPawn

class AUE4CVCharacter
UE4CVCharacter acts like a walking human.

Inherits from ACharacter

class AUE4CVGameMode
Inherits from AGameMode

class FServerConfig

API

class FCommandDispatcher
Engine to execute commands.

class FCommandHandler
Subclassed by *FActionCommandHandler*, *FAliasCommandHandler*, *FCameraCommandHandler*, *FObjectCommandHandler*, *FPluginCommandHandler*

class FActionCommandHandler
Inherits from *FCommandHandler*

class FAliasCommandHandler
Inherits from *FCommandHandler*

class FCameraCommandHandler
Inherits from *FCommandHandler*

class FObjectCommandHandler
Inherits from *FCommandHandler*

class FPluginCommandHandler
Inherits from *FCommandHandler*

CHANGELOG

Development branch

- head : Fix a bug that prevents object mask generation
- **v0.3.8 :**
  - Integrate cnpy into unrealcv
  - Add `vget /camera/depth npy`, which can return tensor as a numpy binary.

- **v0.3.7 :**
  - Add `vget /camera/lit png` to retrieve binary data without saving it.
- **v0.3.6 :**
  - Change docs from markdown to reStructuredText
  - Add docker to automate tests
  - Add `vset /action/keyboard [key_name] [delta]`
- v0.3.5 : Add vexec to support the invocation of blueprint functions, Add `GetWorld()` in `FCommandHandler`.
- v0.3.4 : Delay the object mask painting from initialization code
- v0.3.3 : Add `vget /scene/name`
- **v0.3.2 :**
  - Add `vget /unrealcv/version`
  - Add `vset /action/eyes_distance`
  - Fix `vget /camera/[id]/location` to support multiple cameras
  - Update test code
- v0.3.1 : Fix GWorld issue

v0.3.0 - Stability improvement

- Add support for Unreal 4.13, 4.14
- Stability improvement, fix crash caused by the usage of GWorld
- Fix some incorrect ground truth, blueprint actor not correctly displayed.
- Add playground project
- Add docs to docs.unrealcv.org
- Add API documentation created by doxygen
- Fix an issue that prevents the packaging of games.
- Add vrun command to exec UE4 built-in command

API update:

- `vrun [built-in command]`
- `vset /camera/[id]/moveto [x] [y] [z]` # With collision enabled

v0.2.0 - First public release

Features

- Add communication to UE4Editor and a compiled game
- Add Python and MATLAB client to communicate with UnrealCV server
- Add ground truth extraction, include: depth, object-mask, surface normal
- Add support for Linux, Win and Mac

Initial API, see more details in *the command list*

- `vget /objects`
- `vget /object/[obj_name]/color`
- `vset /object/[obj_name]/color [r] [g] [b]`
- `vget /camera/[id]/location`
- `vget /camera/[id]/rotation`
- `vset /camera/[id]/location [x] [y] [z]`
- `vset /camera/[id]/rotation [pitch] [yaw] [roll]`
- `vget /camera/[id]/[viewmode]`
- `vget /camera/[id]/[viewmode] [filename]`
- `vset /viewmode [viewmode]`
- `vget /viewmode`
- `vget /unrealcv/status`
- `vget /unrealcv/help`

The upcoming release will follow the concept of Semantic Versioning

Contribute

We will be grateful if you help us mature UnrealCV.

Bug Reporting

If you find a bug in UnrealCV, you can search for similar issues in the Github first. If you still cannot solve the problem, please open an issue and describe the bug you encountered.

Requesting A New Feature

UnrealCV is still developing and we really want to make it more helpful to computer vision researchers. Please tell us your expectation about what can be done with UnrealCV. You are welcomed to open an issue and tell us what you want.

Pull Request

If you make improvements to UnrealCV,like fixing a bug or adding a new feature, please submit a pull request. Please make sure that you follow the syntax of UnrealCV. You will be greatly appreciated if you can fully describe the changes you have made.

Contact

UnrealCV is an open source project created by Weichao Qiu. It is hosted in Github and everyone is welcomed to contribute.

- If you want a new feature for your research or found any bug, please submit an issue in our issue tracker.
- Want to provide a compiled binary for the community? Please read the guidance in model zoo.
- Want to learn how to create a game using Unreal Engine 4? Please check their excellent document.

Team Members

UnrealCV is developed and maintained by a group of students working on computer vision (sort alphabetically).

- Siyuan Qiao: Stochastic virtual scene generation
- *Tae Soo Kim*: Deep learning for transferable concepts between the real and the virtual environment
- Weichao Qiu: Constructing virtual worlds to train and diagnose computer vision algorithms
- *Yi Zhang*: Algorithm diagnosis with synthetic data
- *Fangwei Zhong*

UE4 Resources

If you are interested in research projects, please see the synthetic-computer-vision repository.

UE4 is a good choice for research for various reasons:

1. Rich 3D resources
2. Open source
3. Realistic rendering
4. Cross-platform

Here we collect some resources to help you use UE4 and UnrealCV in your research.

3D Resources (models, scenes)

- [Unreal Marketplace]()
- [UE4Arch]()
- [Evermotion]()
- [ShapeNet]()
- [TurboSquid]()

C++ Development

- [UE4 API](https://docs.unrealengine.com/latest/INT/API/)
- [UE4 Answerhub](https://answers.unrealengine.com/)
- [Github repository](https://github.com/EpicGames/UnrealEngine) (This is a private repository, you need to register [here](https://www.unrealengine.com/ue4-on-github) before you can access this repository)

Python Module Index

u

Index