UnrealCV

Release 0.3.10

Nov 22, 2017

Contents

CHAPTER 1

UnrealCV

UnrealCV 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.

Please read Tutorial: Getting Started to learn using UnrealCV.

Images generated from the technical demo RealisticRendering

1.1 Citation

If you found this project useful, please consider citing our paper

```
@article{qiu2017unrealcv,
  Author = {Weichao Qiu, Fangwei Zhong, Yi Zhang, Siyuan Qiao,Zihao Xiao, Tae Soo Kim,
↪ Yizhou Wang, Alan Yuille},
  Journal = {ACM Multimedia Open Source Software Competition},
  Title = {UnrealCV: Virtual Worlds for Computer Vision},
  Year = {2017}
}
```

1.2 Contact

If you have any suggestion or interested in using UnrealCV, please contact us.

CHAPTER 2

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.

2.1 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.

Fig. 2.1: Initial screen of the game

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).

2.2 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.

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

2.2.1 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.

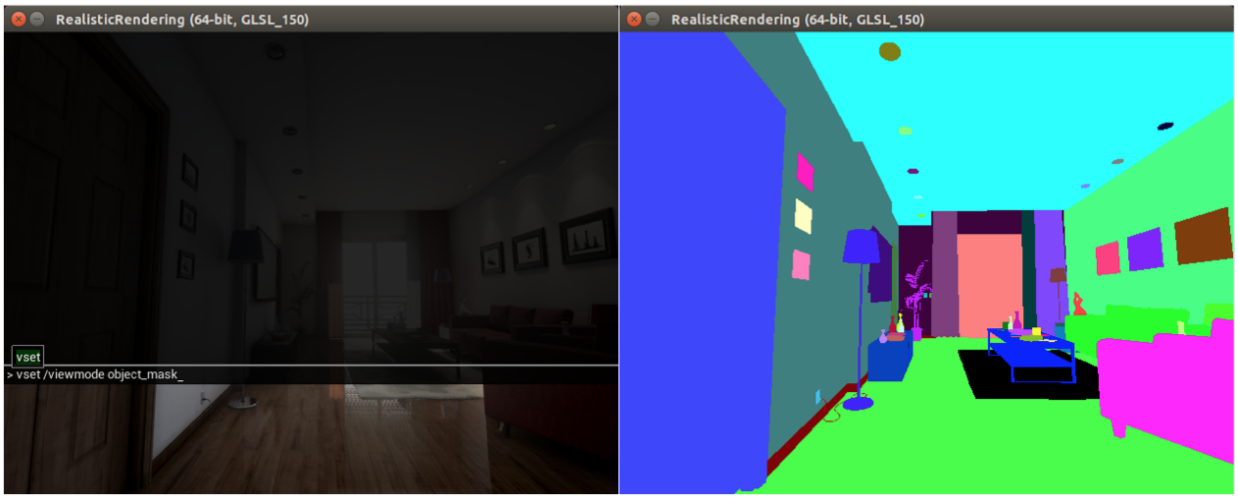


Fig. 2.2: Use console to try UnrealCV commands

2.3 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.

2.3.1 Install UnrealCV python library

```
pip install unrealcv
```

2.3.2 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 examples/10lines.py.

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

2.3.3 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.

2.3.4 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*

2.3.5 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.

```
In [1]: %matplotlib inline
```

CHAPTER 3

Generate Images

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

```
In [2]: import time; print(time.strftime("The last update of this file: %Y-%m-%d %H:%M:%S", time.gmt
```

```
The last update of this file: 2017-10-21 21:26:41
```

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

```
In [3]: 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))
```

3.1 Connect to the game

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

```
In [4]: from unrealcv import client
        client.connect()
```

```
        if not client.isconnected():
            print('UnrealCV server is not running. Run the game downloaded from http://unrealcv.githu
            sys.exit(-1)
```

```
INFO:__init__:211:Got connection confirm: 'connected to RealisticRendering'
```

Make sure the connection works well

```
In [5]: res = client.request('vget /unrealcv/status')
        # The image resolution and port is configured in the config file.
        print(res)
```

```
Is Listening
Client Connected
9000
Configuration
Config file: /Users/qiuwch/unrealcv/UE4Binaries/RealisticRendering/MacNoEditor/RealisticRendering.app
Port: 9000
Width: 640
Height: 480
FOV: 90.000000
EnableInput: true
EnableRightEye: false
```

3.2 Load a camera trajectory

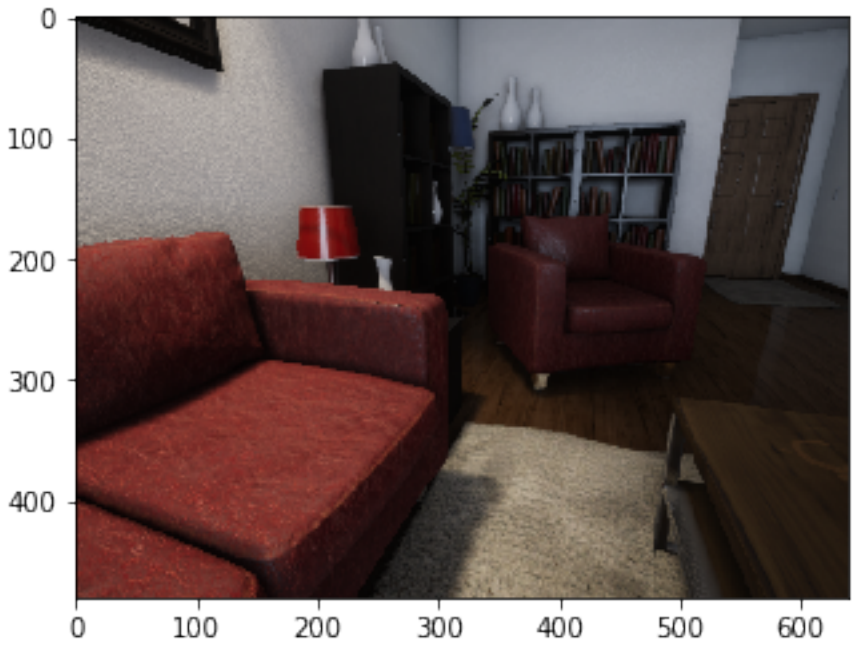
```
In [7]: traj_file = './camera_traj.json' # Relative to this python script
        import json; camera_trajectory = json.load(open(traj_file))
        # We will show how to record a camera trajectory in another tutorial
```

3.3 Render an image

```
In [8]: 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)
```

```
The image is saved to /Users/qiuwch/unrealcv/UE4Binaries/RealisticRendering/MacNoEditor/RealisticRend
(480, 640, 4)
```

```
Out[8]: <matplotlib.image.AxesImage at 0x108dd8310>
```

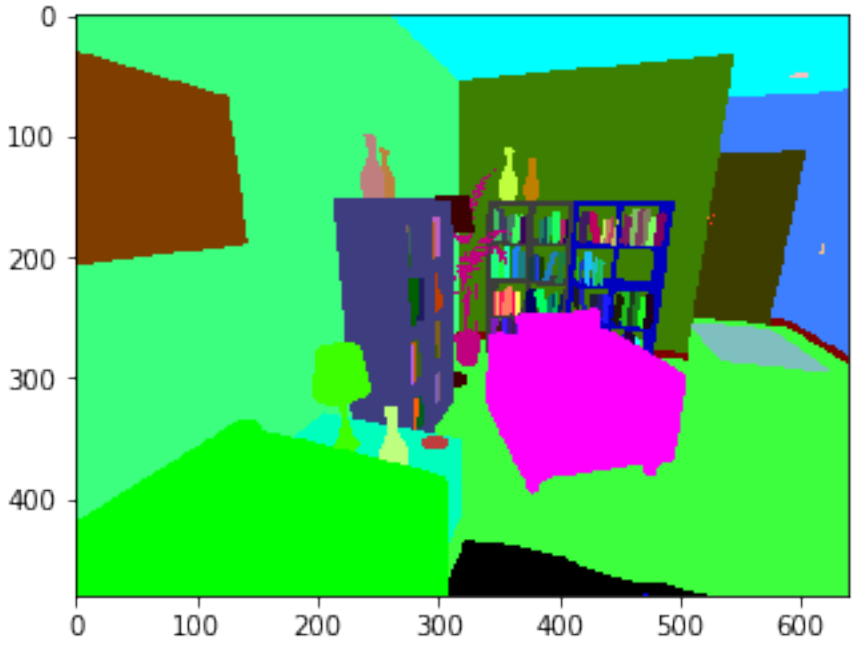


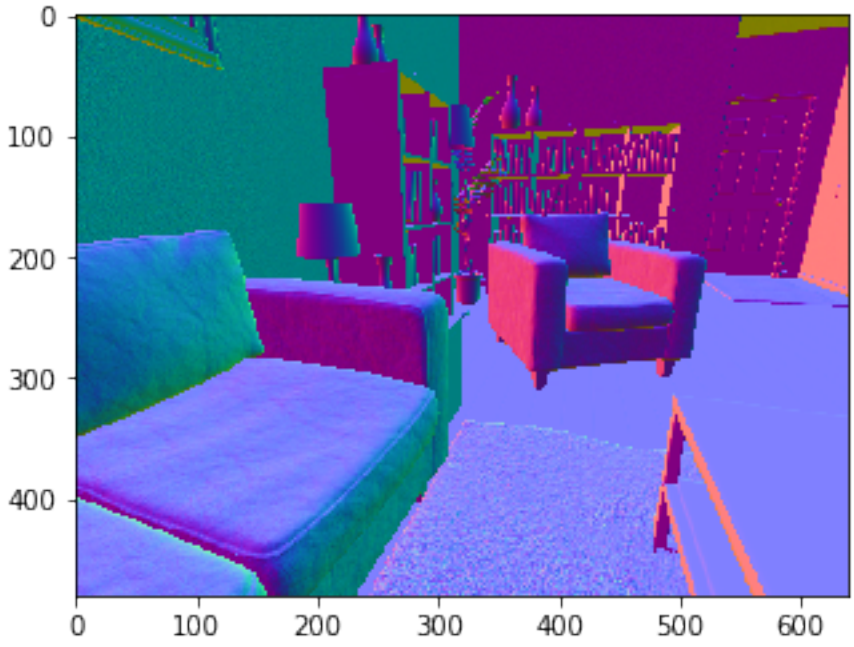
3.4 Ground truth generation

Generate ground truth from this virtual scene

```
In [9]: 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)
```

```
Out[9]: <matplotlib.image.AxesImage at 0x1097db8d0>
```

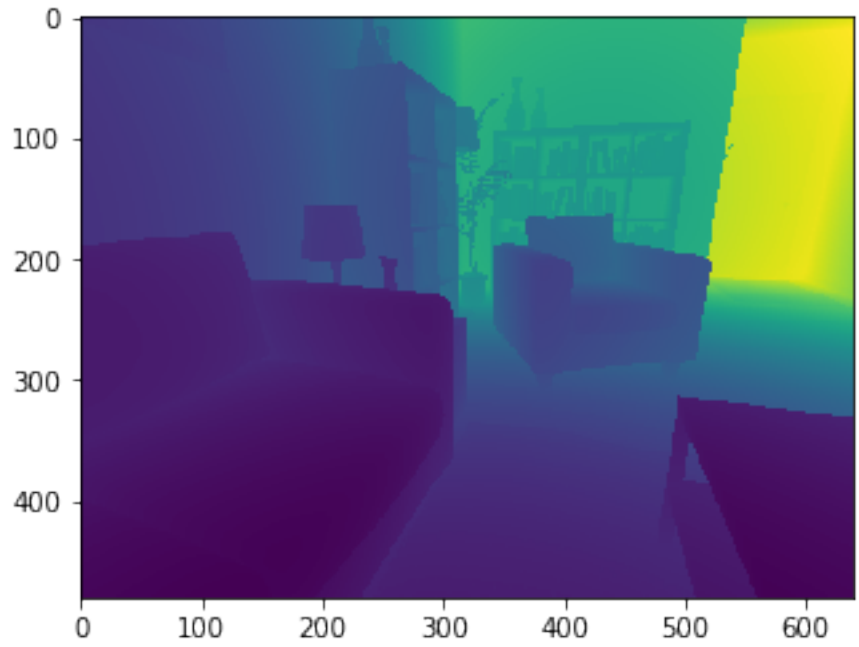




Depth is retrieved as a numpy array For UnrealCV < v0.3.8, the depth is saved as an exr file, but this has two issues. 1. Exr is not well supported in Linux 2. It depends on OpenCV to read exr file, which is hard to install

```
In [10]: res = client.request('vget /camera/0/depth npy')
         depth = read_npy(res)
         plt.imshow(depth)
```

```
Out[10]: <matplotlib.image.AxesImage at 0x10905d090>
```



3.5 Get object information

List all the objects of this virtual scene

```
In [11]: scene_objects = client.request('vget /objects').split(' ')
         print('Number of 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

         id2color = {} # Map from object id to the labeling color
         for obj_id in scene_objects:
             color = Color(client.request('vget /object/%s/color' % obj_id))
             id2color[obj_id] = color
             # print('%s : %s' % (obj_id, str(color)))
```

```
Number of objects in this scene: 296
```

Parse the segmentation mask

```
In [12]: def match_color(object_mask, target_color, tolerance=3):
             match_region = np.ones(object_mask.shape[0:2], dtype=bool)
             for c in range(3): # r,g,b
                 min_val = target_color[c] - tolerance
                 max_val = target_color[c] + tolerance
```

```
                channel_region = (object_mask[:,:,c] >= min_val) & (object_mask[:,:,c] <= max_val)
                match_region &= channel_region

            if match_region.sum() != 0:
                return match_region
            else:
                return None

        id2mask = {}
        for obj_id in scene_objects:
            color = id2color[obj_id]
            mask = match_color(object_mask, [color.R, color.G, color.B], tolerance = 3)
            if mask is not None:
                id2mask[obj_id] = mask
        # This may take a while
        # TODO: Need to find a faster implementation for this
```

3.6 Print statistics of this virtual scene and this image

Load information of this scene

```
In [14]: with open('object_category.json') as f:
             id2category = json.load(f)
         categories = set(id2category.values())
         # Show statistics of this frame
         image_objects = id2mask.keys()
         print('Number of objects in this image:', len(image_objects))
         print('%20s : %s' % ('Category name', 'Object name'))
         for category in categories:
             objects = [v for v in image_objects if id2category.get(v) == category]
             if len(objects) > 6: # Trim the list if too long
                 objects[6:] = ['...']
             if len(objects) != 0:
                 print('%20s : %s' % (category, objects))
```

```
Number of objects in this image: 125
       Category name : Object name
             Shelving : [u'SM_Shelving_7', u'SM_Shelving_6', u'SM_Shelving_9', u'SM_Shelving_8']
                 Bowl : [u'SM_Bowl_29']
                Couch : [u'SM_Couch_1seat_5', u'Couch_13']
                 Book : [u'BookLP_139', u'BookLP_153', u'BookLP_134', u'BookLP_135', u'BookLP_136', u
             DeskLamp : [u'SM_DeskLamp_5']
     CoatHookBacking : [u'CoatHookBacking_7']
               Plant : [u'SM_Plant_8']
                Door : [u'SM_Door_39']
           Trim_Floor : [u'S_Trim_Floor_10']
                Vase : [u'SM_Vase_22', u'SM_Vase_21', u'SM_Vase_20', u'SM_Vase_18', u'SM_Vase_16', u
              Carpet : [u'Carpet_5', u'Carpet_7']
                Room : [u'SM_Room_7']
            FloorLamp : [u'SM_FloorLamp_7']
              Switch : [u'Switch_7']
               Frame : [u'SM_Frame_39']
            WallPiece : [u'WallPiece6_32', u'WallPiece2_24', u'WallPiece1_22', u'WallPiece3_26']
             CoatHook : [u'CoatHook_17', u'CoatHook_16']
   RoundCeilingLight : [u'SM_RoundCeilingLight_4']
         CoffeeTable : [u'SM_CoffeeTable_14']
```

Show the annotation color of some objects

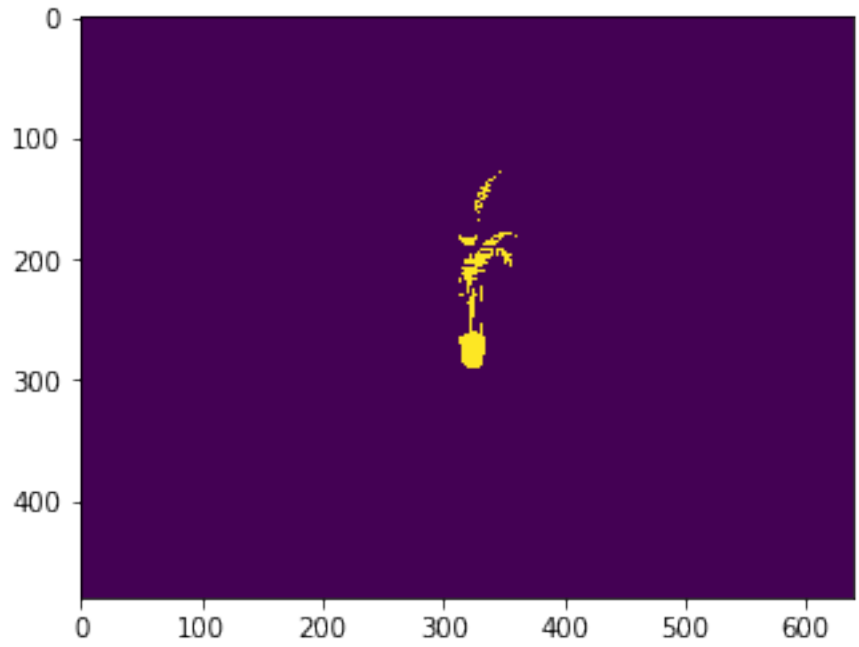
```
In [15]: ids = ['SM_Couch_1seat_5', 'SM_Vase_17', 'SM_Shelving_6', 'SM_Plant_8']
         # for obj_id in ids:
         obj_id = ids[0]
         color = id2color[obj_id]
         print('%s : %s' % (obj_id, str(color)))
         # color_block = np.zeros((100,100, 3)) + np.array([color.R, color.G, color.B]) / 255.0
         # plt.figure(); plt.imshow(color_block); plt.title(obj_id)
```

```
SM_Couch_1seat_5 : (R=255,G=0,B=255,A=255)
```

Plot only one object

```
In [16]: mask = id2mask['SM_Plant_8']
         plt.figure(); plt.imshow(mask)
```

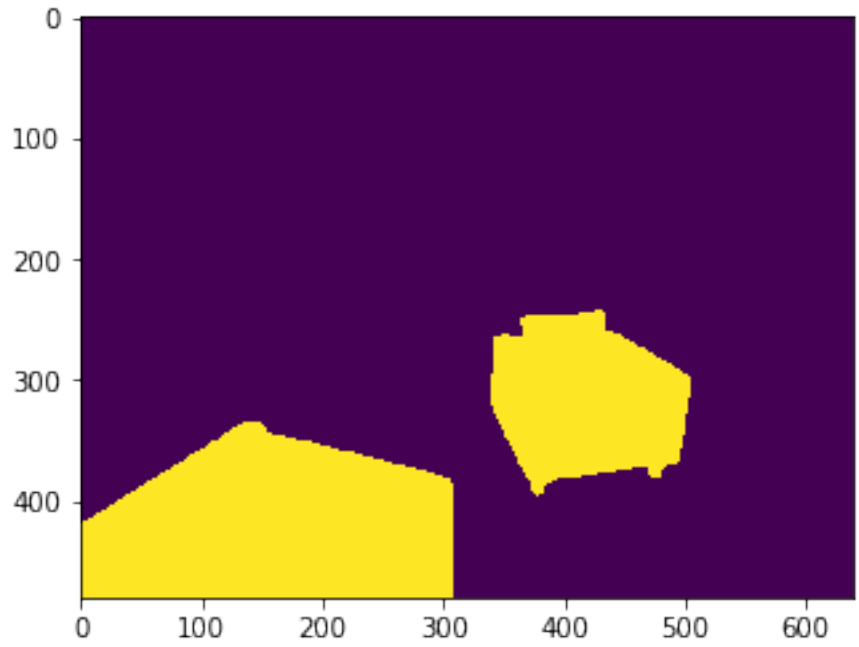
```
Out[16]: <matplotlib.image.AxesImage at 0x10c81eb10>
```



Show all sofas in this image

```
In [17]: couch_instance = [v for v in image_objects if id2category.get(v) == 'Couch']
         mask = sum(id2mask[v] for v in couch_instance)
         plt.figure(); plt.imshow(mask)
```

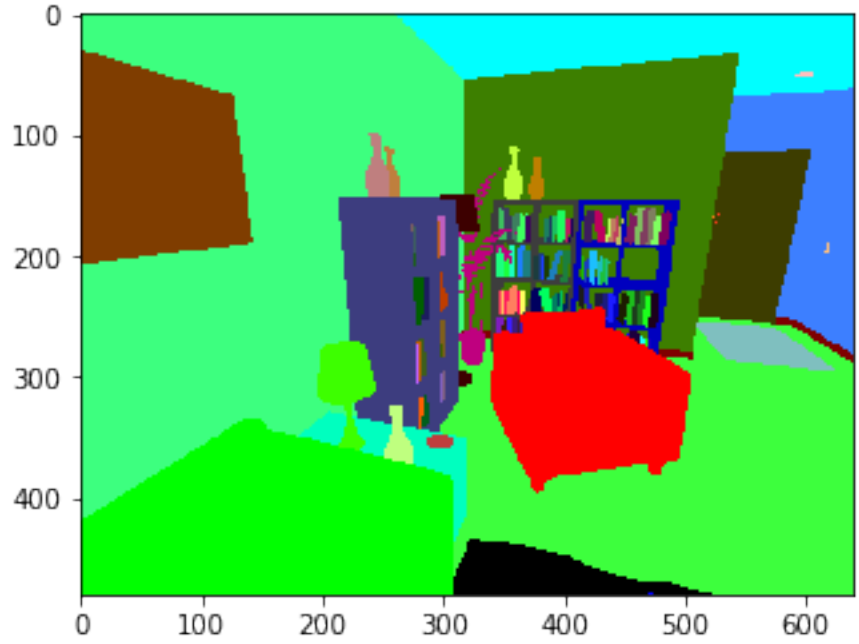
```
Out[17]: <matplotlib.image.AxesImage at 0x10c9eb710>
```



Change the annotation color, fixed in v0.3.9 You can use this to make objects you don't care the same color

```
In [18]: client.request('vset /object/SM_Couch_1seat_5/color 255 0 0') # Change to pure red
         client.request('vget /object/SM_Couch_1seat_5/color')
         res = client.request('vget /camera/0/object_mask png')
         object_mask = read_png(res)
         plt.imshow(object_mask)
```

```
Out[18]: <matplotlib.image.AxesImage at 0x10dc767d0>
```



3.7 Clean up resources

```
In [19]: client.disconnect()
```

CHAPTER 4

More Examples

More examples can be found in the examples folder of the repo.

- List of all unrealcv commands - examples/commands_demo.ipynb
- Stereo image pair generation - examples/stereo_pair_generation.ipynb
- faster_rcnn

CHAPTER 5

Install UnrealCV Plugin

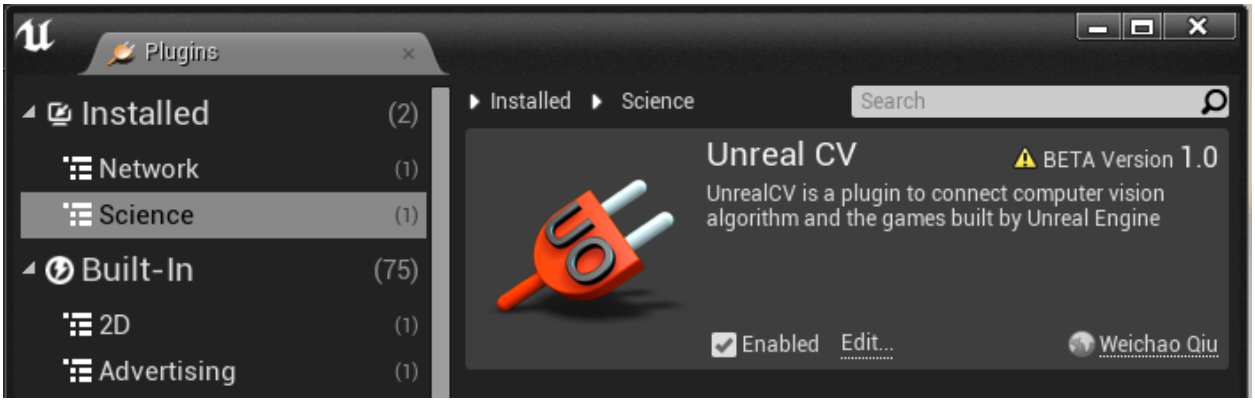
This page briefly describes how to install UnrealCV as a UE4 plugin. Make sure you read *getting started* before trying to use the plugin.

5.1 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.

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



5.2 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.

5.3 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

```
pip install -U unrealcv
    # Install the latest version of unrealcv, the build.py depends on unrealcv.
↪automation module
python build.py
# This script will search common Unreal Engine folders of Windows and Mac
# If this script fails to find UE4 installation path, you can also manually specify␣
↪the engine path
python build.py --UE4 {UE4}
# For example
# python build.py --UE4 "/Users/Shared/Epic Games/UE_4.16"
```

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 Plugins folder.

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`.

5.4 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.

Previous versions of unrealcv depends on using OpenEXR module to generate depth, this requirement has been removed.

CHAPTER 6

Basic usage in a new project

ue4: 4.14, **unrealcv**: v0.3.9, **level**: basic, **updated**: 06/27/2017

This tutorial assumes the basic knowledge of using a command line. If you are a windows user and do not have much knowledge of using command line and find it difficult to understand, please let us know.

6.1 1. Install the plugin (build from source code in this tutorial)

In this tutorial the project and plugin folders are refered as the console varialbe `plugin_folder` and `project_folder`

```
export plugin_folder=$HOME/workspace/unrealcv

# Clone the UnrealCV repository
git clone https://github.com/unrealcv/unrealcv ${plugin_folder}

# Switch to version `v0.3.10`
cd {plugin_folder}
git checkout v0.3.10

# Build the plugin binary
export UE4="/Users/Shared/Epic Games/UE_4.14/"
python build.py --UE4 ${UE4}
```

More details about the plugin installation can be found in *Install UnrealCV Plugin*.

6.2 2. Create a new project and copy the plugin to the project folder

Creat a 'Blueprint - First Person' project.

Refer the project folder with a *project_folder* variable

```
export project_folder="$HOME/Documents/Unreal Projects/MyProject"

# Copy the compiled plugin to the project folder to install it.
cp -r "${plugin_folder}"/Plugins/ "${project_folder}"/Plugins/
```

Restart the UE4 project and make sure the plugin is successfully loaded

6.3 3. Open the Unreal project and make sure the plugin is installed

6.4 4. Try the command in UE4Editor

6.5 5. Try the command is the python console

CHAPTER 7

Usage in the Editor

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

7.1 Run UnrealCV Command

While playing the level in the editor, press ‘ to open the built-in console and run UnrealCV like in the game binary.

7.2 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,

7.3 Edit Object Color

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

The results are as follows,

7.4 Use UnrealCV commands in the editor

You can use UnrealCV commands the same way as in a standalone binary.

7.5 Reference

This page is contributed by the UnrealStereo project and the tool and images will be released soon.

CHAPTER 8

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.

8.1 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.

8.2 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.

CHAPTER 9

Package a game binary

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*

9.1 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

9.1.1 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.

9.1.2 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

- Use script to package a game

```
python build.py --UE4 {UE4 instal path} {uproject path}
```

For example, `python build.py --UE4 "C:\Program Files\Epic Games\UE_4.16" C:\qiuwch\workspace\uprojects\UE4ArchinteriorsVol2Scene1\ArchinteriorsVol2Scene1.uproject`

9.1.3 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.

9.2 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>.

CHAPTER 10

The architecture of UnrealCV

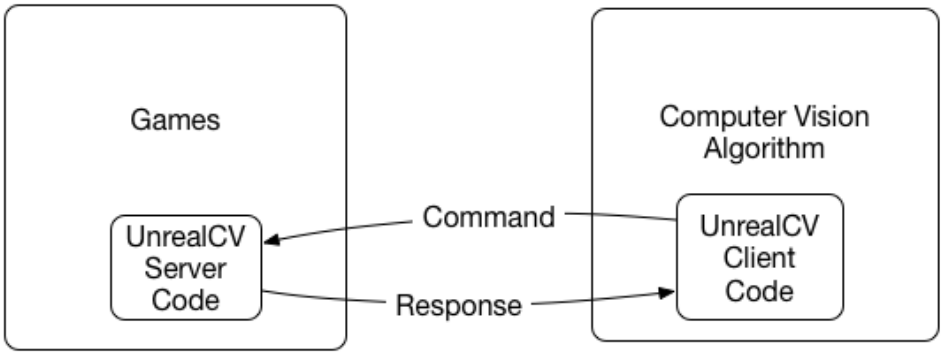


Fig. 10.1: Fig.1: Architecture of UnrealCV

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.

10.1 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
```

CHAPTER 11

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`.

11.1 Command cheatsheet

See this ipython notebook to see an imcomplete demo of available commands.

11.2 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

vget /camera/[id]/pose (v0.3.10) Get camera location [x, y, z] and rotation [pitch, yaw, roll]

vset /camera/[id]/pose [x] [y] [z] [pitch] [yaw] [roll] (v0.3.10) Teleport camera to location [x, y, z] and rotation [pitch, yaw, roll]

vget /camera/[uint]/horizontal_fieldofview (v0.3.10) Get camera horizontal field of view

vset /camera/[uint]/horizontal_fieldofview [FOV] (v0.3.10) Set camera horizontal field of view

vget /camera/[uint]/vis_depth npy (v0.3.10)

vget /camera/[uint]/plane_depth npy (v0.3.10)

11.3 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

vset /object/[str]/show (v0.3.10) Show object

vset /object/[str]/hide (v0.3.10) Hide object

11.4 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

11.5 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

vset /action/game/pause (v0.3.10) Pause the game

vset /action/game/level [level_name] (v0.3.10) Open a new level

vset /action/input/enable (v0.3.10) Enable input

vset /action/input/disable (v0.3.10) Disable input

vset /action/eyes_distance [eye_distance] (v0.3.10) Set the eye distance between left eye and right eye (camera 1). This command might be marked as deprecated when we finish multiple camera support.

11.6 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*.

11.7 Run Blueprint commands

vexec [cmd] TODO

CHAPTER 12

Model Zoo

We provide compiled virtual worlds to play with. All the digital contents belong to the original author. If you want to use UnrealCV plugin in the editor, you can find UE4 projects in *UE4 Resources*

Hint: Model Zoo is still experimental. If you are a beginner, we highly recommend you start with RealisitcRendering to avoid unknown bugs. If you found any bugs, you can report an issue. We will fix them as soon as possible.

Run the downloaded binary

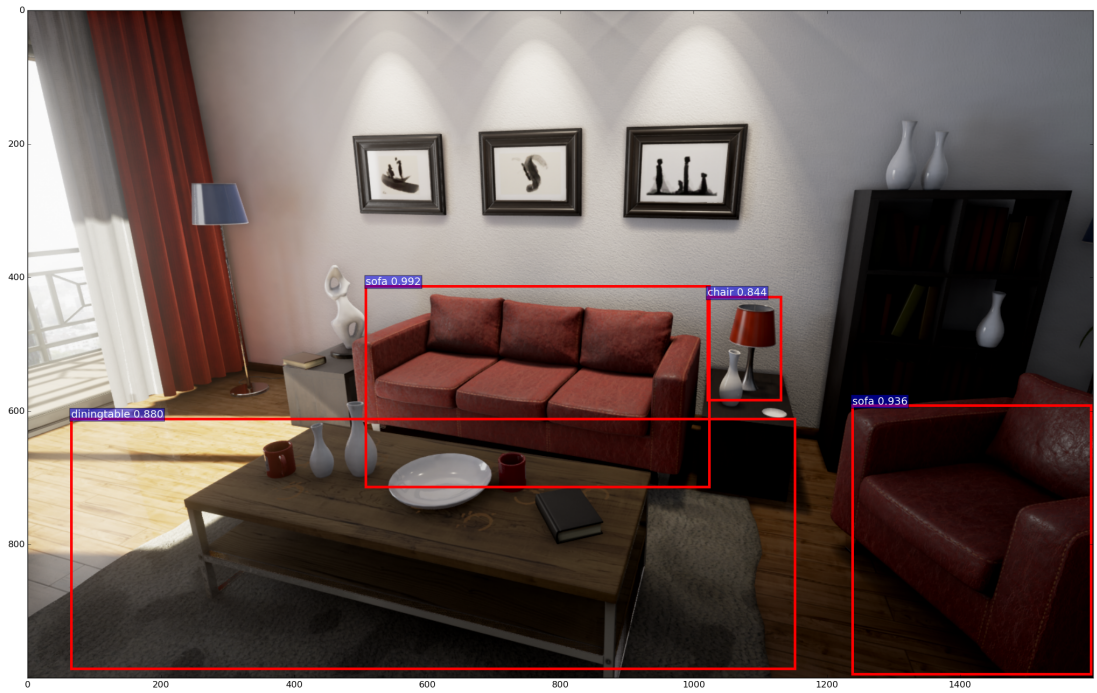
- In Mac: Run `[ProjectName].app`
- In Linux: Run `./[ProjectName]/Binaries/Linux/[ProjectName]`
- In Windows: Run `[ProjectName]/[ProjectName].exe`

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

12.1 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.

Plugin 0.3.10

Download Windows, Linux, Mac

- *Docker*

```
`nvidia-docker run --name rr --rm -p 9000:9000 --env="DISPLAY" --volume="/tmp/.X11-
↪unix:/tmp/.X11-unix:rw" qiuwch/rr:0.3.8`
```

12.2 ArchinteriorsVol2Scene1

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

Preview

Description ArchinteriorsVol2Scene1 is a scene of a 2 floors apartment.

Plugin 0.3.10

Download Windows, Linux, Mac

12.3 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.

Plugin 0.3.10

Download Windows, Linux, Mac

12.4 ArchinteriorsVol2Scene3

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

Preview

Description ArchinteriorsVol2Scene3 is a scene of an office.

Plugin 0.3.10

Download Windows, Linux, Mac,

12.5 UrbanCity

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

Preview

Description UrbanCity is a scene of a block of street.

Plugin 0.3.10

Download Windows, Linux, Mac

CHAPTER 13

Trouble Shooting

13.1 Issues and workarounds

Issues and workaround can be found in issue tracker. Please use the search function before posting your issue, your problem might have already been answered.

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.

13.2 Supported Unreal Engine Version

	4.12	4.14	4.16
Windows			
Linux			
Mac			

13.2.1 Client Support

Python support for python 2 and 3 are verified with tox.

13.2.2 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

13.3 Known issues and solutions

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

- python3 support. See issue #49, Thanks to @jskinn for the idea!
- 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
- Can not connect to the binary

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

13.4 Platform specific issues

13.4.1 Mac

When in mac, the server can not detect the socket disconnection. If the first time connection is successful and the second time is faile. Then please close and re-open it again.

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

13.4.2 Linux

- 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.

CHAPTER 14

CHANGELOG

14.1 Development branch

- **v0.3.10**
  - **Commands contributed in pull request #91, authored by @bennihepp**
    - Add `vget /camera/[id]/pose`, `vset /camera/[id]/pose`
    - Add `vget/vset /camera/[id]/horizontal_fieldofview`
    - Add `vget /camera/[id]/vis_depth npy` and `vget /camera/[id]/plane_depth npy`
    - Add `vset /object/[id]/show`, `vset /object/[id]/hide`
    - Add `vset /action/input/enable`, `vset /action/input/disable`
  - **Add more commands**
    - Add `vget /object/[id]/mobility`, `vget /object/[id]/location`, `vget /object/[id]/rotation`
    - Add `vget /camera/[id]/normal npy`
    - Add `vset /action/eyes_distance [eye_distance]`
    - Add `vset /action/game/pause`
  - Update the python client to support python3
  - Improve documentation
- **v0.3.9**
  - Fix a bug that prevents object mask generation, which is introduced in v0.3.7
  - Fix #53 that the painting of object does not work
  - Fix #49 python3 support, thanks to @jskinn and @befelix

- **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

14.2 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

14.3 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

CHAPTER 15

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 the 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 the UE4 official documentation.

15.1 Team Members

UnrealCV is developed and maintained by a group of students working on computer vision. The list of all contributors can be found here.

Here is a brief introduction of team members. If you want to be added to here, click the Edit on Github button on the top-right of this page.

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

CHAPTER 16

UE4 Resources

If you are interested in using synthetic data in your research projects, please see the synthetic-computer-vision repository for some related works.

UE4 is a good choice for research for various reasons:

- Rich 3D resources
- Open source
- Cross-platform
- Physically based material system

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

16.1 3D Resources (models, scenes)

- Unreal Marketplace - The official marketplace of Unreal Engine
  - The architectural visualization section
  - Unreal Engine feature examples
- UE4Arch - High quality architectural visualization using UE4.
- Evermotion
- TurboSquid - High quality 3D models used in the industry
- ShapeNet - Free 3D models released by the research community

16.2 C++ Development

- UE4 API

- UE4 Answerhub
- UE4 Github Repository (This is a private repository, you need to register here before you can access it).

CHAPTER 17

Development

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. For development, we need to

- Create a C++ game project
- Get the corresponding plugin
- Compile the C++ project
- Add a new command

17.1 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 of UE4 and UnrealCV. It serves as a development base and test platform for UnrealCV team.

We use git-lfs to manage large binary files. Please make sure you have installed git-lfs on your computer. Then you can get the playground project by

```
git lfs clone https://github.com/unrealcv/playground.git
```

17.2 Get the corresponding plugin

Now the folder `Plugins/UnrealCV` in the repository is empty. Please refer to *Install UnrealCV Plugin* to get the corresponding plugin and put it in the folder.

17.3 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

- Install Xcode.
- To generate Xcode Project, right click `playground.uproject` and choose `Service->Generate Xcode Project`.
- Open the `*.xcworkspace` file and build. The plugin code will be compiled together with the game project.

Useful resources for development, *UnrealCV architecture*

17.4 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 function that is missing, you can try to implement it yourself.

The benefits 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.

Note: You are supposed to edit your code in *playground->Plugins->UnrealCV* instead of *UE4->Plugins->UnrealCV*.

First we go through a very simple example which prints a message. Assume that we want to add a commamd `vget /object/helloworld` to print "Hello World!". We need to modify two files: `ObjectHandler.h` and `ObjectHandler.cpp`.

In `ObjectHandler.h`, we need to add a member function:

```
FExecStatus HelloWorld(const TArray<FString>& Args);
```

In `ObjectHandler.cpp`, we define this member function:

```
FExecStatus FObjectCommandHandler::HelloWorld(const TArray<FString>& Args)
{
        FString Msg;
        Msg += "Hello World!";
        return FExecStatus::OK(Msg);
}
```

Then we need to bind the command with the function:

```
void FObjectCommandHandler::RegisterCommands()
{
        ...

        Cmd = FDispatcherDelegate::CreateRaw(this, &
↪FObjectCommandHandler::HelloWorld);
        Help = "Print Hello World";
        CommandDispatcher->BindCommand(TEXT("vget /object/helloworld"), Cmd, Help);

        ...
}
```

After the modification, we can compile and use the new command.

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 function will be implemented in a CommandHandler. CommandDispatcher will use CommandHandler.

CHAPTER 18

Python API

Some useful utility functions for reading data

The unrealcv.automation module to help building the plugin, packaging model zoo binaries, etc.

CHAPTER 19

Contribute

We will be grateful if you help us mature UnrealCV.

19.1 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.

19.2 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.

19.3 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.