Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

MASTER'S THESIS

3D map estimation from a single RGB image

Author: Bc. Matěj Račinský

Thesis supervisor: doc. Ing. Karel Zimmermann, Ph.D. In Prague, May 2018

Author statement for thegraduate thesis:

I declare that the presented work was developed independently and that I have listed all the sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the presentation of university theses.

Prague, date _________

signature

Název práce: Odhad 3D mapy z jednoho RGB obrazu

Autor: Bc. Matěj Račinský

Katedra (ústav): Katedra kybernetiky

Vedoucí bakalářské práce: doc. Ing. Karel Zimmermann, Ph.D.

e-mail vedoucího: zimmerk@fel.cvut.cz

Abstrakt Tato práce se zabývá využitím virtuálních světů z počítačových her jakožto zdroje dat pro strojové učení, a odhadem voxelové mapy z jednoho RGB obrázku za pomoci hlubokého učení. Tato práce zahrnuje skripty pro napojení se na PC hru GTA V a sběr dat z ní pro tvorbu automaticky anotovaných datasetů, a implementaci hluboké neuronové sítě v TensorFlow.

Klíčová slova: Deep learning, Machine learning, GTA V, virtual world, depth estimation, voxelmap estimation, RAGE

Title: 3D map estimation from a single RGB image

Author: Bc. Matěj Račinský

Department: Department of Cybernetics

Supervisor: doc. Ing. Karel Zimmermann, Ph.D.

Supervisor's e-mail address: zimmerk@fel.cvut.cz

Abstract In this thesis we explore virtual worlds used as datasoutce for machine learning and voxelmap estimation from single RGB image with deep learning. This thesis describes principles and omplementation of hooking into GTA V and gathering data from it to create automatically annotated dataset, and implementation of deep neural network in TensorFlow.

Keywords: Deep learning, Machine learning, GTA V, virtual world, depth estimation, voxelmap estimation, RAGE

CONTENTS

1. Introduction **7**
1.1. Problems with machine learning datasets 7
1.2. Gaming industry to the rescue 8

2. Related work **9**

3. Transforming GTA V into the State of the Art simulator **10**
3.1. GTA V introduction 10
3.2. Automotive Simulators 11
3.3. GTA V modding ecosystem 11
3.4. Simulation environment and development stack 12
3.5. GTA V native API and provided data 12
3.6. Reverse-engineering the RAGE rendering pipeline 12

4. Conclusion **14**
4.1. Future work 14

Bibliography **14**

A. Contents of the enclosed CD **17**

CHAPTER

ONE

INTRODUCTION

This thesis aims to solve two problems. The first problem is the lengthy and slow process of manual dataset creation and annotation, and the second problem is voxelmap estimation from single RGB image. Due to increasing interest in synthetic datasets, this thesis aims to be the documentation for using GTA V as simulator for gathering synthetic datasets.

1.1. Problems with machine learning datasets

In recent years, both machine learning and deep learning has experienced great progress in many fields. Deep learning has outperformed many other machine learning approaches by using deep, high-capacity models trained on large datasets. Especially in the field of computer vision, neural networks achieve state of the art results in most of the tasks. Many tasks in computer vision are the first where deep neural networks achieve state of the art results before being used in other fields, and in this field deeper and deeper architectures are being proposed earlier than in other fields.

With larger amount of parameters, the need for large datasets is growing, with currect datasets unable to cover the need for annotated data.

Data has proven to be limiting factor in many computer vision tasks. The main problem is that manual data annotation is exhausting, time-consuming and costly. That is even more significant for pixel-wise annotation which is crucial for tasks of semantic segmentation. Pixel-wise annotated datasets are orders of magnitude smaller than image classification datasets. This is sometimes called “curse of dataset annotation”[23], because more detailed semantic labeling leads to smaller size of dataset.

Many novel neural network architectures are being proposed every year because of increasing computing power. With growing capacity and number of parameters in these new models, there is need for bigger and bigger datasets for training. Data has proven to be limiting factor for many evaluations.

Automatic data gathering and automatic data annotation could potentially solve these problems of lack of datasets in many computer vision tasks.

1.2. Gaming industry to the rescue

In last decades, gaming industry has grown hugely and expanded from closed and specific community into public society and became mainsteam industy.

Gaming industry became big driving force in many fields, and indirectly influenced even machine learning.

The mainsteam model of gaming is on personal computers, where each player has his own gaming PC, along with console gaming. Thanks to ever-growing number of players, lots of money got into industry and the growing demand for better graphics in games caused big improvements in both software-computer graphics and hardware-graphics cards. With lots of money being invested by players in their PCs, GPU manufacturers were able to deliver more powerful GPUs every year and we can see exponential growth of GPU computational power[21].

Big companies in gaming industry have enough resources to develop the state of the art real-time computer graphics and so they deliver AAA games with graphics very near to reality.

Recent papers[16, 20] show that we can use screenshots from PC games to obtain automatically or semi-automatically annotated dataset which improve learning, allow us to outperform same models trained only on real data and achieve state of the art results.

CHAPTER

TWO

RELATED WORK

[20] used GTA V to obtain screenshots and performed semi-automated pixel-wise semantic segmentation. Although the process was not fully automatic, the annotation speed per image was drastically increased, compared to Cityscapes [8], being 771 times faster than fine per-image annotation of Cityscapes and 514 times faster than per-image annotation of CamVid[7]. They extracted 24 966 images from game GTA V, which is roughly two orders of magnitude larger than CanVid and three orders of magnitude larger than semantic annotations for KITTI dataset. Then they trained the prediction module of Yu and Koltun[24] and by using on $\frac{1}{3}$of the CamVid training set (which is) and all 24 966 GTA V screenshots, they outperformed same model trained on whole CamVid training dataset.

For images extraction, they use RenderDoc[17], stand-alone graphics debugger. It intercepts the communication between the game and the GPU and allows to gather screenshots. It's advantage is that it can be used for multiple different games.

[16] use GTA V screenshots, depth and stencil buffer to produce car images and automatically calculate their bounding boxes.

On these generated data, they trained Faster R-CNN[19] only of screenshots from the GTA V game, using up to 200 000 screenshots, which is one order of magnitude bigger than Cityscapes dataset. Using only screenshots for training, they outperformed same architecture trained on Cityscapes, evaluating on KITTI dataset. They used GTA V mod3.3 to hook into GPU calls and to gather screenshots from here.

CHAPTER
THREE

TRANSFORMING GTA V INTO THE STATE OF THE ART SIMULATOR

In this thesis, Grand Theft Auto V (GTA V) game is used for creating synthetic, nearly photo-realistic dataset.

3.1. GTA V introduction

GTA V is action-adventure open-world video game developed by Rockstar North and published by Rockstar Games. The game was released on 17.9.2013 for Playstation 3 and Xbox 360[12] , in 18.11.2014 for PS4 and Xbox One and in 14.4.2015 it was released on PC, Windows[13].

The game is based on proprietary game engine, called RAGE (Rockstar Advanced Game Engine)[18], which is used as base for most of Rockstar Games products.

Until the release on Microsoft, Windows, it has been in development for 5 years with approximate 1000-person team[11]. The world of GTA V was modelled on Los Angeles[15] and other areas of Southen California, with road networks respecting design od Los Angeles map.

As could be expected from AAA game like GTA V, motion capture was used to character'sboth body and facial movements.

There are several reasons why GTA V is better for dataset creationg than other games. To use a game for dataset creation, we have multiple requirements. The graphics of the game must be near photorealistic, because obviously, otherwite it would be useless to be used for computer vision tasks. This disqualifies most of games, mostly indie games, and leaves us only with AAA games produced by big companies.

Also, there should be good-enough way to interact with the game programatically. Usually we want to setup at least part of the environment before gathering data. This part heavily depends on community around the particular game.

In gaming subculture, there are communities where people specialize in reverse-engineering of games and development of modifications to these games. These people are called modders or mod developers, and these unofficial modifications and extension of games are called mods. For few games, developers welcome this kind of activity and sometimes they even release tools to ease the game modding.

In most cases, the game developers simply don't care and in few cases, they actively fight against the reverse-engineering and modding.

The GTA V is second case, where Rockstar Games does not actively try to prevent the reverse-engineering, but they don't release any tools to ease it, either. This results in cyclic process of Rockstar Games releasing new version of game, including backward compatibility (BC) breaks, and community reverse engineering the new version and adjusting their mods to work with the new version.

The modding community around the GTA V is based mostly on community around GTA IV, which was previous big game produced by Rockstar Games. So many tools are just GTA IV based and only modified to work with GTA V. Luckily, the community is large and productive, so we have many mods and many function in GTA V reverse-engineered and thus prepared to interact with programatically.

3.2. Automotive Simulators

Currently, there are some opensorce simulation platforms for automotive industry which could be theoretically used for creating synthetic datasets. But compared to AAA games like GTA V, they have much less resources and much less customers to finance the development. In result, simulators have worse graphics than AAA games and NPC (non playable characters) don't have as sophisticated behaviour. In GTA V, drivers mostly follow traffic regulations, traffic lights and traffic lanes, which leafs to very realistic environment better than simulators can provide.

3.3. GTA V modding ecosystem

Although the modding community is quite big, as it is in lots of opensource communities, essential part of community depends on one person. Here, it is Alexader Blade. In his free time, he reverse-engineered big part of GTA V and developed ScriptHookV[6], library enabling to perform native calls into GTA V in C++ and develop GTA V mods in C++. Currently, more people in community participates in reverse-engineering and they share their knowledge in GTA forum thread[5].

List of all reverse-engineered native functions is kept in following list [4]. Assumingly, GTA V contains ~5200 natives. There is no original native name list of functions in GTA V, name hashes are used instead. During reverse-engineering ande game decompilation, ~2600 native names were discovered using bruteforce and manual checking afterwards. For these functions, number of parameters and returns of these calls are also known. In the native functions list, for big part of functions we know their name, signature and how do they affect the game. The rest remains to be discovered yet.

When new version of game is released, in few days to weeks, new version of ScriptHookV is released, fixing BC breaks.

Other heavily used mod in community is ScriptHookDotNet2, which is built atop of ScriptHookV and creates bridge between C# language and ScriptHookV, effectively allowing to write GTA V mods in C#. It is availabe as open-source [9]. Along with creating bridge between C# and GTA V, it wraps most used native calls into classes, leveraging object-oriented paradigm for mod development using getters and setters as proxies for native calls.

Unlike most of mods, these two mods act more as a framework for mod development.

Next notable mod is NativeUI[14]. It renders windows atop of GTA V GUI and allows us to define custom control panels for manipulating custom functionality in other mods.

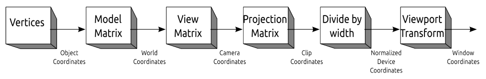

Figure 3.1.: Rendering pipeline

Since GTA V is a game, it requires human interaction. For simulator-like behavior we would want the car to drive autonomously to crawl data without human interaction. This can be done using VAutodrive[10]. This allows us to use NPC automatic behaviour patterns for main player, letting the player randomly wander the world, even in car, without need of human assistance during crawling. Unfortunately, this package is not open-source.

Generally, the community is not united in their view on opensource. Some mods are available open-source on github. Other mods are being distributed only as compiled binaries[1]. Lots of modders develop mostly by trial and error, and no comprehensive documentation for mod development is available, unfortunately. There are some tutorials [22], but they are far from complete and provide only basic knowledge, leavin reader without deeper understanding of underlying principles.

Modders mostly meet online on few GTA forums, where they exchange knowledge[2, 3]. Github or StackOverflow, which are biggest sources of information for usual software development, are not used much in GTA modding community. Due to this fact, these forums, along with source code of open-source mods comprise knowledge-base of mod development.

3.4. Simulation environment and development stack

In this thesis, I use mod based on [16] but enhanced to gain more control of the game. Also, the mod in

In later text, I'll refer to some GTA V native functions or data structures which are output of GTA V native functions. To be consistent and to help understanding, I will use function names from native function list [4].

3.5. GTA V native API and provided data

3.6. Reverse-engineering the RAGE rendering pipeline

As mentioned above3.1, GTA V uses proprietary game engine, Rockstar Advanced Game Engine (RAGE). The basic premise of rendering pipeline is same as in well known graphics engines like OpenGL. The pipeline is shown in figure 3.1. Following section will be discussing mostly computer graphics related problems. Due to some terminology inconsistency between computer graphics and computer vision, all terms used here will be compuer graphics related. Probably most confusion here could be caused by projection matrix. In computer vision, projection matrix is projection from 3D to 2D, the matrix reduces dimension. In computer graphics, all coordinates are kept in 4D, in homogeneous coordinates as long as possible. Here the projection matrix prepresents projection from frustum seen by eye into cuboid space of Normalized Device Coordinates.

In is part, I will describe some transformations between individual RAGE coordinate systems. Some points here will have part of name in lower index. The name of coordinate system will be denoted in upper index. In RAGE there are 6 coordinate systems.

Name	Abbreviation	Example point x
Object Coordinates	O	x^O
World Coordinates	W	x^W
Camera Coordinates	C	x^C
Clip Coordinates	L	x^L
Normalized Device Coordinates	NDC	x^{NDC}
Windows Coordinates	P	x^P

Most of points we handle in GTA are already in world coordinates. But some points, like GAMEPLAY::GET_MODEL_DIMENSIONS= $\begin{pmatrix} x^O_{max} & y^O_{max} & z^O_{max}\end{pmatrix} \begin{pmatrix} x^O_{min} & y^O_{min} & z^O_{min}\end{pmatrix}$ output, are vertex data, in object coordinates.

To get world coordinates of model dimensions, we use traditional rigid body transformation based on ENTITY::GET_ENTITY_ROTATION= $\begin{pmatrix}\alpha & \beta & \gamma\end{pmatrix}$ Euler angles and ENTITY::GET_ENTITY_COORDS= $\begin{pmatrix} x^W & y^W & z^W\end{pmatrix}$.

Because all coordinates will be homogeneous coordinates, the above-mentioned model dimensions vectors will be transformed to following form $\begin{pmatrix} x^O_{max} & y^O_{max} & z^O_{max} & 1\end{pmatrix} \begin{pmatrix} x^O_{min} & y^O_{min} & z^O_{min} & 1\end{pmatrix}$.

So the transitions matrix is

$$W = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\beta)\cos(\gamma) & -\cos(\beta)\sin(\gamma) & \sin(\beta) & x \\ \sin(\alpha)\sin(\beta)\cos(\gamma) + \cos(\alpha)\sin(\gamma) & \cos(\alpha)\cos(\gamma) - \sin(\alpha)\sin(\beta)\sin(\gamma) & -\sin(\alpha)\cos(\beta) & y \\ \sin(\alpha)\sin(\gamma) - \cos(\alpha)\sin(\beta)\cos(\gamma) & \cos(\alpha)\sin(\beta)\sin(\gamma) + \sin(\alpha)\cos(\gamma) & \cos(\alpha)\cos(\beta) & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and whole transformation is, as expected

$$W \begin{bmatrix} x^O_{max} & x^O_{min} \\ y^O_{max} & y^O_{min} \\ z^O_{max} & z^O_{min} \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} x^W_{max} & x^W_{min} \\ y^W_{max} & y^W_{min} \\ z^W_{max} & z^W_{min} \\ w^W_{max} & w^W_{min} \end{bmatrix}$$

The transformation from world coordinate

CHAPTER

FOUR

CONCLUSION

4.1. Future work

BIBLIOGRAPHY

[1] Gta 5 mods, . URL `https://www.gta5-mods.com/`.

[2] Gta5 mods forum, . URL `https://forums.gta5-mods.com/category/5/general-modding-discussion`.

[3] Gta forums, . URL `http://gtaforums.com/forum/370-gta-v/`.

[4] Alexander Blade. Native db. URL `http://www.dev-c.com/nativedb/`.

[5] Alexander Blade. Native db script/native documentation and research, June 2014. URL `http://gtaforums.com/topic/717612-v-scriptnative-documentation-and-research/`.

[6] Alexander Blade. Script hook v, April 2015. URL `http://gtaforums.com/topic/788343-script-hook-v/`.

[7] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recogn. Lett.*, 30(2):88–97, January 2009. ISSN 0167-8655. doi: 10.1016/j.patrec.2008.04.005. URL `http://dx.doi.org/10.1016/j.patrec.2008.04.005`.

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016. URL `http://arxiv.org/abs/1604.01685`.

[9] Crosire. Community script hook v .net, April 2015. URL `https://github.com/crosire/scripthookvdotnet`.

[10] Cyron43. Vautodrive, July 2015. URL `https://www.gta5-mods.com/scripts/vautodrive`.

[11] MICHAEL FRENCH. Inside rockstar north - part 2: The studio, October 2013. URL `https://www.mcvuk.com/development/inside-rockstar-north-part-2-the-studio`.

[12] Rockstar Games. Grand theft auto v is coming 9.17.2013, January 2013. URL `https://www.rockstargames.com/newswire/article/48591/grand-theft-auto-v-is-coming-9172013.html`.

[13] Rockstar Games. Grand theft auto v release dates and exclusive content details for playstation 4, xbox one and pc, September 2014. URL `https://www.rockstargames.com/newswire/article/52308/grand-theft-auto-v-release-dates-and-exclusive-content`.

[14] Guad. Native ui, July 2015. URL `https://github.com/Guad/NativeUI`.

[15] Matt Hill. Grand theft auto v: meet dan houser, architect of a gaming phenomenon, September 2013. URL `https://www.theguardian.com/technology/2013/sep/07/grand-theft-auto-dan-houser`.

[16] M. Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, and Ram Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? In *IEEE International Conference on Robotics and Automation*, pages 1–8, 2017.

[17] Baldur Karlsson. Renderdoc. URL `https://renderdoc.org/`.

[18] Thomas Morgan. Face-off: Grand theft auto 5, September 2013. URL `https://www.eurogamer.net/articles/digitalfoundry-grand-theft-auto-5-face-off`.

[19] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL `http://arxiv.org/abs/1506.01497`.

[20] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *European Conference on Computer Vision (ECCV)*, volume 9906 of *LNCS*, pages 102–118. Springer International Publishing, 2016.

[21] Balázs Tukora and Tibor Szalay. High performance computing on graphics processing units. *Pollack Periodica*, 3:27–34, 08 2008. URL `https://www.researchgate.net/publication/242065578_High_performance_computing_on_graphics_processing_units`.

[22] V4D3R. Gta-mod-dev-tutorial, August 2016. URL `https://forums.gta5-mods.com/topic/1451/tutorial-grand-theft-auto-v-modding-a-few-things-you-should-know/2`.

[23] Jun Xie, Martin Kiefel, Ming-Ting Sun, and Andreas Geiger. Semantic instance annotation of street scenes by 3d to 2d label transfer. *CoRR*, abs/1511.03240, 2015. URL `http://arxiv.org/abs/1511.03240`.

[24] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122, 2015. URL `http://arxiv.org/abs/1511.07122`.

APPENDIX
A

CONTENTS OF THE ENCLOSED CD

appendix content